INGENIERIA EN CIENCIAS INFORMATICAS

FACULTAD 1

Software para la parametrización inicial de un escenario definido para actividades académicas sobre configuración de servicios telemáticos.

Tesis presentada para la culminación de estudios

Autor: Danay v. Horta López

Tutor(es): MSc. As Ruth Yurina Vega Cutiño

Ing. Héctor Lázaro Martínez Miranda

La Habana, 2021

Año 63 de la Revolución

Declaración de autoría:

Declaro por este medio que yo **Danay V. Horta López,** con carné de identidad **98021118197** soy el autor principal del trabajo titulado “Software para la parametrización inicial de un escenario definido para actividades académicas sobre configuración de servicios telemáticos y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los 8 días del mes de diciembre de 2021:

Tutores: MSc. As Ruth Yurina Vega Cutiño

Ing. Héctor Lázaro Martínez Miranda

Autor: Danay V. Horta López

Resumen

En la carrera Ingeniería en Ciencias Informáticas, la asignatura Seguridad y Redes II perteneciente a la disciplina de Sistemas Digitales es responsable de que los estudiantes adquieran competencias en administración de servicios telemáticos. La UCI, se encuentra impulsada por la necesidad de actualizar y optimizar los procesos en los laboratorios. Para ello la virtualización es la mejor alternativa para las clases. El software fomentara el mejor desempeño en cuanto a las actividades académicas relacionadas con los servicios telemáticos desarrollados por la Facultad 1. Eliminando el atraso del estudiante para configurar un nuevo servicio e incurrir en errores humanos. La presente investigación se enfoca en la implementación de un sistema que permita generar ficheros de configuración para el montaje de un escenario correspondiente a los servicios telemáticos de una red de computadoras, quedando definida AUP-UCI como metodología de desarrollo de la investigación, así como, las herramientas que serán utilizadas para la implementación. El desarrollo del sistema permitirá la parametrización de los servicios y ficheros de configuración utilizados en las actividades académicas, así como la optimización del tiempo en las mismas.

Palabras claves: servicios telemáticos, virtualización, ficheros de configuración, parametrización.

Summary

In the Computer Science Engineering program, the subject Security and Networks II, which belongs to the Digital Systems discipline, is responsible for students acquiring competencies in the administration of telematic services. The UCI is driven by the need to update and optimize the processes in the laboratories. For this purpose, virtualization is the best alternative for classrooms. The software will promote the best performance in terms of academic activities related to the telematic services developed by the Faculty 1. Eliminating the student's delay in configuring a new service and incurring human errors. The present research is focused on the implementation of a system that allows the generation of configuration files for the assembly of a scenario corresponding to the telematic services of a computer network, defining AUP-UCI as the methodology for the development of the research, as well as the tools that will be used for the implementation. The development of the system will allow the parameterization of the services and configuration files used in the academic activities, as well as the optimization of time in them.

Keywords: telematic services, virtualization, configuration files, parameterization.

Índice

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) juegan un rol fundamental en el desarrollo, evolución y eficiencia de cualquier empresa, institución y país de manera general. Una de las ventajas y facilidades de las TIC lo constituyen los servicios telemáticos (1). Estos funcionan de manera similar a cualquier servicio en el mundo y poseen dos partes bien identificadas, un servidor que ofrece un servicio y un consumidor, cliente o usuario de este servicio. (1) El servidor es un programa o software y los usuarios finales utilizan un software cliente para acceder a los beneficios del servidor. Estos pueden ser ofrecidos por una empresa, centro de estudio, institución, país, organización, entre otros y el usuario puede consumir múltiples servicios telemáticos de diferentes proveedores. (1)

Estos servicios han tenido avances bastante significativos en temas relacionados a la gestión y almacenamiento de la información con una arquitectura centralizada y descentralizada, basándose en herramientas que se pueden encontrar y gestionar fácilmente en la web, a través de infraestructura como, servidores virtuales y software para el almacenamiento de información operativa, y plataformas o paquetes de servicios multifuncionales en la nube, todos con costos reducidos hasta llegar a ser gratuitos. (2)

Sus herramientas también han evolucionado rápidamente gracias al avance de la ciencia y tecnología, influyendo significativamente en todos los sectores de las sociedades. Sin embargo, aún se requieren nuevas habilidades y destrezas para el uso de las modernas herramientas tecnológicas y medios de información en general, particularmente aquellas que promueven aprendizaje en los diferentes niveles de estudio. (3)

Las TIC, actualmente conocidas como nuevas tecnologías (NT) se han centrado con un mayor porcentaje en la educación, como herramienta educativa, tanto en el aula asistencial como a distancia, fomentando y promoviendo el aprendizaje significativo a través de herramientas tecnológicas. (3)

En la carrera Ingeniería en Ciencias Informáticas, la asignatura Seguridad y Redes II perteneciente a la disciplina de Sistemas Digitales es responsable de que los estudiantes adquieran competencias en administración de servicios telemáticos.

Esta asignatura se imparte mediante conferencias, clases prácticas y laboratorios. La virtualización es la alternativa para las clases en los laboratorios, de forma que se encuentren disponles los recursos tanto de hardware como software para cumplir los objetivos de aprendizaje. Esta nos permite homogenizar todos los recursos, por lo que se llega a estandarizar procedimiento y configuraciones, se consigue que el usuario vea los recursos que necesita como si fueran dedicados y sobre todo mejora la tolerancia a fallos.

Pese a ser la solución viable para afrontar la enseñanza de la administración de los servicios telemáticos, se plantea la dificultad de realizar la parametrización inicial antes de comenzar cada clase o evaluación. Esta tarea de configurar las estaciones de trabajo con las respectivas máquinas virtuales en ocasiones toma tiempo del destinado a la práctica, para lo cual el profesor debe preparar un conjunto de scripts que realicen la configuración de manera semiautomática y en correspondencia con un escenario dado.

Pero ¿Cómo se puede realizar esta tarea por lotes de forma semiautomática?

Este trabajo tiene como **objetivo** desarrollar un software que genere ficheros de configuración para el montaje de un escenario correspondiente a los servicios telemáticos de una red de computadoras. Como **objeto de estudio**, soluciones informáticas para la configuración de laboratorios virtuales y su **campo de acción** son las soluciones informáticas para la configuración de laboratorios de redes de computadoras y servicios telemáticos empleando máquinas virtuales.

Entre las **tareas de investigación** se encuentran:

1. Revisión de la bibliografía relacionada con la parametrización de la configuración de servicios telemáticos.
2. Selección de las tecnologías, herramientas y estándares que serán utilizados como guía del proceso de construcción de la solución propuesta.
3. Definición de la propuesta de solución y requisitos funcionales y no funcionales a implementar.
4. Identificación de los principios de diseño y funcionamiento del sistema para la posterior implementación del mismo**.**
5. Implementación de la solución propuesta.

Los **Objetivos específicos** que sustentan la investigación son:

1. Desarrollar el marco teórico que sustenta la investigación teniendo en cuenta el desarrollo web y la parametrización de un escenario de servicios telemáticos.
2. Describir el análisis y diseño de la solución que permita la parametrización de un escenario de servicios telemáticos.
3. Implementar la solución que permita la parametrización de un escenario de servicios telemáticos.
4. Validar la solución que permita la parametrización de un escenario de servicios telemáticos.

Como hipótesis de investigación se plantea: el desarrollo de un software que genere ficheros de configuración para el montaje de un escenario correspondiente a los servicios telemáticos de una red de computadoras, que utilice herramientas de código abierto, facilite el procesamiento de la información generada y permita automatizar los ficheros de la máquina virtual.

En la confección de este trabajo se utilizaron diferentes **métodos de investigación** como el hipotético-deductivo para la formulación de la hipótesis de trabajo y la concepción de los objetivos a alcanzar.

Método sistémico para la comprensión de las tecnologías actuales. Proporcionando la orientación general para el estudio de los procesos educativos. Ya que emana de su carácter orientador y organizacional como vía de la investigación científica, revela las relaciones, nexos y concatenaciones entre los diferentes procesos, hechos y resultados obtenidos en el camino de la ciencia.

También se empleó el método inductivo-deductivo para trabajar con los elementos extraídos de la bibliografía consultada elaborando conclusiones derivadas que ayuden a seleccionar, organizar y elaborar una propuesta sobre la base del conocimiento existente.

Como métodos empíricos se utilizó el análisis documental en la revisión de la literatura especializada y la medición para la validación de los resultados obtenidos.

El documento de la investigación consta de introducción, tres capítulos, conclusiones y referencia bibliográfica:

El Capítulo 1 brinda los aspectos básicos de los servicios telemáticos, en el cual se exponen una variedad de términos que son claves para el entendimiento de la investigación, soluciones similares al problema que

se plantea, la metodología de desarrollo llevada a cabo y una descripción de las tecnologías y herramientas que serán utilizadas.

En el Capítulo 2 se definen los elementos que describen la propuesta de solución, se enumeran los requisitos funcionales y no funcionales que se deben cumplir para la implementación del sistema, así como se analizan los elementos más relevantes que pueden ser empleados en el diseño o prototipo, tales como los modelados de los aspectos del diseño basados en el diagrama de clases de diseño.

El Capítulo 3 aborda la implementación de la propuesta del diseño de la interfaz como solución a los problemas planteados anteriormente, obteniendo un sistema que genera los ficheros de configuración para las actividades académicas con los servicios telemáticos. Se describe un grupo de pruebas realizadas al software, una vez que termina la implementación para asegurar así que este cumpla con las necesidades requeridas.

Capítulo 1. Fundamentación teórica

Introducción

En este capítulo se presenta el marco de referencia en el que se fundamenta la investigación, el cual está conformado por cuatro apartados principales a saber: términos claves, características generales, metodología del desarrollo y la descripción de las tecnologías y herramientas de desarrollo. Los términos claves agrupan los fundamentos teóricos que soportan el planteamiento del problema y define las disciplinas a la cual pertenece el campo de estudio escogido. Las características generales, por su parte, reúnen los principales aspectos que se evidencian en la literatura en relación con el tema de estudio y sirve como referencia para orientar la investigación, así como la descripción de las tecnologías y herramientas de desarrollo. La metodología del desarrollo tiene como finalidad garantizar la eficacia y la eficiencia en el proceso.

Términos clave

En esta sección se presentan algunas descripciones teóricas sobre temas que son relevantes en el desarrollo de un software que genere ficheros de configuración para los servicios telemáticos. Los términos claves proporcionan, principalmente, la terminología que es utilizada para la propuesta de investigación.

La **virtualización** es un tema muy amplio, por lo que se torna difícil encontrar un concepto que abarque toda su extensión, sin embargo, hay definiciones interesantes que intentan conceptualizarla. Según Bob Muglia, ex vicepresidente para servidores y herramientas de negocios de Microsoft Corporation el término puede definirse como: “Una estrategia para desplegar los recursos del ordenador en diferentes capas aisladas de hardware, software, datos, red, almacenamiento unas de las otras.” La virtualización es una tecnología que permite que varias instancias de máquinas virtuales (MV) se ejecuten en un mismo servidor físico y compartan el hardware del servidor bajo el control de un hipervisor. (4).Se puede decir que la virtualización es la técnica que posibilita la ejecución de una o más maquinas (máquinas virtuales) sobre una única maquina física.

Los hipervisores simplifican la gestión de recursos de hardware, garantizan su utilización eficiente por parte de las MV, aceleran su despliegue y ofrecen un mejor control de la infraestructura computacional disponible. (5) Es el programa de control maestro, con el más alto nivel de privilegios, y administra uno o más sistemas operativos, a los que se refiere como sistemas operativos huéspedes. Cada sistema operativo huésped

administra sus propias aplicaciones como lo hace normalmente en un ambiente no-virtual, con la diferencia de que está aislado del hardware por el monitor de máquina virtual (VMM). Cada SO huésped, con sus aplicaciones, es conocido como una máquina virtual. (6)

La virtualización de redes es el proceso de combinar recursos de software y hardware y funcionalidades de red en una única entidad administrativa basada en software, a la cual se le denomina red virtual. (6) Los componentes principales de una red externa son las redes locales virtuales (VLAN) y los conmutadores de red (*network switches*). Esta definición es probablemente la versión más utilizada del término, a través de las redes privadas virtuales (VPN) y de las VLANs. (6) La virtualización de redes, al implementar la combinación de varias redes físicas en una sola red virtual (VPN), simplifica y facilita el proceso de gestión. Con las VPN también se puede lograr que una sola red esté compuesta por un conjunto de redes dispersas geográficamente. A su vez las VLANs permiten a los administradores unir secciones de redes a conveniencia, dándoles la posibilidad de controlar todo el tráfico que circula por la red, lo cual, además, influye significativamente en la seguridad de la misma. (6) Para que esto se logre es necesario seguir protocolos de red que establecen un conjunto de reglas para controlar la comunicación entre los equipos que se encuentran conectados a internet.

La finalidad de una red es que los usuarios de los sistemas informáticos puedan hacer un mejor uso de los mismos mejorando de este modo el rendimiento global. Alguno de los **servicios de red** son:

FTP: protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. (7)

DNS: sistema para la asignación de nombres de dominio a direcciones IP. 53 (TCP), utilizado por DNS (Domain Name System) y 53 (UDP), utilizado por DNS (Domain Name System) (7)

DHCP: protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor (7).

HTTP: protocolo usado en cada transacción de la World Wide Web. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. (7)

NAT: mecanismo utilizado por enrutadores IP para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles. (7)

POP: protocolo que utilizan los clientes locales de correo para obtener los mensajes de correo electrónico almacenados en un servidor remoto. 110 (TCP), utilizado por POP3 (Post Office Protocol). 995 (TCP), utilizado por POP3 sobre SSL. (7)

SMTP: estándar internacional utilizado para transferencia de correo electrónico (email) entre computadoras. Hoy en día es utilizado exclusivamente para el envío de correos. 25 (TCP), utilizado por SMTP (Simple Mail Transfer Protocol). (7)

SSH: nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. (7)

TELNET: nombre de un protocolo de red que sirve para acceder mediante una red a otra máquina para manejarla remotamente. El puerto que se utiliza generalmente es el 23. (7)

TFTP: protocolo de transferencia muy simple semejante a una versión básica de FTP. TFTP a menudo se utiliza para transferir pequeños archivos entre ordenadores en una red, como cuando un terminal X Window o cualquier otro cliente ligero arranca desde un servidor de red. 69 (UDP), utilizado por TFTP (Trivial File Transfer Protocol). (7)

Shell: intérprete de comandos, es un programa que permite a los usuarios interactuar con el sistema, procesando las órdenes que se le indican. Además de comandos, los shells ofrecen otros elementos para mejorar su funcionalidad, tales como variables, funciones o estructuras de control. El conjunto de comandos internos y elementos disponibles, así como su sintaxis, dependerá del Shell concreto empleado. (8)

Fichero de configuración en sistemas Linux

Una de las principales diferencias que implementa Linux frente a otros sistemas, es el sistema de ficheros. Este es el componente del sistema operativo encargado de administrar y facilitar el uso de las memorias periféricas, entre las principales funciones se destaca la asignación de espacio a los archivos, la administración del espacio libre y del acceso a los datos guardados (9). Entre los sistemas de archivo se encuentran ext4, btrfs y XFS.

Repositorio

Los repositorios son sistemas de información que preservan y organizan materiales científicos y académicos como apoyo a la investigación y el aprendizaje, a la vez que garantizan el acceso a la información (10) Los repositorios institucionales consisten en estructuras web interoperables de servicios informáticos, dedicadas a difundir la perpetuidad de los recursos científicos y académicos (físicos o digitales) de las universidades, a partir de la enumeración de un conjunto de datos específicos (metadatos), para que esos recursos se puedan recopilar, catalogar, acceder, gestionar, difundir y preservar de forma libre y gratuita, de manera que están estrechamente ligados a los ideales y objetivos del acceso abierto. (11)

Según la Real Academia Española y la Asociación de Academias de la Lengua Española se define como parámetro el dato o factor que se toma como necesario para analizar o valorar una situación, y en la matemática es aquella variable en una familia de elementos, que sirve para identificar cada uno de ellos mediante su valor numérico. Por lo que la **parametrización** seria la acción de describir o estudiar algo mediante parámetros o como diría Añorga y colaboradores, es derivar el análisis del objeto en la investigación con elementos medibles u observables que permitan la valoración (12)

Paquetes de software

Ante la necesidad de un nuevo sistema se tienen distintas opciones, una de ellas son los paquetes de software, que según Laudon y Laudon son "un conjunto de programas precodificados y prescritos que están disponibles para ser adquiridos o rentados". Estos programas precodificados comprenden funciones que son consideradas comunes. Pues, aunque la necesidad de los usuarios es distinta, existen un conjunto de ellas que conllevan a unos procesos estandarizados que varían o varían muy poco. (13)

Características	Ventajas	Desventajas
Programas precodificados o prescritos	La mayor parte del trabajo ya fue desarrollado por la persona encargada de proveer el paquete. Los problemas técnicos ya han sido eliminados. El proveedor facilita herramientas y asistencia en la instalación.	Al estar prediseñados los paquetes pueden que no cumplan con todos los requerimientos.
Automatización de tareas	Las tareas se desarrollan de una mejor manera, el utilizar menor tiempo y ser más eficaz.	Se requiere de una alta inversión de recursos para poder obtener la automatización.

Tabla 1. Ventajas y desventajas de los paquetes de software de aplicaciones informáticas.

Características generales

Características de los laboratorios docentes

La facultad 1 cuenta con ocho laboratorios docentes donde se imparten clases de laboratorio, conferencias siempre que requieran algún tipo de software, también capacitaciones y cursos de posgrado. En cuanto a las evaluaciones se realizan exámenes como los de la asignatura Matemática, con el uso de asistentes matemáticos para la modelación de problemas. Están también las asignaturas de Redes que utilizan simuladores de Cisco y Huawei. Programación, con IDEs, cms y frameworks de desarrollo. Base de Datos, con el uso de gestores de base de datos como posgreSQL, entre otras asignaturas que utilizan diversos softwares instalados en los laboratorios docente para el desarrollo de las habilidades necesarias en los estudiantes de pregrado y posgrado.

De estos laboratorios, cinco tienen procesadores i3 y 4gb de RAM, los restantes sen de menores prestaciones con diversos tipos de hardware. Todas las PC utilizan como sistema operativo NOVA 5 ligero

Características de los ficheros de configuración básica de sistemas Linux. (14) (15) (16)

Los sistemas de ficheros pueden ser clasificados mediante varios criterios (rendimiento con manejo de ficheros, soporte de errores, soporte de características añadidas -por ejemplo, soporte de ACL...-), pero uno de los más habituales es el que se presenta a continuación, según su naturaleza: de disco, de red, virtuales o con propósitos especiales.

Sistema de ficheros de disco

Se trata de los sistemas de ficheros que se encuentran en los dispositivos locales de los ordenadores.

- Ext2 (second extended filesystem)/ext3 (third extended filesystem): son los sistemas nativos de Linux. Garantizan la compatibilidad de versiones anteriores, de modo que futuras actualizaciones no requieran “rehacer” el sistema de ficheros. La diferencia más importante entre ext2 y ext3 radica en el soporte de este último de *journaling*.
- ext4 se introdujo en 2008 como sucesor de ext3. Este sistema de archivos es actualmente el estándar para muchos sistemas Linux, como Ubuntu. Su novedad más importante es la función *extents*, que optimiza la gestión de archivos grandes y evita la fragmentación de manera más eficaz que sus predecesores. Con ext4, las particiones se pueden ampliar y reducir según sea necesario, e incluso durante el procesamiento.

- ReiserFS: se trata de un sistema de ficheros con *jounaling* desde su nacimiento, propuesto por la empresa *Namesys*; es la opción por defecto en algunas distribuciones. Normalmente, para ficheros de tamaño pequeño tiene mejor rendimiento que ext2 y ext3. Actualmente la versión que está soportada por la empresa creadora es la Reiser4.
- XFS: se trata de un sistema de ficheros de 64-bits con *journaling* y un excelente rendimiento (sobre todo con ficheros de gran tamaño)
- JFS (*journaling filesystem*): desarrollado por IBM para servidores, se trata de un sistema de ficheros de 64 bits y *journaling*. Fue concebido para servidores que requiriesen alto rendimiento y de ficheros de altas prestaciones. Como característica distintiva, la asignación de inodes no es estática, sino dinámica.
- ISO9660: es el sistema de ficheros estándar para volúmenes de sólo lectura como los CD-ROM.

Sistema de ficheros en red

Este tipo de sistemas de ficheros posibilitan que ordenadores clientes, a través de una red de área local, se conecten a otro servidor y accedan a sus ficheros como si tratase de recursos locales.

- NFS (*network filesystem*): desarrollado inicialmente por Sun Microsystems, suele ser la opción por defecto para sistema de ficheros en red sobre GNU/Linux. El protocolo es independiente de la máquina, del sistema operativo y del protocolo de transporte, ya que implementa onC RPC. Es interesante señalar que todas las opciones son síncronas (respecto al trabajo sobre el fichero).
- CIFS (*common internet filesystem*): también conocido como SMB o Samba, la implementación más utilizada es la desarrollada por Microsoft, y es utilizado en sistemas Windows. Permite compartir ficheros e impresoras por la red, y Linux puede implementar tanto la versión de servidor como de cliente. Es decir, permite la convivencia simultánea de sistemas Windows y GNU/Linux en la misma red de área local.

Sistema de ficheros virtuales

- VFS (*virtual filesystem*): se trata de un nivel lógico superior a los sistemas de ficheros presentados hasta ahora. Básicamente, se trata de un interfaz entre el núcleo (kernel) y el sistema de ficheros real

- SysFS: es un sistema de ficheros virtual que proporciona el kernel 2.6 de Linux. Básicamente, Sysfs proporciona información de los dispositivos del sistema (hardware) y sus controladores hacia el espacio del usuario, permitiendo además configurar alguno de sus parámetros.

Sistema de ficheros especiales

- SWAP: del inglés intercambiar, es el espacio de disco duro (puede ser un fichero o una partición) que se usa para guardar el estado de procesos que no se utilizan (o no caben) en la memoria física.
- GmailFS para Linux: basado en FUSE (el mecanismo de sistema de ficheros en el espacio de usuario) y desarrollado bajo Python, y permite proveer a los usuarios del conocido sistema de correo Gmail un sistema de ficheros accesible.

Configuración de la virtualización

Para la configuración de la virtualización se debe tener presente los diferentes softwares de virtualización, sus ficheros de configuración y los parámetros configurables.

Software de virtualización	Ventajas	Desventajas	Fichero de configuración	Características
VirtualBox	Permite la visión de la máquina virtual en modo pantalla completa, integra el mouse entre la maquina física y la virtual. Permite realizar las modificaciones necesarias o en su configuración.	No posee las funciones de nube empresarial que tienen otras	.vbox Contiene configuraciones como el nombre de la máquina virtual, el tipo de SO, la memoria del sistema.	Código abierto Soporta Linux, Windows, OS/2, Solaris

	Disponible para virtualizar la mayor parte de los sistemas operativos			
VMWare	conexiones de red, el montaje de imágenes y la interacción son fácilmente configurable con el host.	Los costos iniciales son considerables. No se puede virtualizar todo el hardware o software.	.vmx(archivo de configuración de la máquina virtual) .vmxf(archivo de configuración de la máquina virtual adicionales) .nvram(configuración del BIOS o EFI de la máquina virtual)	Código abierto Soporta Windows y Linux
Xen	buen rendimiento (al ser como un hipervisor) no necesita ningún tipo de drivers para poder utilizar los USB 3.0.	No existe la gestión de errores	Xen/xend-config.sxp (configuración de la red) .hvm	Código abierto Soporta Linux, FreeBSD y OpenBSD

Tabla 2. Ficheros de configuración del software de virtualización (17)

Se puede decir que la «máquina virtual» es el contenedor del sistema operativo; como si fuera una partición de disco. (18) Para que esta tenga un buen funcionamiento se deben configurar los parámetros Algunos de los cuales son opcionales, como el acceso a dispositivos ópticos, la red, carpetas compartidas con el anfitrión y USB. Pero otros son definitivos para que la máquina virtual funcione bien.

Parámetros configurables básicos:

- Nombre y Sistema operativo
- Tamaño de memoria

- Disco duro
- Tipo de archivo de disco duro
- Ubicación del archivo y tamaño
- Almacenamiento en unidad de disco duro física

Parámetros para mejorar el desarrollo del SO invitado:

- En la opción Sistema>Placa base se encuentran memoria base, orden de arranque chipset, dispositivo apuntador y características extendidas. En Procesador, están, límite de ejecución, características extendidas y procesador. Le sigue aceleración en la cual sus parámetros se dejan en predeterminado.
- La opción Pantalla tiene como parámetros a configurar memoria de video, numero de monitores, factor de escalado, controlador gráfico y aceleración.
- En opción Audio, habilitar audio (salida y entrada de audio).
- Para poder navegar por internet se debe configurar la opción Red donde existen distintas posibles conexiones. Adaptador 1, permite habilitar el adaptador de red para la navegación a través del ordenador anfitrión.
- Opción Puertos de serie, para conectar la maquina virtual a las interfaces serie, del ordenador anfitrión.
- Interfaz de usuario que tiene como parámetros los botones a mostrar en la interfaz y la mini barra de herramientas.

Parámetros para la instalación del SO invitado:

- Para el arranque de la maquina se debe configurar la opción de Almacenamiento donde se escogerá los parámetros según la necesidad. Dependiendo de la elección en dispositivos de almacenamientos, serán los parámetros en atributo.

Características básicas de los servicios que se quieren pre configurar

Además de la configuración de la máquina virtual también se necesitan configurar los servicios de red. En la siguiente tabla se muestran los servicios y sus descripciones.

Servicios	Ficheros de configuración	Elementos invariantes	Parámetros fundamentales
DNS	named.conf: Archivo principal de configuración named.conf.options:	Método DNS Método DNS IPv6	Servidor DNS primario Servidor DNS secundario

	Opciones genéricas named.conf.local: Especificación particular de este servidor DNS db.127:Especificación dirección de retorno db.root: DNSs de nivel superior Otros archivos: db.0, db.255, db.empty, db.local, rndc.conf, rndc.key, zones.rfc1918	Prioridad resolución DNS Timeout DNS (segundos) Nombre de host	Nombre de dominio Servidor DNS IPv6 primario Servidor DNS IPv6 secundario Nombre de dominio IPv6
DHCP	etc/dhcp3/dhcpd.conf		Dirección IP Máscara de subred Puerta de enlace Servidores DNS Configuración proxy
HTTP	httpd.conf	servidor HTTP reorientación HTTP a casilla de verificación HTTPS	máximo de sesiones tiempo de espera de la sesión puerto HTTP
HTTPS	ssl.conf	servidor HTTPS	puerto HTTPS

Tabla 3. Descripción de los servicios HTTPS, HTTP, DNS y DHTP.

Características de los ficheros Shell-scripts

El sistema a desarrollar debe generar scripts que serán utilizado en las actividades académicas mediante la virtualización, para ello se configurarán ficheros Shell-scripts. El shell de Unix o también shell, es el termino usado en informática para referirse a un intérprete de comandos. En su forma más básica, un shell-script puede ser un simple fichero de texto que contenga uno o varios comandos. Para ayudar a la identificación del contenido a partir del nombre del archivo, es habitual que los shell scripts tengan la extensión «.sh».

Unix tiene 2 categorías principales de shells, Shell tipo Bourne y C shell. El primero se clasifica como: Korn shell (ksh), Bourne shell (sh), POSIX shell (sh) y Bourne Again shell (bash). El segundo como: C Shell (csh) y TENEX (TOPS) C shell (tcsh) (18). Bash es una herramienta popular y poderosa para todo usuario de Linux o administrador de sistemas. (19) Un fichero de bash simplemente contiene un seguido de instrucciones que el intérprete del sistema operativo leerá, comprenderá y ejecutará. (20)

Dash, en el sistema operativo GNU / Linux fue originalmente un enlace simbólico a bash, pero en vista de la complejidad de este, se llevó de NetBSD a Linux y lo renombró dash (*Debian Almquist Shell*). (21) El cual es un Shell que tiene poca usabilidad para las necesidades modernas cuando se utiliza como un Shell de inicio de sesión interactivo, sin embargo, es más rápido en la ejecución de scripts compatibles con POSIX que Bash. Cuando el usuario transmite cualquier vídeo digital en la web, el sistema genera un archivo DASH. Este proporciona un flujo adaptativo dinámico a través de HTTP. El reproductor de vídeo en línea puede ajustarse a la velocidad de bits del vídeo digital gracias a la capacidad del formato de archivo DASH de almacenar contenidos a múltiples velocidades de bits. (22)

El Korn Shell (ksh) es un lenguaje de control y programación de tareas interactivas para entornos UNIX que presenta un mejor rendimiento que los lenguajes tradicionales. Soporta mayores facilidades de operaciones de Entrada/Salida, tipos de datos y atributos, vectores unidimensionales, aritmética entera, facilidades para el control de tareas y funciones. Los archivos de extensión de ksh (. kshrc) pueden ser utilizados por programas distribuidos para la plataforma Linux. Este junto a 139 otros formatos de archivos pertenecen a la categoría de archivos ejecutables.

Softwares similares

Debido a la importancia de los softwares parametrizables, hace años se han venido formulando trabajos de investigación. Tal como es el caso de aquel denominado Configuración y parametrización de la plataforma de repositorio de datos “DSPACE” para la biblioteca NÉSTOR GRAJALES LÓPEZ en la unidad central del Valle del Cauca de Angelica Osorio Espinoza en el 2019. Se enfocó en aplicar las TIC´s en la configuración y Parametrización del software Dspace, para crear e implementar un repositorio institucional que se convirtiera en una herramienta innovadora, fomentando la creación y búsqueda de documentos académicos. Por consiguiente, la realización de este repositorio institucional, le trae unas ventajas muy significativas a la universidad; ya que se logra aumentar la accesibilidad y visibilidad de documentos. (23)

En un segundo lugar, es importante mencionar el denominado Diseño de la parametrización de un sistema de información para el área de cobranza del banco WWB realizado por Juan Carrillo Sánchez en Santiago de Cali en el 2017. En el desarrollo de este, el investigador hace referencia a la implementación de los sistemas de Información en las áreas de cobranzas, una valiosa oportunidad de investigación, mejora de procesos y fortalecimiento del papel del contador dentro de diversos roles para los cuales se está capacitado. (24)

La investigación realizó el diseño de la parametrización de un sistema de información para el área de cobranzas del Banco WWB, que facilito la realización de las labores de captura, procesamiento y transformación de la información del área de cobranzas de las entidades financieras. (24)

Sumado a esto, es preciso mencionar la investigación titulada Soporte de la parametrización contable del software SERVINTE CLINICAL SUITE en el HOSPITAL INFANTIL DE SAN JOSÉ de la estudiante Yisel Gómez Ardila en el 2019. Ella determinó que era necesario implementar un proceso de ajustes al software, que comprendería la parametrización sistemática de las cuentas contables y centros de costos, utilizados como base en los distintos módulos del sistema de gestión de información. Observa la necesidad de realizar ajustes desde el punto de vista contable a la parametrización de los módulos administrativos y financieros, para que la información fluya adecuadamente y lograr información financiera más completa. (25)

El Software SERVINTE CLINICAL SUITE ENTERPRISE: es para gestión y administración de entidades de salud, de la empresa Carvajal Tecnología y Servicios, diseñada para el apoyo de procesos asistenciales y administrativos de las Instituciones prestadoras de Servicios de Salud, EPS y redes caracteriza por ser una solución altamente parametrizable, integrada con dispositivos, equipos biomédicos, sistemas de laboratorio y radiología. (25)

Se puede observar que muchos investigadores han trabajado la parametrización de softwares, pero no se pudo encontrar ninguna investigación avalada sobre la parametrización de ficheros de configuración para maquinas virtuales, por lo que este trabajo investigativo va a ser novedoso en el campo de la telemática.

Metodología del desarrollo

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo (26).Usar una metodología

aporta al desarrollo una garantía de calidad, debido a que es un proceso estructurado, organizando la forma en la que se va a realizar el proyecto. Existen quince tipos de metodologías de desarrollo de software las cuales se pueden dividir en agiles y tradicionales. La primera, son metodologías adaptativas, que permite llevar a cabo, proyectos de desarrollo de software, adaptándolo a los cambios como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente, permitiendo una mejor adaptación al entorno, maximizando la inversión y reduciendo costos (27).

Metodología AUP

El Proceso Unificado Ágil de *Scott Ambler* o *Agile Unified Process* (AUP) en inglés, es una versión simplificada del proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas agiles y conceptos que aún se mantienen válidos (28). El AUP aplica técnicas agiles incluyendo: Desarrollo dirigido por pruebas, modelado ágil, gestión de cambios ágil, refactorización de base de datos para mejorar la productividad.

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto. La UCI decide hacer una variación de la metodología AUP de forma tal que se adapte al ciclo de vida definido para la actividad productiva. (28)

AUP establece cuatro fases: (Inicio, Elaboración, Construcción, Transición). La UCI decide para el ciclo de vida de los proyectos mantener la fase de Inicio, en la cual se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. Se unifican las restantes 3 fases de AUP en una sola, a la que se denomina fase de Ejecución que sería aquella fase en la que 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, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software, y se agrega una

fase de cierre. En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (28)

Tecnologías y herramientas de desarrollo

Servidor web

En el desarrollo y puesta en marcha de este proyecto, resulta indispensable la utilización de un servidor web, se entiende por este como el programa que implementa el protocolo HTTP. Sobre la palabra servidor es necesario distinguir claramente entre la parte que identifica al programa y la parte que identifica al ordenador en sí. El servidor web tiene la función de permanecer a la espera de peticiones HTTP de los clientes, que suelen llevarlas a cabo a través de un navegador web. (28) Los clientes realizan peticiones HTTP al servidor, y esté les responde con el envío de ficheros solicitados, texto plano (html, php) o binarios (gif, jpeg). (29)

Para el funcionamiento de esta aplicación se va a emplear el servidor Nginx ya que es necesario que el servicio funcione sobre plataforma GNU/Linux, dado que es el Sistema Operativo que tiene instalado el ordenador en el que se va a implantar el sistema.

Nginx es de código abierto, en su versión inicial funcionaba en servidores web http. Sin embargo, hoy en día también sirve como proxy inverso, balanceador de carga http y proxy de correo electrónico para IMAP, POP3 y SMTP. (30) Fue creado para solucionar el problema C10K (Problema de optimización de network sockets, con el fin de manejar un gran número de clientes al mismo tiempo, exactamente diez mil conexiones al mismo tiempo), siendo esta una gran e importante diferencia de los servidores web tradicionales.

Características	Ventajas	Desventajas
Balanceo de carga, distribuyendo el tráfico entre varios servidores, redirigiendo cada vez la petición hacia aquella máquina que tenga una menor carga Proxy inverso Alta tolerancia a fallos	Potente y el consumo de recursos es bajo Instalación y configuración es simple y sencilla Brinda flexibilidad en cuanto a las capacidades de configuración	En comparación con Apache el soporte es limitado No maneja los archivos .htaccess, por lo que para agregar reglas hay recargar la configuración

Soporte para TSL, SSL, FastCGI, SCGI o WSGI, entre otros Compatible con el nuevo estándar de direcciones IPv6 Reescritura de urls, para crear urls amigables que nos ayuden en el proceso del posicionamiento web, Geolocalización basada en direcciones IP Proxy SMTP, POP3, IMAP	rápido para el procesamiento de contenido estáticos Capacidad de manejar más de 10 000 conexiones simultáneas Arquitectura orientada a eventos.	

Tabla 4. Características principales, ventajas y desventajas Nginx (31) (32)

Lenguajes de programación

Según Marvin López un lenguaje de programación, en palabras simples, es el conjunto de instrucciones a través del cual los humanos interactúan con las computadoras. Un lenguaje de programación nos permite comunicarnos con las computadoras a través de algoritmos e instrucciones escritas en una sintaxis que la computadora entiende e interpreta en lenguaje de máquina. (33)

En la publicación realizada por Martínez Gómez se menciona a los 10 lenguajes de programación más demandados en el año 2018 (34), de este listado se ha seleccionado el lenguaje de programación Java, PHP y Python, para realizar un análisis comparativo debido a que son muy utilizados en el desarrollo de aplicaciones web

Lenguaje de programación	Características	Ventajas	Desventajas
Java	Máquina virtual propia. Arquitectura neutral multihilo. (35)	Reutilización de código. Fácil conexión con diferentes motores de bases de datos. (35)	Requiere un intérprete. Los compiladores e intérpretes generan más código máquina. (35)
PHP	Lenguaje de programación de dominio específico (36) Lenguaje de código abierto (37)	Capacidad de conexión con la mayoría de los motores de base de datos (38) Permite independencia de plataforma (36)	Si no se establece la configuración correcta se dejan abiertas muchas brechas de seguridad (34) Requiere instalar un servidor web (36)
Python	Es multiplataforma. Permite la programación imperativa, orientada a objetos y funcional. (39)	Su sintaxis es fácil de entender. Su código es más organizado. Tiene una gran comunidad dispuesta a ayudar. (39)	Los programas interpretados son más lentos que los compilados. Inconvenientes al ejecutar procesos multihilos. (40)

Tabla 5. Lenguajes de programación, sus características.

Características	
Legible y elegantes	Imposible escribir código ofuscado.
Scripting	No tiene que declarar constantes y variables antes de utilizarlas. No requiere paso de compilación. Alta velocidad de desarrollo y buen rendimiento.
Sripting	No requiere paso de compilación. Alta velocidad de desarrollo y buen rendimiento.
Código interoperable	Se puede utilizar en múltiples plataformas (más aún que Java). Ejecutar Python dentro de una JVM (Jython)
Simple y poderoso	Soporta objetos y estructuras de datos de alto nivel: strings, listas, diccionarios, etc. Niveles de organización código: funciones, clases, módulos y paquetes. Incluye librerías que contiene un sinfín de clases de utilidad.
Propósito general	Puedes hacer en Python todo lo que pueden hacer con C# o Java.

Tabla 6. Características de Python (41)

Para el funcionamiento de esta aplicación se ha decidido por el empleo de **Python**, un aspecto importante a considerar es que permite la facilidad de extensión esto quiere decir que se puede escribir nuevos módulos

de manera fácil en bajo lenguaje como C o C++ y se puede incluir para aplicaciones que necesiten una interfaz programable. (41) Python cuenta con miles de librerías, módulos, códigos y programas de uso libre lo cual es una ventaja para la implementación de la propuesta de estudio. El servidor en el que se va a alojar la aplicación cuenta con Sistema Operativo Debian GNU/Linux y servidor web Nginx, por lo que el empleo de este lenguaje es el más recomendable. Además de este lenguaje se hará uso de Bash (Bourne-againShelll) para realizar la mayoría de las tareas necesarias. Esta es una popular interfaz de usuario de línea de comandos, específicamente un Shell de Unix; así como un lenguaje de scripting. Bash fue originalmente escrito por Brian Fox para el sistema operativo GNU, y pretendía ser el reemplazo de software libre del shell Bourne. Lanzado por primera vez en 1989, se ha utilizado ampliamente como el intérprete de inicio de sesión (login) predeterminado para la mayoría de las distribuciones de GNU/Linux.

Es compatible con el agrupamiento de nombres de archivo (coincidencia de comodines), tuberías, sustitución de comandos, variables y estructuras de control para pruebas de condición e iteración. Las palabras reservadas, la sintaxis, las variables de ámbito dinámico y otras características básicas del lenguaje se copian de sh. Otras características, por ejemplo, el historial, se copian de csh y ksh.

Entorno de desarrollo integrado

Suárez Falcón define a un entorno de desarrollo integrado (IDE) como una aplicación de software que proporciona al programador servicios integrales para facilitar el desarrollo de software y maximizar su productividad; un IDE está compuesto por un editor de código, herramientas de construcción automática, un depurador y en la mayoría de los casos consta con un auto completado inteligente de código mientras que en algunos casos cuentan con un compilador, un intérprete o ambos. (42)

En la actualidad hay una infinidad de herramientas que cubren todo tipo de necesidades, lo que resulta difícil escoger la adecuada. **Pycharm,** es un entorno de desarrollo integrado, multiplataforma, con asistencia de codificación y editor de código inteligente, refactorizaciones rápidas y seguras, herramienta de desarrollo integrada donde incluye un depurador integrado con un VCS, depuración, prueba y creación de perfiles. Se puede crear y ejecutar pruebas con asistente de código y un testador de pruebas basado en GUI. VCS, implementación y desarrollo remoto gracias a una interfaz de usuario unificada se ahorra tiempo para trabajar. Ofrece soporte de framework específicos para el desarrollo web moderno como Django y tiene soporte integrado para librerías científicas. (43)

Es uno de los IDE de Python más completos, perfecto para todos los niveles de experiencia, de ahí que será el utilizado en el desarrollo del proyecto de la investigación. Incluye funciones inteligentes que ayudan a ser más productivos. También tiene funciones de búsqueda, es personalizable y tiene cerca de un millar de plug-ins para mejorarlo.

Marco de trabajo (framework)

Python considerado uno de los lenguajes favorito de los programadores cuenta con varios framework muy poderosos. Para el desarrollo del trabajo se decidió utilizar **Django** por las características presentadas en la siguiente tabla.

Características	Ventajas	Desventajas

Bajo acoplamiento	Documentación	Velocidad
Patrón MVC (modelo-vista-controlador)	Python	Optimización de base de datos
Diseño de URL elegante	Escalable	
Sistema de plantillas	Desarrollo rápido	
Internacionalización	Apps reusables	
Servidor de pruebas		
Autenticación de usuarios		
Administración		

Tabla 7. Características, ventajas y desventajas de Django

Herramienta para el modelado y herramienta CASE

El Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) fue adoptado en noviembre de 1997 por OMG (*Object Management Group)* como una de sus especificaciones y desde entonces se ha convertido en un estándar de facto para visualizar, especificar y documentar los modelos que se crean durante la aplicación de un proceso de software. UML ha ejercido un gran impacto en la comunidad del software, tanto a nivel de desarrollo como de investigación (44)

Para el modelado se utilizará Visual Paradigm 10.1 por ser una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como ingeniería inversa de bases de datos (44)

Conclusiones del capitulo

A partir de la revisión del marco teórico**,** se comprobó que no existe ninguna aplicación que pueda ser utilizada en el entorno UCI que resuelva la problemática planteada en todos sus aspectos. Una vez comprobada la necesidad del desarrollo de una interfaz parametrizable, se valoraron las metodologías, tecnologías y herramientas a utilizar. Luego de ponderar las facilidades de cada una, se seleccionaron las siguientes: como lenguaje de programación **Python** y **Bash** donde se integrará el framework **Django**. Como IDE de programación Pycharm, para el modelado **Visual Paradigm 10.1 y** como metodología, **AUP vUCI.**

Capítulo 2. Características y descripción de la solución propuesta

Introducción

En el presente capítulo se describe la solución propuesta y el proceso realizado para el desarrollo del software que genere ficheros de configuración correspondiente a los servicios telemáticos. Se especifica la arquitectura del sistema y se detallan los requisitos funcionales y no funcionales, plasmados en el artefacto, se presenta un ejemplo de las Historias de usuarios descritas por el cliente, propio de la metodología AUP vUCI. Se muestra el Modelo de diseño basado en la arquitectura del sistema, así como el Modelo de despliegue, donde se explican los componentes con que contará la aplicación.

Descripción del sistema

La aplicación propuesta aspira a ser una herramienta útil para el desarrollo de las actividades tanto prácticas como evaluativas de la universidad y cumplir los objetivos de aprendizaje de esta. Para definir los principales requisitos del sistema se realizó un análisis de las necesidades planteadas. El software permitirá resolver problemas tales como:

- configurar las estaciones de trabajo con las respectivas máquinas virtuales
- preparar un conjunto de scripts que realicen la configuración de manera semiautomática
- evitar el atraso del estudiante para configurar un nuevo servicio
- evitar incurrir en errores humanos

Funcionará como un servicio para publicar los ficheros de configuración que se generen. Entre las funciones se pueden encontrar: Importar fichero de configuración del software de virtualización. Importar fichero de configuración del repositorio. Definir estructura del laboratorio: Cantidad de servidores, subred y cantidad de clientes. Por cada máquina cliente o servidor, definir: IP, Nombre de NetBIOS. Importar y generar ficheros de configuración de los servicios a partir de definiciones del usuario y publicarlos.

Arquitectura del sistema

Para el desarrollo del sistema se emplea una arquitectura en tres capas, cuyo principal objetivo es la separación de la lógica de negocios de la lógica de diseño. El autor Infante Montero menciona a la arquitectura MTV como la arquitectura utilizada por el framework de desarrollo Django, que consiste en una

modificación en la arquitectura MVC (46). En el artículo Phython - Django Framework de desarrollo web para perfeccionistas basado en el modelo MTV escrito por J. L. Condri Ayala se explica la analogía existente entre las arquitecturas MTV y MVC: el modelo en Django sigue siendo modelo, la vista en Django se llama Template o Plantilla, el controlador en Django se llama Vista. (47),

Los elementos del patrón son los siguientes:

Modelo: En la publicación realizada por García Fuentes menciona al modelo como la fuente única y definitiva de los datos que posee la aplicación desarrollada que permite ejecutar operaciones con ellos (48). En el artículo denominado Python - Django Framework de desarrollo web para perfeccionistas basado en el modelo MTV escrito por J. L. Condri Ayala, establece que el modelo se encuentra en forma de clases de Python y por medio del modelo se puede indicar y controlar el comportamiento de los datos almacenados. (47)

Vista: De acuerdo con la publicación realizada por (48) se menciona a la vista como el elemento de la aplicación que contiene la lógica de negocio; necesaria para devolver una respuesta hacia el cliente que la solicita, también procesa las peticiones o solicitudes que accederán al modelo para poder entregar u obtener los datos. En el artículo denominado Python - Django Framework de desarrollo web para perfeccionistas basado en el modelo MTV publicado por J. L. Condri Ayala, establece que la vista en Django se presenta en forma de clases en forma de funciones en Python, en el que se determinan los datos que serán visualizados entre otras cosas más, permite escribir código Python en lugar de instrucciones SQL para realizar las consultas que se requieran en la vista; determinando que la vista no se responsabiliza del estilo para presentar los datos sino se encarga de los datos. (47)

Plantilla: De acuerdo con la publicación realizada por García Fuentes, la plantilla decide la forma en la que se presentarán los datos devueltos por la vista en el navegador web; utilizando estilos CSS o brindando dinamismo a través de JavaScript (48). En el artículo Python - Django Framework de desarrollo web para perfeccionistas basado en el modelo MTV publicado por (47), indica que las etiquetas utilizadas por Django, permiten que sea flexible para los diseñadores del front-end, estas estructuras son limitadas para evitar un desorden poniendo cualquier tipo de código escrito en Python; con el objetivo de que lógica del sistema siga permaneciendo en la vista.

En la aplicación a desarrollarse se utiliza la arquitectura antes mencionada. El proceso iniciará cuando el navegador envié una solicitud a la aplicación, la capa de rutas interpretará la solicitud y se mostrará la vista apropiada. En la capa de interfaz de usuario se implementará modelo MTV (Model Template View – Modelo Plantilla Vista) ya incorporado en el framework Django; la vista será la encargada de interactuar con el modelo para obtener los datos además será la encargada de llamar a la plantilla para que finalmente la plantilla renderice la respuesta a la solicitud del navegador.

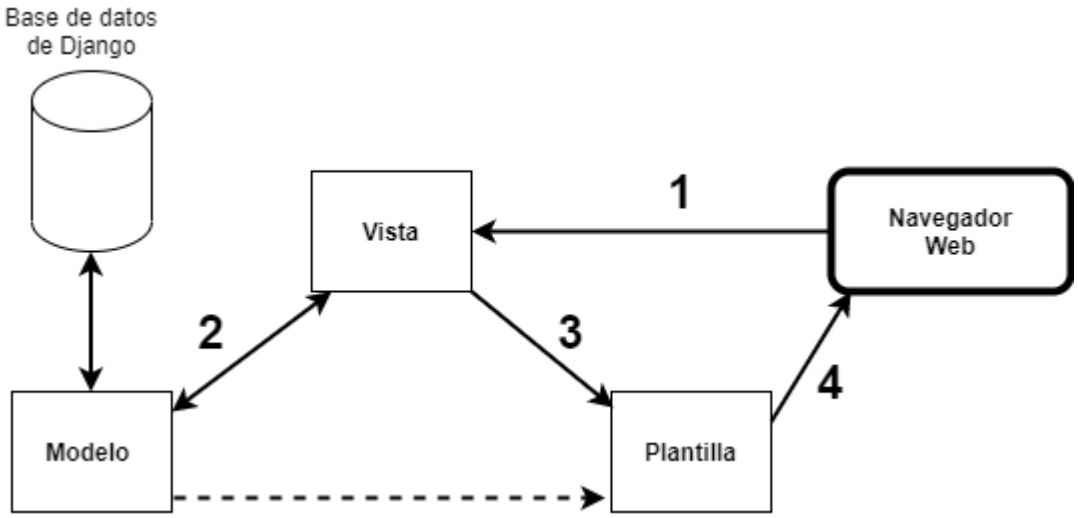


Figura 1. Arquitectura MTV

Patrones de diseño

Un patrón de diseño es una solución estándar para un problema común de programación. En un *software* contribuye a la reutilización de su diseño, ellos nombran, abstraen e identifican los aspectos claves de su estructura, que lo hace útil para la creación de diseños orientados a objetos. (49)

Para el desarrollo del sistema se tuvo en cuenta los patrones de asignación de responsabilidades, conocidos como patrones GRASP acrónimo de *"General Responsibility Assignment Software Patterns"*, los cuales tienen como objetivo fundamental orientar al diseñador en cómo asignar las responsabilidades a cada clase en diferentes circunstancias. (49)

Además, se tuvo en cuenta los patrones Gof **(Pandilla de los Cuatro)** acrónimo de *"Gang of Four"*, los cuales constituyen un catálogo de 23 patrones de diseño publicados por Erich Gamma, Richard Helm, Ralph

Johnson y John Vlissides en el libro *Design Patterns: Elements of Reusable Object-Oriented Software*, los cuales se clasifican en dependencia del propósito para los que hayan sido utilizados.

Patrones GRASP

Controlador: patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. (50)

Creador: es el patrón que guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El objetivo fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al comportarse como creador se da soporte al bajo acoplamiento. (51)

Experto: Posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. (50)

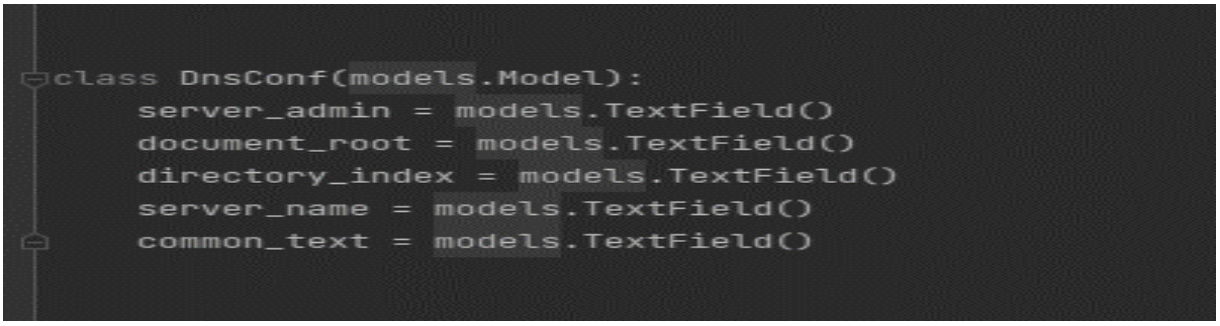


Figura 2 Fragmento de código. (Clase DnsConf)

Alta cohesión: tiene un numero relativamente pequeño de métodos, con funcionalidades altamente relacionadas, y no realiza mucho trabajo, colaborando con otros objetos para compartir el esfuerzo. Las clases con alta cohesión son relativamente fáciles de mantener, entender y reutilizar. (51)

Bajo Acoplamiento: 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 mucha s dependencias lo que facilitara la reutilización de la misma y se reducirá el impacto de los cambios. (51) El uso del patrón se puede ver evidenciado a través de la arquitectura en capas, evitando que si existe algún cambio en clases de las capas inferiores no afectaría múltiples clases en las capas superiores solo a las relacionadas con ellas.

Los patrones *Gand of Four* (GOF) empleados en el desarrollo del sistema, son los dos que se describen a continuación.

***Command*:** Patrón de comportamiento muy sencillo y con muchas aplicaciones útiles, ya que ordena las acciones y tareas de una aplicación, definiendo una clase para cada tarea u operación que implemente una interfaz común.

Abstract Factory: Pertenece a la familia de los patrones creacional, el cual se refiere al proceso de creación de objetos. Este patrón permite que el sistema sea independiente a cómo se creen sus objetos y refleja cómo crear familias de objetos relacionados con productos sin instanciar clases directamente. También, se ve reflejado en las clases controladoras al crear los formularios.

***Visitor*:** Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

Diseño

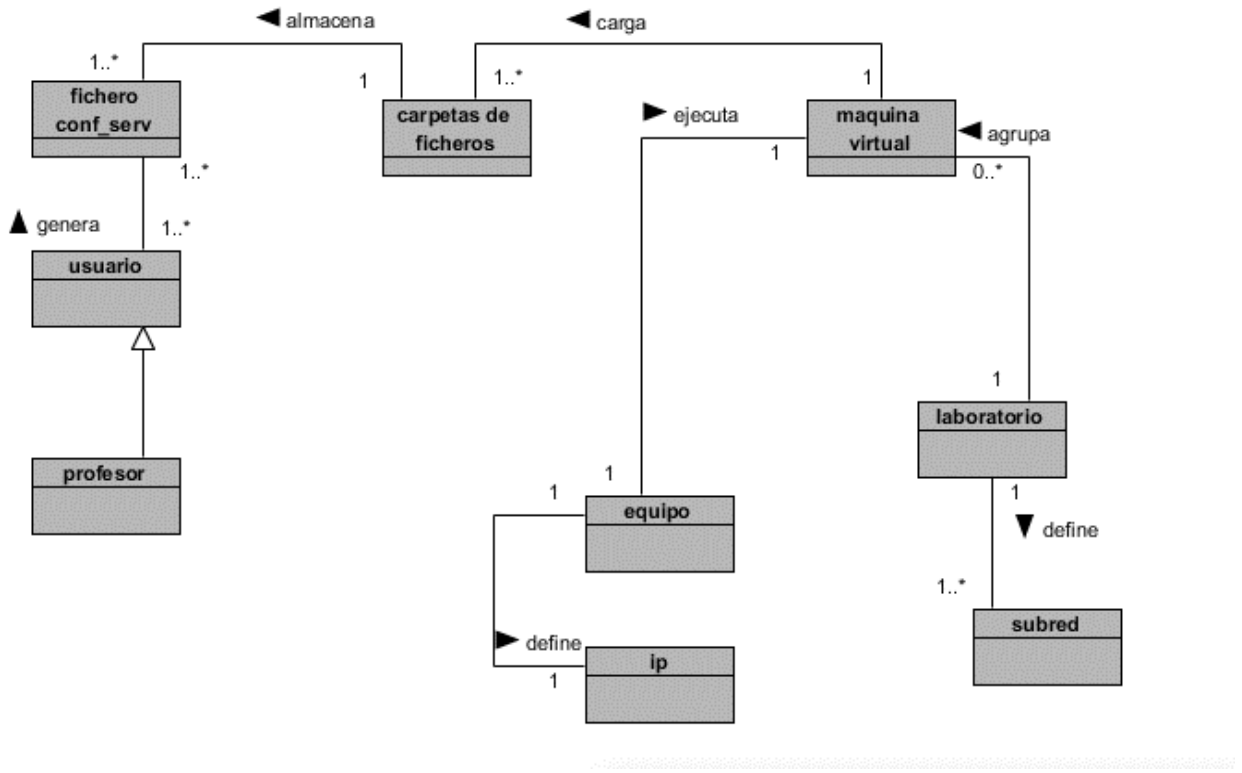


Figura 3. Modelo Conceptual

Descripción de las clases del modelo conceptual

Nombre	Descripción
Usuario	Profesor que utiliza la máquina virtual para generar los ficheros
Máquina virtual	Ejecuta los ficheros de configuración de los servicios y habilita las herramientas que se van a utilizar
Fichero conf_serv	Todos aquellos que genera el usuario para ejecutar en la máquina virtual para hacerle la parametrización básica.
Laboratorio	Aquellos que trabajan con las maquinas virtuales y definen su estructura a través de los equipos y las subredes
Subred	Es la que permite definir la estructura del laboratorio para la comunicación con la red

Equipo	Hardware que es empleado por el usuario para seleccionar los servicios y demás acciones a realizar.
IP	Dirección proporcionada por el servidor que identifica de manera lógica y jerárquica a la interfaz en la red, esta será asignada al equipo
Carpeta de ficheros	Donde se almacenarán todos los ficheros para la configuración de los servicios

Tabla 8. Descripción de las clases del modelo conceptual

Requisitos del sistema

El principal objetivo en determinar las funcionalidades que tendrá el sistema es guiar el desarrollo de la aplicación hacia la obtención de un diseño que cumpla con las expectativas requeridas por el cliente. La especificación de requisitos es la actividad que enumera y describe los requisitos necesarios para el sistema. Se dividen en dos grupos los funcionales (expresan capacidad) y los no funcionales (expresan cualidad).

Las técnicas de identificación de requisitos de software son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos: la entrevista es utilizadas para la recolección de opiniones, criterios o descripciones sobre diferentes actividades. Se lleva a cabo mediante conversaciones estructuradas donde es fundamental que la relación con el cliente esté basada en la confianza para dar a conocer la información de la manera más detallada.

Requisitos Funcionales del sistema: son las capacidades o condiciones que el sistema debe cumplir para satisfacer las necesidades primarias del cliente. A continuación, se definen los requisitos funcionales identificados para el desarrollo del sistema.

Numero	Nombre	Descripción	Complejidad
RF1	Editar usuario	Acción que permite al usuario escoger con que rol va a entrar al sistema	Media
RF2	Capturar parámetros	Acción que permite al usuario(profesor) seleccionar los parámetros que deben llevar los servicios con los que va a trabajar	Alta

RF3	Modificar parámetros	Acción que permite al profesor cambiar los parámetros seleccionados anteriormente por otros diferentes	Baja
RF4	Añadir subredes	Acción que permite al usuario adicionar la subred con la que va a trabajar el equipo	Alta
RF5	Listar subredes	Acción que permite al usuario visualizar la subred para así escoger con la que va a trabajar en el equipo	Baja
RF6	Modificar subredes	Acción que permite al usuario cambiar de subred con la que va a trabajar el equipo a su conveniencia según la necesidad.	Alta
RF7	Eliminar subredes	Acción que permite al usuario eliminar la subred que no le sea factible según la necesidad.	Alta
RF8	Añadir servicios	Acción que permite al usuario adicionar el servicio con el que estará trabajando	Alta
RF9	Eliminar servicios	Acción que permite al usuario eliminar el servicio que no necesita para trabajar.	Alta
RF10	Generar fichero	Acción que permite al usuario crear el fichero según el servicio seleccionado mediante la captura de parámetros	Alta
RF11	Editar fichero	Acción que permite al usuario hacer modificaciones al fichero en dependencia de del servicio seleccionado.	Alta
RF12	Eliminar fichero	Acción que permite al usuario eliminar el fichero.	Baja
RF13	Listar fichero	Acción que permite al usuario visualizar el fichero para así escoger el que necesite.	Media
RF14	Modificar repositorio	Acción que permite modificar el repositorio	Alta

Tabla 9. Requisitos funcionales

Requisitos no funcionales

Requisitos no Funcionales del sistema: Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener, aunque no formen parte de su función. A continuación, se definen los requisitos no funcionales identificados para el desarrollo del sistema.

RNF1. Usabilidad: el sistema debe mostrar una organización de la información que permita navegar por el software de manera intuitiva de acuerdo al orden lógico de la realización de las actividades del proceso, por lo que se debe tener en cuenta el orden visual reflejado en la jerarquía de la información.

RNF2. Rendimiento: el sistema debe ser rápido, el tiempo de respuesta debe ser como mínimo 2 segundos y un máximo de 5 segundos.

RNF3. Software: es necesario el navegador web Mozilla Firefox 17.0 o superior. Para los servidores se debe contar con el servidor web Nginx

RNF4. Hardware: las PC clientes deben contar con: Un microprocesador a velocidad de 1.0 GHz o superior; una memoria RAM de 512 MB o superior. El servidor donde estará instalada la aplicación debe tener: un microprocesador con una velocidad de 2.66 GHz o superior. Contará con 1 GB de memoria RAM y espacio mínimo en almacenamiento de 200 MB.

RNF5. Seguridad: debe mostrar a cada usuario sólo las funcionalidades del sistema sobre las cuales tiene permiso de acceso.

Historias de usuario

Las historias de usuario se usan, en el contexto de la ingeniería de requisitos ágil, como una herramienta de comunicación que combina la fortaleza de ambos medios: escritos y verbal. Describe, en una o dos frases, una funcionalidad de software desde el punto de vista del usuario, con el lenguaje que este emplearía. (45)

Historia de Usuario	
Número: 1	**Nombre:** editar usuario

Usuarios: Estudiante, profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 2	**Iteración Asignada:** 1
Descripción: el usuario será capaz de escoger su rol para poder desarrollar a partir de ahí los servicios que le corresponden	
Observaciones:	

Tabla 10. Historia de usuario 1

Historia de Usuario	
Número: 2	**Nombre:** capturar parámetros
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta
Estimación: 3	**Iteración Asignada:** 1
Descripción: el profesor tendrá la oportunidad de seleccionar los parámetros con los requisitos que debe llevar el fichero	
Observaciones:	

Tabla 11. Historia de usuario 2

Historia de Usuario	
Número: 13	**Nombre:** listar ficheros
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario podrá ser capaz de listar los ficheros a través de las carpetas de ficheros para seleccionar así con los que estará trabajando	
Observaciones:	

Tabla 12. Historia de usuario 13

Las restantes historias de usuario se pueden consultar en los anexos.

Diagrama de clase

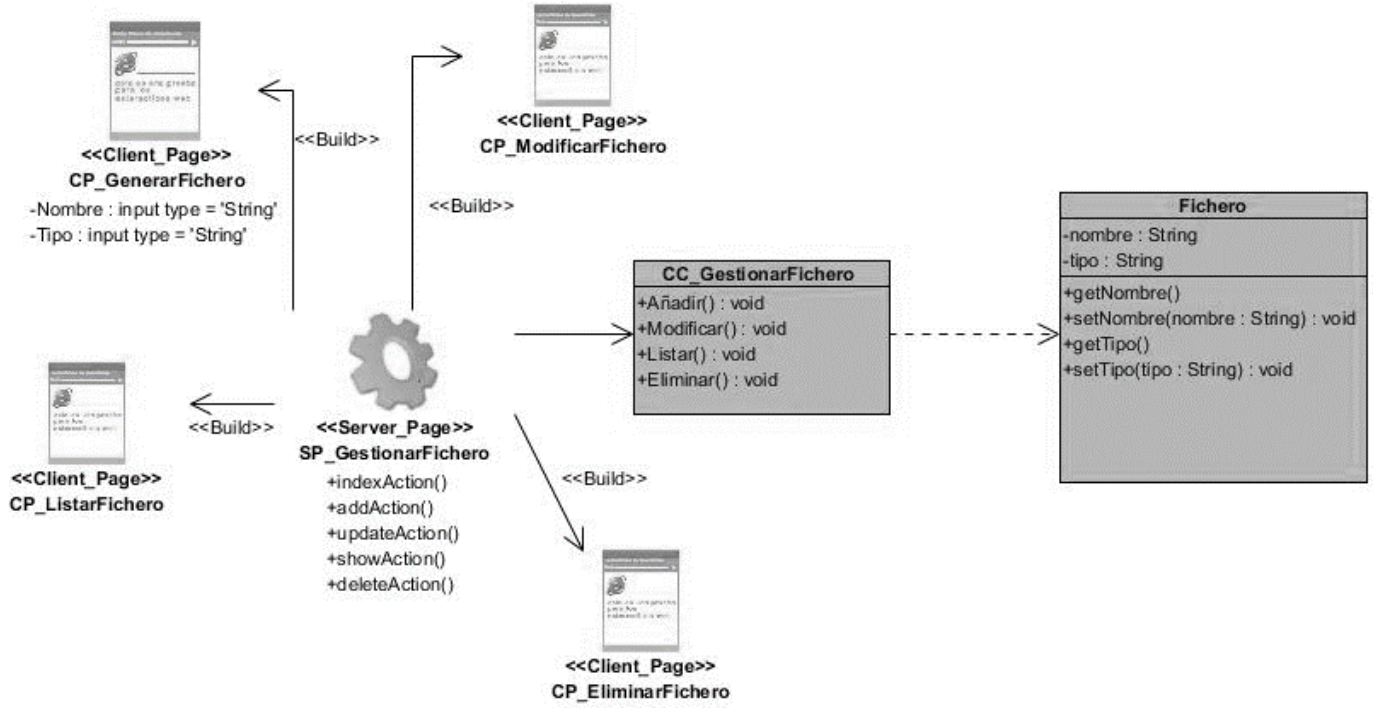


Figura 4. Diagrama de clases del diseño con estereotipos web

Planificación

Plan de iteraciones

Iteraciones	**HU a implementar**	**Tiempo**
1	Editar usuario, capturar parámetros, modificar parámetros.	4 semanas
2	Añadir subredes, modificar subredes, listar subredes, eliminar ficheros, listar ficheros.	3 semanas
3	Modificar repositorio, generar ficheros, editar ficheros.	4 semanas

Tabla 13. Plan de Iteraciones

Entrega

Plan de Entrega		
Iteración 1	**Iteración 2**	**Iteración 3**
20 de octubre de 2021	8 de noviembre de 2021	15 de noviembre de 2021

Tabla 14. Plan de entrega

Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones fisicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposicion de las instancias de componentes de ejecucion en instancias de nodos conectados por enlaces de comunicación.

Figura 5. Diagrama de despliegue

Descripción de la funcionalidad y capacidad del nodo

PC cliente: Ordenador cliente que se conecta a través de un navegador web al servidor central donde reside la aplicación.

Servidor: Servidor central, hospeda todos los componentes necesarios para el funcionamiento del producto. Tiene instalado un servidor web.

Descripción de elementos e interfaces de comunicación

https: Representa la conexión que se va a establecer entre una PC Cliente que se conectará con el servidor central; significa la conexión entre el navegador de usuario y el servidor del sistema.

Conclusiones del capitulo

A modo de conclusión se puede decir que en el capitulo se definieron los requisitos funcionales y no funcionales basados en la descripción del sistema, se estableció la arquitectura que debe tener el sistema a desarrollar, el cual permitirá tener ventajas significativas en las actividades académicas, además de los patrones de diseños que se ven reflejados en la implementación del sistema. Los artefactos generados durante el proceso de desarrollo en UML sirven de base y referencia para la adición en un futuro de mejoras.

Capitulo 3. Pruebas del sistema

Introducción

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas al software es un factor de vital importancia. Se refleja los casos de pruebas y el resultado que se obtendría al poner en prácticas las pruebas en el sistema.

Pruebas

Probar el software es una de las actividades más importante del ciclo de vida del desarrollo, pero tradicionalmente se ha llevado a cabo al final del proceso, cuando el proceso esta terminado y esta a punto de ser liberado. La complejidad del software actual exige que la prueba se ejecute de forma paralela al desarrollo, de tal manera que los errores se encuentren a tiempo y se puedan corregir. (49)

Para determinar el nivel de factibilidad y calidad se le realizaran pruebas al sistema mediante técnicas experimentales usando pruebas de caja negra, unitarias y de carga y estrés.

Pruebas de caja negra

- Pruebas que se llevan a cabo sobre la interfaz del módulo.
- El objetivo es demostrar que las funciones del módulo son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto (no se ve el código).

Pruebas de carga y estrés:

- Pruebas que se llevan acabo para identificar la cantidad de peticiones que el sistema puede soportar.
- En este tipo de pruebas se suele enviar mas peticiones de las que el software podría atender normalmente para saber el comportamiento del mismo.

Pruebas unitarias

Verifican que el nombre de la función o método sea adecuado, que los nombres y tipos de parámetros sean correctos y así mismo el tipo y el valor de lo que se devuelve como resultado.

Casos de pruebas

Las pruebas no garantizan la ausencia de defectos, pero si debe garantizar que el producto que es liberado a producción no tenga defectos de alto impacto en sus funcionalidades. El objetivo de las pruebas no es cero defectos, sino cumplir y demostrar las expectativas del usuario final del software, siempre y cuando la ética de la persona que realiza las pruebas esté orientada al cliente y comprometida con la calidad. (53)

A continuación, se muestra el caso de pruebas realizado a la historia de usuario: Capturar parámetros

Caso de prueba: Capturar parámetros

Descripción general

El sistema muestra una lista de parámetros donde el usuario debe seleccionar que servicios va a utilizar en un inicio.

Condición de ejecución

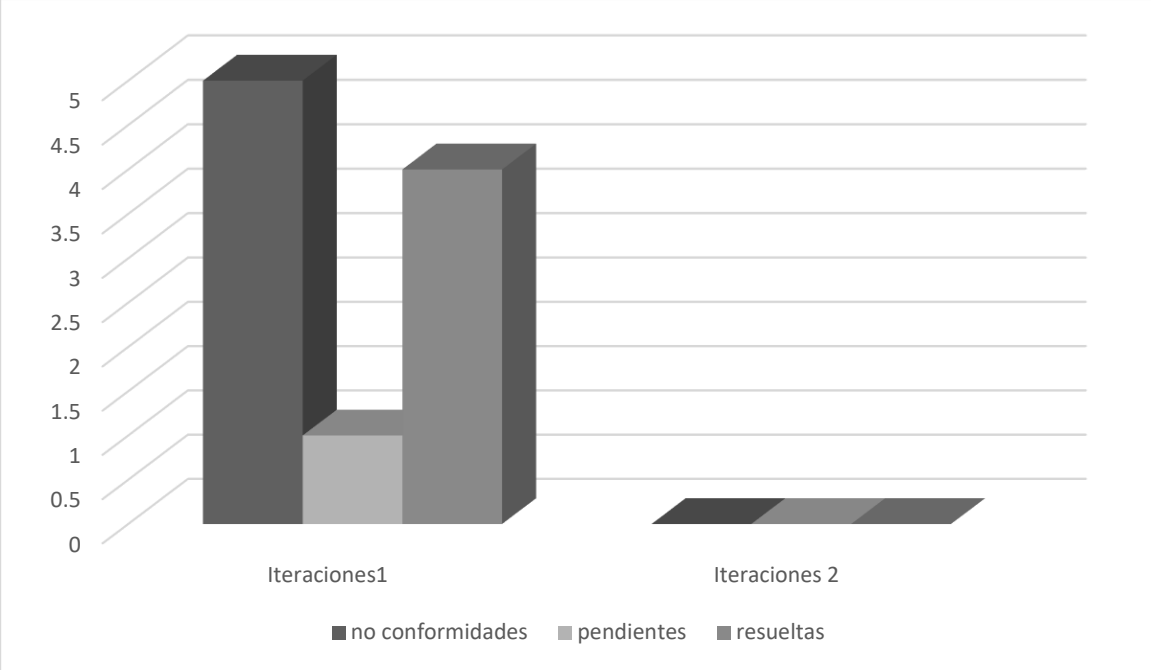
El usuario debe estar autenticado en el sistema.

Nombre de la sección	**Escenario de la sección**	**Descripción de la funcionalidad**	**Flujo del escenario**
SC 1: Capturar parámetros	EC 1.1 El usuario selecciona los parámetros.	El sistema muestra una lista con los servicios a seleccionar.	• Seleccionar los servicios • Click en el botón de continuar
	EC 1.2 El usuario no selecciona ningún parámetro.	El sistema muestra un mensaje indicando que debe seleccionar al menos un parámetro.	• Seleccionar los parámetros • Click en el botón de continuar

Tabla 15. Caso de prueba Capturar parámetros

Resultado de las pruebas funcionales

Las pruebas funcionales se realizaron en dos iteraciones donde se aplicaron los casos de prueba diseñados. A continuación, se muestran los resultados obtenidos en cada iteración de pruebas al sistema, así como la corrección de cada error.



Con el objetivo de validar los requisitos funcionales se realizaron dos iteraciones por requisito en las que se encontraron 5 no conformidades después de haber corregido las mismas, se realizó una segunda iteración en la que se encontraron 0 no conformidades quedando resueltas.

Las no conformidades encontradas fueron:

- errores de interfaz tales como vínculos defectuosos
- 2 validaciones incorrectas
- errores ortográficos
- idioma

Aplicación del método Delphi

El método de Delphi se basa en la entrega de un cuestionario a un panel de expertos de un determinado campo en el que se les pregunta su opinión sobre uno o mas temas en concreto. Las opiniones resultantes se incorporan de forma anónima al cuestionario a modo de retroalimentación. (54)

Este método fue seleccionado para comprobar el producto en caso de estar terminado. En esta etapa se puede realizar la valoración teórica del software, Delphi ofrece múltiples ventajas por ser confiable y con bajo margen de error

Método Delphi simplificado:

- Fase preparatoria
- Selección de los expertos
- Determinación de objetivos y elaboración de cuestionarios
- Fase de consulta
- Realización de la ronda de consulta
- Procesamiento estadístico de los resultados de la ronda
- Retroalimentación de los resultados del procesamiento de las respuestas
- (los pasos 3,4 y 5 se repetirían a lo largo de tantas rondas como marcase el diseño del estudio)
- Fase de resultados
- Determinación del consenso
- Informe de resultados

Estándar de código

Sería lógico afirmar que el código es leído más veces de lo que es escrito, por lo que se hace necesario con el objetivo de lograr consistencia y coherencia en el código establecer pautas a seguir por los programadores que permitan mejorar la legibilidad del mismo, a partir de estas necesidades surgen los estándares o

convenciones de código, los cuales definen un grupo de acuerdos para escribir código fuentes en la mayoría de los lenguajes de programación.

Los estándares de codificación acordados para el desarrollo de la presente solución son los definidos en el documento PEP 8—*Style Guide for* Python *Code*, por Guido van Rossum (54). A continuación, quedan definidos:

Indentacion:

- Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).

- Utilizar una *indentacion* de una tabulación para cada línea con excepción de la primera.

- La *indentacion* se realizará solamente con tabulaciones, no debe utilizarse nunca los cuatro (4) espacios.

Máxima longitud entre líneas:

- Todas las líneas deben estar limitadas a un máximo de setenta y nueve caracteres.

- Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas.

- En cualquier circunstancia se puede utilizar el carácter “\” para cortar las líneas largas.

Líneas en blanco:

- Separar las funciones de alto nivel y definiciones de clases con dos líneas en blanco.

- Las definiciones de métodos dentro de una clase deben separarse por una línea en blanco.

- Se puede utilizar líneas en blanco escasamente para separar secciones lógicas.

Codificaciones:

- Utilizar la codificación UTF-8.

- Se pueden incluir cadenas que no correspondan a esta codificación utilizando “\x”, “\u” o “\U”.

Importaciones:

- Las importaciones deben estar en líneas separadas.

- Siempre deben colocarse al comienzo del archivo.

Deben quedar agrupadas de la siguiente forma:

1. Importaciones de la librería estándar.

2. Importaciones terceras relacionadas.

3. Importaciones locales de la aplicación/librerías.

- Cada grupo de importaciones debe estar separado por una línea en blanco.

- Evitar utilizar espacios en blanco en las siguientes situaciones:

1. Inmediatamente dentro de paréntesis, corchetes y llaves.

2. Inmediatamente antes de una coma, un punto y coma o dos puntos.

3. Antes del paréntesis que comienza la lista de argumentos en la llamada a una función.

4. Inmediatamente antes de un corchete que empieza una indexación.

5. Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro.

Espacios en blancos en expresiones y sentencias:

- Deben rodearse con exactamente un espacio los siguientes operadores binarios:

1. Asignación (=).

2. Asignación de aumentación (+=, -=, etc.).

3. Comparación (==, <, >,>=, <=, !=, <>, in, not in, is, is not).

4. Expresiones lógicas (and, or, not).

- Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.

- No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.

Comentarios:

- Los comentarios deben ser oraciones completas.

- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula.

- Nunca cambiar las minúsculas y mayúsculas en los identificadores de clases, objetos, funciones, etc.

- Si un comentario es corto el punto final puede omitirse.

Cadenas de documentación:

- Deben quedar documentados todos los módulos, funciones, clases y métodos públicos.

- Para definir una cadena de documentación debe quedar encerrada dentro de (“”).

- Los (“”) que finalizan una cadena de documentación deben quedar en una línea a no ser que la cadena sea de una sola línea.

Convenciones de nombramiento:

- Nunca se deben utilizar como simple caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de los números uno y cero.

- Los módulos deben tener un nombre corto y en minúscula.

- Los nombres de clases deben utilizar la convención “*CapWords*” (palabras que comienzan con mayúsculas).

- Los nombres de las excepciones deben estar escrito también en la convención “*CapWords*” utilizando el sufijo “Error”.

- Los nombres de las funciones deben estar escrito en minúscula separando las palabras con un guion bajo “_”.

- Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guion bajo (_).

Diagrama de componente

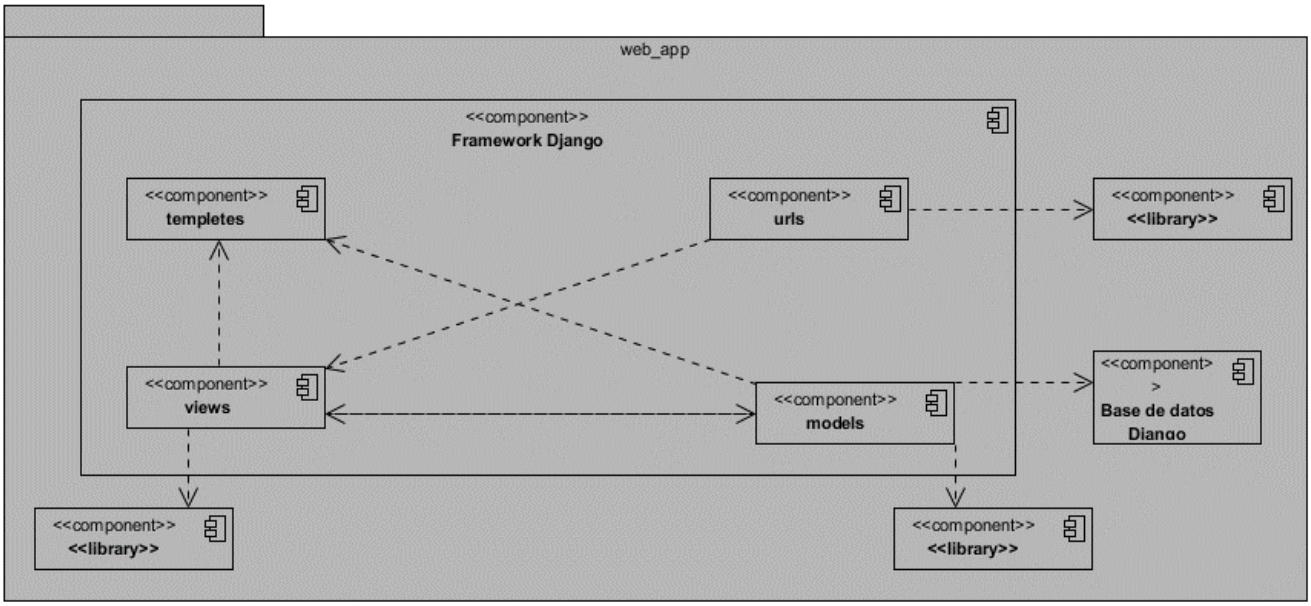


Figura 6. Diagrama componente

Conclusiones del capitulo

Después de haber desarrollado el presente capítulo se arribó a las siguientes conclusiones: La definición de un estándar de codificación permitió aumentar la claridad y reutilización del código, así como establecer pautas para una mejor comprensión del mismo. El desarrollo de un diagrama de componentes permitió una mayor comprensión estructural de la solución.

Conclusiones generales

Finalizada la investigación se arriba a las siguientes conclusiones: A partir de la revisión del marco teórico**,** se comprobó que no existe ninguna aplicación que pueda ser utilizada en el entorno UCI que resuelva la problemática planteada en todos sus aspectos. Una vez comprobada la necesidad del desarrollo de una interfaz parametrizable, se valoraron las metodologías, tecnologías y herramientas a utilizar. se describe la solución propuesta y el proceso realizado para el desarrollo del software que genere ficheros de configuración correspondiente a los servicios telemáticos. Se escogió la arquitectura del sistema y se detallaron los requisitos funcionales y no funcionales, plasmados en el artefacto. La modelación de los estos posibilitó establecer las características de los componentes, la organización lógica del código fuente. El sistema constituyese una solución funcional y con calidad, conforme a las pruebas funcionales, rendimiento y seguridad.

Recomendaciones

Para contribuir a la continuidad de la investigación, se hace la siguiente recomendación:

Realizar un estudio más profundo sobre la parametrización inicial de un escenario definido para actividades académicas sobre configuración de servicios telemáticos y si es posible implementar al completo el sistema y hacer todas las pruebas necesarias para su mejor funcionamiento.

Anexos

Historia de Usuario	
Número: 3	**Nombre:** modificar parámetros
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario se encarga de cambiar los parámetros para los servicios según la tarea que necesite realizar.	
Observaciones:	

Tabla 16 Historia de usuario 3

Historia de Usuario	
Número: 4	**Nombre:** añadir subredes
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario tendrá la opción de añadir la subred para definir la estructura del laboratorio.	
Observaciones:	

Tabla 17 Historia de usuario 4

Historia de Usuario	
Número: 5	**Nombre:** listar subredes
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario podrá ser capaz de listar las subredes con las que se trabajará en el laboratorio.	

Observaciones:

Tabla 18 Historia de usuario 5

Historia de Usuario	
Número: 6	**Nombre:** modificar subredes
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario modifica la subred que será utilizada definiendo así la estructura del laboratorio	
Observaciones:	

Tabla 19 Historia de usuario 6

Historia de Usuario	
Número: 7	**Nombre:** eliminar subred
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario eliminara la subred para así modificarla a su conveniencia	
Observaciones:	

Tabla 20 historia de usuario 7

Historia de Usuario	
Número: 8	**Nombre:** añadir servicios
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el profesor añadirá el servicio que desea utilizar para poder asignar las direcciones temporales al equipo (DNS, DHCP, FTP, HTTP, entre otros).	

Observaciones:

Tabla 21 Historia de usuario 8

Historia de Usuario	
Número: 9	**Nombre:** eliminar servicios
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario será capaz de eliminar el servicio que no le sea necesario.	
Observaciones:	

Tabla 22 historia de usuario 9

Historia de Usuario	
Número: 10	**Nombre:** generar ficheros
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** alta
Estimación: 3	**Iteración Asignada: 3**
Descripción: el profesor será capaz de generar todos los ficheros, permitiendo cumplir con todas las necesidades.	
Observaciones:	

Tabla 23 Historia de usuario 10

Historia de Usuario	
Número: 11	**Nombre:** editar ficheros
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada: 3**

Descripción: el usuario será capaz de editar los ficheros según la necesidad que se plantee.
Observaciones:

Tabla 24 Historia de usuario 11

Historia de Usuario	
Número: 12	**Nombre:** eliminar ficheros
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario eliminara los ficheros que no necesite para la configuración.	
Observaciones:	

Tabla 25 Historia de usuario 12

Historia de Usuario	
Número: 14	**Nombre:** modificar repositorio
Usuarios: profesor	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3	**Iteración Asignada:** 2
Descripción: el usuario podrá modificar el repositorio de Linux para mejorar la iteración y experiencia.	
Observaciones:	

Tabla 26 Historia de usuario 14

Referencias

1. **Mart, Alexander Anglada.** Marco de trabajo para el desarrollo de herramientas orientadas a la gestión e integración de servicios telemáticos de infraestructura en GNU/Linux. 2017.

2. *IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN DE PACIENTES Y HERRAMIENTAS TELEMATICAS EN LA ASOCIACION CREEMOS EN TI .* **MARTINEZ, OSCAR EDGAR ANDRES PERICO.** 2018.

3. **Moncada Jiménez, Yanina Alexandra.** *Las herramientas tecnológicas y el aprendizaje en entornos virtuales de los estudiantes de una institución educativa, Piura, 2020.* 2020.

4. **Rufino, D. Pousa y J.** Evaluation of type-1 hypervisors on desktop-class virtualization hosts. *IADIS International Journal on Computer Science and Information Systems.* 2017.

5. **ALGARNI, S. A., y otros.** Performance Evaluation of Xen, KVM, and Proxmox Hypervisors. *International Journal of Open Source Software and Processes.* 2018.

6. *Virtualización.* **Yenisleidy Fernández Romero, Karen García Pombo.** 3, 2011, Revista digital de las tecnologias de la informacion y las telecomunicaciones , Vol. 10.

7. **Sánchez, Álvaro García, y otros.** *Servicios de Red e Internet 2/E.* 2015.

8. **Aplicaciones Y Servicios Telemáticos -i-, Fundamentos DE y José Fernández Jiménez Fco Javier Muñoz Calle José Ángel Gómez Argudo Ignacio Campos Rivera Juan Antonio Ternero Muñiz, Francisco.** *BLOQUE I: ADMINISTRACIÓN BÁSICA DE S.O. LINUX PRÁCTICA 3: PROGRAMACIÓN SHELL-SCRIPT EN LINUX Práctica 3 Programación Shell-script en Linux.* 2016.

9. **Giner, Raul Sales.** *Comparativa de distribuciones GNU/Linux.* s.l. : Universitat Obreta de Catalunya, 2018.

10. *Importancia de los repositorios para preserrvar y recuperar la informacion.* **Cabrera, Elaine Durperet, Martinez, Denis Gabriel Perez y Cedeño, Mirtha Yris.** 10, Santiago de cuba : s.n., 2015, Vol. XIX.

11. **Elaine, Lic, y otros.** Los repositorios y su importancia para la preservación y recuperación de información. 2015, Vol. 19.

12. **Morales, Julia Añorga, Camejo, Lidisbet Cardoso y Alzate, Jhon H Sepulveda.** *La investigacion cientifica: una mirada desde los presupuestos teoricos de la educacion avanzada.* La Habana : s.n., 2015.

13. **Herrera, Victor.** *Enfoque de Desarrollo de Sistemas: SISTEMAS CON PAQUETES DE SOFTWARE DE APLICACIONES.* 2016.

14. *Sistemas de archivos: qué son y cuáles son los más importantes.* **2020.**

15. ***Sistema de ficheros en Linux: Todo sobre su estructura.*** **Castillo, José Antonio. 1 de ENERO de 2019.**

16. ***¿Qué es el Sistema de Archivos de Linux? Guía fácil.*** **Ebrahim, Mokhtar. 2018.**

17. ***Virtualizacion. Tipos y software utilizado.*** **2021.**

18. ***Tutorial: cómo configurar una máquina virtual en VirtualBox.*** **Alonso, Jose Francisco. 2020, Centro Integrado de Formación Profesional Número Uno de Santander.**

19. ***Guia de Bash Script .*** **B, Gustavo. 2020.**

20. ***Que es un fichero Bash?*** **Vericat, Alex. 2019.**

21. ***Explicación detallada del shell de Linux, bash y dash .*** **2020-2021.**

22. ***Archivos Dash Que es? Como abrir un archivo Dash?*** **2020-2021.**

23. ESPINOZA, ANGÉLICA JACKELINE OSORIO. ***CONFIGURACIÓN Y PARAMETRIZACIÓN DE LA PLATAFORMA DE REPOSITORIO DE DATOS "DSPACE" PARA LA BIBLIOTECA NÉSTOR GRAJALES LÓPEZ EN LA UNIDAD CENTRAL DEL VALLE DEL CAUCA.*** **2019 .**

24. SANCHEZ, JUAN SEBASTIAN CARRILLO. ***Diseño de la parametrizacion de un sistema de informacion para el area de cobranza del banco WWB.*** **2017.**

25. Ardila, Yisel Gómez. ***Soporte de la parametrización contable del software SERVINTE CLINICAL SUITE en el HOSPITAL INFANTIL DE SAN JOSÉ. .*** **2019.**

26. Maida, Esteban Gabriel y Pacienzia, Julian. ***Metodologia de desarrollo de software.*** **Argentina : s.n., 2015.**

27. Navarro, Mirtha E, y otros. ***Seleccion de metodologias agiles e integracion de arquitecturas de software en desarrollo de sistemas de informacion.*** **Buenos Aires : s.n., 2017.**

28. Sanchez, Tamara Rodriguez. ***Metodologia de desarrollo para la actividad productiva de la UCI.*** **la Habana : s.n., 2015.**

29. Neira, Bruno Chavarria y A, Edisson Gudiño De La. ***Implementacion de un servidor web y un diseño de una pagina utilizando herramientas de software libre para el dispensario ¨Sagrada familia¨ de la ciudad de Gayaquil.*** **2017.**

30. ***Que es NGINX y como funciona?*** **B, Gustavo. 2020.**

31. ***Servidor web Nginx, una clara alternativa a Apache.*** **Madrid : s.n., 2015.**

32. ***Que es Nginx? Caracteristicas y ventajas de este servidor web.*** **Borges, Santiago. 2019.**

33. ***Qué es un lenguaje de programación.*** **Mendoza, Marvin Lopez. 2020.**

34. ***Estos serán los 10 lenguajes de programación más demandados en 2018.*** **Gonzalez, R. Martinez. 2018.**

35. ***Levantamiento de los principales procesos para el Restaurante y Servicio de Cáterin Alexander; y automatización del proceso de inventario y el proceso de gestión de reserva de mesas mediante una aplicación basada en Java aplicando la metodología de Progra.*** **Cunalata, D. Moran J. Quito-Ecuador : s.n., 2016, págs. 16-20.**

36. Arias, M. A. ***Aprende Programación Web con PHP y MySQL.*** **2017. págs. 13-17.**

37. PHP. ***¿Qué es PHP?.*** **2018.**

38. Martinez, K. ***Todo sobre PHP- Ventajas y desventajas. .*** **2011.**

39. ***¿Por qué usar Django?.*** **Urquiaga, J. C. Mariños. 2016.**

40. ***¿Cuáles son las principales debilidades de Python como lenguaje de programación?. .*** **Garcia, A. 2017.**

41. Jimmy Rolando Molina Ríos, Nancy Magaly Loja Mora, Mariuxi Paola Zea Ordóñez, Erika Lizbeth Loaiza Sojos. ***Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python .*** **2016.**

42. ***Qué es un IDE?*** **Falcon, Y. Suarez. 2016.**

43. Lopez, Manuel Barrera. ***Desarrollo de una aplicacion para el estudio de los modelos de prevision de la demanda .*** **Sevilla : s.n., 2020.**

44. ***Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso.*** **Hernández, Lionel R. Baquero, y otros. No. 7, La Habana : "Ediciones Futuro", 2016, Vol. Vol. 9.**

45. ***Curso Django: Entendiendo como trabaja Django.*** **Montero, S. Infante. 2012.**

46. ***"Phython - Django Framework de desarrollo web para perfeccionistas basado en el modelo MTV".*** **Ayala, J. L. Condri. 2012.**

47. ***Python.*** **Fuentes, O. A. F. Garcia. 2016.**

48. Larman, Craig. ***UML y patrones.*** **1999.**

49. ***Buenas practicas de programacion.*** **Nuñez, Rafael Manotas. 2016.**

50. ***Plataforma para la integracion de componentes en un sistema de laboratorio remoto.*** **Cabrera, Eric Gutierrez y Alvarez, Yisell Gato. .9, Cuba : Ediciones futuro, 2018, Vol. .11, págs. 33-49.**

51. alexander Menzinsky, Gertrudis Lopez, Juan Palacios, Miguel Angel Sobrino, Ruben Alvarez, Veronica Rivas. ***Historias de usuario.*** **2020.**

52. ***Una revision a la realidad de la automatizacion de las pruebas del software.*** **Serna, Edgar, Martinez, Raquel y Tamayo, Pula. no.1, Mexico : s.n., 2021, Vol. Vol.23.**

53. ***Priorización de casos de prueba en entornos de desarrollo ágil.*** **Loor-Intiago, Jose Miguel, Dapena, Martha Dunia Delgado y Oliva, Perla Beatriz Fernandez. La Habana : s.n., 2020.**

54. ***Guia para la utilizacion de la metodologia Delphi en las etapas de comprobacion de productos terminados tipo software educativo.*** **Garcia, Frank Hernandez y Castillo, Jose Ignacio Robaina. Ciego de Avila : s.n., 2017.**

55. Rossum, Guido van. ***PEP 8--Style Guide for Python Code.*** **2018.**

56. ***REDES Y EDUCACIóN.*** **Adell, Jordi. 1998.**

57. Aranda, Vicente Trigo. Historia y evolución de Internet. ***Manual formativo de ACTA.*** **2004, Vol. 33.**

58. Problemas, E N. LABORATORIO VIRTUAL BASADO EN LA METODOLOGÍA DE APRENDIZAJE BASADO. 2009.

59. Dayer, Juan Antonio Castillo. ***" Diseño e implementación de una aplicación Web para la administración de recursos de investigación del Área de Ingeniería Telemática ".*** **Cartagena : s.n., 2007.**

60. Britos, Daniel Vargas, Laura Arias, Silvia Giraudo, Nicolás Veneranda, Guillermo. Tercera Conferencia de Directores de Tecnología de Información y Comunicación en Instituciones de Educación Superior : Soluciones para la Enseñanza y la Investigación . Laboratorio Remoto Virtual para la Enseñanza de Administración de Redes. 2013.

61. Bourne, E. ***Programación en Shell - Dialnet.***

62. Garcia-Rivero, Alexis Alejandro y Gonzalez-Argote, Javier. Repositorio de investigaciones estudiantiles: tarea necesaria y trascendental. 2020, Vol. 21.

63. Bruce Maxim, Roger S. Pressman. ***Engenharia de Software - 8ª Edição.*** **2016.**

64. ***Lenguaje de programacion Java .*** **Falcon, S. 2018.**

65. ***Virtualizacion -Tipos y software.*** **2021, OpenWebinars .**

66. ***Sistemas de archivos: qué son y cuáles son los más importantes.*** **2020.**

67. Wei-Ling Wu, Ian Hartono Budianto, Chun-Foong Wong, Samuel Ken-En Gan. ***A Review of Apps for Programming: programming languages and making apps with apps.*** **2019.**

68. ***Aspectos básicos de C#.*** **Wagner, Bill. 2020.**

69. ***El patron de diseño MTV en Django y su relacion con el MVC.*** **Cruz, Andres. 2020.**

70. Cruz, Alberto, y otros. ***Especificacion del patron de diseño Memento a traves de un perfil UML.*** **2020.**

71. Guerrero, Brayan Nicolas Aceros, y otros. ***Cartilla de actividades para la aplicación de los patrones de diseño de software básico para los estudiantes del programa de Ingeniería de Sistemas de las Unidades Tecnológicas de Santander. .*** **Santander : s.n., 2020.**