



**HERRAMIENTA INFORMÁTICA PARA LA PROTECCIÓN  
CENTRALIZADA DE LOS PUERTOS USB EN LA DISTRIBUCIÓN  
CUBANA DE GNU/LINUX NOVA**

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

**Autor**

José Eduardo Ramos Marquez

**Tutores**

Ing. Luis Daniel Sierra Corredera

Msc. Ing. Ruth Yurina Vega Cutiño

**La Habana, 2020**

# Agradecimientos

A todos los que hicieron posible la realización de este trabajo.

# Declaración Jurada de Autoría

Declaro por este medio que yo José Eduardo Ramos Marquez, con carné de identidad 96082804583 soy el autor principal del trabajo titulado “Herramientas informáticas para la protección centralizada de los puertos USB en la distribución cubana de GNU/Linux Nova” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los \_\_\_\_\_ días del mes de \_\_\_\_\_ del \_\_\_\_\_

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

José Eduardo Ramos Marquez

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

Ing. Luis Daniel Sierra Corredera

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

Msc. Ing. Ruth Yurina Vega Cutiño.

# Resumen

El desarrollo de las Tecnologías de la Información y las Comunicaciones ha desencadenado grandes avances económicos, políticos y sociales a nivel mundial. Este progreso nos ha hecho casi dependientes de la tecnología y por tanto más vulnerables ante su uso. El aumento gradual de los delitos informáticos provoca una demanda creciente de protección de la información empresarial y personal, a lo que la distribución cubana de GNU/Linux Nova, desarrollada en el Centro de Soluciones Libres (CESOL) en la Universidad de las Ciencias Informáticas, responde con alternativas que garantizan un sistema seguro de ataques y sin puertas traseras conocidas. Restringir el acceso de la conexión a los puertos USB es una de las alternativas para preservar la confidencialidad, integridad y disponibilidad de la información. En el presente trabajo se planteó como objetivo, desarrollar una herramienta que gestione de forma remota y centralizada la restricción de acceso a la información a través de los puertos USB, en la distribución cubana de GNU/Linux Nova. Como resultado se obtuvo el diseño de un sistema informático que permite gestionar de manera remota y centralizada la restricción de acceso a la información a través de los puertos USB, en las estaciones de trabajo que ejecutan GNU/Linux Nova. También es parte de los resultados, la documentación generada de acuerdo a lo que define el escenario cuatro de la metodología AUP-UCI y la estrategia de pruebas a realizar.

Palabras clave: Nova, software libre, puertos USB, seguridad, control de acceso, gestión centralizada

# Abstract

*The development of Information and Communication Technologies has unleashed great economic, political and social advances worldwide. This progress has made us almost dependent on technology and therefore more vulnerable to its use. The gradual increase in computer crimes causes a growing demand for the protection of business and personal information, to which the Cuban distribution of GNU / Linux Nova, developed at the Center for Free Solutions (CESOL) at the University of Computer Science, responds with alternatives that guarantee a secure attack system and no known back doors. Restricting the access of the connection to the USB ports is one of the alternatives to preserve the confidentiality, integrity and availability of the information. The objective of this work was to develop a tool that remotely and centrally manages the restriction of access to information through USB ports, in the Cuban distribution of GNU / Linux Nova. As a result, the design of a computer system was obtained that allows to manage remotely and centrally the restriction of access to USB ports, on workstations running GNU / Linux Nova. Also part of the results is the documentation generated according to what defines scenario four of the AUP-UCI methodology and the testing strategy to be carried out.*

Keywords: Nova, free software, USB ports, security, access control, centralized management

# Índice

Introducción .....	1
Capítulo 1. Fundamentación teórica .....	6
1.1 Conceptos asociados al dominio del problema .....	6
1.2 Análisis de sistemas homólogos.....	8
1.3 Metodología de desarrollo de software .....	11
1.3.1 Metodología de desarrollo de software AUP versión UCI .....	11
1.4 Lenguaje y herramientas para el modelado de la propuesta de solución.....	15
1.5 Lenguaje y herramientas para el desarrollo de la propuesta de solución.....	18
Conclusiones del capítulo.....	19
Capítulo 2. Análisis y diseño del sistema .....	20
2.1 Propuesta de solución .....	20
2.2 Requisitos Funcionales.....	21
2.3 Requisitos no funcionales.....	25
2.4 Historias de Usuario .....	26
2.5 Arquitectura por capas.....	27
2.6 Patrones de diseño.....	29
2.7 Diagrama de paquetes .....	30
2.8 Diagrama de clases.....	31
Conclusiones del capítulo.....	32
Capítulo 3. Implementación y estrategia de pruebas.....	33
3.1 Implementación .....	33
3.2 Diagrama de componentes.....	33
3.3 Estándar de codificación.....	34
3.4 Despliegue de la solución propuesta .....	34
3.5 Requerimientos del hardware .....	35

3.6 Requerimientos del software .....	35
3.7 Estrategia de pruebas .....	36
3.7.1 Pruebas funcionales.....	36
3.7.2 Pruebas no funcionales.....	38
Conclusiones del capítulo.....	39
Conclusiones generales.....	40
Recomendaciones .....	41
Bibliografía.....	42
Anexos .....	46

## Índice de ilustraciones

Ilustración 1. Arquitectura (Elaboración propia).....	28
Ilustración 2. Clase experta (Elaboración propia).....	30
Ilustración 3. Diagrama de paquetes (Elaboración propia).....	31
Ilustración 4. Diagrama de clases (Elaboración propia) .....	32
Ilustración 5. Diagrama de componentes (Elaboración propia) .....	33
Ilustración 6. Diagrama de despliegue (Elaboración propia) .....	35

## Índice de tablas

Tabla 1. Comparación entre sistemas homólogos para Linux (Elaboración Propia) .....	10
Tabla 2. Fases de la variante AUP-UCI (Elaboración Propia) (Sánchez 2015) .....	12
Tabla 3. Requisitos Funcionales (Elaboración propia) .....	22
Tabla 4. Historia de usuario: Autenticar usuario administrador de la aplicación (Elaboración propia) .....	26

Tabla 5. historia de usuario: Cambiar contraseña de usuario administrador (Elaboración propia) .....	27
Tabla 6. Requerimientos de Hardware (Elaboración propia) .....	35
Tabla 7. Caso de prueba de aceptación para la historia de usuario: "Autenticar usuario administrador de la aplicación" (Elaboración propia).....	38

## Índice de anexos

Anexo 1. Historia de usuario #3 "Permitir el uso del dispositivo conectado a la estación de trabajo" (Elaboración propia) .....	46
Anexo 2. Historia de usuario #4. "Notificar al administrador del dispositivo conectado no presente en la base de datos central" (Elaboración propia) .....	47
Anexo 3. Caso de prueba de aceptación para la historia de usuario: "Cambiar contraseña de usuario administrador" (Elaboración propia) .....	48
Anexo 4. Caso de prueba de aceptación para la historia de usuario: " Permitir el uso del dispositivo conectado a la estación de trabajo" (Elaboración propia) .....	48
Anexo 5. Caso de prueba de aceptación para la historia de usuario: " Notificar al administrador del dispositivo conectado no presente en la base de datos del servidor central" (Elaboración propia).....	49



## **Introducción**

Como parte de la política de informatización de la sociedad trazada por el gobierno cubano se lleva a cabo un proceso de migración a estándares abiertos de las estaciones de trabajo en los Órganos y Organismos de la Administración Central del Estado (OACE), con el objetivo de disminuir la dependencia de Microsoft Windows y el software privativo en general. Todo este proceso está sostenido por la distribución cubana de GNU/Linux Nova, un sistema operativo de propósito general basado en Ubuntu desarrollado en el Centro de Soluciones Libres (CESOL) en la Universidad de las Ciencias Informáticas; para satisfacer los requerimientos derivados del mencionado proceso de migración (Albalat et al. 2012).

Para asegurar un correcto proceso de migración a software libre, Nova se desarrolla teniendo en cuenta varios fundamentos, entre los cuales se encuentra el fundamento de la seguridad. Mediante la utilización del modelo de desarrollo colaborativo que propone el movimiento de software libre y el acceso al código fuente, así como su exhaustivo proceso de revisión y auditoría, se asegura un sistema seguro de ataques y sin puertas traseras conocidas (Pierra 2011).

La seguridad de los datos es un factor primordial para los administradores y los desarrolladores de sistemas, ya que existe un alto riesgo de que los empleados o personas mal intencionadas copien, destruyan o supriman información de la empresa. Bases de datos, estrategias empresariales y códigos fuentes de software son los más susceptibles al robo, acarreando serios problemas para las empresas e instituciones.

La mayoría de las fugas de información son ocasionadas por empleados que desconocen las medidas básicas de ciberseguridad dentro la empresa, algunas veces de forma accidental, aunque otras veces, de forma premeditada. En ocasiones los responsables de la ciberseguridad de las organizaciones solo implementan políticas de seguridad en elementos puntuales dentro de la empresa, como la sala de servidores o proteger la red interna de amenazas procedentes desde internet, estas no suelen ser suficientes debido a que las estaciones de trabajo de los empleados es uno de los escenarios fundamentales para proteger la información de dicha organización (Blez et al. 2014).

# Introducción

Entre las formas más comunes de robar información se encuentran las copias en dispositivos extraíbles tales como memoria USB (Universal Serial Bus) o discos duros removibles (Casais y Reinoso 2017). Estos dispositivos son por lo general pequeños y fáciles de ocultar, haciendo difícil tener un estricto control para evitar la fuga de información basado en la confianza de que el personal no violará las políticas internas de seguridad. Por otra parte, resulta tedioso para los responsables de seguridad informática la tarea de otorgar y revocar permisos para la conexión de los dispositivos de almacenamiento a los puertos USB, en cada estación de trabajo.

De acuerdo a la demanda de varias instituciones con tareas claves en el desarrollo socioeconómico y político del estado y partiendo del fundamento antes expuesto, se plantea la necesidad de proteger la información sensible que almacenan sus ordenadores. En un intento para controlar estas amenazas, se establecen políticas internas que tratan de controlar el uso de dispositivos de almacenamiento. La Resolución 127-07 (Ministerio de la Informática y las Comunicaciones 2007) establece el Reglamento de Seguridad para las Tecnologías de la Información y propone que los controles de seguridad que se implementen pueden ser:

- Medios Humanos: Aquí se hará referencia al papel del personal dentro del sistema de seguridad implementado, definiendo sus responsabilidades y funciones respecto al diseño, establecimiento, control, ejecución y actualización del mismo.
- Medios Técnicos de Seguridad: Se describirán los que se han utilizados en función de garantizar niveles de seguridad adecuados, tanto al nivel de software como de hardware, así como la configuración de los mismos

Estos controles deben implementarse de manera conjunta, pues el factor humano es uno de los eslabones más débiles en la cadena de la seguridad informática, y la mejor manera de asegurar un control completo es combinándolo con la colocación de barreras tecnológicas (Avenia 2017).

La situación anteriormente expuesta permite formular el siguiente **problema de investigación**:

¿Cómo restringir de forma remota y centralizada el acceso a la información a través de los puertos USB en la distribución cubana de GNU/Linux Nova?

# Introducción

Para la solución del problema se asume como **objeto de estudio** los mecanismos de restricción de acceso a la información a través de los puertos USB teniendo como **campo de acción** la gestión remota y centralizada de los mecanismos de restricción de acceso a la información a través de los puertos USB en la distribución cubana de GNU/Linux Nova.

Por lo que se propone como **objetivo general**: desarrollar una herramienta que gestione de forma remota y centralizada la restricción de acceso a la información a través de los puertos USB, en la distribución cubana de GNU/Linux Nova.

Como **objetivos específicos**:

- Analizar los referentes teóricos metodológicos sobre los mecanismos de restricción de acceso a la información a través de los puertos USB.
- Definir los requisitos funcionales necesarios para el diseño de una herramienta informática para la gestión remota y centralizada de los puertos USB en la distribución cubana de GNU/Linux Nova.
- Diseñar herramienta para gestionar de forma remota y centralizada la restricción de acceso a la información a través de los puertos USB en la distribución cubana de GNU/Linux Nova.
- Implementar una herramienta para gestionar de forma remota y centralizada la restricción de acceso a la información a través de los puertos USB en la distribución cubana de GNU/Linux Nova.
- Diseñar una estrategia de pruebas que permita evaluar la herramienta para gestionar de forma remota y centralizada la restricción de acceso a la información a través de los puertos USB en la distribución cubana de GNU/Linux Nova.

**Preguntas científicas:**

1. ¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre la restricción de acceso a los puertos USB en GNU/Linux?
2. ¿Cuáles son los mecanismos a tener en cuenta para definir los requisitos funcionales necesarios para el correcto funcionamiento de una herramienta para la gestión remota y centralizada de los puertos USB en GNU/Linux Nova?

# Introducción

3. ¿Cuáles son los elementos que hay que tener en cuenta para el diseño de una herramienta que, mediante una gestión remota y centralizada, restrinja el acceso a la información a través de los puertos USB en GNU/Linux Nova?
4. ¿Cuáles son los componentes necesarios implementar para obtener una herramienta que, mediante una gestión remota y centralizada, restrinja el acceso a la información a través de los puertos USB en GNU/Linux Nova?
5. ¿Qué pruebas de software aplicar para la evaluación de una herramienta que, mediante una gestión remota y centralizada, restrinja el acceso a la información a través de los puertos USB en GNU/Linux Nova?

Dentro de los **métodos teóricos** de investigación empleados se destacan el método **dialéctico** que es utilizado para el estudio crítico de los trabajos anteriores sobre los sistemas de control de acceso a los puertos USB y las políticas de seguridad de las distintas organizaciones, el método **analítico-sintético** para tomar el objetivo general y descomponerlo en partes para observar sus causas y efectos con el fin de obtener requisitos e ideas para dar solución a los objetivos específicos (Rodríguez y Pérez 2017).

Los **métodos empíricos** utilizados son: el **análisis documental** en la revisión de la literatura especializada para la extracción de los elementos más importantes relacionados con los mecanismos de restricción de acceso a los puertos USB y la gestión centralizada de estos mecanismos. Se analizaron documentos legales, tesis y estándares relacionadas con el acceso de los dispositivos de almacenamiento a los puertos USB. Para la validación del modelo, se emplearon los métodos: **experimental** y la **medición** (Hernández-Sampieri y Torres 2018).

## **Aporte práctico.**

Con la realización del presente trabajo se espera obtener un sistema informático; que permita gestionar de manera remota y centralizada la restricción de acceso a la información a través de los puertos USB, en las estaciones de trabajo que ejecutan GNU/Linux Nova. Con su puesta en práctica se podrá mitigar el riesgo de fuga de información a través de dispositivos extraíbles sin sobrecargar las estaciones de trabajo. De igual forma, la gestión centralizada favorecerá la seguridad de la herramienta y facilitará la administración.

El presente trabajo de diploma se encuentra estructurado de la siguiente forma:

# Introducción

En el **Capítulo 1: Fundamentación teórica**, este capítulo está enfocado a la fundamentación teórica del objeto de estudio y el campo de acción, se abordan diferentes temas relacionados con la restricción de acceso a los puertos USB. Además, se realiza una descripción de las herramientas y tecnologías a utilizar en el desarrollo del sistema.

En el **Capítulo 2: Propuesta de solución, análisis y diseño**, en este capítulo se realiza la descripción de la propuesta de solución, se realiza el modelado del negocio y se definen y especifican los requisitos funcionales del sistema. También se analizan los elementos a tener en cuenta para el diseño de la solución informática que permitirá el control de los puertos USB en Nova.

En el **Capítulo 3: Implementación, pruebas y evaluación**, en este capítulo se definen los estándares de implementación, así como la interfaz de usuario, el diagrama de despliegue y requisitos de hardware para el óptimo funcionamiento de la herramienta. Se definen una estrategia de pruebas de software a realizar para validar la calidad de la solución informática tales como pruebas unitarias, de integración, de humo y de aceptación.

## Capítulo 1. Fundamentación teórica

En el presente capítulo se presenta la definición del marco teórico para dar respuesta al problema de investigación: ¿Cómo restringir de forma remota y centralizada el acceso a la información a través de los puertos USB en la distribución cubana de GNU/Linux Nova?

Se abordan los conceptos relacionados al objeto de estudio y se analiza la existencia de posibles soluciones al problema planteado. Se estudia la metodología de desarrollo escogida para obtener una solución informática que permita realizar una gestión centralizada de los mecanismos de control de acceso a la información a través de los puertos USB, así como, las tecnologías y herramientas que serán necesarias para el modelado e implementación de la propuesta de solución.

### 1.1 Conceptos asociados al dominio del problema

Para dar respuesta a la pregunta científica ¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre la restricción de acceso a los puertos USB en GNU/Linux Nova? se considera oportuna comenzar por la definición de los términos (en lenguaje técnico) que van a ser empleados con mayor frecuencia.

**Puerto USB:** Puerto USB es una noción con varios usos. En la informática, el término se emplea para nombrar a una clase de conexión que posibilita el envío y la recepción de información. USB, por su parte, es la sigla correspondiente a Universal Serial Bus, una interfaz que permite la conexión de periféricos a diversos dispositivos, entre los cuales se encuentran los ordenadores y los teléfonos móviles. El puerto USB, por lo tanto, es un componente que tiene la finalidad de conectar distintos dispositivos entre sí. Una impresora, un mouse (ratón), una webcam y unos altavoces son algunos ejemplos de periféricos que pueden conectarse a un puerto USB, sin olvidar los cada vez más populares discos duros externos y las clásicas llaves de memoria o pendrives (Pérez y Gardey 2015).

Estos fueron introducidos en el mercado en el 1996 y en la actualidad existen varias versiones como la 1.1, 2.0, 3.0 y 3.1. Todos los dispositivos de almacenamiento USB son compatibles con todos los puertos, aunque la velocidad del tráfico de información es determinada por el puerto. Como norma general los USB tienen cuatro conectores, dos para la alimentación eléctrica y dos para la transmisión de datos, esta puede ser de tres maneras,

# Capítulo 1

por interrupciones, en bloques o isócrona. Para la transmisión de los datos en los dispositivos de almacenamiento se utiliza la transmisión en bloques la cual permite la movilización de grandes cantidades de información (Rosero 2020).

**Memoria flash o pendrive:** Las memorias flash o pendrive son una clase de chip que se emplea para el almacenamiento y el traslado de datos. Esta tecnología puede encontrarse en tarjetas, dispositivos USB, cámaras digitales, reproductores MP3 y otros elementos tecnológico (Pérez y Meriño 2020).

**Control centralizado:** Según la Real Academia de la Lengua Española, la palabra control hace referencia a la comprobación, inspección, fiscalización, intervención, regulación, manual o automática sobre un sistema, persona o proceso. El término centralizado viene del adjetivo central, que hace referencia a que ejerce su acción sobre todo un campo o sistema (Real Academia Española 2019).

Para la presente investigación, control centralizado hace referencia al aseguramiento de manera remota del cumplimiento de cierto plan o regulación. En particular al cumplimiento de las medidas de seguridad implementadas por una organización en cuestión, así como el registro de los accesos autorizados.

**Gestión remota:** En la informática, es la capacidad de realizar cambios y controlar a distancia un dispositivo. Por lo general se trata de tener total acceso a un dispositivo servidor desde una estación de trabajo cliente (InfoComputer 2020).

**Mecanismos de restricción de acceso al puerto USB:** Los mecanismos de restricción de acceso a los puertos USB son o bien políticas definidas por las organizaciones para controlar el uso de dispositivos de almacenamiento o bien barreras tecnológicas diseñadas con el mismo objetivo (Avenia 2017).

Con el fin de controlar el acceso a los puertos USB y por tanto la posibilidad de atentar contra la seguridad de la información por parte del personal de una organización estas implementan medidas para restringir el acceso a estos puertos. Estas medidas pueden ser políticas de seguridad implantadas por las mismas organizaciones, la implantación de barreras tecnológicas o ambas. Estos mecanismos posibilitan que solo ciertos dispositivos autorizados por el administrador o encargado de la seguridad en una organización tengan acceso a los puertos USB disminuyendo el riesgo de amenazas.

# Capítulo 1

Entre las medidas más utilizadas por las organizaciones son la concientización del personal de las estaciones de trabajo siendo estos el eslabón más débil en cualquier organización y estando en contacto con la información, la dificultad de las contraseñas utilizadas y la custodia de la estación de trabajo cuando el equipo está desbloqueado. Conjunto con estas medidas se suele utilizar herramientas informáticas que limitan el acceso a los puertos USB de las estaciones de trabajo a los dispositivos permitidos por la organización. Estas herramientas se encuentran disponibles para todas las plataformas, variando sus funcionalidades, pero manteniendo su objetivo principal, el restringir el acceso a la información mediante los puertos USB.

**Mecanismos de restricción de acceso a nivel de sistema operativo:** A diferencia de los sistemas operativos Windows que la gestión de los puertos USB es realizada por el microprocesador apoyado en los distintos drivers, en Linux, se encarga el kernel permitiendo el bloqueo de los dispositivos conectados. Un modo de restringir el acceso a los datos mediante los puertos USB es utilizando una de las funcionalidades del kernel, mediante el comando *chmod*, este comando permite alterar los permisos administrativos tanto de archivos como de directorios, mediante el uso de este comando se pueden modificar los permisos del directorio */media/* o si la distribución usa *systemd*, */run/media/* que es donde se montan los dispositivos de almacenamiento o memorias flash (*pendrives*) permitiendo que solo el usuario *root* tenga acceso a ella (Shotts 2019).

**Mecanismos de restricción de acceso a los puertos USB en la distribución cubana de GNU/Linux Nova:** En la distribución cubana de GNU/Linux Nova la restricción de acceso a los puertos USB se encuentra sostenida en las políticas de seguridad de las organizaciones y en la administración local de los permisos para el acceso a los puertos USB (Ortíz 2018).

## 1.2 Análisis de sistemas homólogos

Existen algunas herramientas de gestión y control de puertos USB, algunas comerciales y otras de libre distribución. Las herramientas comerciales o las diseñadas para ejecutarse en sistemas operativos que no son basados en Linux, carecen de valor desde el punto de vista de solución al problema. Sin embargo, se analizan algunos aspectos de las herramientas utilizadas en el sistema operativo Windows, aunque no se toman en cuenta en el desarrollo de la tabla de sistemas homólogos, pues dichos aspectos pueden ser útiles durante el desarrollo de la disciplina de requisitos.



## **BuduLock**

BuduLock es una herramienta para la protección de carpetas mediante contraseñas y el bloqueo de acceso de dispositivos de almacenamiento extraíble como memorias USB desarrollada por Mouse Click Enterprise. Esta herramienta está desarrollada para la plataforma Windows, es gratuito y su fecha de actualización es de febrero del 2019. Una de sus desventajas es la no integración con el menú contextual (Softonic 2020).

## **USB Blocker**

Esta herramienta desarrollada para el sistema operativo Windows en todas sus versiones superiores a XP permite el bloqueo y desbloqueo de puertos USB. Para el bloqueo y desbloqueo de estos es necesario contar con las credenciales administrativas de la estación de trabajo. Es una herramienta ligera y actualmente se encuentra en su versión 1.0, es de uso comercial y cuenta con una licencia de prueba de 7 días (SysTools 2020).

## **USB Block**

USB Block es un software desarrollado para Windows con el fin de bloquear el acceso de dispositivos de almacenamiento. Esta herramienta utiliza un sistema de listas negras y blancas para definir los dispositivos autorizados y para la protección de su configuración cuenta con una contraseña. Es comercial aunque cuenta con una versión de prueba (Newsoftwares.net 2019).

## **USBGuard**

USBGuard es un software desarrollado para las distribuciones Linux para la protección de puertos USB mediante una característica implementada directamente en el kernel de Linux. Esta herramienta crea listas blancas y listas negras basadas en atributos de los dispositivos, una vez conectado un dispositivo USB esta busca secuencialmente en su archivo de configuración para comprobar si ese dispositivo está permitido o rechazado. Este debe ser configurado por el usuario el cual debe especificar que dispositivos permitir en caso de denegarlos todos, o que dispositivos denegar en caso de permitirlos todos haciendo el proceso un poco engorroso y difícil para los usuarios menos familiarizados con la informática (Perez-Gonzalez 2007).

## PPUSB

PPUSB es una herramienta para la protección de los puertos USB desarrollada por el centro CESOL (Centro de Software Libre) para el sistema operativo Nova y de libre licencia. Su característica principal es la de permitir al usuario bloquear o no un nuevo dispositivo de almacenamiento USB mediante indicadores propios de este, como el número de serie. Entre sus principales desventajas se encuentra el acceso con el que cuenta el usuario de la estación de trabajo a este permitiéndole violar las políticas de seguridad a voluntad (Ortíz 2018).

A continuación, se muestra una tabla comparativa de las herramientas de libre distribución:

**Tabla 1. Comparación entre sistemas homólogos para Linux (Elaboración Propia)**

Herramienta	Plataforma	Licencia	Centralizado	Método de control	Lenguaje de programación
<b>PPUSB</b>	Linux	Libre	No	Autorización de unidades USB.	C
<b>USBGuard</b>	Linux.	Libre	No	Autorización de unidades USB.	C++

Las herramientas desarrolladas para el sistema operativo Windows, aunque no se tuvieron en cuenta en la tabla comparativa, fueron analizadas algunas de sus características como el uso de listas blancas y negras, así como, la protección por contraseña del acceso a la configuración de estas.

Las herramientas como USBGuard y PPUSB solo sirven de base teórica para la propuesta de solución, ya que estos sistemas no cumplen los requerimientos y especificaciones definidas por el cliente. En ambos casos el control se lleva de manera local, de estación de trabajo a estación de trabajo. Otro aspecto es el lenguaje de programación, que en el caso de la herramienta PPUSB es C y el lenguaje requerido por el cliente es C ++ debido a las funcionalidades que brinda este siendo un perfeccionamiento este de C (Centro de Soluciones Libres 2017).

Luego del análisis realizado sobre los sistemas existentes se infiere que ninguna resuelve el problema en cuestión, aunque, constituyen buenos ejemplos para el desarrollador. Sus funcionalidades se pueden tomar como base para el desarrollo de una herramienta, que responda a la necesidad de gestión centralizada y permita crear y modificar una lista con los dispositivos que pueden tener acceso a la conexión a través del puerto USB de las estaciones de trabajo, así como, bloquear los que no deben tener acceso.

Para realizar el desarrollo de una herramienta, será necesario definir una metodología de desarrollo de software y las tecnologías necesarias.

## **1.3 Metodología de desarrollo de software**

Una metodología de desarrollo, en ingeniería de software, es un conjunto de herramientas, técnicas, procedimientos y soporte documental encaminados a estructurar, planificar y controlar el proceso de desarrollo de forma organizada y lógica, que tienen como objetivo apoyar a los desarrolladores en la creación de un nuevo software (Kaisler 2005).

### **1.3.1 Metodología de desarrollo de software AUP versión UCI**

La metodología de desarrollo de software AUP es una versión simplificada de la metodología de desarrollo RUP (Rational Unified Process). Describe de una manera fácil y simple de entender la forma de desarrollar aplicaciones de software de negocio empleando técnicas ágiles y conceptos que se mantienen válidos en RUP (Ambler 2005).

Aplica técnicas ágiles que incluyen:

- Desarrollo dirigido por pruebas.
- Modelado Ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

AUP se preocupa especialmente por la gestión de riesgos. Propone que los elementos con alto riesgo obtengan prioridad en el desarrollo y sean abordados en etapas tempranas del mismo. Establece cuatro fases que transcurren de manera consecutiva: Inicio, Elaboración, Construcción y Transición (Ambler 2005).

# Capítulo 1

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP, con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la Universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma, y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, denominada Ejecución, y agregándose también una nueva fase denominada Cierre (Sánchez 2015). En la siguiente tabla se muestran las fases de la variante AUP-UCI.

**Tabla 2. Fases de la variante AUP-UCI (Elaboración Propia) (Sánchez 2015)**

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se establecen la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		

# Capítulo 1

	Cierre	En esta fase se analizan tanto los resultados del proyecto como la ejecución y se realizan las actividades formales de cierre del proyecto.
--	--------	---

La fase que se utilizan en el desarrollo de este trabajo es la de Ejecución, que define 7 disciplinas (Sánchez 2015):

**Modelado de negocio:** El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes:

- Casos de Uso del Negocio (CUN)
- Descripción de Proceso de Negocio (DPN)
- Modelo Conceptual (MC)

**Requisitos:** El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio.

**Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.

**Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

# Capítulo 1

**Pruebas internas:** En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.

**Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

**Pruebas de aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Para modelar el sistema en la disciplina Requisitos se condicionan cuatro escenarios, por lo que resulta oportuno realizar un análisis para identificar el escenario más conveniente para desarrollar la solución de esta investigación (Sánchez 2015).

- **Escenario No 1:** Proyectos que modelen el negocio con casos de uso del negocio solo pueden modelar el sistema con casos de uso del sistema.
- **Escenario No 2:** Proyectos que modelen el negocio con modelo conceptual solo pueden modelar el sistema con casos de uso del sistema.
- **Escenario No 3:** Proyectos que modelen el negocio con descripción de procesos de negocio solo pueden modelar el sistema con descripción de requisitos por proceso.
- **Escenario No 4:** Proyectos que no modelen negocio solo pueden modelar el sistema con historias de usuario.

Para el desarrollo de la herramienta se selecciona el escenario 4 ya que este aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. El proyecto no es extenso contribuyendo así al uso de las historias de usuario, ya que las mismas no debe poseer demasiada información.

En la presente investigación no se hace necesario realizar un estudio a profundidad de otras metodologías existentes, debido a que AUP-UCI es la metodología ya identificada previamente por CESOL. Se selecciona el escenario 4, definido para proyectos que no modelan negocio, sino que modelan historias de usuario, por ser el escenario que más se adecua a las características del desarrollo de la herramienta que se quiere obtener.

## 1.4 Lenguaje y herramientas para el modelado de la propuesta de solución

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software orientado a objetos. Algunas organizaciones los usan en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo. UML es considerado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante este es posible establecer las estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código (Fuentes y Vallecillo 2004).

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

**Los Diagramas de Estructura** enfatizan en los elementos que deben existir en el sistema modelado como son los Diagrama de clases, Diagrama de objetos, Diagrama de estructura compuesta, Diagrama de despliegue y Diagrama de paquetes.

**Los Diagramas de Comportamiento** enfatizan en lo que debe suceder en el sistema modelado, por ejemplo: Diagrama de actividades, Diagrama de casos de uso y Diagrama de estados.

**Los Diagramas de Interacción** son un subtipo de diagramas de comportamiento que enfatizan sobre el flujo de control y de datos entre los elementos del sistema modelado como es el Diagrama de secuencia, Diagrama de comunicación que es una versión simplificada del Diagrama de colaboración, Diagrama de tiempos y Diagrama de vista de interacción.

# Capítulo 1

UML permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos. También documenta todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, entre otros.). Este lenguaje de modelado es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

## **Herramientas CASE**

Para superar las dificultades existentes en el uso de las diversas tecnologías, la industria de computadoras ha desarrollado un soporte automatizado para el desarrollo y mantenimiento de software. Este es llamado Computer Aided Software Engineering (CASE) (Alarcón y Sandoval 2008).

Las herramientas CASE son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de un software. Los estados en el ciclo de vida de desarrollo de un software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. Una herramienta CASE se utiliza para ayudar a las actividades del proceso de software que es utilizado para diseñar e implementar otro software. Las herramientas Case facilitan mejoras en la calidad y productividad del diseño y desarrollo. Estas herramientas proporcionan a los desarrolladores de software grandes ventajas: apoyan a las metodologías y métodos, mejoran la comunicación entre las personas que interactúan con el sistema permitiéndoles compartir mejor su trabajo, establecen métodos eficientes para almacenar y utilizar los datos, hacen más eficaz el mantenimiento y automatizan fragmentos del análisis y diseño pesados y vulnerables a errores (Alarcón y Sandoval 2008).

## **Visual Paradigm**

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Pressman 2002).



# Capítulo 1

Esta herramienta se caracteriza por su disponibilidad en múltiples plataformas (Windows, Linux), diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad. El uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Las capacidades de ingeniería directa e inversa, esta última solo disponible en su versión privada. Modelo y código que permanece sincronizado en todo el ciclo de desarrollo. Disponibilidad de múltiples versiones, con diferentes especificaciones. Tiene una licencia gratuita y una licencia comercial. Soporta aplicaciones Web, las imágenes y reportes generados, no son de muy buena calidad, está disponible en varios idiomas. Generación de código para Java y exportación como HTML. Fácil de instalar y actualizar. Compatibilidad entre ediciones. Soporte de UML versión 2.1. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento. Modelado colaborativo con CVS y Subversión (control de versiones). Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XML. Ingeniería de ida y vuelta. Ingeniería inversa - Código a modelo, código a diagrama. Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL. Generación de código - Modelo a código, diagrama a código (Paradigm 2020).

Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso. Generación de código y despliegue de EJB - Generación de beans para el desarrollo y despliegue de aplicaciones. Diagramas de flujo de datos.

Visual Paradigm 8.1 es una herramienta que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Además de permitir diseñar todos los tipos de diagramas que se necesitan en esta investigación. Permite generar la documentación necesaria para el posterior diseño e implementación del sistema. Tiene también un diseño centrado en casos de uso y enfocado al negocio. Uno de los aspectos más importantes de la selección de esta herramienta es que en la Universidad de Ciencias Informáticas se está trabajando en función de utilizar tecnologías libres y Visual Paradigm 8.1 es una herramienta libre y multiplataforma (Paradigm 2020).

## 1.5 Lenguaje y herramientas para el desarrollo de la propuesta de solución

El lenguaje propuesto para la implementación de la herramienta es C++, este fue diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma (Smith 2020).

La gestión de periféricos en los sistemas operativos en su mayoría está programada en C, lo cual hace recomendable el uso de C++. C++ también es un lenguaje compilado lo que no permite ver el código al usuario una vez terminada la aplicación imposibilitando la búsqueda de vulnerabilidades de la misma maximizando así la seguridad.

Algunas de las características del lenguaje de programación C++ es su sintaxis heredada del lenguaje C, su programación orientada a objetos (POO), su rapidez a la hora de compilar y ejecutar los programas es mucho mayor que otros lenguajes, este también tiene compatibilidad con bibliotecas lo cual a escribir el código más rápidamente (Smith 2020).

### **Visual Studio Code.**

Como editor de código se propone el Visual Studio Code desarrollado por Microsoft para Windows, Linux y macOS, incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Posee soporte incorporado para JavaScript, TypeScript y Node.js además de extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity (Microsoft 2020)).

### **Git**

Para el control de versiones se propone el uso de la herramienta Git la cual es libre y de código abierto que permite el control de proyectos tanto grandes como pequeños con velocidad y eficacia. Git es fácil de aprender y tiene una huella pequeña con un rendimiento increíblemente rápido. Supera a las herramientas Subversion, Perforce y ClearCase con características como ramificación local barata, áreas de preparación convenientes y múltiples flujos de trabajo.

# Capítulo 1

La característica Git que realmente lo distingue de casi cualquier otro SCM (Software Configuration Management) es su modelo de ramificación permitiendo tener múltiples líneas de desarrollo locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de estas lleva segundos.

Esto significa que puede hacer cosas como:

- Cambio de contexto sin fricción.
- Líneas de código basadas en roles.
- Flujo de trabajo basado en funciones.
- Experimentación Desechable.

Hay maneras de lograr algo de esto con otros sistemas, pero el trabajo involucrado es mucho más difícil y propenso a errores. Git hace que este proceso sea increíblemente fácil y cambia la forma en que la mayoría de los desarrolladores trabajan cuando lo aprenden (Chacon y Straub 2014).

## Conclusiones del capítulo

El desarrollo de este capítulo permitió el estudio de los fundamentos teóricos sobre la restricción de acceso a los puertos USB. Luego del estudio de las soluciones existentes se llegó a la conclusión de que es necesario el desarrollo de una nueva herramienta para restringir y controlar el acceso a los puertos USB en las distribuciones cubanas de GNU/Linux Nova, pues las existentes no cumplen con el requisito de control centralizado o con el lenguaje de programación a emplear. La metodología ágil AUP en su variante para la UCI en su escenario 4 es la más adecuada para el desarrollo de la herramienta, por ser la definida en el centro CESOL para el desarrollo de software y por la ventaja que representa que los especialistas tengan ya experiencia en el empleo de la misma. Visual Studio y el Visual Paradigm poseen las características y prestaciones necesarias para el modelado e implementación de una solución informática al problema planteado.

## Capítulo 2. Análisis y diseño del sistema

El presente capítulo responde a las preguntas científicas ¿Qué elementos hay que tener en cuenta para el desarrollo de una solución informática que, mediante una gestión centralizada, restrinja y controle el acceso a los puertos USB en GNU/Linux Nova? y ¿Cuáles componentes son necesarios implementar para el desarrollo de una herramienta que, mediante una gestión centralizada, restrinja y controle el acceso a los puertos USB en GNU/Linux Nova? Para lo cual se enmarca en las disciplinas de requisitos y análisis y diseño de la metodología seleccionada proponiendo una solución a la problemática planteada y definiendo la arquitectura a utilizar, así como, los requisitos y la validación de estos. Se realizan los diagramas pertinentes en lenguaje de modelado unificado (UML) para lograr una mejor comprensión de las clases y relaciones de estas en la herramienta.

### 2.1 Propuesta de solución

Se propone la creación de una herramienta, programada en el lenguaje C++, que permita restringir el acceso a los puertos USB de las estaciones de trabajo que ejecutan el sistema operativo GNU/Linux Nova, a la vez que lleva un registro de los dispositivos autorizados que se conecten. Las acciones de administración se harán de forma remota y los usuarios autorizados a realizar dichas acciones contarán con una contraseña de acceso al mismo.

El proceso de control de acceso a los puertos USB comienza al conectarse un dispositivo de almacenamiento. Automáticamente se toman los datos de este dispositivo como identificador y número de serie y comprueba si estos se encuentran entre los datos de los dispositivos autorizados por el usuario. En caso de encontrarse se le permite el montaje en la unidad USB al dispositivo; en caso contrario, el usuario puede elegir el envío de una notificación pidiendo que sea añadido a la lista blanca el dispositivo. Si esta es enviada y aceptada, se procede al montaje del dispositivo en la unidad del puerto USB, si no, continúa denegándose el acceso, en caso de que el usuario no envíe la notificación pues es denegado el montaje en el puerto USB del dispositivo de almacenamiento.

## **2.2 Requisitos Funcionales**

Para la obtención de los requisitos funcionales se tomaron en cuenta las características de los sistemas homólogos analizados en el capítulo, así como entrevistas con el cliente. En la tabla 3 se muestra el resultado de la captura de los requisitos funcionales:

Tabla 3. Requisitos Funcionales (Elaboración propia)

No.	Nombre	Descripción	Complejidad	Prioridad
RF_1	Cambiar contraseña de usuario administrador.	El sistema debe permitirle al usuario administrador cambiar su contraseña de acceso.	Baja	Media
RF_2	Autenticar usuario administrador de la aplicación.	El sistema debe permitir la autenticación de los usuarios administradores.	Baja	Alta
RF_3	Obtener número de serie del dispositivo conectado.	El sistema debe permitir la obtención del número de serie de un dispositivo conectado.	Media	Alta
RF_4	Obtener identificador de dispositivo conectado.	El sistema debe permitir la obtención del identificador del dispositivo conectado.	Baja	Alta
RF_5	Obtener etiqueta del dispositivo conectado.	El sistema debe permitir la obtención de la etiqueta del dispositivo conectado una vez este	Baja	Alta

## Capítulo 2

		haya sido montado en el sistema operativo.		
RF_6	Comprobar la información obtenida del dispositivo conectado en la base de datos del servidor central.	El sistema debe permitir la validación de los datos obtenidos de un dispositivo USB conectado en el servidor.	Media	Alta
RF_7	Permitir el uso del dispositivo conectado a la estación de trabajo.	El sistema debe permitir el uso de un dispositivo USB conectado a la estación de trabajo siempre y cuando este se encuentre en la lista de dispositivos permitidos.	Baja	Alta
RF_8	Impedir el uso del dispositivo conectado a la estación de trabajo.	El sistema debe denegar el acceso del dispositivo conectado a la estación de trabajo si este no se encuentra en la lista de los dispositivos permitidos.	Baja	Media
RF_9	Notificar al administrador del dispositivo conectado no	El sistema debe permitirle al usuario de la estación de trabajo notificarle al administrador de la herramienta sobre un nuevo	Media	Media

## Capítulo 2

	presente en la base de datos del servidor central.	dispositivo USB que desea que sea permitido informándole el porqué de la petición.		
RF_10	Añadir datos del dispositivo conectado a la base de datos del servidor central.	El sistema debe permitirle al administrador añadir los datos un nuevo dispositivo conectado a la estación de trabajo permitiendo el uso de este.	Media	Alta



## 2.3 Requisitos no funcionales

Los requisitos no funcionales hacen referencia a las propiedades del sistema como restricciones de tiempo de respuesta, estándares, fiabilidad, usabilidad, etc. (Pressman 2002).

Los requerimientos no funcionales se declaran de forma separada a los funcionales, a fin de separar las condiciones funcionales y no. Después de definir la arquitectura se realizó una revisión de los requisitos no funcionales quedando definidos de la siguiente manera:

- Usabilidad:  
RNF\_1 El sistema debe tener un diseño sencillo, con una interfaz amigable y fácil de operar.
- Seguridad:  
RNF\_2 La conexión entre el sistema y el servidor debe ser segura utilizando el protocolo SSH para la comunicación con el mismo.
- Compatibilidad:  
RNF\_3 El sistema ejecuta el sistema operativo GNU/Linux Nova.
- Accesibilidad:  
RNF\_4 El sistema tiene acceso al servidor.
- Fiabilidad:  
RNF\_5 El sistema debe tener soporte para recuperación ante fallos y errores.
- Rendimiento:  
RNF\_6 El sistema debe ser capaz de soportar varias iteraciones simultáneas por parte de los usuarios.

### Técnicas de validación de requisitos

Para la validación de requisitos se realizaron revisiones técnicas formales, diseños de casos de prueba y técnicas de prototipado. En las revisiones técnicas fueron analizados sistemáticamente por parte del desarrollador y el cliente estos requisitos en búsqueda de anomalías u omisiones, los casos de prueba permitirán detectar errores o defectos relacionados con las funcionalidades de la herramienta. Las técnicas de prototipado se utilizarán para mostrar un prototipo de la herramienta a los usuarios finales y clientes para

analizar si dicho modelo cumple con las necesidades presentadas. Con la validación de estos requisitos se da inicio al diseño de una herramienta que gestione los requisitos obtenidos y validados mediante las técnicas definidas anteriormente.

## 2.4 Historias de Usuario

Las historias de usuario describen la salida necesaria, características y funcionalidad del software que se va a elaborar (Pressman 2002). En las siguientes tablas se muestran las historias de usuario referentes a los requisitos funcionales y las interfaces de usuario con las que contará la herramienta, el resto de ellas se encuentran en los anexos.

Estas tablas muestran el nombre de los usuarios encargados, el programador responsable del desarrollo de la historia de usuario, una descripción de la misma y el prototipo de interfaz de usuario propuesto.

**Tabla 4. Historia de usuario: Autenticar usuario administrador de la aplicación (Elaboración propia)**

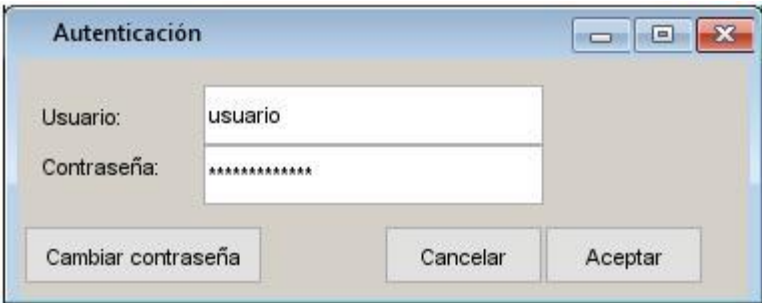
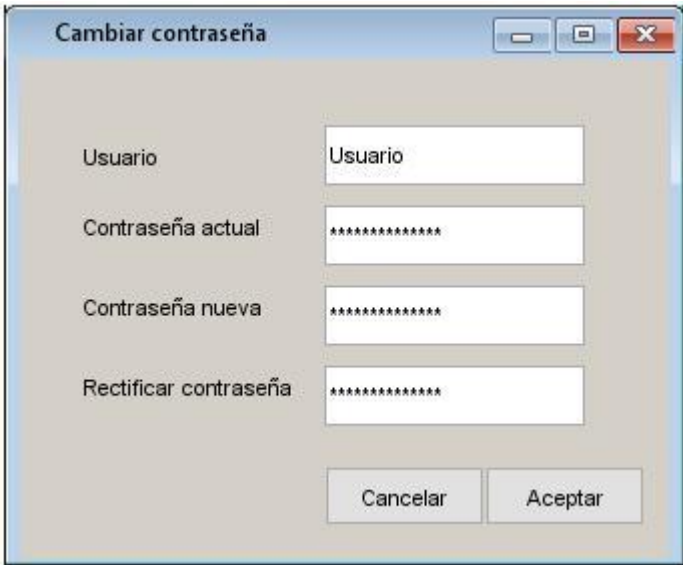
<b>Historia de usuario.</b>	
Número: HU1	Usuarios: Administradores
Nombre de la HU: Autenticar usuario administrador de la aplicación.	
Prioridad: Alta	Riesgo en Desarrollo: Media.
Tiempo estimado: 1 semana	
Programador responsable: José Eduardo Ramos Marquez.	
Descripción: Permitir la autenticación de los usuarios administradores en la aplicación.	
Observaciones:	
Prototipo de interfaz de usuario:	
	

Tabla 5. historia de usuario: Cambiar contraseña de usuario administrador  
(Elaboración propia)

Historia de usuario	
Número: HU2	Usuarios: Administradores
Nombre de la HU: Cambiar contraseña de usuario administrador.	
Prioridad: Alta	Riesgo en Desarrollo: Media.
Tiempo estimado: 1 semana	
Programador responsable: José Eduardo Ramos Marquez	
Descripción: El sistema debe permitir tanto añadir como eliminar los usuarios con permisos administrativos sobre la herramienta, así como la posibilidad de cambiar su contraseña.	
Observaciones:	
Prototipo de interfaz de usuario:	
 The image shows a software window titled "Cambiar contraseña" (Change Password). It contains four input fields: "Usuario" (User) with the text "Usuario", "Contraseña actual" (Current Password) with asterisks, "Contraseña nueva" (New Password) with asterisks, and "Rectificar contraseña" (Repeat Password) with asterisks. At the bottom, there are two buttons: "Cancelar" (Cancel) and "Aceptar" (Accept).	

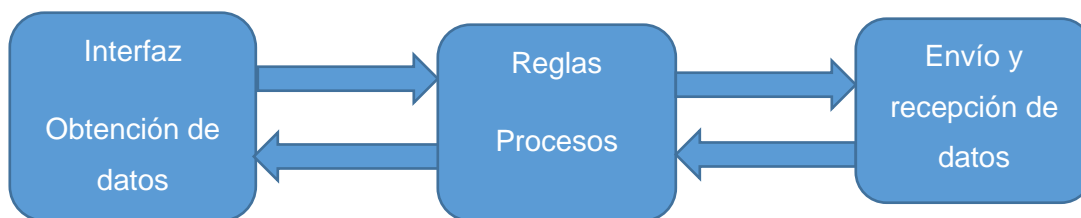
## 2.5 Arquitectura por capas

El diseño arquitectónico es la primera etapa del proceso de diseño y representa un enlace crítico entre los procesos de ingeniería de diseño y requerimientos. El proceso de diseño

# Capítulo 2

arquitectónico está relacionado con el establecimiento de un marco estructural básico que identifica los principales componentes de un sistema y las comunicaciones entre estos componentes. Algunas de las ventajas de diseñar explícitamente y documentar la arquitectura son la comunicación con los stakeholders, análisis del sistema y la reutilización a gran escala (Pressman 2002).

Para la implementación de la aplicación de escritorio de la propuesta de solución se define la arquitectura por capas. La arquitectura por capas permite el desarrollo incremental del sistema permitiendo el consumo de los servicios prestados por una capa finalizada sin estar el software completado, esta arquitectura también soporta bien los cambios y es portable, en la medida en que su interfaz permanezca sin cambios una capa puede ser remplazada por otra capa equivalente y si la interfaz sufre cambios solo se verá afectada la capa adyacente (Sommerville 2005). Se definen 3 capas como muestra la figura a continuación:



**Ilustración 1. Arquitectura (Elaboración propia)**

- Interfaz del cliente: es la encargada de la comunicación del usuario con la herramienta mostrándole las funcionalidades y permitiéndole el trabajo y la interacción con la misma.
- Lógica del negocio: es la capa donde se encontrará alojado las reglas del negocio, es la encargada de procesar las peticiones del usuario y el manejo de los datos almacenados o recibidos por la capa de persistencia y acceso a datos.
- Acceso a datos: Esta capa es la encargada de la comunicación con la API y de enviar estos datos a la capa superior de Lógica de negocio para el procesamiento de los mismos.

Para la comunicación con el servidor y el acceso a datos se define el protocolo de comunicación SSH en el puerto 22 (Suárez 2007).

## 2.6 Patrones de diseño

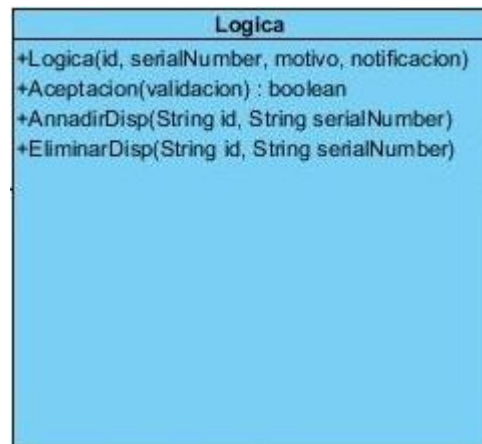
Un patrón de diseño describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico con el objetivo de proporcionar una descripción que permita a un diseñador determinar si el patrón es aplicable al trabajo en cuestión, si puede volverse a usar (con lo que se ahorra tiempo de diseño) y si sirve como guía para desarrollar un patrón distinto en funciones o estructura (Pressman 2002).

Los patrones GoF (Gang of Four) conllevan a una solución implementable con su propio diagrama de clases mediante el análisis de variables tales como la cantidad de desarrolladores, diseños UML, etc. (Guerrero, Suárez y Luz 2013). Estos patrones comúnmente utilizados en aplicación Web, permiten como todo patrón de diseño, ahorrar tiempo y recursos a la hora de implementar una solución informática. Este patrón no fue tomado en cuenta debido a que sería engorroso el proceso de desarrollo ya que el número de desarrolladores se limita a 1 y no existen los diseños UML, además, el negocio se encuentra bien definido, existe una gran comunicación con el cliente y el proyecto es relativamente pequeño. Por lo que se definió que lo óptimo sería la utilización de los patrones GRASP por lo antes planteado.

En lugar de estos se opta por la utilización de los patrones GRASP, GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades), estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman 2003).

### Patrones GRASP

**Experto:** este patrón consiste en asignar una responsabilidad determinada a la clase que tenga la mayor cantidad de información para hacer esta tarea; la clase “experta” en información (Larman 2003). Este patrón es aplicado a la clase Lógica, ilustración 3, debido a que esta es la que tiene acceso a todas las clases necesarias y por ello a la información convirtiéndose en experta para esta responsabilidad facilitando así el entendimiento y bajo acoplamiento.



**Ilustración 2. Clase experta (Elaboración propia)**

**Creador:** este patrón es el encargado de crear instancias de objetos de la “clase A” en el “clase B” si se cumple que B contiene a A, B agrega a A, B contiene los datos de inicialización de A, B registra a A o B utiliza a A muy de cerca (Larman 2003). Este patrón es utilizado en la clase Lógica, ilustración 3, que usa directamente los objetos derivados de la clase Datos para su validación posterior.

**Bajo acoplamiento:** este patrón busca dar soporte a poca dependencia y buscar una mayor reutilización, medir la fuerza en que una clase está conectada con otra, la conoce y recurre a ella(Larman 2003). En el diseño de la propuesta de solución existe un bajo acoplamiento debido a que una clase solo depende de otra para realizar su función, ver diagrama de clases, ilustración 4.

**Alta cohesión:** este patrón buscar asignar las responsabilidades con el fin de mantener baja y controlada la complejidad(Larman 2003). Las clases definidas en el diseño poseen una alta cohesión debido a que la carga de responsabilidades y funciones se encuentra dividida entre ellas permitiéndoles mediante la estrecha relación entre ellas no realizar un trabajo excesivo, ver diagrama de clases, ilustración 4.

## 2.7 Diagrama de paquetes

El sistema se encuentra dividido en agrupaciones lógicas o paquetes unidos por dependencias de forma tal que el modelo, la interfaz gráfica y los servicios se relacionen con el dominio del negocio solamente como se muestra a continuación:

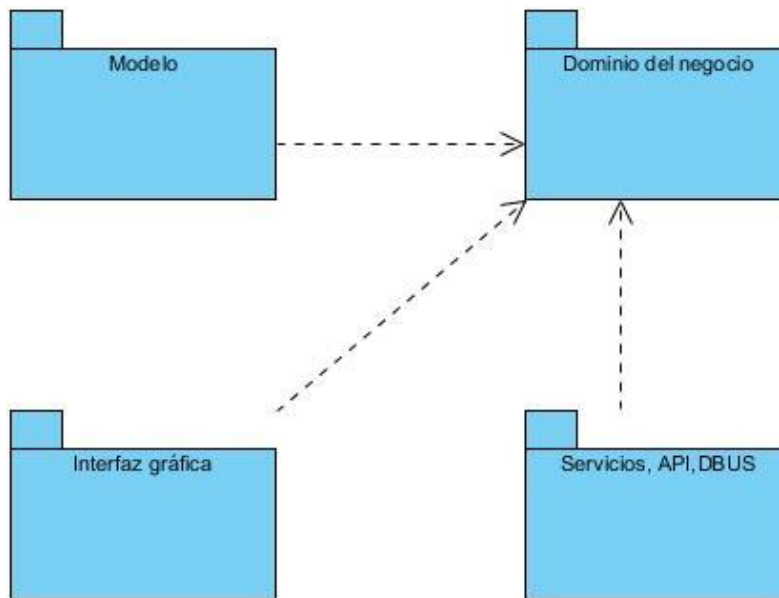


Ilustración 3. Diagrama de paquetes (Elaboración propia)

## 2.8 Diagrama de clases

Con el fin de describir la estructura del sistema se realizó un diagrama de clases en el cual se representan las clases, atributos y métodos, así como la relación entre estas como se muestra en la ilustración 4.

# Capítulo 2

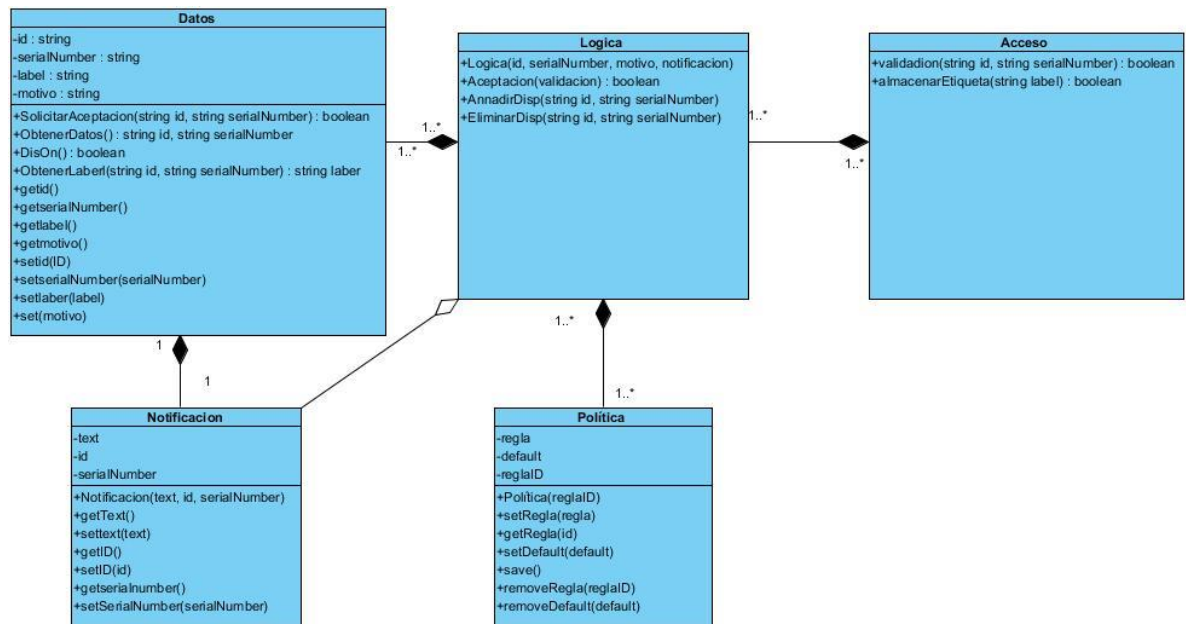


Ilustración 4. Diagrama de clases (Elaboración propia)

## Conclusiones del capítulo

Con la elaboración del capítulo se pudieron documentar los requisitos de una herramienta que gestione de manera remota y centralizada el acceso a los puertos USB, para dar solución al problema científico planteado y a las necesidades del cliente. Se garantiza acceso a la configuración del sistema de manera segura y confiable a los usuarios de administración con la asignación de una contraseña. Se definieron los requisitos no funcionales necesarios para el óptimo funcionamiento de la herramienta a desarrollar. Se validan las necesidades del cliente mediante las técnicas de validación. El sistema a implementar reemplazará a la actual solución, convirtiéndose en una herramienta importante de manejo rápido y preciso que contribuye a fortalecer la política de seguridad informática para preservar la confidencialidad, integridad y disponibilidad de la información.



## Capítulo 3. Implementación y estrategia de pruebas.

El presente capítulo está enfocado en las disciplinas de implementación y en el diseño de las pruebas con el fin de satisfacer las necesidades del cliente.

Para dar respuesta a la pregunta científica ¿Qué pruebas de software aplicar para la evaluación de la herramienta que, mediante una gestión centralizada, restrinja y controle el acceso a los puertos USB en GNU/Linux Nova? se define una estrategia de pruebas.

### 3.1 Implementación

Se procede a materializar en código los resultados obtenidos en la fase de análisis y diseño con el fin de implementar la herramienta que controla y restringe el acceso a los puertos USB en las estaciones de trabajo con GNU/Linux Nova y satisfacer así las necesidades del cliente, desarrollando los requisitos funcionales definidos por este.

### 3.2 Diagrama de componentes

Un diagrama de componentes permite modelar la estructura del software y la dependencia entre componentes, describiendo elementos físicos del sistema y sus relaciones (Letelier 2002). Con el fin de modelar la estructura de la solución informática para la restricción y control de puertos USB se desarrolló el siguiente diagrama de componentes donde se define la dependencia entre los elementos físicos del sistema, el componente Lógica es el encargado de proveer la interfaz gráfica de la herramienta necesaria por el componente Datos para su correcto funcionamiento.



Ilustración 5. Diagrama de componentes (Elaboración propia)

## 3.3 Estándar de codificación

El objetivo de utilizar un estándar de codificación consistente es la legibilidad del código escrito y la calidad del mismo. Mientras más legible, más fácil será para otra persona realizar acciones de mantenimiento o encauzar nuevos desarrollos y mejoras, además de que resulta mucho más sencillo encontrar y corregir errores en un código normalizado para la implementación de la propuesta de solución. Se utilizará el estándar de codificación planteado en el libro “C++: Guía completa para principiantes Aprende Todo sobre el C++ de La A la Z” (Smith 2020). Algunos ejemplos de estos son:

1. Variables y comentarios en español y ajustadas a su significado evitando el uso de abreviaturas, lo mismo se aplica a la identificación de las clases. (Ejemplo: dispositivos\_denegados, dispositivos\_permitidos.)
2. Correcta indentación de los bloques de código agrupándolos y representándolos de una forma clara y concisa de acuerdo a significado y relación. (Ejemplo: 

```
if(horas<24&&minutos<60&&segundos<60){return true;} else{return false;}
```

)
3. Utilización de un espaciado entre los operadores binarios y caracteres para la mejor comprensión del código.
4. Para las variables contables como emplear su nombre completo o una abreviatura amigable, ejemplo, dispositivo1 o disp1, no utilizar d1 o di1.
5. Utilizar terminología aplicable al dominio.
6. Utilización de llaves en las estructuras de control aun teniendo estas una sola instrucción en su interior.
7. Utilización de una forma consistente de apertura y cierre de llaves de manera general.

## 3.4 Despliegue de la solución propuesta

Para el correcto funcionamiento de la solución se propone el siguiente diagrama de despliegue en el que es necesario un servidor que almacene a la API encargada de la validación de la información obtenida por la estación de trabajo y enviada a esta mediante el protocolo de comunicación SSH por el puerto 22 (Suárez 2007).

# Capítulo 3

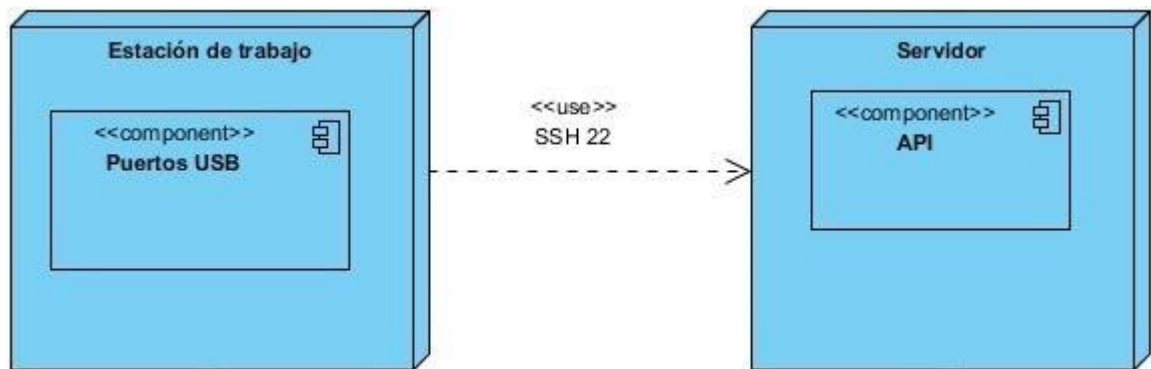


Ilustración 6. Diagrama de despliegue (Elaboración propia)

## 3.5 Requerimientos del hardware

A continuación, se desglosan los requerimientos de hardware recomendados. Estos requisitos no funcionales son fundamentales una vez desplegada la herramienta para la óptima ejecución de la misma.

Tabla 6. Requerimientos de Hardware (Elaboración propia)

Equipo	CPU	RAM	Disco duro
Servidor.	Dual Core 1.6 GHz o mayor.	2 Gb.	10 Gb.
Estación de trabajo.	Procesador x86 o x86_64 a 1GHz o mayor.	128 Mb.	10 Gb.

## 3.6 Requerimientos del software

Para el óptimo funcionamiento de la herramienta una vez desplegada se definen estos requerimientos de software:

- Sistema Operativo Nova Servidor 6.0.
- Sistema Operativo Nova Escritorio 5.0.

## 3.7 Estrategia de pruebas

La prueba es el proceso que se ejecuta en un programa con objeto de encontrar un error, siendo esta la prueba exitosa (Pressman 2002). Con el fin de validar lo implementado y garantizar la calidad de la herramienta se realiza una estrategia de pruebas pertinente para evaluar y analizar las salidas y errores encontrados. Esta estrategia de pruebas tiene como objetivo la detección de errores y redundancias en el código, la completitud de la herramienta y la satisfacción del cliente descubriendo la mayor cantidad de errores mediante el diseño de varios tipos de pruebas, como las pruebas de caja blanca, de aceptación, las pruebas unitarias, pruebas de integración y pruebas de humo.

Estas pruebas no se pueden realizar debido a factores externos como la disponibilidad de internet, comunicación y tecnología, por lo que se realizan hasta el nivel de diseño de la estrategia.

### 3.7.1 Pruebas funcionales

Las pruebas funcionales son aquellas que se le aplican al producto final y permiten detectar hasta que puntos el producto no cumple sus especificaciones, es decir, comprobar su funcionalidad (Lewis 2009). En este aspecto se diseñan varios tipos de prueba como las pruebas unitarias, las pruebas de integración, las pruebas de humo y las pruebas de aceptación.

**Una prueba unitaria** es una forma de comprobar el correcto funcionamiento de una unidad de código. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, que si el estado inicial es válido, entonces el estado final es válido también (Barrientos 2014).

Esta prueba se realiza a las principales funcionalidades de la herramienta como la obtención de los datos de los dispositivos conectados y el control de acceso de los mismos a la par de la implementación de la misma como parte del proceso de desarrollo de la

# Capítulo 3

herramienta y con el fin de identificar los errores según fueran apareciendo evitando así la acumulación de los mismos o la aparición de otros una vez corregidos estos al final de la implementación de la herramienta.

**Las pruebas de integración** son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño (Pressman 2002). Con esta prueba se busca que una vez integrados los componentes no se altere la interfaz gráfica y exista una correcta comunicación entre ellos, brindando los datos y la información necesaria para el correcto funcionamiento de cada una de sus funcionalidades.

**Las pruebas de humo** son aquellas pruebas que pretenden evaluar la calidad de un producto de software previo a una recepción formal, ya sea al equipo de pruebas o al usuario final. Estas deben realizarse diariamente para que las incompatibilidades y otros errores paralizantes puedan descubrirse tempranamente lo que reduce la posibilidad de impacto severo sobre el calendario (Pressman 2002). Para la realización de estas pruebas se diseñarán varios casos de prueba para medir que las entradas y salidas son aceptadas correctamente tomando como base que la integridad de la información externa no varía.

**Las pruebas de aceptación**, también llamadas pruebas del cliente, son un tipo de prueba de caja negra llevada a cabo, mediante casos de uso y escenario centrándose en las características y funcionalidad general del sistema que son visibles y revisables por el cliente (Pressman 2002).

Cuando se construye un software a la medida para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos, realizada por el usuario final en vez de por los ingenieros de software, estas pueden ir desde una prueba informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente (Pressman 2002).

En la tabla 7 se muestra un caso de prueba de aceptación para la historia de usuario “Autenticar usuario administrador de la aplicación”, definiéndose el nombre de la persona a realizar la prueba, la descripción y las condiciones de ejecución de la misma. Las entradas o pasos para su ejecución, así como resultado esperado y evaluación de la prueba.

# Capítulo 3

Tabla 7. Caso de prueba de aceptación para la historia de usuario: "Autenticar usuario administrador de la aplicación" (Elaboración propia)

<b>Caso de prueba de Aceptación.</b>
<b>Nombre de la historia de usuarios:</b> Autenticar usuario administrador de la aplicación.
<b>Nombre de la persona que realiza la prueba:</b>
<b>Descripción de la prueba:</b> La herramienta permite o deniega el acceso a la misma en del usuario autenticado en dependencia de la validez de sus credenciales.
<b>Condiciones de ejecución:</b> La herramienta debe estar en ejecución y con acceso al servidor.
<b>Entrada/Pasos de ejecución:</b>  <ol style="list-style-type: none"><li>1. Ejecutar la aplicación.</li><li>2. Seleccionar la opción autenticar usuario.</li></ol>
<b>Resultado esperado:</b> La herramienta permite o deniega el acceso al usuario de la estación de trabajo teniendo en cuenta la validez de sus credenciales.
<b>Evaluación de la prueba:</b>

Con la aplicación de estas pruebas se pretende evaluar el resultado de la implementación del sistema informático, garantizando que se cumplan las expectativas del cliente proyectadas en los requisitos definidos y las funcionalidades planteadas.

## 3.7.2 Pruebas no funcionales

**Las pruebas de rendimiento** se diseñan para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. La prueba del

# Capítulo 3

rendimiento ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas (Pressman 2002).

Para la realización de esta prueba se llevarán a cabo pruebas de carga y estrés permitiendo identificar cuantas peticiones de aceptación simultáneas puede soportar el sistema y comprobar los límites de este.

Para el proceso de simulación necesario para realizar la prueba se utilizará la herramienta Netdata que permite visualizar y monitorear métricas en tiempo real, optimizada para acumular todo tipo de datos, como uso de CPU, actividad de disco, consultas SQL, etc. Desarrollada para los sistemas operativos Linux de código abierto, licencia gratuita e interfaz amigable al usuario (Netdata 2020).

**Las pruebas de usabilidad por parte del usuario** evalúan el grado en el cual estos pueden interactuar efectivamente con la herramienta y el grado en el que esta guía las acciones del usuario, proporciona retroalimentación significativa y refuerza un enfoque de interacción consistente. En lugar de enfocarse atentamente en la semántica de algún objetivo interactivo, las revisiones y pruebas de usabilidad se diseñan para determinar el grado en el cual la interfaz del software facilita la vida del usuario (Pressman 2002).

## Conclusiones del capítulo

Con el desarrollo de este capítulo se definieron los estándares de codificación para garantizar la comprensión del código, se diseñaron y documentaron pruebas con el fin de asegurar un software de calidad que cumple con los requisitos y funcionalidades definidas por el cliente. Aun cuando no pudo ejecutarse toda la estrategia de prueba diseñada se definieron las bases para la verificación de que el sistema responde adecuadamente cuando se ingresen datos incorrectos y tiene un apropiado nivel de tolerancia ante las acciones inesperadas.

## Conclusiones generales

Luego de la investigación para el desarrollo de la herramienta que permite restringir y controlar el acceso de los puertos USB en las distribuciones cubanas de GNU/Linux Nova, se arriban a las siguientes conclusiones:

- El análisis de la situación problemática existente y el estudio de las soluciones para la restricción del acceso a la información mediante los puertos USB, demostró la necesidad de desarrollar una herramienta para la gestión remota y centralizada del acceso a los puertos USB en las distribuciones cubanas de GNU/Linux Nova.
- La realización de mecanismos de obtención y validación de requisitos permitió la obtención y refinamientos de los requisitos necesarios para satisfacer las necesidades del cliente de una herramienta de gestión remota y centralizada del acceso a los puertos USB en las distribuciones cubanas de GNU/Linux Nova.
- La correcta selección de la metodología, las herramientas y tecnologías para el desarrollo de la solución, permitió el análisis y diseño de una herramienta que gestione de manera remota y centralizada del acceso a los puertos USB en las distribuciones cubanas de GNU/Linux Nova.
- La programación de los requisitos definidos en el análisis y diseño permitirá la implementación de una herramienta para la gestión remota y centralizada del acceso a los puertos USB en las distribuciones cubanas de GNU/Linux Nova
- Las pruebas diseñadas permitirán detectar de manera temprana las no conformidades y errores, posibilitando la corrección de los mismos con resultados satisfactorios, y contribuyendo a la calidad del producto final.



## Recomendaciones

Se recomienda implementar los requisitos diseñados para el desarrollo de una herramienta que permita gestionar de manera remota y centralizada el acceso a los puertos USB en las distribuciones cubanas de GNU/Linux Nova.

Se recomienda realizar las pruebas diseñadas para la recolección de resultados que permitan optimizar las nuevas versiones.

## Bibliografía

- ALARCÓN, A. y SANDOVAL, E., 2008. Herramientas CASE para ingeniería de Requisitos. *Cultura Científica JdC* [en línea], pp. 70-74. Disponible en: [jdc.edu.co](http://jdc.edu.co).
- ALBALAT, M., FÍRVIDA, A., GARCÍA, D. y MACHÍN, J.L., 2012. NOVA LIGERO INCREMENTANDO LA SOCIO-ADAPTABILIDAD DE LA DISTRIBUCION CUBANA DE GNULINUX. [en línea]. La Habana: Disponible en: <https://www.researchgate.net/publication/283291919>.
- AMBLER, S., 2005. The Agile Unified Process (AUP). [en línea]. Toronto: Disponible en: [https://www.researchgate.net/profile/Scott\\_Ambler/publication/267259668\\_The\\_Agile\\_Unified\\_Process\\_AUP/links/55003e7e0cf28e4ac347ee37/The-Agile-Unified-Process-AUP.pdf](https://www.researchgate.net/profile/Scott_Ambler/publication/267259668_The_Agile_Unified_Process_AUP/links/55003e7e0cf28e4ac347ee37/The-Agile-Unified-Process-AUP.pdf).
- AVENIA, C.A., 2017. *Fundamentos de seguridad informática* [en línea]. 1ra. Bogotá: Fondo editorial Areandino. ISBN 9762458219735. Disponible en: [digitk.areandina.edu.co](http://digitk.areandina.edu.co).
- BARRIENTOS, P.A., 2014. *Enfoque para pruebas de unidad basado en la generación aleatoria de objetos*. S.l.: Universidad Nacional de La Plata.
- BLEZ, E., PIERRA, A., FÍRVIDA, D., TEJERA, C., CÁCERES, H., PARKER, A.A. y EVARISTO, M., 2014. USB Device Management in GNU/Linux Systems. *USB Device Management in GNU/Linux Systems* [en línea]. S.l.: s.n., pp. 218-225. Disponible en: [http://link.springer.com/10.1007/978-3-642-55128-4\\_33](http://link.springer.com/10.1007/978-3-642-55128-4_33).
- CASAI, P. y REINOSO, A., 2017. La privacidad en la sociedad de la información. *Tecnología y desarrollo*, vol. 15.
- CENTRO DE SOLUCIONES LIBRES, 2017. *PPUSB*. 2017. La Habana: Universidad de las Ciencias Informáticas. 1.0.
- CHACON, S. y STRAUB, B., 2014. *Pro Git* [en línea]. 2nd. S.l.: s.n. ISBN 978-1484200773. Disponible en: <https://git-scm.com/book/en/v2>.
- FUENTES, L. y VALLECILLO, A., 2004. Una Introducción a los Perfiles UML. . Málaga:
- GUERRERO, C.A., SUÁREZ, J.M. y LUZ, E., 2013. Patrones de Diseño GOF (The Gang of

Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*, vol. 24.

HERNÁNDEZ-SAMPIERI, R. y TORRES, C.P.M., 2018. *METODOLOGÍA DE LA INVESTIGACIÓN* [en línea]. 1ra. Mexico D.F: McGraw-Hill Interamericana. ISBN 124125369856962. Disponible en: <https://www.academia.edu/download/38911499/Sampieri.pdf>.

INFOCOMPUTER, 2020. Qué es el Acceso Remoto y cómo administrarlo. [en línea]. Disponible en: <https://www.info-computer.com/blog/acceso-remoto-administrarlo/>.

KAISLER, S.H., 2005. *Software Paradigms*. S.I.: John Wiley & Sons, Inc. ISBN 456612228272512.

LARMAN, C., 2003. *UML y Patrones* [en línea]. 2da. Madrid, España: s.n. Disponible en: [ic.info.unlp.edu.ar](http://ic.info.unlp.edu.ar).

LETELIER, P., 2002. Desarrollo de Software Orientado a Objeto usando UML. [en línea]. Valencia: Disponible en: <http://www.dsic.upv.es/~uml>.

LEWIS, W., 2009. *Software testing and continuous quality improvement* [en línea]. 3da. New York: Taylor and Francis Group. ISBN 9781420080735. Disponible en: [books.google.com](https://books.google.com).

MICROSOFT, 2020. *MICROSOFT VISUAL STUDIO CODE* [en línea]. 2020. S.I.: 2020. 1.49. Disponible en: <https://code.visualstudio.com/docs>.

MINISTERIO DE LA INFORMÁTICA Y LAS COMUNICACIONES, 2007. RESOLUCION No. 127 /2007. *Gaceta oficial de la República de Cuba*. La Habana:

NETDATA, 2020. Monitor everything in real time – for free. [en línea]. Disponible en: <https://www.netdata.cloud/>.

NEWSOFTWARES.NET, 2019. USB Block. [en línea]. Disponible en: <https://www.newsoftwares.net/usb-block/>.

ORTÍZ, L.D.S.J.M.F.A., 2018. Sistema centralizado para la protección de puertos USB en Nova. . La Habana:

PARADIGM, V., 2020. *Visual Paradigm* [en línea]. 2020. S.I.: s.n. Disponible en:

- <https://www.visual-paradigm.com/>.
- PEREZ-GONZALEZ, I., 2007. *USBGuard* [en línea]. 2007. S.I.: s.n. Disponible en: <https://usbguard.github.io/>.
- PÉREZ, J. y GARDEY, A., 2015. Definición de Puerto USB. *Definicion.de* [en línea]. Disponible en: <https://definicion.de/puerto-usb/>.
- PÉREZ, J. y MERIÑO, M., 2020. Definición de Memoria flash. *Definicion.de* [en línea]. Disponible en: <https://definicion.de/memoria-flash/>.
- PIERRA, A., 2011. *Conceptualización y Reestructuración estratégica de la Distribución Cubana GNU/Linux Nova*. S.I.: Universidad de las Ciencias Informáticas.
- PRESSMAN, R.S., 2002. *Ingeniería de Software, un enfoque práctico*. 7ma. Mexico D.F: McGraw-Hill. ISBN 978-607-15-0314-5.
- REAL ACADEMIA ESPAÑOLA, 2019. Diccionario de la lengua española. *Real Academia Española* [en línea]. Disponible en: <https://dle.rae.es/control?m=form>.
- RODRÍGUEZ, A. y PÉREZ, A.O., 2017. Métodos científicos de indagación y de construcción del conocimiento. *Rev. esc.adm.neg.* [en línea], vol. 82, pp. 199. Disponible en: <https://doi.org/10.21158/01208160.n82.2017.1647>.
- ROSERO, I., 2020. Historia de los puertos USB. *Historia de los puertos USB* [en línea]. Disponible en: <https://es.calameo.com/books/000444167b0a02557397f>.
- SÁNCHEZ, T.R., 2015. *Metodología de desarrollo para la actividad productiva de la UCI*. La Habana: Universidad de las Ciencias Informáticas.
- SHOTTS, W.E., 2019. *The Linux Command Line: a complete introduction* [en línea]. 2. S.I.: William Pollock. Disponible en: [https://drive.google.com/file/d/0Bz\\_k22Jugr3UOU5U1VzQzBRVEE/view](https://drive.google.com/file/d/0Bz_k22Jugr3UOU5U1VzQzBRVEE/view).
- SMITH, B., 2020. *C++: Guía completa para principiantes Aprende Todo sobre el C++ de La A la Z*. Spanish. S.I.: s.n. ISBN B087L1VXHK.
- SOFTONIC, 2020. BuduLock. [en línea]. Disponible en: <https://budulock.softonic.com/>.
- SOMMERVILLE, I., 2005. *Ingeniería del software* [en línea]. 7ma. Madrid: s.n. [Consulta: 13

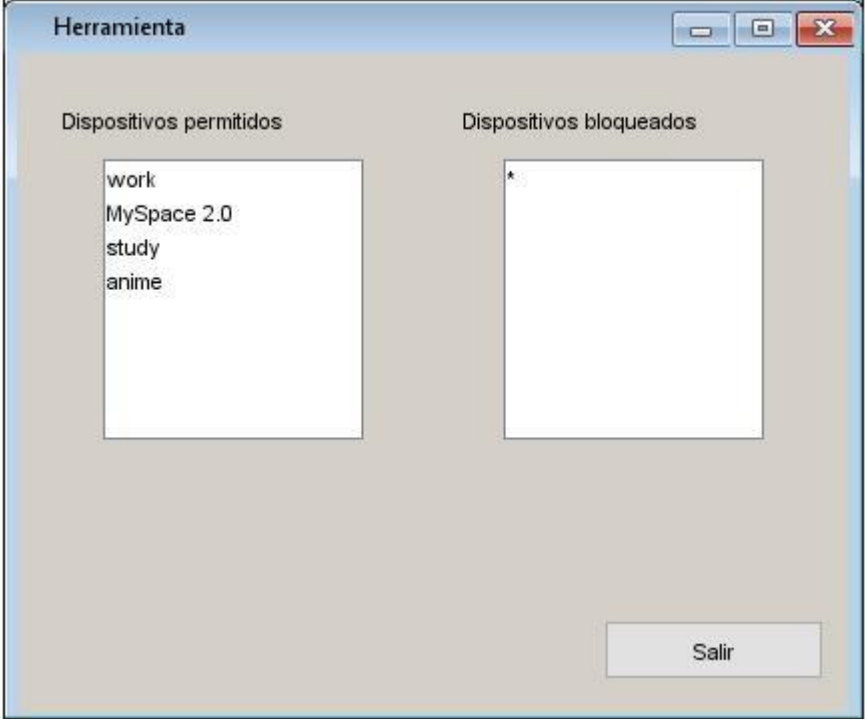
febrero 2020]. ISBN 84-7829-074-5. Disponible en:  
[https://books.google.com.cu/books?hl=es&lr=&id=gQWd49zSut4C&oi=fnd&pg=PR14&dq=sommerville+cap+11&ots=s772wmxuqi&sig=1gW-CCD0Z-wjk1zLsl8iXFdOrpY&redir\\_esc=y#v=onepage&q&f=true](https://books.google.com.cu/books?hl=es&lr=&id=gQWd49zSut4C&oi=fnd&pg=PR14&dq=sommerville+cap+11&ots=s772wmxuqi&sig=1gW-CCD0Z-wjk1zLsl8iXFdOrpY&redir_esc=y#v=onepage&q&f=true).

SUÁREZ, R., 2007. Conexiones remotas con SSH. *Todo linux: la revista mensual para entusiastas de GNU/LINUX*,

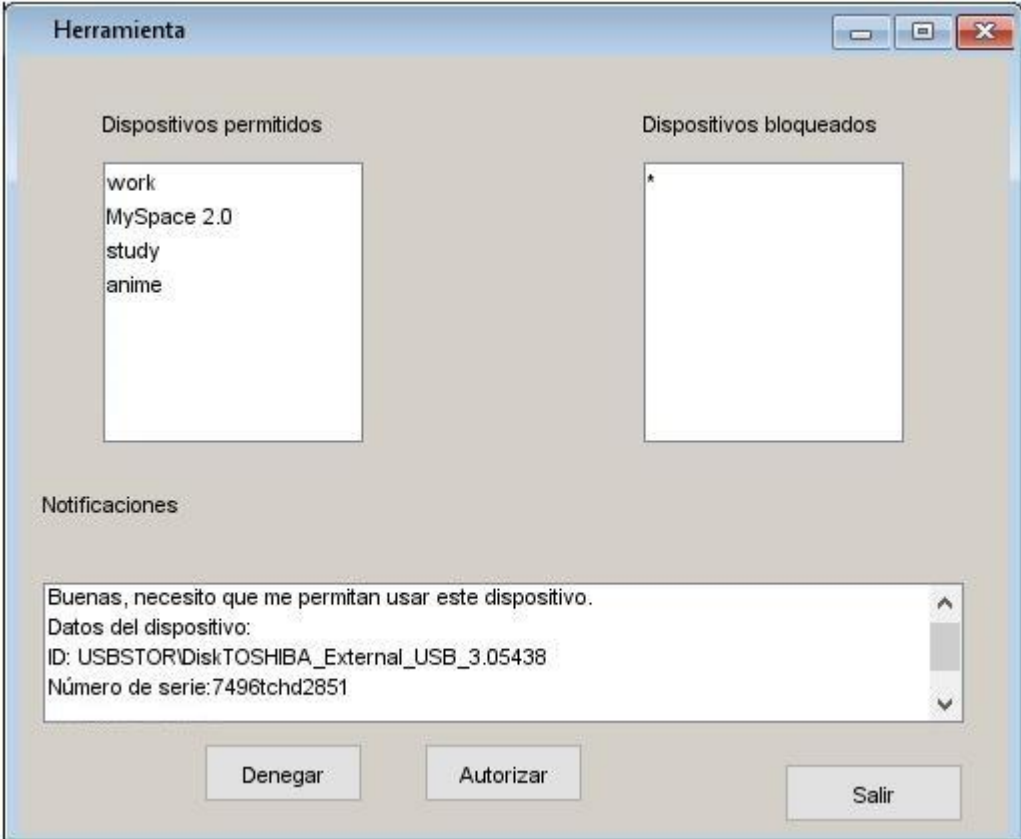
SYSTOOLS, 2020. SysTools USB Blocker Software. [en línea]. Disponible en:  
<https://www.systoolsgroup.com/usb-blocker.html>.

# Anexos

## ANEXO 1. HISTORIA DE USUARIO #3 "PERMITIR EL USO DEL DISPOSITIVO CONECTADO A LA ESTACIÓN DE TRABAJO" (ELABORACIÓN PROPIA)

Historia de usuario	
Número: HU3	Usuarios: Administradores
Nombre de la HU: Permitir el uso del dispositivo conectado a la estación de trabajo.	
Prioridad: Alta	Riesgo en Desarrollo: Alta.
Tiempo estimado: 4 semana	
Programador responsable: José Eduardo Ramos Marquez	
Descripción: El sistema debe permitir el uso del dispositivo de almacenamiento conectado al sistema y mostrar la etiqueta del mismo en la lista de dispositivos conectados permitidos.	
Observaciones:	
Prototipo de interfaz de usuario:	
	

**ANEXO 2. HISTORIA DE USUARIO #4. "NOTIFICAR AL ADMINISTRADOR DEL DISPOSITIVO CONECTADO NO PRESENTE EN LA BASE DE DATOS CENTRAL" (ELABORACIÓN PROPIA)**

<b>Historia de usuario</b>	
Número: HU4	Usuarios: Administradores
Nombre de la HU: Notificar al administrador del dispositivo conectado no presente en la base de datos del servidor central.	
Prioridad: Alta	Riesgo en Desarrollo: Alta.
Tiempo estimado: 4 semana	
Programador responsable: José Eduardo Ramos Marquez	
Descripción: El sistema debe enviar una notificación pidiendo la autorización del dispositivo conectado si este no está autorizado previamente en el servidor, esta notificación estará compuesta por el número de serie e identificador del dispositivo conectado.	
Observaciones:	
Prototipo de interfaz de usuario:	
	

**ANEXO 3. CASO DE PRUEBA DE ACEPTACIÓN PARA LA HISTORIA DE USUARIO: "CAMBIAR CONTRASEÑA DE USUARIO ADMINISTRADOR" (ELABORACIÓN PROPIA)**

<b>Caso de prueba de Aceptación.</b>
<b>Nombre de la historia de usuarios:</b> Gestionar usuario administrador.
<b>Nombre de la persona que realiza la prueba:</b>
<b>Descripción de la prueba:</b> El sistema debe permitir el cambio de contraseña del usuario administrador autenticado en la misma.
<b>Condiciones de ejecución:</b> La herramienta debe estar en ejecución, con acceso al servidor y el usuario debe estar autenticado en la misma como administrador.
<b>Entrada/Pasos de ejecución:</b>  <ol style="list-style-type: none"><li>1. Ejecutar la aplicación.</li><li>2. Autenticarse en la herramienta.</li><li>3. Seleccionar la opción de cambiar contraseña.</li></ol>
<b>Resultado esperado:</b> La herramienta permite el cambio de contraseña en caso de que los parámetros introducidos sean correctos.
<b>Evaluación de la prueba:</b>

**ANEXO 4. CASO DE PRUEBA DE ACEPTACIÓN PARA LA HISTORIA DE USUARIO: " PERMITIR EL USO DEL DISPOSITIVO CONECTADO A LA ESTACIÓN DE TRABAJO" (ELABORACIÓN PROPIA)**

<b>Caso de prueba de Aceptación.</b>
--------------------------------------



<b>Nombre de la historia de usuarios:</b> Permitir el uso del dispositivo conectado a la estación de trabajo.
<b>Nombre de la persona que realiza la prueba:</b>
<b>Descripción de la prueba:</b> El sistema debe validar que los datos del dispositivo conectado se encuentren entre los dispositivos permitidos en la lista blanca del servidor permitiendo el montaje de la unidad.
<b>Condiciones de ejecución:</b> La herramienta debe estar en ejecución y con acceso al servidor.
<b>Entrada/Pasos de ejecución:</b> <ol style="list-style-type: none"> <li>1. Ejecutar la aplicación.</li> <li>2. Conectar el dispositivo USB a la estación de trabajo.</li> </ol>
<b>Resultado esperado:</b> La herramienta permite el montaje y uso del dispositivo conectado.
<b>Evaluación de la prueba:</b>

**ANEXO 5. CASO DE PRUEBA DE ACEPTACIÓN PARA LA HISTORIA DE USUARIO: " NOTIFICAR AL ADMINISTRADOR DEL DISPOSITIVO CONECTADO NO PRESENTE EN LA BASE DE DATOS DEL SERVIDOR CENTRAL" (ELABORACIÓN PROPIA)**

<b>Caso de prueba de Aceptación.</b>
<b>Nombre de la historia de usuarios:</b> Notificar al administrador del dispositivo conectado no presente en la base de datos del servidor central.
<b>Nombre de la persona que realiza la prueba:</b>
<b>Descripción de la prueba:</b> El sistema debe validar que los datos del dispositivo conectado no se encuentren entre los datos de los dispositivos permitidos en la lista blanca del servidor y proceder a enviar una notificación al mismo con la

información de dicho dispositivo para ser aprobado o denegado por los administradores.

**Condiciones de ejecución:** La herramienta debe estar en ejecución y con acceso al servidor.

**Entrada/Pasos de ejecución:**

1. Ejecutar la aplicación.
2. Conectar el dispositivo USB a la estación de trabajo.

**Resultado esperado:** La herramienta bloquea el acceso del dispositivo USB y envía una notificación con los datos del mismo al servidor para ser aprobado no denegada por los administradores.

**Evaluación de la prueba:**