

**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**: **MSc.** Daniel José Olazabal Guerra.

**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 internet. El objetivo es desarrollar una herramienta integrada que optimice este proceso, mejorando la experiencia del usuario y la seguridad. 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**](#_hfbq8kkj2gnc) **10**

[**CAPÍTULO I: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio**](#_96b41zk5gq25) **15**

[1.1 Conceptos Fundamentales:](#_rt19dzxel767) 15

[1.2 Funcionamiento de un servidor proxy](#_sdxjr8rwd7b6) 16

[1.2.2 Métodos de autenticación en sistemas operativos Linux](#_r1s4l8vq2oic) 17

[1.3 Análisis de alternativas comparables](#_z6aqletl4710) 18

[1.3.2 Px](#_v2hhgxsr20p2) 18

[1.3.3 Proxifier](#_wioz37bw5elf) 18

[1.3.4 Cuota-UCI](#_lm23nua47t5k) 19

[1.3.5 UCIntlm](#_4vml7p1v9v3h) 19

[1.3.6 LocalProxy](#_z9cn25ki6oc7) 19

[1.3.7 Consideraciones finales sobre las soluciones estudiadas](#_r724v1jecamp) 19

[1.4 Tecnologías y herramientas](#_kttjcxn63561) 21

[1.4.1 Herramientas](#_izly3v6cc75l) 21

[1.4.2 Tecnologías](#_jw7yv48fej0c) 22

[1.5 Metodologías de desarrollo de software:](#_55cy4u2ww73w) 23

[Conclusiones del capítulo](#_ut5y8ulddd9s) 27

[**CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO**](#_tzx5jzw581v7) **28**

[2.1 Propuesta técnica](#_iex46e99t0qg) 28

[2.2 Captura de requisitos](#_lghwpom1a6gr) 28

[2.2.1 Técnicas para la captura de requisitos](#_1gmjxrrkhyig) 29

[2.3 Requisitos de software](#_737cudwo8hwk) 32

[2.3.1 Requisitos funcionales](#_b8ofnp7b7hkp) 33

[2.3.2 Requisitos No Funcionales](#_4rwnnhdpwk2q) 37

[2.4 Historias de usuario](#_gpnm1qgs8k75) 38

[2.5 Representación y diseño del sistema](#_r68roxcxtd6p) 45

[2.5.1 Representación de la Arquitectura del sistema](#_49xd2lly0git) 45

[2.5.2 Diagrama de clases del diseño](#_9ixbpgef26b2) 47

[2.5.3 Patrones de diseño:](#_x0vzizexo2x) 47

[Conclusiones del capítulo](#_kiey2edsqb6g) 51

[**REFERENCIAS BIBLIOGRÁFICAS**](#_t7l72sgt8ylq) **52**

[**ANEXOS**](#_wwqs17e7m5pi) **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 optimice 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.

## 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.(«Autenticación proxy», s. f) 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 autenticaciónn 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.

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

|  |  |
| --- | --- |
| **Esquema de autenticación** | **Descripción** |
| Básica | 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», s. f.) |
| 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 («What is NTLM & How Does NTLM Authentication Work», s. f.). |
| 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», s. f.). |
| Bearer |  |

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 con especialistas**: Se realizaron entrevistas a especiliastas del centro Cisol , ex en 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 documentacion y tesis de prgrado sobre el funcionamiento de este.
4. **Analisis de Homolofos**
5. 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. Fulano y m se descaraton por su incobatibilidad con el servidor proxy de la universidad .

## 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).

### **1.3.2 Proxifier**

https://www.proxifier.com/

Es una herramienta que permite que las aplicaciones de red que no admiten el funcionamiento a través de servidores proxy lo hagan, por lo general, mediante un proxy HTTP/S o SOCKS. Básicamente, obliga a todas las aplicaciones, independientemente de si cuentan con soporte o no, a utilizar servidores proxy. Al hacerlo, se puede permitir que estas aplicaciones funcionen incluso si se requiere un servidor proxy para acceder a ellas (Zafiriadi, 2023).

### **1.3.3 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)

### **1.3.4 UCIntlm**

Permite utilizar Internet en la empresa autenticando el dispositivo en el servidor proxy de la escuela o compañía (Orozco, 2015; Akiel, 2024).

### **1.3.5 LocalProxy**

proporciona un servicio de proxy local en el dispositivo 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, s. f.)

**A continuación, se presenta una tabla comparativa que resume las principales características de las herramientas anteriormente descritas.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aplicaciones** |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

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

Como resultado del análisis realizado, se presenta a continuación un resumen comparativo de las aplicaciones estudiadas. En la Tabla 2 se detallan las principales características de cada una, destacando aspectos como el soporte para los protocolos Basic, Digest y aquellos correspondientes a la suite NTLM. Además, se incluye información sobre la necesidad de contar con permisos de administrador para su funcionamiento, así como la disponibilidad de su código fuente. Esta comparación permite evaluar de manera más clara las diferencias y similitudes entre las soluciones analizadas, proporcionando una visión integral sobre sus capacidades y limitaciones.

**Tabla 2:** Comparación entre las soluciones estudiadas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Características | ProxyCap | Px-Proxy | Proxifier | Cuota-UCI | UCIntlm | LocalProxy |
| Protocolos  Soportados | HTTP, HTTPS, SOCKS4, SOCKS5 | NTLM, Kerberos | HTTP/S, SOCKS | NTLM, | NTLM | NTLM, |
| Licencia | Comercial. | MIT | BSD-3-Clause | N/A | N/A | N/A |
| Exportación del proxy al sistema | Si | SI | Si | Si | Si | No especificado. |
| Interfaz  intuitiva | Alta | Moderada | Alta | Moderada | Funciona | Si |
| Código Abierto | No | SI | Si | Si | Si | Si |
| Soporte de plataformas | Windows, macOS. | Windows. | Windows, macOS, Linux. | Windows | Android | Android |
| Documentación disponible | Moderada. | Alta. | Moderada. | No | No | No |

Tras el análisis de las herramientas estudiadas, se concluye que no existe una solución que cumpla con todos los requisitos necesarios para la autenticación con servidor proxy web en dispositivos con la distribución GNU/Linux, como Nova. Aunque aplicaciones como Px-Proxy y Proxifier soportan NTLM, su falta de código abierto, interfaz gráfica poco intuitiva y compatibilidad limitada con diversas plataformas las hacen poco viables. Por otro lado, UCIntlm, a pesar de ser desarrollada por la universidad, solo está disponible para Android y presenta limitaciones en cuanto a documentación y soporte en entornos Linux, lo que dificulta su reutilización eficiente sin mejoras sustanciales.

En este contexto, la aplicación Cuota-UCi, aunque presenta algunas deficiencias en su documentación, destaca por ser un producto universitario, lo que ofrece la oportunidad de contribuir a su mejora. Al ser de código abierto, permite revisar su código y analizar cómo se comporta al conectarse a la red. A pesar de ejecutarse en un sistema operativo distinto a Linux, presenta comportamientos similares en muchos casos, especialmente al interactuar con la red, lo que la convierte en un referente útil para una posible adaptación en el contexto de la Universidad de las Ciencias Informáticas.

Dada la falta de una solución de código abierto adecuada para distribuciones Linux como Nova, que permita la autenticación mediante protocolos comunes como NTLM, se propone crear una herramienta a medida para la universidad. Esta aplicación, integrada al sistema de autenticación del proxy, optimizará el acceso a Internet y mejorará la experiencia del usuario, con funcionalidades como ejecución automática al iniciar el sistema, monitoreo en tiempo real del consumo de datos, notificaciones y una interfaz integrada. Como solución personalizada, se garantizará su adaptabilidad, mejora continua y contribución significativa a la infraestructura de redes de la universidad, optimizando el acceso a Internet para los usuarios de Linux Nova. La aplicación Cuota-UCI servirá como referencia, al ser el producto más similar y compatible en este contexto.

## 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», s. f.)

**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)(Jorge Vázquez 2020)

### **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», s. f.)

**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», s. f.). 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», s. f.)

**SQLite 3.45.0:** es un sistema de gestión de bases de datos relacional (RDBMS) ligero, autónomo y sin servidor. A diferencia de otras bases de datos como MySQL o PostgreSQL, SQLite no requiere un proceso de servidor separado para funcionar. En su lugar, la base de datos se almacena en un único archivo en el sistema, lo que la hace ideal para aplicaciones pequeñas, embebidas o móviles.

**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 y mejorándolo. Lo que hace este asistente es sugerirte código para seguir completando un programa que estés desarrollando.

## 1.5 Metodologías de desarrollo de software:

El desarrollo de software enfrenta una serie de desafíos que, en ocasiones, son difíciles de gestionar y pueden impactar el éxito de un proyecto. Para mitigar estos problemas, han surgido diversas metodologías de desarrollo y estándares que buscan optimizar y estructurar el proceso de creación de software. Estas metodologías proporcionan enfoques definidos y prácticas comprobadas que ayudan a manejar el flujo de trabajo, garantizar la calidad del producto final y mejorar la eficiencia durante todo el ciclo de vida del proyecto.

Las metodologías de desarrollo de software son enfoques que ofrecen un marco claro para planificar, ejecutar y entregar proyectos de software. Estas pueden ir desde enfoques lineales hasta los más iterativos, ajustándose a las necesidades de cada proyecto y equipo, e incluso teniendo en cuenta la cultura organizacional.(Abstracta Team 2024).

A pesar de la gran variedad de metodologías existentes, no se puede afirmar que haya una mejor que otra, ya que todas tienen características específicas que responden a contextos distintos. La elección depende de factores como los requisitos del cliente, los plazos, el equipo de trabajo y la naturaleza del proyecto. Aunque sus diferencias son sutiles, el objetivo común es siempre entregar un producto que cumpla con los criterios de calidad, eficiencia y usabilidad.(«Metodologías Para El Desarrollo de Software – Prolosys», s. f.)

**La Adaptación de AUP en la UCI**

La Universidad de las Ciencias Informáticas (UCI) ha adaptado la metodología AUP (Proceso Ágil Unificado) para ajustarse mejor a las necesidades de sus proyectos. La versión personalizada de AUP, conocida como AUP-UCI, conserva la fase de Inicio, aunque con un enfoque modificado respecto a su propósito original. Además, las fases de Elaboración, Construcción y Transición del AUP original se fusionan en una única etapa denominada Ejecución, mientras que se incorpora una nueva fase llamada Cierre. (Tamara 2015)

**Fases de la Variación AUP para la UCI:**

**Inicio:** En esta fase se planifica el proyecto, se define el alcance, y se estiman los recursos y tiempos necesarios, evaluando la viabilidad del proyecto.

**Ejecución:** Se desarrolla el software, ajustando los planes según los requisitos y la arquitectura definida. Incluye la implementación del modelo de negocio, diseño, pruebas, liberación del producto, y entrega de documentación y capacitación al usuario.

**Cierre:** Se evalúan los resultados del proyecto, asegurando la correcta finalización y validación de los entregables, y se realizan actividades formales de conclusión.

AUP propone 7 disciplinas (Modelado de Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas, Despliegue, y Configuración y Gestión de Cambios). Para el ciclo de vida de los proyectos en la UCI, se adoptan estas mismas disciplinas, pero a un nivel más detallado y específico que el definido originalmente en AUP, adaptándose así a las necesidades particulares de la universidad(Sánchez, 2015).

La disciplina de Requisitos dentro de AUP-UCI se enfoca en la creación de un modelo del sistema a desarrollar, abordando tanto los requisitos funcionales como los no funcionales. Para esto, se utilizan tres enfoques principales:

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

Estos enfoques se organizan en cuatro escenarios, los cuales dependen del tipo de modelado de negocio utilizado en el proyecto:

* Si el negocio se modela mediante Casos de Uso del Negocio (CUN), se utiliza CUS.
* Si el negocio se modela con un Modelo Conceptual (MC), se usa CUS.
* Si el negocio se describe a través de una Descripción de Proceso de Negocio (DPN), se usa DRP.
* En proyectos sin un modelado de negocio formal, se utiliza HU.

Estos enfoques permiten adaptar la metodología de requisitos a las necesidades específicas de cada proyecto, mejorando la flexibilidad y la alineación con los requisitos del cliente

Para la investigación en curso, se ha optado por implementar la metodología AUP-UCI, específicamente en su escenario 4, que destaca por la participación activa y continua del cliente. Esta elección responde a la naturaleza colaborativa del proyecto, donde la interacción constante entre el cliente y el equipo de desarrollo es crucial para definir y validar los requisitos del sistema, garantizando que se alineen con las necesidades y expectativas reales.

Este enfoque, que utiliza Historias de Usuario (HU) para modelar el sistema, resulta particularmente adecuado para proyectos donde ya se ha realizado un análisis exhaustivo del negocio a informatizar, logrando una definición clara y precisa del mismo. Además, es especialmente útil en proyectos de alcance limitado, ya que las Historias de Usuario deben ser concisas, evitando la inclusión de información innecesaria.

## Conclusiones del capítulo

Luego del análisis realizado en el presente capítulo, es posible afirmar que el estudio de los protocolos de autenticación en servidores proxy, los métodos de autenticación en sistemas GNU/Linux y las herramientas existentes permitió adquirir el conocimiento teórico necesario para el desarrollo de una solución adecuada a las necesidades de la Universidad de las Ciencias Informáticas (UCI). Además, el análisis comparativo de las soluciones existentes evidenció que ninguna de ellas satisface completamente los requerimientos de autenticación para los usuarios de GNU/Linux Nova, debido a limitaciones en compatibilidad, disponibilidad del código fuente y falta de integración con el entorno institucional. Por último, la selección de tecnologías específicas y la adopción de la metodología de desarrollo AUP-UCI, basada en Historias de Usuario, permitió establecer un marco de trabajo óptimo para la implementación de una solución personalizada que garantice seguridad, eficiencia y facilidad de uso en la autenticación con servidores proxy de la universidad.

# 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.
* Se integrará con una API del sitio "srniplus" que permitirá a los usuarios consultar qué sitios y direcciones IP han consumido recursos asociados a su cuenta, otorgándoles mayor visibilidad y control.

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 reconectar, 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 Informatizacion , 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, tales como ProxyCap, Px-Proxy, Proxifier, Cuota-UCI, UCIntlm y LocalProxy. 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, contribuyendo a la definición de requisitos específicos como la compatibilidad con GNU/Linux Nova (RNF11) y la necesidad de una interfaz gráfica intuitiva (RNF12).

## **Tormenta de ideas:**

## Al inicio del proyecto, se empleó la técnica de tormenta de ideas como mecanismo grupal para generar una visión preliminar de las necesidades del sistema. En estas sesiones participaron usu especialistas del Centro CESOL, quienes expusieron libremente sus ideas y sugerencias respecto a las limitaciones del sistema actual de autenticación y posibles soluciones. Esta técnica resultó especialmente útil para identificar funcionalidades clave, como la automatización del proceso de inicio de sesión y la compatibilidad con GNU/Linux Nova.

## 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)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nombre** | **Descripción** | **Complejidad** | **Prioridad** |
| **RF-01** | autenticación de usuario | Permitir a los usuarios autenticarse en la aplicación utilizando sus credenciales institucionales. | Alta | Alta |
| **RF-02** | Mantener sesión iniciada de forma segura | Permitir mantener la sesión iniciada de manera segura para evitar autenticaciones manuales repetitivas. | Alta | Alta |
| **RF-03** | Permitir cambio de contraseña | Permitir al usuario cambiar la contraseña de su cuenta institucional desde la aplicación. | Alta | Alta |
| **RF-04** | Integrar acceso desde bandeja del sistema | Proporcionar un acceso rápido a la aplicación mediante un ícono en la bandeja del sistema. | Alta | Alta |
| **RF-05** | Monitorear consumo de datos en tiempo real | Permitir visualizar el consumo de datos del usuario en tiempo real. | Alta | Alta |
| **RF-06** | Visualizar historial de direcciones IP conectadas | Mostrar las direcciones IP internas desde donde se ha usado la cuenta del usuario para conectarse. | Alta | Alta |
| **RF-07** | Monitorear aplicaciones conectadas al proxy | Mostrar en tiempo real las aplicaciones o procesos que están utilizando la conexión al proxy, incluyendo nombre y estado de conexión. | Alta | Alta |
| **RF-08** | Controlar servicio CNTLM | Permitir al usuario iniciar, detener y reiniciar el servicio CNTLM directamente desde la aplicación. | Alta | Alta |
| **RF-09** | Ejecutar al iniciar el sistema | Ejecutar automáticamente la aplicación al arrancar el sistema operativo. | Media | Alta |
| **RF-10** | Mostrar estado de conexión | Mostrar el estado actual de la conexión al proxy (conectado, desconectado o error de autenticación). | Media | Alta |
| **RF-11** | Notificar límite de datos alcanzado | Enviar notificaciones cuando el usuario alcance un porcentaje específico de su cuota de datos. | Media | Alta |
| **RF-12** | Visualizar historial de sitios visitados | Mostrar un historial de los sitios web a los que el usuario se ha conectado usando su cuenta. | Media | Media |
| **RF-13** | Configuración manual del proxy | Permitir modificar manualmente los parámetros de configuración del proxy. | Media | Media |
| **RF-14** | Mostrar velocidad de conexión de 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 | Baja |
| **RF-16** | Personalizar notificaciones | Permitir al usuario personalizar el tipo y frecuencia de las notificaciones de la aplicación. | Media | Baja |

### **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, s. f.)

### **Seguridad**

* **RNF1:** Las contraseñas de los usuarios deben almacenarse cifradas en la base de datos.
* **RNF2:** La aplicación debe seguir el principio de mínimo privilegio, asegurando que solo los procesos estrictamente necesarios tengan acceso a las credenciales y configuraciones del proxy.

### **Rendimiento y Eficiencia**

* **RNF3:** La autenticación debe completarse en menos de 3 segundos en condiciones normales de conexión.
* **RNF4:** El consumo de memoria ram de la aplicación no debe exceder los 70 MB durante su funcionamiento.

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

* **RNF5:** La interfaz de usuario debe estar diseñada siguiendo los principios de Material Design.
* **RNF6:** Cada acción del usuario debe generar retroalimentación inmediata mediante notificaciones o mensajes visuales.
* **RNF7**: 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:** Estado de conexión | |
| Programador: Carlos Manuel Vargas Piña | | **Iteración asignada:1** |
| Prioridad: Alta | | **Tiempo estimado:** 2 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 Si la conexión se pierde o el servidor proxy no responde, se actualizará inmediatamente la interfaz para notificar al usuario | | |
| 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 | | |
| * El usuario debe poder configurar el porcentaje en el que desea recibir la notificación * La notificación debe ser clara y no intrusiva, con opciones para descartarla o acceder al monitoreo detallado del consumo. * Debe incluirse una alerta final cuando la cuota esté a punto de agotarse. * 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, Cntlm para la autenticación y variables de entorno para gestionar la conexión del sistema con el proxy, requiere una estructura clara que facilite su desarrollo, mantenimiento y escalabilidad.

El uso de la **arquitectura por capas** permite:

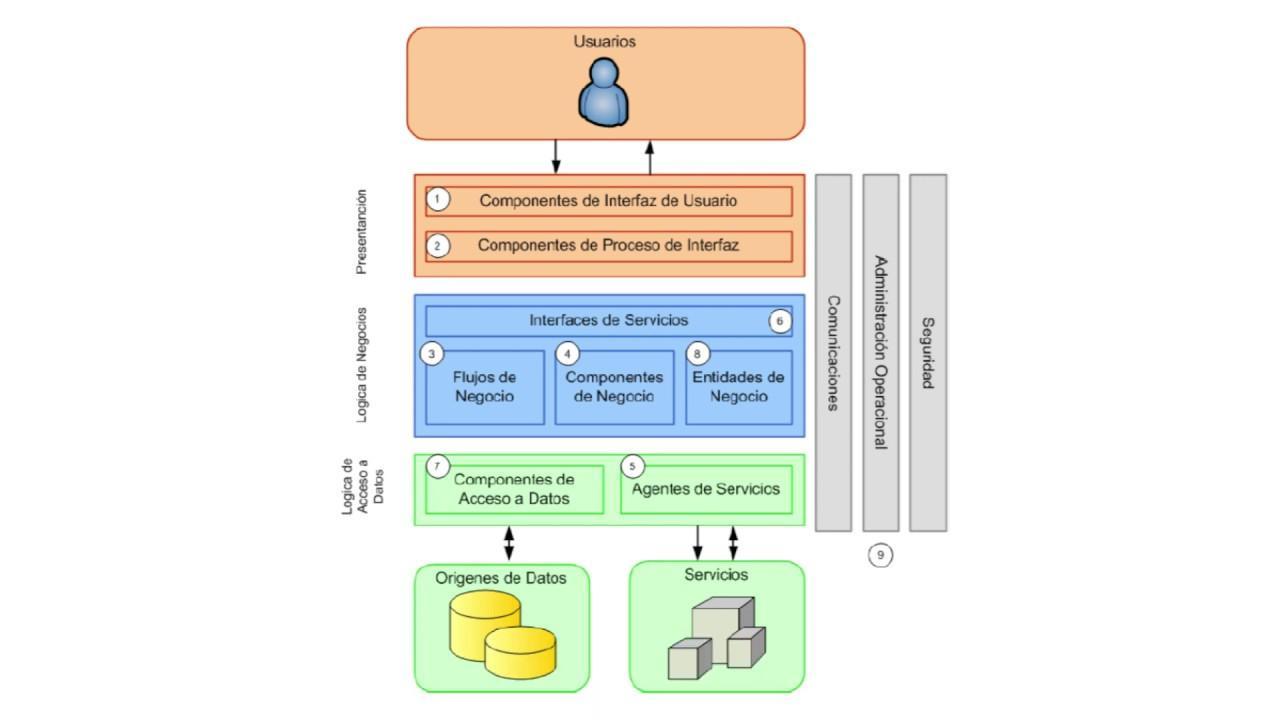
**Modularidad:** Separa la interfaz de usuario, la lógica de autenticación y la integración con el sistema, lo que facilita futuras modificaciones y mejoras sin afectar toda la aplicación.

**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.

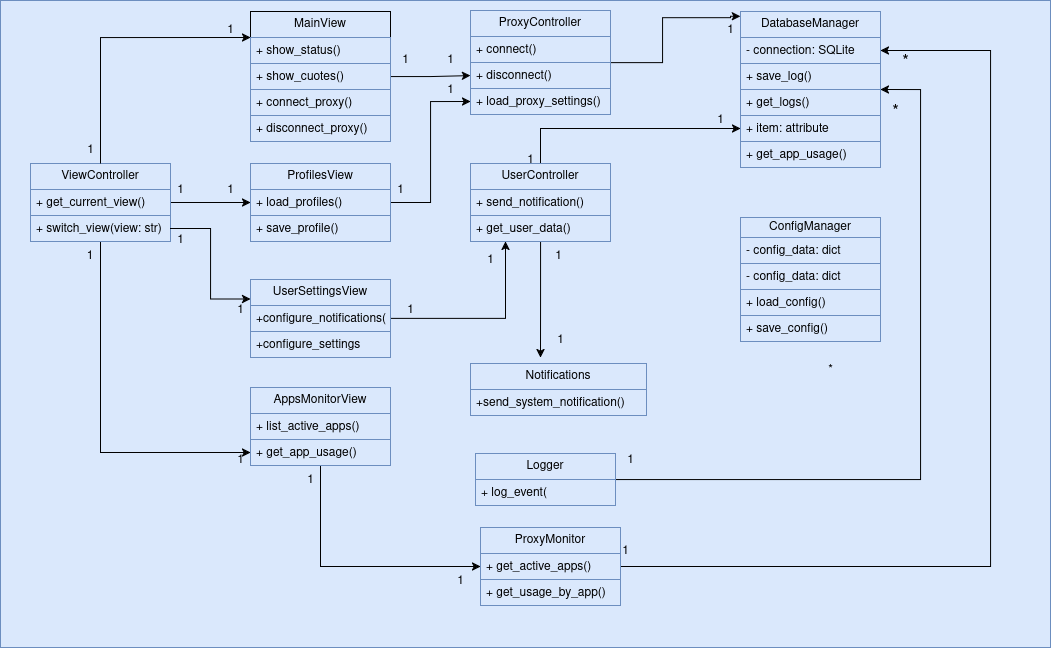
**Seguridad:** Mantiene separada la gestión de credenciales y la lógica de autenticación, reduciendo los riesgos de exposición de datos sensibles.

Esta organización estructurada permitirá desarrollar una solución eficiente, adaptable y fácil de mantener.



**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:**

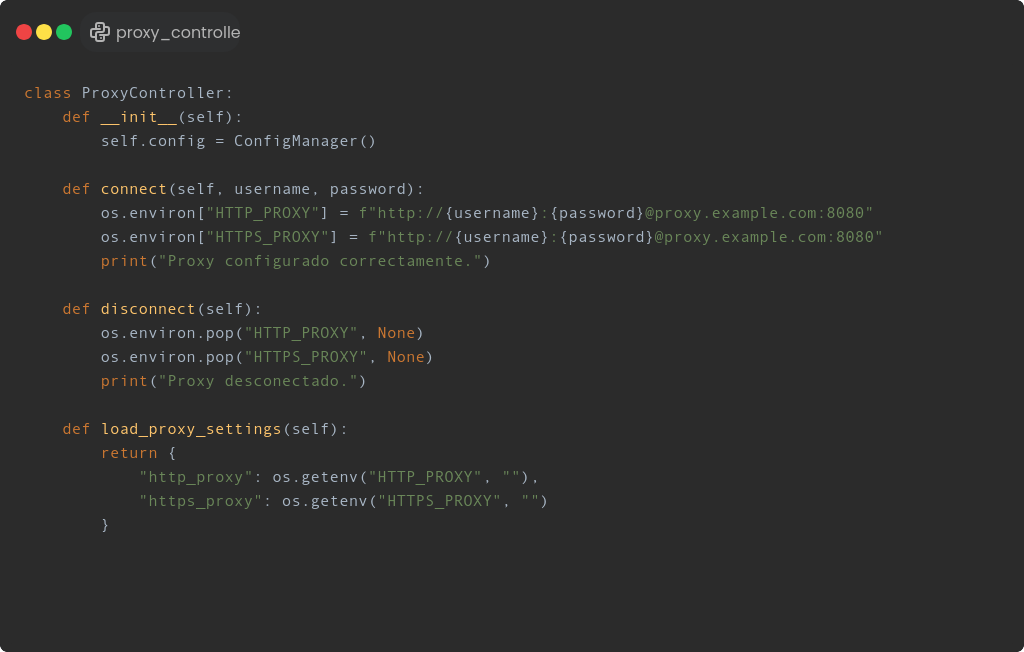
El patrón creador es un tipo de plantilla de patrón de diseño que sirve para resolver tareas de programación en una programación orientada a objetos. Los patrones Builder (o Constructor) facilitan a los desarrolladores el proceso de programación porque no han de rediseñar cada paso que se repite como una rutina de programa. («¿Qué es el patrón de diseño Builder?» 2021)



**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)



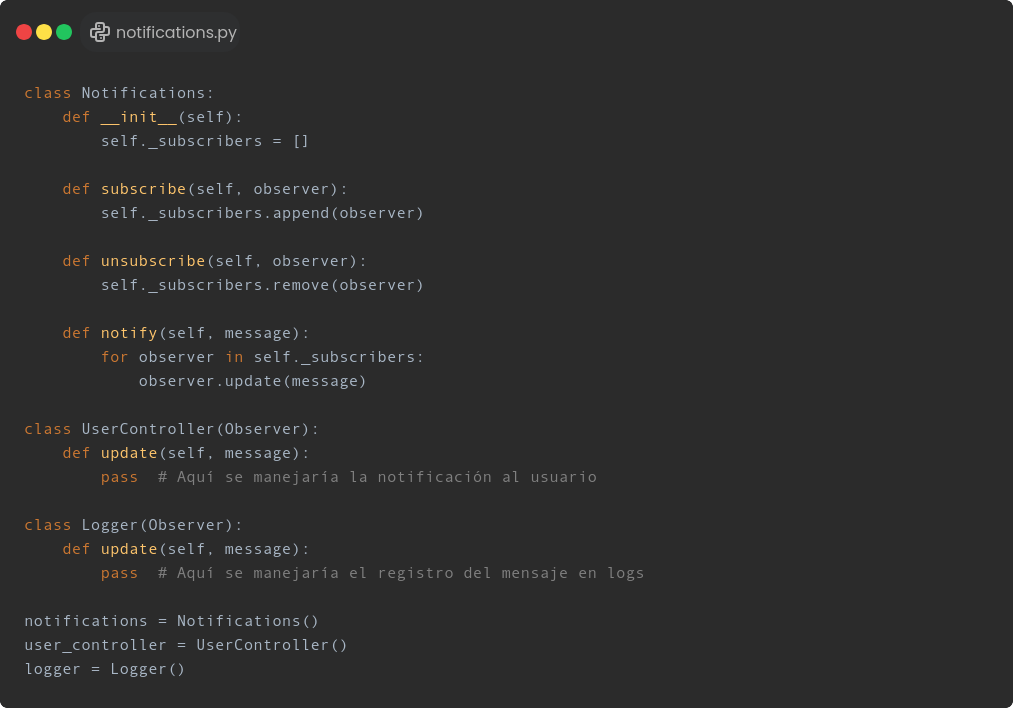
**Ilustración 9**:Ejemplo de uso del patrón Controlador

## 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.

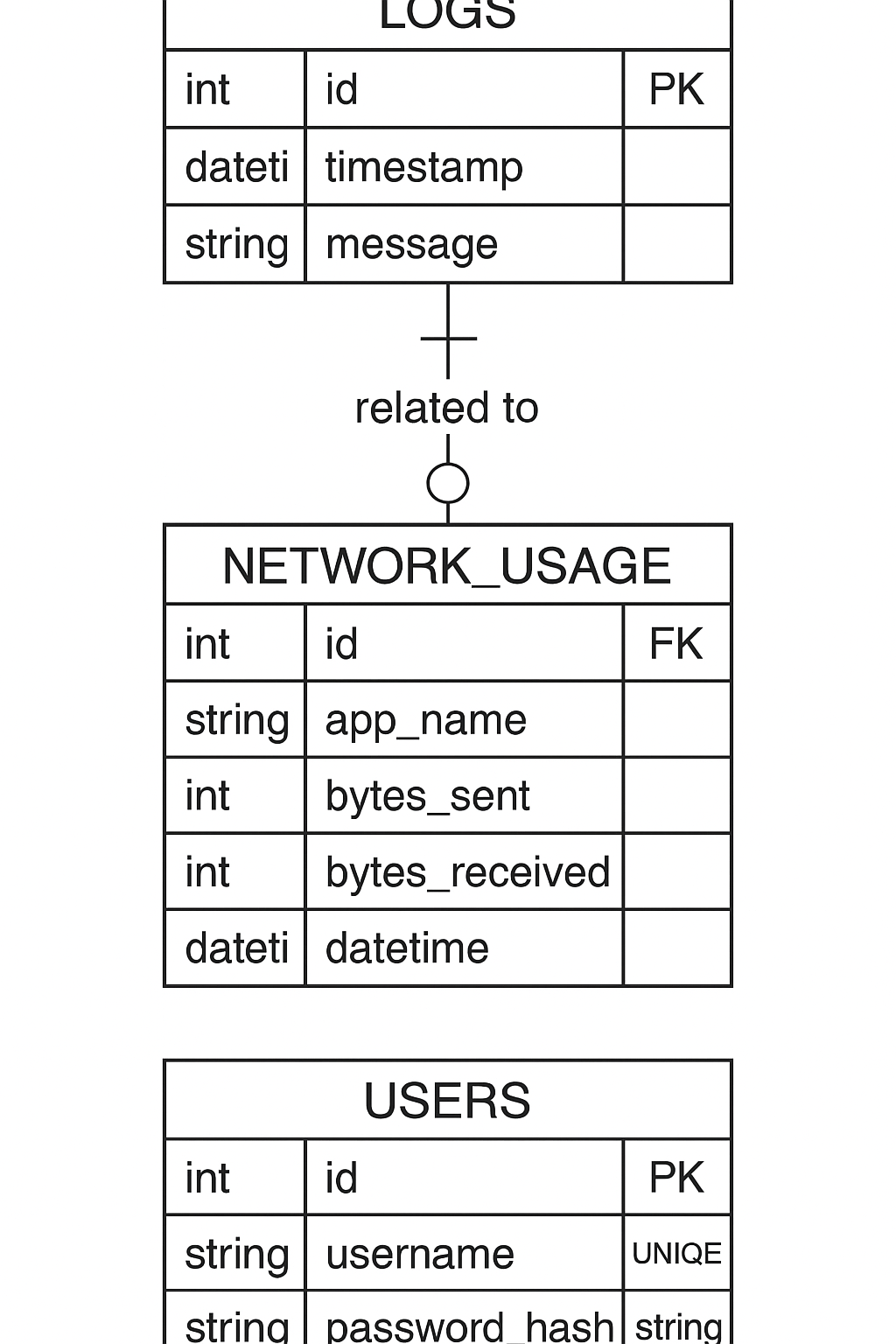


**Ilustración 10**:Ejemplo de uso del patrón Observador:

## 2.5.4 Modelo de datos

El modelado de datos es un proceso detallado que consiste en crear una representación visual de los datos y sus relaciones. Sirve como modelo de cómo se estructuran, almacenan y acceden a los datos para garantizar su coherencia y claridad en la gestión de datos (Pykes 2024).

**Modelo Entidad-Relación**



## 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: Validación de la solución propuesta**

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 Linux (distribución NOVA). Siguiendo la metodología Ágil AUP-UCI, se aplicaron estándares de codificación y buenas prácticas para asegurar una solución eficiente y funcional. Además, se detallan las pruebas realizadas, incluidas las unitarias, de integración y de aceptación, 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. En este contexto, se llevaron a cabo varias iteraciones, cuyos resultados se detallan a continuación:

|  |  |  |
| --- | --- | --- |
| **Iteración** | **Resultados (Historias de Usuario Implementadas)** | **Duración** |
| 1 | Autenticar usuario mediante en Cntlm. Consultar datos de consumo y cuota disponible, Iniciar servicio de proxy local, Detener servicio de proxy local y Gestionar cambio de contraseña | 20 días |
|  |  |  |
|  |  |  |

**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 (ej. ProxyAuthenticationManager).
* **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.

**3.4 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. 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.
* Facilidad de uso: El diseño busca que la herramienta sea fácil de comprender incluso para usuarios sin conocimientos técnicos avanzados. Se redujeron los pasos innecesarios, se simplificaron los flujos de acción, y se agregaron ayudas contextuales donde fue necesario.



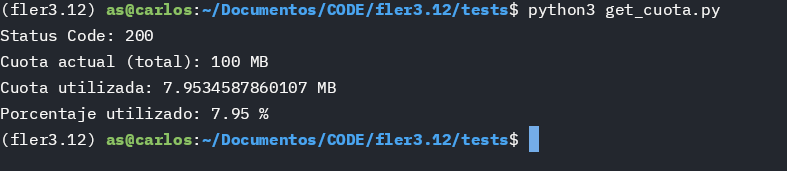
**3.2 Pruebas de software**

**3.2.1 Pruebas Unitarias**

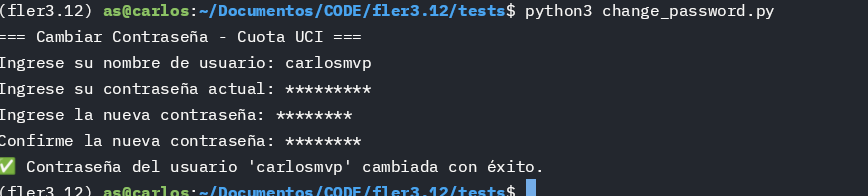
Las pruebas unitarias son el proceso en el que se prueba la unidad funcional de código más pequeña. Las pruebas de software ayudan a garantizar la calidad del código y son una parte integral del desarrollo de software. Una práctica recomendada en el desarrollo de software es escribir el software como 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», s. f.)

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**



**Pruebas de aceptación:**

Las pruebas de aceptación son una fase crucial en el desarrollo de software, cuyo propósito es verificar que el sistema cumple con los requisitos y expectativas establecidos por el cliente o usuario final. Estas pruebas se realizan generalmente al finalizar el proceso de desarrollo, antes de la implementación o lanzamiento del producto, y sirven para garantizar que el software está listo para su uso en un entorno real. («Pruebas de aceptación de usuario (UAT): qué son» 2023)

|  |  |
| --- | --- |
| Caso de prueba de aceptación | |
| Número: HU-01 | **Nombre del requisito:** Autenticación de usuario en la aplicación |
| Descripción: La aplicación debe permitir que el usuario se autentique con sus credenciales institucionales mediante una interfaz gráfica clara e intuitiva. | |
| Condiciones de ejecución:   * El usuario tiene una cuenta activa de la UCI. * La computadora está conectada a la red interna o con acceso al servicio de autenticación de la UCI. * La aplicación "Cuota UCI" está instalada correctamente. | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| Número: HU03\_P1 | **Nombre del requisito:** Visualización en tiempo real del consumo de datos |
| Descripción: La aplicación debe permitir que el usuario visualice su consumo de datos actualizado automáticamente, mediante una representación gráfica o numérica clara y comprensible. | |
| Condiciones de ejecución:   * El usuario debe haber iniciado sesión correctamente con sus credenciales UCI. * La computadora debe tener acceso a Internet o a la red interna que permite la consulta del consumo. * El backend del sistema debe estar activo y disponible para responder a las solicitudes SOAP. | |

**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>