

**Facultad 1**

**Herramienta de Autenticación Proxy Integrada con el Sistema de Autenticación de la UCI para los dispositivos que emplean la distribución de Linux Nova.**

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

**Autor:** Carlos Manuel Vargas Piña

**Tutores**:

**Ing**. Jonathan Llanes Sánchez

**Ing. Alejandro** Romero Martínez

La Habana, marzo de 2025

Año 65 del Triunfo de la Revolución

**DECLARACIÓN DE AUTORÍA**

El(Los) autor(es) del trabajo de diploma con título <título de la tesis entre “comillas”, *en formato de letra Italic* y en **negrita**> concede(n) a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara(n) como único(s) autores de su contenido. Para que así conste firma(n) la presente a los <día> días del mes de <mes> del año <año>.

|  |  |
| --- | --- |
| **<nombre del autor>** | **<nombre del autor>** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Autor | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Autor |
| **<nombre del tutor>** | **<nombre del tutor>** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Tutor | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Tutor |

**RESUMEN**

La presente investigación aborda la problemática de la autenticación en el proxy de la Universidad de las Ciencias Informáticas (UCI) para dispositivos con la distribución GNU/Linux Nova, donde los usuarios enfrentan procesos poco intuitivos y restricciones de acceso a la red universitaria. El objetivo es desarrollar una herramienta integrada que optimice este proceso, mejorando la experiencia del usuario y el control sobre su cuenta. Se emplearon métodos teóricos como el analítico-sintético e histórico-lógico para fundamentar la solución, y empíricos como encuestas y análisis documental para identificar necesidades específicas. La aplicación propuesta incluye ejecución automática al iniciar el sistema, monitoreo en tiempo real del consumo de datos, notificaciones personalizables y una interfaz gráfica integrada en la bandeja del sistema. Los principales hallazgos revelan la inexistencia de soluciones de código abierto plenamente compatibles con Nova, justificando la necesidad de una herramienta propia. Los resultados esperados incluyen una autenticación eficiente, reducción de intervenciones técnicas y mayor control del usuario sobre su conexión. Se concluye que esta solución, basada en tecnologías como Python, Cntlm y Flet, y desarrollada bajo la metodología AUP-UCI, no solo resuelve las limitaciones actuales, sino que también contribuye a la infraestructura tecnológica de la UCI, al ser adaptable y escalable para futuros requerimientos.

**Palabras Clave:** Autenticación, Proxy, Linux, Nova, UCI, Seguridad.

<Términos simples o compuestos que permitan indexar el trabajo de acuerdo a las temáticas principales que aborda. No deben exceder la cantidad de 4 o 5 términos>

***ABSTRACT***

This research addresses the issue of proxy authentication at the University of Computer Sciences (UCI) for devices using the GNU/Linux Nova distribution, where users face unintuitive processes and internet access restrictions. The goal is to develop an integrated tool that optimizes this process, enhancing user experience and security. Theoretical methods such as analytical-synthetic and historical-logical were used to underpin the solution, alongside empirical methods like surveys and documentary analysis to identify specific needs. The proposed application features automatic startup, real-time data consumption monitoring, customizable notifications, and a graphical interface integrated into the system tray. Key findings highlight the lack of fully compatible open-source solutions for Nova, justifying the need for a custom tool. Expected outcomes include efficient authentication, reduced technical interventions, and greater user control over their connection. It is concluded that this solution, built with technologies like Python, Cntlm, and Flet, and developed under the AUP-UCI methodology, not only addresses current limitations but also contributes to UCI’s technological infrastructure, being adaptable and scalable for future needs.

**Keywords:** Authentication, Proxy, Linux Nova, UCI, Security.

**TABLA DE CONTENIDOS**

[**INTRODUCCIÓN**](#_nc4fp34fluxe) **10**

[**CAPÍTULO I: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio**](#_q1oduhky3wa3) **15**

[1.1 Conceptos Fundamentales:](#_m6nufys0s501) 15

[1.2 Funcionamiento de un servidor proxy](#_v77iyjd1kadi) 16

[1.2.2 Métodos de autenticación en sistemas operativos Linux](#_aitrzvmyysvc) 17

[1.3 Análisis de alternativas comparables](#_qk9rxpmexbbm) 18

[1.3.2 Px](#_gl564jdmvwkx) 18

[1.3.3 Proxifier](#_adt34qdg7rbv) 18

[1.3.4 Cuota-UCI](#_2iflw9s2op8) 19

[1.3.5 UCIntlm](#_owpvhsl7gzo4) 19

[1.3.6 LocalProxy](#_nfyeeqrwa7m1) 19

[1.3.7 Consideraciones finales sobre las soluciones estudiadas](#_z4b40wo782yb) 19

[1.4 Tecnologías y herramientas](#_6qu2wdj8a69v) 21

[1.4.1 Herramientas](#_tbk7prir8amf) 21

[1.4.2 Tecnologías](#_t22tvww2qzlp) 22

[1.5 Metodologías de desarrollo de software:](#_yq829cgf6per) 23

[Conclusiones del capítulo](#_hdg0rwkgm4my) 27

[**CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO**](#_d9vbfiubacxw) **28**

[2.1 Propuesta técnica](#_1w8cbzrbv06t) 28

[2.2 Captura de requisitos](#_doo0k96yysvw) 28

[2.2.1 Técnicas para la captura de requisitos](#_2kesxrjgijo4) 29

[2.3 Requisitos de software](#_ds42g3y5yiwn) 32

[2.3.1 Requisitos funcionales](#_uvhzaics67zo) 33

[2.3.2 Requisitos No Funcionales](#_98tau9oyw7av) 37

[2.4 Historias de usuario](#_pn78s7xnnot6) 38

[2.5 Representación y diseño del sistema](#_9zvl8qysm29) 45

[2.5.1 Representación de la Arquitectura del sistema](#_qy7nzr6zltsi) 45

[2.5.2 Diagrama de clases del diseño](#_1cehe2wwmrf) 47

[2.5.3 Patrones de diseño:](#_jrpmmtkpxuvh) 47

[Conclusiones del capítulo](#_yn8fm47stqc1) 51

[**REFERENCIAS BIBLIOGRÁFICAS**](#_vtfkl1r1zq6m) **52**

[**ANEXOS**](#_1uuhf7nhd1ov) **56**

**OPINIÓN DEL(OS) TUTOR(ES)**

# INTRODUCCIÓN

En la era digital actual, donde la información fluye constantemente y nuestras vidas están cada vez más interconectadas, garantizar la ciberseguridad se ha convertido en una prioridad ineludible. Uno de sus pilares fundamentales es la autenticación, el proceso que verifica la identidad de un usuario antes de conceder acceso a sistemas, aplicaciones o datos.(A3Sec, 2024)

En el contexto de las universidades, los sistemas de autenticación cumplen funciones clave, como regular el acceso a plataformas académicas, bibliotecas digitales, redes internas y servicios en línea. Su correcta implementación permite a los estudiantes y al personal administrativo interactuar con los recursos institucionales de manera segura y eficiente. Sin estos mecanismos, la integridad de la información y la experiencia de usuario pueden verse comprometidas.(Educativa, 2019)

La Universidad de las Ciencias Informáticas (UCI) ha implementado un sistema de autenticación a través de un proxy, que filtra y controla el tráfico de internet de sus usuarios. Este sistema requiere que cada usuario valide sus credenciales para poder navegar en la red, lo que garantiza un uso adecuado de los recursos digitales. Sin embargo, aunque existen soluciones que se han integrado fácilmente en sistemas operativos como Windows y Android, su implementación en la distribución GNU/Linux cubana Nova presenta desafíos específicos.

Las computadoras que emplean la distribución GNU/Linux Nova en la Universidad de las Ciencias Informáticas (UCI) dependen de la autenticación en el proxy para acceder a internet. Sin embargo, este proceso presenta diversas dificultades que afectan tanto la experiencia de los usuarios como la gestión eficiente de los recursos tecnológicos.

Uno de los principales inconvenientes es la necesidad de autenticación manual y repetitiva, lo que genera pérdida de tiempo y frustración, especialmente cuando se requiere acceso inmediato a plataformas académicas y otros servicios en línea. Además, la falta de información clara sobre el estado de la conexión dificulta la identificación y resolución de problemas de autenticación o desconexión, interrumpiendo actividades académicas y de investigación que dependen de una conexión estable.

Otro problema relevante es la gestión ineficiente de credenciales. Al no contar con un mecanismo seguro para su almacenamiento, los usuarios deben ingresar sus datos manualmente en cada conexión, lo que aumenta el riesgo de errores y vulnerabilidades de seguridad. Asimismo, la ausencia de un sistema de monitoreo en tiempo real del consumo de datos impide un uso eficiente de la cuota asignada y puede provocar interrupciones inesperadas del servicio al alcanzar el límite disponible.

En comparación con otras soluciones en sistemas operativos como Windows y Android, donde la autenticación en proxies suele manejarse con herramientas integradas o clientes específicos, Nova carece de una solución nativa eficiente. La falta de funcionalidades e integración con el sistema operativo también representa una limitación importante. Esto impide la configuración automática del proxy y la autenticación sin intervención del usuario, reduciendo la eficiencia y autonomía en el uso de los recursos tecnológicos. Además, los usuarios sin privilegios administrativos no pueden modificar la configuración del proxy, lo que restringe aún más su acceso a internet y el aprovechamiento de recursos digitales

Estos inconvenientes no solo dificultan la navegación, sino que también pueden afectar el acceso a materiales de estudio, investigación y otros recursos esenciales para la comunidad universitaria. Frente a esta problemática, se hace necesario un análisis detallado de las soluciones existentes y una evaluación de las necesidades específicas de la UCI con respecto a la autenticación en el sistema operativo.

A partir de esta **situación problemática**, la presente investigación se estructura en torno al siguiente **problema de investigación**: ¿Cómo diseñar e implementar una herramienta integrada que mejore el proceso de autenticación en el proxy de la Universidad de las Ciencias Informáticas para los dispositivos que emplean la distribución GNU/Linux Nova, mejorando la eficiencia, seguridad y experiencia del usuario? De acuerdo a este planteamiento, se define el **objeto de estudio:** proceso de autenticación de usuarios en el proxy a través de herramientas integradas. El **campo de acción:** proceso de autenticación de usuarios en el proxy a través de herramientas integradas para dispositivos que emplean distribución de Linux Nova.

El **objetivo general** que guiará el desarrollo de la investigación es: desarrollar una aplicación integrada con el sistema de autenticación del proxy de la Universidad de las Ciencias Informáticas que optimice el acceso a internet en dispositivos que emplean distribución de Linux Nova.

Para cumplir el objetivo, se plantean las siguientes **preguntas científicas**:

1. ¿Cuáles son los referentes teóricos fundamentales que sustentan la investigación relacionada con el proceso de desarrollo de una aplicación integrada con el sistema de autenticación del proxy de la Universidad de las Ciencias Informáticas que optimice el acceso a internet en los dispositivos que emplean la distribución de Linux Nova?
2. ¿Qué propuesta de solución se define para facilitar el proceso de autenticación en el proxy de la Universidad de las Ciencias Informáticas en los dispositivos que emplean la distribución de Linux Nova?
3. ¿Qué componentes son necesarios para implementar una aplicación integrada con el sistema de autenticación del proxy de la Universidad de las Ciencias Informáticas que optimice el acceso a internet en los dispositivos que emplean la distribución Linux Nova?
4. ¿Qué resultados se obtendrán al validar el funcionamiento de la aplicación integrada con el sistema de autenticación del proxy de la Universidad de las Ciencias Informáticas que optimice el acceso a internet en los dispositivos que emplean la distribución Linux Nova?
5. ¿Cómo impactará la aplicación propuesta en la eficiencia del proceso de autenticación, la seguridad de las credenciales y la experiencia de los usuarios en comparación con el método actual?

Para abordar efectivamente las preguntas planteadas, se llevarán a cabo las siguientes **tareas de investigación**, las que contribuirán al logro del objetivo general propuesto:

1. Analizar los referentes teóricos fundamentales que sustentan la investigación relacionada con el proceso de desarrollo de una aplicación integrada con el sistema de autenticación del proxy para dispositivos con distribución Nova de Sistema Operativo Linux.
2. Diseñar una aplicación integrada con el sistema de autenticación del proxy para dispositivos con distribución Nova de Sistema Operativo Linux.
3. Implementar las funcionalidades de la aplicación integrada con el sistema de autenticación del proxy para dispositivos con distribución Nova de Sistema Operativo Linux.
4. Validar el funcionamiento de la aplicación integrada con el sistema de autenticación del proxy para dispositivos con distribución Nova de Sistema Operativo Linux.
5. Evaluar el impacto de la aplicación en la eficiencia del proceso de autenticación, la seguridad de las credenciales y la experiencia de los usuarios en comparación con el método actual.

**Métodos de Investigación**

Para el desarrollo de esta investigación, se emplearán métodos teóricos y empíricos que permitirán analizar, diseñar, implementar y validar la solución propuesta para la autenticación en el proxy de la Universidad de las Ciencias Informáticas (UCI) en dispositivos que emplean la distribución de Linux Nova.

### **Métodos teóricos:**

**Analítico-Sintético:** Este método permitirá realizar un exhaustivo análisis de teorías y documentos relevantes al ámbito de los sistemas de autenticación y el uso de proxies en entornos universitarios. A través de esta técnica, se identificarán las características y necesidades específicas del proceso de autenticación en la UCI, basándose en el análisis de los referentes teóricos consultados.

**Histórico-lógico:** Se utilizará para estudiar la evolución histórica y las tendencias actuales en la autenticación de usuarios mediante proxies, con énfasis en entornos educativos y en distribuciones Linux. También permitirá examinar soluciones similares implementadas en otros sistemas operativos y evaluar metodologías de desarrollo, tecnologías, herramientas, lenguajes de programación y enfoques de modelado utilizados en investigaciones previas.

**Modelación:** Se empleará para representar los procesos, características y componentes de la solución propuesta. Para ello, se utilizará un lenguaje de modelado unificado (UML), que permitirá diseñar los artefactos resultantes del proceso de ingeniería de software. Estos modelos describirán la funcionalidad de la aplicación, la interacción de los usuarios con el sistema y la arquitectura general de la solución, en correspondencia con los estándares de la metodología AUP-UCI escenario 4.

### **Métodos empíricos:**

**Observación:** Este método facilitará la obtención de conocimiento sobre el funcionamiento de soluciones similares en otros sistemas operativos. A partir de esta observación, se estudiará el contexto específico de la UCI y las herramientas de autenticación actualmente empleadas, lo que permitirá identificar las mejores prácticas y tendencias para el desarrollo de la solución.

**Análisis documental:** Se realizará una revisión bibliográfica detallada de fuentes de información relevantes, incluyendo artículos científicos, documentación técnica y normativas sobre autenticación en redes universitarias y proxies en entornos Linux. Este análisis contribuirá a la fundamentación teórica de la investigación y a la concepción de la solución propuesta.

**Encuesta o entrevista:** Se utilizarán encuestas y entrevistas con usuarios y administradores de red para conocer sus expectativas y necesidades en relación con la autenticación en el proxy. En la fase de validación, estas herramientas permitirán evaluar el nivel de satisfacción con la aplicación desarrollada, asegurando que cumple con los requisitos funcionales establecidos.

# CAPÍTULO I: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio

Este capítulo explora el desafío de la autenticación en servidores proxy, enfocándose en el contexto específico de la Universidad de las Ciencias Informáticas (UCI) y las necesidades particulares de sus usuarios. El objetivo es resaltar la relevancia del problema y justificar la creación de una solución personalizada que permita a los usuarios de GNU/Linux Nova autenticarse de manera eficiente y segura, mientras mejora la experiencia de uso y optimiza el acceso a los recursos de red. Para ello, se analizan aspectos clave como los protocolos de autenticación, las metodologías de desarrollo y las herramientas tecnológicas disponibles.

La investigación se sustenta en un análisis exhaustivo de fuentes bibliográficas, estudios de casos y soluciones existentes, tanto a nivel internacional como local, teniendo en cuenta las particularidades del entorno de la UCI.

## 1.1 Conceptos Fundamentales:

**HTTP**: de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente).(Generalidades del protocolo HTTP - HTTP | MDN, 2024)

**Autenticación proxy**: válida y verifica la solicitud de un cliente. Los proxies también sirven como dispositivos de control de acceso que bloquean las solicitudes hasta que el usuario o el cliente proporcionan credenciales válidas de permiso de acceso al proxy (ProxyScrape 2022).

**407 Autenticación proxy requerida**: El código de estado de respuesta de error del cliente HTTP indica que la solicitud no tuvo éxito porque carece de credenciales de autenticación válidas para el servidor proxy que se encuentra entre el cliente y el servidor con acceso al recurso solicitado («407 Se requiere autenticación de proxy - HTTP | MDN» 2024).

**Servidor proxy:**Un servidor proxy proporciona una puerta de enlace entre los usuarios e Internet. Es un servidor denominado “intermediario”, porque está entre los usuarios finales y las páginas web que visitan en línea. («Servidor proxy», 2025)

**Servidores Squid:** es un servidor de almacenamiento en caché de proxy que proporciona servicios de proxy y caché para el Protocolo de transporte de hipertexto (HTTP) y otros protocolos populares.Actúa como intermediario entre los servidores web y los clientes. (About Squid Proxy Servers, 2025)

## 1.2 Funcionamiento de la autenticación en un servidor proxy

En este epígrafe se presentan los fundamentos teóricos asociados a la autenticación mediante servidores proxy, con el objetivo de lograr una comprensión integral de su funcionamiento. Estos conceptos resultan esenciales para aplicar de manera efectiva los conocimientos adquiridos en el diseño y desarrollo de la solución propuesta.

***1.2.2* Autenticación proxy**

## La Autenticación proxy es un mecanismo que verifica la identidad de los usuarios antes de concederles acceso a los recursos a través de un servidor proxy. Su funcionamiento consiste en interceptar las solicitudes del cliente y exigir credenciales antes de permitir la conexión. Este proceso se basa en el protocolo HTTP, donde el cliente envía una solicitud con información adicional, y el servidor proxy responde solicitando autenticación mediante el encabezado Proxy-Authenticate. Para acceder, los usuarios deben proporcionar un nombre de usuario y contraseña, los cuales se incluyen en la cabecera Proxy-Authentication de la solicitud HTTP. En entornos con proxies universitarios, las credenciales pueden asignarse a cada usuario o configurarse de manera individual para cada proxy, garantizando así un acceso seguro y controlado (ProxyScrape, 2022). Este procedimiento se compone de las siguientes etapas.

1. El cliente envía una solicitud al servidor proxy para acceder a un recurso (por ejemplo, una página web)
2. El proxy detecta que el recurso solicitado requiere autenticación y responde con un código de estado HTTP 407 Proxy Authentication Required.
3. El cliente recibe el 407 y entonces debe enviar nuevamente la solicitud incluyendo credenciales en el encabezado Proxy-Authorization
4. El cliente agrega el encabezado Proxy-Authorization con el esquema de autenticación que usa el proxy
5. El proxy recibe la solicitud autenticada, verifica las credenciales:

Si son correctas, deja pasar la solicitud al servidor de destino.

Si son incorrectas, responde nuevamente con 407 Proxy Authentication Required

1. El servidor de destino responde al proxy, y el proxy reenvía la respuesta al cliente.

### **1.2.2** **Métodos de autenticación proxy**

El sistema de autenticación para proxies en los sistemas GNU/Linux, como en el caso de Nova, permite utilizar diferentes esquemas de autenticación mediante el protocolo HTTP. Estos esquemas pueden variar en términos de seguridad, disponibilidad y compatibilidad con el software del cliente y el servidor. A continuación, en la Tabla 1, se presentan los métodos de autenticación examinados en esta investigación, con una breve descripción de cada uno, para poder seleccionar el más adecuado según las necesidades de seguridad, la infraestructura existente y la facilidad de implementación.

|  |  |
| --- | --- |
| **Esquema de autenticación** | **Descripción** |
| Basic | La autenticación básica HTTP es un sencillo mecanismo de desafío y respuesta con el que un servidor puede solicitar información de autenticación (un ID de usuario y una contraseña) a un cliente. El cliente pasa la información de autenticación al servidor en una cabecera de autorización. («Autenticación proxy», 2022.) |
| Digest | Es usado para confirmar la identidad de un usuario antes de servir información sensible, como el historial de transacciones de un banco. Se aplica una función hash a la contraseña antes de ser enviada sobre la red, lo que resulta más seguro que enviarla en texto plano como en la autenticación básica («Digest access authentication» 2020). |
| NTLM | Es un protocolo de autenticación de desafío-respuesta utilizado para autenticar a un cliente. Cuando el cliente solicita acceso, el servicio le envía un desafío que requiere una operación matemática con su token de autenticación. El resultado se devuelve al servicio, que lo valida directamente o lo envía al controlador de dominio para su verificación y este acepta o rechaza la solicitud. (NTLM, 2020) |
| Kerberos | El protocolo de autenticación de redes informáticas sin contraseña, desarrollado por el MIT, fue creado para abordar y resolver problemas de seguridad en redes. Actualmente, es ampliamente adoptado por diversas organizaciones para implementar el inicio de sesión único . Este protocolo permite la transmisión segura de los datos de identidad del usuario a las aplicaciones, y se basa en dos funciones clave: la autenticación y la seguridad(«What Is Kerberos and How Does It Work? | Ping Identity», 2020.). |

**Tabla 1***: Descripción de los esquemas de autenticación estudiados.*

Para determinar los esquemas de autenticación que se implementarán en la solución a desarrollar, se llevó a cabo el siguiente proceso:

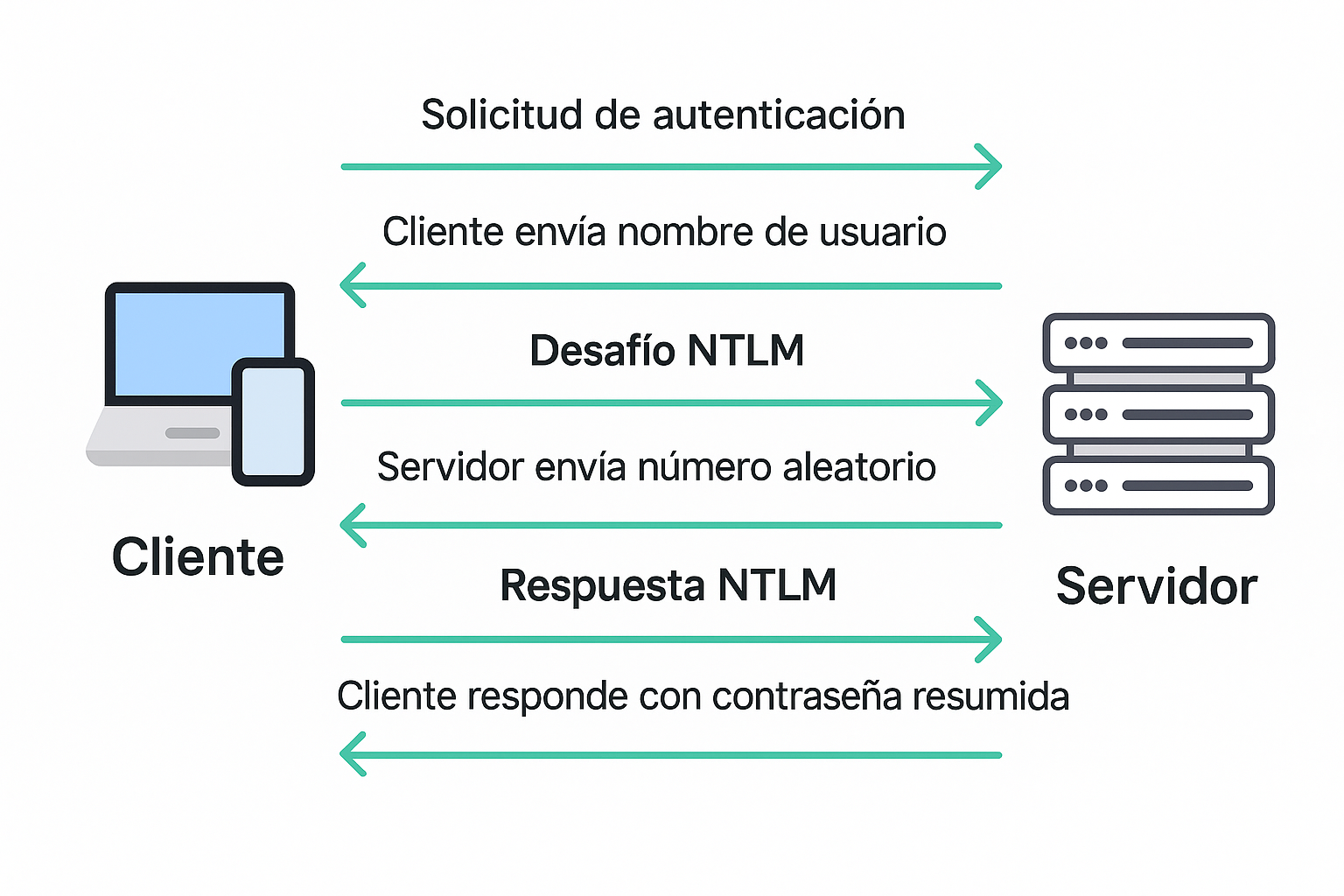
1. **Estudio de esquemas de autenticación en aplicaciones similares**: Se analizaron los esquemas de autenticación soportados en las distribuciones GNU/Linux como Nova.
2. **Entrevistas a especialistas**: Se realizaron entrevistas a especialistas del centro Cisol , y en de el de Redes y Administradores de servidores proxy de la UCI, con el objetivo de conocer los esquemas de autenticación más utilizados y sus particularidades en entornos similares al de la UCI.
3. **Análisis del soporte en el servidor proxy Squid:** Se evaluó el soporte de los esquemas de autenticación identificados en las acciones anteriores en el servidor proxy Squid y se analizaron la documentación y tesis de pregrado sobre el funcionamiento de este.

A partir de los hallazgos obtenidos en las fases anteriores, se optó por utilizar NTLM como el esquema de autenticación principal en la solución a desarrollar.El protocolo NTLM representa una opción ideal para entornos Linux debido a su equilibrio entre compatibilidad, estabilidad y soporte comunitario. Su implementación ha sido ampliamente adoptada y perfeccionada en sistemas GNU/Linux, lo que garantiza un funcionamiento confiable respaldado por una documentación sólida y abundante (Sánchez-Oro, 2023). Asimismo, NTLM muestra una alta compatibilidad con servidores proxy como Squid. Gracias a esta compatibilidad, se pueden establecer mecanismos de autenticación robustos sin recurrir a soluciones complejas o poco soportadas.Todo ello convierte a NTLM en una alternativa técnica y operativa confiable para la autenticación en redes que utilizan distribuciones como Nova.

**Proceso de Autenticación NTLM:**

La autenticación mediante el protocolo NTLM involucra tres entidades principales: el cliente, el servidor y el controlador de dominio. Este último almacena la información relacionada con las contraseñas de los usuarios y, en ocasiones, realiza cálculos en nombre del servidor. El proceso se desarrolla en los siguientes pasos(alvinashcraft, 2022):

1. **Inicio de sesión del usuario:** El usuario introduce su nombre de dominio, nombre de usuario y contraseña en el sistema cliente. El cliente genera un hash criptográfico de la contraseña y descarta la contraseña en texto plano.​
2. **Envío del nombre de usuario:** El cliente transmite el nombre de usuario al servidor en texto claro.​
3. **Generación del desafío:** El servidor crea un valor aleatorio de 16 bytes, conocido como "desafío", y lo envía al cliente.​
4. **Respuesta al desafío:** El cliente cifra el desafío utilizando el hash de la contraseña y devuelve el resultado al servidor.​
5. **Verificación con el controlador de dominio:** El servidor remite al controlador de dominio el nombre de usuario, el desafío original y la respuesta proporcionada por el cliente.​
6. **Cálculo y comparación**: El controlador de dominio recupera el hash de la contraseña asociado al nombre de usuario y utiliza este para cifrar el desafío. Luego, compara este resultado con la respuesta enviada por el cliente.​
7. **Resultado de la autenticación:** Si los valores coinciden, la autenticación es exitosa; de lo contrario, se rechaza el acceso.​



**Proceso autenticación NTLM**

## 1.3 Análisis de alternativas comparables

En este epígrafe se realiza un análisis exhaustivo de las principales soluciones que permiten la autenticación a través de servidores proxy web, tanto en el contexto nacional como internacional. Se examinan las ventajas y limitaciones de cada una de estas alternativas, con el fin de ofrecer una visión comparativa que contribuya a la comprensión de su aplicabilidad y eficacia.

**1.3.1 Alternativas**

**ProxyCap**

Es una herramienta que permite dirigir el tráfico de Internet a través de servidores proxy de manera flexible. Ofrece soporte para diversos tipos de proxies, como HTTP, HTTPS, SOCKS4 y SOCKS5, y permite enrutar tráfico específico de aplicaciones o sitios web a través de un proxy. Su interfaz fácil de usar no requiere experiencia técnica y proporciona reglas personalizables para controlar con precisión cómo se maneja el tráfico de Internet, brindando un mayor control sobre la privacidad y seguridad en línea (Shim, 2024). (https://proxycap.en.lo4d.com)

### **Cntlm**

Es un proxy de autenticación NTLM diseñado para acceder a servicios web a través de un proxy que requiere autenticación NTLM, como es común en entornos corporativos. Actúa como un intermediario local entre las aplicaciones del usuario y el proxy corporativo, gestionando la autenticación NTLM de manera transparente. (Brave, 2024) (https://cntlm.sourceforge.net/)

### **Cuota-UCI**

Es un software diseñado para mostrar el consumo de Internet de los usuarios en la Universidad de Ciencias Informáticas, de acuerdo con el servicio web disponible en https://cuotas.uci.cu/servicios/v1/InetCuotasWS.wsdl. Esta herramienta cuenta con una interfaz visual para la gestión del proxy Cntlm y ha sido desarrollada utilizando el framework Qt (Torres, 2015)(https://codecomunidades.prod.uci.cu/antoniodjog/Cuota)

### **UCIntlm**

Permite utilizar Internet en la empresa autenticando el dispositivo en el servidor proxy de la escuela o compañía (Orozco, 2015; Akiel, 2024)(https://github.com/jadolg/UCIntlm)

### **LocalProxy**

Proporciona un servicio de proxy local en el dispositivo con sistema operativo Android con el objetivo principal de realizar el proceso de autenticación con un servidor proxy, facilitando el acceso a Internet en las aplicaciones instaladas. , que permiten el acceso a Internet a través de un servidor proxy institucional(Rodriguez, 2016.)(https://codecomunidades.prod.uci.cu/darodriguez/LocalProxy)

### **1.3.2 Consideraciones finales sobre las soluciones estudiadas**

A modo resumen se elaboró la siguiente tabla resumen que permite visualizar de manera clara las principales características técnicas de las soluciones estudiadas, en función de su relevancia para el desarrollo de la aplicación integrada al sistema operativo Nova.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aplicaciones** | **Soporta NTLM** | **Código Abierto** | **Documentación** | **Soporte para Linux** | **Integrada al sistema** |
| **ProxyCap** |  |  |  | x |  |
| **CNTLM** | x | x | x | x |  |
| **Cuota-UCI** | x | x |  |  | x |
| **UCIntlm** | x | x |  | (solo Android) | x |
| **LocalProxy** | x | x | (limitada) |  | x |

El análisis comparativo realizado evidencia que ninguna de las herramientas evaluadas satisface plenamente los requisitos para una aplicación optimizada e integrada al sistema operativo Nova. Aunque existen soluciones como Cuota-UCI, cuyo código abierto permite comprender su conexión con los servidores de la universidad, presentan limitaciones significativas. Por ejemplo, ProxyCap no es compatible con Linux; CNTLM, aunque funcional, no cuenta con una interfaz gráfica ni con integración a servicios institucionales; Cuota-UCI no ha sido adaptada al entorno Nova; UCIntlm está restringida a dispositivos Android; y LocalProxy carece de documentación suficiente y ofrece escasa flexibilidad. Esta situación demuestra la necesidad de desarrollar una nueva herramienta que se integre de manera eficiente con Nova. Para ello, se utilizará CNTLM como intermediario para la autenticación mediante el protocolo NTLM, y se tomará como referencia el funcionamiento de Cuota-UCI, especialmente en lo relacionado con su interacción con los servidores de la universidad.

## 1.4 Tecnologías y herramientas

La selección de tecnologías es un aspecto fundamental en el desarrollo de una solución informática, ya que debe alinearse con los requisitos específicos del software a implementar. En esta sección se describen y justifican las tecnologías seleccionadas, así como las herramientas que se utilizarán para su desarrollo.

### **1.4.1 Herramientas**

**Zed 0.18:** Es un editor de código abierto, moderno, ligero y de alto rendimiento, desarrollado en Rust para garantizar velocidad y eficiencia. Diseñado por exdesarrolladores de Atom, ofrece una interfaz minimalista y herramientas optimizadas para la productividad. Soporta múltiples lenguajes, integración con terminal y atajos de teclado eficientes. Su principal innovación es la colaboración en tiempo real, permitiendo que varios usuarios editen simultáneamente. Zed busca ser una alternativa ágil a editores como VS Code y Neovim, ofreciendo una experiencia fluida sin consumo excesivo de recursos.(«Zed - The Editor for What’s Next»)

**Excalidraw:** es una herramienta online de dibujo de código abierto que permite crear bocetos y diagramas de manera fácil e intuitiva. Su interfaz de usuario minimalista hace que dibujar sea tan intuitivo como en papel, lo cual es ideal para brainstorming, diagramas de flujo y wireframing. Además, Excalidraw permite la colaboración en tiempo real, permitiendo a varios usuarios trabajar juntos en un mismo proyecto simultáneamente. (Codemotion 2024)

### **1.4.2 Tecnologías**

**Python3.10:** es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma («Python» 2025). Esta versión ha sido elegida específicamente porque es la última disponible en Nova, lo que garantiza compatibilidad con el entorno y facilita su implementación sin necesidad de modificaciones adicionales.

**Cntlm 0.92.3 :** es un proxy HTTP de autenticación NTLM/NTLMv2 de código abierto, diseñado para facilitar el acceso a redes corporativas que requieren autenticación NTLM. Actúa como un intermediario entre tu máquina y el proxy corporativo, gestionando la autenticación y permitiendo que aplicaciones que no soportan NTLM accedan a internet sin problemas(«Cntlm: Fast NTLM Authentication Proxy in C», 2025)

**Flet 0.26:**  es un marco de trabajo que permite a los desarrolladores crear aplicaciones en tiempo real para web, dispositivos móviles y escritorios utilizando Python, sin necesidad de experiencia en desarrollo frontend ni movil. Construido sobre Flutter, Flet simplifica el modelo de Flutter combinando "widgets" en "controles" listos para usar, adoptando un modelo de programación imperativa. Esto asegura que las aplicaciones tengan una apariencia profesional y sean compatibles con múltiples plataformas. («Flet: Power of your favourite language + Flutter UI» 2023)

**Plyer 2.2:** es una biblioteca de Python que proporciona una interfaz multiplataforma para acceder a características específicas del hardware y del sistema operativo, como notificaciones, sensores, GPS, cámara, entre otros. Su objetivo es facilitar el desarrollo de aplicaciones que requieran funcionalidades nativas sin tener que escribir código específico para cada plataforma(«Plyer 2.2.0.dev0 documentation», 2017.). Esta biblioteca de Python se utilizará para enviar notificaciones al usuario sobre su consumo de cuota.

**Git 2.48:** es un sistema de control de versiones distribuido y de código abierto, diseñado para gestionar proyectos de desarrollo de software de manera eficiente y rápida, incluso aquellos con un gran volumen de archivos de código fuente. Permite a múltiples desarrolladores trabajar simultáneamente en un proyecto, llevando un registro detallado de los cambios realizados y facilitando la colaboración.(«Git», 2015.)

**GitHub Copilot:** es un asistente basado en Inteligencia Artificial que ayuda a escribir código en tiempo real. GitHub empezó a mostrar este sistema a mediados de 2021, y poco a poco ha ido desarrollándolo . Lo que hace este asistente es sugerirte código para seguir completando un programa que estés desarrollando.

## 1.5 Metodologías de desarrollo de software:

Las metodologías de desarrollo de software proporcionan un marco estructurado para planificar, ejecutar y entregar proyectos de software, adaptándose a las características de cada proyecto y equipo, así como a la cultura organizacional. Estas metodologías varían desde enfoques lineales hasta iterativos, permitiendo su ajuste según las necesidades específicas del proyecto y del equipo involucrado (Abstracta Team, 2024).

El desarrollo de software enfrenta diversos retos que, si no se gestionan adecuadamente, pueden comprometer la calidad y el éxito del proyecto. Para abordar estos desafíos, la Universidad de las Ciencias Informáticas (UCI) ha adoptado una versión personalizada del Proceso Ágil Unificado (AUP), conocida como AUP-UCI, que se adapta a las particularidades del entorno académico y organizacional de la institución (Sánchez, 2015). La metodología AUP, que originalmente consta de cuatro fases, ha sido simplificada en las siguientes:

En la disciplina de Requisitos, se prioriza la construcción de un modelo que represente adecuadamente el sistema a desarrollar, tomando en cuenta tanto los requisitos funcionales como los no funcionales. Para ello, se emplean tres enfoques principales:

* **Casos de Uso del Sistema (CUS)**
* **Historias de Usuario (HU)**
* **Descripción de Requisitos por Proceso (DRP)**

La selección del enfoque adecuado depende del tipo de modelado de negocio utilizado en el proyecto. AUP-UCI establece cuatro escenarios, lo que permite adaptar la disciplina de requisitos a distintos contextos (Sánchez, 2015). Los escenarios definidos son los siguientes:

1. Si se usan Casos de Uso del Negocio (CUN), se utiliza CUS.
2. Si se emplea un Modelo Conceptual (MC), se usa CUS.
3. Si se utiliza una Descripción de Proceso de Negocio (DPN), se opta por DRP.
4. Si no existe un modelado formal de negocio o ya se ha realizado un análisis exhaustivo del negocio, se emplean HU.

**Selección del Escenario 4 para la Investigación**

En este proyecto se ha elegido el Escenario 4 de AUP-UCI, fundamentado en los siguientes aspectos:

1. **Uso de Historias de Usuario (HU**): Este escenario se enfoca en la utilización de HU para modelar los requisitos del sistema.
2. **Participación activa del cliente:** El escenario requiere una participación continua y activa del cliente, un aspecto clave en este proyecto colaborativo, donde la interacción constante garantiza que los requisitos se definan y validen conforme a las necesidades reales.
3. **Adecuación al contexto:** Este enfoque resulta idóneo cuando ya se ha realizado un análisis previo exhaustivo del negocio a informatizar, facilitando una definición precisa de los requisitos. Además, es adecuado para proyectos de alcance limitado, ya que las HU promueven la claridad y evitan la inclusión de información innecesaria.

## 1.6 Conclusiones del capítulo

A partir del análisis realizado, se concluye que la autenticación en servidores proxy representa un desafío técnico significativo en entornos institucionales como la Universidad de las Ciencias Informáticas (UCI), especialmente para los usuarios del sistema operativo GNU/Linux Nova. La necesidad de garantizar un acceso seguro, eficiente y transparente a los recursos de red justifica el desarrollo de soluciones personalizadas. Se evidenció que el protocolo NTLM, por su compatibilidad, estabilidad y amplia adopción en entornos Linux, constituye la alternativa más adecuada para implementar la autenticación en este contexto. Asimismo, se analizaron distintas herramientas existentes tanto internacionales como desarrolladas en la propia universidad, lo que permitió identificar buenas prácticas y limitaciones en su diseño y funcionalidad. Estos elementos fundamentan la decisión de desarrollar una aplicación integrada con Nova, que no solo automatice la autenticación NTLM mediante CNTLM, sino que también facilite el control del consumo de internet, mejorando así la experiencia del usuario y promoviendo una gestión eficiente de los recursos institucionales.

# CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

Este capítulo presenta la solución propuesta para la autenticación en el servidor proxy de la Universidad de las Ciencias Informáticas (UCI). Se describen los aspectos técnicos de la aplicación y el proceso de captura de requisitos, abarcando tanto los funcionales como los no funcionales, con base en el análisis de sistemas similares y encuestas a usuarios de GNU/Linux Nova.

Además, se incluyen las historias de usuario que detallan las principales funcionalidades del sistema. En la fase de análisis y diseño, se define una arquitectura por capas que garantiza modularidad y escalabilidad, junto con la aplicación de patrones de diseño que optimizan la estructura del código. Finalmente, se presentan diagramas clave como el modelo de datos y el diagrama de clases, proporcionando una visión clara de la estructura del sistema.

## 2.1 Propuesta técnica

Se propone desarrollar una aplicación de escritorio personalizada para la Universidad de las Ciencias Informáticas (UCI), destinada a usuarios del sistema operativo GNU/Linux Nova. El objetivo principal es facilitar y asegurar un método de autenticación eficiente en el servidor proxy de la institución.

**Características y Tecnologías:**

1. Interfaz Gráfica Intuitiva: Se implementará una interfaz de usuario clara y fácil de usar desarrollada con el framework Flet.
2. Notificaciones Integradas: Mediante la librería Plyer, la aplicación ofrecerá notificaciones personalizadas e integradas con el sistema operativo para mantener informado al usuario.
3. Monitoreo y Control:
4. Gestión Simplificada: La interfaz permitirá una configuración y gestión sencilla de los parámetros del proxy.

* Incluirá un sistema para monitorear el consumo de datos en tiempo real.

1. Ícono en la Bandeja del Sistema (System Tray):La aplicación incluirá un ícono persistente en la bandeja del sistema que permitirá visualizar de manera rápida el estado de conexión al proxy. Además, ofrecerá acceso a acciones rápidas como conectar, desconectar o abrir la ventana principal de configuración.

## 2.2 Captura de requisitos

La captura de requisitos es una fase esencial en el desarrollo de cualquier sistema o aplicación de software. Este proceso se centra en identificar, documentar y validar de manera precisa las necesidades, expectativas y restricciones tanto de los usuarios finales como de otras partes interesadas (conocidas como stakeholders). El propósito fundamental es asegurar que el software resultante cumpla con los objetivos definidos y satisfaga las expectativas. Una correcta captura de requisitos es crucial porque define con claridad qué debe hacer el sistema y cómo debe comportarse. Estos requisitos documentados sirven como pilar fundamental para guiar las etapas posteriores de diseño, desarrollo y pruebas del software. (Ana Bescós Domínguez 2019)

**2.2.1 Técnicas para la captura de requisitos**

Con el objetivo de identificar las necesidades de los usuarios en el desarrollo de la solución Cuota-UCI, se emplearon técnicas destinadas a determinar y documentar los requisitos funcionales y no funcionales del sistema. Esta actividad se llevó a cabo de manera continua a lo largo del ciclo de vida del proyecto, combinando diversas estrategias de identificación en diferentes etapas.

Durante esta fase de investigación, se consideraron como principales fuentes para la recolección de requisitos las siguientes:

**Análisis de las políticas de acceso a internet en la Universidad de las Ciencias Informáticas (UCI):** Se examinaron los mecanismos de autenticación a través de la red de la universidad, así como los sistemas de control de cuota de navegación implementados .

**Observación direct**a: Se realizó un estudio detallado de los procedimientos actuales de autenticación y consumo de cuota, identificando debilidades y oportunidades de mejora.

**Consulta a especialistas**: Se mantuvieron entrevistas y sesiones de consulta con expertos en redes y conectividad del Centro de Informatización , quienes aportaron criterios técnicos fundamentales para el diseño de la solución.

**Análisis de Sistemas Existentes:** Se llevó a cabo un estudio comparativo de aplicaciones similares a la solución propuesta, Este análisis, basado en el método histórico-lógico, permitió examinar la evolución de las soluciones de autenticación a lo largo del tiempo y extraer lecciones valiosas de implementaciones anteriores. La evaluación de los sistemas existentes facilitó la identificación de ventajas y limitaciones.

## 2.3 Requisitos de software

Los requisitos de un proyecto de software son las funciones, características y restricciones que debe cumplir el producto final. En otras palabras, los requisitos definen qué debe hacer el software, cómo debe verse y las condiciones que deben cumplirse para que se considere exitoso. (Solutions 2021)

**2.3.1 Requisitos funcionales**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nombre** | **Descripción** | **Complejidad** | **Prioridad** |
| RF-01 | Autenticación de usuario | Permitir a los usuarios autenticarse en la aplicación utilizando sus credenciales de la universidad. | Alta | Alta |
| RF-03 | Cambiar de contraseña | Permitir al usuario cambiar la contraseña de su cuenta institucional desde la aplicación. | Alta | Alta |
| RF-04 | Integrar acceso desde bandeja sistema | Proporcionar un acceso rápido a la aplicación mediante un ícono en la bandeja del sistema. | Alta | Alta |
| RF-05 | Monitorear consumo datos tiempo real | Permitir visualizar el consumo de datos del usuario en tiempo real. | Media | Alta |
| RF-08 | Controlar servicio CNTLM | Permitir al usuario iniciar, detener y reiniciar el servicio CNTLM directamente desde la aplicación. | Alta | Alta |
| RF-13 | Configuración manual del proxy | Permitir modificar manualmente los parámetros de configuración del proxy. | Media | Alta |
| RF-02 | Mantener sesión iniciada | Permitir mantener la sesión iniciada para evitar autenticaciones manuales repetitivas. | Alta | Media |
| RF-07 | Monitorear aplicaciones conectadas | Mostrar en tiempo real las aplicaciones o procesos que están utilizando la conexión al proxy, incluyendo nombre y estado de conexión. | Media | Media |
| RF-09 | Ejecutar al iniciar el sistema | Ejecutar automáticamente la aplicación al arrancar el sistema operativo. | Media | Media |
| RF-10 | Mostrar estado de conexión | Mostrar el estado actual de la conexión al proxy (conectado, desconectado o error de autenticación). | Media | Media |
| RF-11 | Notificar límite de datos alcanzado | Enviar notificaciones cuando el usuario alcance un porcentaje específico de su cuota de datos. | Media | Alta |
| RF-14 | Mostrar velocidad de conexión red | Mostrar la velocidad actual de subida y bajada de la conexión de red utilizada por el proxy. | Media | Media |
| RF-15 | Cerrar sesión | Permitir a los usuarios cerrar sesión en el momento que lo deseen. | Media | Media |
| RF-16 | Personalizar notificaciones | Permitir al usuario personalizar el tipo y frecuencia de las notificaciones de la aplicación. | Media | Media |

### **2.3.2 Requisitos No Funcionales**

#### Los requisitos no funcionales, también conocidos como "requisitos de calidad" o "requisitos de atributos del sistema", son características y criterios que describen cómo debe ser el rendimiento, la seguridad, la usabilidad y otros aspectos de un sistema o software más allá de su funcionalidad básica. (Arizbé Ken, 2023)

### **Seguridad**

* **RNF1:** Las contraseñas de los usuarios deben almacenarse cifradas .

### **Rendimiento y Eficiencia**

* **RNF2:** La autenticación debe completarse en menos de 4 segundos en condiciones normales de conexión.
* **RNF3:** El consumo de memoria ram de la aplicación no debe exceder los 70 MB durante su funcionamiento.

### **Usabilidad y Experiencia del Usuario**

* **RNF4:** La interfaz de usuario debe estar diseñada siguiendo los principios de Material Design.
* **RNF5:** Cada acción del usuario debe generar retroalimentación inmediata mediante notificaciones o mensajes visuales.
* **RNF6**: El ícono de la bandeja del sistema debe mostrar el estado actual de la conexión mediante cambios de color o símbolos.

## 

## 2.4 Historias de usuario

|  |  |  |
| --- | --- | --- |
| Historia de usuario | | |
| Número: HU-01 | **Nombre del requisito:** Autenticación de usuario | |
| Programador: Carlos Manuel Vargas Piña | | **Iteración asignada:1** |
| Prioridad: Alta | | **Tiempo estimado:** 4 días |
| Riesgo en desarrollo: | | **Tiempo real:** |
| Descripción: Debe permitir a los usuarios autenticarse con sus credenciales de la UCI en la aplicación ingresando su usuario y contraseña en una interfaz gráfica. | | |
| Observaciones:El usuario debe poder ingresar su usuario y contraseña en una interfaz clara e intuitiva. | | |
| Prototipo de interfaz: | | |

**Tabla 4** Historia de usuario 01: Autenticación de usuario

|  |  |  |
| --- | --- | --- |
| Historia de usuario | | |
| Número: HU-02 | **Nombre del requisito:** Controlar servicio CNTLM | |
| Programador: Carlos Manuel Vargas Piña | | **Iteración asignada:1** |
| Prioridad: Alta | | **Tiempo estimado:** 5 días |
| Riesgo en desarrollo: | | **Tiempo real:** |
| Descripción: La aplicación debe monitorear en tiempo real el estado de la conexión al proxy y reflejarlo visualmente en la interfaz. Los estados posibles incluyen: conectado y desconectado notificar al usuario.  La aplicacion debe mostrar un boton con icono para activar o descativar la conexion mediante proxy | | |
| Observaciones: El usuario debe visualizar un indicador en la interfaz de la aplicación y en la bandeja del sistema con colores diferenciados para cada estado (verde para conectado, rojo para desconectado), | | |
| Prototipo de interfaz: | | |

***Tabla*** *5 Historia de usuario 02:* *Estado de conexión*

|  |  |  |
| --- | --- | --- |
| Historia de usuario | | |
| Número: HU-03 | **Nombre del requisito:** Monitoreo de consumo | |
| Programador: Carlos Manuel Vargas Piña | | **Iteración asignada:1** |
| Prioridad: Alta | | **Tiempo estimado:** 2 días |
| Riesgo en desarrollo:   * Frecuencia de actualización: Si se consulta en intervalos muy cortos, podría generar sobrecarga en el sistema o retrasos en la interfaz. | | **Tiempo real:** |
| Descripción: La aplicación debe permitir a los usuarios visualizar en tiempo real el consumo de datos utilizados a través de sus credenciales | | |
| **Observaciones:**   * La información debe actualizarse automáticamente sin necesidad de interacción del usuario. * Se debe incluir una representación gráfica y numérica del consumo. | | |
| Prototipo de interfaz: | | |

**Tabla** 6 Historia de usuario 03 :Monitoreo de consumo

|  |  |  |
| --- | --- | --- |
| Historia de usuario | | |
| Número: HU-04 | **Nombre del requisito:** Notificación de Límite de Datos | |
| Programador: Carlos Manuel Vargas Piña | | **Iteración asignada:1** |
| Prioridad: Alta | | **Tiempo estimado:** 3 días |
| Riesgo en desarrollo: | | **Tiempo real:** |
| Descripción: La aplicación debe monitorear el consumo de datos y notificar al usuario cuando alcance un porcentaje configurable de su cuota disponible. El usuario podrá seleccionar distintos umbrales de notificación (ej. 50%, 75%, 90%). Las notificaciones deben ser no intrusivas y ofrecer opciones para descartarlas o acceder a detalles sobre el consumo | | |
| **Observaciones**   * La notificación debe ser clara y no intrusiva, con opciones para descartarla * La funcionalidad debe ser opcional y configurable desde la interfaz de la aplicación. | | |
| Prototipo de interfaz: | | |

**Tabla 7** Historia de usuario 04 :Notificación de Límite de Datos

|  |  |  |
| --- | --- | --- |
| Historia de usuario | | |
| Número: HU-05 | **Nombre del requisito: Integración con Bandeja del Sistema** | |
| Programador: Carlos Manuel Vargas Piña | | **Iteración asignada:primera** |
| Prioridad: Media | | **Tiempo estimado:**3 días |
| Riesgo en desarrollo: | | **Tiempo real:** |
| Descripción: La aplicación debe integrarse con la bandeja del sistema para proporcionar acceso rápido a sus funcionalidades. Se debe mostrar un ícono en la bandeja que refleje el estado de la conexión mediante colores o símbolos diferenciados (ej. verde para conectado, rojo para desconectado). Esta funcionalidad debe activarse automáticamente al iniciar la aplicación y debe poder desactivarse desde la configuración. Al hacer clic en el ícono, se desplegará un menú con opciones como Conectar/Desconectar, Mostrar la aplicación y Quitar el ícono, permitiendo una gestión rápida y sencilla. | | |
| * La aplicación debe mostrar un ícono en la bandeja del sistema. * El ícono debe cambiar de color o símbolo según el estado de la conexión (verde para conectado, rojo para desconectado). * Al hacer clic en el ícono, debe desplegarse un menú con las opciones():   + Conectar/Desconectar   + Mostrar aplicación   + Cerrar aplicación | | |
| Prototipo de interfaz: | | |

**Tabla 8** Historia de usuario 05:Integración con Bandeja del Sistema

## 2.5 Representación y diseño del sistema

### **2.5.1 Representación de la Arquitectura del sistema**

## La arquitectura, referida al software, es un concepto que surge ya en los años 60 y se refiere a una planificación basada en modelos, patrones y abstracciones teóricas, a la hora de realizar una pieza de software de cierta complejidad y como paso previo a cualquier implementación. De esta forma se dispone de una guía teórica detallada que nos permite entender cómo van a encajar cada una de las piezas de nuestro producto o servicio (Pablo Huet 2022).

El desarrollo de software requiere estructuras bien definidas que faciliten la organización del código, la escalabilidad y el mantenimiento del sistema. Una de las arquitecturas más utilizadas para aplicaciones de escritorio es la arquitectura por capas, un modelo que separa la aplicación en módulos independientes, cada uno con una función específica.

En el contexto del desarrollo de una aplicación para la autenticación en el proxy, la arquitectura por capas proporciona varios beneficios clave. Esta aplicación, que empleará Flet para la interfaz gráfica, Python para la lógica del programa, interacción con Cntlm , manejo de datos y conexión con el sistema operativo Nova , requiere una estructura clara que facilite su desarrollo, mantenimiento y escalabilidad.

El uso de la arquitectura por capas permite:

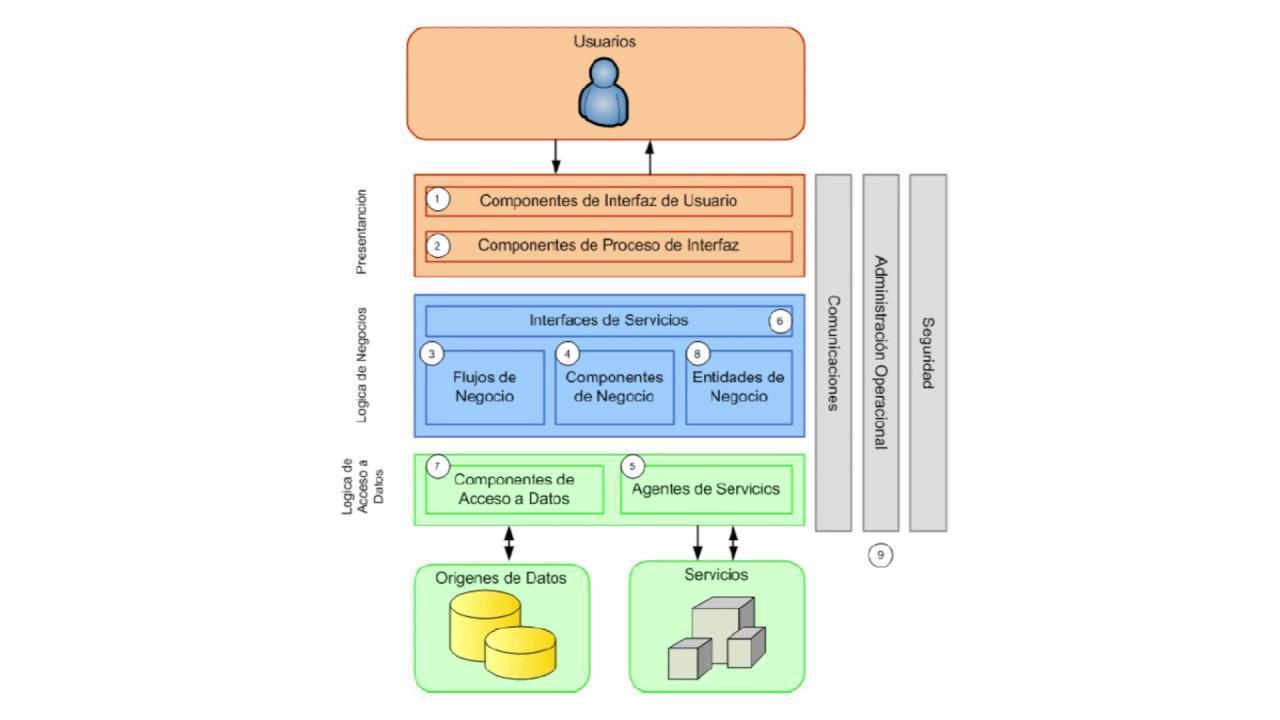
**Modularidad:** Separa la interfaz de usuario, la lógica de autenticación y la integración con el sistema, lo que facilita futuras modificaciones y mejoras sin afectar toda la aplicación.

**Mantenibilidad:** Si es necesario actualizar Cntlm, cambiar la forma en que se almacenan las configuraciones o mejorar la interfaz, los cambios se pueden realizar sin afectar otras partes del sistema.

**Escalabilidad:** Permite agregar nuevas funcionalidades, como notificaciones avanzadas o monitoreo en tiempo real de las aplicaciones que usan el proxy, sin comprometer el funcionamiento general.

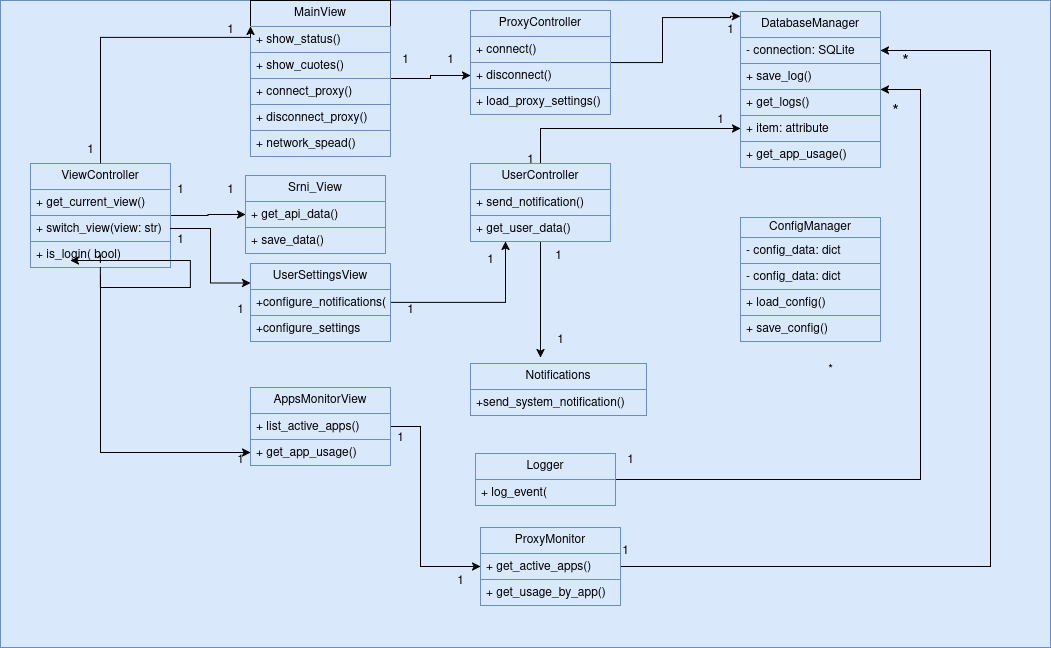
**Seguridad:** Mantiene separada la gestión de credenciales y la lógica de autenticación, reduciendo los riesgos de exposición de datos sensibles.

Esta organización estructurada permitirá desarrollar una solución eficiente, adaptable y fácil de mantener. Además, facilita que, en caso de requerir cambiar la tecnología utilizada en alguna parte específica de la aplicación en el futuro, dicha modificación pueda implementarse sin afectar las demás componentes del sistema.



**Ilustración 6:** Representación Arquitectura por capas

### **2.5.2 Diagrama de clases del diseño**



***Ilustración 7:*** Representación diagrama de clases

### **2.5.3 Patrones de diseño:**

Los patrones de diseño son soluciones generales, reutilizables y probadas que abordan problemas comunes en el diseño de software. Se presentan como plantillas estructuradas que permiten identificar y resolver situaciones recurrentes de manera eficiente. Estas soluciones han sido desarrolladas y refinadas a lo largo del tiempo por la comunidad de desarrolladores a través de la experiencia y la experimentación. (Canelo 2020)

El uso de patrones de diseño ofrece múltiples beneficios, entre ellos:

* Facilitan la resolución de problemas de diseño en el desarrollo de software.
* Se basan en buenas prácticas y pueden aplicarse en diferentes proyectos.
* Mejoran la legibilidad, mantenibilidad y escalabilidad del código.
* Promueven principios esenciales como bajo acoplamiento y alta cohesión.
* Son universales y pueden implementarse en cualquier lenguaje de programación.

En este proyecto de tesis, se han aplicado diversos patrones de diseño para mejorar la arquitectura, modularidad y eficiencia del sistema. Entre los principales patrones utilizados se encuentran:

**GRASP**: es un conjunto de pautas y patrones que ayudan a los desarrolladores a tomar decisiones informadas sobre la asignación de responsabilidades a objetos y clases dentro del diseño orientado a objetos. Su objetivo es crear sistemas de software bien estructurados, mantenibles y flexibles.(Chand 2024)

**Creador:**

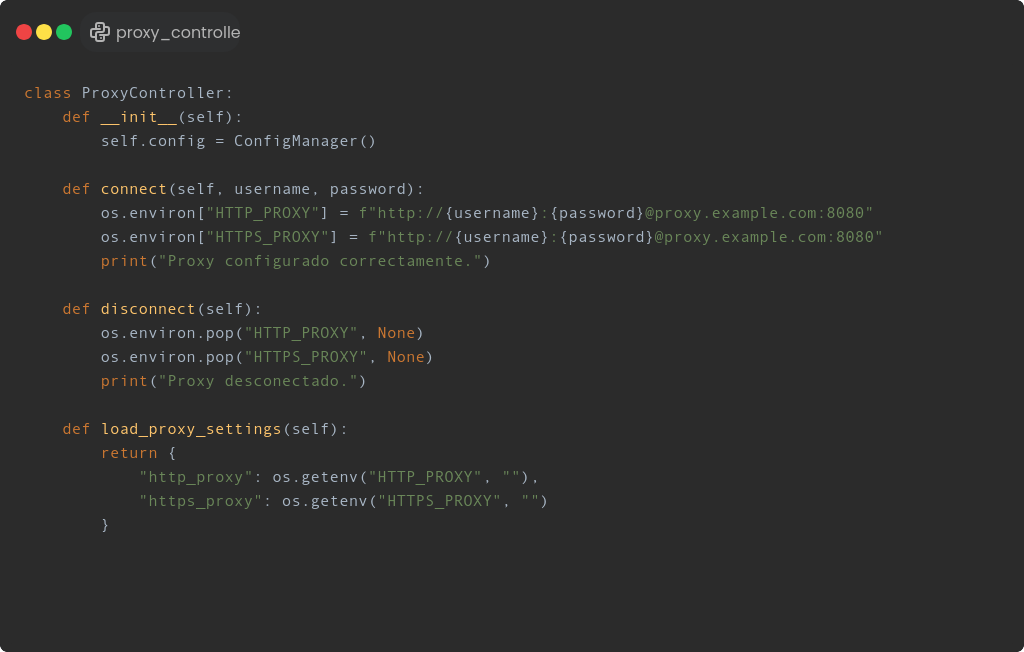
El patrón creador es un tipo de plantilla de patrón de diseño que sirve para resolver tareas de programación en una programación orientada a objetos. Los patrones Builder (o Constructor) facilitan a los desarrolladores el proceso de programación porque no han de rediseñar cada paso que se repite como una rutina de programa («¿Qué es el patrón de diseño Builder?» 2021). La clase ConfigManager es responsable de crear y administrar directamente los objetos de configuración que necesita para operar . Esta asignación es lógica, natural, y mantiene la cohesión y claridad en el diseño del sistema.



**Ilustración 8**:Ejemplo de uso del patrón Creador

**Controlador:**

El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (Rafael Nuñez Manotas 2022). La Clase ProxyController actúa como intermediaria entre la interfaz de usuario y los componentes que manejan la lógica del sistema, como la configuración y el acceso a la base de datos. Recibe las solicitudes del sistema (por ejemplo, conectar o desconectar el proxy).



**Ilustración 9**:Ejemplo de uso del patrón Controlador

## Bajo Acoplamiento:

Sugiere que las clases deben tener la menor dependencia posible entre sí para facilitar el mantenimiento, la reutilización y la comprensión del  **(aqui poner la referencia de low acoplamiento)**. En la aplicaciónI, este principio se aplica de forma efectiva en la clase ViewController, la cual se encarga de gestionar el cambio entre las distintas vistas sin acoplarse directamente a sus implementaciones internas. Gracias a esto, las vistas pueden ser reemplazadas o modificadas de forma independiente sin afectar la lógica de navegación general del sistema.



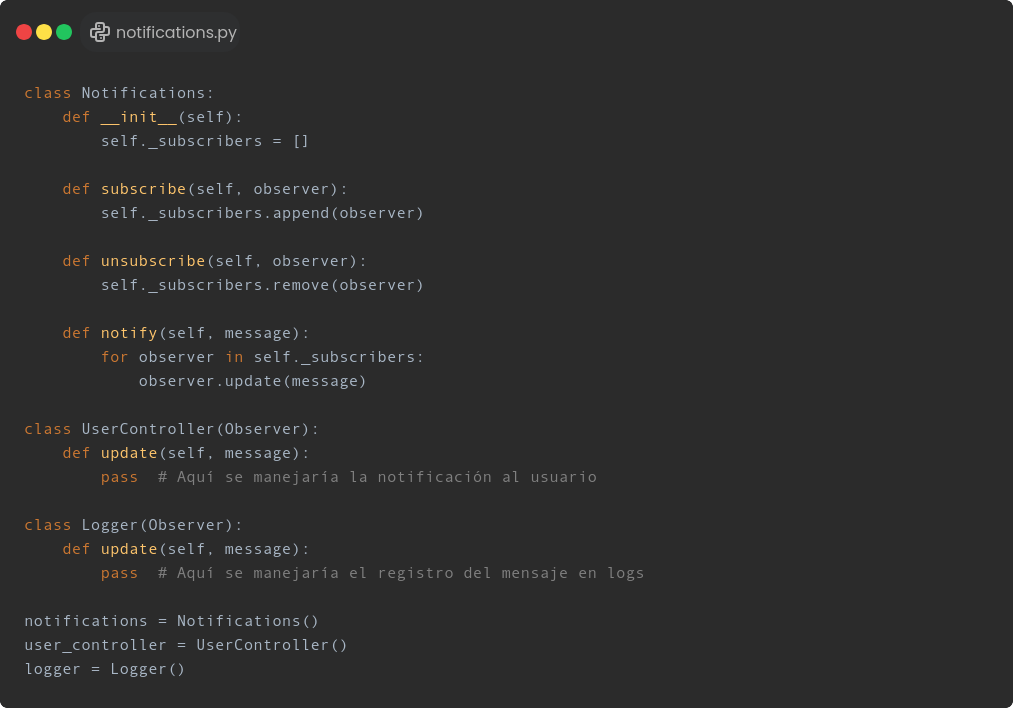
**Ilustración 9**:Ejemplo de uso del patrón Bajo Acoplamiento

## Patrones de diseño GOF:

## El objetivo principal de los patrones es facilitar la reutilización de diseños y arquitecturas software que han tenido éxito capturando la experiencia y haciéndola accesible a los no expertos.

**Observador:**

Es uno de los patrones de diseño de software más populares. Esta herramienta ofrece la posibilidad de definir una dependencia uno a uno entre dos o más objetos para transmitir todos los cambios de un objeto concreto de la forma más sencilla y rápida posible(referencia) . La Clase Notificaions reacciona a eventos generados por otros componentes del sistema. Implementar el patrón Observer permitiría desacoplar la lógica de notificación del resto del sistema, promoviendo la reutilización y mantenimiento del código.



**Ilustración 10**:Ejemplo de uso del patrón Observador:

**Singleton**

Asegura que una clase tenga una única instancia y proporcione un punto de acceso global a ella. Este patrón es especialmente útil cuando se necesita garantizar una configuración centralizada y coherente en toda la aplicación (aquí la referencia de zotero). La clase DatabaseManager garantiza que exista una única instancia de conexión a la base de datos durante toda la ejecución del sistema. Tener múltiples instancias podría generar conflictos de acceso, bloqueos de escritura o inconsistencias en los datos

## 

**Conclusiones del capítulo**

En este capítulo se definieron con mayor precisión las funcionalidades de la aplicación a partir de la captura de requisitos y la elaboración de historias de usuario, lo que permitió estructurar una solución acorde a las necesidades de los usuarios.

El análisis de herramientas similares, junto con las encuestas realizadas, permitió identificar mejoras clave para optimizar tanto el proceso de autenticación como la experiencia de uso. Además, la adopción de una arquitectura por capas y la aplicación de patrones de diseño facilitaron la organización del sistema de forma modular y escalable, lo que contribuirá a su mantenimiento y evolución. La representación del diseño mediante diagramas ayudó a visualizar de manera clara su estructura y funcionamiento.

**Capítulo III: Implementación y validación**

Este capítulo describe el proceso de implementación de la aplicación , que permite la autenticación de usuarios, monitoreo en tiempo real del consumo de datos y control del servicio Cntlm en Nova . Se aplicaron estándares de codificación y buenas prácticas para asegurar una solución eficiente y funcional. Además, se detallan las pruebas realizadas, incluidas las unitarias y de integración para garantizar que el sistema cumpla con los requisitos y brinde una experiencia de usuario óptima.

**3.1 Implementación**

Una vez que las historias de usuario estuvieron claras y el diseño de la aplicación finalizado, se puso en marcha la codificación de la solución. Adoptando un enfoque iterativo e incremental, buscamos entregar versiones funcionales de la aplicación rápidamente, para luego completar los detalles restantes del desarrollo de forma gradual. Esta estrategia se enmarca dentro de la fase de ejecución de la metodología Ágil Unified Process – Universidad de las Ciencias Informáticas (AUP-UCI), adaptada para proyectos internos de la UCI.

**3.2 Estándares de implementación:**

Para asegurar la calidad, consistencia y mantenibilidad del código Python desarrollado, así como facilitar la colaboración dentro del equipo, se definieron los siguientes estándares de codificación, basados en las mejores prácticas y guías de estilo de Python:

* Formato del Código (PEP 8):
* Se seguirá estrictamente el estilo "Pythonic" adhiriéndose a las directrices del PEP 8 para indentación (4 espacios), espaciado, longitud de línea (recomendado 79 caracteres para código, 72 para docstrings/comentarios), importaciones y organización general del código.
* Se utilizará la herramienta Zed y el atajo de teclado ALT+Shift+F (o herramientas equivalentes como Black, Flake8, isort) para mantener un formato limpio, coherente y conforme a PEP 8 de forma automática.
* Comentarios y Docstrings:
* Se usarán comentarios de una sola línea (#) para explicar lógica compleja o decisiones de implementación específicas dentro del código.
* Se utilizarán Docstrings ("""Docstring aquí""") multilínea al inicio de módulos, clases, funciones y métodos para describir su propósito, argumentos, valores de retorno y cualquier excepción que puedan lanzar, siguiendo las convenciones del PEP 257.
* Módulos: Nombres cortos, en minúsculas, preferiblemente sin guiones bajos (module.py).
* Clases: Se empleará UpperCamelCase
* Funciones y Métodos: Se utilizará snake\_case (minúsculas con guiones bajos) (ej. authenticate\_user, validate\_connection).
* Variables: Se utilizará snake\_case (minúsculas con guiones bajos) (ej. user\_name, connection\_status).
* Constantes: Se escribirán en UPPER\_SNAKE\_CASE (mayúsculas con guiones bajos) (ej. MAX\_RETRIES, DEFAULT\_TIMEOUT) y se definirán a nivel de módulo.
* Uso de GitHub Copilot: Se empleará GitHub Copilot como herramienta de asistencia para acelerar la codificación. Sin embargo, toda sugerencia generada será revisada cuidadosamente por el desarrollador para garantizar que cumple con los estándares de calidad, coherencia, corrección y seguridad del proyecto.

**3.3 Interfaz gráfica de usuario**

En el desarrollo de Cuota UCI, se prestó especial atención al diseño de la interfaz de usuario (UI), con el propósito de ofrecer una experiencia fluida, intuitiva y coherente con las necesidades de los usuarios de la Universidad de las Ciencias Informáticas (UCI). La interfaz no solo actúa como puente entre el sistema y el usuario, sino que también juega un papel clave en la eficiencia y aceptación de la herramienta.

Para lograrlo, se adoptaron principios de diseño centrado en el usuario, orientados a maximizar la usabilidad y la comprensión del sistema. Entre los enfoques aplicados, se destacan:

* **Claridad y simplicidad**: La interfaz fue diseñada para presentar la información de forma directa, evitando elementos innecesarios que puedan distraer o dificultar la navegación. El objetivo es que cualquier estudiante o usuario técnico pueda utilizar la aplicación sin curva de aprendizaje.
* **Consistencia visual y funcional**: Se mantuvieron patrones visuales y de interacción homogéneos en todas las vistas haciendo uso del estilo Material Desing. Esto facilita que el usuario identifique fácilmente funciones como iniciar o detener el proxy, cambiar de perfil, o gestionar reglas del cortafuegos.
* **Organización visual eficaz:** Mediante el uso de jerarquías visuales basadas en colores, tamaños de fuente y espaciado, se guiaron los puntos de atención hacia las acciones más importantes: estado del servicio, selección de perfil activo, y trazas más recientes.
* **Retroalimentación inmediata:** Cada acción realizada por el usuario, como activar o restablecer la configuración, genera una respuesta visual inmediata (mensajes, cambios de color, iconos), permitiendo al usuario entender el estado actual del sistema en todo momento.

**3.2 Pruebas de software**

Las pruebas de software son un proceso crucial para verificar que una aplicación cumpla con los requisitos y funcione correctamente. Consisten en ejecutar el sistema bajo condiciones específicas para identificar defectos y asegurar su comportamiento esperado. Estas pruebas pueden realizarse de manera manual o automatizada, y se basan en comparar los resultados obtenidos con los resultados esperados. El objetivo principal es detectar errores, mejorar la calidad del software y garantizar su fiabilidad y seguridad antes de ser liberado al usuario final.

### **3.2.1 Estrategia de Pruebas**

**Objetivos:**

* Validar los artefactos generados en las diferentes etapas del desarrollo de software.
* Asegurar el funcionamiento correcto de cada una de las funcionalidades implementadas en la aplicación.
* Evaluar el comportamiento de la aplicación en cuanto a eficiencia y rendimiento, garantizando que cumpla con los parámetros definidos para el sistema.
* Confirmar que se satisfacen todos los requisitos previamente establecidos.
* Detectar posibles defectos o fallos presentes en el sistema.
* Incrementar la confianza en el nivel de calidad del producto sometido a prueba.

**Tipos y niveles de Pruebas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Prueba** | **Método** | **Herramienta** | **Alcance** |
| Funcionalidad | Caja Negra | N/A | Todas las funcionalidades |
| Seguridad | N/A |  | Se aplica sobre el sistema |
| Rendimiento | N/A | htop | Se prueba el rendimiento y los recursos que consume la aplicación |

**Etapas de Pruebas**

|  |  |  |
| --- | --- | --- |
| **Etapa** | **Tipo de Prueba** | **Objetivo** |
| Vista de Presentación | Funcionalidad | Validar la aceptación del diseño de las interfaces de comunicación del sistema por parte del cliente. |
| Vista Lógica | Unitaria, Seguridad y Rendimiento | Verificar los artefactos que se generan durante el desarrollo de la vista lógica de la aplicación. |

**Responsables**

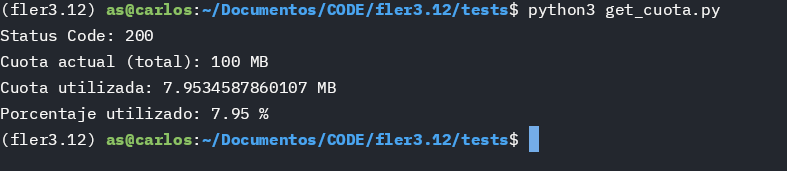
|  |  |  |
| --- | --- | --- |
| **Nombre** | **Rol** | **Responsabilidades** |
| Carlos Manuel Vargas Pina | Probador.Diseñador de prueba. | Responsable de definir y aplicar los métodos de prueba y asegurar una implementación exitosa. |

**3.2.2 Pruebas Unitarias**

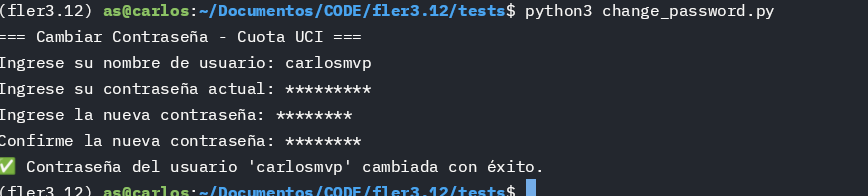
Las pruebas unitarias son el proceso en el que se prueba la unidad funcional de código más pequeña. Las pruebas de software ayudan a garantizar la calidad del código y son una parte integral del desarrollo de software. Una práctica recomendada en el desarrollo de software es escribir el software como unidades pequeñas y funcionales, y luego escribir una prueba unitaria para cada unidad de código. Puede escribir primero pruebas unitarias como código. Luego, ejecute ese código de prueba de forma automática cada vez que realice cambios en el código del software. De esta forma, si una prueba falla, puede aislar con rapidez el área del código que tiene el error. Las pruebas unitarias imponen paradigmas de pensamiento modular y mejoran la cobertura y calidad de las pruebas. («¿Qué son las pruebas unitarias?: explicación de las pruebas unitarias en AWS», 2023.)

En la figura se muestra el método correspondiente al RF-18 cambiar contraseña y el RF-6 monitoreo de consumo.

**Monitoreo de Consumo**



**Cambiar Contraseña**



**3.2.3 Prueba de validación;**

**HU-01**

# Casos de Prueba - Autenticación de Usuario

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Escenario** | **Descripción** | **Nombre de usuario** | **Contraseña** | **Respuesta de la aplicación** | **Flujo central** |
| **EC1 Autenticación válida (HU-1)** | El usuario introduce el nombre de usuario y contraseña para autenticarse en la aplicación. | carlosmvp | ContraseñaPrueba.\*3 | La aplicación redirige al dashboard. | 1. El usuario ingresa nombre de usuario y contraseña y presiona "Iniciar sesión".  2. La aplicación realiza una petición al servidor para verificar las credenciales.  3. Si son válidas, redirige a la vista principal. |
| **EC2 Credenciales inválidas (HU-1)** | El usuario ingresa credenciales incorrectas. | username | Test\_error\_password | La aplicación muestra el mensaje: "Credenciales inválidas". | 1. El usuario ingresa nombre de usuario y contraseña y presiona "Iniciar sesión".  2. La aplicación hace una petición al servidor para verificar las credenciales.  3. Si no son válidas, muestra el mensaje de error. |
| **EC3 Credenciales vacías (HU-1)** | El usuario deja vacíos los campos de usuario y contraseña. | vacío | vacío | La aplicación muestra el botón de "Iniciar sesión" deshabilitado. | — |
| **EC4 Error de red (HU-1)** | El usuario ingresa su usuario y contraseña y presiona "Iniciar sesión". | carlosmvp | Contraseña | La aplicación muestra un mensaje de error indicando que no está conectado a la red UCI. | 1. La aplicación intenta hacer una petición al servidor de la UCI. 2. Detecta que no está conectado a la red UCI. 3. Muestra el mensaje de error. |

**Prueba de Iniciar proxy**

**— por hacer**

**REFERENCIAS BIBLIOGRÁFICAS**

* «407 Se requiere autenticación de proxy - HTTP | MDN». 2024. 24 de julio de 2024. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/407>.
* Abstracta Team. 2024. «Metodologías de desarrollo de software: guía completa | Abstracta». 8 de noviembre de 2024. <https://es.abstracta.us/blog/metodologias-de-desarrollo-de-software-guia-completa/>.
* Ana Bescós Domínguez. 2019. «Desarrollo de software: importancia y técnicas de la captura de requisitos». 7 de septiembre de 2019. <https://www.knowmadmood.com/blog/desarrollo-de-software-importancia-y-tcnicas-de-la-captura-de-requisitos>.
* «Autenticación proxy». s. f. Accedido 24 de febrero de 2025. <https://es.proxyscrape.com/blog/autenticaci%C3%B3n-proxy>.
* Brave, Little Miss. 2024. «Cntlm Basics: A Rookie’s Handbook for Enterprise Proxy Setup». Medium (blog). 14 de junio de 2024. [https://medium.com/@papapapaya11/Cntlm-basics-a-rookies-handbook-for-enterprise-proxy-setup-3c1d145b0a52](https://medium.com/@papapapaya11/cntlm-basics-a-rookies-handbook-for-enterprise-proxy-setup-3c1d145b0a52).
* Canelo, Miriam Martínez. 2020. «Qué son los Patrones de Diseño de software / Design Patterns». Profile Software Services (blog). 24 de junio de 2020. <https://profile.es/blog/patrones-de-diseno-de-software/>.
* Chand, Mehar. 2024. «GRASP (General Responsibility Assignment Software Patterns)». Medium (blog). 6 de enero de 2024. <https://medium.com/@mehar.chand.cloud/grasp-general-responsibility-assignment-software-patterns-7d1104b0aad5>.
* Cloudflare. s. f. «¿Qué es HTTP?» Accedido 27 de febrero de 2025. <https://www.cloudflare.com/es-es/learning/ddos/glossary/hypertext-transfer-protocol-http/>.
* «Cntlm: Fast NTLM Authentication Proxy in C». s. f. Accedido 24 de febrero de 2025. [https://Cntlm.sourceforge.net/?utm\_source=chatgpt.com](https://cntlm.sourceforge.net/?utm_source=chatgpt.com).
* Codemotion. 2024. «Excalidraw: Diagramas forjados en piedra». Codemotion Magazine. 16 de julio de 2024. <https://www.codemotion.com/magazine/es/open-source-es/excalidraw-una-introduccion-completa-y-resena/>.
* Escobar, Nasheli. 2015. «Qué es el superusuario en Linux y cuál es su importancia». Hipertextual. 7 de octubre de 2015. <http://hipertextual.com/2015/10/superusuario-en-linux>.
* Fernández, Yúbal. 2024. «Qué es un proxy y cómo puedes utilizarlo para navegar de forma más anónima». Xataka. 2 de julio de 2024. <https://www.xataka.com/basics/que-es-un-proxy-y-como-puedes-utilizarlo-para-navegar-de-forma-mas-anonima>.
* «Flet: Power of your favourite language + Flutter UI». 2023. Codemonk Blog. 18 de mayo de 2023. <https://codemonk.io/blog/flet-power-of-your-favorite-language-flutter-ui/>.
* «Git». s. f. Accedido 24 de febrero de 2025. <https://git--scm-com.translate.goog/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc>.
* Jorge Vázquez. 2020. «Draw.io: mucho más que mapas mentales». INTEF. 8 de junio de 2020. <https://intef.es/observatorio_tecno/draw-io-mucho-mas-que-mapas-mentales/>.
* «Metodologías Para El Desarrollo de Software – Prolosys». s. f. Accedido 26 de febrero de 2025. <https://prolosys.com.mx/metodologias-para-el-desarrollo-de-software/>.
* Pablo Huet. 2022. «Arquitectura de software: Qué es y qué tipos existen | OpenWebinars». OpenWebinars.net. 24 de agosto de 2022. <https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/>.
* «Plyer 2.2.0.dev0 documentation». s. f. Accedido 24 de febrero de 2025. <https://plyer.readthedocs.io/en/latest/?utm_source=chatgpt.com>.
* ProxyScrape. 2022. «Autenticación proxy - 2 formas únicas». 19 de febrero de 2022. <https://es.proxyscrape.com/blog/autenticaci%C3%B3n-proxy>.
* «Python». 2025. En Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=Python&oldid=165639624>.
* «¿Qué es el patrón de diseño Builder?» 2021. IONOS Digital Guide. 22 de febrero de 2021. <https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/patron-de-diseno-builder/>.
* Rafael Nuñez Manotas. 2022. «Patrón controlador ~ Buenas Practicas de Programación». 8 de mayo de 2022. <http://buenaspracticasupc.blogspot.com/2016/05/patron-controlador.html>.
* Rodriguez, Danel. s. f. «LocalProxy». IzzyOnDroid. Accedido 26 de febrero de 2025. <https://android.izzysoft.de/repo/apk/uci.localproxy>.
* Santander Universidades. 2020. «Metodologías de desarrollo de software: ¿qué son?» 21 de diciembre de 2020. <https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html>.
* Solutions, Visure. 2021. «Definición de requisitos: ¿Qué es y cómo aplicarlo? | Guía Completa - Soluciones Visure». Visure Solutions (blog). 8 de febrero de 2021. <https://visuresolutions.com/es/blog/definici%C3%B3n-de-requisitos/>.
* Tamara, R.S. 2015. «Metodología de Desarrollo de Software Variación de AUP para la UCI - EcuRed». 4 de junio de 2015. <https://www.ecured.cu/Metodolog%C3%ADa_de_Desarrollo_de_Software_Variaci%C3%B3n_de_AUP_para_la_UCI>.
* «Variables de entorno de proxy: todo curl». s. f. Accedido 27 de febrero de 2025. <https://everything-curl-dev.translate.goog/usingcurl/proxies/env.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc>.
* «What Is Kerberos and How Does It Work? | Ping Identity». s. f. Accedido 25 de febrero de 2025. <https://www.pingidentity.com/en/resources/identity-fundamentals/authentication-authorization-protocols/kerberos.html>.
* «Zed - The Editor for What’s Next». s. f. Zed. Accedido 24 de febrero de 2025. [https://www.zed.dev](https://www.zed.dev/).

# ANEXOS

<Contenido de los anexos con igual tipo de fuente Arial, pero a tamaño 11 puntos e interlineado 1.0 puntos. Debe tratar de sólo utilizarse aquellos anexos imprescindibles para complementar lo presentado en la memoria escrita y que no excedan las ocho (8) o diez (10 páginas). Deben aparecer uno a continuación del otro sin necesidad de saltos de página entre estos>