Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título:

Módulo para la administración de los servidores web en HMAST.

Autora:

Nurisel Palma Pérez

Tutores:

Ing. Amaury Viera Hernández

Ing. Susana Sánchez Ortiz

La Habana, Cuba, junio 2013

“Año 55 de la Revolución”


¿Para qué sino para poner
paz entre los hombres, han de ser
los adelantos de la ciencia?
José Martí

Declaración de autoría

Declaro ser la única autora de este trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2013.

_____

Nurisel Palma Pérez

_____ _____

Ing. Amaury Viera Hernández Ing. Susana Sánchez Ortiz

Dedicatoria

A mis abuelos Caridad y Pedro, que aunque no están conmigo físicamente siempre seguirán presentes en mi mente y mi corazón.

A mis abuelos Paulina y Arcadio por todo el cariño que me han dado.

A mis padres, Olga y Jorge, por ser el sostén, la fuerza y la guía que me permitieron llegar hasta donde estoy hoy.

A mi hermano Jose, que ya es todo un hombre y ha estado ahí en cada momento que me ha hecho falta.

A Yoandry, mi primer y único amor.

Agradecimientos

A mis padres, mi hermano y mi novio por ser las personas más importantes en mi vida y luchar junto a mí tantos años.

A toda mi familia, abuelos, tíos y primos por quererme tanto.

A mis amigos de siempre Lisandra y Ariel.

Al inolvidable "pikete" Yucel, Merlyn, Daniel, Dayli, Yadiel y Yasiel que en los momentos más difíciles de mi vida me dieron apoyo incondicional y las fueras para seguir adelante.

A todos mis amigos, Iliana, Alexander, María Leisy, José Carlos, en fin a todos.

A mis tutores, especialmente a Susana que siempre estuvo a mi lado

A Jailen por guiarme y orientarme todo el tiempo hacia el camino correcto.

A Yasiel, Yadiel y Alexander que en estos cinco años y durante el desarrollo de la tesis estuvieron ahí para aclararme cada duda y nunca decir que no.

A los integrantes de HMAST, que formamos un verdadero equipo y unidos supimos salir adelante.

A todos los que contribuyeron de una forma u otra, los profesores de SIMAYS, mis compañeros de guagua, mis vecinos, a todos gracias.

Resumen

En la Universidad de las Ciencias Informáticas al frente del proceso de migración a software libre y plataformas de código abierto se encuentra el Departamento SIMAYS, perteneciente al Centro de Software Libre (CESOL). En este departamento, como apoyo a la migración se están desarrollando aplicaciones como la Herramienta de Migración y Administración de Servicios Telemáticos (HMAST). Entre los servicios telemáticos se enmarcan los servicios de solicitud y respuesta que hacen uso del protocolo HTTP, que son ofrecidos por los servidores web. Un servidor web puede contener un número considerablemente grande de hosts virtuales para publicar el contenido, por lo que en algunos casos la administración de estos de forma manual puede convertirse en una tarea engorrosa para los administradores de redes. HMAST posee funcionalidades para administrar los usuarios, las tareas programadas y los servicios DHCP y los que implementan el protocolo SSH. Sin embargo, en esta herramienta también se hace necesario la administración de los servidores web. El presente trabajo de diploma tiene el propósito de desarrollar un módulo para HMAST que administre los servidores web de forma que se asegure la adaptabilidad a las diferentes empresas e instituciones cubanas. Para la concepción del mismo se realiza el estudio de diferentes servidores web, específicamente los de código abierto, determinándose Apache 2 como servidor a administrar. También se documentan las diferentes tecnologías a usar y la metodología de desarrollo de software como guía del proceso. El módulo permite la administración del servidor seleccionado basándose en sus principales secciones de configuración.

Palabras claves: Administración, Apache 2, Secciones de configuración, Servidores

Índice

Introducción 1

Capítulo 1: Los servidores web 5

1.1 Servicios telemáticos 5

1.2 Servidores web 6

1.3 Servidores web más usados 7

1.4 Apache 2 8

1.5 Nginx 10

1.6 Necesidades en las diferentes empresas e instituciones cubanas 11

1.7 Selección del servidor 12

1.8 Administración de Apache 2 14

1.9 Descripción de la herramienta HMAST 16

1.9.1 Arquitectura 16

1.9.2 Funcionalidades que ofrece 17

1.9.3 Consideraciones para implementar un módulo para HMAST 17

1.10 Herramientas y tecnologías a emplear 18

1.10.1 Framework 18

1.10.2 Lenguaje de programación. 19

1.10.3 Entorno de Desarrollo Integrado (IDE) 19

1.10.4 Herramientas CASE. 20

1.10.5 Sistema de Control de Versiones 20

1.10.6 Herramienta de diseño gráfico Pencil 21

1.10.7 Augeas 21

1.11 Metodología de desarrollo de software 21

1.11.1 Metodología SXP 22

Conclusiones parciales 22

Capítulo 2: Módulo para la administración de los servidores web. 24

2.1 Propuesta del módulo 24

2.1.1 Funcionamiento de Apache 2 24

2.2 Artefactos generados en el desarrollo 26

2.2.1 Lista de Reserva del Producto (LRP) 26

2.2.2 Plan de Liberación 28

2.2.3 Historias de Usuario 28

2.3 Arquitectura del módulo 37

2.4 Diagrama de paquetes 38

2.5 Patrones de diseño empleados 41
2.5.1 Patrones GRASP 41
Conclusiones parciales 42
Capítulo 3: Implementación y pruebas. 43
3.1 Tareas de Ingeniería 43
Tareas de Ingeniería para Gestionar hosts virtuales asignados a Apache 2. 43
Tarea de Ingeniería para Manejar estados de Apache2. 45
Tarea de Ingeniería para Modificar puertos para las conexiones en Apache 2. 45
Tarea de Ingeniería para Especificar módulos a usar en Apache 2. 46
Tarea de Ingeniería para Especificar parámetros en el servidor principal de Apache 2. 46
Tarea de Ingeniería para Especificar parámetros en el MPM prefork de Apache 2. 46
Tarea de Ingeniería para Especificar parámetros para el manejo de conexiones. 47
3.2 Definición de las pruebas a aplicar 47
3.3 Realización de pruebas al código 48
3.4 Pruebas realizadas por el cliente 58
Conclusiones parciales 59
Conclusiones 61
Recomendaciones 62
Referencias Bibliográficas 63
Bibliografía General 66
Anexos 68
Glosario de términos 87

Introducción

Cuba ha identificado desde muy temprano la necesidad e importancia de lograr una cultura digital como una de las características para alcanzar la eficiencia y competitividad en sus diferentes esferas y procesos. Para darle cumplimiento a dicho objetivo, se ha propuesto el dominio e introducción en la práctica social de las Tecnologías de la Información y las Comunicaciones (TIC). Es a partir de 1996 que se dan los primeros pasos en cuanto a la informatización de la sociedad, definida como el proceso de utilización ordenada y masiva de las TIC para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad [1].

Este proceso ha ido encaminado y enfocado a los pilares fundamentales del proceso revolucionario: la educación, la salud y la cultura. No se puede lograr una utilización masiva de las TIC sin haber elevado la calidad de la educación, instrumentado un proceso de educación continua y ampliado la cultura general de la población sobre estas tecnologías. Como expresara el compañero Fidel Castro Ruz el 15 de enero de 1960, en un discurso ante la Sociedad Espeleológica: *"El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento..." [2].*

Como meta en la estrategia de informatización se traza la necesidad de globalización del conocimiento, trabajando en cuanto al perfeccionamiento de la enseñanza general y la universalización de la universidad. Desde marzo del 2002 se introdujo en todas las escuelas del sistema educacional la enseñanza de la computación. Se extendió el uso de las TIC como apoyo a los programas de clases en los diferentes centros de enseñanza primaria, secundaria, pre-universitaria, tecnológica y universitaria del país, con el objetivo de elevar la calidad de la educación cubana.

A mediados del 2002, como una idea del compañero Fidel dentro del marco de la Batalla de Ideas, se crea la Universidad de las Ciencias Informáticas (UCI). Actualmente cuenta con catorce centros de desarrollo de software especializados en diferentes perfiles y líneas de investigación, fundamentalmente relacionados con la educación, la salud, la industria y la política del país de migrar a plataformas de código abierto. Esta última es una de las directrices rectoras en Cuba en estos momentos, pues tiene relevancia referente a aspectos políticos, económicos y tecnológicos.

El uso del software libre representa la no utilización de productos informáticos que demanden la autorización de sus propietarios para su explotación. Es desarrollado de forma colectiva y cooperativa, con

el objetivo de beneficiar a toda la comunidad. Su utilización no implica gastos adicionales por concepto de cambio de plataforma de software, por cuanto es operable en el mismo soporte de hardware con que cuenta el país. Permite su adaptación a los contextos de aplicación, al contar con su código fuente, lo cual garantiza un mayor porciento de efectividad, además de la corrección de sus errores de programación y la obtención de las actualizaciones y las nuevas versiones [3]. Por tales razones existe una inminente necesidad de la realización de un proceso continuo y organizado para migrar a plataformas de software libre.

En la UCI al frente del proceso de migración se encuentra el Departamento SIMAYS, perteneciente al Centro de Software Libre (CESOL). El mismo es especializado en ofrecer servicios integrales en migración, asesoría y soporte a tecnologías de software libre y código abierto. Tiene como misión proveer a los clientes de un servicio de migración eficiente y en el menor tiempo posible, con el objetivo de optimizar la infraestructura tecnológica con el mejor retorno de inversión. Su investigación va dirigida a tres líneas fundamentales: el desarrollo de estrategias, guías y planes de migración; las herramientas de soporte al proceso de migración y la migración de los servicios telemáticos y aplicaciones de escritorio. Como apoyo a la migración se están desarrollando aplicaciones como son la Plataforma de Migración y la Herramienta de Migración y Administración de Servicios Telemáticos (HMAST).

Entre los servicios telemáticos se enmarcan los servicios de solicitud y respuesta que hacen uso del Protocolo de Transferencia de Hipertexto (*HTTP*, *Hypertext Transfer Protocol*), el cual pertenece a la capa de aplicación del modelo de referencia de Interconexión de Sistemas Abiertos (OSI, *Open System Interconnection*). Estos servicios son ofrecidos por los servidores web, los cuales tienen asignada la responsabilidad de publicar contenido en forma de hipertextos, páginas web o páginas HTML (*HyperText Markup Language*), que son almacenados en hosts virtuales y accedidos por los usuarios. En un mismo servidor puede existir un número considerablemente grande de hosts virtuales, por lo que en algunos casos la administración de estos de forma manual puede convertirse en una tarea engorrosa para los administradores de redes.

El desarrollo de HMAST tiene el propósito de disponer de una aplicación que se emplee en las diferentes empresas e instituciones cubanas que hagan uso, en la mayoría de las PC servidoras[1], de sistemas operativos privativos. Por tanto, esta herramienta permitirá administrar y migrar los servicios telemáticos a plataformas libres. Está compuesta por un sistema base que tiene las funcionalidades necesarias para administrar los usuarios, las tareas programadas y los servicios, entre otros. Esta herramienta se

1 Computadora con un software servidor instalado.

encuentra en su primera versión, y cuenta con los módulos para administrar los servicios DHCP y los que implementan el protocolo SSH. Sin embargo, en HMAST también se hace necesario la administración de los servidores web.

Dada la problemática existente, se define como **problema científico** de la investigación el ¿cómo garantizar la administración de los servidores web en la herramienta HMAST de forma que se asegure la adaptabilidad a las diferentes empresas e instituciones cubanas?

Con vista a la solución del problema planteado el **objeto de estudio** del trabajo son los servidores web, enmarcándose el **campo de acción** a los servidores web de código abierto.

El **objetivo general** está centrado en desarrollar un módulo para HMAST que administre los servidores web de forma que se asegure la adaptabilidad a las diferentes empresas e instituciones cubanas. Derivándose los siguientes **objetivos específicos**:

- Caracterizar los servidores web a partir de la bibliografía especializada en los mismos.
- Analizar y diseñar un módulo para la herramienta HMAST que permita la administración de los servidores web.
- Implementar la solución propuesta.
- Realizar las pruebas correspondientes al módulo implementado.

Como materialización y contribución al cumplimiento de los mismos, las **tareas de investigación** planificadas son:

- Caracterización de los servidores web a partir de la bibliografía especializada en los mismos.
- Análisis y diseño de un módulo para la administración de los servidores web en HMAST.
- Implementación del módulo definido.
- Realización de las pruebas correspondientes al módulo implementado.

La **idea a defender** de la investigación es que el desarrollo de un módulo para la administración de los servidores web en la herramienta HMAST, permitirá realizar la administración de los mismos garantizando la adaptabilidad a las diferentes empresas e instituciones cubanas.

Para el desarrollo de la investigación se utilizó el método científico **Analítico-Sintético** pues permitió el estudio de diferentes fuentes bibliográficas para extraer los elementos más importantes que se relacionan

con los servidores web de forma general y los de código abierto. Se realizaron además resúmenes y valoraciones de conceptos relevantes relacionados con este tema y con la administración de los servidores web.

Se empleó además el método teórico de la **Modelación**, debido a que se crearon abstracciones con el objetivo de explicar la realidad. Se elaboraron representaciones explícitas del entendimiento acerca de la estructura de almacenamiento lógico de un host virtual y la forma en que se selecciona el host virtual que debe atender una petición en Apache 2. En el marco del diseño del módulo propuesto se modeló el diagrama de paquetes.

La investigación está desarrollada en 3 capítulos, cuya estructura se describe a continuación:

En el primer capítulo "**Los servidores web**" se realiza el estudio del estado del arte donde se hace referencia a los aspectos relacionados con los servidores web, como conceptos, características, entre otros. Además se abordan ejemplos de estos servidores enfocándose en los más usados según las estadísticas. Se detallan las características de HMAST y se realiza la selección del servidor, las tecnologías y la metodología de desarrollo de software orientada al desarrollo del sistema propuesto.

En el segundo capítulo "**Módulo para la administración de los servidores web**" se hace referencia a la solución que se propone, planteándose cómo se va a desarrollar esta; se definen cuáles serán las funcionalidades a implementar. Además, se describe el proceso ágil basado en las Historias de Usuario correspondientes a las nuevas funcionalidades, prototipos de interfaz de usuario, entre otros artefactos de interés. Se define la arquitectura del sistema y los patrones de diseño a emplear.

En el tercer capítulo "**Implementación y pruebas**" se realiza la implementación del módulo propuesto tomando como guía las Tareas de Ingeniería generadas por la metodología SXP. Además se describen, diseñan, realizan y controlan los casos de pruebas aplicados al sistema.

Capítulo 1: Los servidores web

En el marco del estudio del estado del arte, en el presente capítulo se hará referencia a conceptos relacionados con los servidores web y específicamente los de código abierto. Como resultado del estudio se seleccionará el servidor web más indicado a emplear para el desarrollo del módulo.

1.1 Servicios telemáticos

La comunicación ha tenido gran relevancia desde el surgimiento de la humanidad por constituir un elemento vital en la evolución y desarrollo de la misma. Con el transcurso del tiempo y la evolución tecnológica, la comunicación a distancia empezó a incorporar elementos como fueron las computadoras y se estableció la interconexión entre equipos informáticos de todo tipo.

La ciencia que trata la conectividad y comunicación a distancia entre procesos (conjunto de instrucciones que se ejecutan en una computadora) se denomina teleinformática o también conocida como telemática. Puede definirse como el conjunto de máquinas, técnicas y métodos relacionados entre sí que permiten el proceso de datos a distancia y que participan en la convergencia entre las telecomunicaciones y la informática. En la actualidad se encuentra suficientemente implantada y desarrollada para dar servicio a la mayor parte de las necesidades existentes [4].

Los servicios telemáticos son aquellos que, utilizando como soporte servicios básicos, permiten el intercambio de información entre terminales con protocolos establecidos para sistemas de interconexión abiertos [5].

Cuando a través de una red de comunicaciones se pretende comunicar un sistema informático con otro, se hace necesaria la existencia de un conjunto de elementos físicos y lógicos que permitan el desarrollo de dicho proceso. La comunicación no es solo la conexión entre los equipos sino todo el conjunto de elementos que permiten el entendimiento entre ambos, independientemente de sus características individuales.

Uno de estos elementos son los protocolos. Un protocolo es un conjunto de normas que permiten el intercambio de información entre dos dispositivos o elementos de un mismo nivel. Además, articulan métodos y procesos para la detección y corrección de errores [6].

La comunicación entre aplicaciones que se ejecutan en distintos sistemas es lo que realmente se pretende

en una red teleinformática. El nivel superior de una arquitectura estructurada proporciona los servicios necesarios para la comunicación entre aplicaciones y se denomina Nivel de Aplicación. Entre los protocolos pertenecientes a este nivel se encuentra el Protocolo de Transferencia de Hipertexto (HTTP). HTTP es el protocolo destinado al control de la transferencia de datos en la World Wide Web, proporcionando un vehículo de entrega para gráficos, video, imágenes, hipertexto u otros datos en la web.

Enmarcado en los servicios telemáticos existentes se encuentran los servicios de solicitud y respuesta mediante el protocolo HTTP, que son ofrecidos por los servidores web.

1.2 Servidores web

Un servidor web es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente [7]. Mientras que comúnmente se utiliza la palabra servidor para hacer referencia a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones antes descritas [8].

Un servidor web se mantiene a la espera de peticiones HTTP que realiza el usuario mediante el uso de un cliente HTTP, lo que se suele conocer como un navegador web. Se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado páginas web, hipertextos o páginas HTML, que son interpretados por el cliente y mostrados al usuario en el formato correspondiente. Proporciona los recursos mediante el protocolo HTTP o el protocolo HTTPS, la versión segura, cifrada y autenticada de HTTP.

Existen varios tipos de servidores web, entre los cuales se encuentran los siguientes:

Servidores basados en procesos

Se basan en la obtención de paralelismo mediante la duplicación del proceso de ejecución. Es el predecesor de los demás diseños. El más simple de este tipo de diseños es en el que el proceso principal con la llegada de una nueva conexión se duplica creando una copia exacta que atenderá la misma.

Servidores basados en hilos

Le son aplicables los conceptos básicos respecto al funcionamiento de un servidor basado en procesos, heredando muchas de sus características, entre ellas la de la simplicidad en su diseño e implementación. Las principales diferencias de los dos modelos residen en el propio concepto de hilo. Tienen la ventaja que

la creación de un hilo es menos costosa que la de un proceso. Varios hilos de un mismo proceso comparten el mismo espacio de memoria, propiciando que puedan compartir datos entre ellos, sin embargo, esto implica un riesgo de seguridad.

Servidores basados en sockets no bloqueantes o dirigidos por eventos

Basan su funcionamiento en la utilización de lecturas y escrituras asíncronas sobre sockets. Utilizan una llamada al sistema que examine el estado de los sockets con los que trabaja, donde cada sistema operativo implementa una o más funciones de examen de sockets. Estas funciones tienen como objetivo inspeccionar el estado de un grupo de sockets asociados a cada una de las conexiones. Este tipo de servidores a pesar de caracterizarse por su velocidad, tienen la desventaja que la concurrencia es simulada, existiendo un solo proceso y un solo hilo, desde el cual se atienden todas las conexiones.

1.3 Servidores web más usados

Existen numerosos servidores web dentro de los que se encuentran Microsoft-IIS, Apache, Nginx, LiteSpeed, Google Servers, Tomcat, Lighttpd, IBM Servers, Yahoo Traffic Server, Jetty, AOLserver, Roxen y Caudium. Algunos son más usados que otros debido a que poseen características distintivas que marcan la diferencia en cuanto a su selección.

La información sobre el uso de servidores web es proporcionada por algunos sitios especializados en ofrecer estadísticas de diferentes tecnologías en la web. Ejemplo de ello es el sitio de la compañía inglesa Netcraft que realiza mensualmente mediciones de uso. Según el reporte del mes de febrero del presente año, los tres servidores web más usados son Apache, Microsoft IIS y Nginx, como se puede apreciar en el Anexo 1 [9].

Por otra parte, Web Technology Surveys (W3Techs) también proporciona estadísticas diarias del porcentaje de sitios web que hacen uso de determinado servidor web. En el Anexo 2 se pueden observar las del día 25 de febrero del presente año, donde Apache, Microsoft IIS y Nginx son los tres servidores web más usados [10].

Una vez analizadas las estadísticas en ambos reportes, se llega a la conclusión que los dos servidores web de código abierto más usados son Apache y Nginx. A continuación se hará referencia a las características de cada uno.

1.4 Apache 2

Apache, lanzado en el año 1995, es un servidor web de código abierto, robusto, flexible, rápido y eficiente, continuamente actualizado, adaptado a los nuevos protocolos y cuya implementación se realiza de forma colaborativa. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo conocido como el Grupo Apache. Apache 2 es una profunda revisión del servidor Apache, centrándose en la escalabilidad, seguridad y rendimiento. Las principales revisiones de código se han llevado a cabo para crear una arquitectura realmente escalable. Apache cuenta con características importantes como:

Multiplataforma: Es utilizado en multitud de sistemas operativos, lo que lo hace prácticamente universal. Actualmente funciona en Windows 2000/NT/9x, Linux, Unix, FreeBSD, Solaris, Mac OS X, entre otros.

Introducción de Módulos de Multiprocesamiento (*MPM*, *Multi-Processing Module*s) en el manejo de peticiones: En Apache cada solicitud es atendida por un hilo separado o proceso y utiliza sockets sincrónicos [11]. Con la creación de la versión 2.0 se introducen los MPM, como una solución propuesta por el Grupo Apache debido a que la anterior arquitectura no trabajaba bien bajo plataformas que no estaban centradas en procesos como en el caso de Windows. Los MPM son responsables de conectar con los puertos de red de la máquina, aceptar las peticiones y generar los procesos hijos que se encargan de servirlas. Existen diferentes MPM, encontrándose entre ellos *prefork, worker, event, threaded* y *perchild,* cada uno basado en un funcionamiento diferente. Los sitios web que necesitan más que nada escalabilidad pueden usar un MPM hebrado como *worker*, mientras que los sitios web que requieran por encima de otras cosas estabilidad o compatibilidad con software antiguo pueden usar *prefork*. Además, se pueden configurar funcionalidades especiales como servir diferentes hosts con diferentes identificadores de usuario con el uso de *perchild* [12]**.**

Sistema de módulos dinámico: Apache se caracteriza por poseer un sistema de módulos dinámico, donde estos pueden ser habilitados y deshabilitados dinámicamente con la ejecución de un comando y después de haber reiniciado el proceso servidor. Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona. Existen infinidad de módulos disponibles, todos ellos denominados mod_XXX, donde el nombre XXX designa el tipo de funcionalidad que está pensado que desempeñen. Esta orientación permite al servidor ser personalizado de forma muy precisa, instalando o desinstalando servicios a demanda de los usuarios del mismo y que de esta forma solo estén en marcha aquellas funcionalidades realmente necesarias.

Soporte de hosts virtuales: Apache es además uno de los primeros servidores web en soportar hosts

virtuales basados en IP y/o hosts virtuales basados en nombre. Puede estar sirviendo cientos de sitios web clientes desde un único servidor.

Abundante soporte de documentación: El Grupo Apache usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada. Además de este grupo, cientos de personas han contribuido al proyecto con código, ideas y documentación.

Apache cuenta además con otras características como:

- Soporte del último protocolo HTTP 1.1[2].
- Configuración simple y potente a través de ficheros.
- Soporte para CGI (*Common Gateway Interface*).
- Soporte de FastCGI.
- Soporte de autentificación HTTP.
- Perl integrado.
- Soporte de scripts PHP.
- Soporte de servlets de Java.
- Servidor proxy integrado: Apache se puede convertir en un *caching* (*forward*) proxy server. Un forward proxy es un servidor intermedio que se sitúa entre el cliente y el servidor que tiene los contenidos. En este proceso se emplea el módulo mod_proxy.
- Soporte de SSI (*Server Side Includes*).
- Soporte de SSL (*Secured Sockect Layer*).
- Negociación avanzada de los contenidos que gestiona el servidor.
- Reescritura de las URL.
- Ejecución SetUID de programas CGI.
- Chequeo avanzado de sintaxis de la URL.
- Filtrado I/O: Otro cambio importante en Apache 2.0 es que proporciona arquitectura para I/O jerarquizada. Esto significa que una salida de un módulo puede convertirse en una entrada de otro

2 Con el surgimiento de HTTP 1.1, un navegador web puede enviar solicitudes en paralelo.

módulo. El efecto de este filtrado es muy interesante. Por ejemplo, la salida producida por scripts de CGI, que es procesado por el módulo mod_cg, puede ahora pasarse al módulo mod_include responsable de las SSI. En otras palabras, los scripts de CGI pueden producir un output en forma de etiquetas SSI, que se pueden procesar antes de que el output final se envíe al navegador web [13].

1.5 Nginx

Nginx fue creado por Ígor Sysóiev, el principal administrador de sistemas del buscador ruso Rambler. En 2002 empezó a crear su propio servidor y en 2004 publicó la primera versión de este producto. Se trata de un servidor web HTTP de código abierto, licenciado bajo la Licencia BSD simplificada. Posee características importantes como:

Multiplataforma: Inicialmente se creó para funcionar en sistemas operativos Unix, pero más tarde también apareció una versión compatible con Windows, por tanto se utiliza en sistemas como GNU/Linux, FreeBSD, Solaris, Mac OS X y Windows.

Arquitectura orientada a eventos para el manejo de peticiones: En lugar de abrir subprocesos para cada petición, Nginx utiliza una arquitectura basada en eventos que hace que su consumo de recursos crezca de forma predecible [14]. En el momento que se inicia el servidor existe en memoria solo el proceso principal. Este no es el que procesa las peticiones hechas por el cliente, sino otros procesos generados por él. Desde el archivo de configuración se puede definir la cantidad de procesos, las conexiones máximas por proceso y otros parámetros.

Sistema de módulos estático: Los módulos no pueden ser cargados de forma dinámica, deben ser incluidos en la compilación. Además, no se pueden desactivar en tiempo de ejecución debido a que están completamente compilados e integrados en el binario principal [15].

Soporte para hosts virtuales: Posee soporte para hosts virtuales basados en IP y/o hosts virtuales basados en nombre.

Escaso soporte de documentación: No posee aún la cantidad de documentación y usuarios de otros servidores web [16].

Se caracteriza además por otros aspectos como:

- Incluye servicios de correo electrónico con acceso al IMAP (*Internet Message Protocol)* y al servidor POP (*Post Office Protocol)*.

- Está listo para ser utilizado como un proxy inverso con opciones de caché.
- Tiene soporte para SSL, FastCGI, streaming de vídeo, autenticación, compresión gzip y balanceo de carga.
- Manejo de archivos estáticos, archivos de índices y autoindexado.
- Los parámetros de configuración están contenidos en el fichero nginx.conf, el cual está fuertemente estructurado para facilitar su lectura [17].
- Poseer módulo de reescritura de URL.
- Perl integrado.
- Control de acceso basado en la dirección IP del cliente y la autenticación HTTP BASIC.
- Transmisión de FLV y MP4 [18].

Las tres versiones existentes son: Estable, de Legado y de Desarrollo. Estable es la recomendada para utilización general. Legado es la versión que fue Estable hasta que la nueva apareciese. Desarrollo, a pesar de ser la que posee mayor probabilidad de que aparezcan nuevos errores, es la que contiene las nuevas funciones y en la que los errores del programa son eliminados primero [19].

Actualmente el sistema es usado por una larga lista de sitios web conocidos. Empresas como la compañía de TV online Hulu utilizan Nginx por su estabilidad y configuración simple. Otros usuarios, como Facebook y WordPress.com, lo utilizan porque la arquitectura asíncrona del servidor web deja una pequeña huella de memoria y bajo consumo de recursos, haciéndolo ideal para el manejo de múltiples y cambiantes páginas web activas [20].

1.6 Necesidades en las diferentes empresas e instituciones cubanas

Cada empresa o institución cubana tiene características diferentes en cuanto a aspectos relacionados con el alojamiento de contenido a publicar en un servidor web, como son:

- Cantidad de contenido a publicar: La cantidad de contenido a publicar puede variar de una empresa a otra, donde en algunos casos puede convertirse en un número muy grande.
- Direcciones IP asignadas: La PC servidora puede tener asignadas una o varias direcciones IP.
- Acceso de usuarios: Dependiendo de la empresa y del propósito que posea el contenido publicado,

la cantidad de usuarios que acceden al mismo puede variar considerablemente.

- Manejo de peticiones: Cada empresa puede necesitar gestionar el manejo de peticiones pertenecientes a la conexión a un sitio web de forma diferente, con el objetivo de lograr más eficiencia en este proceso.
- Funcionalidades necesarias para la administración: Para la administración de cada host virtual se deben incluir funcionalidades ofrecidas por el servidor, donde para cada host virtual estas varían dependiendo de las características del contenido publicado.
- Conocimiento de los administradores: Los administradores de servicios telemáticos tal vez no posean un nivel de conocimiento elevado en cuanto a la administración del servidor, quedando afectado en estos casos el aprovechamiento de todas las ventajas o características que ofrezca el mismo.

Tomando como referente los posibles entornos que pueden existir en las diferentes empresas e instituciones cubanas, se necesita un servidor web que se caracterice por:

- Realizar la asignación de hosts virtuales a las direcciones IP correspondientes a la PC servidora de forma que todos los contenidos sean publicados y se realice una buena gestión de las direcciones IP.
- Gestionar las peticiones basándose en la cantidad de usuarios que puedan tener acceso al contenido y logrando la eficiencia en este proceso según el objetivo de la empresa.
- Administrar los hosts virtuales en cuanto a las funcionalidades que requieran, sin hacer uso innecesario de otras que no se estén empleando.
- Poseer buen soporte de documentación que permita la autopreparación y adquisición de conocimientos por parte de los administradores del servidor.

1.7 Selección del servidor

De los servidores antes mencionados se hace una comparación haciendo referencia a sus semejanzas y diferencias en cuanto a las características más importantes, como se muestra en la Tabla 1.

Aspecto a comparar	Nginx	Apache 2
Sistemas Operativos compatibles	Multiplataforma	Multiplataforma
Tipos de hosts virtuales que soporta	Basados en nombre y/o basados en IP.	Basados en nombre y/o basados en IP.

Manejo de peticiones	Se basa en una arquitectura orientada a eventos	Se basa en sockets sincrónicos, hilos y procesos. Posee diferentes MPM para los diferentes entornos existentes.
Sistema de módulos	Estático	Dinámico y además posee una gran cantidad de módulos.
Soporte de documentación	Escaso	Abundante
Estadísticas de uso	Según las estadísticas de uso de servidores web reportadas por Netcraft, Nginx alcanzó el 1% de uso en enero del 2008 y ha ido aumentando poco a poco hasta llegar a 12,85% en febrero del presente año, como se aprecia en la Figura 1.	Según las estadísticas de uso de servidores web reportadas por Netcraft, desde marzo de 1996 hasta febrero del presente año Apache ha sido el servidor web más usado, como se aprecia en la Figura 1.

Tabla 1: Comparación de los servidores web Nginx y Apache.

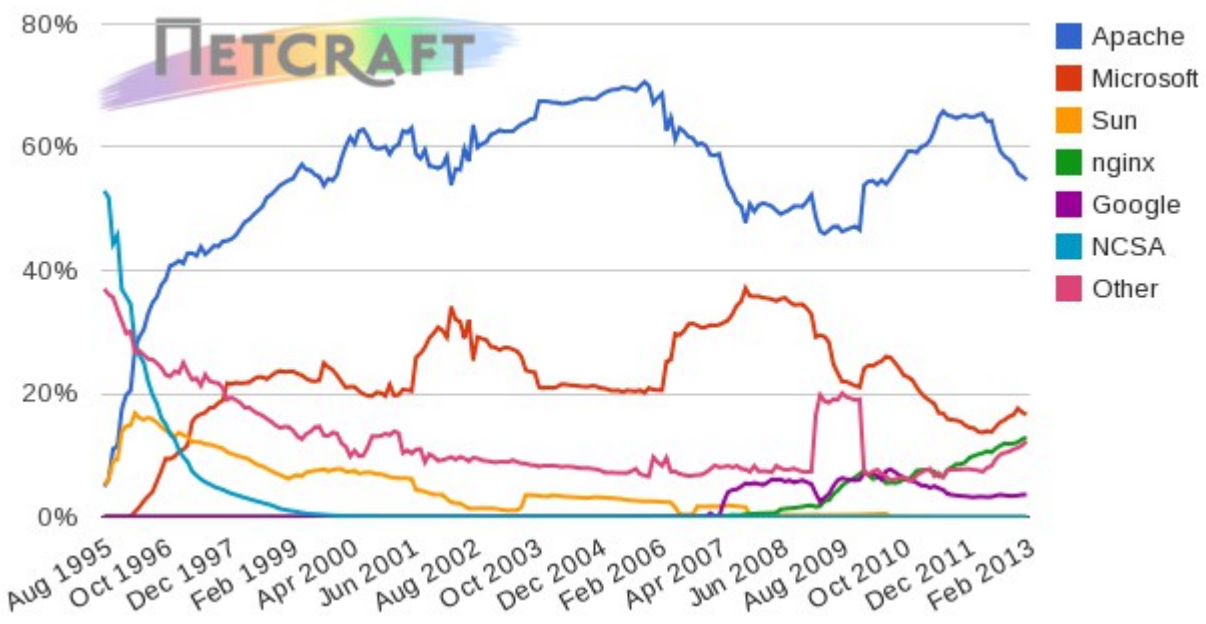


Figura 1: Estadísticas del uso de servidores web desde agosto de 1995 hasta febrero del presente año según Netcraft [21].

Nginx y Apache se asemejan por ser multiplataforma y soportar hosts virtuales basados en nombre y/o hosts virtuales basados en IP, sin embargo se diferencian en cuanto a varios aspectos como el manejo de peticiones, sistema de módulos, soporte de documentación y estadísticas de uso. Por tanto se decide

seleccionar Apache 2 como servidor web a administrar por las ventajas que posee frente a Nginx. Se realiza la selección teniendo en cuenta que las características de Apache se ajustan y cumplen con las necesidades de las empresas e instituciones en las que se va a emplear el módulo a desarrollar, por tanto permitirá una administración a la medida.

1.8 Administración de Apache 2

El servidor Apache no posee una interfaz de usuario gráfica para su administración. Se trata de un sencillo archivo de configuración llamado apache2.conf o httpd.conf, según el sistema operativo, en este caso es apache2.conf debido a que se trabajará con los sistemas operativos Nova, Ubuntu o Debian. Se pueden mover partes de la configuración de Apache a diversos ficheros. Un ejemplo es poder aislar la configuración de cada host virtual en un archivo independiente para cada una, o bien distribuir la configuración de diferentes localizaciones físicas a ficheros independientes más pequeños y fáciles de manejar que se encuentren directamente en dichas localizaciones. No es necesario concentrar toda la configuración en un solo archivo de gran tamaño. De esta forma se evita que el fichero de configuración principal apache2.conf alcance un tamaño demasiado grande y por tanto sea más complejo de manejar [22]. Toda la configuración referente a Apache se encuentra en el directorio /etc/apache2, el cual incluye:

- *apache2.conf*: Fichero principal de configuración de Apache 2.
- *httpd.conf*: Fichero de configuración que se mantiene por compatibilidad con la anterior versión de Apache y por algunos módulos especiales. Se considera obsoleto.
- *mods-available*: Directorio con los módulos disponibles. Cada módulo consta de un fichero para cargar (.load) y otro, opcional, para la configuración (.conf). Los ficheros binarios de los módulos (.so) están en /usr/lib/apache2/modules.
- *mods-enabled*: Directorio con los módulos activos. Normalmente son enlaces simbólicos a los ficheros de un módulo de la carpeta anterior.
- *ports.conf*: Fichero para configurar el puerto de comunicación HTTP. En este también se especifica la directiva NameVirtualHost para los hosts virtuales basados en nombre.
- *README*: Documentación sobre este directorio.
- *sites-available*: Directorio con los ficheros de configuración de los hosts virtuales disponibles. Se recomienda crear cada host virtual en un fichero.

- *sites-enabled*: Directorio con los ficheros de configuración de los hosts virtuales activos. Se usa la misma filosofía que para activar módulos. Si se quiere activar un host virtual se crea un enlace directo en esta carpeta que apunte a sites-available.
- *envvars*: Fichero que incluye las variables de entorno que se deseen cargar.
- *magic*: Datos para el módulo mod_mime_magic.

La configuración se realiza mediante directivas y secciones. Las directivas son variables almacenadas en el archivo de texto de configuración para alterar y controlar el funcionamiento de Apache en tiempo de ejecución según sus valores y después de haber reiniciado el proceso servidor [23]. Tienen la siguiente estructura: *NombreDeDirectiva ValorDeDirectiva*. Se aplican a todas las páginas del servidor pero si se quiere que ciertas de ellas se apliquen a páginas, directorios o ficheros concretos, se tendrán que usar secciones de configuración.

Antes de efectuar cualquier cambio en el archivo principal, es conveniente realizar una copia de seguridad del mismo ya que si Apache encuentra algún error en la configuración, no arrancará. En este se pueden incluir diferentes ficheros de configuración mediante la directiva *Include*.

Los archivos de configuración contienen dos tipos de información: comentarios y directivas. Las líneas que comienzan con un carácter # son tratadas como líneas de comentario y son ignoradas por el servidor cuando analiza el archivo. Exceptuando estas y las líneas en blanco, el resto de las líneas son tratadas como directivas completas o como directivas parciales. Cada vez que se realice una modificación es necesario reiniciar o recargar Apache 2 para que los cambios tengan efecto.

Algunos parámetros son generales para el servidor, mientras que otros se pueden configurar de manera independiente para cada conjunto de directorios o ficheros o para un servidor virtual concreto. La configuración de Apache está dividida en tres secciones fundamentales, aunque las directivas de cada una pueden aparecer mezcladas y desordenadas.

- *Sección 1*: Entorno global. Describe el funcionamiento general del servidor. Incluye parámetros como los puertos, MPM y manejo de conexiones, en algunos casos el fichero principal contiene la configuración y en otros la ruta a otro fichero.
- *Sección 2*: Servidor principal. Se describe la configuración del servidor principal, que es la base sobre la que se construyen los hosts virtuales. La configuración está contenida en el archivo principal.

- *Sección 3*: Hosts virtuales. Se pueden alojar más de un host virtual en el mismo servidor, cada uno en un fichero de configuración independiente. En el archivo principal aparece la ruta al directorio que contiene todos estos ficheros.

Una vez definido el servidor a administrar y sus características en cuanto a la administración, se hace necesaria la descripción de la herramienta HMAST y los requerimientos que deben tener los módulos a integrar, pues constituye la base sobre la que se desarrolle el módulo propuesto.

1.9 Descripción de la herramienta HMAST

HMAST cuenta con un sistema base que permite la administración de las PC servidoras de forma remota, el cual tiene las funcionalidades necesarias para administrar los usuarios, las tareas programadas y los servicios, entre otros.

1.9.1 Arquitectura

La arquitectura presentada por la herramienta HMAST propone el diseño de una arquitectura N-Capas orientada al dominio, distribuida en cinco componentes o paquetes, como se muestra en el Anexo 3. La interacción entre los mismos se realiza a través de interfaces y utilizando inyección de dependencias.

La capa de **Presentación** es la encargada de presentar al usuario los conceptos de negocio mediante una interfaz de usuario (IU), facilitar la explotación de dichos procesos, informar sobre la situación de los procesos de negocio e implementación de las reglas de validación de dicha interfaz. Para la implementación de la misma se utiliza el módulo MVC de Spring.

La capa de **Aplicación** es responsable de realizar llamadas a servicios de capas inferiores (Dominio). Los servicios que publica (servicios de aplicación) tienen la responsabilidad, por ejemplo, de adaptar la información que le llega a los requerimientos de los servicios de dominio. En esta capa también se ubican operaciones de trazas, seguridad, envío de correos electrónicos, cuando no forman parte estricta del negocio.

La capa de **Dominio** constituye el hilo conductor de la aplicación, sus componentes solo dependen de la Capa de Infraestructura Transversal. Implementa la lógica de dominio (reglas de negocio), es responsable de las validaciones. Define las interfaces de persistencia a datos (contratos de repositorio) pero no los implementa. Sus componentes no están ligados a tecnologías específicas.

La capa de **Persistencia** tiene asignada la responsabilidad de contener el código necesario para persistir

los datos. Los principales componentes que contendrá la capa son los repositorios, que son clases que implementan los contratos de repositorios definidos en la capa de Dominio.

Las responsabilidades de la capa **Infraestructura Transversal** están dadas a promover la reutilización de código, contiene las operaciones de seguridad, logging, monitoreo del sistema, mecanismos de persistencia reutilizables, validadores genéricos y todas aquellas operaciones que se puedan llamar desde otras capas.

1.9.2 Funcionalidades que ofrece

La herramienta cuenta con funcionalidades como son:

- **Gestión de servidores lógicos:** Permite la adición, edición y eliminación de los datos de un servidor lógico, además permite la conexión remota y desconexión a un servidor seleccionado.
- **Gestión de servicios telemáticos asociados a un servidor lógico:** Permite la adición, edición y eliminación de los datos de un módulo, así como activación y desactivación de los mismos.
- **Gestión de las variables de configuración asociadas a un servidor lógico:** Permite cargar y salvar las variables de configuración de los servicios telemáticos encontrados en un servidor lógico (ficheros de configuración, nombre de módulos, demonios, entre otros).

1.9.3 Consideraciones para implementar un módulo para HMAST

Para la implementación de un módulo que se desee integrar en HMAST se debe tener en cuenta que:

- La lógica de Aplicación no deberá incluir ninguna lógica del Dominio, solo tareas de coordinación relativas a requerimientos técnicos de la aplicación, como conversiones de formatos de datos de entrada a entidades del Dominio, llamadas a componentes de Infraestructura para que realicen tareas complementarias.
- Se debe garantizar que no se envíen hacia y desde la capa de Presentación objetos de Dominio, en su lugar deben viajar Objetos de Transferencia de Datos (*DTO, Data Object Transfer*).
- Las clases de servicios deben ser las únicas responsables (vías de acceso) de acceder a los repositorios, no se puede implementar código de persistencia a datos en la capa de Dominio.
- Solo se puede acceder a la información almacenada en los servidores haciendo uso de los repositorios.

- Es importante que todo el código reutilizable por más de un repositorio se ponga a disposición de todos en la capa de Infraestructura Transversal.

1.10 Herramientas y tecnologías a emplear

Para el desarrollo del módulo propuesto se definió el uso de las siguientes herramientas y tecnologías, pues constituye un requisito debido a que HMAST está desarrollada con las mismas.

1.10.1 Framework

Un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio [24].

- **Spring**

Spring Framework, también conocido simplemente como Spring, es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java, aunque también existe una versión para la plataforma .NET. Por su diseño, ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. Aunque contiene muchas características organizadas en grandes módulos, que le dan una funcionalidad muy amplia, sus características principales son la inyección de dependencias y la programación orientada a aspectos [25].

La inyección de dependencias tiene como objetivo lograr un bajo acoplamiento entre los objetos de la aplicación. Con este patrón de diseño, los objetos no crean o buscan sus dependencias sino que estas son dadas al objeto. El contenedor (*Context*) es el encargado de realizar este trabajo al momento de instanciar el objeto. Se invierte la responsabilidad en cuanto a la manera en que un objeto obtiene la referencia a otro. De esta manera, los objetos conocen sus dependencias por su interfaz. Así la dependencia puede ser intercambiada por distintas implementaciones a través del contenedor.

La otra característica relevante es la Programación Orientada a Aspectos (AOP), que es un paradigma de

programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Gracias a la AOP se pueden capturar los diferentes conceptos que componen una aplicación en entidades bien definidas, de manera apropiada en cada uno de los casos y eliminando las dependencias entre cada uno de los módulos. De esta forma, se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables.

1.10.2 Lenguaje de programación.

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un programa informático. Un lenguaje de programación permite a un programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos, además de qué acciones debe tomar bajo una variada gama de circunstancias [26].

- **Java**

Lenguaje de programación que a diferencia de C++ no hace uso de punteros. Java fue diseñado como un lenguaje orientado a objetos, los cuales agrupan en estructuras encapsuladas tanto sus datos como las funcionalidades. Proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Es interpretado y compilado a la vez. Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores. Soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas [27].

1.10.3 Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Puede estar pensado para su utilización con un único lenguaje de programación o para varios de estos. Además existen entornos de desarrollo en los que se pueden utilizar varios lenguajes de programación.

- **Netbeans**

Netbeans es un Entorno de Desarrollo Integrado (IDE) gratuito de código abierto, hecho principalmente para el lenguaje de programación Java. Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis, entre otros. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, aunque permite crear aplicaciones con Python, identificar errores y el debuger.

1.10.4 Herramientas CASE.

Las herramientas CASE (*Computer Aided Software Engineering*) son diversas aplicaciones informáticas que tienen el propósito de aumentar la productividad en el desarrollo de software, disminuyendo el costo de las mismas en términos de dinero y tiempo. Pueden contribuir en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el cálculo de costos, la compilación automática, el proceso de realizar un diseño del proyecto, la implementación de parte del código automáticamente con el diseño dado, la documentación o la detección de errores, entre otras [28].

- **Visual Paradigm para UML**

Visual Paradigm para UML es una herramienta CASE, que soporta el ciclo de vida completo del desarrollo de software. Es potente y fácil de utilizar, permite el modelado visual UML propiciando la rápida construcción de aplicaciones de calidad. Es una herramienta colaborativa, pues soporta múltiples usuarios trabajando sobre el mismo proyecto. Con el uso de esta se pueden dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Esta herramienta se caracteriza por: soporte para varias versiones de UML, una versión gratuita y otra comercial, permitir la ingeniería inversa (código a modelo, código a diagrama), permitir la generación de código (modelo a código, diagrama a código), permitir exportar el diseño de la base de datos para varios gestores y ser un producto multiplataforma [29].

1.10.5 Sistema de Control de Versiones

Un sistema de control de versiones es una combinación de tecnologías y prácticas para seguir y controlar

los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en la documentación y en las páginas web [30]. Todo este proceso se realiza manteniendo una correcta gestión sobre las versiones de la información almacenada.

- **RapidSVN**

RapidSVN es un cliente de interfaz gráfica para la comunicación con servidores Subversion. Está escrito en C++ y distribuido bajo licencia GPL. Este facilita el versionado de ficheros, desde una interfaz sencilla e intuitiva y se encuentra disponible para plataformas Windows, Linux, MAC OS X y Solaris. Es una herramienta rápida y eficiente.

1.10.6 Herramienta de diseño gráfico Pencil

Pencil es una herramienta gratuita y de código abierto para diseñar gráficos y animarlos en dos dimensiones. La interfaz gráfica de Pencil dispone de los elementos básicos para crear dibujos y personalizarlos al estilo deseado. Se caracteriza por poseer soporte para dibujo de diagramas y exportar los dibujos a diferentes formatos de salida. Permite el vínculo entre páginas, debido a que los elementos de un dibujo se pueden vincular a una página específica en el mismo documento.

1.10.7 Augeas

Augeas es una herramienta de edición de ficheros de configuración del sistema GNU/Linux. Es un editor que analiza los archivos en sus formatos nativos y los transforma en un tipo abstracto de datos llamado árbol y viceversa. La manipulación de la configuración la realiza mediante este árbol y los cambios que sean aplicados a él se guardan de forma nativa en dichos archivos de configuración. Un nodo del árbol tiene asignada una etiqueta y un valor, además le corresponden varios hijos que son una lista de nodos que pueden tener la misma etiqueta. Augeas es una herramienta modular, pero a diferencia de otras herramientas modulares esta depende totalmente de sus módulos, debido a que son los encargados de permitir la transformación de un fichero en árbol y viceversa. A estos módulos se le llaman *Lenses* y son los encargados de brindar soporte a cada fichero de configuración, pues cada uno describe un archivo [31].

1.11 Metodología de desarrollo de software

Las metodologías de desarrollo de software son el conjunto de procedimientos, técnicas, herramientas y

un soporte documental, que ayuda a los desarrolladores a realizar nuevo software. La metodología define quién debe hacer qué, cuándo y cómo debe hacerlo para obtener los distintos productos parciales y finales. Se clasifican en dos tipos: tradicionales y ágiles.

Aunque el proceso de desarrollo de software incluye otros aspectos importantes y determinantes, es precisamente la metodología utilizada, el elemento rector a lo largo del mismo. Por tal motivo es que se debe analizar, de acuerdo a las características del software a desarrollar y del equipo de desarrollo, cuál es la más óptima. Por las ventajas que ofrecen las metodologías ágiles, se propone su uso para el desarrollo del trabajo de diploma, específicamente la metodología SXP. Además, esta es la definida por el proyecto en el desarrollo de sistemas similares.

1.11.1 Metodología SXP

SXP es un híbrido de metodologías ágiles que toma las mejores prácticas de las metodologías SCRUM y XP además de regirse por los lineamientos de calidad de la UCI. Con la aplicación de esta metodología se logra la gestión de un equipo de forma que se tengan siempre medidos los progresos y que el trabajo se realice de forma eficiente. Se caracteriza por una programación rápida o extrema, teniendo como parte del equipo al usuario final, siendo uno de los requisitos para llegar el éxito del proyecto.

Consta de 4 fases principales: Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto; Desarrollo en la cual se realiza la implementación del sistema hasta que esté listo para ser entregado; Entrega que es responsable de la puesta en marcha y Mantenimiento donde se realiza el soporte para el cliente.

Se caracteriza por ser una metodología iterativa e incremental, basada en Historias de Usuario (HU), con pequeñas mejoras unas tras otras, está atenta al cambio y permite que el equipo de programación se mantenga en una interacción frecuente con el cliente o usuario. Está indicada especialmente para proyectos de pequeños equipos de trabajo, con requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Permite que el trabajo se realice de forma unida, en la misma dirección, con un objetivo claro, siguiendo el avance de las tareas a realizar [32].

Conclusiones parciales

Con la revisión y análisis de la bibliografía especializada en los servidores web y específicamente los de código abierto, se ha logrado la comprensión de los conceptos más relevantes relacionados con los

mismos, sus características y clasificaciones. Partiendo de dichos conocimientos se seleccionó Apache 2 como servidor a administrar, por ajustarse además a las necesidades de las diferentes empresas e instituciones cubanas. La definición de las características que posee HMAST permitió definir los requerimientos que debe tener un módulo para integrarse a la herramienta. Las tecnologías a emplear para el desarrollo del módulo debido a que son las que se requieren en HMAST son: framework Spring, lenguaje de programación Java, IDE Netbeans. Visual Paradigm para UML como herramienta CASE, RapidSVN como sistema de control de versiones, Pencil para el diseño gráfico y Augeas para la configuración de ficheros; donde todo el proceso está guiado por la metodología ágil de desarrollo de software SXP.

Capítulo 2: Módulo para la administración de los servidores web.

Una vez definido Apache 2 como servidor a administrar y detallados los requerimientos que se necesitan en HMAST, en el presente capítulo se definirán las características y funcionalidades que tendrá el sistema, las tareas a realizar para darle cumplimiento a las mismas, así como la arquitectura y patrones de diseño a emplear.

2.1 Propuesta del módulo

El presente trabajo de diploma propone el desarrollo de un módulo para la administración de los servidores web, permitiendo realizar la administración de los mismos en la herramienta HMAST de forma que se asegure la adaptabilidad a las diferentes empresas e instituciones cubanas. Como servidor web a administrar se seleccionó Apache 2, por tanto el sistema está estructurado basándose en su funcionamiento. El módulo permitirá la administración de Apache 2 en diferentes PC servidoras de forma remota. Se podrán administrar las tres secciones de configuración de Apache 2 vistas anteriormente, donde la configuración del entorno global afecta tanto al servidor principal como a los hosts virtuales, ya sean basados en nombre o basados en IP. Facilita un mejor entendimiento por parte de los usuarios para la gestión de los hosts virtuales debido a que se muestra de forma detallada la estructura de almacenamiento lógico de los mismos.

2.1.1 Funcionamiento de Apache 2

Apache trae consigo un sitio web por defecto que es la base sobre la que se construyen los hosts virtuales, este es llamado servidor principal. Los hosts virtuales heredan configuración del mismo, lo que da lugar a una configuración del host virtual mucho más manejable. Como se había hecho mención en el capítulo anterior, Apache soporta hosts virtuales basados en nombre y/o hosts virtuales basados en IP.

Una PC servidora puede tener asignadas una o varias direcciones IP y cada dirección IP puede escuchar por una lista de puertos específicos y además por todos los puertos, que en la configuración se representa con el símbolo de asterisco (*). A cada combinación *direcciónIP:puerto* se puede asignar un host virtual basado en IP o una lista de hosts virtuales basados en nombre.

Un host virtual basado en IP se refiere a la asignación de un solo host virtual a una única combinación *direcciónIP:puerto*, por tanto esta dirección IP escuchando por ese puerto tendría asociado un único

directorio raíz. Por otra parte una lista de hosts virtuales basados en nombre se refiere a la asignación de varios hosts virtuales a una única combinación *direcciónIP:puerto*, donde esta dirección IP escuchando por ese puerto tendría asociado un directorio raíz por cada host virtual. La diferencia de este tipo con el anterior es que para la creación de una lista de hosts virtuales basados en nombre se debe especificar una directiva *NameVirtualHost direcciónIP:puerto* en el fichero */etc/apache2/ports.conf*, aclarando que para ese par existen varios hosts virtuales basados en nombre.

Pueden existir además hosts virtuales asignados a todas las direcciones IP, que en la configuración al igual que en los puertos, se representa con el símbolo de asterisco (*). En el caso específico de la combinación de todas las direcciones IP con todos los puertos (*:*), cuando se asigna más de un host virtual no es necesario especificar la directiva *NameVirtualHost*. Para cada dirección IP asignada a la máquina (incluyendo todas) que contenga algún host virtual, correspondería una estructura de almacenamiento lógico como la que se muestra en la Figura 2.

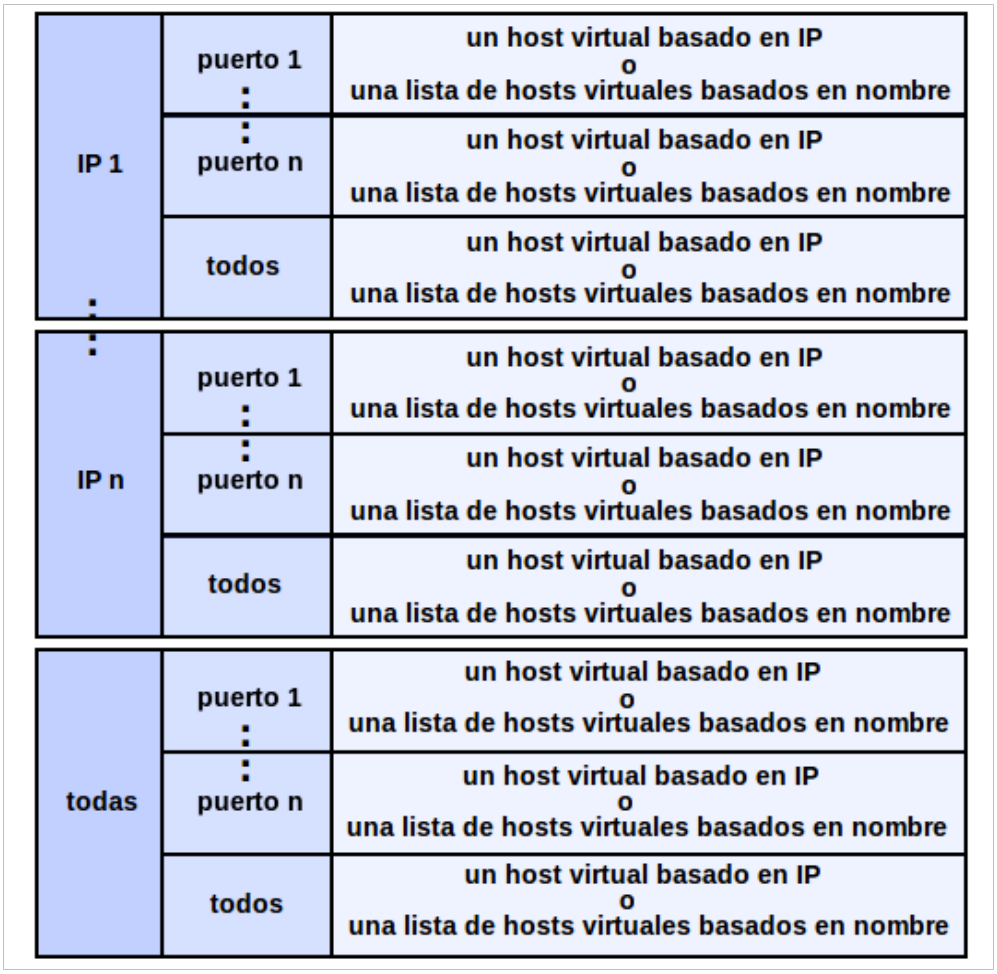


Figura 2: Estructura de almacenamiento lógico de los hosts virtuales en Apache 2.

Un mismo host virtual puede estar asignado a varias combinaciones *direcciónIP:puerto*. En este caso, aunque en el directorio donde se guardan los ficheros de configuración de cada host virtual, aparece este host solo una vez, en la estructura de almacenamiento lógico se guarda en cada combinación a la que esté asignado.

Al iniciarse Apache 2 se genera una lista por cada combinación *direcciónIP:puerto* y se inserta en una tabla hash. Si esta combinación es usada en una directiva *NameVirtualHost*, la lista contiene todos los hosts virtuales basados en nombre correspondientes a la misma. Si la combinación es usada en una directiva *NameVirtualHost* pero no tiene hosts asignados, se ignora la directiva y se registra un error. Para los hosts virtuales basados en IP la lista en la tabla hash está vacía.

Cuando llega una petición al servidor se realiza una búsqueda del host virtual que debe atenderla, realizando varios pasos tal como se muestra en el Anexo 4. En caso de no existir coincidencias entre la dirección IP y/o el puerto de la petición con los asignados a los hosts virtuales, la petición será atendida por el servidor principal.

2.2 Artefactos generados en el desarrollo

En el desarrollo del módulo guiado por la metodología SXP, se generaron artefactos como son la Lista de Reserva del Producto, el Plan de Liberación, las Historias de Usuario y las Tareas de Ingeniería.

2.2.1 Lista de Reserva del Producto (LRP)

La LRP contiene los requisitos agrupados y ordenados según su prioridad, definiendo así la planificación del trabajo a realizar. Durante el proceso de desarrollo de software (entre iteraciones) esta puede ser modificada y mejorada con nuevos requerimientos, garantizando una mejor calidad del producto final.

Prioridad	Ítem *	Descripción	Estimación (semanas)	Estimado por
Muy Alta				
	1	Mostrar hosts virtuales asignados a Apache 2.	1,0	Nurisel Palma
	2	Adicionar host virtual a Apache 2.	1,8	
	3	Modificar host virtual asignado a Apache 2	1,8	
	4	Eliminar hosts virtuales asignados a Apache 2.	1,0	
	5	Deshabilitar host virtual asignado a Apache 2.	0,2	

	6	Habilitar host virtual asignado a Apache 2	0,2	
	7	Instalar Apache 2.	0,2	
	8	Iniciar Apache 2.	0,2	
	9	Detener Apache 2.	0,2	
	10	Reiniciar Apache 2.	0,2	
	11	Recargar Apache 2.	0,2	
Alta				
	12	Modificar puertos para las conexiones en Apache 2.	1,0	
	13	Mostrar módulos pertenecientes a Apache 2.	0,2	
	14	Deshabilitar módulo perteneciente a Apache 2.	0,4	
	15	Habilitar módulo perteneciente a Apache 2.	0,4	
Media				
	16	Mostrar parámetros de configuración del servidor principal de Apache 2.	0,5	
	17	Modificar parámetros de configuración del servidor principal de Apache 2.	0,5	
	18	Mostrar parámetros en el MPM prefork de Apache 2.	0,5	
	19	Modificar parámetros en el MPM prefork de Apache 2.	0,5	
	20	Mostrar parámetros para el manejo de conexiones en Apache 2.	0,5	
	21	Modificar parámetros para el manejo de conexiones en Apache 2.	0,5	
RNF (Requisitos No Funcionales)				
	1	Todas las salidas del sistema deben ser precisas y veraces.		
	2	El sistema contará con un manual de ayuda.		
	3	La aplicación se ejecutará en los sistemas operativos Nova, Ubuntu o Debian.		
	4	Usar como lenguaje de programación Java.		

	5	El IDE a utilizar será NetBeans 7.2.		
	6	Usar como framework de desarrollo Spring.		
	7	Usar Augeas para la configuración de ficheros.		

2.2.2 Plan de Liberación

El Plan de Liberación contiene las iteraciones que se van a realizar y las características correspondientes a cada una. Se definen las Historias de Usuario que se van a desarrollar en cada iteración y la estimación de tiempo en semanas.

Liberación	Descripción de la iteración	Orden de la HU a implementar	Duración total (semanas)
1	Dar cumplimiento a las Historias de Usuario de prioridad Muy Alta.	1,2	7 semanas.
2	Dar cumplimiento a las Historias de Usuario de prioridad Alta.	3,4	2 semanas
3	Dar cumplimiento a las Historias de Usuario de prioridad Media.	5, 6, 7	3 semanas

2.2.3 Historias de Usuario

Las Historias de Usuario especifican las tareas que debe realizar el sistema, lo que equivale a los casos de uso en el proceso unificado. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo.

Historia de Usuario	
Número: HMAST_HTTP_1	**Nombre Historia de Usuario:** Gestionar hosts virtuales asignados a Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 6 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 6 semanas

Descripción: El software permitirá al usuario mostrar, adicionar, eliminar, modificar, habilitar y deshabilitar hosts virtuales.

Observaciones: El sistema debe permitir al usuario:

- **Mostrar hosts virtuales:** La funcionalidad comienza cuando se selecciona la opción Hosts Virtuales del panel a la izquierda. Primeramente se selecciona la dirección IP de la cual se quieren listar los hosts virtuales asignados, esta puede ser una dirección específica o todas (*). Se mostrará la lista con todos los hosts virtuales existentes agrupados por los puertos de escucha y de ellos estarán marcados los que estén habilitados.
- **Adicionar hosts virtuales:** La funcionalidad comienza cuando se selecciona la opción Hosts Virtuales del panel a la izquierda, se especifica la dirección IP y una vez mostrado el listado de hosts asignados, se selecciona adicionar un host virtual. Posteriormente se muestra la interfaz de la adición con el campo de la dirección IP lleno, aunque puede modificarse. En caso que se haya escogido un puerto, la interfaz de la adición tendría los campos de la dirección IP y el puerto llenos, aunque pueden ser modificados. Para la adición los parámetros se dividen en cuatro grupos.

  1. Generales:

  Esta se divide a su vez en tres bloques:

  Parámetros generales:

  * Nombre del fichero: Se especificará el nombre del fichero que contendrá el host virtual. El campo es obligatorio.

  * Dirección IP y puerto: Se pueden adicionar una o varias combinaciones de estos parámetros. La dirección IP puede ser una de las que están asignadas a la PC servidora o la palabra todas. Además se debe seleccionar un puerto específico o todos los puertos por los que esté escuchando Apache. Puede especificarse un número de puerto entre 0 y 65535, aunque debe validarse que no esté siendo usado por otro servicio. Es obligatorio especificar al menos una combinación.

  * Directorio raíz: La ruta debe ser de un directorio que exista en el servidor.

  * Nombre del host: Se debe validar que sea un nombre de dominio válido. Sin embargo se debe notificar al usuario que aunque sea un nombre válido si no está configurado en el DNS el host

virtual no funcionará.

* Alias del nombre: Puede especificarse más de un alias. Se hacen las mismas validaciones que para el nombre del host.

Soporte SSL:

* Habilitar soporte SSL: Se puede habilitar o deshabilitar.

* Fichero de certificado: Si selecciona la opción por defecto se utiliza entonces el fichero /etc/ssl/certs/ssl-cert-snakeoil.pem, en caso contrario se puede seleccionar otro. La ruta debe ser de un fichero que exista en el servidor.

* Fichero clave de certificado: Si selecciona la opción por defecto se utiliza entonces el fichero /etc/ssl/private/ssl-cert-snakeoil.key, en caso contrario se puede seleccionar otro. La ruta debe ser de un fichero que exista en el servidor.

Errores:

* Fichero de registro de errores: Se debe validar que el parámetro entrado sea una ruta.

* Dirección de correo del administrador: Se debe validar que sea una dirección de correo válida, compuesta por un nombre de usuario, ”@” y un nombre de dominio.

* Información sobre el servidor: El usuario podrá seleccionar una de las opciones, mostrar la información, no mostrarla o mostrarla incluyendo la dirección de correo del administrador.

2. Archivos índices:

El software permitirá gestionar los archivos del servidor Apache como índice para el directorio raíz del host virtual. Pueden especificarse varios nombres de archivos, el servidor irá buscando los archivos en el mismo orden en que aparezcan listados. Si el archivo está ausente, Apache devuelve la lista de directorios creando una página HTML. En esta pestaña se mostrará al usuario una lista con todos los archivos índices, donde se puede cambiar el orden de los mismos. También permitirá adicionarlos, eliminarlos y modificarlos. En el adicionar y el modificar se debe validar que el parámetro entrado sea un nombre. Para eliminar uno o varios elementos de la lista, estos deben ser seleccionados. Para modificar se debe seleccionar el archivo deseado.

3. Alias de directorios:

En esta pestaña se mostrará al usuario una lista con todos los directorios a los que se les haya establecido un alias de acceso. También permitirá adicionar, eliminar y modificar los mismos. Para adicionar, los parámetros requeridos son el alias de acceso y la ruta del directorio. Se debe validar en el adicionar y el modificar que ambos parámetros sean rutas. Para eliminar uno o varios elementos de la lista, estos deben ser seleccionados. Para modificar se debe seleccionar el elemento deseado.

4. Permisos de acceso:

Se especifican permisos de acceso para ficheros y directorios. En ambos casos se pueden especificar diferentes opciones como son los orígenes a permitir o denegar, que pueden ser todos o una lista específica. También se puede establecer el orden en que van a ser leídas estas listas. En el caso de los directorios también se especifican opciones adicionales como son:

* None: No hay opciones.

* All: Todas las opciones excepto para Multiviews.

* ExecCGI: Está permitida la ejecución de scripts CGI.

* FollowSymLinks: El servidor sigue los enlaces simbólicos en el directorio.

* Includes: Están permitidos los comandos SSI.

* IncludesNOEXEC: Se puede embeber un conjunto restringido de comandos SSI en páginas SSI. Los comandos SSI que no están disponibles son #exec y #include.

* Indexes: Si se solicita una URL que está integrada en un directorio y no hay archivo índice en ese directorio, entonces el servidor devuelve una lista formateada del directorio.

* Multiviews: Soporta las visualizaciones múltiples de los contenidos.

* SymLinksIfOwnerMatch: Se sigue un enlace simbólico sólo si los propietarios del enlace y del destino coinciden.

En el primer campo se puede seleccionar la opción directorio y especificar una ruta o seleccionar la opción fichero y especificar un nombre, es un campo obligatorio. Si es un directorio se habilita el bloque de Opciones. Debe validarse que la ruta o el nombre sean correctos, según la opción especificada. En el bloque Orden puede seleccionarse una de las dos opciones. En los bloques Orígenes a permitir y Orígenes a denegar, para el caso de las listas pueden establecerse

diferentes orígenes válidos como:

- Un nombre de dominio completo o parcial.

- Una dirección IP completa o parcial. Ejemplos: 10, 10.4, 10.58.3

- Un par dirección IP/máscara. Hay dos casos válidos: 10.1.0.0/255.255.0.0 ó 10.1.0.0/16.

Se mostrará al usuario una lista con los directorios y ficheros a los que se les ha establecido algún permiso. También permitirá adicionar, eliminar y modificar los mismos. Para adicionar, los parámetros requeridos son los mencionados anteriormente. Para eliminar uno o varios elementos de la lista, estos deben ser seleccionados. Para modificar se debe seleccionar el elemento deseado.

- **Modificar hosts virtuales:** Una vez mostrada la lista con todos los hosts virtuales asignados a la dirección IP seleccionada, el usuario para modificar uno, solo tendrá que seleccionarlo. Las interfaces y parámetros son los mismos que en el Adicionar, todos los parámetros pueden modificarse excepto el nombre del fichero.
- **Eliminar hosts virtuales:** Una vez mostrada la lista con todos los hosts virtuales asignados a la dirección IP seleccionada, el usuario para eliminar uno o varios, solo tendrá que seleccionarlos, posteriormente se muestra un mensaje de confirmación.
- **Habilitar hosts virtuales:** Una vez mostrada la lista con todos los hosts virtuales asignados a la dirección IP seleccionada y de ellos marcados los que están habilitados, el usuario para habilitar otro, solo tendrá que marcarlo.
- **Deshabilitar hosts virtuales:** Una vez mostrada la lista con todos los hosts virtuales asignados a la dirección IP seleccionada y de ellos marcados los que están habilitados, el usuario para deshabilitar alguno solo tendrá que desmarcarlo.

Prototipos de interfaz: Ver Anexo 5: Gestionar hosts virtuales asignados a Apache2. Anexo 6: Adicionar hosts virtuales. Pestaña Generales. Anexo 7: Adicionar hosts virtuales. Pestaña Archivos índices. Anexo 8: Adicionar archivo índice en un host virtual. Anexo 9: Adicionar host virtual. Pestaña Alias de directorios. Anexo 10: Adicionar alias a un directorio en un host virtual. Anexo 11: Adicionar host virtual. Pestaña Permisos de acceso. Anexo 12: Adicionar permisos de acceso en un host virtual.

Historia de Usuario

Número: HMAST_HTTP_2	**Nombre Historia de Usuario:** Manejar estados de Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Permitirá manejar los estados del servidor Apache 2 realizando diferentes acciones como: instalar, iniciar, reiniciar, detener y recargar.	
Observaciones: Cuando se accede al módulo en caso que el servidor no esté instalado, se podrá instalar y posteriormente administrarlo, donde la interfaz principal permitirá la gestión de los hosts virtuales. Las acciones de iniciar, reiniciar y detener se realizarán en el momento que el usuario lo desee. En el caso de recargar se realizará de forma automática después de cada configuración.	
Prototipos de interfaz: Ver Anexo 13: Primer paso para instalar el servidor Apache 2. Anexo 14: Segundo paso para instalar el servidor Apache 2. Anexo 15: Tercer paso para instalar el servidor Apache 2. Anexo 16: Manejar estados de Apache 2.	

Historia de Usuario	
Número: HMAST_HTTP_3	**Nombre Historia de Usuario:** Modificar puertos para las conexiones en Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Permitirá modificar los puertos que utiliza Apache 2 para las conexiones.	
Observaciones: La modificación de los puertos utilizados para las conexiones en Apache 2 se realizará de forma implícita cuando se guarden todos los cambios realizados, una vez que haya finalizado la gestión de todos los hosts virtuales. Se adicionarán todos los puertos que hayan sido asignados a algún host virtual, en caso que no se encuentren en la lista de puertos de Apache 2.	
Prototipos de interfaz: No tiene interfaz visual.	

Historia de Usuario	
Número: HMAST_HTTP_4	**Nombre Historia de Usuario:** Especificar módulos a usar en Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: El software permitirá al usuario mostrar, habilitar y deshabilitar módulos.	
Observaciones: El sistema permitirá al usuario: • **Mostrar módulos:** La funcionalidad comienza cuando se selecciona la opción Configuración Global del panel a la izquierda, específicamente la pestaña Módulos. Se mostrará una lista con todos los módulos existentes en Apache y de ellos estarán marcados los que estén habilitados. • **Habilitar módulos:** Una vez mostrada la lista con todos los módulos existentes y de ellos marcados los que están habilitados, el usuario para habilitar otro solo tendrá que marcarlo. • **Deshabilitar módulos:** Una vez mostrada la lista con todos los módulos existentes y de ellos marcados los que están habilitados, el usuario para deshabilitar alguno solo tendrá que desmarcarlo.	
Prototipo de interfaz: Ver Anexo 17: Especificar módulos a usar en Apache 2.	

Historia de Usuario	
Número: HMAST_HTTP_5	**Nombre Historia de Usuario:** Especificar parámetros en el servidor principal de Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 3
Prioridad en Negocio: Media	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Se muestran y modifican algunos parámetros en el servidor principal de Apache 2.	
Observaciones: El software permitirá al usuario mostrar y modificar los siguientes parámetros. • **Dirección de correo del administrador:** Se refiere a la dirección de correo electrónico que se	

muestra cuando el servidor genera un error en la página. Se debe validar que su sintaxis sea correcta, en caso contrario se notificará al usuario.

- **Información sobre versión del servidor:** Se refiere a la información que se devuelve dentro de la cabecera HTTP que envía el servidor. Los posibles valores de menor a mayor información son: Prod, Min, Os y Full.
- **Fichero de registro de errores:** Hace referencia a la ubicación del fichero de registro de errores en las consultas. Se debe validar que la ruta sea de un fichero y no de un directorio.

Prototipo de interfaz: Ver Anexo 18: Especificar parámetros en el servidor principal de Apache 2.

Historia de Usuario	
Número: HMAST_HTTP_6	**Nombre Historia de Usuario:** Especificar parámetros en el MPM prefork de Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 3
Prioridad en Negocio: Media	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana

Descripción: Permitirá mostrar y modificar las parámetros de configuración pertenecientes al MPM *prefork*. Este módulo crea un grupo de procesos hijos para servir peticiones. Cada proceso hijo tiene un sólo hilo, por lo que si Apache inicia 20 procesos hijos, puede servir 20 peticiones simultáneamente. En el caso de que ocurra un error grave y un proceso hijo muera, sólo se pierde una petición: la que está atendiendo el hijo que acaba de cesar la actividad. El máximo y el mínimo de procesos hijos son parámetros configurables, permitiendo adaptar Apache a las capacidades de la PC servidora y también fijarle un máximo de carga. Dentro de estos parámetros máximo y mínimo el servidor genera más o menos procesos en función de la carga de trabajo actual.

Observaciones: El software permitirá al usuario mostrar y modificar los siguientes parámetros, especificando en cada uno solo números enteros positivos:

- **Número de procesos hijos que se crean al iniciar Apache.**
- **Mínimo número de procesos hijos inactivos.** Un proceso inactivo o en espera es aquel que no está atendiendo ninguna petición.

- **Máximo número de procesos hijos inactivos:** Si hay más procesos hijos en espera que el número especificado en este parámetro, entonces el proceso padre elimina el exceso de procesos para liberar recursos y optimizar el uso de los mismos. El máximo número de procesos hijos inactivos siempre tiene que ser mayor que el mínimo.
- **Número máximo de procesos hijos que pueden crearse:** Debe tenerse en cuenta que establecer un número muy alto puede colapsar la máquina, por lo que se debe ajustar cuidadosamente a la memoria disponible y capacidades del servidor.
- **Número máximo de peticiones que un proceso hijo atenderá durante su existencia:** Pasado este número, el proceso hijo se eliminará. Las peticiones ilimitadas es para valor 0 y otra cantidad sería para valores mayores o iguales que 1.

Prototipo de interfaz: Ver Anexo 19: Especificar parámetros en el MPM prefork de Apache 2.

Historia de Usuario	
Número: HMAST_HTTP_7	**Nombre Historia de Usuario:** Especificar parámetros para el manejo de conexiones en Apache 2.
Modificación de Historia de Usuario Número: 1	
Usuario: Nurisel Palma Pérez	**Iteración Asignada:** 3
Prioridad en Negocio: Media	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: El software permitirá al usuario mostrar y modificar parámetros para el manejo de conexiones en Apache 2.	

Observaciones: Se podrán modificar los parámetros:

- **Tiempo que Apache esperará antes de cerrar la conexión:** Se especifica con un número dado en segundos. Debe ser un número entero positivo.
- **Estado de la utilización de conexiones HTTP persistentes:** Activa o desactiva la utilización persistente de conexiones TCP en Apache. Si está activa da instrucciones a Apache para que utilice conexiones persistentes y, de ese modo, se pueda utilizar una sola conexión TCP para varias transacciones.
- **Tiempo que el servidor esperará peticiones subsiguientes en conexiones persistentes:**

Limita el número de segundos que Apache debe esperar una solicitud posterior antes de cerrar la conexión. Una vez que se recibe la solicitud, se aplica el tiempo de espera especificado anteriormente. Debe ser un número entero positivo.

- **Número de solicitudes permitidas por conexión:** Limita el número de solicitudes permitidas por conexión. La cantidad es ilimitada para valor 0 y otra cantidad sería para valores mayores o iguales que 1. Debe ser un número entero positivo.

Prototipo de interfaz: Ver Anexo 20: Especificar parámetros para el manejo de conexiones en Apache 2.

2.3 Arquitectura del módulo

En el desarrollo del módulo se hace uso de la arquitectura propuesta por la herramienta HMAST pero el alcance del desarrollo del mismo no incluye la interfaz visual, por lo que no se implementará la capa de Presentación. En las capas de Aplicación, Dominio y Persistencia se inserta un paquete con todo lo referente al módulo, como se aprecia en la Figura 3. El paquete correspondiente a la capa de Infraestructura Transversal contendrá todas las clases que sean reutilizables.

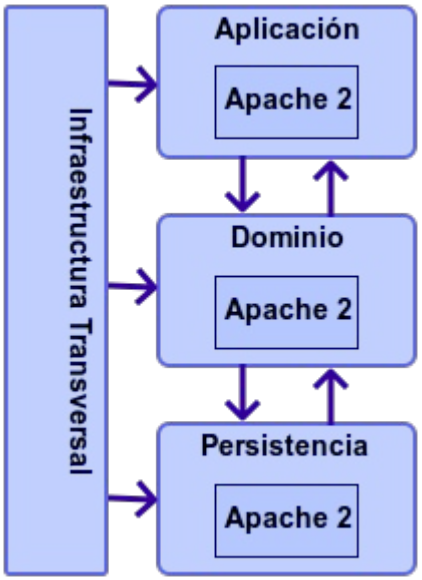


Figura 3: Arquitectura del módulo propuesto.

2.4 Diagrama de paquetes

Para la representación del módulo basada en la organización de paquetes y sus elementos, se diseñó el diagrama de paquetes tomando como marco de referencia la arquitectura propuesta. Existe un paquete representando cada capa y dentro de este uno llamado *apache2* que contiene todo lo referente al módulo. Las funcionalidades de todo el módulo están categorizadas en tres grupos: *GlobalConfiguration* que contiene todo lo referente al servidor principal, el MPM *prefork* y las conexiones del servidor; *VirtualHost* que es responsable de todos los parámetros de gestión de los hosts virtuales y *ApacheServer* que posee todas las funcionalidades para el manejo de los estados de Apache 2.

En las siguientes figuras, Figura 4, Figura 5, Figura 6 y Figura 7 se representan las diferentes partes del diagrama de paquetes perteneciente a la historia de usuario *Especificar parámetros para el manejo de conexiones en Apache 2,* perteneciente a *GlobalConfiguration*. A cada grupo corresponde un flujo de acciones similares independientemente de su funcionalidad.

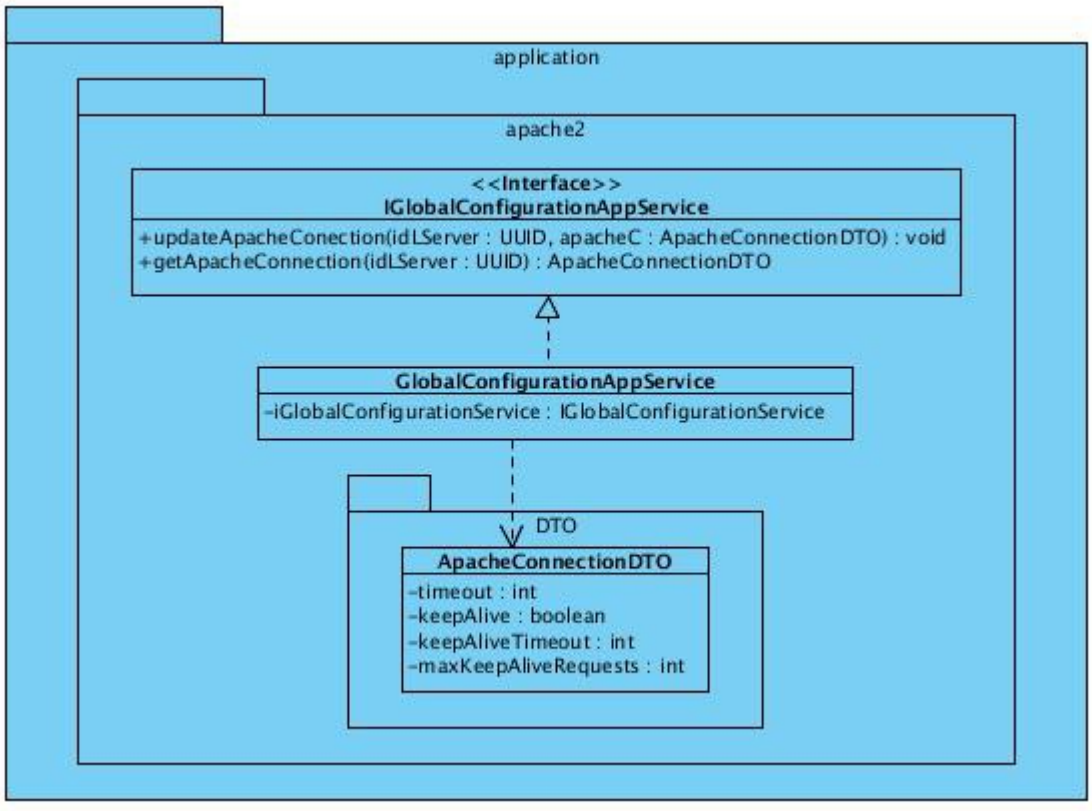


Figura 4: Diagrama de paquetes correspondiente a la capa de Aplicación.

En la figura Figura 4 se representa la capa de Aplicación, en el paquete *apache2* existe una interface *IGlobalConfigurationAppService* que define los métodos que serán llamados en un futuro desde Presentación y una clase *GlobalConfigurationAppService* que los implementa. En *apache2* se encuentra también un paquete *DTO* que contiene el objeto de transferencia de datos *ApacheConnectionDTO* que será enviado desde y hacia la capa de Presentación y del cual depende *GlobalConfigurationAppService.* Esta última es la encargada de realizar las conversiones de DTO a entidad y de entidad a DTO.

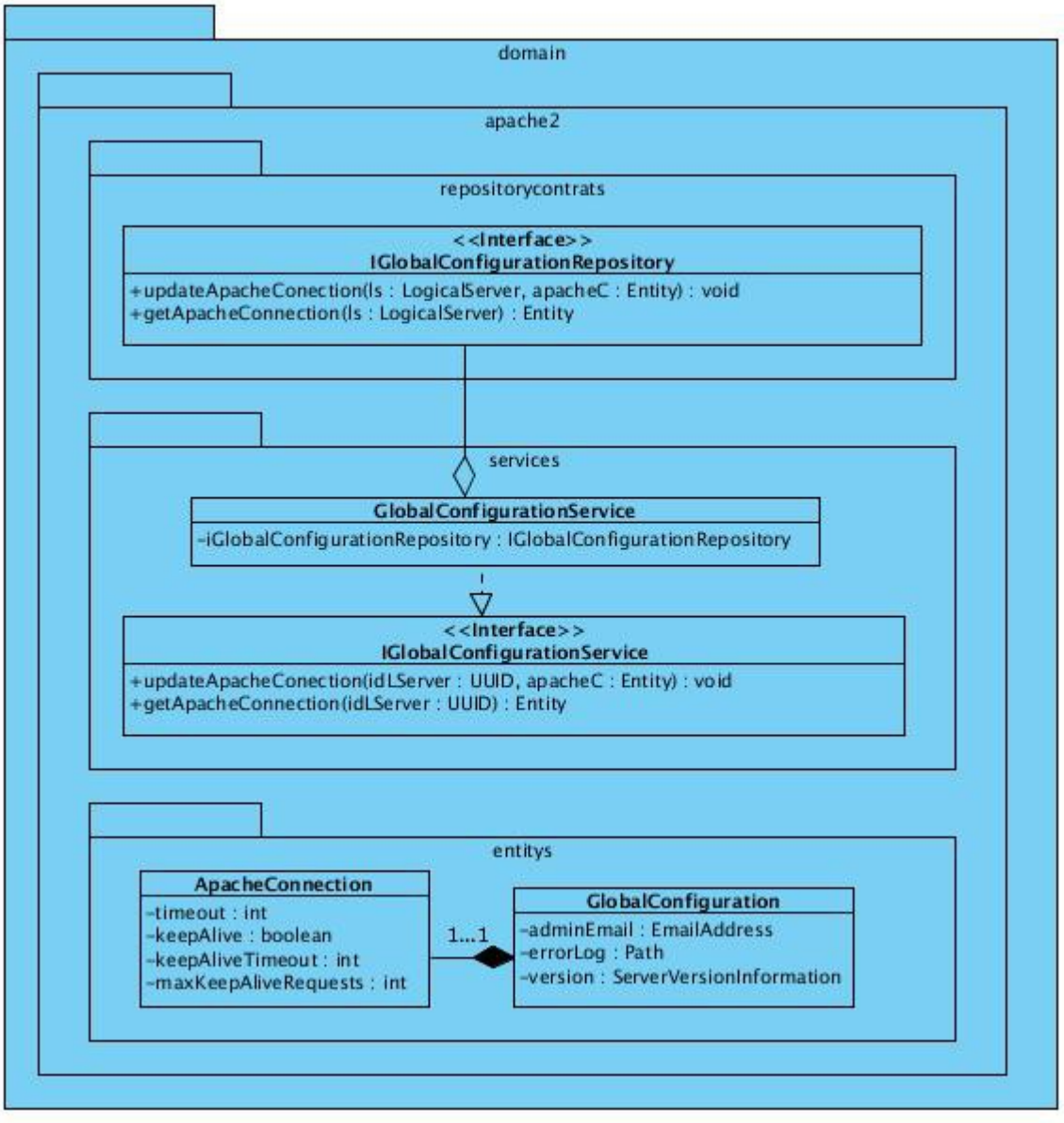


Figura 5: Diagrama de paquetes correspondiente a la capa de Dominio.

Las capas Aplicación y Dominio se relacionan debido a que *GlobalConfigurationAppService* posee una relación de agregación con *IGlobalConfigurationService,* que se realiza mediante una inyección de dependencia. La capa de Dominio está representada por la Figura 5 y contiene dentro del paquete *apache2* tres paquetes llamados *entitys*, *services* y *repositorycontrats*.

- *entitys:* contiene las entidades *ApacheConnection* y *GlobalConfiguration.* En cada una de ellas se realizan validaciones atómicas para cada atributo.
- *services:* se realizan las validaciones que no se realizaron en las entidades. Contiene la interface *IGlobalConfigurationService* que define las funcionalidades y *GlobalConfigurationService* que las implementa. *GlobalConfigurationService* posee una relación de agregación con *IGlobalConfigurationRepository* que se realiza mediante una inyección de dependencia.
- *repositorycontrats:* define los contratos de repositorio en *IGlobalConfigurationRepository* pero no los implementa.

El paquete *apache2* correspondiente a la capa de Persistencia, como se aprecia en la Figura 6, contiene otro paquete llamado *repository*, en el cual se encuentra la clase *GlobalConfigurationRepository* que es la encargada de realizar la persistencia de los datos implementando los contratos de repositorio definidos en *IGlobalConfigurationRepository.* La clase *GlobalConfigurationRepository* posee una relación de dependencia con la entidad *GlobalConfiguration* debido a que contiene un mapa que almacena objetos que son instancias de la misma.

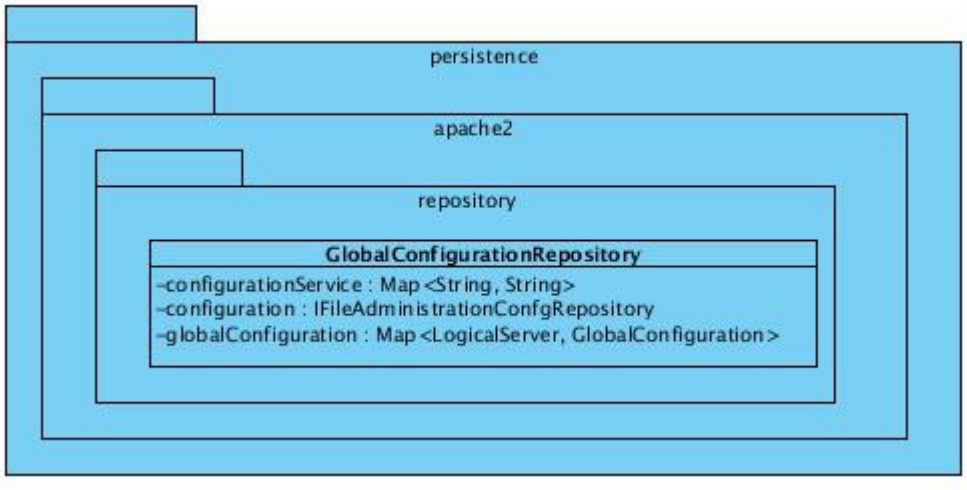


Figura 6: Diagrama de paquetes correspondiente a la capa de Persistencia.

La Figura 7 representa el diagrama de paquetes correspondiente a la capa de Infraestructura Transversal,

que contiene la entidad *Entity* la cual es padre de todas las entidades definidas en Dominio. En esta, en caso necesario se incluyen además clases que sean reutilizables por otras capas.

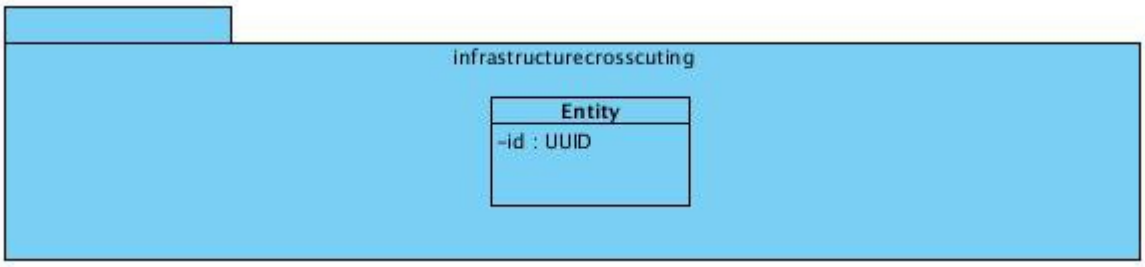


Figura 7: Diagrama de paquetes correspondiente a la capa de Infraestructura Transversal.

2.5 Patrones de diseño empleados

Los patrones de diseño son principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares [33].

2.5.1 Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresado en forma de patrones. Para el desarrollo del módulo se hace uso de los siguientes patrones GRASP:

Creador: Se pone de manifiesto en la capa de Aplicación cuando se realiza la conversión de clases de tipo DTO a entidades, pues se crean instancias de las entidades debido a que las clases DTO contienen los datos de inicialización de estas. Igualmente se manifiesta en la conversión de entidades a DTO.

Experto: La principal aplicación de este patrón es que cada capa tiene un objetivo y una responsabilidad asignada, siendo solo ella la encargada de implementarla.

Bajo Acoplamiento: Las inyecciones de dependencias proporcionadas por el framework Spring, permiten el uso de este patrón en el sistema. Los objetos no crean o buscan sus dependencias sino que estas son dadas al objeto.

Alta Cohesión: La cohesión es una medida de cuán enfocadas están las responsabilidades de una clase. Se manifiesta en el módulo en la parte de las validaciones, donde se realizaron en las entidades validaciones relacionadas con sus atributos y las que ya no dependían de ella se realizaron en servicio,

garantizando así no hacer un trabajo enorme.

Conclusiones parciales

En el presente capítulo a partir del estudio realizado en el Capítulo 1 y del funcionamiento de Apache 2, se definieron las características y funcionalidades del sistema propuesto. Con el empleo de la metodología de desarrollo de software SXP, se detallaron los artefactos generados en el desarrollo como son la Lista de Reserva del Producto, el Plan de Liberación y las Historias de Usuario. Partiendo de los requisitos para la implementación de un módulo para HMAST, se propuso la arquitectura a emplear, tomándose esta como base para el diseño del diagrama de paquetes. Tomando como referencia el diagrama diseñado se realizó una descripción del empleo de los patrones de diseño.

Capítulo 3: Implementación y pruebas.

Los procesos de desarrollo de software implican la realización de una serie de actividades que garanticen la calidad del mismo. Es por ello que primeramente se realiza la implementación y posterior a esta se define una etapa de pruebas con el objetivo de mostrar errores no detectados hasta entonces. En el presente capítulo se definen las Tareas de Ingeniería como apoyo al proceso de implementación y se describen las pruebas realizadas para la verificación de la calidad del módulo implementado.

3.1 Tareas de Ingeniería

Las Tareas de Ingeniería se generan como artefactos en la metodología SXP, facilitando el entendimiento en el proceso de implementación debido a que organizan el mismo con la definición de actividades asociadas a las Historias de Usuario. A continuación se definen las Tareas de Ingeniería correspondientes a cada una de las Historias de Usuario especificadas en el capítulo anterior.

Tareas de Ingeniería para Gestionar hosts virtuales asignados a Apache 2.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HMAST_HTTP_1
Nombre Tarea: Verificar funcionamiento de los hosts virtuales basados en IP y los hosts virtuales basados en nombre.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se verifica el funcionamiento de los hosts virtuales basados en IP y los hosts virtuales basados en nombre realizando pruebas en los ficheros de configuración. Las especificaciones para cada host virtual se hacen mediante la directiva contenedora <*VirtualHost direcciónIP:puerto*>, donde en una misma directiva pueden existir varias combinaciones *direcciónIP:puerto*. Para los hosts virtuales basados en nombre, el argumento *direcciónIP:puerto* es el mismo especificado en la directiva *NameVirtualHost*. Para el caso de la combinación de todas las direcciones IP con todos los puertos, cuando se adiciona más de un host virtual no es necesario poner la directiva *NameVirtualHost*.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HMAST_HTTP_1
Nombre Tarea: Probar acciones de habilitar y deshabilitar hosts virtuales.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se prueban las acciones de habilitar y deshabilitar hosts virtuales. Los hosts virtuales disponibles se encuentran en */etc/apache2/sites-available* y de ellos los que están activos en */etc/apache2/sites-enabled*. Las acciones de habilitar y deshabilitar no están directamente relacionadas con el fichero de configuración, sino que se realizan en la consola con los comandos *a2ensite nombre_host* y *a2dissite nombre_host* respectivamente.	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** HMAST_HTTP_1
Nombre Tarea: Probar acción de adicionar un host virtual.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Para la adición de un host virtual se debe crear un fichero en el directorio */etc/apache2/sites-available* que lleve por nombre el introducido por el usuario.	

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** HMAST_HTTP_1
Nombre Tarea: Verificar directivas de configuración de los hosts virtuales.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se verifican directivas de configuración de los hosts virtuales, como son DocumentRoot, ServerName, ServerAlias, SSLEngine, SSLCertificateFile, SSLCertificateKeyFile, ErrorLog, ServerAdmin, ServerSignature, DirectoryIndex, Alias, <Directory>, <Files>, Allow, Deny y Order.	

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** HMAST_HTTP_1
Nombre Tarea: Probar la clase InetAddress de Java.	

Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se realizan pruebas de la clase InetAddress de Java la cual se especializa en la obtención de las direcciones IP asignadas a una máquina dada.	

Tarea de Ingeniería para Manejar estados de Apache2.

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** HMAST_HTTP_2
Nombre Tarea: Probar manejo de estados en el servidor Apache 2.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se realizan acciones para el manejo de estados en el servidor Apache 2. Estas no están directamente relacionadas con el fichero de configuración, sino que se realizan en la consola con los siguientes comandos: • Instalar: *apt-get install apache2* • Iniciar: *service apache2 start* • Reiniciar: *service apache2 restart* • Detener: *service apache2 stop* • Recargar: *service apache2 reload*	

Tarea de Ingeniería para Modificar puertos para las conexiones en Apache 2.

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** HMAST_HTTP_3
Nombre Tarea: Probar configuración de puertos de escucha en Apache 2.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se realizan pruebas de la configuración de los puertos de escucha en Apache 2, la cual se encuentra en el fichero */etc/apache2/ports.conf*. El puerto TCP que usa Apache para la conexión se especifica mediante la directiva Listen. La sintaxis de la directiva es *Listen [direcciónIP:]puerto*, donde	

direcciónIP es un parámetro opcional, debido a que cuando no se pone significa que es para todos los puertos por los que esté escuchando Apache 2.

Tarea de Ingeniería para Especificar módulos a usar en Apache 2.

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** HMAST_HTTP_4
Nombre Tarea: Probar acciones de habilitar y deshabilitar módulos de Apache 2.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se prueban acciones de habilitar y deshabilitar módulos de Apache 2. Todos los módulos disponibles en Apache se encuentran en /etc/apache2/mods-available y de ellos los que están habilitados en /etc/apache2/mods-enabled. Habilitar y deshabilitar no están directamente relacionadas con el fichero de configuración, sino que se realizan en la consola con los comandos *a2enmod nombre_módulo* y *a2dismod nombre_módulo* respectivamente.	

Tarea de Ingeniería para Especificar parámetros en el servidor principal de Apache 2.

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** HMAST_HTTP_5
Nombre Tarea: Verificar funcionamiento de directivas en el servidor principal de Apache 2.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se verifica el funcionamiento de directivas en el servidor principal de Apache 2. Esta configuración se realiza en el fichero de configuración principal */etc/apache2/apache2.conf*. Las directivas son ServerAdmin, ServerTokens y ErrorLog.	

Tarea de Ingeniería para Especificar parámetros en el MPM prefork de Apache 2.

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** HMAST_HTTP_6
Nombre Tarea: Probar funcionamiento de directivas en el MPM prefork de Apache 2.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	

Descripción: Se verifica el funcionamiento de directivas en el MPM *prefork* de Apache 2. Su configuración se realiza en el fichero de configuración principal */etc/apache2/apache2.conf* específicamente dentro de la directiva de bloque <IfModule mpm_prefork_module></IfModule>. Las directivas son StartServers, MinSpareServers, MaxSpareServers, MaxClients y MaxRequestsPerChild.

Tarea de Ingeniería para Especificar parámetros para el manejo de conexiones.

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** HMAST_HTTP_7
Nombre Tarea: Probar funcionamiento de directivas para el manejo de conexiones en Apache 2.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,2
Programador Responsable: Nurisel Palma Pérez	
Descripción: Se verifica el funcionamiento de directivas para el manejo de conexiones en Apache 2. Las directivas son Timeout, KeepAlive, KeepAliveTimeout y MaxKeepAliveRequests.	

Una vez definidas las Tareas de Ingeniería como guía y apoyo al proceso de implementación y culminado este, se realizan las pruebas correspondientes al módulo implementado.

3.2 Definición de las pruebas a aplicar

La realización de pruebas es la tarea de demostrar que un programa realiza las funciones para las cuales fue construido. Las pruebas no aseguran la ausencia de errores, sino que están orientadas a demostrar que existen defectos en el software. El diseño de casos de prueba se basa en diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo.

La estrategia para la prueba del software se puede ver como una espiral donde inicialmente la prueba de unidad analiza la codificación, posteriormente la prueba de integración que se basa en el diseño, después la prueba de validación que se centra en el análisis de requisitos y por último la prueba del sistema para la ingeniería del mismo, como se aprecia en la Figura 8. La prueba de unidad analiza cada unidad del software (cada módulo individualmente), asegurando que funciona adecuadamente como una unidad, haciendo un uso intensivo del método de prueba de caja blanca, ejercitando caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores. La prueba de integración posee como foco de atención el diseño y la construcción de la arquitectura del software. En la prueba de validación se validan los requisitos establecidos como parte del análisis de

requisitos del software, comparándolos con el sistema que ha sido construido. Finalmente en la prueba del sistema se prueba como un todo el software [34].

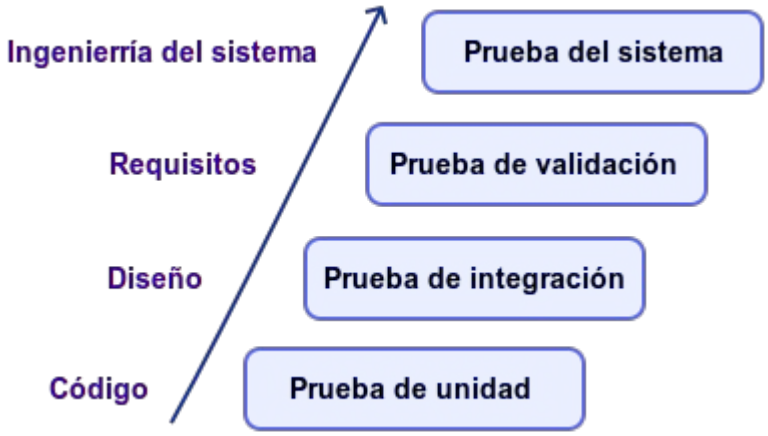


Figura 8: Etapas en la prueba del software

En el presente trabajo de diploma para la realización de pruebas, como el desarrollo del módulo no incluye la integración a otros y no posee interfaz visual, se selecciona la aplicación de las pruebas unitarias, por tanto se aplicará el método caja blanca y específicamente la técnica del camino básico.

La prueba de caja blanca, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que [35]:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

3.3 Realización de pruebas al código

Para verificar la calidad del módulo implementado se realizaron pruebas haciendo uso de la técnica del camino básico, que fue aplicada en todas las funcionalidades. A continuación se muestran cada uno de los pasos realizados para la aplicación de la misma en la funcionalidad que posee la responsabilidad de adicionar un host virtual.

En esta se tiene en cuenta todo el código correspondiente a la funcionalidad (el código de todas las capas). Primeramente se divide el código en bloques, como se muestra en la Tabla 2 y posteriormente, en función de la numeración obtenida, se representa el flujo de control lógico mediante la notación del grafo de flujo, que se aprecia en la Figura 9.

```
public void addVirtualHost(UUID idLogicalServer, VirtualHostNewDTO virtualHostDTO) throws ELogicalServerNotFound, ELogicalServerNotExist, UnknownHostException, EInvalidPath, InvalidAddress, EFieldEmpty, IOException, InterruptedException, JSchException, EInvalidFileRoute, EPortOutOfRange, SftpException, EInvalidDirectoryRoute, EInvalidPort, EExistVirtualHost, EPortIsNotANumber, ECantNotDeleteFile, EPortInUse, EInvalidAccessList, EInvalidDirectoryAliasList, ENotExistIpAddress, ERepeatedElements, EInvalidDefaultPortSSL, EListEmpty, EInvalidFileName, EInvalidSubDomain{
```

```
1       VirtualHostTempDTO temp=new VirtualHostTempDTO(null, null, null, null);
        int position=ExistVirtualHost(idLogicalServer, virtualHostDTO.getId());
```

```
2 y 3   if(position!=-1 && position!=-2){
```

```
4           temp=listVirtualHost.get(idLogicalServer).get(position);
        }
```

```
5               VirtualHostDTO vhost=new VirtualHostDTO(virtualHostDTO.getFileName(),
    virtualHostDTO.getIpPort(), virtualHostDTO.getDirectoryRoot(),
    virtualHostDTO.getHostName(), virtualHostDTO.isEnabledSSLsupport(),
    virtualHostDTO.getKeyFile(), virtualHostDTO.getCertificateFile(),
    virtualHostDTO.getErrorLog(), virtualHostDTO.getAdminEmail(),
    virtualHostDTO.getInformation(), temp.getAccess(), temp.getDirectoryAlias(),
    temp.getDirectoryIndex(), virtualHostDTO.getHostAlias(), idLogicalServer);
        VirtualHost virtualHost=convertVirtualHostDTO_Entity(vhost);
                                                                LogicalServer
    logicalServer=serversAdministrationService.getLogicalServer(idLogicalServer);
        if(verifyIpAddress(logicalServer, virtualHost.getIpPort())){
```

```
6           if(verifyDirectoryRoot(logicalServer, virtualHost.getDirectoryRoot())){
```

```
7               int count=portUsed(idLogicalServer, virtualHost.getIpPort());
                if(count==0){
```

```
8                   if(iVirtualHostRepository.searchLogicalServer(logicalServer)){
```

```
9            if(!iVirtualHostRepository.virtualHostNameExists(logicalServer, virtualHost)){
```

```
10                      iVirtualHostRepository.addVirtualHost(logicalServer, virtualHost);
```

11	}else{
	throw new EExistVirtualHost("Exist a virtual host with this name");
	}

12	}else{
	throw new ELogicalServerNotFound("Logical server not found");
	}

13	}else{
	throw new EPortInUse("There are "+count+" ports used by other service");
	}

14	}else{
	throw new EInvalidDirectoryRoute("Invalid directory route");
	}

15	}else{
	throw new ENotExistIpAddress("Not exist ip address in the PC");
	}

16	}

Tabla 2: Separación en bloques del código de adicionar un host virtual.

A partir del grafo de flujo de la Figura 9, que tiene 22 aristas y 16 nodos, se obtiene la complejidad ciclomática dando como resultado:

V(G) = Aristas - Nodos + 2

V(G) = 22 – 16 + 2

V(G) = 8

Del valor de la complejidad ciclomática se obtiene que existen 8 caminos independientes, que son:

- **Camino 1:** 1-2-3-4-5-6-7-8-9-10-16
- **Camino 2:** 1-2-3-4-5-6-7-8-9-11-16
- **Camino 3:** 1-2-3-4-5-6-7-8-12-16
- **Camino 4:** 1-2-3-4-5-6-7-13-16

- **Camino 5:** 1-2-3-4-5-6-14-16
- **Camino 6:** 1-2-3-4-5-15-16
- **Camino 7:** 1-2-3-5-6-7-8-9-10-16
- **Camino 8:** 1-2-5-6-7-8-9-10-16

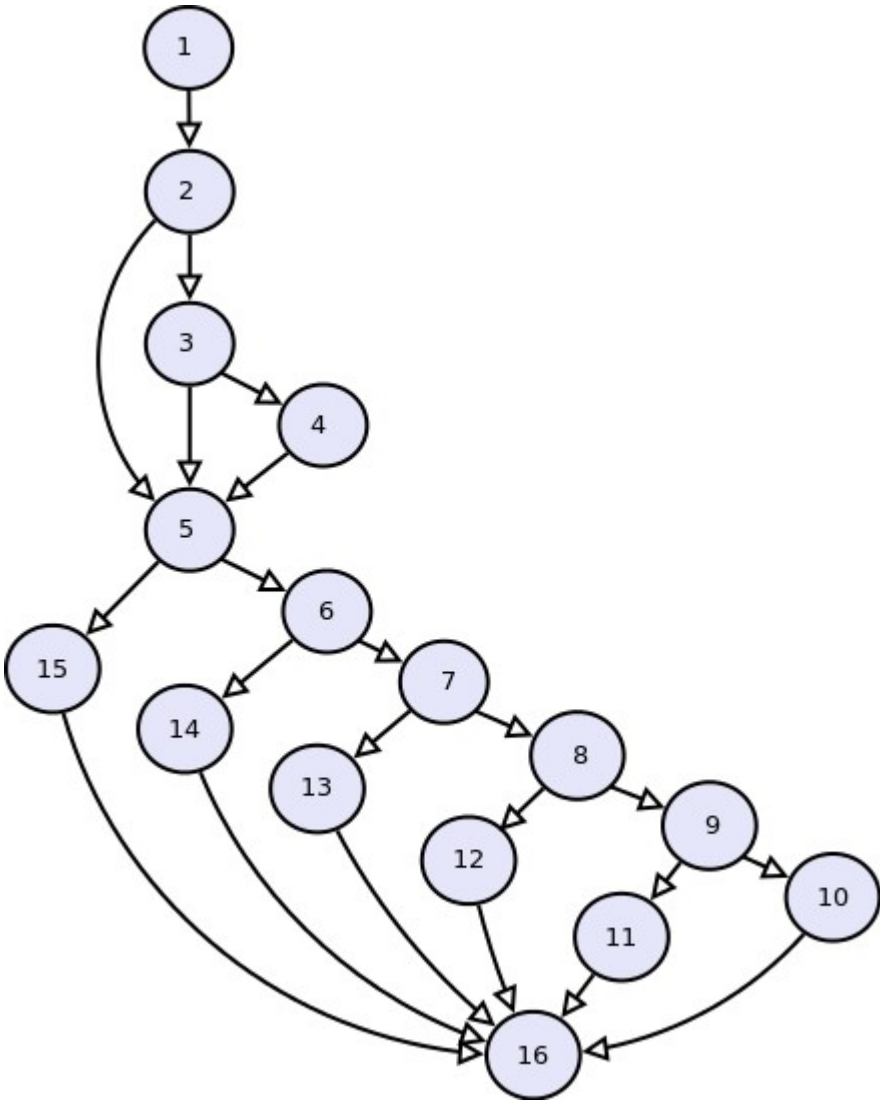


Figura 9: Grafo de flujo correspondiente a la funcionalidad adicionar host virtual.

Si se pueden diseñar pruebas que fuercen la ejecución de los caminos obtenidos se garantizará que se habrá ejecutado al menos una vez cada sentencia del código y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa. Por tanto, se diseñan casos de prueba para ser aplicados a cada camino

de forma tal que los parámetros de entrada cumplan con las posibles condiciones, como se muestra en las siguientes tablas.

Caso de Prueba para el Camino 1
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez
Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos: • *fileName*: newVirtualHost1 • *List<IpAddressPortNumDTO>*: 10.53.4.65:8080 • *directoryRoot*: /var/www • *hostName*: prueba1.uci.cu • *enabledSSLsupport*: on • *keyFile*: /etc/ssl/private/ssl-cert-snakeoil.key • *certificateFile*: /etc/ssl/certs/ssl-cert-snakeoil.pem • *errorLog*: /var/log/apache2/error.log • *adminEmail*: webmaster@uci.cu • *information*: email • *hostAlias*: null • *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b
Resultado esperado: Como ya existen atributos correspondientes al host virtual en el mapa temporal, se integran a los nuevos parámetros, todos los atributos son válidos, ya existe el servidor lógico en el mapa y no hay un host virtual con ese nombre, por lo que se realiza la adición.
Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 2
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez
Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos:

- *fileName*: newVirtualHost1
- *List<IpAddressPortNumDTO>*: 10.53.4.65:*
- *directoryRoot*: /var/www
- *hostName*: sitio2.uci.cu
- *enabledSSLsupport*: off
- *keyFile*: null
- *certificateFile*: null
- *errorLog*: /var/log/apache2/error.log
- *adminEmail*: webmaster@uci.cu
- *information*: off
- *hostAlias*: null
- *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b

Resultado esperado: Como ya existen atributos correspondientes al host virtual en el mapa temporal, se integran a los nuevos parámetros, todos los atributos son válidos, ya existe el servidor lógico en el mapa pero como ya existe un host virtual con ese nombre, se lanza una excepción.

Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 3
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez
Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos: • *fileName*: newVirtualHost3 • *List<IpAddressPortNumDTO>*: *:8080, 10.53.41.62:5412 • *directoryRoot*: /var/www • *hostName*: sitio2.uci.cu

- *enabledSSLsupport*: off
- *keyFile*: null
- *certificateFile*: null
- *errorLog*: /var/log/apache2/error.log
- *adminEmail*: webmaster@uci.cu
- *information*: off
- *hostAlias*: null
- *idLogicalServer*: 84cf6668-3556-422d-bfe0-5e1d93eedac9

Resultado esperado: Como ya existen atributos correspondientes al host virtual en el mapa temporal, se integran a los nuevos parámetros, todos los atributos son válidos, pero como no existe el servidor lógico en el mapa se lanza una excepción.

Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 4
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez

Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos:

- *fileName*: newVirtualHost4
- *List<IpAddressPortNumDTO>*: 10.53.4.65:320
- *directoryRoot*: /var/www
- *hostName*: sitio2.uci.cu
- *enabledSSLsupport*: off
- *keyFile*: null
- *certificateFile*: null
- *errorLog*: /var/log/apache2/error.log

- *adminEmail*: webmaster@uci.cu
- *information*: off
- *hostAlias*: null
- *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b

Resultado esperado: Como ya existen atributos correspondientes al host virtual en el mapa temporal, se integran a los nuevos parámetros, pero como uno de los puertos es usado por otro servicio se lanza una excepción.

Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 5
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez

Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos:

- *fileName*: newVirtualHost5
- *List<IpAddressPortNumDTO>*: 10.53.4.65:320
- *directoryRoot*: /var/www
- *hostName*: sitio2.uci.cu
- *enabledSSLsupport*: off
- *keyFile*: null
- *certificateFile*: null
- *errorLog*: /var/log/apache2/error.log
- *adminEmail*: webmaster@uci.cu
- *information*: off
- *hostAlias*: null
- *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b

Resultado esperado: Como ya existen atributos correspondientes al host virtual en el mapa temporal, se integran a los nuevos parámetros, pero como la ruta del directorio raíz no es correcta se lanza una excepción.
Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 6
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez
Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos: • *fileName*: newVirtualHost6 • *List<IpAddressPortNumDTO>*: 10.53.4.65:320 • *directoryRoot*: /var/www • *hostName*: sitio2.uci.cu • *enabledSSLsupport*: off • *keyFile*: null • *certificateFile*: null • *errorLog*: /var/log/apache2/error.log • *adminEmail*: webmaster@uci.cu • *information*: off • *hostAlias*: null • *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b
Resultado esperado: Como ya existen atributos correspondientes al host virtual en el mapa temporal, se integran a los nuevos parámetros, pero como la dirección IP no es una de las asignadas a la PC servidora se lanza una excepción.
Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 7
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez
Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos: • *fileName*: newVirtualHost7 • *List<IpAddressPortNumDTO>*: 10.53.4.65:320 • *directoryRoot*: /var/www • *hostName*: sitio2.uci.cu • *enabledSSLsupport*: off • *keyFile*: null • *certificateFile*: null • *errorLog*: /var/log/apache2/error.log • *adminEmail*: webmaster@uci.cu • *information*: off • *hostAlias*: null • *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b
Resultado esperado: No existen atributos correspondientes al host virtual en el mapa temporal debido a que se encuentra el servidor lógico en el mapa temporal pero no un host virtual con ese identificador, todos los atributos son válidos, ya existe el servidor lógico en el mapa y no hay un host virtual con ese nombre, por lo que se realiza la adición.
Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

Caso de Prueba para el Camino 8
Descripción de la prueba: Se verifica la adición de un host virtual.
Nombre de la persona que realiza la prueba: Nurisel Palma Pérez
Entrada: Se pasa como parámetro un VirtualHostNewDTO que tiene como atributos:

• *fileName*: newVirtualHost8 • *List<IpAddressPortNumDTO>*: 10.53.4.65:320 • *directoryRoot*: /var/www • *hostName*: sitio2.uci.cu • *enabledSSLsupport*: off • *keyFile*: null • *certificateFile*: null • *errorLog*: /var/log/apache2/error.log • *adminEmail*: webmaster@uci.cu • *information*: on • *hostAlias*: null • *idLogicalServer*: 24de9a6c-71d4-47c1-aefa-0cdd6480a19b
Resultado esperado: No existen atributos correspondientes al host virtual en el mapa temporal debido a que no se encuentra el servidor lógico en el mapa temporal, todos los atributos son válidos, ya existe el servidor lógico en el mapa y no hay un host virtual con ese nombre, por lo que se realiza la adición.
Resultado obtenido: Se ha recorrido el camino correcto obteniéndose el resultado esperado.

En los ocho casos de prueba se obtuvieron resultados satisfactorios, demostrándose por tanto:

- La ejecución de cada sentencia del código al menos una vez.
- La ejecución de cada condición en sus vertientes verdadera y falsa.
- La validez de las estructuras internas de datos.

3.4 Pruebas realizadas por el cliente

Con el fin de demostrar el cumplimiento del objetivo planteado y la satisfacción del cliente con el resultado obtenido, se realizaron pruebas para validar la calidad del módulo implementado. El propósito de estas es verificar si se cumple la adaptabilidad del módulo a las diferentes empresas e instituciones cubanas.

La norma cubana ISO/IEC 9126 elaborada por la Oficina Nacional de Normalización, plantea que la

calidad del producto del software se debe evaluar usando un modelo de calidad definido. El modelo de calidad para la calidad interna y externa categoriza los atributos en seis características: la funcionalidad, la confiabilidad, la usabilidad, la eficiencia, la mantenibilidad y la portabilidad. Esta última abarca atributos como son la adaptabilidad, instalabilidad, coexistencia, reemplazabilidad, portabilidad y conformidad. Específicamente en el módulo desarrollado se verificará la adaptabilidad. Según la norma, la adaptabilidad se define como la capacidad del producto de software de ser adaptado a los ambientes especificados sin aplicar acciones o medios de otra manera que aquellos suministrados con el propósito de que el software cumpla sus fines [36]. En este caso los ambientes especificados son los definidos en el epígrafe 1.6 Necesidades en las diferentes empresas e instituciones cubanas.

El cliente probó cada una de las funcionalidades mediante un *Main*[3] de pruebas, verificando que existen funcionalidades implementadas que permiten:

- Adicionar un host virtual pasando como parámetro todos sus atributos. Se pueden adicionar múltiples hosts virtuales a una misma combinación *direcciónIP:puerto,* debido al soporte existente para hosts virtuales basados en nombre y/o hosts virtuales basados en IP. De esta forma se realiza un menor consumo de direcciones IP y se aprovecha al máximo cada dirección asignada a la PC servidora.
- Actualizar el estado de los módulos en Apache 2. Se pueden activar y desactivar módulos en tiempo de ejecución de forma que solo estén habilitados los que sean realmente necesarios, sin sobrecargar al servidor innecesariamente.
- Actualizar los parámetros de configuración del MPM *prefork*. Se pueden configurar todos los parámetros correspondientes, permitiendo adaptar Apache 2 a las capacidades de la PC servidora y al objetivo que se desee lograr para el manejo de peticiones.

Dadas las funcionalidades existentes y el correcto funcionamiento de las mismas, se determina que el módulo sí puede ser adaptado a las necesidades de las diferentes empresas e instituciones cubanas.

Conclusiones parciales

Con el desarrollo del capítulo, aplicando la metodología de desarrollo de software SXP se generaron las Tareas de Ingeniería que sirvieron como base para la implementación del módulo diseñado en el capítulo

3 Proyecto de pruebas en java que posee solo una clase la cual contiene un método main en el cual se realizan llamadas a cada una de las funcionalidades.

anterior. Partiendo del alcance del desarrollo se determinó la prueba de unidad como la más indicada, utilizándose específicamente la técnica del camino básico del método caja blanca. Las pruebas realizadas permitieron determinar el correcto funcionamiento del módulo implementado así como analizar que el mismo es adaptable a las diferentes empresas e instituciones cubanas.

Conclusiones

Con la realización del presente trabajo de diploma se da cumplimiento a cada uno de los objetivos específicos propuestos:

- La caracterización de los servidores web y específicamente los de código abierto permitió la adquisición de los conocimientos necesarios para seleccionar Apache 2 como el más indicado a usar para satisfacer las necesidades de las diferentes empresas e instituciones cubanas.
- La aplicación del método Modelación propició el entendimiento del funcionamiento de Apache 2 para la definición de los requisitos y funcionalidades necesarios en el módulo a implementar y a partir de estos y de la arquitectura propuesta el diseño del diagrama de paquetes.
- El análisis y diseño del módulo definido permitió generar Tareas de Ingeniería correspondientes a cada Historia de Usuario que guiaran el proceso de implementación del módulo.
- La selección de las pruebas indicadas facilitó la determinación de la calidad del módulo y el cumplimiento del objetivo propuesto debido a que el módulo posee funcionalidades que le permiten ser adaptado a las diferentes empresas e instituciones cubanas.

Los resultados obtenidos han demostrado que el desarrollo de un módulo para la administración de Apache 2 en la herramienta HMAST puede garantizar la administración de forma que se asegure la adaptabilidad a las diferentes empresas e instituciones cubanas. El módulo facilita realizar asignaciones de hosts virtuales a direcciones IP de forma eficiente y con un menor consumo de las mismas, permite en tiempo de ejecución especificar el uso de los módulos realmente necesarios y ajustar los parámetros para el manejo de peticiones según el objetivo que posea la empresa. Además los administradores del servidor podrán autoprepararse y adquirir los conocimientos suficientes debido al abundante soporte de documentación que posee Apache 2.

Recomendaciones

El módulo implementado con la realización del presente trabajo de diploma posee las principales funcionalidades correspondientes a cada una de las tres secciones de configuración de Apache 2, por lo que se recomienda agregar para mejorar el funcionamiento del mismo, aspectos como:

- Especificación del MPM que se desee usar, así como la administración de los restantes MPM que trae Apache 2 por defecto, que son: *event* y *worker*.
- Desarrollo de una interfaz visual que permita la administración de Apache 2 mediante el uso de las funcionalidades implementadas en el módulo.

Referencias Bibliográficas

[1] Anón. Cubaminrex...Cuba en la Cumbre Mundial sobre la Sociedad de la Inform. [citado 5 octubre 2012]. Available from world wide web: <http://www.cubaminrex.cu/sociedad_informacion/cuba_si/Informatizacion.htm>.

[2] Castro Ruz, Fidel. Discurso pronunciado por el Comandante Fidel Castro Ruz, Primer Ministro del Gobierno Revolucionario, en el acto celebrado por la Sociedad Espeleológica de Cuba, en la Academia de Ciencias, el 15 de enero de 1960. enero 1960. [citado 19 octubre 2012]. Available from world wide web: <http://www.cuba.cu/gobierno/discursos/1960/esp/f150160e.html>.

[3] Paumier Samón, Ramón, Yoandy Pérez Villazón, y Abel Meneses Abad. Guía Cubana de Migración a Software Libre. [citado 24 noviembre 2012]. Available from world wide web: <http://es.scribd.com/doc/17779155/Guia-Cubana-Migracion-a-Software-Libre>.

[4] Alcalde Lancharro, Eduardo, y Jesús García Tomás. *Introducción a la Teleinformática.* [Madrid]: MCGRAW-HILL/INTERAMERICANA, 1993.

[5] Aya, Andrés. CAVELIER ABOGADOS/INTRODUCCIÓN AL TEMA DE LAS TELECOMUNICACIONES EN COLOMBIA-CONCEPTOS BÁSICOS. junio 2001. [citado 19 octubre 2012]. Available from world wide web: <http://www.cavelierabogados.com/econtent/Newsdetail.asp?ID=1237&IDCompany=7>.

[6] Ídem 4.

[7] Jose Gregorio. Servidor web. junio 2011. [citado 23 octubre 2012]. Available from world wide web: <http://www.slideshare.net/josegregoriob/servidor-web-8451426>.

[8] Anón. Qué es un servidor web ? [citado 1 octubre 2012]. Available from world wide web: <http://www.misrespuestas.com/que-es-un-servidor-web.html>.

[9] Anón. Web Server Survey | Netcraft. [citado 18 febrero 2013]. Available from world wide web: <http://news.netcraft.com/archives/category/web-server-survey/>.

[10] Anón. Usage Statistics and Market Share of Web Servers for Websites, March 2013. [citado 1 marzo 2013]. Available from world wide web: <http://w3techs.com/technologies/overview/web_server/all>.

[11] Nedelcu, Clément. Nginx HTTP Server. Aanchal Kumar, julio 2010.

[12] Anón. Módulos de MultiProcesamiento (MPMs) - Servidor HTTP Apache. [citado 13 noviembre 2012].

Available from world wide web: <http://httpd.apache.org/docs/2.2/es/mpm.html>.

[13] Kabir, Mohammed. La Biblia Del Servidor Apache 2. Anaya Multimedia, 2003 Available from world wide web: <http://es.scribd.com/doc/24039203/Mohammed-Kabir-La-Biblia-Del-Servidor-Apache-2>.

[14] Ferri-Benedetti, Fabrizio. Nginx - Descargar. [citado 11 diciembre 2012]. Available from world wide web: <http://nginx.softonic.com/>.

[15] Ídem 11.

[16] Ídem 14.

[17] Ídem 11.

[18] Anón. nginx. [citado 25 marzo 2013]. Available from world wide web: <http://nginx.org/en/>.

[19] Anón. NginxEs. [citado 7 noviembre 2012]. Available from world wide web: <http://wiki.nginx.org/NginxEs>.

[20] Anón. Usemos Linux: NGINX: una interesante alternativa a Apache. junio 2012. [citado 7 noviembre 2012]. Available from world wide web: <http://usemoslinux.blogspot.com/2012/06/nginx-una-interesante-alternativa.html>.

[21] Anón. February 2013 Web Server Survey | Netcraft. [citado 27 febrero 2013]. Available from world wide web: <http://news.netcraft.com/archives/2013/02/01/february-2013-web-server-survey.html#more-7865>.

[22] Ídem 13.

[23] Anón. Servidor http Apache.

[24] González Rodríguez, Leover Armando. Alternativas para el desarrollo de Aplicaciones Web. junio 2011.

[25] Vargas Ayulo, Cesar, Wilfredo Alvites Quiroz, y Diego Chavez Estrada. Desarrollo de Aplicaciones Web II: noviembre 2012. noviembre 2012. [citado 26 marzo 2013]. Available from world wide web: <http://webpegasus.blogspot.com/2012_11_01_archive.html>.

[26] Anón. Lenguaje de programación / definición de Lenguaje de programación - Un lenguaje de programación es una técnica estándar de comunicaci. [citado 22 febrero 2013]. Available from world wide web: <http://www.diclib.com/lenguaje%20de%20programaci%C3%B3n/show/es/es_wiki_10/74813>.

[27] Anón. Características del lenguaje Java. [citado 22 febrero 2013]. Available from world wide web:

<http://www.iec.csic.es/criptonomicon/java/quesjava.html>.

[28] Contreras Martínez, José Luís, Ana García Domene, Daniel Martínez Espadas, y Begoña Morillas Guijarro. Práctica 4 - Herramienta CASE: Umbrello. [citado 26 marzo 2013]. Available from world wide web: <http://es.scribd.com/doc/86321835/Practica-4-Herramienta-CASE-Umbrello>.

[29] Anón. Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) por Visual Paradigm International Ltd. - reporte y descarga. [citado 26 marzo 2013]. Available from world wide web: <http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/>.

[30] Anón. Control de Versiones. [citado 26 marzo 2013]. Available from world wide web: <http://producingoss.com/es/vc.html>.

[31] Tain Domínguez, Yuliette, y Román Miguel Valdivia Genó. Augeas, propuesta tecnológica para la gestión de archivos de configuración de sistemas GNU/Linux. mayo 2009.

[32] Peñalver Romero, Gladys Marsi. MA-GMPR-UR2. Metodología ágil para proyectos de software libre. junio 2008.

[33] Olivares Rojas, Juan Carlos. Patrones de diseño.

[34] S. Pressman, Roger. *Ingeniería de software. Un enfoque práctico.* Quinta edición.

[35] Ídem 34.

[36] Oficina Nacional de Normalización. INGENIERÍA DE SOFTWARE—CALIDAD DEL PRODUCTO—PARTE 1: MODELO DE LA CALIDAD (ISO/IEC 9126-1:2001, IDT). abril 2005.

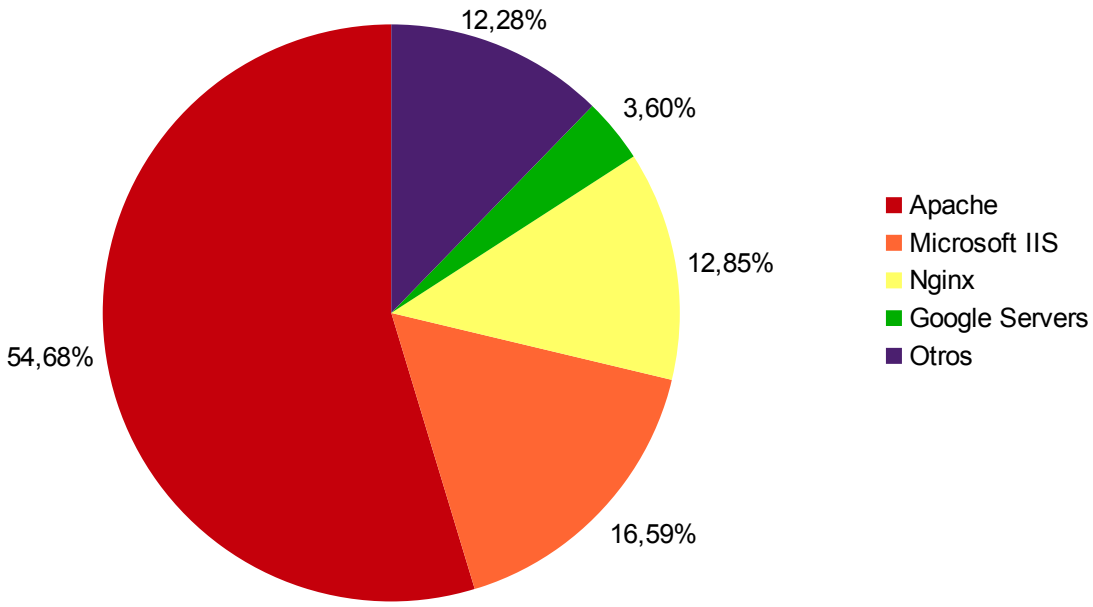
Bibliografía General

- Alcalde Lancharro, Eduardo, y Jesús García Tomás. *Introducción a la Teleinformática.* MCGRAW-HILL/INTERAMERICANA, 1993.
- Anón. Características. [citado 1 octubre 2012a]. Available from world wide web: <http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html>.
- Anón. Cubaminrex...Cuba en la Cumbre Mundial sobre la Sociedad de la Inform. [citado 5 octubre 2012b]. Available from world wide web: <http://www.cubaminrex.cu/sociedad_informacion/cuba_si/Informatizacion.htm>.
- Anón. Guía Breve de Servicios Web. [citado 19 octubre 2012c]. Available from world wide web: <http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb>.
- Anón. Índice de Directivas - Servidor HTTP Apache. [citado 24 mayo 2013d]. Available from world wide web: <http://httpd.apache.org/docs/2.0/es/mod/directives.html>.
- Anón. Metodologías tradicionales y metodologías ágiles. [citado 25 octubre 2012d]. Available from world wide web: <http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm>.
- Anón. ¿Qué es INFOSOC? octubre 2012. [citado 5 octubre 2012]. Available from world wide web: <http://www.mic.gov.cu/sitiomic/servlet/hinfosoc>.
- Castillo Arbelo, Reidiel, y Pablo Soria Acosta. Herramienta para la Migración y Administración de Servidores (HMAS). junio 2012.
- Díaz Ruiz, Olga. UCI: la mirada de un decenio. octubre 2012, 3.
- González, Maricarmen. Cuba avanza en la migración al software libre | Atenas. 2009. [citado 5 octubre 2012]. Available from world wide web: <http://www.atenas.cult.cu/node/6500>.
- Kabir, Mohammed. *La Biblia Del Servidor Apache 2.* Anaya Multimedia, 2003 Available from world wide web: <http://es.scribd.com/doc/24039203/Mohammed-Kabir-La-Biblia-Del-Servidor-Apache-2>.
- Mateu, Carles. *Desarrollo de aplicaciones web.* primera Fundació per a la Universitat Oberta de

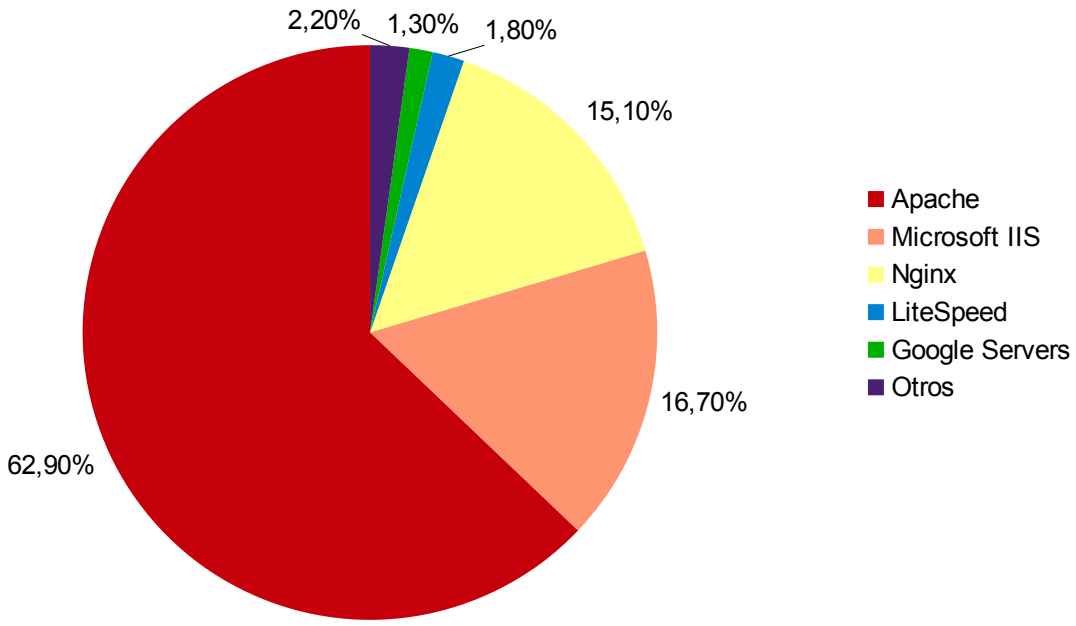
Catalunya, marzo 2004.

- Nedelcu, Clément. Nginx HTTP Server. Aanchal Kumar, julio 2010.
- Peñalver Romero, Gladys Marsi. MA-GMPR-UR2. Metodología ágil para proyectos de software libre. junio 2008.
- S. Pressman, Roger. *Ingeniería de software. Un enfoque práctico.* Quinta edición.

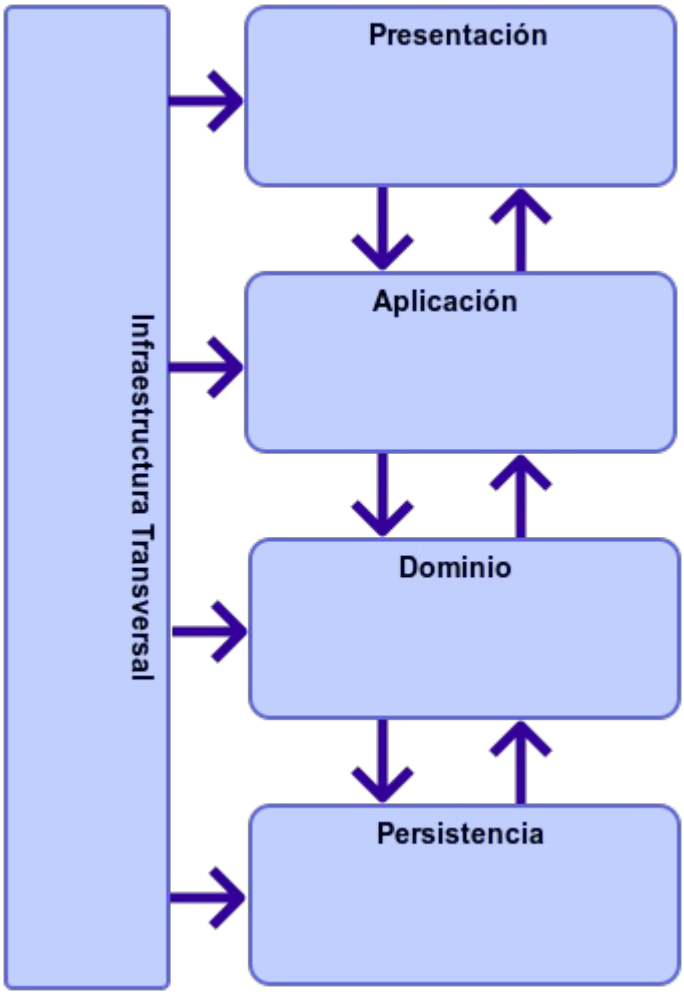
Anexos

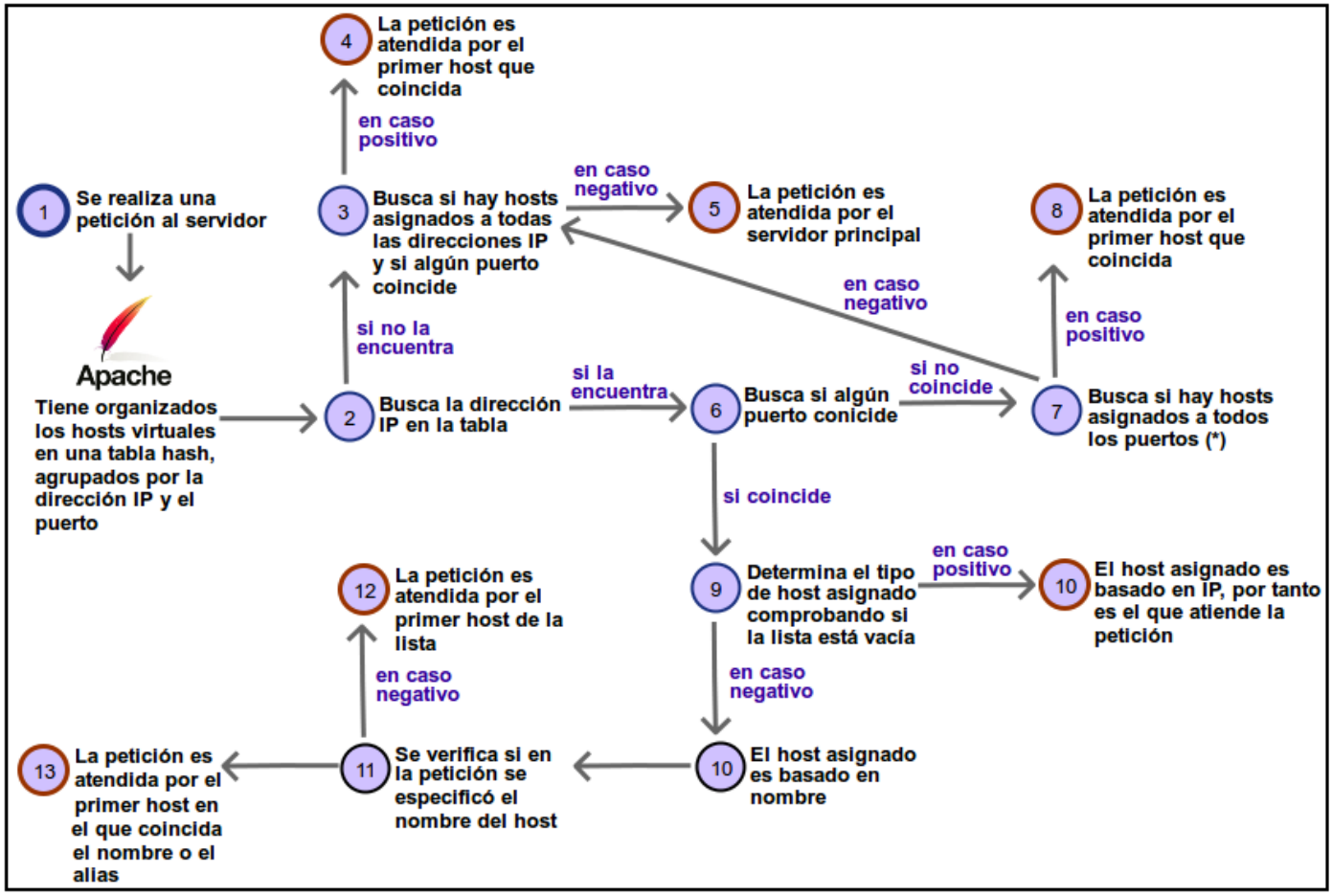


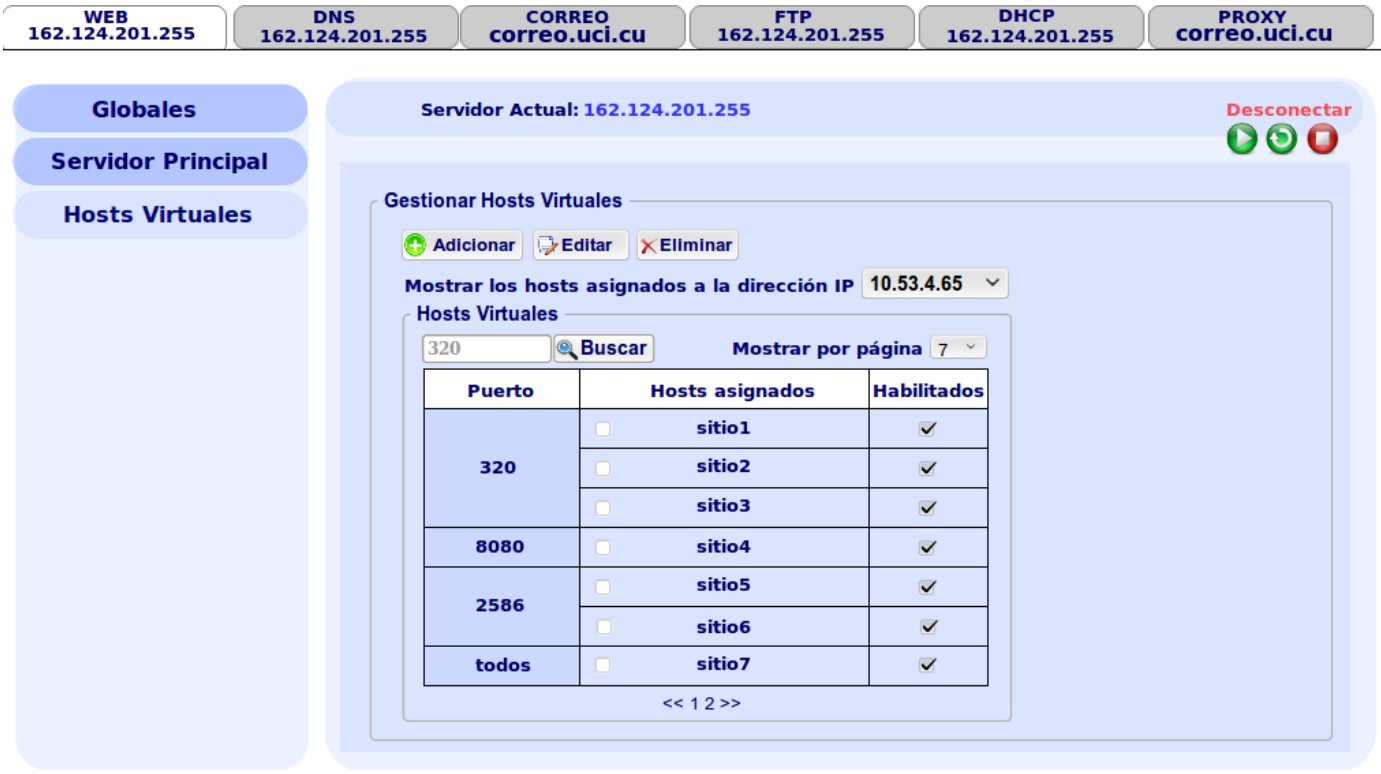
Anexo 1: Estadísticas del uso de servidores web según la compañía Netcraft.



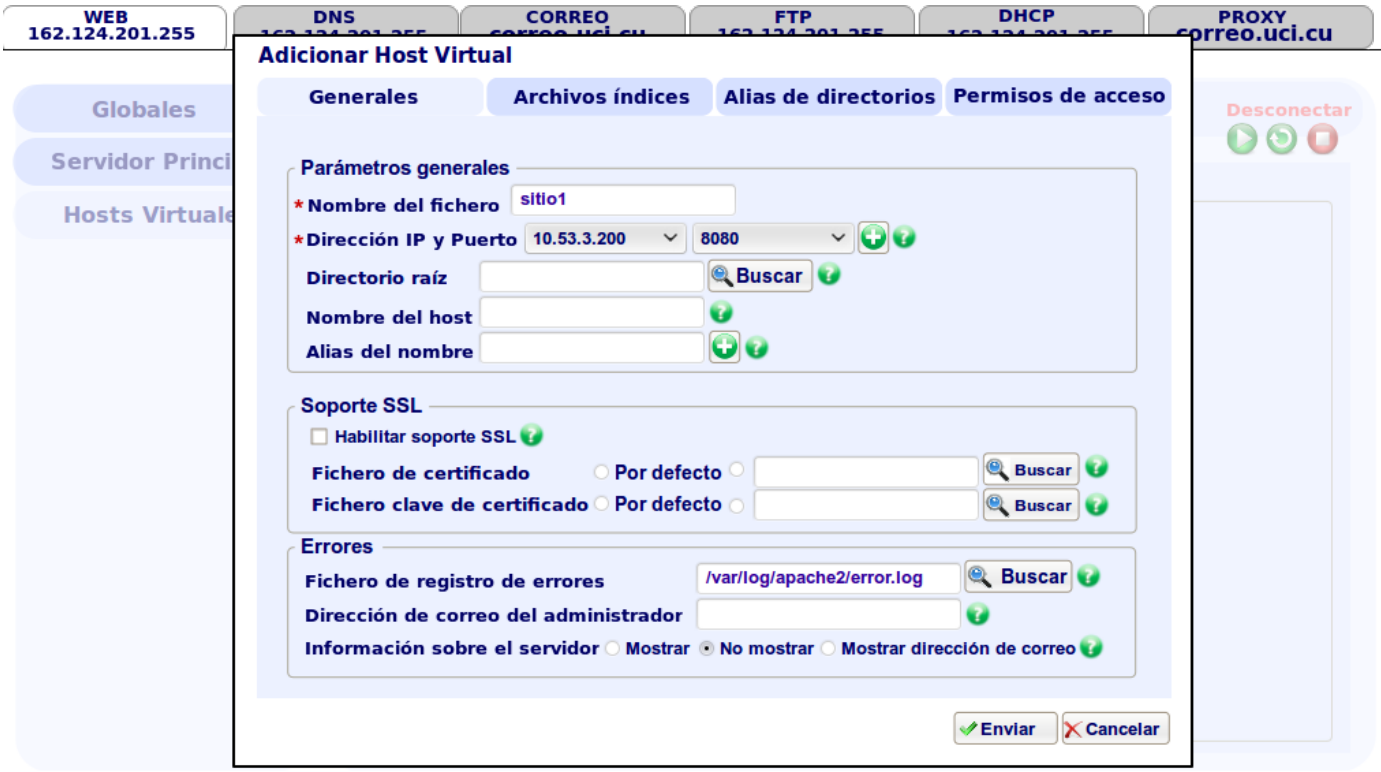
Anexo 2: Estadísticas del uso de servidores web según W3Techs.


Presentación
Aplicación
Dominio
Persistencia
Infraestructura Transversal

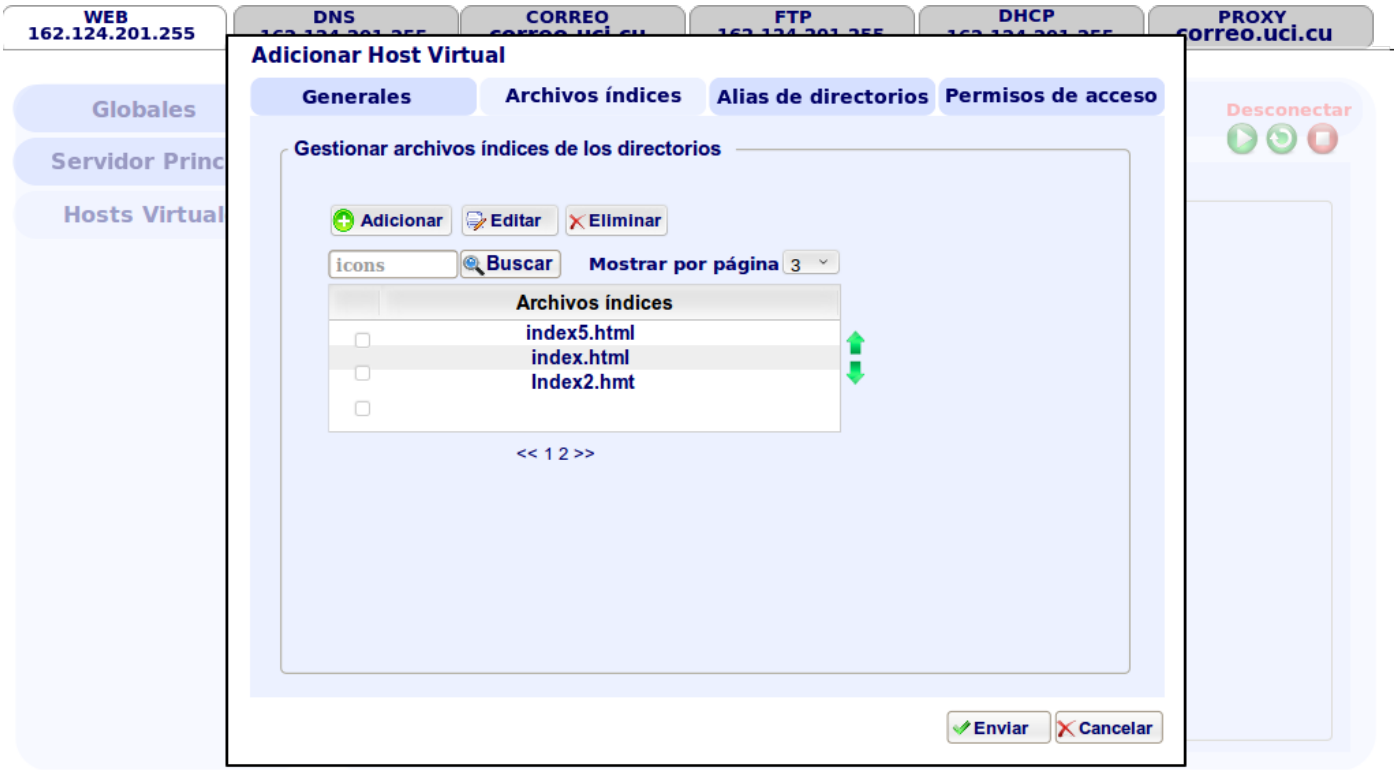

1 Se realiza una petición al servidor
Apache
Tiene organizados los hosts virtuales en una tabla hash, agrupados por la dirección IP y el puerto
2 Busca la dirección IP en la tabla
si no la encuentra
3 Busca si hay hosts asignados a todas las direcciones IP y si algún puerto coincide
en caso positivo
4 La petición es atendida por el primer host que coincida
en caso negativo
5 La petición es atendida por el servidor principal
si la encuentra
6 Busca si algún puerto conicide
si no coincide
7 Busca si hay hosts asignados a todos los puertos (*)
en caso positivo
8 La petición es atendida por el primer host que coincida
en caso negativo
si coincide
9 Determina el tipo de host asignado comprobando si la lista está vacía
en caso positivo
10 El host asignado es basado en IP, por tanto es el que atiende la petición
en caso negativo
10 El host asignado es basado en nombre
en caso negativo
11 Se verifica si en la petición se especificó el nombre del host
en caso negativo
12 La petición es atendida por el primer host de la lista
13 La petición es atendida por el primer host en el que coincida el nombre o el alias



Anexo 5: Gestionar hosts virtuales asignados a Apache2.

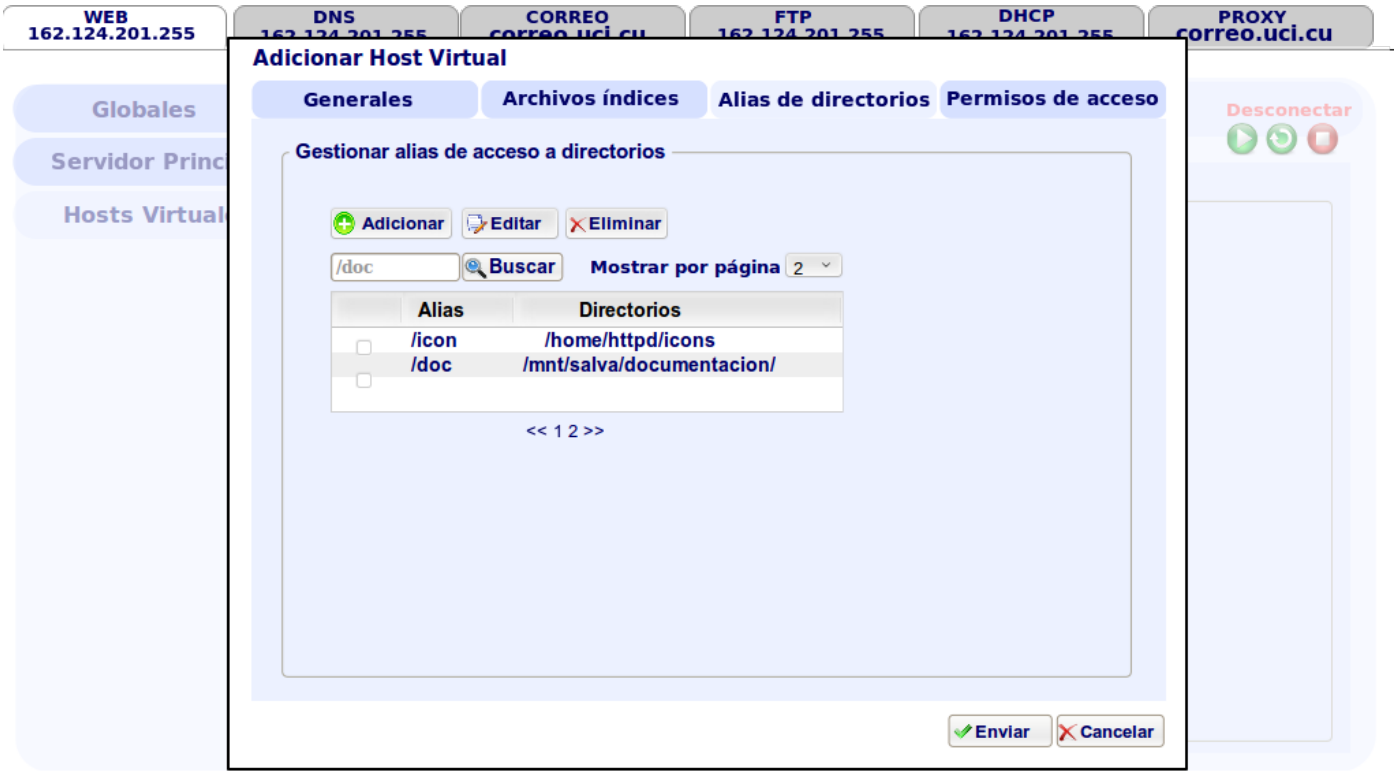


Anexo 6: Adicionar hosts virtuales. Pestaña Generales.

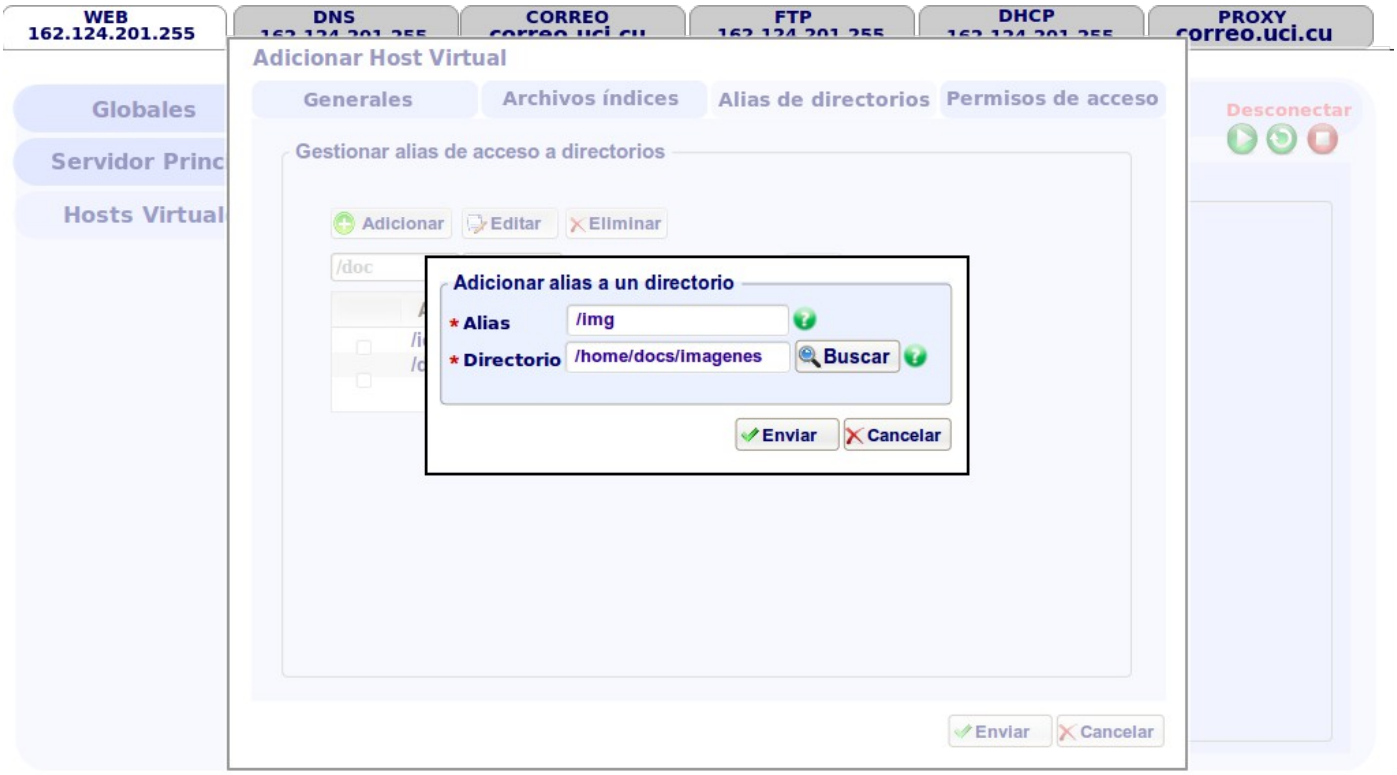


Anexo 7: Adicionar hosts virtuales. Pestaña Archivos índices.

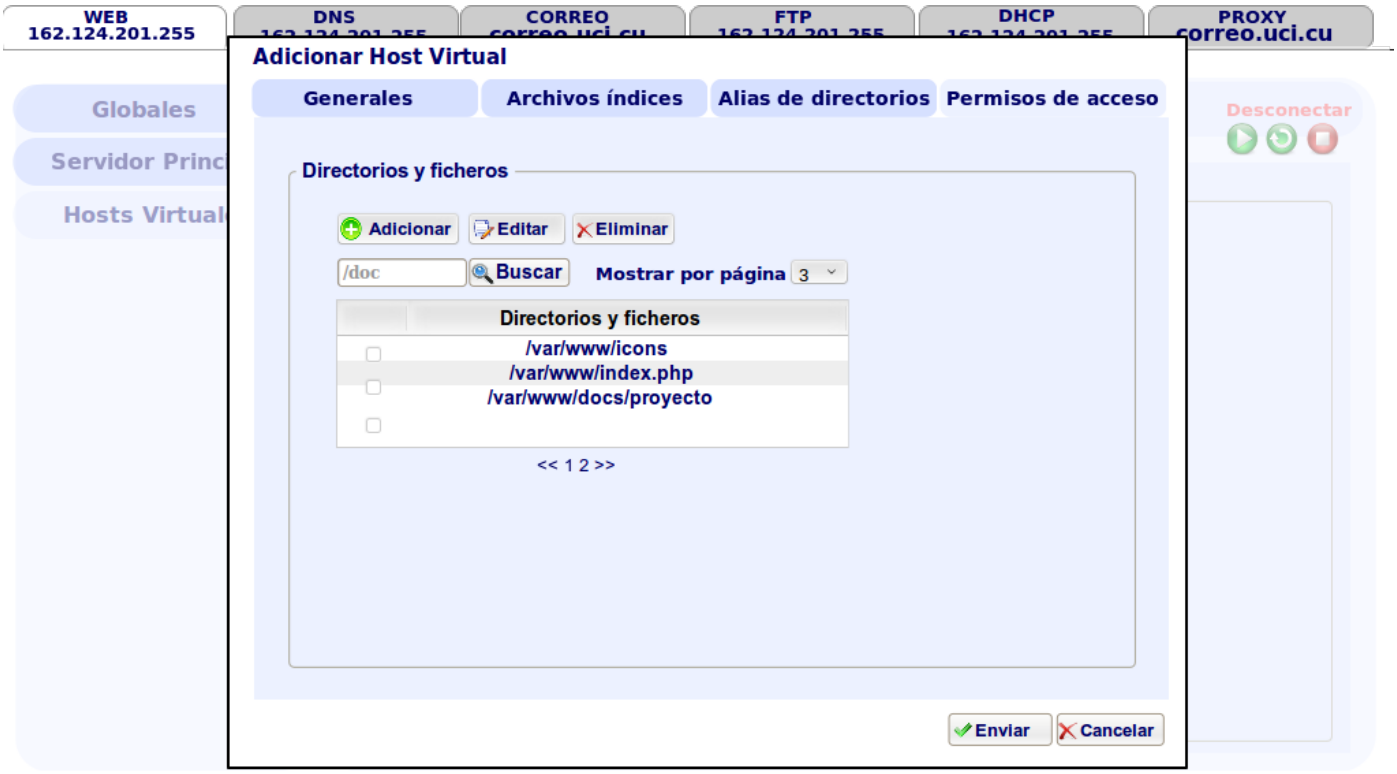
Anexo 8: Adicionar archivo índice en un host virtual.



Anexo 9: Adicionar host virtual. Pestaña Alias de directorios.

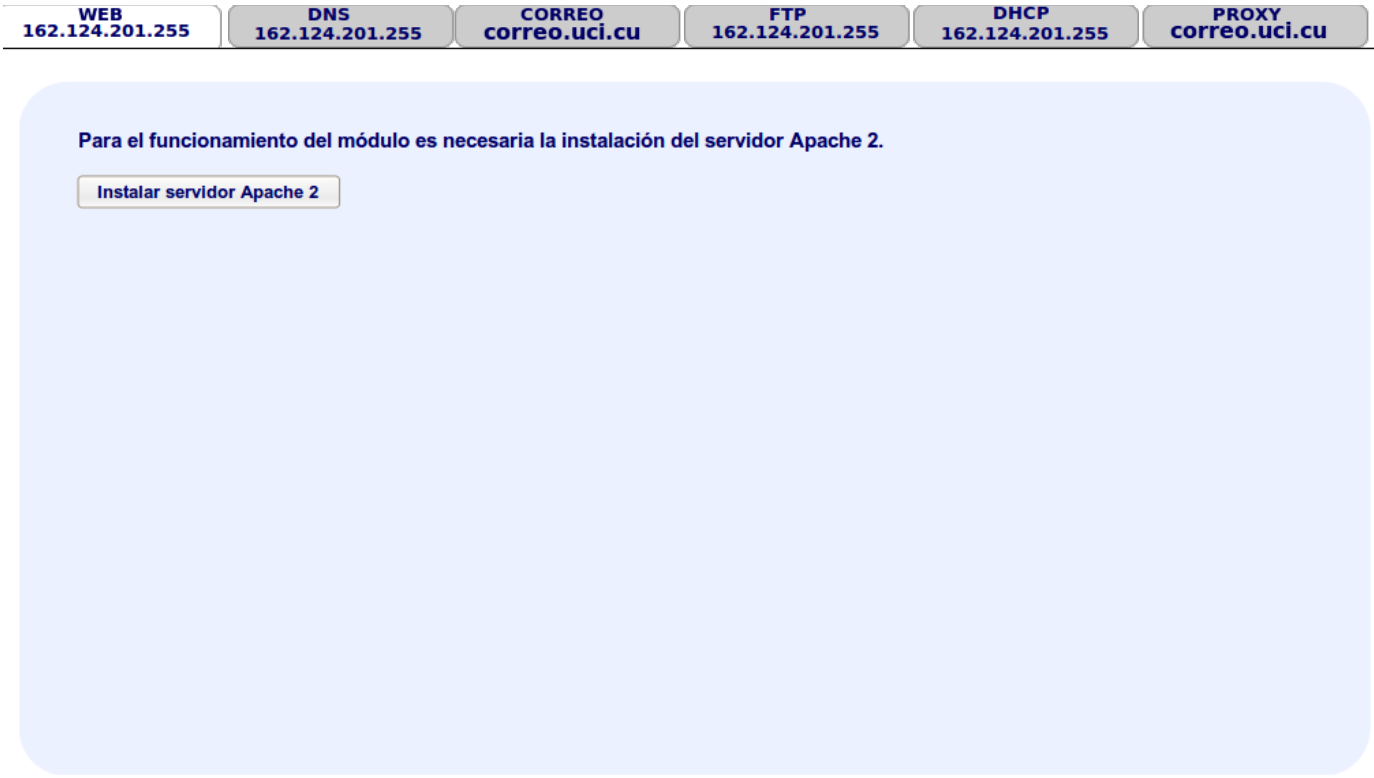


Anexo 10: Adicionar alias a un directorio en un host virtual.



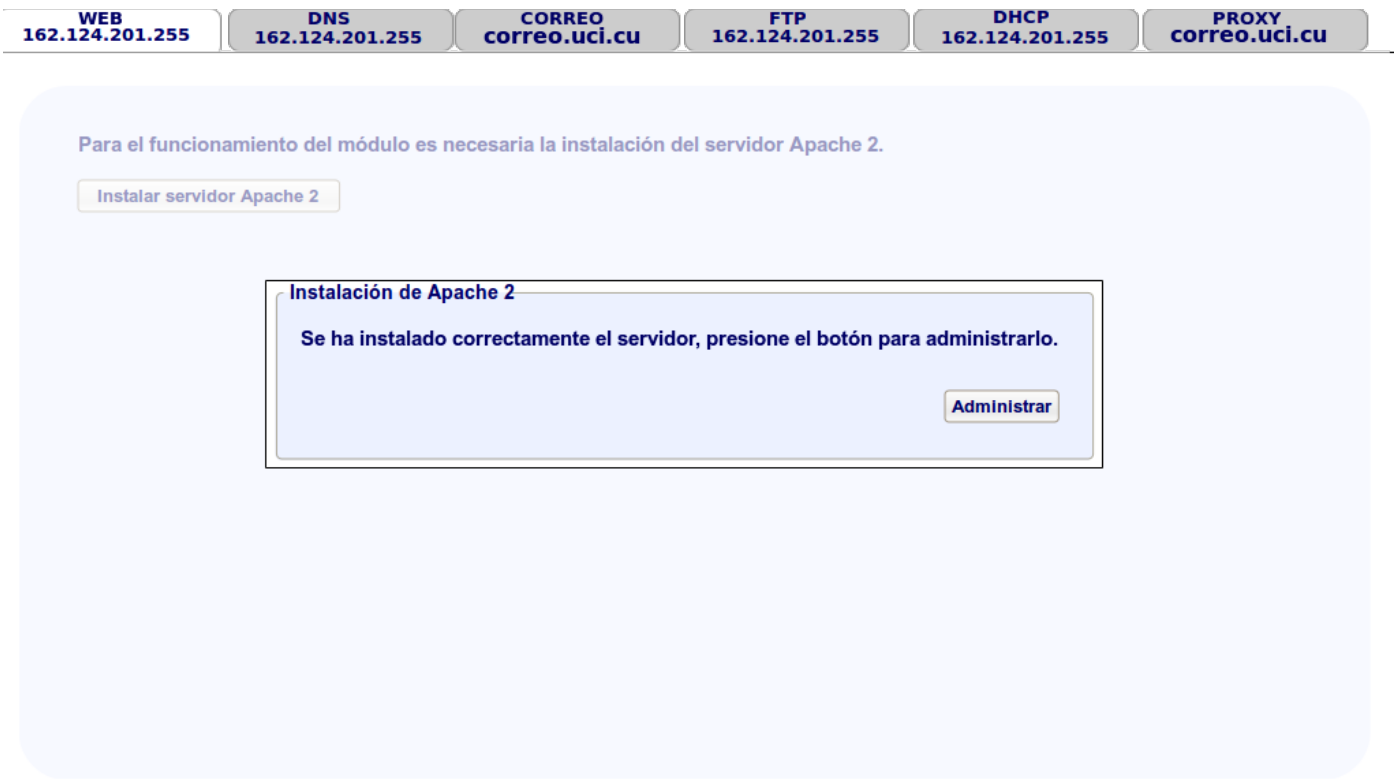
Anexo 11: Adicionar host virtual. Pestaña Permisos de acceso.

Anexo 12: Adicionar permisos de acceso en un host virtual.



Anexo 13: Primer paso para instalar el servidor Apache 2.

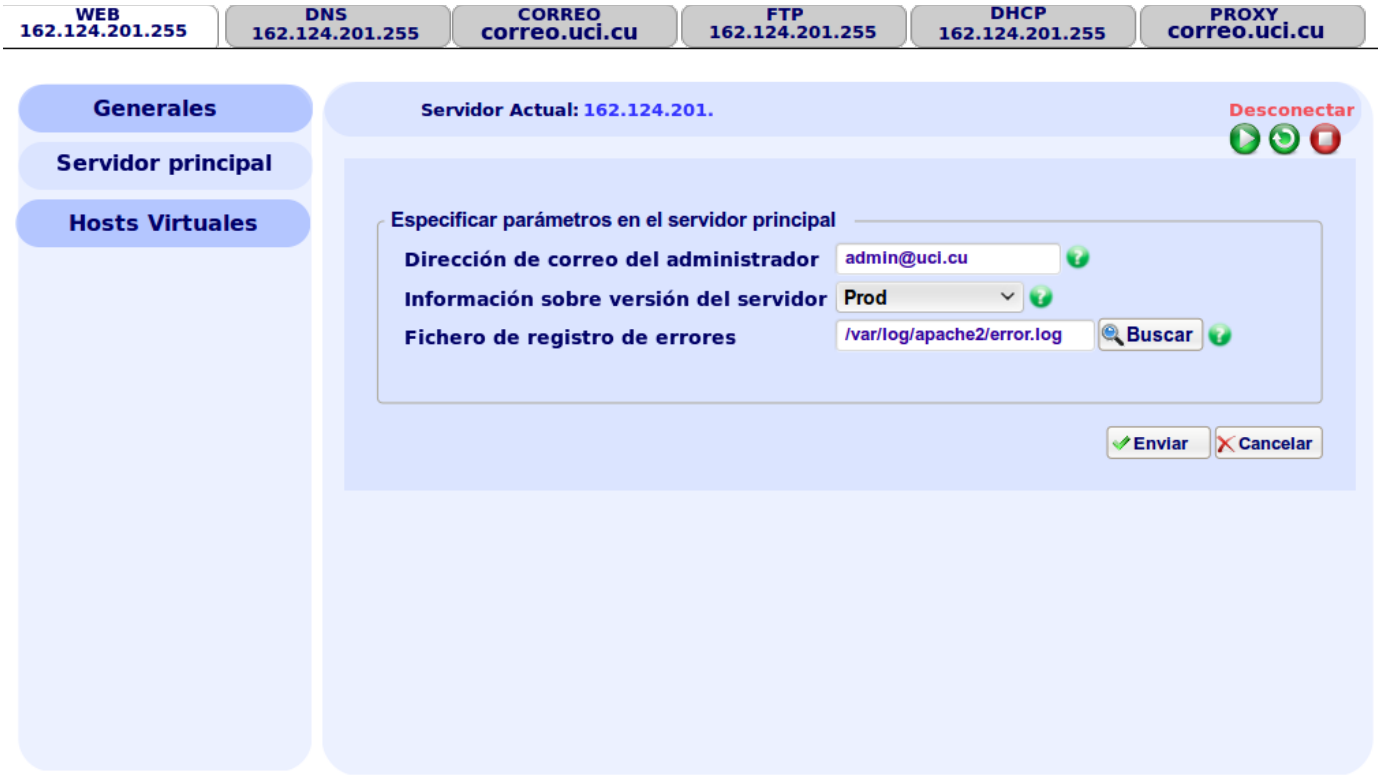
Anexo 14: Segundo paso para instalar el servidor Apache 2.



Anexo 15: Tercer paso para instalar el servidor Apache 2.

Anexo 16: Manejar estados de Apache 2.

Anexo 17: Especificar módulos a usar en Apache 2.



Anexo 18: Especificar parámetros en el servidor principal de Apache 2.

Anexo 19: Especificar parámetros en el MPM prefork de Apache 2.

Anexo 20: Especificar parámetros para el manejo de conexiones en Apache 2.

Glosario de términos

Directivas: son variables almacenadas en el archivo de texto de configuración para alterar y controlar el funcionamiento de Apache en tiempo de ejecución según sus valores y después de haber reiniciado el proceso servidor.

Directorio raíz: directorio que contiene los contenidos asociados al host virtual.

Hilo: es un proceso totalmente aislado y es el responsable de ejecutar el código contenido en el espacio de direcciones del proceso.

Host virtual basado en IP: se refiere a la asignación de un solo host virtual a una única combinación direcciónIP:puerto, por tanto esta dirección IP escuchando por ese puerto tendría asociado un único directorio raíz.

Lista de hosts virtuales basados en nombre: se refiere a la asignación de varios hosts virtuales a una única combinación *direcciónIP:puerto*, donde esta dirección IP escuchando por ese puerto tendría asociado un directorio raíz por cada host virtual.

Módulos de Multiprocesamiento (MPM): son los encargados del manejo de peticiones en Apache 2, responsables de conectar con los puertos de red de la máquina, aceptar las peticiones y generar los procesos hijos que se encargan de servirlas.

PC servidora: computadora con un software servidor instalado.

Proceso: es una instancia u ocurrencia de un programa en ejecución, es propietario de una serie de recursos como un espacio de direcciones en memoria, ficheros, hilos, entre otros. Para que un proceso sea capaz de hacer algo debe ser propietario de al menos un hilo (thread), aunque puede contener varios.

Protocolo: es un conjunto de normas que permiten el intercambio de información entre dos dispositivos o elementos de un mismo nivel.

Servicios telemáticos: son aquellos que, utilizando como soporte servicios básicos, permiten el intercambio de información entre terminales con protocolos establecidos para sistemas de interconexión abiertos.

Servidor proxy: es un sistema que se sitúa entre el host del cliente y el servidor al que quiere acceder. Cuando un host solicita un recurso remoto utilizando una URL, el servidor proxy recibe esta solicitud y

utiliza el recurso para satisfacer la solicitud del cliente.

Servidor web: es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

Servlet: Los servlets son módulos escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java. Los servlets son un reemplazo efectivo para los CGI en los servidores que los soporten ya que proporcionan una forma de generar documentos dinámicos utilizando las ventajas de la programación en Java como conexión a alguna base de datos, manejo de peticiones concurrentes, programación distribuida, entre otros.

Socket: es un punto o medio de comunicación entre dos aplicaciones que permite que un proceso emita o reciba información de otro proceso estando los dos en distintas máquinas.