

# Módulo para la administración remota del servidor web Apache2 en la plataforma Nova ARST

TRABAJO DE DIPLOMA PARA OPTAR POR EL  
TÍTULO DE INGENIERO EN CIENCIAS  
INFORMÁTICAS  
JULIO JASAN BADELL ARGUDIN

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS | La Habana, Cuba.

*Quienes han cultivado el hábito de la persistencia parecen tener una especie de seguro contra el fracaso. No importa las veces que se vean derrotados, siempre terminan por subir el último peldaño de la escalera.*

*Napoleon Hill*

## *Declaración de autoría*

Declaro por este medio que yo Julio Jasan Badell Argudin, con carnet de identidad 95012629061, soy el autor principal del trabajo titulado "Módulo para la administración remota del servidor web Apache2 en la plataforma Nova ARST" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los \_\_ días del mes \_\_ del año \_\_\_\_.

Julio Jasan Badell Argudin

\_\_\_\_\_  
Firma del autor

Ing. Lexys Manuel Días Alonso

\_\_\_\_\_  
Firma del tutor

Ing. Hanny Valdés Hernández

\_\_\_\_\_  
Firma del tutor

## *Datos de contacto*

Tutora:

Ing. Hanny Valdés Hernández  
Universidad de las Ciencias Informáticas  
hanny@uci.cu

Tutor:

Ing. Lexys Manuel Días Alonso  
Universidad de las Ciencias Informáticas  
lmdiaz@uci.cu

# *Agradecimientos*

Que decir, existen muchas personas a las que tengo que agradecer el estar hoy aquí. Tataré de dejar lo menos posible pendiente.

En primer lugar, voy a agradecerle a ti mamá, por estar siempre, por guiarme y apoyarme en cada decisión tomada, por no dejar que desistiera de estudiar una carrera universitaria al salir del servicio militar.

A ti papa, por los consejos dados en momentos difíciles, aunque no parezca, por tu ausencia en otros. Por enseñarme a tomar, aunque tu no aguantes dos cervezas.

A mi abuela, por siempre dejarme un poco de todo lo que ella coma, por coserme la ropa y por ser, aunque no muy cariñosa, la mejor abuela de todos los tiempos.

A Christy, que, aunque hallas llegado de última, has sabido ganarte un lugar importante en mi vida, que no va a desaparecer jamás. Por empujarme a un mundo que tal vez no, pero me gusta pensar que no me atrevería a transitar sin ti. Te amo.

A los hermanos de siempre, Marlon, Yanqui, Jorge Lázaro, Alejandro, Chachi, que, aunque no estemos todos siempre en contacto, basta solo un aviso para que todos corramos al llamado del deber. Jajajaja

A Suset, con quien compartí gran parte de mi travesía en esta universidad y quien me enseñó muchas cosas, entre ellas, que no necesariamente debes estar al lado de quien queremos, que aprender a soltar es una lección que todos debemos aprender cuanto antes.

A la nueva familia que me llevo de esta universidad, pero bueno, a ustedes no los voy a nombrar, jajajaja, ya va, Miguel Luis, que siempre esta inventado algo para enredarme, pero con quien he tenido las conversaciones más profundas e interesantes; Emilio Antonio, baff de lo mejor que pudo haberme pasado, siempre que necesite de unas risas, iré por ti. Esteban, el autor del libro <sup>^</sup>El dominó soy yo<sup>^</sup>, conmigo hasta el final, gracias por todo man, se te quiere. A Olivia mi gordita favorita, Karlucha, Roxana

A Danish, amiga, voy a extrañar y a comer para tu apartamento, aunque sé que como me quieres tanto no vas a dejar de joderme.

Sigo con el mejor grupo de baile de la UCI como decimos sus integrantes, de Nefilim, a los fundadores, Aarom, Luis, Pochy, a los menos viejos Eddy, Yohanderley, Héctor, Adan, Jhoan, Reynier, Lio, Adrian, a las chicas Adria, Elizabeth, Gretel, Liz Jhanet, Maylin, Patricia, chicos, no saben lo que significa para mi todo el tiempo junto a ustedes.

A las amistades que conocí por transitividad, y hoy día forman parte de mi círculo de amigos: Laurita, Leo, Mauricio, Enier, Kilmer, Ruben, Enrico, Roger.

Y, por último, pero no menos importante, quiero agradecerme a mí, por creer en mí, por todo este trabajo duro, por no tener días libres, por nunca renunciar, por ser siempre alguien que da y trata de dar más de lo que recibo, por tratar de hacer las cosas más bien que mal, por ser yo siempre.

Gracias a todos los presentes por compartir este momento junto a mí, y nada, vámonos de fiesta.

## *Dedicatoria*

Dedicado a mi madre, a mi padre, familiares y amigos que siempre creyeron que esto sería posible, a la Universidad de las Ciencias Informáticas, a ti, que estas leyendo esto, espero esta investigación sirva de algo para o que seas que estes buscando.

## *Resumen*

Hoy en día, el software de código abierto es el núcleo de nuestros navegadores web, sistemas operativos y muchos otros aspectos de nuestra vida diaria. Un dilema del mundo real que está resolviendo el software de código abierto es la logística del transporte. Existen diversos tipos de software libre, entre ellos, software de servidores web. Apache2 es un servidor web multiplataforma que posee una arquitectura basada en procesos y basada en hilos. Con el propósito de facilitar la administración de los servicios telemáticos, en CESOL se desarrolló la plataforma Nova ARST. Esta plataforma permite la administración remota desde una estación de trabajo con la distribución Nova Escritorio, de los servicios telemáticos ofrecidos por el servidor. Sin embargo, actualmente la plataforma no permite la configuración de Apache 2, dificultándose la configuración de este servicio, administrar procesos y los diferentes hilos de ejecución del servicio. La presente investigación se propone, desarrollar un módulo para la administración remota del servidor web Apache 2 en la plataforma Nova ARST. El módulo se integrará a la plataforma y tendrá como objetivo fundamental permitir a cualquier computadora conectada mediante el protocolo SSH al servidor, y de manera segura la administración del servidor web en cuestión de forma remota, siguiendo la metodología AUP-UCI, y los lenguajes de programación utilizados en el desarrollo de la plataforma Nova ARST, ya que este es un requisito necesario para la integración del módulo. Como resultado se obtuvo un módulo que administra de forma remota el servidor web Apache 2 diseñado sobre la base de patrones GRASP y GoF. Además, se aplicaron pruebas a nivel de unidad, funcionalidad e integración, para las cuales se obtuvo resultados satisfactorios.

Palabras Clave: Apache2, módulo, servidor web, administración remota.

## *Abstract*

*Today, open-source software is at the core of our web browsers, operating systems and many other aspects of our daily lives. One real-world dilemma that open-source software is solving is transportation logistics. There are several types of open-source software, including web server software. Apache2 is a cross-platform web server that has a process-based, thread-based architecture. In order to facilitate the administration of telematic services, CESOL developed the Nova ARST platform. This platform allows remote administration of the telematic services offered by the server from a workstation with the Nova Desktop distribution. However, currently the platform does not allow the configuration of Apache 2, making it difficult to configure this service, manage processes and the different threads of execution of the service. The present research proposes to develop a module for the remote administration of the Apache 2 web server on the Nova ARST platform. The module will be integrated to the platform and its main objective will be to allow any computer connected through the SSH protocol to the server, and in a secure way the administration of the web server in question remotely, following the AUP-UCI methodology, and the programming languages used in the development of the Nova ARST platform, since this is a necessary requirement for the integration of the module. As a result, we obtained a module that remotely manages the Apache 2 web server designed on the basis of GRASP and GoF patterns. In addition, unit, functionality and integration tests were applied, for which satisfactory results were obtained.*

*Keywords: Apache2, module, web server, remote administration.*



## Índice

Introducción .....	1
Capítulo 1: Fundamentación teórica de la administración remota del servidor web Apache 2 en la plataforma Nova ARST. ....	5
1.1    Conceptos fundamentales .....	5
1.1.1 Servidor .....	5
1.1.2 Servidor web .....	6
1.1.3 Apache 2 .....	7
1.1.4 Módulo .....	10
1.1.5 Administración remota .....	11
1.1.6 Software de administración remota .....	12
1.2 Descripción de la plataforma Nova ARST .....	13
1.2.1 Arquitectura de Nova ARST .....	13
1.3 Estudio de herramientas para la administración remota de Apache 2 .....	14
1.3.1 Webmin .....	14
1.3.2 Apache GUI .....	14
1.3.3 ApacheConf .....	15
1.4 Resultado de los estudios realizados a las herramientas que administran Apache 2 .....	15
1.5 Metodología de desarrollo de software .....	16
1.6 Lenguajes y herramientas para el modelado de la solución .....	17
Conclusiones parciales .....	20
Capítulo 2: Análisis y diseño del módulo para la administración remota del servidor web Apache 2 en la plataforma Nova ARST .....	21
2.1 Propuesta solución .....	21
2.2 Artefactos generados .....	22
2.3 Levantamiento de requisitos .....	22
2.3.1 Técnicas para la obtención de requisitos .....	22
2.3.2 Especificación y descripción de requisitos funcionales .....	24
2.4 Descripción de requisitos de software mediante Historias de Usuario .....	25
2.5 Arquitectura del módulo de administración remota para el servidor web Apache 2 en la plataforma Nova ARST .....	28
2.5.1    Diagrama de paquetes para representar la arquitectura .....	29

# Índice

2.5.2 Diagrama de clases .....	30
2.6 Patrones de diseño .....	31
2.6.1 Patrones GRASP .....	31
Conclusiones parciales .....	34
Capítulo 3: Implementación, pruebas y evaluación del módulo para la administración remota del servidor web apache 2 en la plataforma Nova ARST. ....	35
3.1 Implementación .....	35
3.2 Estándares de implementación .....	35
3.3 Diagrama de Despliegue .....	38
3.4 Diagrama de Componentes.....	38
3.5 Pruebas de software .....	40
3.5.1 Tipos de pruebas .....	40
3.5.2 Métodos de pruebas .....	41
3.5.3 Aplicación de las pruebas de software .....	41
3.5.4 Resultado de las pruebas funcionales realizadas.....	43
Conclusiones parciales .....	46
Conclusiones generales.....	47
Recomendaciones .....	48
Referencias .....	49
Anexos .....	51

## Índice de Imágenes

Imagen 1 Administración remota Fuente: Elaboración propia .....	11
Imagen 2 Arquitectura del módulo Fuente: elaboración propia .....	29
Imagen 3 Diagrama de Paquetes Fuente: elaboración propia.....	30
Imagen 4 Diagrama de clases añadir virtual host Fuente: Elaboración propia .....	31
Imagen 5 Fragmento de código donde se evidencia el patron experto Fuente: Visual Studio Code.....	32
Imagen 6 Fragmento de código de la clase def ServicesCopy Fuente: Visual Studio Code .....	32
Imagen 7 Fragmento de código de la clase ConnectioSSH Fuente: elaboración propia ..	33
Imagen 8 Fragmento de código del método ServicesSSHStop Fuente: elaboración propia .....	34
Imagen 9 Estándares de Python Fuente: Visual Studio Code .....	35
Imagen 10 Importaciones en React Fuente: Visual Studio Code .....	36
Imagen 11 Como comentar en Python Fuente: Visual Studio Code.....	37
Imagen 12 Reglas de Python Fuente: Visual Studio Code .....	37
Imagen 13 Diagrama de Despliegue Fuente: Elaboración propia .....	38
Imagen 14 Diagrama de componentes Fuente: Elaboración propia .....	39
Imagen 15 Prueba unitaria método InstallUninstall Fuente: Visual Studio Code .....	42
Imagen 16 Resultado de la prueba unitaria al método InstallUninstall .....	42
Imagen 17 Resultado de las pruebas funcionales Fuente: elaboración propia .....	44

Tabla 1 Tabla comparativa de relación entre herramientas que administran Apache 2 Fuente: elaboración propia	16
Tabla 2 Requisitos funcionales Fuente: elaboración propia	24
Tabla 3 Requisitos no funcionales Fuente: elaboración propia	25
Tabla 4 Historia de Usuario: Instalar Apache 2 Fuente: elaboración propia	25
Tabla 5 Historia de Usuario: Iniciar Apache 2 (Elaboración propia)	26
Tabla 6 Historia de Usuario: Comprobar Apache 2 Fuente: elaboración propia	27
Tabla 7 Tabla descriptiva del diagrama de componentes Fuente: Elaboración propia	39
Tabla 8 Caso de prueba Comprobar Apache 2 Fuente: elaboración propia	42
Tabla 9 Caso de prueba para Listar Virtual Host Fuente: elaboración propia	43
Tabla 10 Historia de Usuario Desinstalar Apache 2 Fuente: Elaboración propia	52
Tabla 11 Historia de Usuario Detener Apache 2 Fuente: Elaboración propia	52
Tabla 12 Historia de Usuario Reiniciar Apache 2 Fuente: Elaboración propia	53
Tabla 13 Historia de Usuario Recargar Apache 2 Fuente: Elaboración propia	53
Tabla 14 Historia de Usuario Añadir host virtual Fuente: Elaboración propia	53
Tabla 15 Historia de Usuario Deshabilitar Apache 2 Fuente: Elaboración propia	54
Tabla 16 Historia de Usuario Habilitar Apache 2 Fuente: Elaboración propia	54
Tabla 17 Historia de Usuario Configurar Apache 2 Fuente: Elaboración propia	55

## Introducción

Según la Fundación de *Software Libre* (*Free Software Foundation*, en inglés) el *software* libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*; de modo más preciso, se refiere a cuatro libertades de los usuarios: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del mismo, y adaptarlo a sus necesidades; de distribuir copias, con la cual se puede ayudar a otros y de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie, lo cual el acceso al código fuente es un requisito previo para dichas libertades (1).

Hoy en día, el *software* de código abierto (*software open source*) es ahora el núcleo de nuestros navegadores *web*, sistemas operativos y muchos otros aspectos de nuestra vida diaria. Según una encuesta realizada en 2016, el 90 por ciento de las empresas afirman que el código abierto mejora la eficiencia, interoperabilidad e innovación de las mismas. Un dilema del mundo real que está resolviendo el *software* de código abierto es la logística del transporte. Ya sea que se trate de un gobierno municipal que establece rutas óptimas de autobús y tren ligero, un hospital que proporciona a los pacientes la ruta mejor y más rápida a sus instalaciones en un momento determinado, una compañía petrolera que planifica la ubicación de sus oleoductos o un fabricante que envía sus productos de manera eficiente y rentable (2). Existen diversos tipos de *software* libre, entre ellos, *software* de servidores *web*.

Actualmente son numerosos los servidores *web*, entre ellos, *Microsoft-IIS*, *Apache 2*, *Nginx*, *LiteSpeed*, *Google*, *Sun Java System*, *Lighttpd*, *IBM Servers* y *Yahoo Traffic Server*. 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* (3). *Apache 2* y *Nginx* son los dos servidores *web* de código abierto más usados. Según el sitio *Netcraft*, hasta enero del 2022 el uso de *Apache* es de 24% y el de *Nginx* 32%; por otra parte, según el reporte realizado por *W3Techs* hasta el 21 de febrero del 2022, el uso de *Apache* es de 30,7% y de *Nginx* 33,1%.

Cuba no está ajena al desarrollo de las tecnologías de *software* libre y con el afán de proveer a su población, organismos e instituciones de alternativas, surge la distribución cubana GNU/Linux Nova. Solución, de *software* libre o código abierto que se comercializa bajo marcas registradas, de acuerdo a cinco líneas de alto impacto: Salud, Administración Pública, Educación, Empresa-Industria y Telemática.

Nova es una distribución de GNU/Linux hecha en la Universidad de las Ciencias Informáticas (UCI), para garantizar la soberanía tecnológica, la seguridad y la sostenibilidad de las tecnologías del país. Esta nueva distribución cuenta con variantes en dependencia de su empleo y funcionalidades, una de ellas es Nova Servidores, orientada a servidores usando estándares abiertos y adaptada a los entornos de las empresas cubanas. Entre sus características se encuentran la configuración fácil e intuitiva a través de la herramienta *nova-manager*, destinada a la administración de los servicios telemáticos y la compatibilidad con hardware obsoleto en el entorno empresarial. Incluye en el sistema base la posibilidad de instalar y configurar diversas tecnologías desde la propia interfaz de instalación,

## Introducción

asegurando que el usuario no requiera o dependa del acceso a un repositorio para la instalación. Además, implementa varios módulos entre los que se encuentra Apache2.

Con el propósito de facilitar la administración de los servicios telemáticos, en el centro de *software* libre CESOL, se desarrolló la plataforma Nova ARST. Esta plataforma permite la administración remota desde una estación de trabajo con la distribución Nova Escritorio, de los servicios telemáticos ofrecidos por el servidor. El empleo de esta plataforma facilita a los administradores de redes, no tener que memorizar comandos y acceder de forma remota a los diferentes servidores desde una interfaz amigable e intuitiva. Sin embargo, actualmente la plataforma no permite la configuración de Apache 2, esto dificulta la configuración de este servicio, provoca en ocasiones la pérdida de tiempo y la duplicación de esfuerzos, administrar procesos y los diferentes hilos de ejecución del servicio; entre ellos la configuración del sistema de nombres de dominio (DNS por sus siglas en inglés), crucial para la creación de host virtuales y el acceso a redes alojadas en el servidor.

Tomando como punto de partida la situación descrita, la presente investigación se plantea como **problema de investigación**: ¿Cómo realizar la administración remota del servidor *web* Apache 2 desde la plataforma Nova ARST?

El problema planteado tiene como **objetivo de estudio** los servidores web implementados en las distribuciones de *software* libre GNU/Linux y el **campo de acción** estará enmarcado en la administración remota de Apache 2 para la plataforma Nova ARST.

El **objetivo general** es desarrollar un módulo para la administración remota del servidor *web* Apache 2 en la plataforma Nova ARST.

Para un mejor desarrollo de la investigación se plantean como **objetivos específicos**:

1. Elaborar el marco teórico de la investigación sobre la administración remota del servidor *web* Apache 2.
2. Diseñar un módulo para la administración remota del servidor *web* Apache 2 en la plataforma Nova ARST.
3. Implementar un módulo para la administración remota del servidor *web* Apache 2 en la plataforma Nova ARST.
4. Evaluar el módulo para la administración remota del servidor *web* Apache 2 en la plataforma Nova ARST.

Las **tareas de la investigación** planteadas son:

- Realización del marco teórico de la investigación a partir de los elementos que intervienen en el desarrollo de herramientas para la administración remota del servidor *web* Apache 2.
- Caracterización de los módulos que utilizan las herramientas que permitan la administración remota de Apache 2.
- Definición de los requisitos funcionales y no funcionales a tener en cuenta para la propuesta de solución.
- Implementación del módulo para la administración remota del servidor *web* Apache 2 en la plataforma Nova ARST.
- Validación del módulo para la administración remota del servidor *web* Apache 2 en la plataforma Nova ARST.

# *Introducción*

Los Métodos de investigación científica empleados para la realización de este trabajo de diploma se encuentran en dos categorías: métodos teóricos y métodos empíricos.

## **Métodos Teóricos**

Permiten estudiar las características del objeto de investigación que no son observables directamente y posibilitan el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada, su relación con otros fenómenos, así como su aislamiento como objeto estudiado.

**Análisis Histórico Lógico:** se utiliza para, mediante estudio de sistemas similares, conocer la evolución y funcionamiento de la administración remota del servidor web Apache 2 en otras distribuciones de software libre.

**Analítico-Sintético:** Utilizado para conocer los servidores web que implementen administración remota, así como las herramientas que se utilizan para su uso.

## **Métodos empíricos**

Son los llamados métodos de recolección de información en un proceso investigativo, de los cuales se pueden fijar: la observación, la entrevista, el cuestionario y el análisis documental.

**Método Entrevista:** La entrevista es una técnica de gran utilidad en la investigación cualitativa para recabar datos; se define como una conversación que se propone un fin determinado distinto al simple hecho de conversar. De los intercambios realizados con los especialistas funcionales del sistema se generaron una serie de no conformidades que dieron paso al alcance del trabajo de diploma. Se utilizó para para investigar acerca de la plataforma Nova ARST y para conocer cómo se configura el servidor web Apache en la UCI (Anexo 1).

## **Estructura de la investigación**

El documento está conformado por introducción, tres capítulos, conclusiones generales, referencias bibliográficas y anexos, que recogen la información abordada en toda la investigación:

### **CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA ADMINISTRACIÓN REMOTA DEL SERVIDOR WEB APACHE 2 EN LA PLATAFORMA NOVA ARST.**

Este capítulo aborda la fundamentación teórica que sirve de base a la investigación del problema planteado. Se realiza un estudio del estado del arte del tema asociado, a nivel nacional e internacional. Queda plasmada una síntesis de las herramientas existentes referentes a la situación identificada y se describen la metodología y tecnologías en las que se apoya la solución del problema.

### **CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO PARA LA ADMINISTRACIÓN REMOTA DEL SERVIDOR WEB APACHE 2 EN LA PLATAFORMA NOVA ARST.**

En este capítulo se transforman los requisitos funcionales en el diseño del futuro componente para su posterior implementación a partir de la arquitectura definida por la dirección del proyecto. Se adquieren un conjunto de artefactos de gran valor para la fase de construcción como son: la especificación de los componentes, de la arquitectura y la

## *Introducción*

línea base. Igualmente se especifica la utilización de un conjunto de patrones identificados en la línea base obtenida.

**CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBAS Y EVALUACIÓN DEL MÓDULO PARA LA ADMINISTRACIÓN REMOTA DEL SERVIDOR WEB APACHE 2 EN LA PLATAFORMA NOVA ARST.**

En este capítulo a partir del diseño y especificación de los componentes generados en el Capítulo 2 se describe cómo implementar el subsistema, se realiza la validación de la solución desarrollada a través de diferentes tipos de pruebas y se muestra el diagrama de despliegue a partir de la ubicación de cada uno de los nodos que serán usados.



## Capítulo 1: Fundamentación teórica de la administración remota del servidor web Apache 2 en la plataforma Nova ARST.

En este capítulo se realiza un estudio referente al servidor web de código abierto Apache 2. Se definen conceptos necesarios para el entendimiento del contenido expuesto. Se describe Nova ARST, así como su arquitectura y se especifican las tecnologías, herramientas y metodologías a utilizar en el desarrollo del módulo para la administración remota del servidor web en dicha herramienta.

### 1.1 Conceptos fundamentales

Para poder comprender los términos expuestos en la investigación es necesario definir los conceptos asociados a la misma.

#### 1.1.1 Servidor

En informática, se conoce como servidor (del inglés *server*) a un ordenador que forma parte de una red informática y provee determinados servicios al resto de los ordenadores de la misma, llamados a su vez estaciones o clientes. Dicho ordenador debe contar con una aplicación específica capaz de atender las peticiones de los distintos clientes y brindarles respuesta oportuna, por lo que en realidad dentro de una misma computadora física (hardware) pueden funcionar varios servidores simultáneos (software), siempre y cuando cuenten con los recursos logísticos necesarios.

Los servidores, son los encargados de atender las solicitudes de los clientes de una red determinada, y administrar los recursos disponibles a la misma para que cada cliente pueda acceder a la información o a los periféricos que necesita. En ese sentido, los servidores pueden tener funciones muy distintas, tales como:

- **Servidores de archivos.** Almacenan los ficheros o archivos de información y alimentan con ellos a una red.
- **Servidores de Directorio Activo/Dominio.** Administran la información relacionada con la red, sus usuarios, equipos y grupos internos.
- **Servidor de impresión.** Gestiona un conjunto de impresoras disponibles para una red, otorgando acceso a ellas y administrando la cola de impresión.
- **Servidor de correo.** Gestiona el flujo de correo electrónico entre, desde y hacia los clientes de una red, enviando y recibiendo mensajes y almacenando el historial de los mismos.
- **Servidor de Proxy.** Su rol es de respaldo, almacenando durante un tiempo y en memoria caché una copia de las páginas web disponibles para la red, para acelerar el acceso a las mismas o permitir la recuperación de datos si la original se cae.
- **Servidor web.** Almacena el contenido necesario para una o varias páginas web y administra el acceso ordenado al mismo, para que los navegadores de los clientes puedan “renderizar” un sitio web.
- **Servidor DNS.** Almacena la información necesaria para asociar un nombre de dominio con una serie de direcciones IP de los equipos vinculados a ella (sus servidores web).
- **Servidor DHCP.** Encargado de asignar las direcciones IP dinámicas (cambiantes) a los clientes que se conectan a una red.

- **Servidor FTP.** Almacena información puntual de los usuarios y permite el acceso privado a la misma entre equipos.
- **Servidor de juego.** Aquellos específicamente dedicados a almacenar información para que los clientes puedan acceder al mismo tiempo a un programa recreativo (juegos de video masivos, generalmente)(1).

## 1.1.2 Servidor web

Un servidor web es un programa informático que procesa cualquier aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con un cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. Un servidor web se mantiene a la espera de peticiones de ejecución que le hará un cliente o un usuario de internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.

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

- **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(2).
- **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(2).
- **Servidores basados en sockets no bloqueantes o dirigidos por eventos:** que 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 operativos que implementan 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(2).

Los servidores web son programas de uso cotidiano en Internet, que emplean para comunicarse diversos protocolos de datos, siendo el más común y de alguna manera estándar el HTTP (*HyperText Transfer Protocol*). Sin embargo, es posible también usar el término para referirse al ordenador en el que están guardados los archivos que componen un sitio web, junto al software necesario para cumplir con la conexión de datos web.

Un servidor web opera en un ordenador aguardando las solicitudes de parte del navegador web de un cliente, brindando los datos solicitados para componer una página web o, en su

defecto, un mensaje de error. Los servidores web pueden ser de dos clases: estáticos y dinámicos.

- **Los servidores estáticos:** consisten en una computadora en donde está almacenada la información y un servidor HTTP que responde a los protocolos de pedido. Su nombre proviene del hecho de que los archivos se envían tal y como están almacenados.
- **Los servidores dinámicos:** son servidores estáticos que contienen software adicional (usualmente aplicaciones y bases de datos) que les permiten actualizar la información solicitada antes de enviarla al cliente.

Algunos de los servidores web más empleados son los siguientes:

- **Nginx (2004).** Un servidor web y Proxy desarrollado por la empresa homónima.
- **Apache (1995).** Es un servidor web HTTP de código abierto, que sirve para computadores Unix, Windows y Macintosh, desarrollado y mantenido por una comunidad de usuarios que conforman la *Apache Software Foundation*.
- **Internet Information Services o IIS (1993).** Servidor web y conjunto de servicios diseñados para *Microsoft Windows* que fue originalmente incluido en su versión NT.
- **Cherokee (2001).** Es un servidor web multiplataforma escrito en lenguaje C, disponible bajo Licencia Pública General de GNU, de software libre.
- **Tomcat (1999).** Una distribución de Apache conocida también como Jakarta Tomcat, opera bajo el principio de los servlets (Java).

Nginx y Apache son los dos servidores web más utilizados del mundo. Entre ambos poseen aproximadamente dos tercios del mercado. Según datos de W3Techs, Nginx posee en torno al 33,5 % del mercado y Apache en torno al 31,5 % —30,7 % y 23 % respectivamente según datos de Netcraft.

### 1.1.3 Apache 2

Apache, lanzado en el año 1994, 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. En Apache podemos aplicar una alta personalización a través de su sistema modular, de forma que podemos activar o desactivar diversas funcionalidades a través de los módulos de Apache. Estos módulos de Apache hay que usarlos con cautela ya que pueden afectar a la seguridad y funcionalidades del servidor web(3).

La función esencial del servidor Apache es servir las webs alojadas en el servidor a los diversos navegadores como Chrome, Firefox, Safari, entre otros navegadores web. Apache consigue que la comunicación entre el servidor web y el cliente web (usuario que solicita la información) sea fluida y constante. Haciendo que cuando un usuario haga una petición HTTP a través de navegador para entrar a una web o URL específica, Apache devuelva la información solicitada a través del protocolo HTTP.

## Arquitectura para el manejo de peticiones

En Apache 2 cada solicitud es atendida por un hilo separado o proceso y utiliza sockets sincrónicos. Los módulos de procesamiento múltiple (MPM) son responsables de conectar con los puertos de red de la PC servidora, aceptar las peticiones y generar los procesos hijos que se encargan de servirlos. Los sitios web que necesitan más que nada escalabilidad pueden usar un MPM 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*(3).

## Soporte para protocolos basados en CGI

Apache posee soporte para CGI y FastCGI (del inglés *Fast Common Gateway Interface*), a través de módulos que se pueden cargar en el servidor.

### Protocolo CGI

Cuando un cliente intenta visitar una página dinámica, el servidor web recibe la solicitud y la reenvía a una aplicación de terceros. La aplicación procesa el script de forma independiente y devuelve la respuesta producida al servidor web, que luego reenvía la respuesta al cliente. Para que el servidor web se comuniquen con esa aplicación, el protocolo CGI se inventó a principios de la década de 1990, permite que un servidor HTTP y un script CGI compartan la responsabilidad de responder a las solicitudes de los clientes. Esta tecnología parece lo suficientemente simple y eficiente al principio, sin embargo, viene con algunos inconvenientes principales(3).

- Se genera un proceso único para cada solicitud. La memoria y otra información de contexto se pierden de una solicitud a otra.
- Iniciar un proceso puede consumir muchos recursos para el sistema. Cantidades masivas de solicitudes simultáneas (cada vez que se genera un proceso) podrían saturar rápidamente un servidor.

### Protocolo FastCGI

El protocolo CGI es relativamente ineficiente para los servidores que están sujetos a grandes cargas. A mediados de los años 90, la voluntad de encontrar soluciones llevó a *Open Market* a desarrollar una evolución de CGI: FastCGI. Este es un protocolo de comunicación que se ejecuta a través de sockets, lo que implica que hay un cliente y un servidor. Aunque el objetivo sigue siendo el mismo, FastCGI ofrece mejoras significativas sobre CGI. En lugar de engendrar un nuevo proceso para cada solicitud, emplea procesos persistentes que vienen con la capacidad de manejar solicitudes múltiples(3).

## Características de Apache 2

Apache 2 se caracteriza por ser multiplataforma, poseer abundante soporte de documentación y hacer uso de módulos MPM para el manejo de peticiones. Los módulos pueden ser habilitados y deshabilitados dinámicamente con la ejecución de un comando y después de haber reiniciado el proceso del servidor. 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.

- **Multiplataforma:** es utilizado en multitud de sistemas operativos, lo que lo hace prácticamente universal. Actualmente funciona en Windows, GNU/Linux, Unix, BSD, Mac OS X, Solaris, Novell NetWare, OS/2, TPF, OpenVMS, eCS, AIX, z/OS, HP-UX, entre otros(3).
- **Sistema de módulos dinámico:** Apache utiliza el concepto de objetos compartidos dinámicos (DSO, del inglés *Dynamic Shared Object*) en el que puede cargar o descargar módulos en tiempo de ejecución después de compilar Apache. Usando DSO, los módulos no están incluidos en el proceso principal de Apache, por lo tanto, le permite cargar o descargar módulos dinámicamente(3).
- **Soporte de hosts virtuales:** El servidor Apache puede ejecutarse como un único servidor web para un único sitio o puede servir a cientos de sitios como hosts virtuales. De esta manera, muchos sitios web comparten los recursos subyacentes, como la CPU, los recursos de disco y las direcciones de Protocolo de Internet (IP) de un único servidor web(3).
- **Servidor proxy integrado:** un servidor proxy se encuentra entre las PC de los clientes e Internet, realmente es solo una capa adicional entre el cliente y el servidor. En términos generales, un servidor *proxy* actúa como un servidor para los *hosts* del cliente y como un cliente para los servidores remotos. Apache se puede convertir en un servidor proxy caché, el cual es un servidor intermedio que se sitúa entre el cliente y el servidor que tiene los contenidos(3).
- **Requerimientos de hardware:** si el servidor sirve páginas estáticas mayoritariamente, la CPU no va a constituir un factor significativo en el rendimiento. Por el contrario, si genera una gran cantidad de contenido dinámico utilizando SSI (del inglés *Server Side Includes*), *scripts* CGI y similares, el servidor Apache va a necesitar un procesador rápido. Sin embargo, una CPU rápida no significa una ventaja cuando se va a servir un alto volumen de contenido dinámico(3).

## Instalación y configuración de Apache 2

Apache se instala en el directorio `/etc/apache2` con la ejecución del comando `apt-get install apache2`. Este servidor web no posee una interfaz de usuario gráfica para su administración, se trata de un 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 el sistema operativo Nova. 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(3).

Los ficheros de configuración de Apache 2 que podemos encontrar en su ruta de instalación son los siguientes:

- **conf-available:** directorio que contiene los ficheros de configuración disponibles.
- **conf-enabled:** directorio que contiene los ficheros de configuración activos.

- **mods-available:** directorio con los módulos disponibles, incluyendo los MPM. 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.
- **sites-available:** directorio con los ficheros de configuración de los hosts virtuales disponibles. Se recomienda crear cada host virtual en un fichero independiente.
- **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.
- **apache2.conf:** fichero principal de configuración de Apache 2.
- **envvars:** fichero que incluye las variables de entorno que se deseen cargar.
- **magic:** datos para el módulo mod\_mime\_magic.
- **ports.conf:** fichero para configurar el puerto de comunicación HTTP.

La configuración de Apache está dividida en tres secciones fundamentales, aunque las directivas de cada una pueden aparecer mezcladas y desordenadas, estas son:

- **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 puede 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.

## 1.1.4 Módulo

Un módulo es un componente de software o parte de un programa que contiene una o más rutinas. Uno o más módulos desarrollados independientemente forman un programa. Una aplicación de software de nivel empresarial puede contener varios módulos diferentes, y cada módulo sirve operaciones comerciales únicas y separadas.

Los módulos facilitan el trabajo de un programador al permitir que el programador se concentre en un solo área de la funcionalidad de la aplicación de software. Los módulos se incorporan típicamente en el programa (*software*) a través de interfaces(4).

Apache se adapta a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir que funcionalidades van a ser incluidas en el servidor seleccionando que módulos se van a usar, ya sea al compilar o al ejecutar el servidor.

Apache 2 extiende este diseño modular hasta las funcionalidades más básicas de un servidor web. El servidor viene con una serie de Módulos de MultiProcesamiento que son los responsables de conectar con los puertos de red de la máquina, aceptar las peticiones, y generar los procesos hijo que se encargan de servirlos.

La extensión del diseño modular a este nivel del servidor ofrece dos beneficios importantes:



# Capítulo 1

- Apache puede soportar de una forma más fácil y eficiente una amplia variedad de sistemas operativos. En concreto, la versión de Windows de Apache es mucho más eficiente, porque el módulo mpm\_winnt puede usar funcionalidades nativas de red en lugar de usar la capa POSIX como hace Apache 1.3. Este beneficio se extiende también a otros sistemas operativos que implementan sus respectivos MPMs.
- El servidor puede personalizarse mejor para las necesidades de cada sitio web. Por ejemplo, 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 (perchild).

A nivel de usuario, los módulos de multiprocesamiento (MPMs) son como cualquier otro módulo de Apache. La diferencia más importante es que solo un MPM puede estar cargado en el servidor en un determinado momento(5).

## 1.1.5 Administración remota

La administración remota es una forma eficiente de proporcionar al administrador el acceso y control a una o múltiples computadoras, realizando ciertos tipos de prácticas desde un equipo local y que las mismas se ejecuten en otro equipo remoto, independientemente de la distancia física.

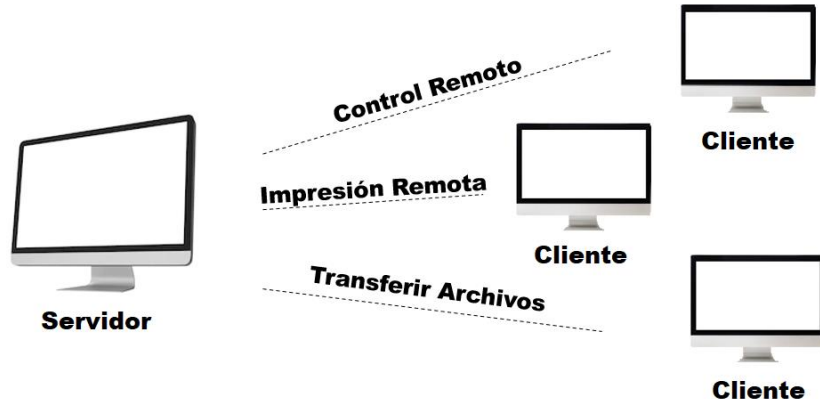


Imagen 1 Administración remota Fuente: Elaboración propia

Hoy en día el acceso remoto es posible a través de(6):

- **Banda ancha por cable:** comparte el ancho de banda con muchos usuarios.
- **DSL (línea de abonado digital):** utiliza una red telefónica y no siempre está disponible si la infraestructura es deficiente.

- **Servicio de Internet móvil:** utiliza dispositivos móviles a través de una conexión inalámbrica; solo es posible si hay una red celular disponible.
- **Satélite:** utiliza satélites para proporcionar acceso a Internet.
- **Banda ancha de fibra óptica:** una de las mejores formas de transferir grandes cantidades de datos y hacerlo rápidamente.
- **VPN / LAN / WAN:** utiliza una red segura y encriptada que crea un túnel de datos entre dispositivos o servidores.
- **Uso compartido de escritorio:** las herramientas de software o las aplicaciones permiten compartir archivos; ideal para seminarios web, conferencias o presentaciones.
- **PAM (Gestión de acceso privilegiado):** las herramientas supervisan el acceso a cuentas privilegiadas en una organización; necesario para transferencias de archivos seguras y acceso a datos confidenciales.
- **VPAM (Vendor Privileged Access Management):** intercambio de red seguro con proveedores o contratistas; puede otorgar acceso solo a partes de un servidor.

Los softwares de administración remota proporcionan características para hacer más segura la conexión remota(7).

## 1.1.6 Software de administración remota

Un software de administración remota ofrece la posibilidad de acceder y controlar, total o parcialmente, computadoras desde cualquier parte del mundo. Lo hace a través de Internet o una red local para ejecutar diferentes actividades deseadas entre un usuario y otro. Este tipo de herramientas se basan en la tecnología de servidor/cliente. El servidor se ejecuta en el computador que es controlado, que a su vez recibe instrucciones del cliente que es instalado como host remoto.

Por lo general, este tipo de programas de administración remota trabajan en segundo plano, sin embargo, estos requieren de autorización del usuario para poder iniciar y permitir acceso remoto(8).

Algunas características que debe cumplir un software para administración remota se exponen a continuación(9).

- **Facilidad de uso:** aumentar la productividad y hacerte la vida un poco más fácil es primordial en esta búsqueda. Debes buscar características que te hagan más fácil el uso de este software, como la capacidad de trabajar en torno a servidores de seguridad y la capacidad de acceder remotamente a equipos que no tienen una dirección IP estática.
- **Herramientas para compartir:** este uso compartido se refiere a las comunicaciones bidireccionales entre el PC remoto y el dispositivo local. Esto incluye la sincronización de archivos, la posibilidad de arrastrar y soltar archivos entre los dos y la capacidad para llevar a cabo conversaciones privadas de chat entre los dos.
- **Características del acceso remoto:** entre ellas se incluyen el control completo del PC remoto, la posibilidad de acceder a un equipo remoto desde más de un dispositivo, la capacidad de encender el ordenador y apagarlo remotamente. Deben de disponer de algo más que el acceso simple a los archivos y la interacción plena del PC a distancia, incluso que seas capaz de reproducir música y encender la cámara web.



- **Seguridad:** sin la excepcional seguridad (al nivel de la seguridad de la banca en línea), la gente no usaría los servicios de acceso remoto de. Encriptar las transferencias de datos y poder bloquear las direcciones IP de acceso remoto. También nos fijamos en los servicios que pueden bloquear los monitores y teclados para los que están cerca del PC remoto no puedan ver lo que haces. Si incluyen un tiempo de espera automático, mejor ya que si te olvidas de salir de una sesión, otros no podrán acceder al PC remoto después.
- **Ayuda y Soporte:** debe ser fácil de usar, pero si surgen problemas deben ser fácilmente accesibles al soporte técnico. Los mejores servicios remotos ofrecen conexión telefónica, Twitter y correo electrónico. La documentación en forma de manual de usuario los archivos de ayuda integrados y una base de conocimientos en línea de búsqueda también es bastante útil.

## 1.2 Descripción de la plataforma Nova ARST

La plataforma Nova ARST es una nueva plataforma que se está desarrollando en el Centro de Software Libre (CESOL) de la UCI para la distribución cubana de GNU/Linux Nova Servidores. La ventaja que ofrece esta nueva plataforma es que permite la fácil configuración de cualquiera de los módulos que se integran en la misma, de una manera mucho más fácil e intuitiva para cualquier usuario, aunque este no presente mucha experiencia en la configuración de los mismos. Otra ventaja que ofrece esta plataforma es que se puede acceder a ella desde cualquier estación de trabajo sin necesidad de desplazarse a donde esté el servidor físicamente y esto es posible gracias a que utiliza una arquitectura cliente-servidor.

Arquitectura Cliente-servidor, también conocida como Modelo Cliente-servidor o simplemente Cliente-servidor. Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes. Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos(10).

Para el desarrollo de la plataforma Nova ARST se utilizaron como *framework* de desarrollo Django, ReactBootstrap.js y Bootstrap. Como lenguajes de programación fueron utilizados Python y JavaScript; para el marcado de hipertexto se utilizó HTML y como lenguaje de hoja de estilos, CSS. En esta plataforma se desea integrar entre otros, servicios como la configuración de DNS, políticas de seguridad y los servidores web.

### 1.2.1 Arquitectura de Nova ARST

Nova ARST está basada en el modelo cliente-servidor, específicamente en el modelo de n-capas, utilizando la interacción de tres capas a través de interfaces.

- **Capa de presentación:** esta capa es la encargada de brindar la información al usuario y a su vez también es la capa con la que el usuario interactúa directamente introduciendo los parámetros necesarios para configurar cada uno de los servicios presentes en la plataforma.

- Capa de aplicación y dominio: en esta capa es donde se encuentra implementada toda la lógica del negocio. Es la capa encargada de realizar todos los procesos que no son visibles por los usuarios y que este solicita mediante la capa de presentación y la de persistencia de datos, ya que también se encarga de transmitir los datos introducidos por el usuario a esta última.
- Capa de persistencia de datos: esta capa se encarga de recibir y guardar los datos ya sea en una base de datos o un fichero de configuración y también de brindar dichos datos cuando son solicitados por algún usuario.

Una vez definidas las características de Nova ARST, posteriormente se hace un estudio de las herramientas que administran de forma remota Apache 2, para determinar si cumplen con los requisitos para ser integradas.

## 1.3 Estudio de herramientas para la administración remota de Apache 2

Aunque no son muchas, existen herramientas que automatizan la administración remota del servidor web Apache 2. Entre ellas se encuentran Webmin, Apache GUI y ApacheConf.

### 1.3.1 Webmin

Webmin es un programa que simplifica el proceso de gestión de un sistema Linux o Unix. Normalmente, se debe editar manualmente los archivos de configuración y ejecutar comandos para crear cuentas, configurar un servidor web y administrar el reenvío de correo electrónico. Webmin permite realizar estas tareas, entre otras, a través de una interfaz web fácil de usar y actualiza automáticamente todos los archivos de configuración necesarios(11).

### Módulo de Webmin para la administración de Apache 2

La herramienta Webmin incluye un módulo de administración y control del servidor web apache. Este módulo se instala automáticamente en Webmin cuando se ha instalado Apache.

La ventana principal de este módulo muestra tres opciones de configuración:

- **Global Configuration:** permite configurar parámetros globales de funcionamiento del servicio.
- **Existing Virtual Host:** muestra una lista de todos los sitios web que administra el servidor.
- **Create Virtual Host:** sirve para crear un nuevo sitio web virtual. Para que el servidor web pueda administrar otros sitios web

Desde el enlace “configuración de módulo” se accede a una ventana en la que se puede configurar el entorno de trabajo del propio módulo de configuración, así como la “configuración del sistema” donde se pueden consultar y modificar las rutas y nombres de los archivos de configuración de Apache y los comandos de control del servicio.

### 1.3.2 Apache GUI

El proyecto Apache GUI se utiliza para proporcionar una GUI de servidor HTTP Apache de código abierto basada en Java. La solución se implementa como una aplicación web a la que se puede acceder a través de un navegador web.

Entre las configuraciones que brinda Apache GUI, se encuentran:

- Iniciar, detener y reiniciar Apache
- Ver, instalar y eliminar módulos de apache
- Buscar palabras clave en los archivos de configuración
- Editar configuraciones
- Probar la configuración del servidor en busca de errores.

### 1.3.3 ApacheConf

ApacheConf es el *Shell* para configurar servidores web Apache. Representa toda la información del archivo httpd.conf, de los archivos incluidos, de los archivos de registro, .htaccess, .htpasswd y .htgroup en la vista estructurada. Todas las directivas del servidor están agrupadas por categoría y todos estos grupos están representados como un árbol. Además, todos los registros del servidor, archivos .htaccess, archivos de contraseñas de sus usuarios, entre otros, están disponibles solo con un clic del mouse. De esta forma, se puede ver toda la estructura del servidor a la vez y se puede administrar fácilmente todas las directivas del servidor, así como los directorios y hosts virtuales(12).

ApacheConf tiene una serie de características que le permiten editar su httpd.conf y administrar su servidor de manera fácil y rápida, haciendo que su trabajo sea más eficiente y ordenado. Entre ellas:

- Representa toda la información del archivo de configuración principal de Apache en una vista estructurada. Todas las directivas del servidor están agrupadas por categoría (directivas globales, directorios y hosts virtuales) y todos estos grupos están representados como un árbol para facilitar el acceso.
- Trabaja con varios servidores web.
- Interacción con su servidor Apache. Puede iniciar, detener o reiniciar un servidor Apache en una computadora local o en una computadora remota de manera fácil y rápida.
- Edición del archivo httpd.conf usando GUI y manualmente en un editor especial con resaltado de sintaxis, marcadores, números de línea, plegado de código.
- Inicio rápido, detención y reinicio del servidor web Apache.
- Descarga y carga de los archivos de configuración directamente desde (al) servidor remoto a través de conexiones SSH o FTP.
- Carga automática de archivos de configuraciones externas, como archivos de la directiva 'Include', archivos .htaccess, .htpasswd, .htgroup, CustomLog y ErrorLog.

### 1.4 Resultado de los estudios realizados a las herramientas que administran Apache 2

En la tabla 1, se describe, a modo de resumen, la relación de estas herramientas atendiendo a los requerimientos que se tuvieron en cuenta.

Requerimientos:

- Sistemas Operativos basados en GNU/Linux
- Código abierto
- Plataforma Web
- Arquitectura y tecnologías para integración a Nova ARST

Tabla 1 Tabla comparativa de relación entre herramientas que administran Apache 2 Fuente: elaboración propia

Herramienta	Sistemas Operativos basados en GNU/Linux	Código abierto	Plataforma Web	Arquitectura y tecnologías para integración a Nova ARST
Webmin	Si	Si	Si	No
Apache GUI	Si	Si	Si	No
ApacheConf	No	Si	No	No

Webmin, a pesar de las facilidades que brinda, el módulo que posee para la administración remota de Apache 2, no cumple con las características en cuanto a arquitectura y tecnologías para ser integrado en Nova ARST. Apache GUI también brinda las funcionalidades necesarias para administrar Apache 2, pero está basado en Java, lo cual impide su integración a Nova ARST. ApacheConf no es una plataforma web sino una aplicación de escritorio desarrollada para el sistema operativo Windows. Teniendo en cuenta el análisis anterior, se decide entonces crear un módulo para administrar Apache 2. Este módulo debe ser de código abierto, debe desarrollarse para integrarse a Nova ARST, su sistema operativo va a ser basado en GNU/Linux.

## 1.5 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 un 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.

La UCI desarrolló una versión de la metodología de desarrollo de software AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma, y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, denominada Ejecución, y agregándose también una nueva fase denominada Cierre.

**Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance de este, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no.

**Ejecución:** en esta fase se ejecutan las actividades requeridas para desarrollar el software incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se realizan pruebas al producto.

**Cierre:** en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Los roles definidos por esta metodología son: jefe de proyecto, planificador, analista, arquitecto de información (opcional), desarrollador, administrador de la configuración, cliente/proveedor de requisitos, administrador de calidad, probador, arquitecto de sistema y administrador de base de datos. Además, AUP-UCI define cuatro escenarios para modelar el sistema (Rodríguez Sánchez 2019).

Para el desarrollo del módulo para la administración remota del servidor web Apache 2 en la plataforma Nova ARST, con el fin de lograr la homogeneidad del sistema se emplea la metodología de desarrollo de software AUP-UCI. Se emplea el escenario número 4, debido a que el proyecto fue evaluado y como resultado se obtuvo un negocio bien definido. Este escenario indica que proyectos que no modelen negocio solo pueden modelar el sistema con historias de usuario (HU).

## 1.6 Lenguajes y herramientas para el modelado de la solución

El módulo a desarrollar debe ser integrado a Nova ARST, por lo tanto, las herramientas y tecnologías que se empleen, se deben corresponder con las mismas de la herramienta principal.

*Frontend* es la parte de un sitio web que interactúa con los usuarios, por eso decimos que está del lado del cliente. *Backend* es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web, por eso decimos que el *backend* corre del lado del servidor. Estos dos conceptos explican a grandes rasgos cómo funciona un sitio o aplicación web y son fundamentales para cualquier persona que trabaje en el mundo digital (Platzi).

Estas herramientas serán usadas para desarrollar el Backend del módulo:

### Lenguaje de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente (2).

- Python: El lenguaje de programación es ampliamente utilizado por empresas de todo el mundo para construir aplicaciones web, analizar datos, automatizar operaciones y crear aplicaciones empresariales fiables y escalables (15).

### Framework

Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software (16).

- Django: Es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles (14).

Las herramientas que se utilizarán para el Frontend son las siguientes:

## Lenguaje de programación

- JavaScript: es el lenguaje de programación que debes usar para añadir características interactivas a tu sitio web, (por ejemplo, juegos, eventos que ocurren cuando los botones son presionados o los datos son introducidos en los formularios, efectos de estilo dinámicos, animación, entre otros.

## Framework

- React.js: es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre.
- Bootstrap: es sin duda el framework CSS más utilizado del mundo, creado por la gente de Twitter. Si ya lo conoces y tienes costumbre de trabajar con él, te encantará esta biblioteca. Te ofrece multitud de componentes con los estilos de Bootstrap, pero basados en React listos para utilizar y conseguir una interfaz atractiva instantáneamente.

Otras herramientas que se utilizan para desarrollar el módulo:

## Editor de código

- Visual Studio Code: Editor de código fuente independiente que se ejecuta en Windows, macOS y Linux. La elección principal para desarrolladores web y JavaScript, con extensiones para admitir casi cualquier lenguaje de programación(17).

## Lenguaje para el modelado

- UML: El Lenguaje Unificado de Modelado (UML) desempeña un rol importante no solo en el desarrollo de software, sino también en los sistemas que no tienen software en muchas industrias, ya que es una forma de mostrar visualmente el comportamiento y la estructura de un sistema o proceso. el UML ayuda a mostrar errores potenciales en las estructuras de aplicaciones, el comportamiento del sistema y otros procesos empresariales(17).

## Herramienta de modelado

- Visual Paradigm: es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Para el modelado de la solución se empleará esta herramienta, diseñada para la construcción de los sistemas de software a gran escala de manera fiable y con un enfoque orientado a objetos(18).

## Lenguaje de hoja de estilos

- CSS: son las siglas en inglés para «hojas de estilo en cascada» (*Cascading Style Sheets*). Básicamente, es un lenguaje que maneja el diseño y presentación de las páginas web, es decir, cómo lucen cuando un usuario las visita. Funciona junto con el lenguaje HTML que se encarga del contenido básico de las páginas.

## Control de versiones

Un sistema de control de versiones es una herramienta que hará un seguimiento de los cambios en los archivos y permitirá la coordinación de diferentes desarrolladores que trabajan en partes de su sistema al mismo tiempo(2). En el proyecto donde se realiza el módulo se utiliza Git por políticas de seguridad informática de la universidad.

- Git: es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git cuentan con una buena representación en la base de talentos disponibles para el desarrollo de software, y este sistema funciona a la perfección en una amplia variedad de sistemas operativos e IDE (entornos de desarrollo integrados)(19).

## Conclusiones parciales

El estudio de módulos que permiten administración remota de Apache 2, no dan solución a al problema en cuestión, pero aportan ideas a la solución de dicho problema, y sirven de base para desarrollar una idea general de cómo se debe representar la solución final. Además, permitió definir los lenguajes y tecnologías para el desarrollo permiten elaborar un producto de calidad y brindan la garantía de contar con las libertades de *software* libre una amplia documentación y robusta comunidad de usuarios.



## Capítulo 2: Análisis y diseño del módulo para la administración remota del servidor web Apache 2 en la plataforma Nova ARST.

En este capítulo se presenta la propuesta de solución a desarrollar, de la cual se generan un conjunto de artefactos como salida del sistema, en los cuales se representa las actividades que se realizan para una mejor comprensión de las mismas. Se definen los requisitos que son las exigencias con las que el subsistema tiene que cumplir, así como la descripción de los mismos.

### 2.1 Propuesta solución

Al realizar un análisis sobre el engorroso trabajo que representa el despliegue específico de una herramienta de administración remota para el servidor web Apache 2 que se integre a la plataforma Nova ARST, se propone el desarrollo de un módulo que facilite la administración remota del servidor web. El módulo se integrará a la plataforma y tendrá como objetivo fundamental permitir a cualquier computadora conectada mediante el protocolo SSH al servidor, y de manera segura la administración del servidor web en cuestión de forma remota. Este módulo está enfocado en configurar las funcionalidades de Apache 2, como lo son servir las webs alojadas en el servidor a los diversos navegadores, mantener el carácter modular del servidor web para no limitar así las diferentes funcionalidades que se puedan requerir en cualquier momento, a través de los módulos de Apache 2 y gestionar los virtual host.

#### 2.1.1 Modelo de conceptual

El modelo de dominio o conceptual se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. El modelo conceptual es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir.

El modelo conceptual es tomado como el punto de partida para el diseño del sistema. Cuando se realiza la programación orientada a objetos, el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo conceptual constituye una primera versión del sistema(20). En la imagen 2 se muestra el modelo conceptual de la propuesta de solución.

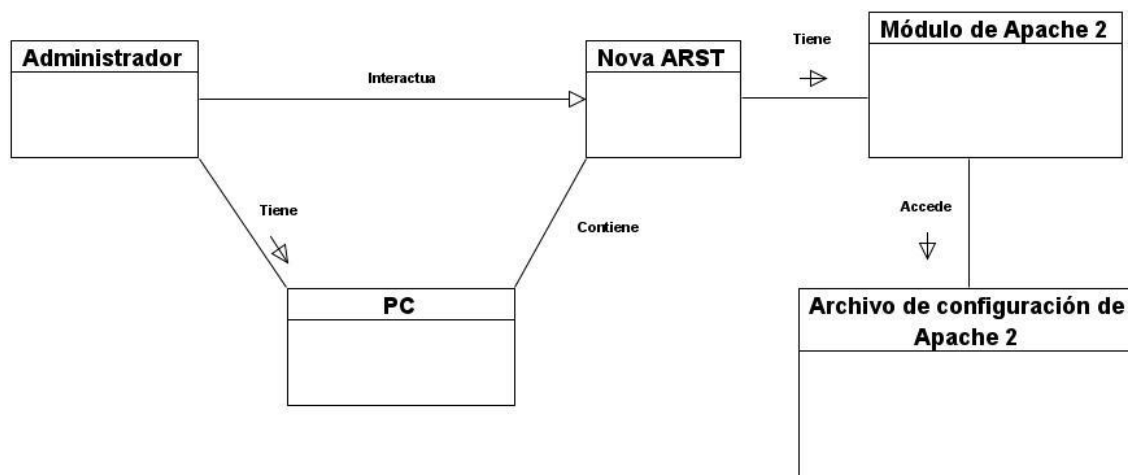


Imagen 2 Modelo Conceptual Fuente: Elaboración propia

### 2.2 Artefactos generados

El proceso de desarrollo del módulo guiado por la metodología AUP-UCI genera como principal artefacto las Historias de Usuario. Teniendo en cuenta que no se modela el negocio, se ajustan las funcionalidades que se describen en un documento de Especificación de Requisitos de Software al escenario 4 que establece esta metodología, el cual se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido, además se recomienda en proyectos no muy extensos, puesto que una HU no debe poseer demasiada información.

### 2.3 Levantamiento de requisitos

Los requisitos de un sistema son la descripción de lo que es sistema debe hacer, los servicios que proporciona y las restricciones de sus operaciones. Dichos requerimientos reflejan las necesidades de los clientes del sistema. El proceso de obtención, análisis, documentación y validación de esos servicios y restricciones se llama ingeniería de requisitos. Los requisitos se clasifican en funcionales (servicios que el sistema debe proporcionar) y no funcionales (restricciones de los servicios o funciones ofrecidos por el sistema)(21).

Según la IEEE un requerimiento o requisito es la condición o capacidad que debe poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.

Los requisitos de software pueden dividirse en dos tipos o dos categorías, los cuales son funcionales y no funcionales.

**Requisitos funcionales:** son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requisitos.

**Requisitos no funcionales:** tienen que ver con características que de una u otra forma puedan limitar el sistema por ejemplo el rendimiento (en tiempo y espacio), interfaces de usuarios, fiabilidad (robustez de sistema), mantenimientos, de seguridad, auditabilidad y otros.

A continuación, se describe cómo se obtuvieron los requisitos de la propuesta de solución, su especificación, descripción y validación.

#### 2.3.1 Técnicas para la obtención de requisitos

Una etapa fundamental en proyectos de ingeniería de software, es la identificación y documentación de los requerimientos del futuro sistema al comienzo del proyecto, pues en numerosas ocasiones se ha demostrado que es cuando pueden prevenirse errores que puedan significar el fracaso del proyecto.

Debido a la complejidad que este proceso implica se han trasado técnicas que permiten realizarlo de una manera más eficiente. Las técnicas de identificación de requisitos de software según (21) son procesos para la obtención de información sobre el sistema requerido y sistemas existentes, permite separar los requerimientos del sistema y los del usuario. Algunas fuentes de información son documentaciones, *stakeholders* y

especificaciones de sistemas similares. Estas técnicas permiten identificar las necesidades de negocio de los clientes y los usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle, requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar(18).

Entre el grupo de técnicas que de forma clásica se han utilizado en el proceso de desarrollo, para el levantamiento de requisitos de este módulo, en este epígrafe se abordarán las técnicas utilizadas para realizar el levantamiento de requerimientos de software, entre ellas las de productos del "mundo real", observación, mesas de trabajo(22).

- **Productos del "mundo real":** Teniendo en cuenta los productos existentes similares al que se desea desarrollar, se identificaron requisitos que resultan imprescindibles en el desarrollo del mismo.
- **Observación:** Por medio de esta técnica el analista obtiene información de primera mano sobre la forma en que se efectúan las actividades. Este método permite observar la forma en que se llevan a cabo los procesos y, por otro, verificar que realmente se sigan todos los pasos especificados. Como sabemos, en muchos casos los procesos son una cosa en papel y otra muy diferente en la práctica. Los observadores experimentados saben qué buscar y cómo evaluar la relevancia de lo que observan (19). Se realizó un estudio (activo y pasivo) en el entorno de trabajo de los usuarios e interesados en el proyecto (*Stakeholders*), identificando de esta forma nuevos requerimientos para el desarrollo del módulo.
- **Mesas de trabajo (*Workshops*):** Un Workshop es un taller, por lo que se necesitan a varias personas para poder desarrollar esta actividad. La función del Workshop no es otra que la de adquirir nuevos conocimientos o habilidades durante el desarrollo del mismo, así como conseguir obtener otros puntos de vista de sus integrantes y el trabajo en equipo(23). Se organizaron diferentes reuniones con usuarios, clientes e interesados de proyecto para así identificar nuevos requisitos indispensables en el desarrollo del módulo.
- **Estudio de documentación:** El estudio de varios tipos de documentación como manuales y reportes, puede proporcionar al analista información valiosa con respecto a las organizaciones y a sus operaciones. La documentación difícilmente refleja la forma en que realmente se desarrollan las actividades, o donde se encuentra el poder de la toma de decisiones. Sin embargo, puede ser de gran importancia para introducir al analista al dominio de operación y el vocabulario que utiliza(24). En la sección de anexos (Anexo 2), se presenta una guía de estudio de la documentación del proceso de administración remota de servidores *web*, cuáles son los datos que son adquiridos y la forma de representarlos.

Las técnicas llevadas a cabo para la recopilación de información relacionada con las herramientas de administración remota de servidores *web*, junto con las opiniones de los trabajadores del centro CESOL, esclarecieron aspectos relacionados con la ingeniería de requisitos, específicamente la fase de obtención de requisitos permitiendo al autor de la presente investigación, captar las principales necesidades de *software* y humanas en las que se debe centrar, las mismas serán presentadas en el epígrafe siguiente.

## 2.3.2 Especificación y descripción de requisitos funcionales

Una especificación de requisitos de software es un documento que se crea cuando debe especificarse una descripción detallada de todos los aspectos del software que se va a elaborar, antes de que el proyecto comience(18).

### Requisitos Funcionales:

Tabla 2 Requisitos funcionales Fuente: elaboración propia

Número	Nombre del requisito	Descripción	Prioridad
RF1	Instalar Apache 2.	Permite realizar la instalación Apache 2.	Alta
RF2	Desinstalar Apache 2	Permite desinstalar Apache 2	Alta
RF3	Iniciar Apache 2	Permite iniciar Apache 2 cuando esta inactivo.	Alta
RF4	Detener Apache 2	Permite detener Apache 2 cuando está activo.	Alta
RF5	Reiniciar Apache 2	Permite reiniciar Apache 2 para cargar los cambios realizados en el servidor.	Alta
RF6	Recargar Apache 2	Permite recargar Apache 2.	Alta
RF7	Añadir host virtual	Permite añadir un host virtual a Apache 2.	Alta
RF8	Deshabilitar Apache 2	Deshabilita Apache 2.	Media
RF9	Habilitar Apache 2	Habilita Apache 2.	Media
RF10	Comprobar Apache 2	Permite comprobar el estado de Apache 2.	Media
RF11	Configurar Apache 2	Permite realizar la configuración de Apache 2.	Media
RF12	Modificar Virtual Host en Apache 2	Permite modificar un virtual host a Apache 2.	Media
RF13	Listar Virtual Host en Apache 2	Permite listar un virtual host a Apache 2.	Media
RF14	Eliminar Virtual Host en Apache 2	Permite eliminar un virtual host a Apache 2.	Media

### Requisitos no funcionales:

Las clasificaciones de atributo de calidad utilizados, se rigen bajo la norma ISO 25010(25).

Tabla 3 Requisitos no funcionales Fuente: elaboración propia

## 2.4 Descripción de requisitos de software mediante Historias de Usuario

Una historia de usuario es una explicación general e informal de una función de software

Atributo de calidad	Número	Descripción
Adecuación Funcional	RNF.1	El módulo debe realizar todas las operaciones indicadas por el usuario en cada momento: añadir, listar, modificar y eliminar archivos del directorio.
	RNF.2	Al módulo solo podrán acceder usuarios con permiso de administración.
Seguridad	RNF.3	Para realizar alguna configuración en una PC servidora, el usuario debe ingresar el usuario, contraseña e ip de la PC donde desea realizar la configuración.
	RNF.4	El módulo debe permitir la conexión mediante el protocolo SSH.
	RNF.5	El módulo garantiza la protección contra el acceso a datos e información no autorizados y previene los accesos o modificaciones no autorizados a datos de las configuraciones del servicio Apache 2.
Fiabilidad	RNF.6	El módulo debe ser capaz de recuperarse después de haberse producido un fallo de software.
Compatibilidad	RNF.7	El módulo debe ser compatible con Nova ARST.
	RNF.8	El módulo debe ser capaz de coexistir con los demás módulos que se integren a la plataforma Nova ARST y no dificultar su funcionamiento.
Portabilidad	RNF.9	El módulo desarrollado debe permitir ser adaptado a diversos entornos.

escrita desde la perspectiva del usuario final. Su propósito es articular cómo proporcionará una función de software valor al cliente. Las historias de usuario son uno de los componentes centrales de un programa ágil. Ayudan a proporcionar un marco centrado en el usuario para el trabajo diario, lo que impulsa la colaboración y la creatividad y mejora el producto en general(26).

Tabla 4 Historia de Usuario: Instalar Apache 2 Fuente: elaboración propia

HISTORIA DE USUARIOS	
Número: RF1	Nombre del requisito: Instalar Apache 2.

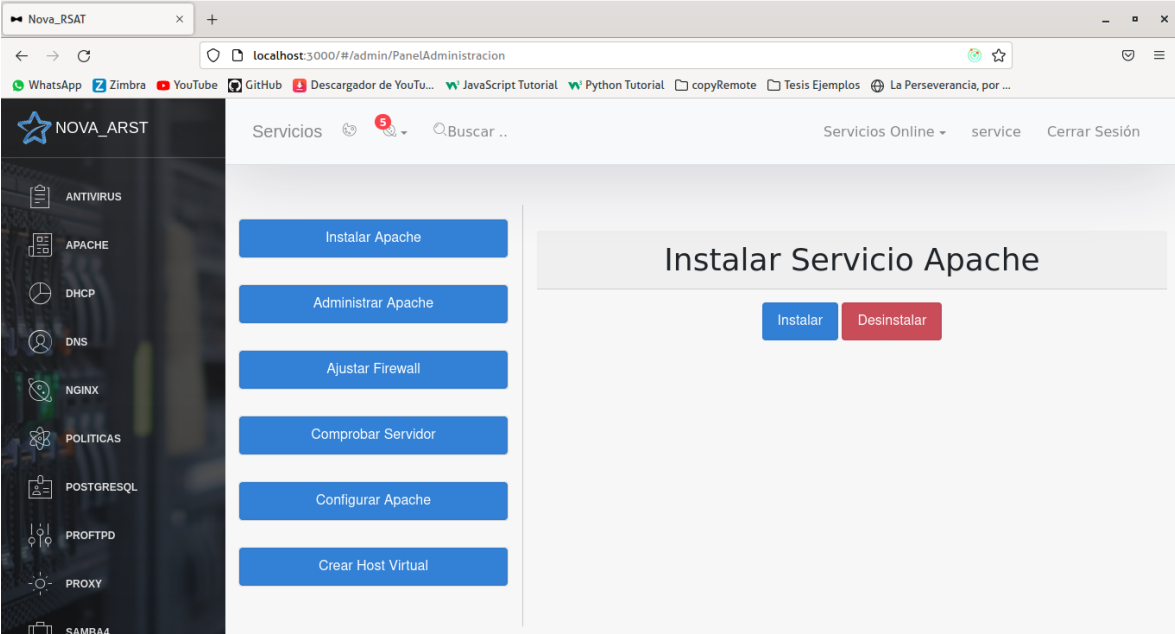
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 20 horas
<p><b>Descripción:</b></p> <p>El módulo permite al usuario instalar el servidor web Apache 2 mediante la plataforma Nova ARST. El usuario selecciona la opción de “Instalar” del menú lateral izquierdo. Al finalizar la instalación se le notificara al usuario mostrando un recuadro con los detalles de la instalación. En caso de que el servidor web Apache 2 este instalado ya, el bloque mostrará la información de la versión instalada.</p>	
<p><b>Prototipo elemental de la interfaz gráfica de usuario:</b></p> 	

Tabla 5 Historia de Usuario: Iniciar Apache 2 (Elaboración propia)

HISTORIA DE USUARIOS	
<b>Número:</b> RF3	<b>Nombre del requisito:</b> Iniciar Apache 2
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 12 horas

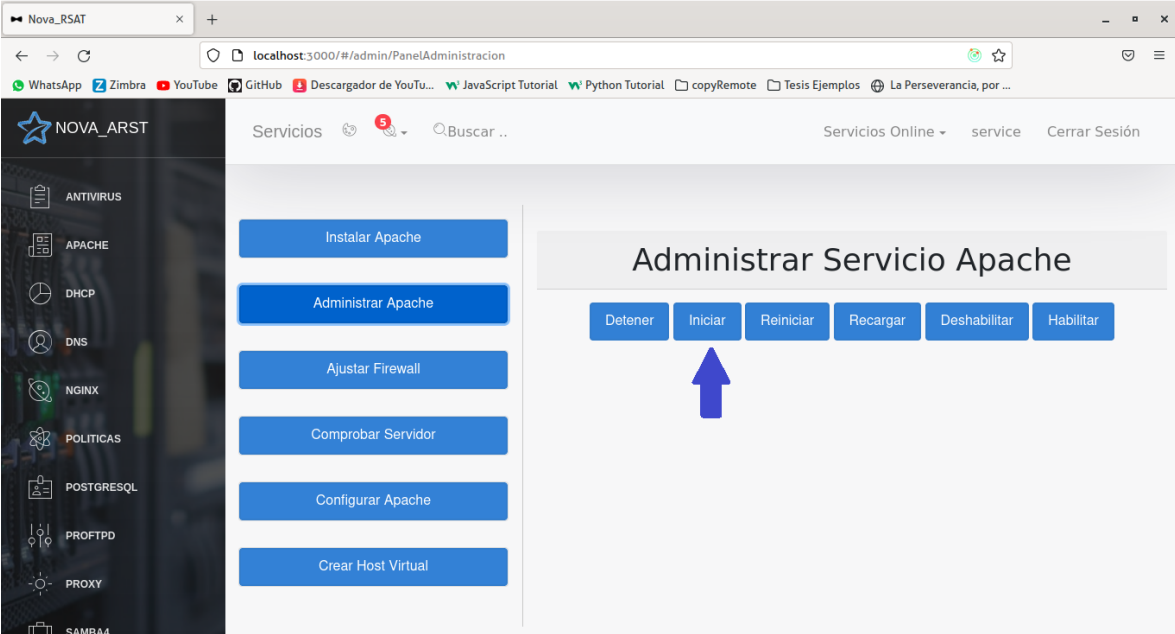
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 15 horas
<p><b>Descripción:</b></p> <p>El módulo permite al usuario iniciar el servicio del servidor web Apache 2. El usuario selecciona la opción de “Administrar Apache 2” del menú lateral izquierdo, luego acciona el botón “Iniciar” y, si el servicio esta inactivo, se inicia el servicio de Apache 2. En caso de que el servicio este iniciado ya y el usuario desee detenerlo, debe seleccionar la opción de “Detener” situada justo al lado del botón “Iniciar”.</p>	
<p><b>Prototipo elemental de la interfaz gráfica de usuario:</b></p> 	

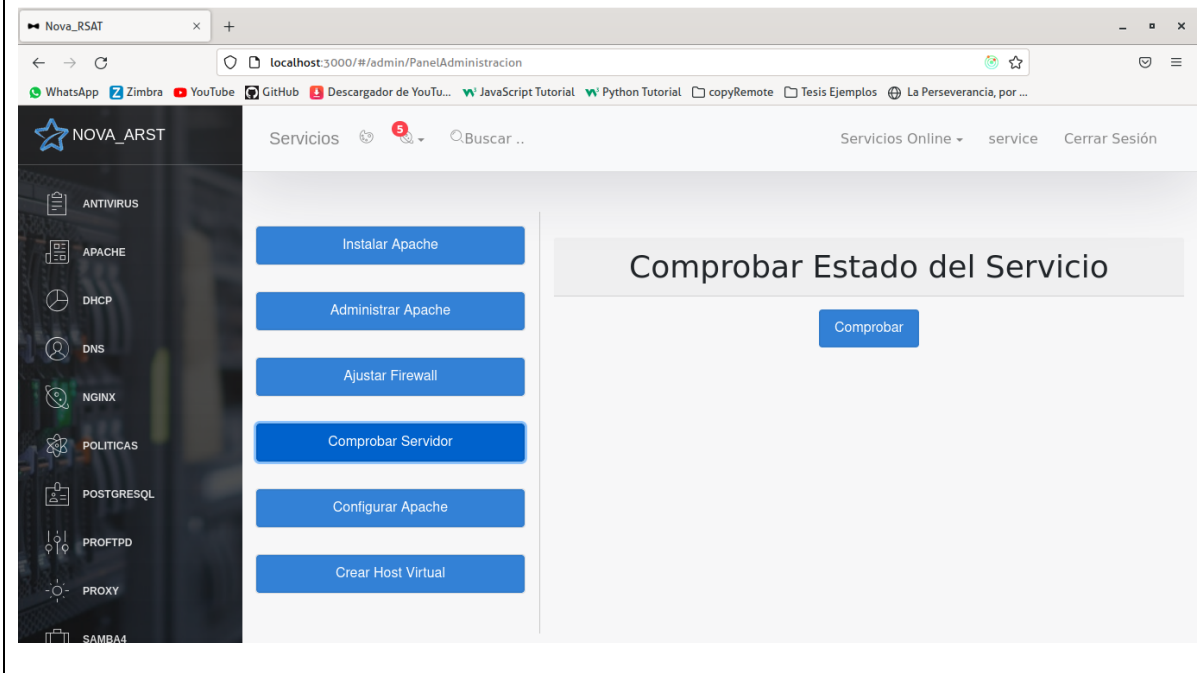
Tabla 6 Historia de Usuario: Comprobar Apache 2 Fuente: elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF10	<b>Nombre del requisito:</b> Comprobar Apache 2
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 10 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 7 horas

## Descripción:

El módulo permite al usuario comprobar el estado del servidor web Apache 2. El usuario selecciona la opción de “Comprobar Servidor” del menú lateral izquierdo, luego acciona el botón “Comprobar” y se muestra un recuadro con el estado actual del servicio de Apache 2.

## Prototipo elemental de la interfaz gráfica de usuario:



## 2.5 Arquitectura del módulo de administración remota para el servidor web Apache 2 en la plataforma Nova ARST.

Para el diseño del módulo se eligió como arquitectura la n-capas, específicamente en este caso con 3 capas basándose en que es la arquitectura con la que se desarrolló la plataforma Nova ARST y justo la arquitectura es uno de los aspectos a tener a cuenta para poder integrar un módulo en dicha plataforma.

La programación web basada en el modelo de desarrollo por capas es un método que permite programar en capas y usa el concepto de la arquitectura cliente servidor, donde las capas están compuestas por presentación, negocio y acceso a datos.



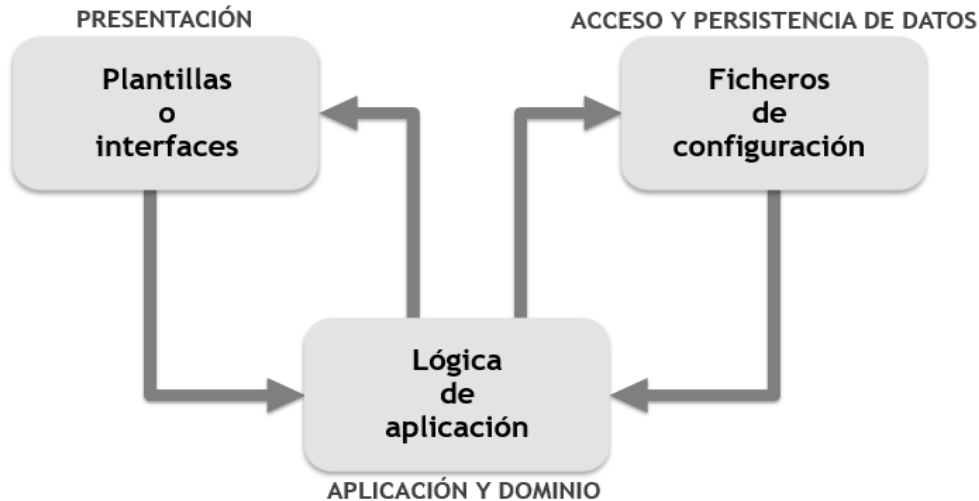


Imagen 3 Arquitectura del módulo Fuente: elaboración propia

El módulo cuenta con una capa de presentación que es donde se encuentran todas las plantillas o interfaces con las que el usuario va a interactuar directamente. Una capa intermedia que es en la que se encuentra toda la lógica de la aplicación, es decir todas las clases controladoras que permiten la comunicación entre las otras dos capas, y la última capa, la capa de acceso a datos que es donde se guardan los datos nuevos o se pueden consultar datos que se encuentren anteriormente introducidos. Esta última capa está integrada por las clases encargadas de manipular todos los ficheros de configuración del servidor Apache 2.

### 2.5.1 Diagrama de paquetes para representar la arquitectura

Un diagrama de paquete es un diagrama de bloques que tiene paquetes cuyo propósito es una organización modelo. Los diagramas de paquetes son diagramas estructurales que se emplean para mostrar la organización y disposición de diversos elementos de un modelo en forma de paquetes. Un paquete es una agrupación de elementos UML relacionados, como diagramas, documentos, clases o, incluso, otros paquetes. Cada elemento está anidado dentro de un paquete, que se representa como una carpeta de archivos dentro del diagrama, y que luego se organiza jerárquicamente dentro del diagrama(27).

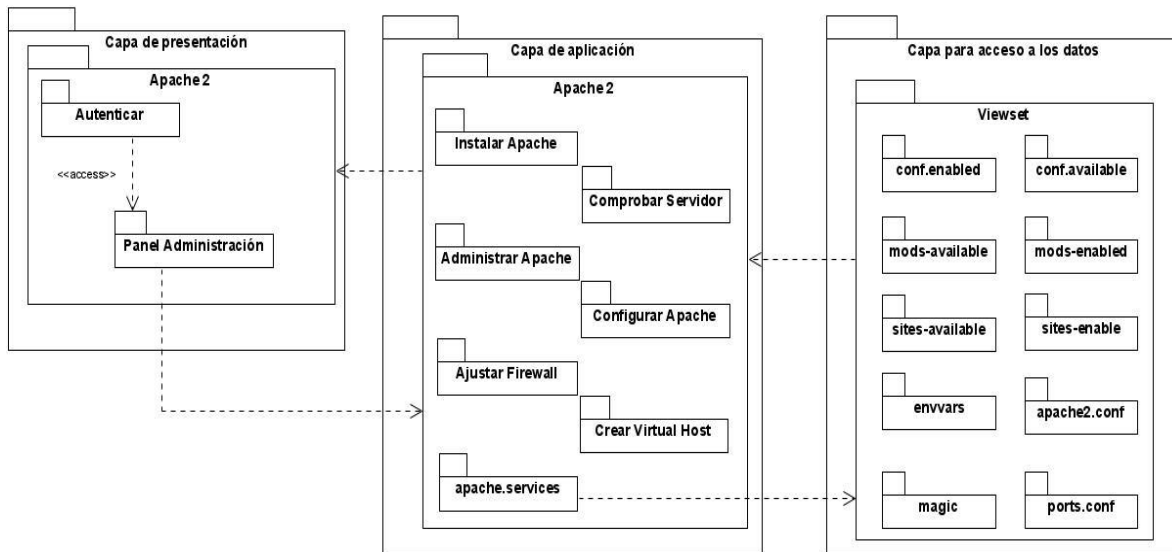


Imagen 4 Diagrama de Paquetes Fuente: elaboración propia

La imagen 3 representa el diagrama de paquetes con las diferentes capas de arquitectura del módulo.

En la **capa de presentación** está el paquete Apache 2 que contiene los paquetes Autenticar y Panel Administración, este último se encarga de implementar los métodos para el envío y recepción de datos para cada una de las vistas del módulo.

En la **capa aplicación**, se encuentra el paquete Apache 2 que contiene los paquetes Instalar Apache, Administrar Apache, Ajustar Firewall, Comprobar Servidor, Configurar Apache y Crear Virtual Host, donde se definen los métodos que se llamarán en la capa presentación. Además, cuenta con el paquete **apache.services** que es el que se encarga de ejecutar los métodos que modificaran los ficheros en la capa de acceso a datos.

La **capa de acceso a datos** es la que contiene todos los ficheros de configuración del servidor web Apache 2, en esta capa es donde se modifican y listan los ficheros y peticiones realizados en la capa de presentación a través de la capa de aplicación.

## 2.5.2 Diagrama de clases

El diagrama de clases es un gráfico que representa el comportamiento del sistema en forma gráfica y es parte del diseño de software. Debe desarrollarse antes de la generación y, a su vez, contar con especificación formal para aplicaciones web, tal como el modelo de desarrollo por capas(28).

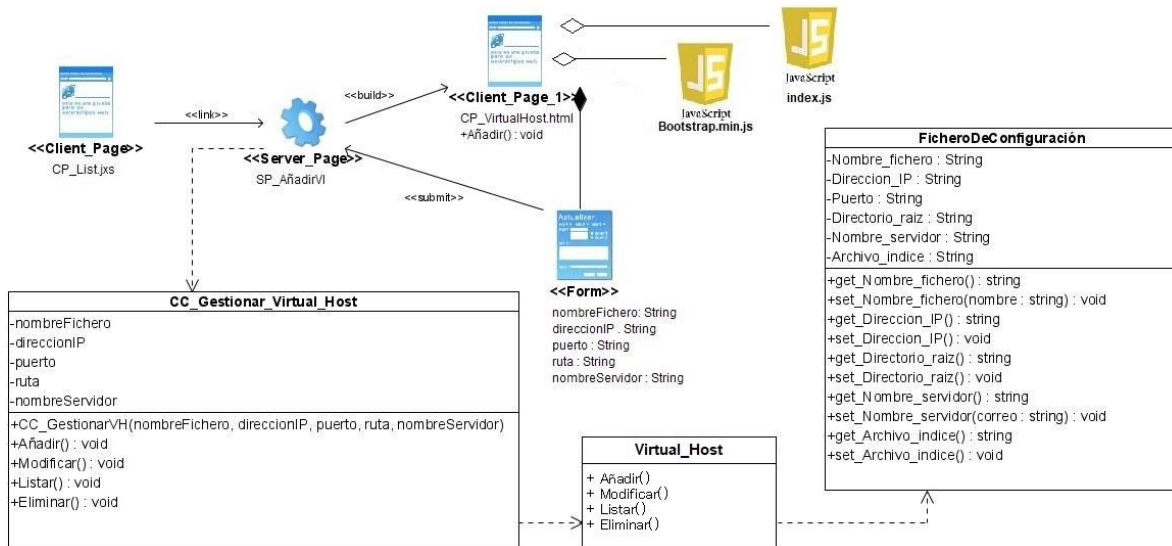


Imagen 5 Diagrama de clases añadir virtual host Fuente: Elaboración propia

La imagen 4 representa el diagrama de clases correspondiente al requisito funcional "Añadir Virtual Host".

## 2.6 Patrones de diseño

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. En el modelo del diseño del módulo se aplican los Patrones Generales de Software para Asignar Responsabilidades (GRASP, del inglés *General Responsibility Assignment Software Patterns*) y GoF (del inglés *Gang of Four*)(2).

### 2.6.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. En el desarrollo del módulo se utilizaron los siguientes patrones GRASP:

**Experto:** se basa en asignar una responsabilidad a la clase que cuenta con la información necesaria para cumplir dicha responsabilidad. Permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para realizar lo que se le oriente (Gutiérrez y Gato 2018). En este caso el patrón se evidencia en la función **def InstallUninstall** la cual se encarga de instalar el servidor web Apache 2, siendo esta función la única en carga de esta tarea.

```
1 def InstallUninstall(self, request, pk=None):
2     cnx = ConnectionSSH(str(IP_SERVER), USER_SERVER, password = PASSWORD_SERVER)
3     outputComand = cnx.execute(f'echo "{PASSWORD_SERVER}" | sudo -S apt install apache2 -y')
4     print("Respuesta ", outputComand)
5     return Response(outputComand)
```

Imagen 6 Fragmento de código donde se evidencia el patron experto Fuente: Visual Studio Code

**Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón se evidencia en la clase **def ServicesCopy** quien se encarga de crear y modificar ficheros de configuración, así como enviarlos a una dirección específica.

```
1 def ServicesCopy(self,request,pk=None):
2     result = request.data
3     nombre = result['apache2']
4     file = open(f"/home/jjbadell/Documentos/{nombre}.conf" , "w")
5     file.write("server{"+ os.linesep)
6     file.write("    listen "+ result['ip']+ " "+ result['puerto']+";"+os.linesep)
7     file.write("    listen [::]:"+result['puerto']+";"+os.linesep)
8     file.write("    root "+ result['directorio']+";"+os.linesep)
9     file.write("    server_name "+ result['nombreServidor']+ ";"+ os.linesep)
10    file.write("}")
11    file.close()
12    cnx = ConnectionSSH(str(IP_SERVER),USER_SERVER, password = PASSWORD_SERVER)
13    result = cnx.put(f"/home/jjbadell/Documentos/{nombre}.conf", f"/home/nova/{nombre}.conf")
14    print("ver result",result)
15    return Response({"res":"Servicio"})
```

Imagen 7 Fragmento de código de la clase def ServicesCopy Fuente: Visual Studio Code

**Bajo Acoplamiento:** Este patrón asigna una responsabilidad para mantener el bajo acoplamiento. La idea es tratar de que una clase no dependa de muchas otras, así esa clase no tendrá muchas dependencias, lo que facilitará la reutilización de la misma y se reducirá el impacto de los cambios. La clase ConnectionSSH.py se desarrolló siguiendo el patrón bajo acoplamiento ya que su funcionamiento no depende de muchas otras clases garantizando así un mejor funcionamiento del módulo.

```
1 class ConnectionSSH(object):
2     """Connects and logs into the specified hostname.
3     Arguments that are not given are guessed from the environment."""
4
5     def __init__(self,
6                 host,
7                 username=None,
8                 private_key=None,
9                 password=None,
10                port=22,
11                ):
12
13         # type: (object, object, object, object, object) -> object
14         self._sftp_live = False
15         self._sftp = None
16         if not username:
17             username = os.environ['LOGNAME']
```

Imagen 8 Fragmento de código de la clase ConnectionSSH Fuente: elaboración propia

**Alta Cohesión:** Una clase con alta cohesión tiene un número relativamente pequeño de métodos, con funcionalidades altamente relacionadas, y no realiza mucho trabajo, colaborando con otros objetos para compartir el esfuerzo si la tarea es extensa. Las clases con alta cohesión son relativamente fáciles de mantener, entender y reutilizar (Gutiérrez y Gato 2018). En este caso también es la clase ConnectionSSH.py ya que esta solo contiene los métodos necesarios para garantizar la conexión con el servidor.

### 2.6.2 Patrones GOF

Los patrones GOF complementan a los patrones GRASP y en ocasiones se puede encontrar una contraposición entre este tipo de patrones, e incluso, podría inferirse que algunos patrones GOF son variantes de los patrones GRASP, es por ello que la decisión de utilizar uno u otro debe tomarse con precaución y aplicarse solo en el ámbito necesario (Gutiérrez & Gato, 2018).

En el desarrollo del módulo se empleó el siguiente patrón GoF:

**Patrón Solitario (Singleton):** es un patrón de tipo creacional con el objetivo de garantizar la existencia de una única instancia para una clase y posibilitar el acceso global a dicha instancia. Su empleo se evidencia cuando se realiza una conexión SSH a un servidor, pues con una única instancia del objeto conexión se realizan las operaciones sobre este. Tal es el caso de la clase ConnectionSSH, donde con una única instancia se realizan todas las operaciones en el servidor.

```
1 def ServicesSSHStop(self, request, pk=None):
2     cnx = ConnectionSSH(str(IP_SERVER), USER_SERVER, password = PASSWORD_SERVER)
3     outputComand = cnx.execute(f'echo "{PASSWORD_SERVER}" | sudo -S service apache2 stop')
```

Imagen 9 Fragmento de código del método ServicesSSHStop Fuente: elaboración propia

### Conclusiones parciales

El análisis y diseño del módulo propuesto para la administración de la plataforma Nova ARST fue posible mediante el levantamiento de requisitos, las historias de usuario, la arquitectura y patrones de diseños seleccionados, permitiendo el entendimiento de las funcionalidades del módulo en cuestión y su comportamiento. Dando, de esta forma, una guía para su desarrollo de una manera más práctica y entendible para futuras versiones del módulo.

## Capítulo 3: Implementación, pruebas y evaluación del módulo para la administración remota del servidor web apache 2 en la plataforma Nova ARST.

La concepción de la propuesta del sistema presentada en el capítulo anterior, ayuda al programador a entender mejor las funcionalidades que busca el cliente y, por tanto, ser capaz de llevarlas a código entendible por la computadora, o lo que es lo mismo implementar el sistema. Durante toda la fase de implementación y después de esta se desarrollan un conjunto de pruebas con el objetivo de asegurar la calidad del módulo y que el mismo cumpla con todas las peticiones del cliente. Como principales elementos en este capítulo se especifican los casos de pruebas aplicados a la solución desarrollada para validar su correcto funcionamiento.

### 3.1 Implementación

La implementación es un proceso que comienza cuando se crea una aplicación en un equipo de desarrollo y termina cuando está instalada y lista para ejecutarse en el equipo de un usuario. Describe cómo los elementos del modelo de diseño se implementan en términos de componentes, en otras palabras, toma el resultado del modelo de diseño para generar el código final del sistema. Dicho código está determinado por el lenguaje de programación y tiene como objetivo llevar a cabo la implementación de cada una de las clases significativas del diseño.

### 3.2 Estándares de implementación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, este debe tender siempre a lo práctico. Un código completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez.

Para facilitar el entendimiento del código y fijar un modelo a seguir, se establecen estándares de codificación.

#### Reglas de indentación en PEP 8

Python define la lógica de su código utilizando bloques indentados, por lo tanto, se recomienda indentar usando 4 espacios, dado que los espacios ocupan el mismo tamaño en casi todos los tipos de fuente y 4 es un número aceptable para la separación visual de los bloques. Como se observa en la figura 4.

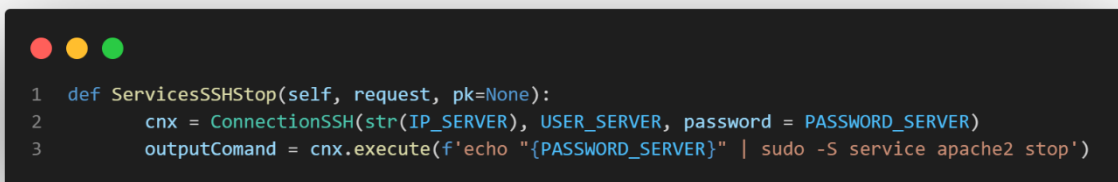
A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in Python and demonstrates indentation. Line 1 is a function definition: `def ServicesSSHStop(self, request, pk=None):`. Line 2 is an indented assignment: `cnx = ConnectionSSH(str(IP_SERVER), USER_SERVER, password = PASSWORD_SERVER)`. Line 3 is an indented assignment: `outputComand = cnx.execute(f'echo "{PASSWORD_SERVER}" | sudo -S service apache2 stop')`. The indentation for lines 2 and 3 is consistent, using four spaces.

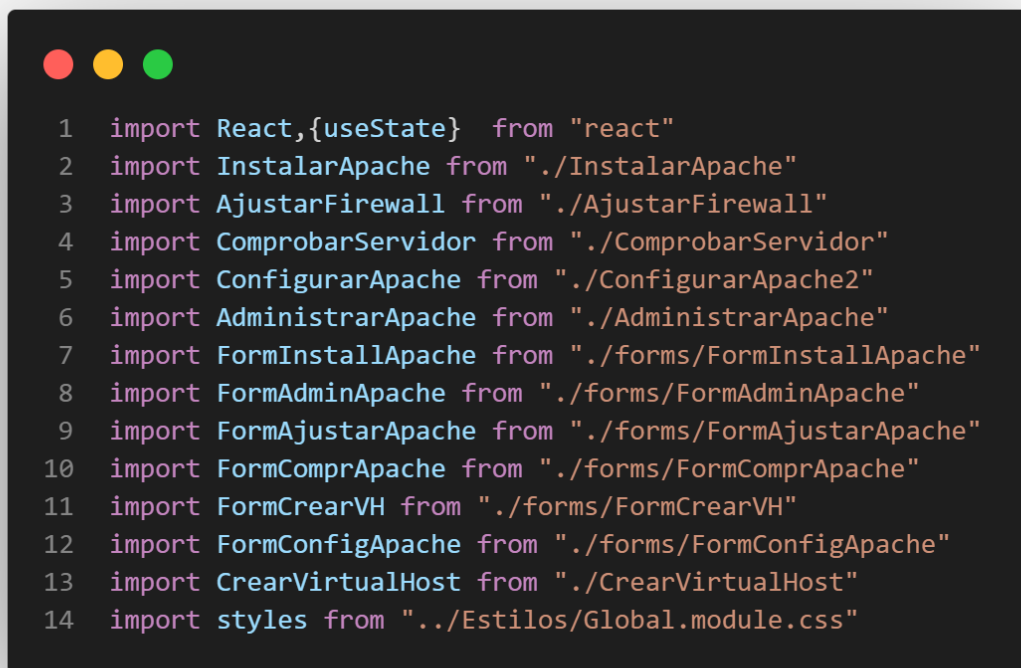
Imagen 10 Estándares de Python Fuente: Visual Studio Code



## Importaciones de código

Para las importaciones de paquetes existen reglas.

- El uso de *import* debe de ser en líneas separadas.
- El orden de importación es: librería estándar, librerías de terceros, librerías propias o locales.
- Las importaciones absolutas son los recomendados.
- Las importaciones relativas solo pueden ser explícitas y preferiblemente simples y cortas.
- Importaciones usando \* (*wildcard*) deben de ser evitadas.



```
1 import React,{useState} from "react"
2 import InstalarApache from "./InstalarApache"
3 import AjustarFirewall from "./AjustarFirewall"
4 import ComprobarServidor from "./ComprobarServidor"
5 import ConfigurarApache from "./ConfigurarApache2"
6 import AdministrarApache from "./AdministrarApache"
7 import FormInstallApache from "./forms/FormInstallApache"
8 import FormAdminApache from "./forms/FormAdminApache"
9 import FormAjustarApache from "./forms/FormAjustarApache"
10 import FormComprApache from "./forms/FormComprApache"
11 import FormCrearVH from "./forms/FormCrearVH"
12 import FormConfigApache from "./forms/FormConfigApache"
13 import CrearVirtualHost from "./CrearVirtualHost"
14 import styles from "../Estilos/Global.module.css"
```

Imagen 11 Importaciones en React Fuente: Visual Studio Code

La PEP 8 contempla muchas reglas sobre cómo tratar los comentarios que se pueden ver a continuación:

- Los comentarios pueden ser contradictorios, y requieren que se actualicen con el código, por tanto, si se pueden evitar haciendo código más claro, mejor.
- Deben de componerse de frases completas.
- Deben de comenzar en mayúscula a no ser que comiencen con el nombre de un identificador, el cual se respetará siempre su tamaño.
- Los bloques de comentarios se componen de varios párrafos terminados en puntos, y terminar en dos espacios en blancos salvo la última frase.
- Los comentarios deben de ser en Ingles a no ser «que se esté 120%» seguro de que se leerán siempre en otro idioma y serán claros.

- Los bloques se aplican al código justo después de donde se encuentran y con la misma indentación.
- Los comentarios de línea comienzan con al menos 2 espacios y un # y un espacio después.

```
1 # Metodo para desinstalar apache 2
2 @action(methods=['get'], detail=False, url_path='sshUninstall', url_name='sshUninstall')
3 def UninstallApache(self, request, pk=None):
4     conexion = ConnectionSSH(str(IP_SERVER),USER_SERVER, password = PASSWORD_SERVER)
5     outputComand = conexion.execute(f'echo "{PASSWORD_SERVER}" | sudo -S apt-get purge apache2 -y')
6     print("Respuesta ", outputComand)
7     return Response(outputComand)
```

Imagen 12 Como comentar en Python Fuente: Visual Studio Code

### Convenciones de nombres

Las siguientes reglas aplican para los nombres de cada tipo de objeto.

- Caracteres a evitar: o, O, l y L (letra O y letra L) dado que se pueden confundir con un 0 o un 1 dependiendo del tipo de fuente usada.
- Caracteres ASCII: se utilizan en la librería estándar siempre.
- Nombre de módulos y paquetes: se escriben en minúsculas y se desaconseja el uso de barras bajas, manteniendo los nombres lo más cortos posibles.
- Nombrado de clases: se usan el Formato Capital, donde la primera letra de cada palabra es mayúscula.
- Nombres de tipos de variables: en Formato Capital.
- Excepciones: igual que el de clases, pero con el sufijo «Error».
- Nombres de variables y funciones: se usan palabras en minúsculas separadas por barras bajas.
- Métodos en clases: el primer parámetro para métodos de instancia es self y para métodos de clase cls.
- Constantes: en mayúsculas.

```
1 def UninstallApache(self, request, pk=None):
2     conexion = ConnectionSSH(str(IP_SERVER),USER_SERVER, password = PASSWORD_SERVER)
3     outputComand = conexion.execute(f'echo "{PASSWORD_SERVER}" | sudo -S apt-get purge apache2 -y')
4     print("Respuesta ", outputComand)
5     return Response(outputComand)
```

Imagen 13 Reglas de Python Fuente: Visual Studio Code

## 3.3 Diagrama de Despliegue

El modelo de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Los nodos representan objetos físicos existentes en tiempo de ejecución, sirven para modelar recursos que tiene memoria y capacidad de proceso y puede ser tanto ordenadores como dispositivos, memoria o personas

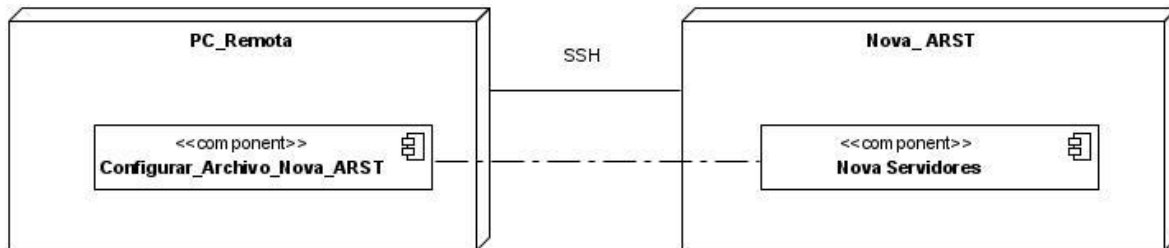


Imagen 14 Diagrama de Despliegue Fuente: Elaboración propia

Descripción:

**PC Remota:** Esta PC de escritorio contiene el componente (Configuración de archivos para Nova ARST) el cual es el componente que permite realizar las configuraciones del sistema de archivos de manera remota utilizando Secure Shell (SSH) en español cubierta segura. El protocolo SSH trabaja como el parámetro de seguridad que procura la administración de los aspectos relacionados a la conexión entre redes. Es bien conocido ya que a través de los servidores se realiza una constante transmisión de información desde un computador a otro. Desde SSH se pueden realizar copias seguras de los archivos que se transmiten a través de la dirección seleccionada, en caso de necesitarlas. Por defecto SSH utiliza el protocolo TCP de la capa de transporte.

**Nova ARST:** Es una herramienta desarrollada por el Centro de Software Liber SECOL que permite, entre otras funcionalidades, la administración y configuración remota del servidor.

**Nova Servidores:** Distribución de GNU/Linux orientada a servidores usando estándares abiertos y adaptada a los entornos de las empresas cubanas. Entre sus características se encuentran la configuración fácil e intuitiva a través de la herramienta nova-manager, destinada a la administración de los servicios telemáticos y la compatibilidad con hardware obsoleto en el entorno empresarial.

## 3.4 Diagrama de Componentes

Los diagramas de componentes representan las relaciones entre los componentes individuales del sistema mediante una vista de diseño estática. Pueden ilustrar aspectos de modelado lógico y físico. En el contexto del UML, los componentes son partes modulares de un sistema independientes entre sí, que pueden remplazarse con componentes equivalentes. Son auto contenidos y encapsulan estructuras de cualquier grado de complejidad. Los elementos encapsulados solo se comunican con otros a través de interfaces. Los componentes no solo pueden proporcionar sus propias interfaces, sino que también pueden utilizar las interfaces de otros componentes, por ejemplo. Para acceder a sus funciones y servicios. A su vez, las interfaces de un diagrama de componentes, por

ejemplo, para acceder a sus funciones y servicios. A su vez, las interfaces de un diagrama de componentes documentan las relaciones y dependencias en una arquitectura de software.

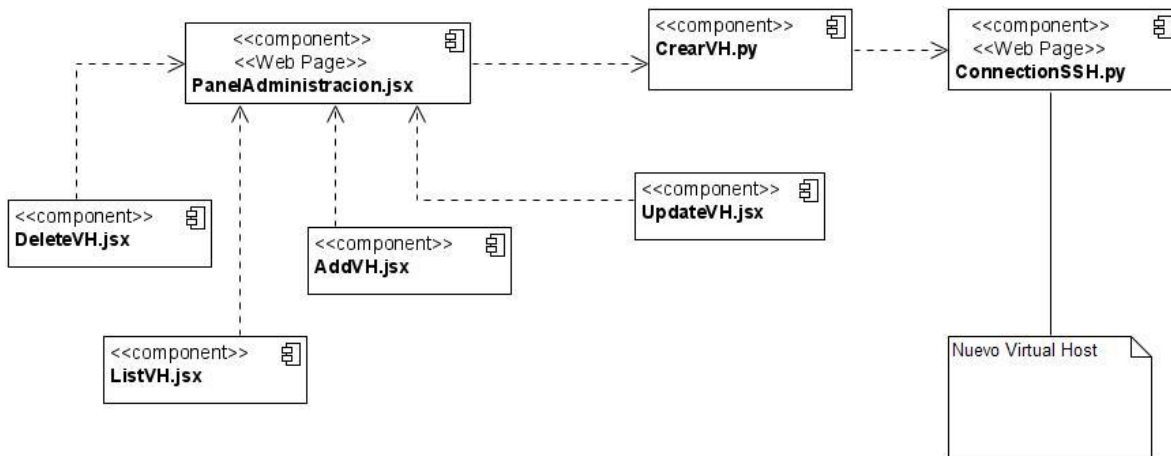


Imagen 15 Diagrama de componentes Fuente: Elaboración propia

A continuación, en la tabla 7 se describen los elementos que componen el diagrama de componentes mostrado:

Tabla 7 Tabla descriptiva del diagrama de componentes Fuente: Elaboración propia

		Descripción
<b>Modelo</b>	<b>NuevoVirtualHost</b>	Representa el fichero que corresponde a cada virtual host creado.
<b>Vistas</b>	<b>AddVH.jsx</b>	Esta clase contiene el formulario que se le muestra al usuario para crear un virtual host.
	<b>ListVH.jsx</b>	Lista del virtual host creados y para su eliminación.
	<b>UpdateVH.jsx</b>	Esta clase contiene un formulario que se le muestra al para modificar un virtual host.
	<b>DeleteVH.jsx</b>	Elimina un virtual host creado.
	<b>PanelAdministracion.jsx</b>	Página principal del módulo.

<b>Template</b>	<b>CrearVH.py</b>	En esta clase permite realizar todas las acciones de administración de los virtual host.
	<b>ConnectionSSH.py</b>	La clase que proporciona la conexión mediante el protocolo SSH con el servidor.

## 3.5 Pruebas de software

Las pruebas de software son una parte integral del ciclo de vida del desarrollo de software (SDLC). Las pruebas son la forma en que puede estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario. Ya sea que realice sus pruebas manualmente o a través de la automatización, cuanto antes y más a menudo pueda llevar a cabo pruebas, más probable es que identifique errores y errores, no sólo ahorrándole a usted y a su equipo de posibles simulacros de incendio más adelante, sino también asegurándose de que su aplicación de software haya sido revisada y auditada a fondo antes de que esté frente a sus usuarios. Si los problemas se arrastran al entorno de producción, los más caros y lentos que van a solucionar(29).

### 3.5.1 Tipos de pruebas

Las pruebas de software se pueden dividir en dos tipos diferentes: pruebas funcionales y no funcionales. Diferentes aspectos de una aplicación de software requieren diferentes tipos de pruebas, como pruebas de rendimiento, pruebas de escalabilidad, pruebas de integración, pruebas unitarias entre otras. Cada uno de estos tipos de pruebas de software ofrece una excelente visibilidad de la aplicación, desde el código hasta la experiencia del usuario.

#### **Pruebas funcionales**

Las pruebas funcionales se llevan a cabo para comprobar las características críticas para el negocio, la funcionalidad y la usabilidad. Las pruebas funcionales garantizan que las características y funcionalidades del software se comportan según lo esperado sin ningún problema. Los tipos de pruebas funcionales incluyen pruebas unitarias, pruebas de interfaz, pruebas de regresión, entre otras.

#### **Pruebas no funcionales**

Las pruebas no funcionales son como pruebas funcionales; sin embargo, la principal diferencia es que esas funciones se prueban bajo carga para el rendimiento de los observadores, fiabilidad, usabilidad y escalabilidad. Las pruebas no funcionales, como las pruebas de carga y esfuerzo, normalmente se llevan a cabo mediante herramientas y soluciones de automatización. Además de las pruebas de rendimiento, los tipos de pruebas no funcionales incluyen pruebas de instalación, pruebas de confiabilidad y pruebas de seguridad(29).

### 3.5.2 Métodos de pruebas

A medida que las empresas de tecnología moderna maduran, mediante la adopción de prácticas de integración continua, se está otorgando un nivel de importancia cada vez mayor a las pruebas y la automatización de pruebas. Entre los tipos más comunes de pruebas de software, existen pruebas de alto nivel: prueba unitaria, prueba de integración, prueba de extremo a extremo (E2E, sistema), prueba de aceptación, prueba de caja blanca (estructural, transparente), prueba de caja negra (funcional, conductual, caja cerrada), prueba de caja gris, prueba manual, prueba estática, pruebas dinámicas, pruebas visuales o de interfaz de usuario, prueba de humo, prueba de carga, prueba de regresión y pruebas de inserción(29). Existen muchos términos diferentes que se usan en las pruebas de software.

Para realizar una correcta comprobación acerca de la funcionalidad, el rendimiento y la experiencia del usuario, se realizaron las siguientes pruebas(29):

- **Pruebas unitarias:** Las pruebas unitarias se centran en probar piezas/unidades individuales de una aplicación de software al principio del SDLC. Cualquier función, procedimiento, método o módulo puede ser una unidad que se someta a pruebas unitarias para determinar su corrección y comportamiento esperado. Las pruebas unitarias son las primeras pruebas que los desarrolladores realizan durante la fase de desarrollo.
- **Pruebas de integración:** Las pruebas de integración implican probar diferentes módulos de una aplicación de software como grupo. Una aplicación de software se compone de diferentes submódulos que trabajan juntos para diferentes funcionalidades. El propósito de las pruebas de integración es validar la integración de diferentes módulos juntos e identificar los errores y problemas relacionados con ellos.
- **Pruebas funcionales:** Las pruebas funcionales se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción. A veces, se confunden las pruebas de integración con las funcionales, ya que ambas requieren que varios componentes interactúen entre sí. La diferencia es que una prueba de integración puede simplemente verificar que puedes hacer consultas en la base de datos, mientras que una prueba funcional esperaría obtener un valor específico desde la base de datos, según dicten los requisitos del producto.

### 3.5.3 Aplicación de las pruebas de software

En el presente sub epígrafe se representarán con ejemplos las pruebas realizadas al módulo para la administración remota del servidor web Apache 2 en la plataforma Nova ARST.

#### **Pruebas unitarias**

A continuación, se muestran una de las pruebas unitarias realizadas, donde para su realización se utilizó como herramienta la librería **pytest** de Python y comprobar su correcto funcionamiento.

```

1 def InstallUninstall(self, request, pk=None):
2     cnx = ConnectionSSH(str(IP_SERVER), USER_SERVER, password = PASSWORD_SERVER)
3     outputComand = cnx.execute(f'echo "{PASSWORD_SERVER}" | sudo -S apt install apache2 -y')
4     print("Respuesta ", outputComand)
5     return Response(outputComand)

```

Imagen 16 Prueba unitaria método InstallUninstall Fuente: Visual Studio Code

```

C:\Windows\System32\cmd.exe
H:\zzz>pytest
===== test session starts =====
platform win32 -- Python 3.7.7rc1, pytest-7.2.0, pluggy-1.0.0
rootdir: H:\zzz
collected 1 item

test_api.py . [100%]

===== 1 passed in 0.06s =====
H:\zzz>

```

Imagen 17 Resultado de la prueba unitaria al método InstallUninstall

En la sección de anexos se encuentran las pruebas unitarias restantes.

Luego de realizadas las pruebas unitarias se obtuvo como resultado de trabajo de funcionalidades del módulo para la administración remota del servidor web Apache 2 en la plataforma Nova ARST es correcto, se comprobó que cada sentencia del código se ejecuta al menos una vez.

### Pruebas funcionales

A continuación, en la tabla 8 se presentan los casos de prueba para las Historias de Usuarios Comprobar Apache 2 y Instalar Apache 2 utilizando el método de caja negra bajo la técnica de partición de equivalencia.

Tabla 8 Caso de prueba Comprobar Apache 2 Fuente: elaboración propia

Escenario	Descripción	Respuesta del sistema	Flujo central

EC1 Mostrar estado del servidor web Apache 2.	Mostrar los resultados de la petición realizada a Apache 2 a través de la interfaz de la plataforma Nova ARST.	Se mostrará el estado del servidor web Apache 2.	Ejecutar el comando para hacer la petición al servidor.
EC1.2 Mostrar el estado del servidor web Apache sin conexión remota.	Mostrar los resultados de la petición realizada a Apache 2 a través de la interfaz de la plataforma Nova ARST.	No se mostrará el estado del servidor web Apache 2.	No se podrá ejecutar el comando para hacer la petición al servidor y se mostrará un mensaje de error de conexión.

Tabla 9 Caso de prueba para Listar Virtual Host Fuente: elaboración propia

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Instalar Apache 2	Permite realizar la instalación del servidor web Apache 2.	Alerta: La instalación ha finalizado. Apache 2 se instaló correctamente.	Seleccionar el botón Aceptar.
EC 1.2 Apache ya está instalado	En el caso de que Apache 2 ya este instalado en el sistema se podrá administrar.	Mensaje: Apache 2 está instalado y funcionando. Por favor escoja una de las siguientes opciones.	Seleccionar el botón "Iniciar"

### 3.5.4 Resultado de las pruebas funcionales realizadas

Se detectaron problemas en el período de pruebas de validación. Para una mayor organización y entendimiento se clasificaron estos problemas como no conformidades relevantes e irrelevantes. Esta clasificación esta justificada por los siguientes aspectos:

**No conformidades relevantes:** son las no conformidades referentes a las funcionalidades del módulo desarrollado: validaciones incorrectas o respuestas diferentes a las descritas con anterioridad.

**No conformidades irrelevantes:** son las no conformidades referentes al diseño de la propuesta solución y errores ortográficos.



Se realizaron dos iteraciones de pruebas, en la primera iteración se identificaron siete no conformidades, de ellas, cuatro eran relevantes y tres irrelevantes. Se lograron resolver todas las no conformidades en la primera iteración.

### No conformidades relevantes

- El módulo no detiene el servicio de Apache 2 al accionar el botón ``Detener``.
- El módulo no muestra los resultados de la comprobación del estado del servicio Apache 2 al accionar el botón ``Comprobar``.
- El módulo no muestra las notificaciones al finalizar una petición.
- El recuadro donde se muestra la información del servicio aparece en blanco.

### No conformidades irrelevantes

- El módulo tiene errores ortográficos en la sección ``Administrar Apache 2``.
- Las notificaciones tienen una tipografía ilegible.
- Los botones tienen diferentes tamaños y diseños.

En la segunda iteración no se detectaron no conformidades.

La gráfica siguiente, resume los resultados obtenidos en la realización de las pruebas:



Imagen 18 Resultado de las pruebas funcionales Fuente: elaboración propia

### Pruebas de integración

A partir del requisito funcional 1: Instalar Apache 2, se deben tener en cuentas los siguientes aspectos:

1. Como la solución será integrada a una plataforma web no es necesario definir un sistema operativo para el ordenador desde donde se accederá a Nova ARST.

## *Capítulo 3*

2. Es necesario contar en su dispositivo un navegador web (Mozilla Firefox, Google Chrome, Safari, Microsoft Edge, entre otros).
3. Verificar que se cuenta con las credenciales necesarias para acceder al panel de administración de la plataforma.
4. Verificar que se tiene conexión con el ordenador donde se desea realizar la instalación del servicio.
5. Verificar que se tiene acceso al ordenador donde se desea instalar el servicio (usuario, contraseña y dirección ip del ordenador).
6. En el panel de administración, seleccionar la opción de instalar apache 2.

A partir de lo anterior se demuestra (ver Anexos) que la solución desarrollada por el investigador resuelve el problema planteado.

Al finalizar la realización de las pruebas de integración se concluye que:

- La ejecución de las pruebas de integración permitió verificar el trabajo conjunto de los componentes del sistema en cuestión.
- No se encontró ninguna no conformidad. Se llega a la conclusión que existe una correcta integración entre los componentes internos del sistema.

### Conclusiones parciales

El uso de los estándares de codificación definidos permitió desarrollar un código reutilizable y fácil de entender. La elección de la estrategia de pruebas con un enfoque incremental propició comprobar el módulo en cada una de sus partes y las relaciones entre ellas. Con la realización de las pruebas se verificó que todas las instrucciones del módulo se ejecutan al menos una vez, que los componentes se integran correctamente, se validó que el módulo se ajusta al sistema.

## Conclusiones generales

Una vez finalizada la investigación se desarrolló un módulo para la administración remota del servidor web apache 2 en la plataforma Nova ARST, considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación se arriban a las siguientes conclusiones:

La realización del marco teórico de proceso de administración de Apache 2, permitió comprender los principales conceptos para el desarrollo de un módulo para la administración remota de Apache 2 e integrarlo a la plataforma Nova ARST.

Los sistemas existentes para la administración remota del servidor web Apache 2, que se estudiaron, no representaron una solución al problema, pero sirvieron de base para elaborar una idea general de cómo se deseaba representar la solución final.

El modelo conceptual enmarcó el contexto donde se usará el módulo y permitió la captura de requisitos funcionales y no funcionales, que fueron agrupados en historias de usuario generando una visión para la creación de un módulo de administración remota del servidor web Apache 2 en la plataforma Nova ARST.

La implementación del módulo para la administración remota del servidor web apache 2 en la plataforma Nova ARST, permitió darles respuesta a los requisitos definidos y con ello a las necesidades del cliente.

La ejecución de pruebas al módulo para la administración remota del servidor web apache 2 en la plataforma Nova ARST, siguió una estrategia de pruebas, que contempla pruebas unitarias, pruebas funcionales y pruebas de integración, arrojando como resultado un total de siete no conformidades, corregidas en dos iteraciones, lo cual permitió la validación de los requisitos identificados y la aceptación por parte del cliente.

El módulo desarrollado tiene un gran aporte empresarial, debido a que en cualquier empresa cubana donde se necesite la administración de red, se puede desplegar la plataforma y hacer uso del módulo para la administración remota de Apache 2, facilitando y automatizando la misma.

## Recomendaciones

Se recomienda para próximas versiones del producto que se le pueda añadir a las configuraciones de Apache la opción de añadir, desactivar y activar módulos MPM.

# Referencias Bibliográficas

## Referencias

1. **Free Software Foundation.** GNU. [En línea] 28 de 6 de 2022. [Citado el: 15 de 11 de 2022.] <https://www.gnu.org/philosophy/free-sw.es.html>.
2. **Joven Club.** jovenclub.com. [En línea] 19 de 8 de 2021. [Citado el: 20 de 9 de 2022.] <https://www.jovenclub.cu/que-es-el-software-libre-caracteristicas-y-ventajas/>.
3. **Pérez, Nurisel Palma.** *Solución Informática para la selección de Apache 2 y Nginx durante la migración a código abierto.* La Habana : s.n., 2019. Tesis Maestría.
4. **Chapaval, Nicole.** Platzi. [En línea] <https://platzi.com/blog/que-es-frontend-y-backend/>.
5. **Rodríguez, Rachel Molina.** *Módulo para administrar el servidor web Nginx desde la Migración y Administración de Servicios Telemáticos.* 2017.
6. **¿Para qué sirve Python? Razones para utilizar este lenguaje de programación. Visus, Andrés.** 2020, Esic, págs. <https://www.esic.edu/rethink/tecnologia/para-que-sirve-python#:~:text=El%20lenguaje%20de%20programaci%C3%B3n%20Python,aplicaciones%20empresariales%20fiables%20y%20escalables>.
7. **Edix. Edix.** [En línea] 26 de 7 de 2022. <https://www.edix.com/es/instituto/framework/>.
8. **Microsoft 365 Team. Microsoft. Microsoft.** [En línea] 24 de 9 de 2019. <https://www.microsoft.com/es-ww/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling#:~:text=El%20Lenguaje%20Unificado%20de%20Modelado,de%20un%20sistema%20o%20proceso..>
9. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico. Quinta edición.* S.I. s.l. : McGraw-Hill Companies, 2002.
10. **Douglass, Powel.** *Agile Systems Engineering.* 2016.
11. **Introducción al análisis y diseño orientado a objetos.** LARMAN, C. México : s.n., 1999, UML y Patrones.
12. **Plataforma para la integración de componentes en un Sistema de Laboratorio Remoto.** Gutiérrez, Eric y Gato, Yisell. Septiembre de 2018, Serie Científica de la Universidad de las Ciencias Informáticas, págs. 33-49.
13. **E.V.A., UCI.** Conferencia#5 Modelo Cliente-Servidor. Teleinformática II. [En línea] <http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml>.
14. **Thanh, CircleCi - Vinny.** [En línea] 2020. <https://circleci.com/blog/testing-methods-all-developers-should-know/>.
15. **Vargas, Sergio. Aplicatta.** [En línea] <https://www.appicatta.cl/index.php/soluciones/metodologia-appicatta/analisis-y-diseno>.

## *Referencias Bibliográficas*

16. Krall, César. *Aprende a Programar*. [En línea] 2020. [https://www.aprenderaprogramar.com/index.php?option=com\\_content&view=article&id=688:i-que-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163](https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:i-que-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163).
17. *Propuesta de Estándar de desarrollo o codificación (Primera Entrega)*. Arevalo, María. 2012.
18. IONOS. *Digital Guide*. [En línea] 23 de Septiembre de 2020. [Citado el: 30 de Septiembre de 2022.] <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-componentes/>.

## Anexos

### Anexo 1: Entrevista

Entrevista realizada al personal que desarrollo la plataforma Nova ARST.  
Buenas, usted ha sido seleccionado para realizarle una entrevista para conocer más sobre la plataforma Nova ARST. Es necesario responda las siguientes preguntas basándose en su experiencia práctica y teórica como desarrollador.

Pregunta 1: ¿Dónde se puede encontrar documentación sobre la plataforma?

Pregunta 2: ¿Cómo se realiza la conexión remota a otra PC en la plataforma?

Pregunta 3: ¿Qué parámetro debe de cumplir un módulo para integrarse a la plataforma?

Pregunta 4: ¿Cada cuánto tiempo se actualiza el estado de la plataforma?

Entrevista realizada al personal que de administra redes en el centro CESOL.  
Usted ha sido seleccionado para realizarle una entrevista referente al proceso de administración en el centro CESOL.

Pregunta 1: ¿Usted usa Apache 2 como servidor web?

Pregunta 2: ¿Cómo es el proceso de administración de Apache 2?

Pregunta 3: ¿Cuánto tiempo le toma configurar una PC servidora?

Pregunta 4: ¿Qué tipo de configuraciones son las más frecuentes?

### Anexo 2: Documentación sobre el proceso de administración remota de servidores web.

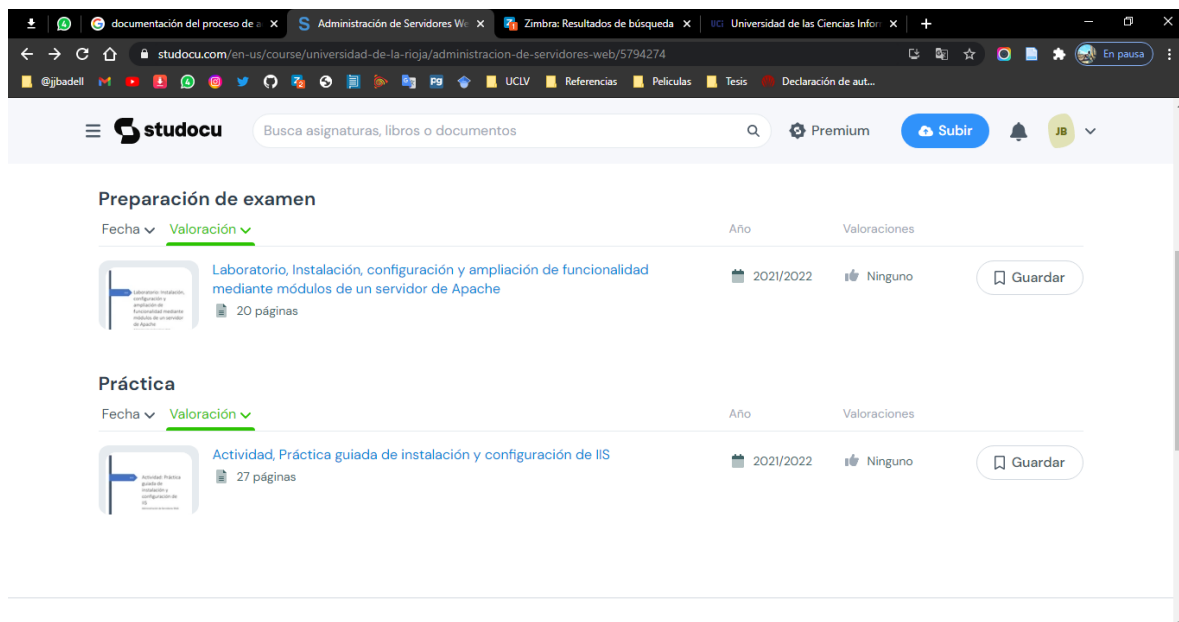


Imagen 19 Documentación sobre proceso de administración remota de servidores web Fuente: Studocu



Para acceder al enlace del sitio donde se encuentra la documentación consultar referencia(30).

### Anexo 3: Historias de Usuario

Tabla 10 Historia de Usuario Desinstalar Apache 2 Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF2	<b>Nombre del requisito:</b> desinstalar Apache 2.
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 26 horas
<b>Descripción:</b> El módulo permite al usuario desinstalar el servidor web Apache 2 mediante la plataforma Nova ARST. El usuario selecciona la opción de “Instalar” del menú lateral izquierdo y luego acciona el boto Desinstalar. Al finalizar la desinstalación se le notificara al usuario mostrando un recuadro con los detalles de la desinstalación. En caso de que el servidor web Apache 2 no este instalado, el bloque mostrará un mensaje de que no se puede desinstalar apache 2 porque no existe le fichero.	

Tabla 11 Historia de Usuario Detener Apache 2 Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF4	<b>Nombre del requisito:</b> Detener Apache 2.
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 20 horas
<b>Descripción:</b> El módulo permite al usuario detener el servicio del servidor web Apache 2. El usuario selecciona la opción de “Administrar Apache 2” del menú lateral izquierdo, luego acciona el botón “Detener” y, si el servicio esta activo, se detiene el servicio de Apache 2.	

Tabla 12 Historia de Usuario Reiniciar Apache 2 Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF5	<b>Nombre del requisito:</b> Reiniciar Apache 2
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 20 horas
<b>Descripción:</b> El módulo permite al usuario reiniciar el servicio del servidor web Apache 2. El usuario selecciona la opción de “Administrar Apache 2” del menú lateral izquierdo, luego acciona el botón “Reiniciar”.	

Tabla 13 Historia de Usuario Recargar Apache 2 Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF6	<b>Nombre del requisito:</b> Recargar Apache 2
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 20 horas
<b>Descripción:</b> El módulo permite al usuario recargar el servicio del servidor web Apache 2. El usuario selecciona la opción de “Administrar Apache 2” del menú lateral izquierdo, luego acciona el botón “Recargar”.	

Tabla 14 Historia de Usuario Añadir host virtual Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF7	<b>Nombre del requisito:</b> Añadir host virtual
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2

<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 25 horas
<b>Descripción:</b> <p>El módulo permite al usuario añadir un nuevo host virtual el servicio del servidor web Apache 2. El usuario selecciona la opción de “Crear Host Virtual” del menú lateral izquierdo, luego acciona el botón “Host Virtuales” y llena el formulario para añadir un nuevo host.</p>	

Tabla 15 Historia de Usuario Deshabilitar Apache 2 Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF8	<b>Nombre del requisito:</b> Deshabilitar Apache 2
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 15 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 18 horas
<b>Descripción:</b> <p>El módulo permite al usuario deshabilitar el servicio del servidor web Apache 2. El usuario selecciona la opción de “Administrar Apache 2” del menú lateral izquierdo, luego acciona el botón “Deshabilitar”.</p>	

Tabla 16 Historia de Usuario Habilitar Apache 2 Fuente: Elaboración propia

HISTORIA DE USUARIOS	
<b>Número:</b> RF9	<b>Nombre del requisito:</b> Habilitar Apache 2.
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 15 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 30 horas
<b>Descripción:</b>	

El módulo permite al usuario habilitar el servicio del servidor web Apache 2. El usuario selecciona la opción de “Administrar Apache 2” del menú lateral izquierdo, luego acciona el botón “Habilitar”.

*Tabla 17 Historia de Usuario Configurar Apache 2 Fuente: Elaboración propia*

HISTORIA DE USUARIOS	
<b>Número:</b> RF11	<b>Nombre del requisito:</b> Configurar Apache 2.
<b>Programador:</b> Julio Jasan Badell Argudin	<b>Iteración Asignada:</b> 2
<b>Prioridad:</b> Media	<b>Tiempo Estimado:</b> 20 horas
<b>Riesgo en Desarrollo:</b> No aplica	<b>Tiempo Real:</b> 28 horas
<p><b>Descripción:</b></p> <p>El módulo permite al usuario configurar el servicio del servidor web Apache 2. El usuario selecciona la opción de “Configurar Apache” del menú lateral izquierdo, luego acciona el panel de selección para seleccionar las opciones de configuración predefinidas, en caso de que no se encuentre la configuración que desea, selecciona la opción “otra opción” y rellena los campos del formulario para introducir las configuraciones personalizadas”, al finalizar acciona el botón “Configurar”.</p>	