

**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**FACULTAD 1**



**Módulo de autenticación con el directorio activo de Nova-LTSP**

**Trabajo de diploma para optar por el título de Ingeniero en Ciencias  
Informáticas**

**Autor:**

Alejandro Rodas Cueto

**Tutores:**

Ing. Yadiel Pérez Villazón

Ing. Lisdania de la Caridad Delgado Olivera

**LA HABANA, JUNIO DE 2018**  
**“AÑO 60 DE LA REVOLUCIÓN”**

## **DECLARACIÓN DE AUTORÍA**

Declaro por este medio que yo Alejandro Rodas Cueto, con carné de identidad 94061423968 soy el autor principal del trabajo titulado “Módulo de Autenticación con el Directorio Activo de Nova-LTSP” 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 firmo la presente a los \_\_\_\_días del mes de junio del año 2018.

---

Firma del Autor

---

Ing. Lisdania de la Caridad Delgado Olivera

Firma de la Tutora

---

Ing. Yadiel Pérez Villazón

Firma del Tutor

## DEDICATORIA

*Dedico este triunfo de mi vida principalmente a mi madre, mis abuelos y mi hermano. A mi pedacito de cielo y familiares. A mi bebé Isabella María y a todas esas personas que tienen en mí un lugar especial.*

## **AGRADECIMIENTOS**

*A mis tutores y consultante Yadiel Pérez Villazon, Lisdania de la C. Delgado Olivera y Lexys Manuel Díaz Alonzo por estar involucrados en la guía durante el desarrollo de este proceso de tesis.*

*A todos los profesores que fueron forjando mi crecimiento académico durante todo mi transcurso en la universidad, en especial a Ivaniel quien sin tener obligación alguna siempre estuvo dispuesto a ayudarme.*

*A todos mis compañeros, esos que ahora están aquí presentes y a los que fueron quedando durante el transcurso de la carrera, por todos esos momentos de locura, tensión, presión y diversión que pasamos.*

*A mis amigos, especialmente a José, Humberto y Lachy por escucharme, aguantarme y animarme a seguir adelante. Gracias por estar no sólo en los buenos momentos.*

*En especial:*

*A mi abuelo Pancho que tanto me consentía cuando pequeño y guiaba a medida de sus posibilidades, a mi abuela Ana que siempre me tiraba de las orejas cuando me salía del buen camino, a mi bisabuela Lola que era toda dulzura conmigo, a mi abuela Adela (abuela mima) que me mira desde el cielo y sé que estaría muy orgullosa de mi en este momento. A mi hermano Adrián que sin él los momentos no serían tan divertidos y además por posibilitar mi tranquilidad mental al estar lejos de casa porque sé que él está ahí, a mi tío Yoyi por suplir el papel de padre cuando es necesario, tanto conmigo como con mi hermano, a mi novia Bexy por ayudarme en todo cuanto puede y por todos los momentos, malos y buenos que hemos pasado juntos, a toda mi familia en general por darme siempre su apoyo y **MÁS ESPECIAL AÚN** a mi madre, por todos los sinsabores que le he hecho pasar y aun así estar incondicionalmente para mí en todo momento, sabiendo aconsejarme en cada una de las situaciones y darme mi palmadita en la espalda para seguir adelante.*

## RESUMEN

La distribución cubana GNU/Linux Nova cuenta con la plataforma de administración de clientes ligeros Nova-LTSP que brinda un conjunto de funcionalidades para administrar estos clientes. Sin embargo, tiene limitantes en la autenticación de los usuarios debido a que estos deben acceder primero con una contraseña local del servidor y otra para consumir cualquier servicio telemático que requiera de credenciales. No se utiliza el directorio activo para la gestión de todos los usuarios, lo cual hace perder diversas ventajas en cuanto a la organización, pierde control desde un solo punto de los permisos a los recursos, no puede controlar el comportamiento de los equipos y permisos de los usuarios de forma muy concreta, entre otras. Debido a esta situación la presente investigación tuvo como objetivo desarrollar un módulo de autenticación con el directorio activo de Nova-LTSP. La propuesta de solución estuvo guiada por la metodología de desarrollo de *software* Variación AUP-UCI, se seleccionó para la implementación Python 3.5 como lenguaje de programación sobre el *framework* Django en su versión 2.4.10 y como entorno integrado de desarrollo PyCharm2018. Esta solución permite a los administradores administrar las configuraciones de PBIS para unir el servidor LTSP al dominio del Directorio Activo, administrar la autenticación de los usuarios al Directorio Activo, devolver el estado de un usuario y listar los usuarios del directorio activo. Las pruebas realizadas arrojaron como resultado que el módulo implementado responde a los requerimientos definidos por el cliente.

**Palabras clave:** autenticación, clientes ligeros, módulo, Nova-LTSP

## ÍNDICE DE CONTENIDO

INTRODUCCIÓN .....	10
CAPÍTULO 1: Fundamentación teórica sobre el proceso de autenticación a través de los controladores de dominio.....	15
1.1 Clientes Ligeros .....	15
1.1.2 Arquitectura cliente-servidor .....	15
1.1.3 Proceso de autenticación a través de los controladores de dominio en Windows.....	16
1.1.4 LTSP .....	17
1.1.5 Proceso de autenticación a través de los controladores de dominio en Nova-LTSP.....	17
1.2 Estudio de sistemas homólogos.....	18
1.2.1 Pasos para unir a un dominio con Nova-LTSP .....	22
1.3 Metodología de desarrollado de software .....	23
1.4 Lenguaje y herramienta de modelado.....	24
1.4.1 Lenguaje unificado UML .....	24
1.4.2 Herramienta CASE .....	25
1.4.3 Visual Paradigm .....	25
1.5 Lenguajes y herramientas de desarrollado .....	26
1.5.1 Python .....	26
1.5.2 PyCharm .....	27
1.5.3 Framework .....	28
1.5.4 Django.....	28
1.5.5 Bootstrap .....	29
1.5.6 JQuery.....	29
1.6 Conclusiones parciales .....	29

CAPÍTULO 2: Análisis y diseño de la propuesta de solución .....	31
2.1 Propuesta de solución .....	31
2.4 Requisitos no funcionales .....	32
2.5 Historias de Usuario.....	33
2.6 Diseño del módulo .....	39
2.6.1 Arquitectura de software .....	39
2.6.2 Patrones de diseño.....	41
2.7 Diagrama de paquetes.....	46
2.8 Modelo de despliegue.....	48
2.9 Conclusiones parciales .....	49
CAPÍTULO 3: Implementación y pruebas de la propuesta de solución .....	50
3.1 Modelo de implementación .....	50
3.2 Estándares de codificación .....	52
3.3 Pruebas de software .....	55
3.3.1 Estrategias de prueba.....	55
3.4 Valoración económica, novedad y aporte social .....	69
3.5 Conclusiones parciales .....	70
CONCLUSIONES GENERALES .....	71
RECOMENDACIONES .....	1
REFERENCIAS BIBLIOGRÁFICAS .....	2
ANEXOS.....	7
Anexo1: Entrevista con el cliente .....	7
Anexo2: Acta de Aceptación .....	8
Anexo3: Logro alcanzado de la investigación .....	9

## ÍNDICE DE TABLAS

Tabla 1: Listado de requisitos funcionales .....	32
Tabla 2: Historia de usuario .....	<b>¡Error! Marcador no definido.</b>
Tabla 3: Historia de usuario .....	35
Tabla 4: Historia de usuario .....	37
Tabla 5: Historia de usuario .....	38
Tabla 6: Técnica de camino básico .....	57
Tabla 7: Tabla de rutas .....	60
Tabla 8: Prueba de unidad #1 .....	61
Tabla 9: Prueba de unidad #2 .....	61
Tabla 10: Prueba de unidad #3 .....	61
Tabla 11: Caso de prueba de aceptación #1 .....	63
Tabla 12: Caso de prueba de aceptación #2 .....	64
Tabla 13: Caso de prueba de aceptación #3 .....	65
Tabla 14: Caso de prueba de aceptación #4 .....	65
Tabla 15: Resultado de pruebas del sistema .....	68



## ÍNDICE DE ILUSTRACIONES

Ilustración 1: Arquitectura cliente servidor.....	16
Ilustración 2: Funcionamiento del MTV de Django .....	39
Ilustración 3: Separación de las clases de la aplicación .....	40
Ilustración 4: Patrón Experto .....	42
Ilustración 5: Patrón Creador .....	43
Ilustración 6: Patrón Bajo Acoplamiento.....	44
Ilustración 7: Patrón Alta cohesión .....	45
Ilustración 8: Patrón Controlador.....	46
Ilustración 9: Diagrama de paquetes.....	47
Ilustración 10: Modelo de despliegue .....	48
Ilustración 11: Diagrama de componentes .....	51
Ilustración 12: Gráfica del programa que describe el flujo de control lógico .....	59
Ilustración 13: Resultado pruebas de validación .....	66
Ilustración 14: Acta de aceptación .....	8
Ilustración 15: Logro alcanzado de la investigación .....	9

## INTRODUCCIÓN

La migración hacia plataformas GNU/Linux es de suma importancia en países con bajo presupuesto económico debido a que reduce los costos por licencias, tanto por el sistema operativo instalado como por las aplicaciones que se ejecutan en él. En el mundo, el uso y la creación de distribuciones de sistemas basados en tecnologías de software libre y código abierto de tipo GNU/Linux, han ido evolucionando progresivamente. Esto se debe a que les permiten a los usuarios ciertas libertades como: estudiar, ejecutar, copiar, distribuir, cambiar y mejorar el software, teniendo como elemento significativo el acceso al código fuente. El empleo de soluciones de código abierto permite en general disponer del código fuente y realizar las adaptaciones necesarias acorde a las necesidades del usuario final y no depender de un software privativo para consumir cualquier servicio telemático.

El concepto de telemática refiere a la combinación de la informática y de la tecnología de la comunicación para el envío y la recepción de datos. La noción se asocia a diferentes técnicas, procesos, conocimientos y dispositivos propios de las telecomunicaciones y de la computación (1). Actualmente los servicios telemáticos se despliegan en servidores en un sistema operativo que generalmente requiere de una autenticación por parte del usuario para el consumo de los mismos. Debido que los servidores cuentan con un gran número de usuarios y diferentes servicios con características únicas, y todos protegidos con usuario/contraseña y que provienen de diferentes fabricantes, desarrollados en diferentes lenguajes y plataformas podrían facturar inconvenientes de no estar centralizado el proceso de autenticación mediante un Directorio Activo.

El Directorio Activo (*Active Directory*) o Controlador de Dominio es la herramienta que se utiliza para la organización y gestión de los recursos de una red de ordenadores y todo lo que ello implica: usuarios, servicios, puestos, impresoras, permisos y servidores. Mediante esta herramienta se puede asignar derechos de usuario y permisos de control de acceso a los consumidores del dominio según sea necesario. El servicio de Controlador de Dominio proporciona la capacidad de establecer un único inicio de sesión y un repositorio central de información para toda su infraestructura, lo que simplifica ampliamente la administración de usuarios y equipos, proporcionando además la obtención de un acceso mejorado a los recursos en redes, sin él muchas funcionalidades (directivas de grupo, las jerarquías de dominio y la instalación “centralizada” de aplicaciones) no se podrían llevar a cabo (2).

Cuba no es la excepción de la utilización del Controlador de Dominio y debido a esto la Oficina de Seguridad para las Redes Informáticas (OSRI) dictamina que en los sistemas en que es posible el acceso por múltiples usuarios se dispondrá para cada uno de ellos de un identificador de usuario personal y único. Las personas a las que se asignen identificadores de usuarios responden por las acciones que con ellos se realicen (3). El uso de un Directorio Activo responde perfectamente a lo que plantea la OSRI, por lo que se resume esta resolución a lo siguiente: para cada institución que utilice servicios telemáticos donde se gestionan usuarios se debe utilizar un Directorio Activo para la gestión de los mismos<sup>1</sup>. Para hacer uso de este Directorio Activo la distribución cubana GNU/Linux Nova, cuenta con la plataforma de administración de clientes ligeros LTSP (*Linux Terminal Server Project*) que es un conjunto de aplicaciones que proporcionan la capacidad de ejecutar Linux en computadores de pocas prestaciones o de bajo costo (clientes ligeros), requisito que en los sistemas GNU/Linux actuales, incluyendo a Nova, se logra mediante la autenticación con dos credenciales. De mantenerse esta forma de autenticación y no utilizar un directorio activo se estarían perdiendo muchas ventajas específicas en diversos aspectos, como son:

- **Organización:** permite crear grupos para facilitar la administración. Ejemplo, puedes crear un grupo con los usuarios de un departamento.
- **Permisos:** control desde un sólo punto de los permisos a los recursos de la red. Ejemplo: puedes asignar a una carpeta permisos de lectura a un departamento, mientras que otros no pueden entrar y ciertas personas tienen control total.
- **Autenticación:** cualquier usuario puede entrar en otro equipo de la red con sus credenciales, y tendrá los permisos que se le asignaron. Ejemplo, cuando Pepe cambie de ordenador no hay que volver a configurar.
- **Políticas:** puedes controlar el comportamiento de los equipos y permisos de los usuarios de forma muy concreta. Ejemplo: cambiar el fondo de pantalla del escritorio en todos los equipos, o la caducidad de las contraseñas de todos los usuarios.
- **Escalabilidad:** es un sistema que funciona en un solo servidor con tres usuarios y para miles de usuarios repartidos por varias sedes y varios servidores repartiendo la carga.

<sup>1</sup> RESOLUCION No. 127 /2007del Ministerio de la Informática y las Comunicaciones, Sección Quinta, artículo 45 (Resolución 127 Sobre el Reglamento de Seguridad Informática, 2007)

- **Autenticación externa:** permite que otras aplicaciones lean los datos. Ejemplo: una aplicación de contabilidad no requiere otra clave para entrar, lee la del usuario en el directorio activo.
- **Replicación:** implementa características para la replicación de todos los datos entre servidores del directorio activo. Ejemplo: si la empresa tiene dos sedes los usuarios, permisos, ... se sincronizan solos automáticamente.

En correspondencia con lo antes expuesto se plantea como **problema de investigación:** ¿Cómo garantizar la autenticación de los usuarios con un controlador de dominio en Nova-LTSP, para brindar una autenticación centralizada de los clientes ligeros?

La presente investigación centra su **objeto de estudio** en el mecanismo de autenticación de servidores mediante controladores de dominio, enmarcado en el **campo de acción:** El proceso de autenticación a través de los controladores de dominio en Nova-LTSP. Para darle solución al problema anteriormente planteado se define el siguiente **objetivo general:** Desarrollar un módulo para Nova-LTSP que permita la autenticación de los usuarios con un controlador de dominio en Nova-LTSP, para brindar una autenticación centralizada de los clientes ligeros. Dicho objetivo general se desglosa en los siguientes **objetivos específicos:**

1. Elaborar los referentes teóricos de la investigación sobre la autenticación con un controlador de dominio en Nova-LTSP.
2. Diseñar un módulo para Nova-LTSP que permita la autenticación de los usuarios con un controlador de dominio.
3. Implementar el módulo para Nova-LTSP que permita la autenticación de los usuarios con un controlador de dominio.
4. Evaluar el módulo para Nova-LTSP que permita la autenticación de los usuarios mediante la aplicación de métodos, técnicas y pruebas.

Como sustento y guía de la presente investigación se formulan las siguientes preguntas científicas:

1. ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de autenticación de los usuarios con un controlador de dominio en Nova-LTSP?

2. ¿Qué aspectos deben tenerse en cuenta para realizar el diseño del módulo para la autenticación de los usuarios con un controlador de dominio en Nova-LTSP?
3. ¿Cuáles son las herramientas y tecnologías adecuadas para implementar el módulo de autenticación de los usuarios con un controlador de dominio en Nova-LTSP?
4. ¿Qué métodos, técnicas y pruebas aplicar para la evaluación del módulo de autenticación de los usuarios con un controlador de dominio en Nova-LTSP?

Para facilitar el cumplimiento del objetivo propuesto y de las tareas de investigación se emplean **métodos teóricos** de la investigación científica.

#### **Métodos teóricos:**

- ✓ **Histórico-Lógico:** Es utilizado en el análisis de los sistemas homólogos, de manera que permita buscar elementos que los caractericen y aspectos para fundamentar la propuesta de solución a la problemática planteada.
- ✓ **Analítico-Sintético:** Es utilizado con el fin de descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

#### **Métodos empíricos**

- ✓ **Entrevista:** Se realizó una entrevista al cliente para obtener información acerca de las funcionalidades que debía cumplir el módulo ( Ver Anexo 1 ).

El presente documento de trabajo de diploma está compuesto por introducción, tres capítulos, conclusiones generales, recomendaciones, referencias bibliográficas y anexos.

**Capítulo 1: Fundamentación teórica sobre el proceso de autenticación a través de los controladores de dominio:** Se definen los principales conceptos asociados al dominio del problema que son indispensables para el desarrollo y comprensión de esta investigación. Además, se definen las herramientas y tecnologías que se van a utilizar en el desarrollo de la aplicación.

**Capítulo 2: Análisis y diseño de la propuesta de solución:** Se describe la propuesta de solución para resolver el problema planteado y se especifican los requisitos funcionales y no funcionales. Por otra parte, se describen las historias de usuario correspondientes a los requisitos funcionales, la arquitectura y los patrones de diseño que se utilizarán durante la implementación.

**Capítulo 3: Implementación y pruebas de la propuesta de solución:** Se documentan los artefactos asociados a la implementación de la propuesta de solución y se realizan los casos de prueba para lograr el desarrollo de un software con la calidad requerida. Además, se establecen los estándares de codificación que se tuvieron en cuenta para el desarrollo del sistema.

## **CAPÍTULO 1: Fundamentación teórica sobre el proceso de autenticación a través de los controladores de dominio**

### **Introducción**

En el presente capítulo se abordan los conceptos fundamentales que sustentan la presente investigación. Se realiza un estudio del proceso de autenticación para finalmente seleccionar los aspectos a tener en cuenta en la propuesta de solución, según el resultado obtenido. Además, se definen la metodología de desarrollo y los lenguajes y herramientas a utilizar para elaborar la propuesta de solución.

Para lograr un mejor entendimiento del problema de investigación se tuvo en cuenta conceptos y procesos importantes tales como: clientes ligeros y arquitectura cliente servidor. Estas definiciones guiarán el desarrollo de la propuesta de solución. A continuación, se definen los mismos:

#### **1.1 Clientes Ligeros**

Un cliente ligero es un ordenador de bajas prestaciones que depende principalmente del servidor central para las tareas de procesamiento. Este servidor se encarga de distribuir los escritorios virtuales entre todos los clientes. Basa su eficiencia en la utilización de los recursos mínimos para su funcionamiento, un cliente ligero no procesa ningún tipo de dato, por lo que no se requiere una máquina potente, dejando ese trabajo al equipo servidor (4).

##### **1.1.2 Arquitectura cliente-servidor**

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es un solo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivos y los servidores de correo. Mientras que sus

propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma (5)

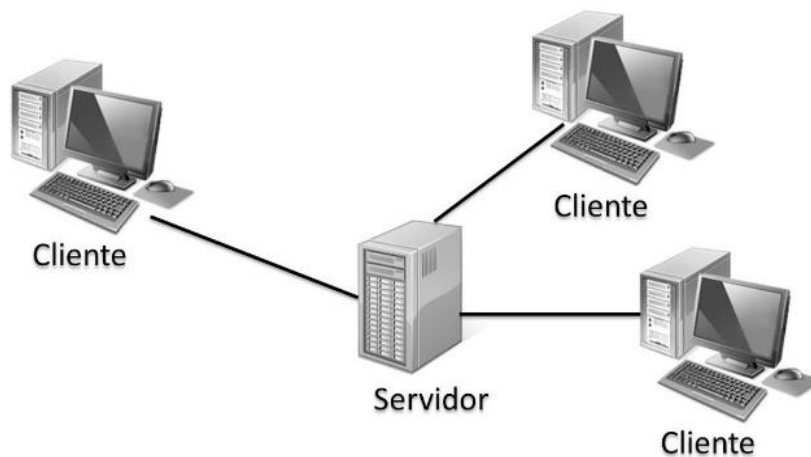


Ilustración 1: Arquitectura cliente servidor

### 1.1.3 Proceso de autenticación a través de los controladores de dominio en Windows

Es el proceso mediante el cual el sistema operativo recibe las credenciales de un usuario o el servicio y asegura que la información de presentación futura en el destino de autenticación. En el caso de un equipo unido al dominio, el destino de autenticación es el controlador de dominio. Las credenciales de autenticación son documentos digitales que se asocian a la identidad del usuario a algún tipo de prueba de autenticidad, como un certificado, una contraseña o un PIN.

De forma predeterminada, se validan las credenciales con la base de datos de administrador de cuentas de seguridad o SAM (*Security Account Manager*) en el equipo local o de *Active Directory* en un equipo unido al dominio, a través del servicio de *Winlogon*. Las credenciales se recopilan a través de la entrada del usuario en la interfaz de usuario de inicio de sesión o mediante programación a través de la interfaz de programación de aplicaciones abreviada como API (*Application Programming Interface*).

La información de seguridad local se almacena en el registro bajo *HKEY\_LOCAL\_MACHINE\SECURITY*. La información almacenada incluye la configuración de directiva, valores de seguridad predeterminados e información de cuenta, como las credenciales de inicio de sesión en caché. También se almacena una copia de la base de datos de SAM aquí, aunque está protegido contra escritura (6).



#### 1.1.4 LTSP

LTSP es una colección de software que convierte una instalación GNU/Linux normal en un servidor de terminal. Esto permite usar clientes ligeros de baja potencia y bajo costo (o hardware viejo que ya se tenga) para ser utilizados como terminales para armar una red servidor-clientes (7).

#### 1.1.5 Proceso de autenticación a través de los controladores de dominio en Nova-LTSP

Originalmente, Linux (las GNU herramientas y las bibliotecas que se ejecutan en él) no se creó con un mecanismo de autenticación único en mente. Como consecuencia de esto, los programadores de aplicaciones de Linux tardaron en crear su propio esquema de autenticación. Administra para ello por buscar los nombres y algoritmos *hash* de contraseña en */etc/passwd* (archivo contenedor de Linux de credenciales de usuario) o proporcionar un mecanismo totalmente diferente e independiente. La gran cantidad resultante de mecanismos de autenticación era difícil de administrar. En 1995, Sun<sup>2</sup> propone un mecanismo denominado módulos de autenticación conectable. PAM (*Pluggable Authentication Modules* o en español módulos de autenticación conectables) proporciona un conjunto común de autenticación de las API que podría utilizar todos los desarrolladores de aplicaciones, junto con un servidor configurado por el administrador. Mediante las API PAM para la autenticación y el nombre de servidor cambiar (NSS) las API para buscar información del usuario, los programadores de aplicaciones de Linux podrían escribir menos código y los administradores de Linux podría tener un único lugar para configurar y administrar el proceso de autenticación. La mayoría de las distribuciones de Linux se suministran con varios módulos de autenticación PAM, incluidos los módulos que admiten la autenticación a un directorio LDAP (*Lightweight Directory Access Protocol*) y la autenticación mediante *Kerberos*. Puede utilizar estos módulos para autenticarse en *Active Directory* (8).

#### LDAP

<sup>2</sup> The Sun es un periódico en formato tabloide publicado en el Reino Unido. Es actualmente el diario más leído en idioma inglés, con una tirada de alrededor de 3 200 000 ejemplares y unos 8 500 000 lectores

LDAP (“Lightweight Directory Access Protocol”, «Protocolo Ligero de Acceso a Directorios») es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio. Se usó inicialmente como un *front-end* o interfaz final para X.500, pero también puede usarse con servidores de directorio únicos y con otros tipos de servidores de directorio (9).

## **KERBEROS**

Kerberos es un protocolo de autenticación de redes de ordenador creado por el MIT (Instituto Tecnológico de Massachusetts o Instituto de Tecnología de Massachusetts) que permite a dos ordenadores en una red insegura demostrar su identidad mutuamente de manera segura. Sus diseñadores se concentraron primeramente en un modelo de cliente-servidor; y brinda autenticación mutua, tanto cliente como servidor verifican la identidad uno del otro (10).

### **1.2 Estudio de sistemas homólogos**

#### **PBIS**

*Power Broker Open* es un proyecto de comunidad de código abierto patrocinado por *Beyond Trust Software*. El proyecto *Power Broker Open* (anteriormente conocido como *Likewise Open*) se lanzó en diciembre de 2007. Desde entonces, más de 100.000 organizaciones del sector público y privado han utilizado *Power Broker Open* para la autenticación central de un Directorio Activo para sistemas Linux, UNIX y Mac. El primer módulo del proyecto *Power Broker Open* es *Power Broker Identity Services*, o PBIS. PBIS es una implementación completa, compatible con Windows, basada en una arquitectura moderna. Brinda una solución modular y programática que presta especial atención a la claridad, extensibilidad y usabilidad. (11)

#### **Características:**

- ✓ *Power Broker Identity Services* utiliza PAM (*Pluggable Authentication Modules*) y NSS (*Name*

*Service Switch*). Admite la autenticación Kerberos, NTLM<sup>3</sup>, SPNEGO<sup>4</sup>

- ✓ PowerBroker Open está disponible bajo una licencia GPL / LGPL v2 o con una licencia comercial.
- ✓ Se une a los sistemas Linux, UNIX y Mac OS a *Active Directory* en un solo paso a través de una herramienta GUI o desde la línea de comandos.
- ✓ Autentica a los usuarios con un solo nombre de usuario y contraseña en sistemas Windows y no Windows.
- ✓ Almacena las credenciales en caché, por lo que si pierde el acceso a la red o el controlador de dominio está inactivo, sigue permitiendo la correcta autenticación del usuario siempre y cuando se halla autenticado satisfactoriamente antes y no se hayan cambiado las credenciales en el servidor.

## **Winbind**

Winbind es componente de la suite de programas Samba que resuelve los problemas de inicio de sesión unificados, está pensada para organizaciones que tienen una infraestructura existente basada en dominio NT en la cual desean poner estaciones trabajo o servidores de UNIX. Winbind permitirá que estas organizaciones desplieguen las estaciones de trabajo de UNIX sin tener que mantener una infraestructura de cuentas separada. Está diseñado sobre una arquitectura cliente/servidor. Un demonio *Winbindd* escucha en un socket Unix esperando las solicitudes. Estas peticiones son normalmente de los clientes NSS y PAM y se procesan secuencialmente. (12)

Winbind unifica la gestión de cuentas UNIX y Windows NT permitiendo a una máquina Unix volverse un miembro completo de un dominio NT. Una vez hecho, la máquina Unix verá a los usuarios y grupos NT

<sup>3</sup> La autenticación NTLM predeterminada y la autenticación Kerberos usan las credenciales de usuario de Microsoft Windows NT asociadas a la aplicación que realiza la llamada para intentar la autenticación con el servidor. Cuando se usa la autenticación NTLM no predeterminada, la aplicación establece el tipo de autenticación en NTLM y usa un objeto *NetworkCredential* para pasar el nombre de usuario, la contraseña y el dominio al host. (51).

<sup>4</sup> El mecanismo del protocolo SPNEGO (Simple and Protected GSS-API Negotiation Mechanism) permite negociar con el navegador para establecer el mecanismo de autenticación que debe utilizarse. El navegador suministra información de autenticación *Kerberos*. (52)

como si fueran usuarios y grupos nativos permitiendo usar el dominio NT de la misma forma que se usa NIS+ <sup>5</sup> en entorno exclusivos UNIX.

El resultado final es que cuando cualquier programa en la máquina Unix solicita al sistema operativo que busque un nombre usuario o un grupo, la consulta se resuelve preguntando al controlador de dominio NT del dominio correspondiente. Como winbind enlaza con el sistema operativo a bajo nivel (vía módulo de resolución NSS de la biblioteca C), la redirección al controlador de dominio NT es completamente transparente.

Los usuarios de la máquina Unix pueden utilizar los nombres de usuarios y grupos NT como si fueran nombres *nativos*. Se pueden cambiar propietarios de ficheros para que sean propiedad de usuarios de usuarios del dominio NT o incluso iniciar una sesión en la máquina Unix y lanzar una sesión X-Window como miembro del dominio.

La única indicación obvia de que se está usando Winbind que los nombres de usuarios y grupos tienen la forma *DOMAINuser* and *DOMAINgroup*. Esto es necesario para permitir a Winbind determinar a qué controlador de dominio hay que redireccionar para la búsqueda particular y qué dominios de confianza se están referenciando.

### **Características:**

La integración de UNIX y Microsoft Windows NT a través de un inicio de sesión ha sido considerada durante mucho tiempo una quimera en entorno heterogéneos de computación.

<sup>5</sup> NIS + es un servicio de directorio desarrollado por Sun Microsystems para reemplazar su antiguo 'NIS' (Servicio de información de red). Está diseñado para eliminar la necesidad de duplicar en muchas computadoras los datos de configuración como cuentas de usuario, nombres de host y direcciones, información de la impresora y montajes de discos NFS en sistemas individuales, en lugar de usar un repositorio central en un servidor maestro, simplificando la administración del sistema.

Hay alguna otra utilidad sin la cual la interoperabilidad entre redes UNIX y Microsoft Windows sufriría mucho. Es necesario que haya un mecanismo para compartir ficheros entre sistema UNIX y que pueda asignar como propietarios a usuarios y grupos del dominio con integridad.

Winbind es un componente de la suite de programas Samba que resuelve los problemas de inicio de sesión unificados. Winbind usa una implementación UNIX de las llamadas RPC de Microsoft, PAM (*Pluggable Authentication Modules*), y el servicio de nombres (NSS, *Name Service Switch*) para permitir a los usuarios de dominios NT aparecer y operar como usuarios UNIX en una máquina UNIX.

Winbind proporciona tres funciones separadas:

- ✓ Autenticación para credenciales de usuario (vía PAM).
- ✓ Resolución de identidad (vía NSS).
- ✓ Winbind mantiene una base de datos llamada `winbind_idmap.tdb` en la cual guarda las asociaciones entre UNIX UIDs / GIDs y NT SIDs. Esta asociación se usa sólo para usuario y grupos que no tienen unos UID/GID locales. Guarda los UID/GID del rango `uid/gid` de `idmap` que ha asociado al SID NT. Si se ha especificado el *backend idmap* como `ldapsam:url`, entonces en lugar de usar la asociación local Winbind la obtendrá de la base de datos LDAP.

## **SSSD**

SSSD proporciona un conjunto de demonios para administrar el acceso a directorios remotos y mecanismos de autenticación. Proporciona una interfaz NSS y PAM para el sistema y un sistema *backend* conectable para conectarse a múltiples fuentes de cuenta diferentes, así como a la interfaz D-Bus. También es la base para proporcionar auditoría de clientes y servicios de políticas para proyectos como FreeIPA. Proporciona una base de datos más robusta para almacenar usuarios locales, así como también datos de usuarios extendidos. (13)

## **Resultados del estudio de sistemas homólogos**

Se selecciona la herramienta Pbis para unir el servidor LTSP debido a que es la que posee las características que más se relacionan a lo que se plantea como propuesta de solución, debido a las

siguientes razones:

Likewise contiene características muy similares a lo que se quiere realizar en la propuesta de solución, pero Pbis es la versión más actual de Likewise.

- ✓ Winbind y SSSD, en comparación a Pbis carecen en temas como mayor sencillez en su configuración para la unión a un dominio, posibilita la autenticación central de un Directorio Activo para sistemas Linux, UNIX y Mac uniéndose a los dominios de un Directorio Activo en un solo paso a través de una herramienta GUI o desde la línea de comandos. Por otra parte, Winbind mantiene una base de datos llamada winbind\_idmap.tdb en la cual guarda las asociaciones entre UNIX UIDs / GIDs y NT SIDs. SSSD proporciona una base de datos más robusta para almacenar usuarios locales, así como también datos de usuarios extendidos lo cual los hace más complicados en su uso. Por otro lado, PBIS, almacena las credenciales en caché, por lo que si pierde el acceso a la red o el controlador de dominio está inactivo, sigue permitiendo la correcta autenticación del usuario siempre y cuando se halla autenticado satisfactoriamente antes y no se hayan cambiado las credenciales en el servidor. Por otra parte, esta cumple con todos los requisitos necesarios para llevar a cabo la correcta confección del módulo y fue la utilizada en todas las migraciones realizadas al sistema operativo GNU/LINUX Nova.

### **1.2.1 Pasos para unir a un dominio con Nova-LTSP**

1. Instalar la herramienta PBIS.
2. Establecer el nombre que tendrá la imagen en el dominio.
3. Editar tres ficheros de configuración para especificar el “host”, “dominio e IP de salida de la entidad” y “exportar la configuración”.
4. Ejecutar el comando que une la imagen al dominio.
5. Se comprueba que la imagen este unida al dominio.
6. Reiniciar el servidor de PBIS.
7. Se verifica el estado del servidor de PBIS.
8. Se configuran los usuarios y grupos para que puedan tener acceso al dominio.
9. Se configuran las PAM.

### 1.3 Metodología de desarrollado de software

“El Proceso Unificado Ágil (AUP, por sus siglas 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 ágiles y conceptos que aún se mantienen válidos en RUP. Esta metodología aplica las técnicas más comunes del desarrollo ágil (14):

- ✓ Desarrollo dirigido por pruebas.
- ✓ Modelado ágil.
- ✓ Gestión de cambios ágil.
- ✓ Refactorización de la base de datos.

A continuación, se describe la metodología seleccionada para guiar el proceso de desarrollo de software de la presente investigación:

**AUP-UCI** es una variación de la metodología AUP6, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. En aras de aumentar la calidad del software que se produce esta metodología se apoya en el Modelo CMMI-DEV v1.3. (por sus siglas en inglés Capability Maturity Model Integration Development, Integración de Modelos de Madurez de Capacidades para Desarrollo). El mismo constituye una guía para aplicar las mejores prácticas y obtener un producto o servicio con calidad en una entidad desarrolladora

#### **Fases Variación AUP-UCI**

- ✓ **Inicio:** El objetivo de esta fase es llevar a cabo las actividades relacionadas con la planeación del proyecto. Se realiza un estudio inicial de la organización que actúa como cliente y se obtiene información clave acerca del alcance del proyecto, se realizan estimaciones de tiempo y esfuerzo, y finalmente se decide si se ejecuta o no.
- ✓ **Ejecución:** En esta etapa se ejecutan las actividades requeridas para desarrollar el *software*.

<sup>6</sup> Proceso Unificado Ágil, del inglés *Agile Unified Process*.

Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.

- ✓ **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se llevan a cabo las actividades formales de cierre de proyecto.

Se selecciona la metodología AUP-UCI en su escenario 4 pues se adapta al ciclo de vida definido para la actividad productiva de la UCI, y da la posibilidad al cliente de siempre acompañar al equipo de desarrollo para convenir los requisitos y así poder implementarlos. Además, fue aprobada cuando la institución se sometió al proceso de certificación CMMI nivel 2 para el desarrollo de *software*.

## 1.4 Lenguaje y herramienta de modelado

El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización (15)

### 1.4.1 Lenguaje unificado UML

El Lenguaje Unificado de Modelado es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Muestra las decisiones y los conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre tales sistemas (16).

Para llevar a cabo el diseño del módulo se propone utilizar UML, en su versión 2.0, por las siguientes características:

- ✓ **Visualizar:** permite expresar de una forma gráfica un sistema de forma tal que otro lo pueda entender.
- ✓ **Especificar:** permite especificar cuáles son las características de un sistema antes de su construcción.
- ✓ **Construir:** a partir de los modelos especificados se pueden construir los sistemas diseñados.



- ✓ **Documentar:** los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

### **Ventajas y desventajas del UML (17) :**

#### Ventajas

- ✓ Se puede usar para diferentes tipos de sistemas
- ✓ Consolida muchas de las notaciones y conceptos más usadas orientados a objetos.
- ✓ Es fácilmente entendible

#### Desventajas

- ✓ No es un método de desarrollo.
- ✓ Al no ser un método de desarrollo es independiente del ciclo de desarrollo
- ✓ No se presta con facilidad al diseño de sistemas distribuidos.

### **1.4.2 Herramienta CASE**

Las herramientas de Ingeniería de Software Asistidas por Computadora (por sus siglas en inglés *Computer-Aided Software Engineering*, CASE) son algunas de las herramientas claves para el desarrollo de aplicaciones informáticas, este tipo de herramienta se utiliza para ayudar a actividades del proceso de *software* tales como la ingeniería de requisitos, el diseño, el desarrollo de aplicaciones y las pruebas (18).

### **1.4.3 Visual Paradigm**

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (19).

Se empleará Visual Paradigm, en su versión 8.0, para modelar los diagramas que se generen en cada una de las etapas del proceso de desarrollo de *software*, además de brindar las siguientes ventajas.

- ✓ **Dibujo:** facilita el modelado de UML, proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que se ajusta al estándar soportado por la herramienta.
- ✓ **Corrección sintáctica:** controla que el modelado sea correcto
- ✓ **Coherencia entre diagramas:** dispone de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, esto evita las duplicidades.
- ✓ **Integración con otras aplicaciones:** permite integrarse con otras aplicaciones como herramientas ofimáticas, lo cual aumenta la productividad.
- ✓ **Trabajo multiusuario:** permite el trabajo en grupo, proporciona herramientas de compartición de trabajo.
- ✓ **Reutilización:** dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- ✓ **Generación de código:** permite generar código de forma automática, esto reduce los tiempos de desarrollo y evita errores en la codificación del *software*.
- ✓ **Generación de informes:** permite generar diversos informes a partir de la información introducida en la herramienta.

## 1.5 Lenguajes y herramientas de desarrollo

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (20).

### 1.5.1 Python

*Python* es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y multiplataforma. Es administrado por la *Python*

*Software Foundation*. Posee una licencia de código abierto, denominada *Python Software Foundation License*, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1 (21).

Se utilizará Python 3.5, esta versión contiene muchas de las características que se lanzaron por primera vez en Python 3.1. Las mejoras en esta versión incluyen (22):

- ✓ Un tipo de diccionario ordenado.
- ✓ Nuevas funciones de prueba unitarias que incluyen omisión de prueba, nuevos métodos de afirmación descubrimiento de prueba.
- ✓ Un módulo mucho más rápido.
- ✓ Establecer y comprensión de diccionarios.
- ✓ Vistas del diccionario.
- ✓ Nueva sintaxis para anidado con declaraciones.

### **1.5.2 PyCharm**

PyCharm es un entorno de desarrollo integrado (IDE) utilizado en la programación de computadoras, específicamente para el lenguaje Python. Proporciona análisis de código, un depurador gráfico, un probador de unidades integrado, integración con sistemas de control de versiones (VCS) y es compatible con el desarrollo web con Django además de ser multiplataforma, con versiones de Windows, macOS y Linux. Para la realización de la propuesta de solución se hará uso de PyCharm2018.

PyCharm ofrece las siguientes funciones (23):

- ✓ Auto-completamiento de código.
- ✓ Señalamiento de errores con soluciones fáciles.
- ✓ Posibilita una fácil navegación para proyectos y código.
- ✓ Mantiene el código bajo control de chequeos, asistencia de pruebas, refactorizaciones y un conjunto de inspecciones que posibilitan codificar de forma limpia y sostenible.

### 1.5.3 Framework

Un *framework*, entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software (24).

### 1.5.4 Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–vista–controlador. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés *Don't Repeat Yourself*). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos (25).

Django ofrece las siguientes facilidades (26) :

- ✓ Sistema de plantillas para separar la presentación de un documento de sus datos.
- ✓ Construcción automática de interfaces de administración.
- ✓ Vistas genéricas que recogen ciertos estilos y patrones comunes en su desarrollo y los abstraen, de modo que se puede escribir rápidamente vistas comunes de datos sin tener que escribir mucho código.
- ✓ Sistema para que no tengan que ser recalculadas cada vez que se piden.
- ✓ Integración con bases de datos y aplicaciones existentes.
- ✓ Construcción de aplicaciones multilenguaje permitiendo especificar cadenas de traducción de más de cuarenta idiomas.

Para el desarrollo del módulo se decidió usar Django, en su versión 2.4.10, pues es un marco de trabajo de desarrollo web totalmente implementado sobre Python con el que se pueden crear y mantener aplicaciones de alta calidad.

### 1.5.5 Bootstrap

Bootstrap es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como “responsive design” o diseño adaptativo. Para el desarrollo del módulo se utiliza Bootstrap, en su versión 3.0, porque se integra además con la biblioteca JQuery y está basado en herramientas actuales y potentes como CSS3 y HTML5 (27).

Para el desarrollo del módulo se utiliza Bootstrap, en su versión 3.0, porque se integra además con la biblioteca JQuery y está basado en herramientas actuales y potentes como CSS3 y HTML5.

### 1.5.6 JQuery

*JQuery* es un framework de **JavaScript** para interactuar con los documentos HTML, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. Consiste en un único fichero JavaScript que contiene las funcionalidades comunes del DOM<sup>7</sup> (Document Object Model), eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX (28).

Para el desarrollo del módulo se decidió utilizar JQuery, en su versión 1.10.3, para las validaciones no funcionales en la parte del cliente, por ser un marco de trabajo que facilita la selección de elementos HTML, la creación de animaciones y evita la implementación de funcionalidades comunes.

## 1.6 Conclusiones parciales

Se abordaron conceptos asociados al dominio de la presente investigación, permitiendo una mejor comprensión del objeto de estudio. El análisis de sistemas de similar propósito permitió determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución. La selección de la metodología, herramientas y tecnologías posibilitó obtener la base

<sup>7</sup> DOM define un estándar para acceder a documentos HTML y XML:

tecnológica de la propuesta de solución. Para guiar el proceso de desarrollo del módulo se adoptó como metodología de desarrollo de software AUP-UCI. Se definió utilizar la herramienta CASE *Visual Paradigm* 8.0 para el modelado de los artefactos del análisis y diseño de la solución. Por otra parte, para la implementación se utilizó *Python* 3.5 como lenguaje de programación sobre el *framework* Django en su versión 2.4.10, apoyado en el IDE PyCharm2018.

## **CAPÍTULO 2: Análisis y diseño de la propuesta de solución**

En el presente capítulo se establecen las principales características con que cuenta la solución, ya sean requisitos funcionales o no funcionales, describiendo las historias de usuario correspondientes. Además, se describe la arquitectura y los patrones de diseño a aplicar. También se describe la propuesta de solución donde se incluye la arquitectura del *software* para conocer la manera en que se estructura, el diagrama de paquetes, el diagrama de clases, dejando así bien claro cómo se confeccionara la propuesta de solución.

A continuación, se expone la propuesta del módulo para la autenticación con el directorio activo de Nova-LTSP, donde se describen las principales funcionalidades que el mismo va a poseer.

### **2.1 Propuesta de solución**

En la presente investigación se propone el desarrollo de un módulo que permita, desde Nova-LTSP administrar las configuraciones de PBIS para unir el servidor LTSP al dominio del Directorio Activo, administrar la autenticación de los usuarios al Directorio Activo, devolver el estado de un usuario, y listar los usuarios del Directorio Activo.

Dentro de la sección Directorio Activo de la plataforma Nova-LTSP se encontrarán todas las funcionalidades correspondientes al módulo de autenticación con el Directorio Activo de Nova-LTSP. El módulo permitirá instalar la herramienta PBIS y realizar todas las configuraciones necesarias en ella para garantizar la unión del servidor LTSP con un Directorio Activo al insertar datos al sistema como Nombre del servidor LTSP, Dirección IP del Directorio Activo, Dirección IP del Directorio Activo (DNS), Usuario y Clave. También permite administrar la autenticación de los usuarios alojados en este Directorio Activo lo cual posibilita activar o desactivar la autenticación mediante un cliente ligero a los usuarios al directorio activo, además de devolver el estado de un usuario para conocer si el usuario tiene activado la autenticación mediante un cliente ligero al directorio activo o no, y por ultimo nos deja conocer todos los usuarios registrados en el Directorio Activo al cual está unido el servidor LTSP mostrándolo en un listado.

### **2.2 Requisitos**

En la ingeniería de sistemas y la ingeniería de software, la Ingeniería de requisitos o Ingeniería de requerimientos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requisitos de las partes interesadas, que pueden entrar en conflicto entre ellos (29).

### 2.3 Requisitos funcionales

Tabla 1: Listado de requisitos funcionales

No.	Nombre del requisito funcional	Descripción
<b>Prioridad</b>		<b>Alta</b>
RF1	Administrar las configuraciones de PBIS para unir el servidor LTSP al dominio del Directorio Activo	Permite la unión del servidor LTSP a un dominio
RF2	Administrar la autenticación de los usuarios al Directorio Activo	Activa o desactiva la autenticación mediante un cliente ligero a los usuarios al directorio activo
RF3	Mostrar el estado de todos los usuarios	Permite conocer de cada usuario si tiene activado o no la autenticación mediante un cliente ligero al directorio activo
RF4	Listar los usuarios del directorio activo	Lista todos los usuarios del directorio activo

### 2.4 Requisitos no funcionales

Un requisito no funcional (RNF) especifica los criterios que se deben usar para calificar el funcionamiento de un sistema, en lugar de un comportamiento específico. En general, los requisitos funcionales definen lo que el sistema debería de hacer, mientras que los requisitos no funcionales verifican cómo un sistema debería de ser. Requisitos no funcionales son a menudo llamados las “cualidades de un sistema” (30).



## **Restricciones del diseño e implementación**

**RnF1:** Se utilizará el lenguaje de programación Python y el *framework* Django.

**RnF2:** La interfaz visual debe mantener el estilo y diseño de la plataforma de administración de clientes ligeros Nova-LTSP.

**RnF3:** El sistema cumplirá con los patrones GRASP: Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador seleccionados por el equipo de desarrollo.

## **Eficiencia**

**RnF4:** El sistema debe permitir que los usuarios interactúen con él de manera concurrente.

**RnF5:** El sistema debe ser capaz de responder 5000 peticiones en 5 segundos como máximo.

## **Soporte**

**RnF6:** Se podrá acceder al sistema desde los navegadores web Mozilla Firefox en su versión 44 y Chrome en su versión 48.

## **Hardware**

**RnF7:** El sistema se puede desplegar en un servidor que disponga de 128 Mb de memoria RAM o superior y 50 Mb de disco duro o superior.

## **2.5 Historias de Usuario**

En la metodología AUP-UCI una de las formas que se utiliza para describir los requisitos de *software* son las Historias de Usuario (HU). Se describe brevemente y en un lenguaje natural, las características que el sistema debe poseer. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en un corto período de tiempo (31).

Tabla 2: Historia de Usuario

Historia de Usuario	
<b>Número:</b> HU_1	<b>Nombre del requisito:</b> Administrar la autenticación de los usuarios al Directorio Activo
<b>Programador:</b> Alejandro Rodas Cueto	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 40 horas
<b>Riesgo en Desarrollo:</b> Alto	<b>Tiempo Real:</b> 40horas
<p><b>Descripción:</b> La HU inicia cuando el usuario autenticado en la plataforma Nova-LTSP con privilegios de administración accede al menú Directorio activo, ubicado en el panel izquierdo de la plataforma y selecciona la opción: “Administrar las configuraciones de PBIS”. Muestra una pestaña al usuario en la cual se recibirán los datos que se le pasarán al método para su ejecución.</p> <p><b>Nombre del servidor LTSP</b> (Campo obligatorio, campo de texto, este campo solo puede contener letras y caracteres)</p> <p><b>Dirección IP del Directorio Activo</b> (Campo obligatorio, campo de texto, este campo solo puede contener direcciones IP validas)</p> <p><b>Dirección IP del Directorio Activo</b> (Campo obligatorio, campo de texto, este campo solo puede contener un DNS valido)</p> <p><b>Usuario</b> (Campo obligatorio, campo de texto)</p> <p><b>Clave</b> (Campo obligatorio, campo de texto)</p> <p>El usuario introduce la información y presiona el botón: “Salvar”. El sistema verifica la información y envía un mensaje de confirmación, finalizando así la HU.</p>	
<p><b>Observaciones:</b></p> <ol style="list-style-type: none"> <li>1. Si el usuario no posee la herramienta Pbis, el sistema emite un mensaje notificando el error.</li> </ol>	
<b>Prototipo elemental de interfaz gráfica del usuario</b>	

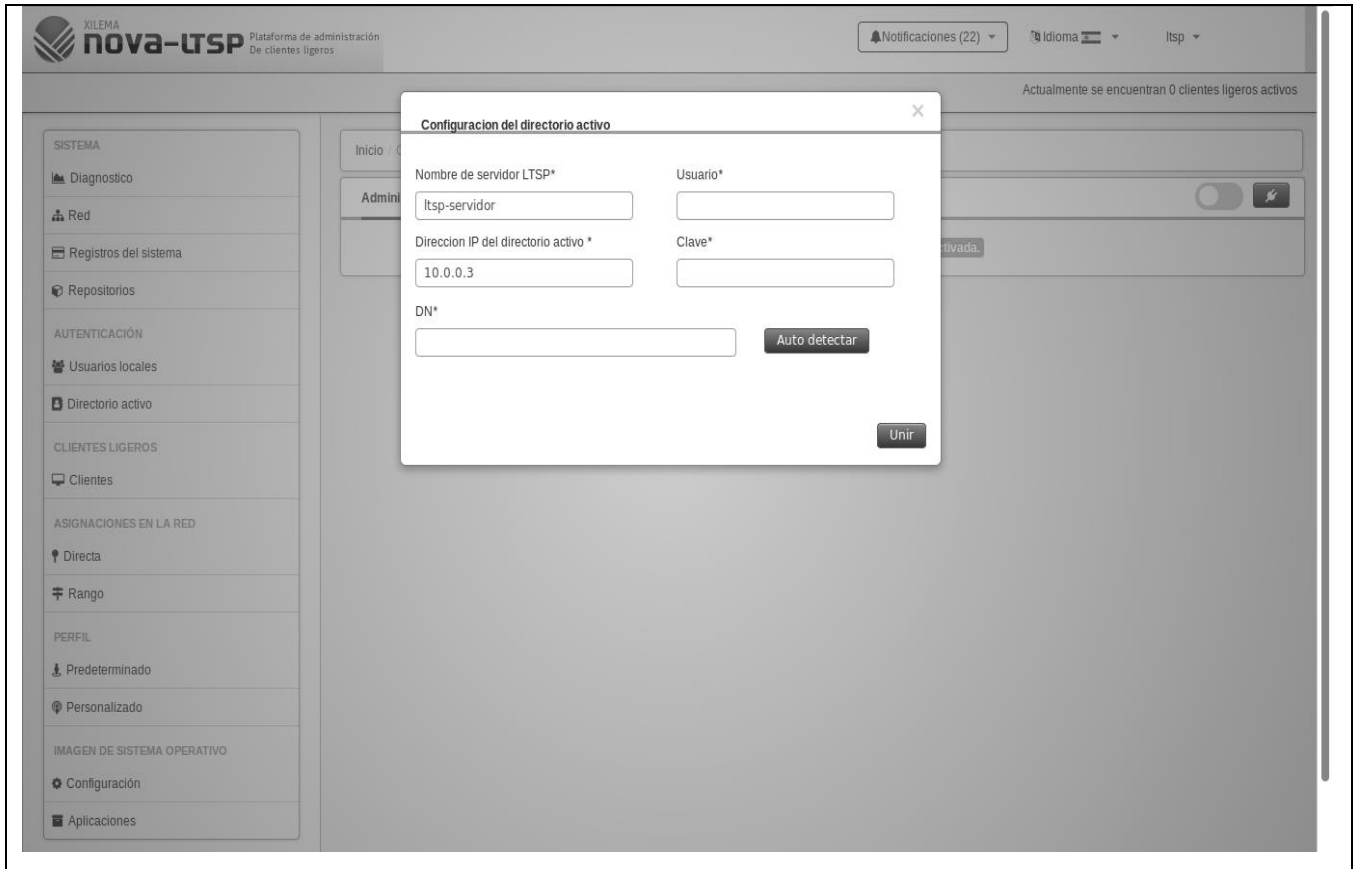


Tabla 3: Historia de usuario

Historia de Usuario	
<b>Número:</b> HU_2	<b>Nombre del requisito:</b> Administrar la autenticación de los usuarios al Directorio Activo
<b>Programador:</b> Alejandro Rodas Cueto	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 8 horas
<b>Riesgo en Desarrollo:</b> Alto	<b>Tiempo Real:</b> 8 horas

**Descripción:** La HU inicia cuando el usuario autenticado en la plataforma Nova-LTSP con privilegios de administración accede al menú Directorio activo, ubicado en el panel izquierdo de la plataforma y selecciona un usuario, esto a la vez brinda la opción de administrar la autenticación del mismo que posibilita activar o desactivar la autenticación mediante un cliente ligero a los usuarios al Directorio Activo.

**Observaciones:** El usuario debe estar registrado en el Directorio Activo y añadido en el servidor LTSP.

### Prototipo de la interfaz gráfica del usuario

	Nombre	Usuario
<input checked="" type="checkbox"/>	ysanchezl	Yasel Sánchez López
<input checked="" type="checkbox"/>	ymzayas	Yaima Mariño Zayas
<input checked="" type="checkbox"/>	amelia	Daisy Milián Zayas
<input type="checkbox"/>	cborges	Cecilia Caridad Borges Zayas
<input checked="" type="checkbox"/>	yasserc	Yasser Cancio Amaro
<input type="checkbox"/>	yasleny	Yasleni González Carvajal
<input type="checkbox"/>	sury	Surama Borges Zayas
<input type="checkbox"/>	yasserjm	Yasser Jiménez Martínez
<input type="checkbox"/>	yamonteagudo	Yasmany Alfonso Monteagudo
<input type="checkbox"/>	yasmanyrm	Yasmany Rodríguez Morejon

Showing 1 to 10 of 85 entries (filtered from 9,688 total entries)

Tabla 4: Historia de Usuario

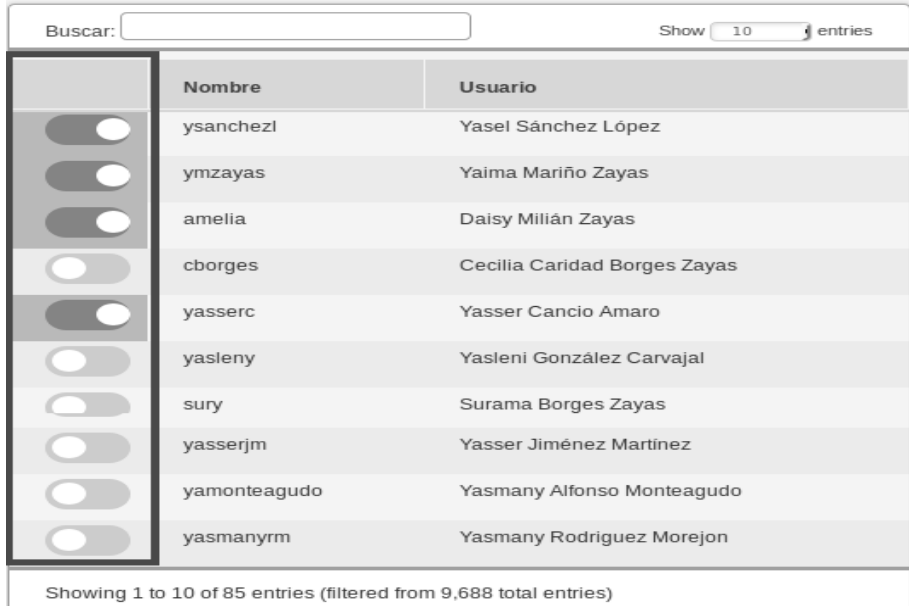
Historia de Usuario	
<b>Número:</b> HU_3	<b>Nombre del requisito:</b> Mostrar el estado todos los usuarios
<b>Programador:</b> Alejandro Rodas Cueto	<b>Iteración Asignada:</b> 1
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 5 horas
<b>Riesgo en Desarrollo:</b> Alto	<b>Tiempo Real:</b> 5 horas
<p><b>Descripción:</b> La HU inicia cuando el usuario autenticado en la plataforma Nova-LTSP con privilegios de administración accede al menú Directorio activo, ubicado en el panel izquierdo de la plataforma. Muestra en la parte izquierda de la vista (Estado del Directorio Activo) si el usuario tiene activado la autenticación mediante un cliente ligero al directorio activo.</p>	
<p><b>Observaciones:</b> El usuario debe estar registrado en el Directorio Activo y añadido en el servidor LTSP.</p>	
<p><b>Prototipo elemental de interfaz gráfica del usuario</b></p>  <p>The screenshot shows a user directory interface. At the top, there is a search bar labeled 'Buscar:' and a 'Show 10 entries' dropdown. Below this is a table with two columns: 'Nombre' and 'Usuario'. Each row in the table has a toggle switch on the left. The table lists 10 users, including 'ysanchezl', 'ymzayas', 'amelia', 'cborges', 'yasserc', 'yasleny', 'sury', 'yasserjm', 'yamonteagudo', and 'yasmanyrm'. At the bottom of the table, it says 'Showing 1 to 10 of 85 entries (filtered from 9,688 total entries)'.</p>	

Tabla 5: Historia de Usuario

Historia de Usuario																																		
<b>Número:</b> HU_4	<b>Nombre del requisito:</b> Listar los usuarios del directorio activo																																	
<b>Programador:</b> Alejandro Rodas Cueto	<b>Iteración Asignada:</b> 1																																	
<b>Prioridad:</b> Alta	<b>Tiempo Estimado:</b> 5 horas																																	
<b>Riesgo en Desarrollo:</b> Alto	<b>Tiempo Real:</b> 5 horas																																	
<b>Descripción:</b> La HU inicia cuando el usuario autenticado en la plataforma Nova-LTSP con privilegios de administración accede al menú Directorio activo, ubicado en el panel izquierdo de la plataforma y en la parte derecha de la vista muestra todos los usuarios registrados en el Directorio Activo																																		
<b>Observaciones:</b> El usuario debe estar registrado en el Directorio Activo y añadido en el servidor LTSP																																		
<b>Prototipo elemental de interfaz gráfica del usuario</b>																																		
<p>The screenshot shows a user interface for listing users. At the top, there is a search bar labeled 'Buscar:' and a 'Show 10 entries' dropdown. Below this is a table with two columns: 'Nombre' and 'Usuario'. To the left of the table is a vertical column of toggle switches. The table contains 10 rows of user data. At the bottom, a footer indicates 'Showing 1 to 10 of 85 entries (filtered from 9,688 total entries)'.</p> <table border="1"> <thead> <tr> <th></th> <th>Nombre</th> <th>Usuario</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>ysanchezl</td> <td>Yasel Sánchez López</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>ymzayas</td> <td>Yaima Mariño Zayas</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>amelia</td> <td>Daisy Milián Zayas</td> </tr> <tr> <td><input type="checkbox"/></td> <td>cborges</td> <td>Cecilia Caridad Borges Zayas</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>yasserc</td> <td>Yasser Cancio Amaro</td> </tr> <tr> <td><input type="checkbox"/></td> <td>yasleny</td> <td>Yasleni González Carvajal</td> </tr> <tr> <td><input type="checkbox"/></td> <td>sury</td> <td>Surama Borges Zayas</td> </tr> <tr> <td><input type="checkbox"/></td> <td>yasserjm</td> <td>Yasser Jiménez Martínez</td> </tr> <tr> <td><input type="checkbox"/></td> <td>yamonteagudo</td> <td>Yasmany Alfonso Monteagudo</td> </tr> <tr> <td><input type="checkbox"/></td> <td>yasmanyrm</td> <td>Yasmany Rodríguez Morejon</td> </tr> </tbody> </table>			Nombre	Usuario	<input checked="" type="checkbox"/>	ysanchezl	Yasel Sánchez López	<input checked="" type="checkbox"/>	ymzayas	Yaima Mariño Zayas	<input checked="" type="checkbox"/>	amelia	Daisy Milián Zayas	<input type="checkbox"/>	cborges	Cecilia Caridad Borges Zayas	<input checked="" type="checkbox"/>	yasserc	Yasser Cancio Amaro	<input type="checkbox"/>	yasleny	Yasleni González Carvajal	<input type="checkbox"/>	sury	Surama Borges Zayas	<input type="checkbox"/>	yasserjm	Yasser Jiménez Martínez	<input type="checkbox"/>	yamonteagudo	Yasmany Alfonso Monteagudo	<input type="checkbox"/>	yasmanyrm	Yasmany Rodríguez Morejon
	Nombre	Usuario																																
<input checked="" type="checkbox"/>	ysanchezl	Yasel Sánchez López																																
<input checked="" type="checkbox"/>	ymzayas	Yaima Mariño Zayas																																
<input checked="" type="checkbox"/>	amelia	Daisy Milián Zayas																																
<input type="checkbox"/>	cborges	Cecilia Caridad Borges Zayas																																
<input checked="" type="checkbox"/>	yasserc	Yasser Cancio Amaro																																
<input type="checkbox"/>	yasleny	Yasleni González Carvajal																																
<input type="checkbox"/>	sury	Surama Borges Zayas																																
<input type="checkbox"/>	yasserjm	Yasser Jiménez Martínez																																
<input type="checkbox"/>	yamonteagudo	Yasmany Alfonso Monteagudo																																
<input type="checkbox"/>	yasmanyrm	Yasmany Rodríguez Morejon																																

## 2.6 Diseño del módulo

Es la etapa donde se analizan cada una de las especificaciones solicitadas por el cliente, además se secciona el software, viendo sus funciones, como se mostrará en pantalla entre otras más que conlleva el diseño de software (32).

A continuación, se define la arquitectura de software y los patrones de diseño que utiliza el módulo a implementar.

### 2.6.1 Arquitectura de software

Arquitectura de Software, porque, a semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software. En el libro "*An introduction to Software Architecture*", David Garlan y Mary Shaw definen que la Arquitectura es un nivel de diseño que hace énfasis en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema" (33).

La arquitectura que se utilizará para el desarrollo del módulo es la empleada por Django, el cual sigue una arquitectura Modelo-Vista-Controlador, solo que hace una adaptación de esta a Modelo-Vista-Plantilla (a partir de ahora MTV por sus siglas en inglés, *Model Template View*). Por tanto, el sistema propuesto hereda una arquitectura MTV, debido a que se desarrollará sobre el marco de trabajo Django.

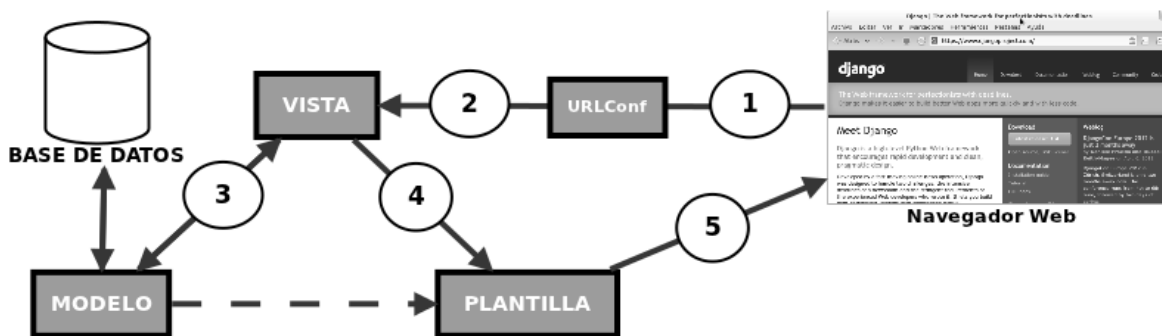


Ilustración 2: Funcionamiento del MTV de Django

**Modelo:** contiene toda la información sobre los datos. Cada una de las entidades del Directorio Activo se encuentra en el modelo en forma de clases de Python, y sus atributos se almacenan en variables con ciertos

parámetros. También estos archivos poseen métodos, lo que permite indicar y controlar el comportamiento de los datos.

**Vista:** es la capa de la lógica de negocios, contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Esta capa sirve de “puente” entre el modelo y la plantilla, se presenta en forma de funciones de Python y su función principal es determinar qué datos serán visualizados en las plantillas.

**Plantilla:** recibe los datos de la vista y luego los organiza para la presentación al navegador web. Básicamente es una página HTML (*HyperText Markup Language*) con algunas etiquetas extras que son propias del Django

La siguiente figura representa la separación de las clases de la aplicación en el marco de trabajo según el patrón arquitectónico MVT.

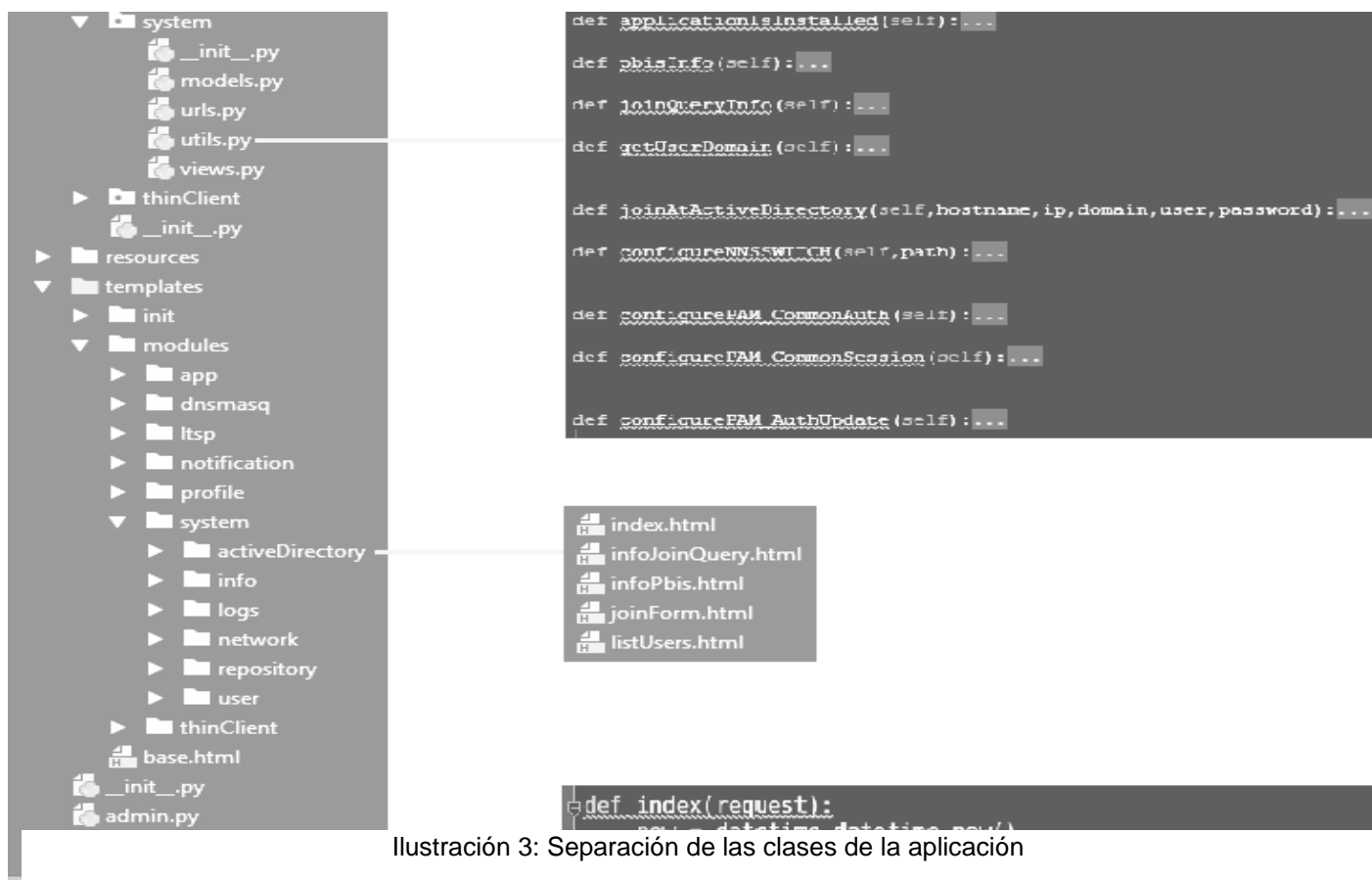


Ilustración 3: Separación de las clases de la aplicación



## 2.6.2 Patrones de diseño

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.” En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (34).

### Patrones GRASP

Los patrones GRASP (por sus siglas en inglés, *General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (35).

Los patrones GRASP que se utilizaron en la propuesta de solución fueron:

#### Experto

Este patrón tiene como objetivo principal asignar una responsabilidad determinada a la clase que tenga la mayor cantidad de información para hacer esta tarea. Es la razón anterior la que le da su apellido Experto “en información”. Este patrón se evidencia en la clase *LogEntry* y *QueryFilter* que son las que tienen acceso a todas las entidades necesarias y por ello a la información, por lo cual se le asigna la responsabilidad de generar todos los eventos y consultas que se requieren.



Ilustración 4: Patrón Experto

## Creador

Se asigna la responsabilidad a una clase de crear cuando contiene, agrega, compone, almacena o usa otra clase, lo que brinda una alta posibilidad de reutilizar la clase creadora. Este patrón se pondrá de manifiesto en la implementación de la función *Logentry\_register*.

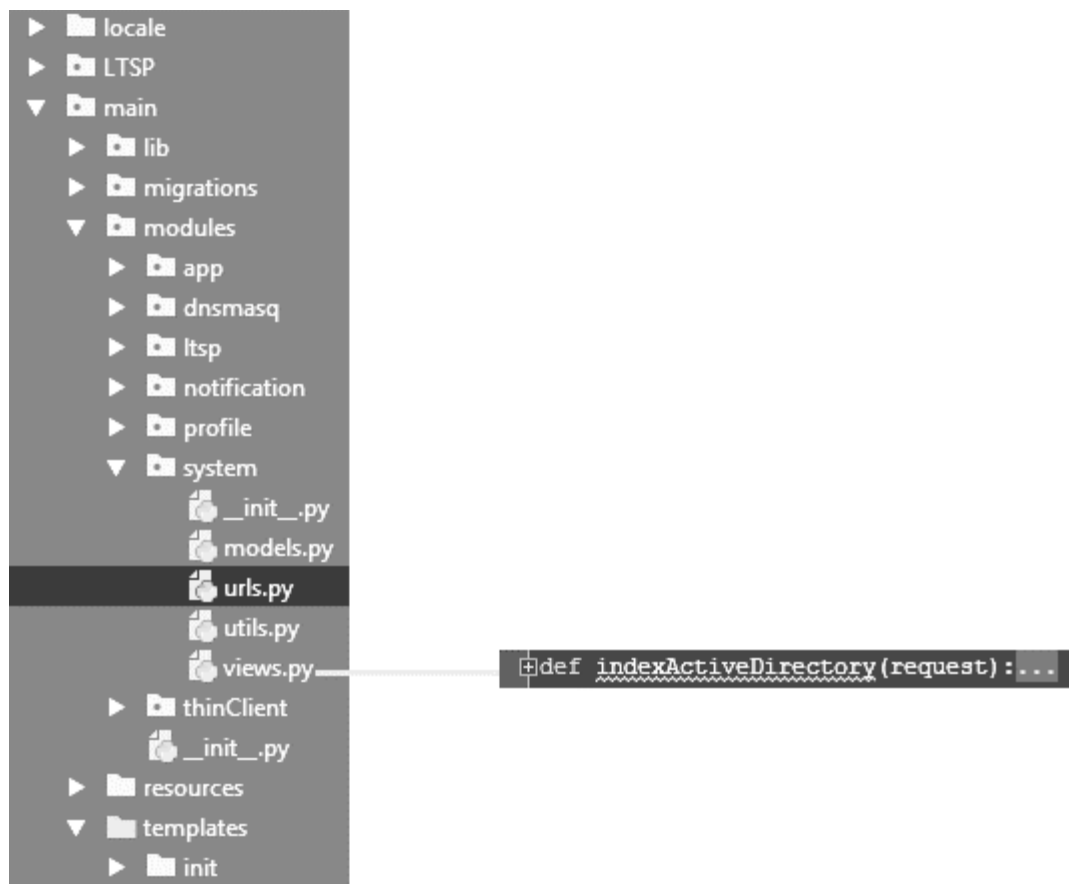


Ilustración 5: Patrón creador

## Bajo acoplamiento

Es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Este patrón ya viene incluido con Django que permite un bajo acoplamiento entre las piezas, lo que evita las dependencias, por ejemplo, a la hora de realizar cambios en las configuraciones de las URL, en la Base de Datos y las plantillas HTML, basta solo con realizarlo una sola vez.



Ilustración 6: Patrón Bajo acoplamiento

## Alta Cohesión

Asigna responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo. Este patrón incrementa la claridad y facilita la comprensión del diseño. Una de las características de Django es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, se puede observar en el sistema que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona. Esto hace posible que el sistema sea flexible a cambios sustanciales con efecto mínimo.

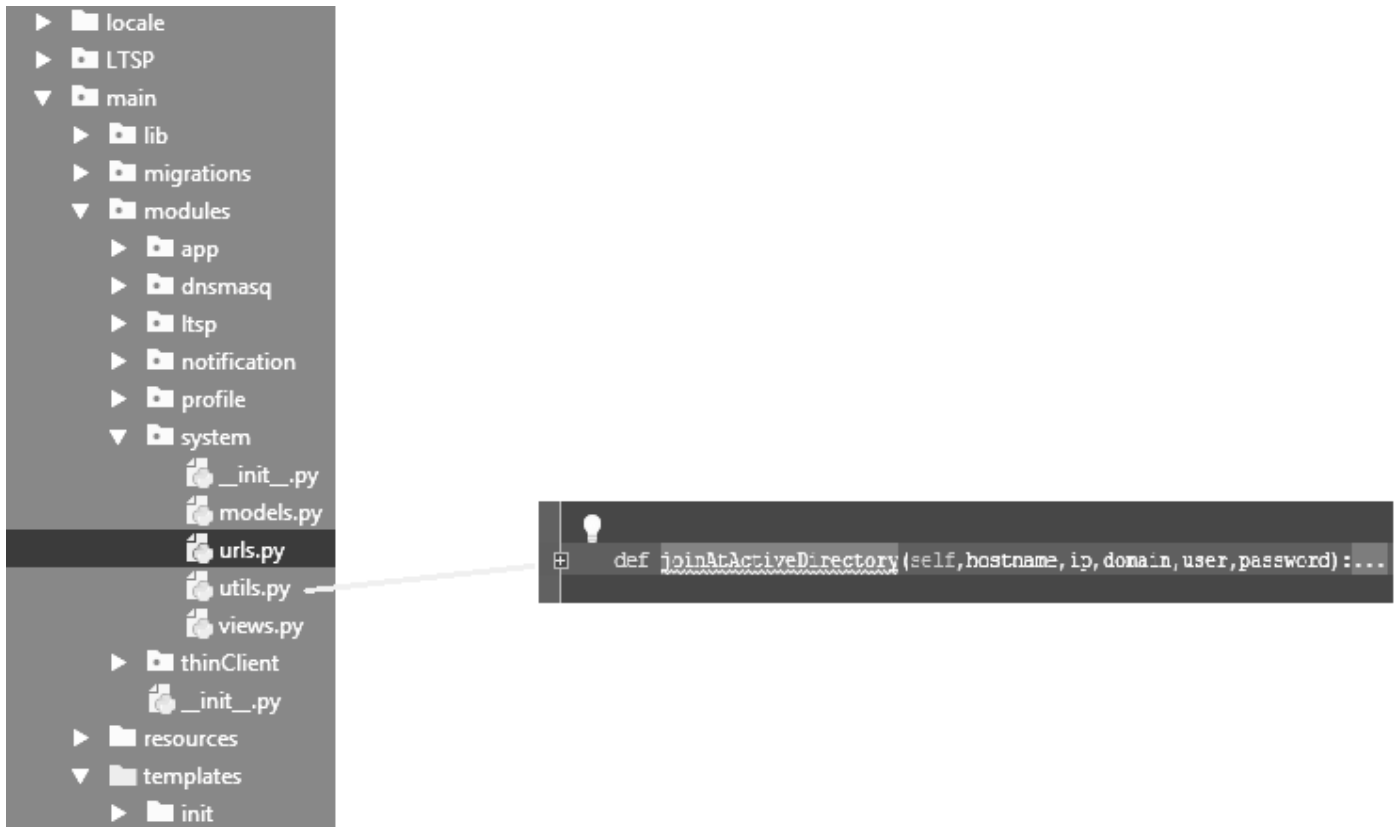


Ilustración 7: Patrón Alta cohesión

## Controlador

Asignar la responsabilidad de administrar un mensaje de evento del sistema a una clase que representa el sistema global, dispositivo, subsistema o representa un escenario de caso de uso en el que tiene lugar el evento del sistema. Este patrón es empleado en todo el sistema debido a que cada uno de los eventos generados por el usuario es redirigido a una clase o función controladora que realiza las operaciones solicitadas, manteniendo siempre la alta cohesión.

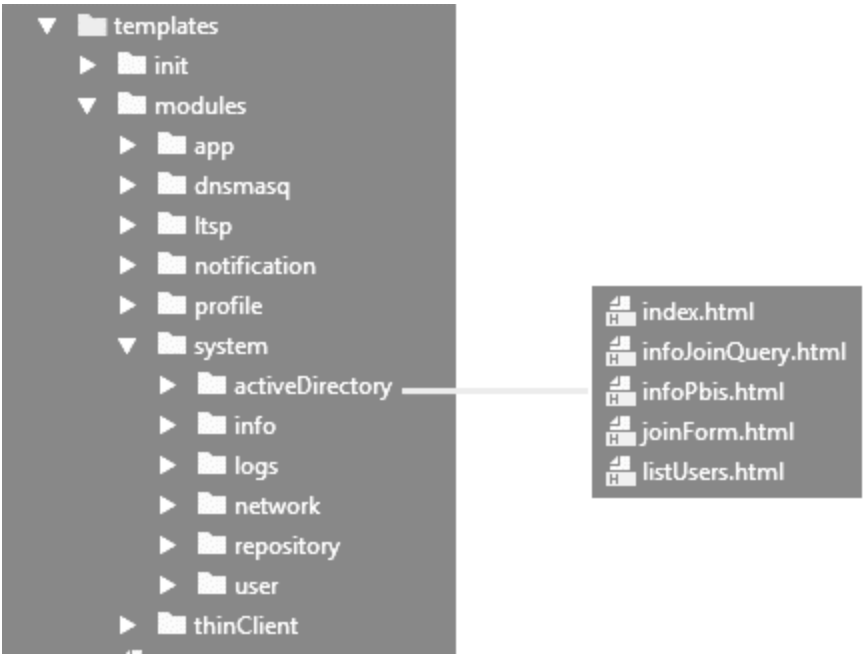


Ilustración 8: Patrón Controlador

## 2.7 Diagrama de paquetes

Un diagrama de paquetes en el Lenguaje Unificado de Modelado representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema (36).

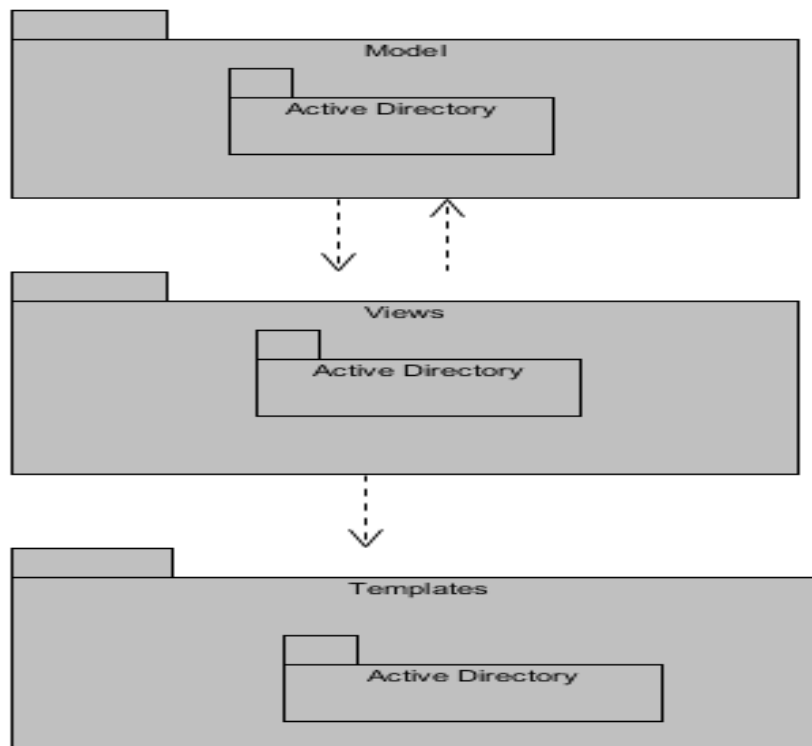


Ilustración 9: Diagrama de paquetes

### Paquete Model

**Active Directory:** contiene la información necesaria para administrar la herramienta que une el servidor LTSP y unirla a un dominio.

### Paquete Views

**Active Directory:** contiene la información necesaria para funcionar como intermediario entre el modelo y la plantilla.

### Paquete Templates

**Active Directory:** muestra toda la información suministrada por el modelo.

## 2.8 Modelo de despliegue

El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los artefactos software en nodos (usualmente plataforma de hardware). Modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (37).

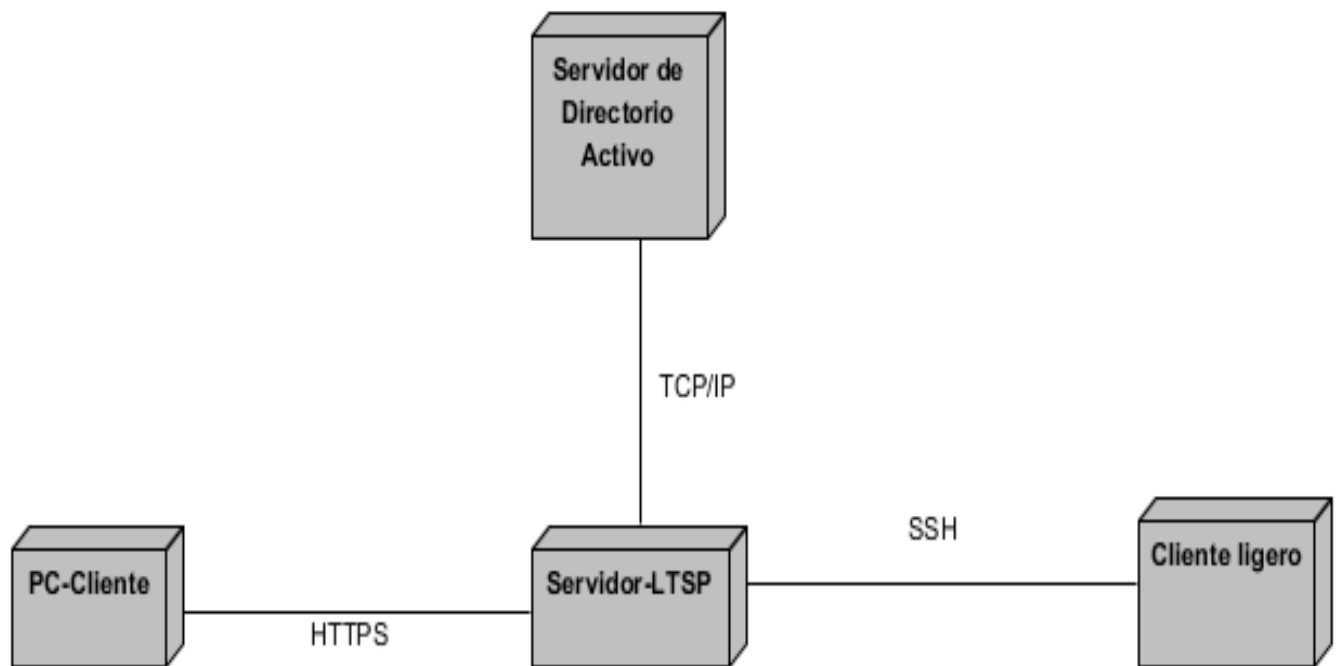


Ilustración 10: Modelo de despliegue

Las relaciones que existen entre nodos representan los protocolos de comunicación que se utilizan para acceder a cada uno. La PC Cliente representa las computadoras de los usuarios que se conectan al sistema, las cuales realizan peticiones al servidor LTSP mediante el protocolo HTTPS. Este servidor mantiene una conexión mediante el protocolo de comunicación segura SSH con los clientes ligeros y el servidor de Directorio activo y al servidor de bases de datos mediante el protocolo TCP/IP.



## 2.9 Conclusiones parciales

En este capítulo se han abordado los elementos del análisis y diseño del módulo para la autenticación con un controlador de dominio en Nova-LTSP, en el cual se arriban a la siguiente conclusión: La propuesta de solución permitió conocer las principales características y funcionalidades que debe cumplir la aplicación para lograr la aceptación por el cliente, en este proceso se identificaron 4 requisitos funcionales y 7 no funcionales. Además de que:

- ✓ Adoptar la arquitectura de software Modelo-Vista-Plantilla propuesta por el *framework* de desarrollo utilizado, permitió una propicia organización del sistema a implementar.
- ✓ Sirvieron de guía para desarrollar las distintas funcionalidades y de este modo satisfacer las necesidades detectadas todos los requisitos, tanto funcionales como no funcionales obtenidos a partir del proceso de identificación de los mismos.
- ✓ Las descripciones de las historias de usuario y la elaboración de los diagramas de clases del diseño y el diagrama de paquetes posibilitaron una mejor comprensión del funcionamiento de la propuesta de solución.

## **CAPÍTULO 3: Implementación y pruebas de la propuesta de solución**

En el presente capítulo se muestran los artefactos correspondientes a las etapas de implementación y prueba del sistema, así como los estándares de codificación que debe seguir el equipo de desarrollo para implementar el *software*. De acuerdo a la metodología que se utiliza, se especifican, de los tipos de pruebas que esta plantea, los que serán empleados para darle validez a los requisitos funcionales y garantizar el óptimo funcionamiento de la aplicación. Además, se realiza una valoración de los resultados obtenidos al aplicar las pruebas.

### **3.1 Modelo de implementación**

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (38).

Un diagrama de implementación muestra:

- ✓ Las dependencias entre las partes de código del sistema (diagramas de componentes).
- ✓ La estructura del sistema en ejecución (diagrama de despliegue).

#### **Diagrama de componentes**

Un componente es una parte física de un sistema (modulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes (38).

Respecto a los componentes

- ✓ Es implementado por una o más clases/objetos del sistema.
- ✓ Es una unidad autónoma que provee una o más interfaces.
- ✓ Las interfaces representan un contrato de servicios que el componente ofrece.

Los componentes pueden ser:

- ✓ Archivos
- ✓ Código fuente + Cabeceras
- ✓ Librerías compartidas (DLLs)
- ✓ Ejecutables
- ✓ Paquetes

Muestra como el sistema está dividido en componentes y las dependencias entre ellos.

- ✓ Proveen una vista arquitectónica de alto nivel del sistema.
- ✓ Ayuda a los desarrolladores a visualizar el camino de la implementación.
- ✓ Permite tomar decisiones respecto a las tareas de implementación y los Skills requeridos.

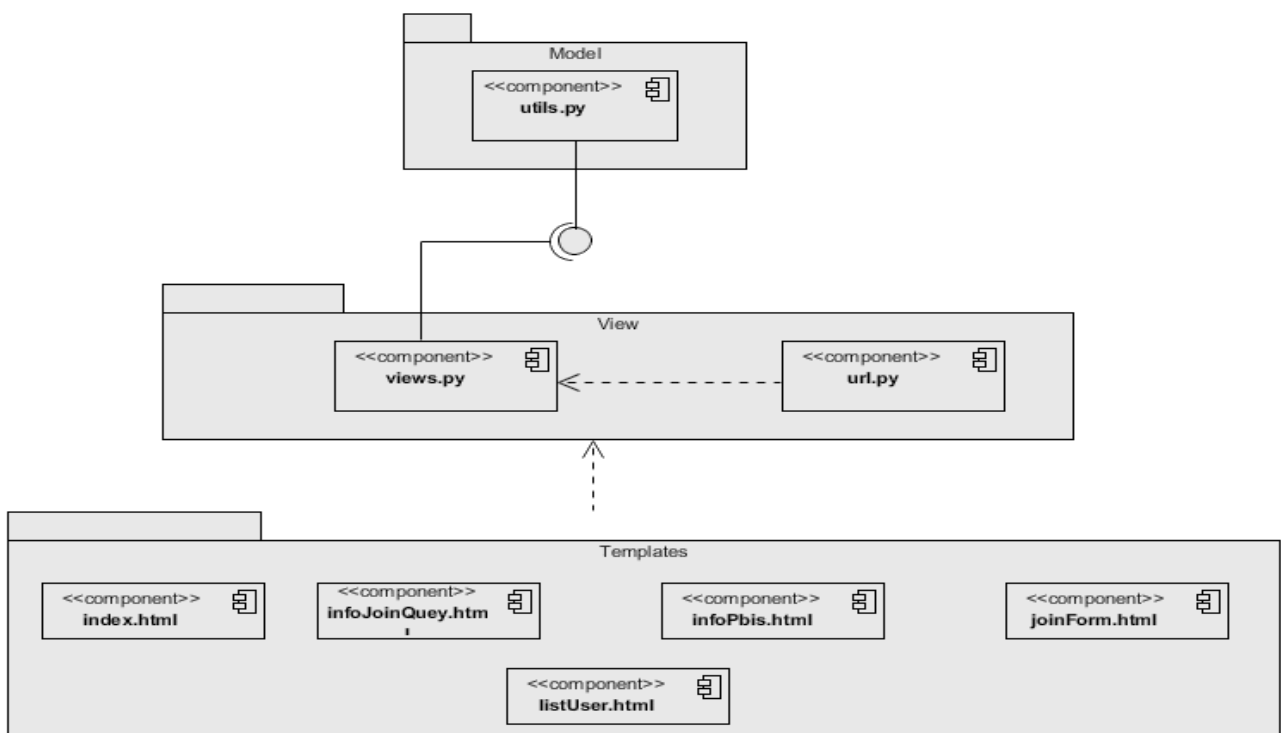


Ilustración 11: Diagrama de componentes

## Descripción de los componentes del sistema

**Plantilla:** paquete que agrupa a las plantillas encargadas de presentar la información al usuario.

**Vista:** paquete que agrupa a todos los componentes que interactúan con el paquete de clases Modelo.

**Modelo:** paquete que contiene la clase encargada de enviarle la información a la vista.

### 3.2 Estándares de codificación

Estándar de programación (también llamado Estilo de programación o convención de código) es un término que describe convenciones para escribir código fuente en ciertos lenguajes de programación. El estilo de programación es frecuentemente dependiente del lenguaje de programación que se haya elegido para escribir. La forma de escribir código es propia de cada programador y completamente diferente a la forma de cualquier otro. De la forma usada depende la facilidad para entender el código y retomar ciertas partes realizadas por otros integrantes, así como la depuración de las mismas (39).

#### Identación

- ✓ Usa cuatro espacios por cada nivel de indentación.
- ✓ Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).
- ✓ No se mezclarán tabuladores y espacios en la codificación. El método de indentación más popular en Python es con espacios. El segundo más popular con tabulaciones, sin mezclar unos con otros. Cualquier código indentado con una mezcla de espacios y tabulaciones debe ser convertido a espacios exclusivamente.

#### Máxima longitud de las líneas

- ✓ Se limitarán todas las líneas a un máximo de 79 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.
- ✓ Las importaciones siempre deben colocarse al comienzo del archivo.
- ✓ Las importaciones deben estar agrupadas de la siguiente forma:
  - Importaciones de la librería estándar.
  - Importaciones terceras relacionadas.
  - Importaciones locales de la aplicación / librerías.
- ✓ Cada grupo de importaciones debe estar separado por una línea en blanco.

## **Espacios en blanco en expresiones y sentencias**

- ✓ Evitar utilizar espacios en blanco en las siguientes situaciones:
  - Inmediatamente dentro de paréntesis, corchetes y llaves.
  - Inmediatamente antes de una coma, un punto y coma o dos puntos.
  - Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función.
  - Inmediatamente antes de un corchete que empieza una indexación.
  - Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro.
- ✓ Deben rodearse con exactamente un espacio los siguientes operadores binarios:

- Asignación (=).
  - Asignación de aumentación (+=, -=).
  - Comparación (==, <, >, >=, <=, !=, <>, in, not in, is, is not).
  - 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 y funciones.
- ✓ Si un comentario es corto el punto final puede omitirse.

### **Comentarios en bloque**

- ✓ Deben estar indentados al mismo nivel que el código a comentar.
- ✓ Cada línea de un comentario en bloque comienza con un numeral (#) y un espacio en blanco.

### **Comentarios en línea**

- ✓ Se recomienda utilizarlos escasamente.
- ✓ Se debe definir comenzando por un numeral (#) seguido de un espacio en blanco.
- ✓ Deben ubicarse en la misma línea que se desea comentar.

### **Convenciones de nombramientos**

- ✓ 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 guión bajo “\_”.
- ✓ Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guión bajo (\_).

### 3.3 Pruebas de software

Las pruebas de software (en inglés *software testing*) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o *stakeholder*<sup>8</sup>. Es una actividad más en el proceso de control de calidad<sup>9</sup>. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo (40).

#### 3.3.1 Estrategias de prueba

Una estrategia para la parte de prueba de un proyecto describe el enfoque y los objetivos generales de las tareas de prueba. Incluye las fases de prueba (unidad, integración y sistema) que se deben seguir y las clases de pruebas (función, rendimiento, carga, tensión) que se deben realizar (41).

<sup>8</sup> Una parte interesada hace referencia a una persona, organización o empresa que tiene interés u empresa en una organización dada.

<sup>9</sup> El control de calidad es el conjunto de los mecanismos, acciones y herramientas realizadas para detectar la presencia de errores.

La estrategia define:

Herramientas y técnicas de prueba que se deben utilizar.

- ✓ Qué criterios de satisfacción y terminación se utilizarán. Por ejemplo, los criterios pueden permitir que el software avance hasta la prueba de aceptación cuando se haya ejecutado satisfactoriamente el 95% de los guiones de prueba. Otro criterio es la cobertura de código. Este criterio puede ser, en un sistema de seguridad crítica, que las pruebas deben cubrir el 100% del código.
- ✓ Los requisitos de recursos se ven afectados por consideraciones especiales, o tienen implicaciones de planificación, como:
  - probar todas las interfaces en sistemas externos
  - simular daño físico o amenaza a la seguridad

### 3.3.1.1 Pruebas de unidad

Todas las modernas arquitecturas de software incorporan la modularidad como uno de sus elementos esenciales. Así tenemos la posibilidad de tener a distintos programadores trabajando en distintos módulos sin que esto signifique un problema. De la misma forma es posible imaginar a los programadores ejecutando pruebas locales sobre sus módulos, a fin de tener una comprobación del buen funcionamiento -autónomo- de lo que han estado programando. A esta forma de prueba se le llama pruebas de unidad o también prueba unitaria. Entendiendo formalmente por esto lo siguiente:

**Prueba de Unidad:** nombre que reciben los procedimientos de pruebas locales a un módulo del sistema. Por definición dichas pruebas cubren la funcionalidad propia del módulo tanto con una perspectiva de caja blanca como de caja negra; pero prestando poca o ninguna atención a la integración con otros módulos. (42).

### Prueba de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos, logrando como resultado que disminuya en un gran porcentaje el



número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que (43):

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

La prueba de unidad que se realiza hace uso del método de caja blanca y de la técnica del camino básico. El método del camino básico permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño procedural y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

A continuación, se realiza la técnica del camino básico a la funcionalidad Listar usuarios que permite mostrar en una lista todos los usuarios del Directorio Activo.

Tabla 6: Técnica de camino básico

(1)	<code>def getUserDomain(self):</code>
(2)	<code>listUser=list()</code>
(3)	<code>try:</code>
(4)	<code>output = ComandExecute("getent passwd   grep 'PBIS:' ").execute()</code>
(5)	<code>if output.code==0:</code>
(6)	<code>data = output.standarOutput.split("\n")</code>

(7)	for item in data:
(8)	element=item.split(":")
(9)	user=element[0]
(10)	id=element[2]
(11)	name=element[4]
(12)	diccio={}
(13)	diccio["id"]=id
(14)	diccio["user"]=user
(15)	diccio["name"]=name
(16)	listUser.append(diccio)
(17)	return {"code":output.code,"listuser":listUser}
(18)	else:
(19)	self.notify.add(1, "No se pudo obtener la información sobre el Directorio Activo" + str(output.error))
(20)	return {"code": output.code, "listuser": output.getError()}
(21)	except:

(22)	<code>self.notify.add(1,"No se pudo obtener la información sobre el Directorio Activo")</code>
(23)	<code>return {"code": 500, "listuser": "No se pudo obtener la información sobre el Directorio Activo"}</code>

Luego de numerar las líneas de código, se diseña la gráfica del programa que describe el flujo de control lógico empleando nodos y aristas.

### Complejidad Ciclomática

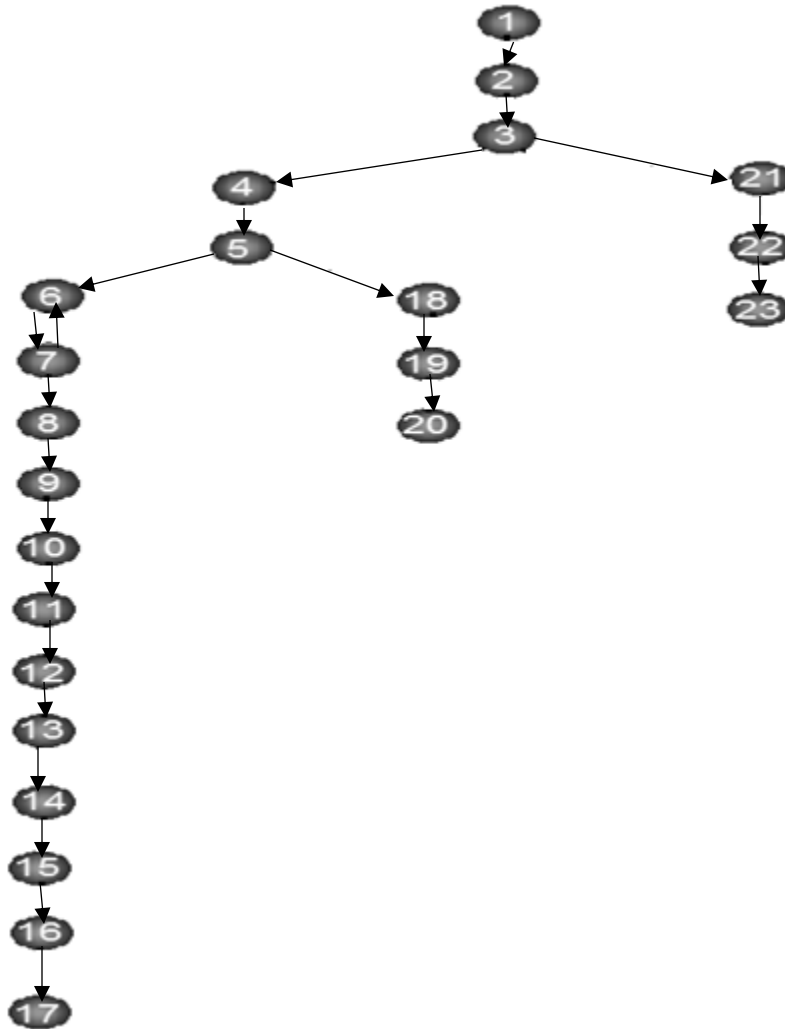


Ilustración 12: Gráfica del programa que describe el flujo de control lógico

La complejidad ciclomática, es una métrica del software en ingeniería del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Es una de las métricas de software de mayor aceptación, ya que ha sido concebida para ser independiente del lenguaje (44).

A partir del grafo obtenido con 23 nodos y 23 aristas se calcula la complejidad ciclomática  $V(G)$ , la cual constituye una métrica de software que proporciona una medida cuantitativa de la complejidad lógica del programa

$$V(G) = (\text{cantidad\_aristas} - \text{cantidad\_nodos}) + 2$$

$$V(G) = (23 - 23) + 2 = 2$$

Una ruta independiente es cualquier ruta del programa que ingrese al menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición (44). La cantidad de rutas independientes son establecidas por la complejidad Ciclométrica, por tanto, se identifican tres rutas como se muestra en la siguiente tabla:

Tabla 7: Tabla de rutas

No Ruta	Caminos
1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
2	1,2,3,4,5,18,19,20
3	1,2,3,21,22,23

El valor de  $V(G)$  ofrece además un límite superior del número de pruebas que debe diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones (44). Por tanto, se diseñan casos de pruebas para ser aplicados a cada ruta independiente.

Tabla 8: Prueba de unidad #1

Caso de Prueba de Unidad	
No. ruta: 1	Ruta: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
Nombre de la persona que realiza la prueba: Alejandro Rodas Cueto	
Descripción de la prueba: Listar los usuarios	
Entrada: Se ejecuta el comando <i>getent passwd   grep 'PBIS:'</i>	
Resultado esperado: El sistema muestra todos los usuarios	
Evaluación de la Prueba: Satisfactoria. Se obtuvo el resultado esperado.	

Tabla 9: Prueba de unidad #2

Caso de Prueba de Unidad	
No. ruta: 2	Ruta: 1,2,3,4,5,18,19,20
Nombre de la persona que realiza la prueba: Alejandro Rodas Cueto	
Descripción de la prueba: Listar los usuarios	
Entrada: Se ejecuta el comando <i>getent passwd   grep 'PBIS:'</i>	
Resultado esperado: El sistema envía una notificación	
Evaluación de la Prueba: Satisfactoria. Se obtuvo el mensaje esperado.	

Tabla 10: Prueba de unidad #3

Caso de Prueba de Unidad	
No. ruta: 3	Ruta: 1,2,3,4,5,6,9,10,11,12,13,14,15,16,17,18,19,20
Nombre de la persona que realiza la prueba: Alejandro Rodas Cueto	

<b>Descripción de la prueba:</b> Listar los usuarios.
<b>Entrada:</b> Se ejecuta el comando <code>getent passwd   grep 'PBIS:'</code>
<b>Resultado esperado:</b> El sistema envía una notificación
<b>Evaluación de la Prueba:</b> Satisfactoria. Se obtuvo el mensaje esperado.

### Resultados de las pruebas de unidad

Se realizaron 3 iteraciones de la prueba unitaria. En la primera iteración se detectaron 4 no conformidades; en la segunda, 1 no conformidad la cual fue resuelta para la tercera iteración. Las no conformidades detectadas estaban asociadas a errores de validación.

#### 3.3.1.2 Pruebas de integración

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funciona junto. Las pruebas de integración (algunas veces llamadas integración y testeo) es la fase de la prueba de software en la cual módulos individuales de software son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden a las pruebas del sistema (44).

### Resultados de las pruebas de integración

Las pruebas de integración realizadas permitieron identificar una URL duplicada en diferentes módulos de la plataforma Nova-LTSP. Luego de solucionar esta no conformidad, el módulo para la autenticación con el directorio activo de Nova-LTSP se integró correctamente con dicha plataforma.

#### 3.3.1.3 Pruebas de validación

Las pruebas de validación en la ingeniería de software son el proceso de revisión que verifica que el sistema de software producido que cumple con las especificaciones y que logra su cometido. Es normalmente una parte del proceso de pruebas de software de un proyecto, que también utiliza técnicas tales como

evaluaciones, inspecciones y tutoriales. La validación es el proceso de comprobar que lo que se ha especificado es lo que el usuario realmente quería (44).

A continuación, se presentan los casos de prueba correspondientes a las historias de usuario administrar las configuraciones de PBIS para unir el servidor LTSP al dominio del Directorio Activo, administrar la autenticación de los usuarios al Directorio Activo, devolver el estado de un usuario y listar los usuarios del directorio activo a partir de los cuales el cliente ejecutó las pruebas:

Tabla 11: Caso de Prueba de aceptación #1

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> Nova-LTSP-1-1	<b>Nombre Historia de Usuario:</b> Administrar las configuraciones de PBIS para unir el servidor LTSP al dominio del Directorio Activo
<b>Nombre de la persona que realiza la prueba:</b> Alejandro Rodas Cueto	
<b>Descripción de la prueba:</b> El sistema permitirá la unión de una imagen de cliente ligero a un dominio.	
<b>Condiciones de Ejecución:</b> El sistema debe instalar la Herramienta Pbis, configurar 3 ficheros de configuración, reiniciar el servicio de Pbis, configurar el login y el host del usuario y configurar las PAM.	
<b>Entrada/Pasos de Ejecución:</b>	
<ol style="list-style-type: none"> <li>1. Entrar al módulo Active Directory</li> <li>2. Instalar Pbis</li> <li>3. Configurar el nombre del servidor</li> <li>4. Editar los 3 ficheros de configuración</li> <li>5. Llenar campos</li> <li>6. Unir al dominio</li> </ol>	

<p>7. Reiniciar el servicio Pbis</p> <p>8. Configura el login y el home del usuario</p> <p>9. Se realizan las 3 configuraciones de las PAM (<i>common-session, common-auth y pbis.pam-auth-update</i>)</p>
<p><b>Resultado esperado:</b> El sistema une satisfactoriamente la imagen de cliente ligero al dominio y muestra una notificación exitosa.</p>
<p><b>Evaluación de la Prueba:</b> Satisfactoria.</p>

Tabla 12: Caso de Prueba de aceptación #2

Caso de Prueba de Aceptación	
<p><b>Código Caso de Prueba:</b> Nova-LTSP-2-2</p>	<p><b>Nombre Historia de Usuario:</b> Administrar la autenticación de los usuarios al Directorio Activo.</p>
<p><b>Nombre de la persona que realiza la prueba:</b> Alejandro Rodas Cueto</p>	
<p><b>Descripción de la prueba:</b> El sistema permitirá activar o desactivar la autenticación mediante un cliente ligero a los usuarios al directorio activo.</p>	
<p><b>Condiciones de Ejecución:</b> Cliente ligero debe estar creado</p>	
<p><b>Entrada/Pasos de Ejecución:</b></p> <ol style="list-style-type: none"> <li>1. Entrar al módulo <i>Active Directory</i></li> <li>2. Activará o desactivará su autenticación</li> </ol>	
<p><b>Resultado esperado:</b> El sistema activa o desactiva la autenticación y muestra una notificación exitosa.</p>	
<p><b>Evaluación de la Prueba:</b> Satisfactoria.</p>	



Tabla 13: Caso de prueba de aceptación #3

Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> Nova-LTSP-3-3	<b>Nombre Historia de Usuario:</b> Devolver el estado de un usuario
<b>Nombre de la persona que realiza la prueba:</b> Alejandro Rodas Cueto	
<b>Descripción de la prueba:</b> Permite conocer si el usuario tiene activado la autenticación mediante un cliente ligero al directorio activo	
<b>Condiciones de Ejecución:</b> El usuario debe estar autenticado en la plataforma Nova-LTSP.	
<b>Entrada/Pasos de Ejecución:</b> <ol style="list-style-type: none"> <li>1. Entrar al módulo <i>Active Directory</i></li> <li>2. Mostrar el estado del usuario</li> </ol>	
<b>Resultado esperado:</b> El sistema muestra el estado del usuario.	
<b>Evaluación de la Prueba:</b> Satisfactoria.	

Tabla 14: Caso de Prueba de aceptación #4

Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> Nova-LTSP-4-4	<b>Nombre Historia de Usuario:</b> Listar los usuarios del directorio activo
<b>Nombre de la persona que realiza la prueba:</b> Alejandro Rodas Cueto	
<b>Descripción de la prueba:</b> El sistema permitirá listar todos los usuarios registrados en el Directorio Activo.	

<b>Condiciones de Ejecución:</b> Los usuario deben estar registrado en el Directorio Activo.
<b>Entrada/Pasos de Ejecución:</b> <ol style="list-style-type: none"> <li>1. Entrar al módulo <i>Active Directory</i>.</li> <li>2. Mostrar los usuarios registrados en el Directorio Activo.</li> </ol>
<b>Resultado esperado:</b> El sistema muestra los usuarios del Directorio Activo.
<b>Evaluación de la Prueba:</b> Satisfactoria.

### Resultados de las pruebas de validación

Las pruebas de validación se ejecutaron tras concluir cada una de las tres iteraciones de implementación, de manera general en la primera iteración se detectaron 13 no conformidades, agrupadas en no conformidades de código, interfaz y ortografía, de ellas fueron solucionadas 9. En la segunda se detectaron 8 no conformidades incluyendo las 4 no resueltas en la iteración anterior, aquí se resolvieron 7 de ellas. Para una tercera iteración, solo se encontró la no conformidad que quedó pendiente en la iteración anterior la misma se solucionó satisfactoriamente y en una cuarta iteración no se encontraron no conformidades.

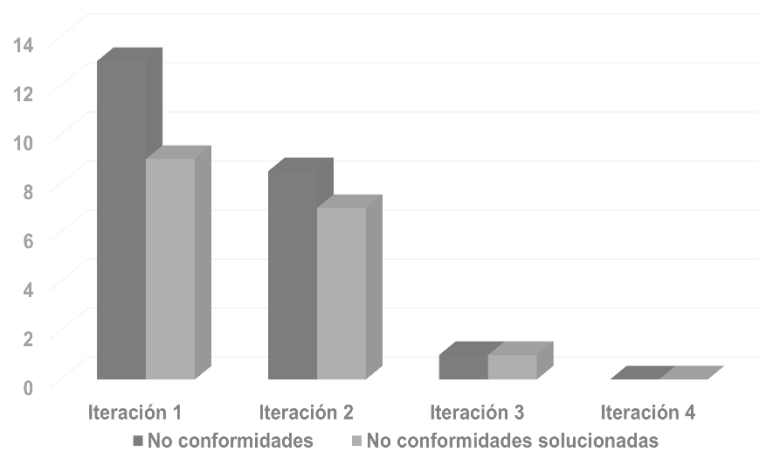


Ilustración 13: Resultado pruebas de validación

#### 3.3.1.4 Pruebas del sistema

Las pruebas de sistema engloban una serie de pruebas cuyo propósito fundamental es ejercitar el sistema de cómputo que buscan probar al sistema como un todo. Están basadas en los requerimientos generales y abarca todas las partes combinadas del sistema (44). El tipo de prueba de sistema aplicada a la solución fue la prueba de rendimiento.

Las pruebas de rendimiento (también conocidas como pruebas de carga y estrés) son las pruebas que se realizan, desde una perspectiva, para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad <sup>10</sup>, fiabilidad <sup>11</sup> y uso de los recursos (45).

Para la realización de las pruebas de carga y estrés se utilizó la herramienta Apache JMeter en su versión 2.11. JMeter es un proyecto de Apache que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. Además, es una aplicación de código abierto. JMeter puede ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, Servicios web, JMS, HTTP y conexiones TCP genéricas. JMeter puede también ser configurado como un monitor, aunque es comúnmente considerado una solución ad-hoc respecto de soluciones avanzadas de monitoreo (46).

Las pruebas se realizaron desde una computadora con 2GB de RAM, microprocesador Intel Core i3 con 2.30 GHz y sistema operativo Nova 6.0. A continuación, se describen las variables que miden el resultado de las pruebas de carga y estrés realizadas al sistema.

**Muestra:** cantidad de peticiones realizadas para cada URL.

<sup>10</sup> Es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para adaptarse y reaccionar sin perder calidad, o bien de manejar el crecimiento continuo de trabajo de manera fluida o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

<sup>11</sup> El término fiabilidad es descrita en el diccionario de la Real Academia Española (RAE) como "probabilidad de buen funcionamiento de algo".

**Media:** tiempo promedio en milisegundos en el que se obtienen los resultados.

**Mediana:** tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.

**Min:** tiempo mínimo que demora un hilo en acceder a una página.

**Max:** tiempo máximo que demora un hilo en acceder a una página.

**Línea 90%:** máximo tiempo utilizado por el 90% de la muestra, al resto de la misma le llevó más tiempo.

**% Error:** porcentaje de error de las páginas que no se llegaron a cargar de manera satisfactoria.

**Rendimiento (Rend):** el rendimiento se mide en cantidad de solicitudes por segundo.

**KB/s:** el rendimiento se mide en cantidad de kilobytes<sup>12</sup> 10 por segundo.

Debido a que el módulo está integrado a una plataforma de administración no es necesario establecer una cantidad elevada de usuarios conectados concurrentemente al sistema, no obstante, se tomó como referencia un total de 100, 500 y 1000 usuarios con la finalidad de medir el rendimiento de la aplicación y asegurar que esta pueda procesar la carga esperada.

Se simularon las peticiones realizadas al sistema por un total de 100, 500 y 1000 usuarios simultáneamente, obteniéndose los resultados que se muestran en la tabla a continuación:

Tabla 15: Resultado de pruebas del sistema

Usuarios	Muestra	Media	Mediana	Línea 90%	Min.	Max.	%Error	Rend.	KB/s
100	8250	1260	1397	1724	1	2666	0	75.7	655.3
500	42000	4170	2113	3110	5	321789	0.48	73.3	663.7
1000	68399	9672	2260	5142	2	506901	2.09	44.2	346.7

<sup>12</sup> Kilobytes: Unidad de almacenamiento de información equivalente a 1024 bytes.

Las pruebas realizadas muestran que el sistema es capaz de responder a 8250 peticiones de 100 usuarios conectados simultáneamente en un tiempo promedio de 1260 milisegundos (1.3 segundos aproximadamente) con 0% de error. Esto evidencia que el sistema puede procesar la carga esperada, cumpliéndose de este modo el requisito no funcional 5.

Por otra parte, se realizaron 42000 peticiones iniciadas por 500 usuarios y en este caso el sistema respondió en 4170 milisegundos (4.2 segundos aproximadamente) como tiempo promedio. Esto demuestra que el sistema responde en el tiempo esperado a un conjunto de peticiones 8.4 veces mayor que el propuesto en el requisito no funcional 5, aunque no fue capaz de responder correctamente el 0.48% de las peticiones realizadas.

Por último, y con el objetivo de analizar el comportamiento del sistema en condiciones extremas, se realizó una prueba de estrés para un conjunto de 1000 usuarios conectados simultáneamente. En este caso, el sistema no responde adecuadamente, siendo capaz de responder solamente a 68399 peticiones de las 82500 esperadas, en aproximadamente el doble del tiempo propuesto en el requisito no funcional 5.

### **3.4 Valoración económica, novedad y aporte social**

Los lineamientos 131, 152, 223, 226 y 228 de la política económica y social del Partido y la Revolución favorecen la integración de la ciencia, la tecnología, y la innovación en función del desarrollo social y económico del país. En consecuencia, el vínculo universidad-empresa juega un papel fundamental, poniendo al servicio de las necesidades empresariales, a los profesionales y estudiantes de las universidades.

El módulo desarrollado en la presente investigación forma parte del convenio entre la UCI y la Empresa Industrial para la Informática, las Comunicaciones y la Electrónica (GEDEME) entidad que ensambla y comercializa clientes ligeros con la Distribución cubana GNU/LINUX Nova desarrollada en la universidad. El sistema le permitirá a un administrador con mínimos esfuerzos, administrar las configuraciones de PBIS para unir el servidor LTSP al dominio del Directorio Activo, administrar la autenticación de los usuarios al Directorio Activo, devolver el estado de un usuario y listar los usuarios del directorio activo. Además, sirve de apoyo para aquellas empresas que no posean un capital muy amplio para sus inversiones y deseen informatizar algunos de sus procesos lo puedan hacer mediante la utilización de clientes ligeros,

computadoras con un *hardware* generalmente barato.

### **3.5 Conclusiones parciales**

En este capítulo se han abordado los elementos de la implementación del módulo de autenticación con el directorio activo de Nova-LTSP, así como las pruebas realizadas al mismo y los resultados obtenidos; arribando a las siguientes conclusiones:

- ✓ Mediante la elaboración de los diagramas de componentes, se brinda una mejor comprensión de la estructura de los componentes del sistema implementado.
- ✓ El uso correcto de los estándares de codificación permitió que el código del sistema desarrollado fuera legible para lograr una fácil y mejor comprensión del mismo, la cual es de utilidad para el mantenimiento del sistema.
- ✓ La aplicación de las pruebas de unidad, integración, validación y rendimiento a la solución desarrollada permitió detectar errores que aquejaban el funcionamiento del sistema, lo que brindo la posibilidad erradicarlos para que el mismo cumpliera de manera satisfactoria con los requisitos funcionales definidos en la etapa de análisis.

## **CONCLUSIONES GENERALES**

A partir de la investigación realizada y de los resultados obtenidos referentes al desarrollo del módulo de autenticación con el directorio activo de Nova-LTSP, se concluye que:

- ✓ Las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación, permitieron una mayor comprensión de la propuesta de solución.
- ✓ Las pruebas diseñadas y ejecutadas arrojaron como resultado que el sistema implementado responde a los requerimientos definidos por el cliente.
- ✓ Como resultado de la implementación se obtuvo el módulo para la autenticación con el directorio activo de Nova-LTSP que cumple con los 4 requerimientos funcionales identificados en la fase de ejecución.
- ✓ Las pruebas diseñadas y ejecutadas validaron el correcto funcionamiento del sistema, así como el cumplimiento de todos los requisitos definidos.

## **RECOMENDACIONES**

Una vez concluida la investigación y el desarrollo de la propuesta de solución, se recomienda:

- ✓ Agregar una funcionalidad que permita gestionar los usuarios del Directorio Activo.



## REFERENCIAS BIBLIOGRÁFICAS

1. **Alcantud Marín, Francisco.** «Introducción a la telemática. Redes y servicios telemáticos.». En Universitat de València. Teleformación. Diseño para todos. p. 199. ISBN 9788437033310. [En línea] <https://books.google.es/books?id=iikUYGEGKTUC&pg=PA199&lpg=PA199#v=onepage&q&f=false>.
2. **García Chico, Joanna.** *Estudio de viabilidad de Directorio Activo en Linux.* Madrid : [https://orff.uc3m.es/bitstream/handle/10016/11906/PFC\\_Joanna\\_Garcia\\_Chico.pdf;jsessionid=AAA4EA6313DB335423E655FF46359C3D?sequence=1](https://orff.uc3m.es/bitstream/handle/10016/11906/PFC_Joanna_Garcia_Chico.pdf;jsessionid=AAA4EA6313DB335423E655FF46359C3D?sequence=1), 2011.
3. **OSRI.** Ministerio de las comunicaciones y la informática. [En línea] 11 de enero de 2007. [Citado el: 12 de enero de 2018.] <http://www.poljgrave.sld.cu/download/R%20127-07.pdf>.
4. **JESÚS ANTONIO ALVAREZ CEDILLO, MICHAEL KLAUSS LINDIG , HIND TAUD, PATRICIA PEREZ ROMERO, ELIZABETH ACOSTA GONZAGA.** PROYECTO DE INVESTIGACIÓN: Clientes Ligeros IPN. [En línea] [sappi.ipn.mx/cgpi/archivos\\_anexo/20080245\\_6371.pdf](http://sappi.ipn.mx/cgpi/archivos_anexo/20080245_6371.pdf). 20080245.
5. **TOALOMBO Ortiz, Vicente Agustín.** *Estudio comparativo de la tecnologías de clientes ligeros LTSP, TCos, Microsoft Terminal Server orientado a reutilización de PC' S.* 2012.
6. **Technet Magazine.** [En línea] Microsoft, diciembre de 2008. [Citado el: 15 de enero de 2018.] <https://technet.microsoft.com/es-es/library/2008.12.linux.aspx>.
7. **McQuillan, James A.** LTSP - Linux Terminal Server Project. [En línea] 2006. [Citado el: 15 de enero de 2018.] <http://ltsp.mirrors.tds.net/pub/ltsp/docs/ltsp-4.1-en.html#AEN67>.
8. **Technet Magazine.** [En línea] Microsoft, diciembre de 2008. [Citado el: 15 de enero de 2018.] <https://technet.microsoft.com/es-es/library/2008.12.linux.aspx>.
9. **OpenLDAP.** [En línea] OpenLDAPFoundation, 2011. <http://www.openldap.org/doc/admin24/backends.html#SQL>.

10. Kerberos: The Network Authentication Protocol. [En línea] 3 de mayo de 2011. [Citado el: 16 de enero de 2018.] <http://web.mit.edu/kerberos/>.
11. Boulton, Richard. BeyondTrust. [En línea] Powerbroker Open Project, 4 de octubre de 2016. [Citado el: 16 de enero de 2018.] <https://github.com/BeyondTrust/pbis-open/wiki>.
12. Manual de referencia. *Capítulo 21. Winbind: Uso de cuentas de Dominio*. [En línea] 27 de Junio de 2002 . [Citado el: 16 de enero de 2018.] <http://www.bdat.net/documentos/samba/html/winbind.html>.
13. sssd(8) - Linux man page. *sssd - System Security Services Daemon* . [En línea] febrero de 2011. [Citado el: 17 de enero de 2018.] <https://linux.die.net/man/8/sssd>.
14. Universidad de las Ciencias Informáticas. *Metodología de desarrollo para la Actividad productiva de la UCI*. La Habana : Ediciones Futuro Cuba, 2015.
15. RUMBAUGH, James, JACOBSON, Ivar and BOOCH, Grady. *El lenguaje Unificado de Modelado. Manual de Referencia*. Madrid : Madrid :. Addison Wesley,. c2000. xxii, 526 p. ; 25 cm +. Edición ; 1a. ed, 2000. 8478290370.
16. James Rumbaugh, Ivar Jaboson y Grady Booch. *El lenguaje unificado del modelado. Manual de referencia*. Madrid : Madrid :. Addison Wesley,. c2000. xxii, 526 p. ; 25 cm +. Edición ; 1a. ed , 2000. 8478290370.
17. Pressman, Roger S. *Ingeniería del software: Un enfoque práctico, 7ma Edición*. Mexico : Eddison Wesley, 5005.
18. —. *Ingeniería del software: Un enfoque práctico, 7ma Edición*. Mexico : Eddison Wesley, 2005.
19. Pressman, Roger S. *Ingeniería de Software, un enfoque práctico. Quinta edición*. S.I. Mexico : Eddison Wesley, 2005.
20. Le Parisien. *Sensaget*. [En línea] Española, 20 de abril de 2005. [Citado el: 19 de enero de 2018.]

[http://dictionnaire.sensagent.leparisien.fr/Lenguaje\\_de\\_programaci%C3%B3n/es-es/](http://dictionnaire.sensagent.leparisien.fr/Lenguaje_de_programaci%C3%B3n/es-es/).

21. Le Parisien. [En línea] Española, 20 de abril de 2005. [Citado el: 19 de enero de 2018.]

<http://dictionnaire.sensagent.leparisien.fr/python/es-es/>.

22. Python Software Foundation. *Python*. [En línea] 3 de julio de 2010. [Citado el: 19 de enero de 2018.] <https://www.python.org/download/releases/2.7/>.

23. All Python tools. *Python*. [En línea] 2010. [Citado el: 19 de enero de 2018.]

<https://www.jetbrains.com/pycharm/>.

24. Gross, Dr. Thomas R. y Schmidt,, Dr. Douglas C. *Framework Design A Role Modeling Approach*. Universität Hamburg : s.n., 2000. Diss. ETH No. 13509.

25. The web framework for perfectionists with deadlines. *Django*. [En línea] 2005. [Citado el: 19 de enero de 2018.] <https://www.djangoproject.com/>.

26. Holovaty, Adrian y Kaplan-Moss, Jacob. *The Definitive Guide to Django Web Development Done Right*. New York : Springer-Verlag, 2009.

27. OTTO, Mark and THORNTON, Jacob. Bootstrap 3, el manual oficial. *LibrosWeb.es*. [En línea] 4 de marzo de 2006. [Citado el: 20 de enero de 2018.] [http://librosweb.es/libro/bootstrap\\_3/](http://librosweb.es/libro/bootstrap_3/).

28. Manual de jQuery. [En línea] 10 de septiembre de 2000. [Citado el: 20 de enero de 2018.]

<http://www.desarrolloweb.com/manuales/manual-jquery.html>.

29. Investigación IT. [En línea] 15 de enero de 2002. [Citado el: 20 de enero de 2018.]

<http://investigacionit.com.ar/requisitos-o-requerimientos/>.

30. SoftQaNetwork. [En línea] 17 de abril de 2003. [Citado el: 20 de enero de 2018.]

<http://www.softqanetwork.com/requisitos-no-funcionales-nfr>.

31. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 10 de

febrero de 2004. [Citado el: 3 de febrero de 2018.] [http://www.cyta.com.ar/ta0502/b\\_v5n2a1.htm](http://www.cyta.com.ar/ta0502/b_v5n2a1.htm).

32. OKhostin. [En línea] 6 de febrero de 2015. [Citado el: 10 de febrero de 2018.]

<https://okhosting.com/blog/que-es-el-diseno-de-software/>.

33. Kruchten, Philippe. *Architectural Blueprints--The 4+1 View Model of Software Architecture*. University of British-Colombia : s.n., 2014.

34. Developer Network. [En línea] 20 de mayo de 2003. [Citado el: 15 de febrero de 2018.]

<https://msdn.microsoft.com/es-es/library/bb972240.aspx>.

35. LARMAN, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. México : Prentice Hall : s.n. ISBN 970-17-0261-1.

36. *OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2*. New York : s.n., 2007. p. 149.

37. [En línea] 5 de marzo de 2007. [Citado el: 21 de febrero de 2018.]

[http://www.sparxsystems.com.ar/resources/tutorial/uml2\\_deploymentdiagram.html](http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html).

38. LARMAN, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. México : Prentice Hall : s.n., 1999. ISBN 970-17-0261-1..

39. Python Software Foundation. [En línea] 10 de junio de 2006. [Citado el: 25 de febrero de 2018.]

<https://www.python.org/dev/peps/pep-0008/>.

40. Kaner, Cem. The Seven Basic Principles of the Context-Driven School. *context-driven-testing.com*. [En línea] 17 de noviembre de 2001. [Citado el: 2 de abril de 2018.] [context-driven-testing.com](http://context-driven-testing.com).

41. [En línea] 27 de octubre de 2006. [Citado el: 2 de abril de 2018.]

[http://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/core.base\\_rup/guidances/concepts/test\\_strategy\\_9981F03E.html](http://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/core.base_rup/guidances/concepts/test_strategy_9981F03E.html).

42. [En línea] <https://synergix.wordpress.com/2008/03/15/definimos-pruebas-de-unidad-como/>.
43. Informatica Juridica.co. [En línea] 12 de noviembre de 2006. [Citado el: 2 de abril de 2018.] <http://www.informatica-juridica.com/trabajos/procedimiento-realizar-pruebas-caja-blanca/>.
44. SOMMERVILLE, Ian. *Ingeniería del software. 7ma Edición*. 2005. ISBN ISBN 84-7829-074-5.
45. —. *Ingeniería del software. 7ma Edición. 2005*. 2005. ISBN ISBN 84-7829-074-5..
46. ApacheJMETER. [En línea] 15 de enero de 2006. [Citado el: 12 de abril de 2018.] <http://jmeter.apache.org/>.

## **ANEXOS**

### **Anexo1: Entrevista con el cliente**

1. ¿Existe alguna forma en la actualidad para realizar la autenticación de clientes ligeros con el directorio activo de Nova-LTSP?
2. En caso positivo ¿De qué manera lo hace?
3. ¿Por qué usted considera necesario el uso de un Directorio Activo para la gestión de los clientes ligeros?
4. ¿Qué facilidades le gustaría a usted que le brindara el *software* a desarrollar?
5. ¿Cuál es el rol de la persona que interactuará con el *software*?

## Anexo2: Acta de Aceptación



### Acta de Aceptación de productos de trabajo

#### ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En el cumplimiento del Convenio de colaboración establecido entre el Centro de Software Libre (CESOL) y el estudiante Alejandro Rodas Cueto de la facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto: Módulo de autenticación con el directorio activo de Nova-LTSP se hace entrega del producto que se relaciona a continuación:

- Módulo de autenticación con el directorio activo de Nova-LTSP

La parte cliente luego de haber revisado el producto de trabajo relacionado anteriormente procede a firmar la aceptación de los mismo en total conformidad.



Entrega	Recibe
Nombre y Apellidos: Alejandro Rodas Cueto	Nombre y Apellidos: Yasiel Pérez Villazón
Cargo: Estudiante Facultad 1	Cargo: Lider de proyecto
Firma: 	Firma: 
Fecha: 24/05/18	

Ilustración 14: Acta de aceptación

Anexo3: Logro alcanzado de la investigación



Jornada Científica Estudiantil

**FEU**  
Federación Estudiantil Universitaria  
Universidad de las Ciencias Informáticas

La Federación Estudiantil Universitaria y el Vicedecanato de Formación de la Facultad 1 otorga el presente

**Reconocimiento**  
**A: Alejandro Rodas Cueto**

Por obtener la categoría de  
**Relevante**

con la ponencia: Módulo de autenticación con el directorio activo de Nova LTSP.

  
Alejandro Alvarez Chirino  
Presidente FEU-Fac1

  
MSc Aylín Estrada Velazco  
Vicedecana de Formación

Dado a los 21 días del mes de Mayo del 2018.

Ilustración 15: Logro alcanzado de la investigación