UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 1

Herramienta informática para la gestión centralizada de dispositivos USB en la distribución cubana GNU/Linux Nova

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Roxatne María Moreno Ramírez

Tutores:

M. Sc. Juan Manuel Fuentes Rodríguez

Ing. Miosotis Toledo Rodríguez

La Habana, 29 de noviembre de 2022

"Año 64 de la Revolución"

Agradecimientos

Primeramente, doy gracias a Dios por permitirme tener tan buena experiencia dentro de mi universidad, quiero agradecer infinitamente a mis padres, abuelos, mis hermanas y mi novio por apoyarme y darme ánimos para no rendirme y continuar luchando para lograr ser una profesional.

Agradecerles también a todos mis profesores y en especial a mis tutores por su ayuda, paciencia y dedicación durante todo este proceso.

Agradecer además a mis amigos por los buenos momentos.

Gracias a todos por ayudarme a conseguir este valioso logro.

Dedicatoria

Quiero dedicarles este logro a mis padres, mis abuelos y a mi hermana Ange, gracias por ser como son, porque han ayudado a construir y forjar la persona que soy, los amo.

Declaración de la autoría

Declaro por este medio que yo **Roxatne María Moreno Ramírez**, con carné de identidad **98100120677** soy el autor principal del trabajo titulado **"Herramienta informática para la gestión centralizada de dispositivos USB en la distribución cubana 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 ___29____ días del mes de _____noviembre_____ del ___2022_____

_____

Roxatne Moreno Ramírez

Autor

_____ _____

M. Sc. Juan Manuel Fuentes Rodríguez

Tutor

Ing. Miosotis Toledo Rodríguez

Tutor

Resumen

Con el objetivo de lograr la soberanía tecnológica, se desarrolla un proceso de migración hacia software libre en el que se emplean soluciones informáticas desarrolladas por el Centro de Software Libre (CESOL), perteneciente a la Universidad de las Ciencias Informáticas. Nova es una variante de GNU/Linux diseñado para estaciones de trabajo. Actualmente no existe una herramienta para la gestión centralizada de dispositivos USB para el sistema operativo GNU/Linux Nova. Por esto se hace necesario tener un control de los dispositivos USB que se conectan a una estación de trabajo para evitar que se puedan ingresar virus en la red o el robo de información sensible. La presente investigación tuvo como objetivo desarrollar una herramienta para la gestión centralizada de dispositivos USB en GNU/Linux Nova, para ello se utilizaron tecnologías en su mayoría libres, diferentes patrones de diseño y la arquitectura Modelo-Vista-Plantilla. Con las pruebas aplicadas se comprobó el correcto funcionamiento de la herramienta, demostrando que la misma satisface las necesidades del cliente.

Palabras claves: centralizar, gestión, gestión centralizada, Nova, USB (Universal Serial Bus).

Índice

Índice de figuras

Índice de tablas

Introducción

El avance tecnológico y la aparición de dispositivos como computadoras y teléfonos móviles han llevado a que se creen nuevas herramientas para darles vida y mejorar su funcionamiento. La informática a lo largo de los años ha mejorado la calidad de vida de las personas gracias a la creación y desarrollo de numerosas aplicaciones y softwares que son usados diariamente y que sirven de provecho para el desarrollo de la sociedad siendo uno de sus mejores aportes el de mantener a las personas informadas y actualizadas (1).

Una computadora es una máquina electrónica digital programable que ejecuta una serie de comandos para procesar datos de entrada, obteniendo convenientemente información que luego se envía a las unidades de salida. Está constituida por dos partes esenciales, el software que es su parte intangible, un conjunto de comandos que dan órdenes y se encarga del funcionamiento de esta, y el hardware que es los elementos físicos que ella necesita para funcionar (2).

Los periféricos son accesorios o equipos que se conectan a la unidad central de procesamiento (CPU) de una computadora; existen 4 tipos de periféricos, entre ellos están los periféricos de almacenamiento. Estos son dispositivos que permiten almacenar datos fuera de las computadoras, pero también compartirlos con ellas cuando sea necesario (3).

Los dispositivos USB (Universal Serial Bus) permiten almacenar datos extraídos de la computadora y también enviarle datos almacenados en ella, se caracterizan según su capacidad y su velocidad de transferencia de datos. Su capacidad puede ser 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 gigabyte (GB) y hasta 1 terabyte (1024 GB), actualmente existen 6 tipos de velocidades: USB 1.0(velocidad más baja), USB 1.1, USB 2.0, USB 3.0, USB 3.1 y USB 3.2(velocidad más alta) (4).

La información constituye uno de los activos más importantes de cualquier organización, por lo que se hace necesario preservar y proteger la confidencialidad, la disponibilidad e integridad de la información. Un robo de información, un error informático o un pirateo pueden traer consecuencias negativas para cualquier empresa, así como la paralización de la actividad (5).

El uso de dispositivos de almacenamiento portátiles va en aumento y plantea dos grandes preocupaciones para una organización: el robo de datos y la inyección de malware (código maligno, software malicioso o software malintencionado, es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora sin el consentimiento de su propietario) (6). Para un empleado descontento es muy fácil inyectar malware o robar información importante del negocio utilizando un dispositivo USB. Para evitar que ocurran tales situaciones, es necesario gestionar y controlar todos los dispositivos USB que se conecten a toda la red de computadoras.

La Universidad de las Ciencias Informáticas (UCI) como parte de su proceso de informatización de la sociedad cubana cuenta con 10 centros de desarrollo de software, estos conforman una red de trabajo colaborativo que opera bajo normas y procedimientos comunes, posibilitando la reutilización de componentes y eficiencia industrial (7).

Uno de sus centros es el Centro de Software Libre (CESOL), en él se desarrolla la Distribución Cubana de GNU/Linux Nova.

Nova es una distribución cubana de GNU/Linux, cuenta con 3 ramas: Nova Escritorio, Nova Ligero y Nova Servidor. Nova Escritorio es una variante de Nova diseñado para el trabajo en estaciones de trabajo con prestaciones relativamente modernas, propone un ambiente de trabajo libre de distracciones y exhibe un entorno familiar a Windows con el objetivo de reducir la resistencia al cambio en el proceso de migración (8).

En gran parte de las instituciones de los Organismos de la Administración Central del Estado se utiliza la distribución cubana GNU/Linux Nova. Debido a la necesidad de proteger la información que se almacena en las computadoras que ahí se utilizan, se considera que la mejor manera de mantener a salvo esta información es controlando los dispositivos mediante una herramienta tecnológica. Actualmente existen varias herramientas para el control de los dispositivos USB, desafortunadamente la gran mayoría son destinadas a usarse en Microsoft Windows como: DriveLock, DeviceLock, MyUSBonly entre otros. En Linux existen muy pocas herramientas capaces de gestionar y controlar las entradas de los dispositivos USB en los ordenadores, entre ellas está USBGuard y PPUSB, esta última desarrollada por CESOL (9).

Tanto PPUSB como USBGuard cumplen con varias de las características necesarias para proteger los puertos USB en Nova, pero usarlas resulta un proceso engorroso para los responsables de seguridad informática de las instituciones, ya que dichas herramientas están diseñadas para ser administradas de forma local, esto trae como consecuencia que el usuario desde su estación de trabajo, o computadora como también se le conoce, cuente con el privilegio de gestionar y controlar los dispositivos conectados y de esta forma podrá violar las políticas de seguridad informáticas establecidas.

Actualmente no existe una herramienta para la gestión centralizada de dispositivos USB para el sistema operativo GNU/Linux Nova en un local o área de trabajo. Se hace necesario controlar centralmente todos los dispositivos USB que se conectan a una estación de trabajo y si dicha estación de trabajo está conectada en red a otras, esto puede traer consigo que puedan ingresar virus en la red o el robo de información sensible. Por tal motivo es necesario tener un control

central de los dispositivos que tienen acceso a las estaciones de trabajo y por consiguiente a un área de trabajo.

Teniendo en cuenta la situación descrita anteriormente, se plantea como **problema de la investigación:** ¿Cómo facilitar la gestión centralizada de los dispositivos USB de un área de trabajo en la distribución cubana GNU/Linux Nova?

Del planteamiento anterior se deriva como **objeto de estudio** de esta investigación: el proceso de gestión centralizada de los dispositivos USB. Se propone como **campo de acción** la gestión centralizada de los dispositivos USB en las distribuciones GNU/Linux.

Teniendo en cuenta el problema a resolver se define como **objetivo general** desarrollar una herramienta informática que facilite la gestión centralizada de los dispositivos USB de un área de trabajo en la distribución cubana GNU/Linux Nova.

De este se derivan los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación sobre el proceso de gestión centralizada de dispositivos USB en un área de trabajo.
2. Diseñar una herramienta informática que facilite la gestión centralizada de los dispositivos USB de un área de trabajo en la distribución cubana GNU/Linux Nova.
3. Implementar una herramienta informática que facilite la gestión centralizada de los dispositivos USB de un área de trabajo en la distribución cubana GNU/Linux Nova.
4. Evaluar la herramienta informática que facilite la gestión centralizada de los dispositivos USB de un área de trabajo en la distribución cubana GNU/Linux Nova.

Para dar cumplimiento a los objetivos específicos planteados se definen las siguientes **preguntas científicas:**

1. ¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre el proceso de gestión centralizada de los dispositivos USB?

2. ¿Cuáles son los aspectos a tener en cuenta para realizar el diseño de una herramienta informática para la gestión centralizada de dispositivos USB?

3. ¿Qué componentes son necesarios implementar en el desarrollo de una herramienta informática para la gestión centralizada de dispositivos USB en GNU/Linux?

4. ¿Qué pruebas de software aplicar para la evaluación de la herramienta informática para la gestión centralizada de dispositivos USB en GNU/Linux?

Los **métodos de investigación** empleados para la realización de esas tareas son:

Métodos teóricos:

1. Método **analítico-sintético:**

   Análisis: Permite la descomposición mental del objeto o fenómeno en sus múltiples relaciones o componentes para facilitar su estudio.

   Síntesis: Establece mentalmente la unión entre las partes previamente analizadas, permite descubrir sus características generales y las relaciones esenciales entre ellas.

   Su objetivo en la investigación es analizar las teorías, documentos, etc., permitiendo la extracción de los elementos más importantes que se relacionan con el proceso de gestión centralizada de los dispositivos USB.

Métodos empíricos:

1. **Encuesta:** se realizó con el objetivo de recopilar información en cuanto al nivel de satisfacción de los usuarios potenciales de la solución desarrollada.
2. **Observación:** se sometió la herramienta a varias pruebas y se constató el logro del objetivo trazado.
3. **Entrevista:** para determinar qué elementos pueden ser reutilizados durante la gestión centralizada de dispositivos USB, además de evaluar la importancia que se le concede a la aplicación en este proceso.

La presente investigación se estructura en introducción, tres capítulos, conclusiones, bibliografía, recomendaciones y anexos que permiten comprender mejor la investigación realizada.

En el *Capítulo 1. Fundamentación teórica del proceso de gestión centralizada de los dispositivos USB*. En este capítulo se hace un análisis relacionado con la tecnología y la metodología que se utilizará para alcanzar el objetivo. Se analiza el estado actual referente a la gestión centralizada de dispositivos USB y se exponen los conceptos fundamentales relacionados. Se realiza un estudio de las herramientas y la metodología a utilizar para el desarrollo del sistema.

El *Capítulo 2. Análisis y diseño de la herramienta informática para la gestión centralizada de dispositivos USB.* En este capítulo se presentan los requisitos del sistema a ser desarrollados, su descripción y todo el proceso de análisis y diseño de manera detallada de acuerdo a lo establecido en la metodología utilizada, además de cómo quedará desplegado el sistema.

El *Capítulo 3. Implementación, pruebas y evaluación de la herramienta informática para la gestión centralizada de dispositivos USB.* Se muestra una descripción de la validación de la propuesta, quedará plasmado cómo será implementado el sistema, su estructuración en clases y componentes que garantizan la capacidad operacional del producto, con la explicación detallada

de cada componente y su funcionalidad. Se describe la estrategia de pruebas utilizada y los resultados obtenidos de las mismas.

CAPÍTULO 1: Fundamentación teórica del proceso de gestión centralizada de los dispositivos USB

Introducción

En el presente capítulo se exponen los temas teóricos y conceptuales de la investigación. Para ello se ofrecen fundamentos teóricos y conceptos relacionados con el tema, analizando resultados obtenidos por otros investigadores, así como publicaciones de diversas fuentes actualizadas que permitan crear el marco teórico que sustenta la investigación. Se define la metodología de desarrollo de software utilizada para la elaboración de la propuesta de solución, así como los lenguajes y herramientas empleados en el modelado y desarrollo de la herramienta de gestión centralizada de dispositivos USB en GNU/Linux Nova.

1.1 Definición de conceptos

Se denomina gestión al correcto manejo de los recursos de los que dispone una determinada organización como por ejemplo empresas, organismos públicos, organismos no gubernamentales, entre otros. La gestión puede abarcar una larga lista de actividades, pero siempre se enfoca en la utilización eficiente de estos recursos, en la medida en que debe maximizarse sus rendimientos (10). Según Guillermo Westreicher gestión es un conjunto de procedimientos y acciones que se llevan a cabo para lograr un determinado objetivo (11), Julián Pérez Porto expresa como gestión la acción y consecuencia de administrar o gestionar algo (12).

La centralización indica la acción de concentrar la autoridad sobre una persona o grupo en específico de individuos con el fin de optimizar el sistema de trabajo de una organización o administración de una empresa (13). Según el diccionario de la Real Academia Española de la Lengua (RAE), centralizar es la acción de reunir varias cosas en un centro común (14) Julián Pérez Porto y Ana Gardey dicen que centralizar es reunir varias cosas en un centro común o hacer que distintas cosas dependan de un poder central (15).

La gestión centralizada se refiere a la estructura organizativa de mantener la autoridad de toma de decisiones en la cima. Se puede expresar como gestión centralizada cuando un responsable o un ejecutivo de una organización dirige y gestiona un proceso con el objetivo de mejorar la calidad del trabajo de manera rápida y ágil.

Un lugar de trabajo o área de trabajo son las zonas en la que los trabajadores de una empresa u organización permanecen en razón de su trabajo. Se consideran incluidos en esta definición

los servicios higiénicos, locales de descanso, locales de primeros auxilios, comedores, entre otros, Myriam Quiroa define como área de trabajo todos los departamentos que conforman una empresa, para que esta pueda trabajar de forma eficiente (16).

1.2 Proceso de gestión y control centralizado de dispositivos USB

La gestión y control de dispositivos USB es el proceso de monitorear y restringir el uso de dispositivos USB dentro de una red para protegerla frente a las amenazas internas. La función de protección de los dispositivos USB permite a los administradores limitar el alcance del uso del dispositivo USB de forma selectiva en función de los diversos roles y departamentos. Esta función permite a los administradores controlar centralmente el uso de varios dispositivos USB en la red al bloquearlos o deshabilitarlos, también permite tener una lista blanca de dispositivos que independientemente de dónde se conecten estarán siempre disponibles en nuestra red de computadoras (17).

1.3 Aplicaciones informáticas para la gestión y control centralizado de dispositivos USB

En la actualidad existen diversas aplicaciones dedicadas a la gestión y control de los dispositivos USB en Linux, entre ellas están, Protector de punto final de CoSoSys, USBGuard y PPUSB. A continuación, se presenta una breve caracterización de cada una de ellas.

1.3.1 Protector de punto final de CoSoSys

Protector de punto final de CoSoSys es una aplicación que monitorea y controla las transferencias de datos desde los puntos finales a los dispositivos de almacenamiento extraíbles y protege contra la pérdida de datos, puede protegerlo de las amenazas internas y la fuga accidental de datos que se producen debido a los dispositivos extraíbles. Esta herramienta proporciona la posibilidad de abrir o bloquear el acceso a varios dispositivos como USB, teléfonos inteligentes, tarjetas de red Wifi, tabletas, impresoras, etc (18).

Cuenta con un módulo de control de dispositivos y un módulo de protección con reconocimiento de contenido de Endpoint Protector se utilizan para proteger los datos en movimiento. El módulo de control de dispositivos se encarga de todos los dispositivos que están conectados a su sistema y el módulo de protección se encarga del flujo de datos a través de todas las aplicaciones web.

El control de dispositivos es una forma eficaz de controlar las conexiones de los dispositivos físicos que se conectan a una máquina. Este módulo permite administrar los derechos de los dispositivos, edita la configuración de los dispositivos, desinstala o elimina los dispositivos, controla las transferencias de archivos entre otras funciones (19).

El módulo de protección consciente de contenido realiza una inspección del contenido real de los archivos, desde documentos Word hasta hojas de cálculo y correos electrónicos, a través de este módulo el administrador puede hacer cumplir políticas de contenido sólidas para evitar transferencias de archivos no intencionales o intencionales de datos confidenciales de la empresa. El módulo de protección basada en contenido permite crear, revisar y editar políticas para controlar los archivos transferidos a través de la red (20).

1.3.2 USBGuard

USBGuard es un software que ofrece un mecanismo de lista blancas y listas negras basadas en atributos de los dispositivos, hace uso de una infraestructura de bloqueo de dispositivos incluida en el kernel de Linux. Cuando se conecte un dispositivo USB USBGuard explora el dispositivo y a continuación busca en su archivo de configuración para comprobar si ese dispositivo está permitido o no. Esta herramienta no viene instalada de forma predeterminada en ninguna de las distribuciones de Linux, pero se puede instalar en ellas usando el código fuente (21).

Para comenzar la protección, se hace mediante el comando shell **USBGuard** y su subcomando **generate-policy** y así generar una política inicial. La herramienta genera una política de permisos para todos los dispositivos conectados actualmente en el sistema.

Cualquier dispositivo que se conecte a la máquina no funcionará a menos que aparezca como (**allow**) en la lista de dispositivos la cual es visible a través del comando **lsusb** (22).

1.3.3 PPUSB

PPUSB está enfocado a la protección de los puertos USB de forma local, al conectar un dispositivo USB al ordenador se muestra aviso de la entrada del dispositivo con las opciones de Autorizar y Desautorizar este dispositivo, solo un administrador podrá acceder a estas opciones. En el momento en que este dispositivo sea Autorizado estará disponible en el ordenador y se podrá trabajar con él, si es Desautorizado este dispositivo no estará disponible y no se podrá realizar ninguna acción con él.

1.4 Análisis comparativo de herramientas para la gestión centralizada de dispositivos USB

En el siguiente epígrafe se realizará un análisis comparativo de las herramientas informáticas para la gestión y control de los dispositivos USB.

Tabla 1 Comparación de herramientas de gestión y control de dispositivos USB (Elaboración propia)

Herramientas para la gestión y control de dispositivos USB / Criterios de comparación	Protector de punto final de CoSoSys	USBGuard	PPUSB
¿Compatible con Nova?	No	Si	Si
Tipo de Control	Remoto	Local	Local
¿Interfaz visual?	Si	No	Si
¿De pago?	Si	No	No

Actualmente para la distribución cubana GNU/Linux Nova se necesita una herramienta compatible con dicha distribución, que tenga interfaz visual, debe ser totalmente gratis y además debe poder controlar remotamente todos los dispositivos USB que se conectan en toda la red de computadoras. La comparación anterior evidencia que las herramientas estudiadas son muy buenas para la protección y control de los dispositivos USB, pero no cubren en su totalidad las necesidades existentes para la distribución cubana GNU/Linux Nova. Como se puede observar la herramienta más completa es PPUSB, pero tiene como desventaja su tipo de control que es local; lo que demuestra la necesidad de desarrollar una herramienta capaz de gestionar centralmente los dispositivos USB en la distribución cubana GNU/Linux Nova.

Aun así, el estudio de estas herramientas brindó conocimientos sobre la manera en que se gestionan y controlan los dispositivos USB. Para el desarrollo de la propuesta de solución se tuvo en cuenta algunas de las funcionalidades que estas aplicaciones ofrecen.

1.5 Definición del proceso de gestión centralizada de dispositivos USB

La propuesta de solución se basa en el desarrollo de un Sistema de Gestión Centralizada de dispositivos USB dicho sistema estará encargado de gestionar todos los locales de un área de trabajo, así como los dispositivos USB que se conectan en las computadoras de cada uno de esos locales.

El sistema será controlado por un administrador, este podrá añadir un local, modificar los datos de uno existente y eliminar el que el considere conveniente. Este sistema debe permitir recibir datos de la aplicación de escritorio PPUSB, así como el estado (autorizado o denegado) de los de todos los dispositivos USB que se conectan en las computadoras, además de poder listar todas las computadoras existentes por cada local.

El sistema a desarrollar, facilitará en gran medida el control de los dispositivos USB ya que contará con funcionalidades que posibilitan el aumento de la agilidad y organización de este proceso.

1.6 Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de técnicas, procedimientos y herramientas que ayudan a los desarrolladores de software a implementar sistemas informáticos, tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo (23). En el epígrafe 1.5.1 se define la metodología de desarrollo de software empleada en la elaboración de la propuesta de solución.

1.6.1 Variación de AUP para la UCI

La metodología de desarrollo de software Variación AUP para la UCI es una variación de AUP (Agile Unified Process, Proceso Unificado Ágil) que logra adaptarse al ciclo de vida definido en los proyectos productivos de la universidad (24).

Esta metodología cuenta con 3 fases:

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.

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, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Cierre: en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Existen tres formas de encapsular los requerimientos en la variación AUP-UCI: Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requerimientos por proceso (DRP), agrupados en cuatro escenarios como se muestra a continuación:

Escenario No 1:

Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No 2:

Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No 3:

Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4:

Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Figura 1 Representación gráfica de los escenarios de la metodología AUP-UCI (Tamara Rodríguez Sánchez, 2015)

Características por escenarios.

Escenario No 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización (24).

Escenario No 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que

ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (24).

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados (24).

Escenario No 4: 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. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información (24).

Basado en lo antes expuesto, se decide emplear el escenario cuatro ya que es factible aplicarlo a proyectos bien definidos, además se tuvo en cuenta que el cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requerimientos y así poder implementarlos, probarlos y validarlos.

1.7 Tecnologías de Desarrollo

1.7.1 Frontend

Frontend es la parte de un sitio web o aplicación web que se ve desde un navegador. Es decir, la interfaz gráfica que gestiona toda la interacción con el usuario.

El desarrollo frontend está basado en **HTML** y **CSS**, los lenguajes de maquetación que permiten definir la estructura y estilos de una página web, las funcionalidades, animaciones y otros elementos del frontend son programados con **JavaScript**, un lenguaje de programación para definir la lógica de nuestra aplicación, recibir las solicitudes de los usuarios y enviárselos al backend (25).

El frontend sirve para realizar la interfaz de un sitio web, desde su estructura hasta los estilos, tales como la definición de los colores, texturas, tipografías, secciones, entre otros. Su uso es determinante para que el usuario tenga una buena experiencia dentro del sitio o aplicación (26).

Dentro de los elementos del frontend están:

- Estructuras de navegación: este elemento se refiere al orden en que se organizan las diferentes páginas de un sitio web y a los componentes que se vinculan entre sí para realizar diferentes funciones dentro del sitio.
- Layout: también nombrado como diseño de página, se refiere a todos los componentes de la página web, por ejemplo: ubicación del menú, botones, footer (parte inferior de un sitio web) todo lo necesario para que un sitio sea útil y fácil de navegar.
- Contenido web: es todo aquello que brinde información relevante o interesante para los usuarios. Es importante destacar que el contenido no tiene que ser necesariamente texto, puede incluir sonido o materiales interactivos.
- Imágenes: son los recursos visuales que ayudan a aumentar el interés de los usuarios. Esto también puede incluir videos, animaciones, mapas, gráficas, infografías, ilustraciones, diagramas, etc.
- Logotipo: para que un sitio web tenga mayor identidad es vital que contenga el logotipo que represente a la marca o empresa.
- Diseño gráfico: este elemento engloba todo lo relacionado con cómo se ve el sitio web y su apariencia: colores, formas, tipografías, tamaños, etc.

HTML5: Lenguaje de etiquetas de hipertexto

HTML (Lenguaje de Marcas de Hipertexto, del inglés *HyperText Markup Language*) es el componente más básico de la web, define el significado y la estructura del contenido web. Además de HTML, se utilizan otras tecnologías para describir la apariencia de una página web (CSS) y la funcionalidad (JavaScript) (27).

HTML utiliza "marcas" para etiquetar texto, imágenes y otro contenido para mostrarlo en un navegador Web. Un elemento HTML se distingue de otro texto en un documento mediante "etiquetas", que consisten en el nombre del elemento rodeado por "<" y ">". El nombre de un elemento dentro de una etiqueta no distingue entre mayúsculas y minúsculas. Es decir, se puede escribir en mayúsculas, minúsculas o una mezcla (27).

Un documento HTML contiene un árbol de etiquetas, que tiene una organización básica. La etiqueta raíz de un documento HTML es <html> y dentro tiene dos etiquetas principales: <head> (es la información de cabecera de la página) y la etiqueta <body> (es la información del cuerpo, es decir, lo que se verá en el navegador cuando el usuario acceda al sitio web) (28).

CSS3: Hojas de Estilo en Cascada

Hojas de Estilo en Cascada (del inglés *Cascading Style Sheets*) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML (en-US) (incluyendo varios lenguajes basados en XML como SVG, MathML o XHTML). CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios (29).

CSS es utilizado para diseñar y dar estilo a las páginas web, por ejemplo, alterando la fuente, color, tamaño y espaciado del contenido, dividirlo en múltiples columnas o agregar animaciones y otras características decorativas. Este módulo proporciona un suave comienzo hacia el dominio de CSS con los conceptos básicos acerca de su funcionamiento, la sintaxis y la manera en que puedes comenzar a utilizarlo para agregar estilos al HTML (29).

JavaScript 1.7

JavaScript es un lenguaje de programación que te permite implementar funciones complejas en páginas web, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo entre otros. JavaScript abarca diferentes tipos de software, como juegos, programas de computadora, aplicaciones web entre otros (30).

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). JavaScript es el único lenguaje de programación que entienden de forma nativa los navegadores.

JavaScript depende del entorno en que se ejecute para ofrecer objetos y métodos por los que los scripts pueden interactuar con el mundo exterior, por ejemplo, en HTML para importar scripts se hace por medio de la etiqueta <scripts> (31).

1.7.2 Backend

Backend es un término de desarrollo web que hace referencia a un tipo de programación en el que se configuran todos los aspectos lógicos de una página web o aplicación, el backend es la programación de todo lo que el usuario final no ve, es decir, el acceso a las bases de datos, el procesamiento de los datos ingresados por los usuarios, y la ejecución de un script (32).

El backend son todos los códigos ocultos que sirven para que una página web o aplicación funcione correctamente. Además, de su estructura y organización depende la experiencia de usuario. De igual forma, el backend se encarga de optimizar otros elementos y recursos como la seguridad y privacidad en un sitio web o aplicación (26).

Dentro de las aplicaciones backend están:

- Inicios de sesión: cuando una persona accede a un sitio web o aplicación utiliza un correo electrónico y contraseña, esta información es validada y resguardada por el servidor, que consulta su base de datos y así identifica y permite el acceso al usuario.
- Carrito de compras: este elemento permite la compra de productos en línea y sirve para facilitar la selección de diferentes productos o servicios que algún usuario desee comprar.
- Cookies: muchos sitios utilizan cookies para realizar un seguimiento de aquello que los usuarios vieron anteriormente, lo que les permite sugerir otros contenidos (o productos) de interés.
- Formularios de contacto: si un visitante del sitio web se interesa por recibir más información o ponerse en contacto, se debe contar con un elemento que sea capaz de vincular al usuario con la empresa.

Python 3.10.0

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, capaz de realizar cualquier programa, desde aplicaciones Windows, servidores de red y páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad (33).

1.7.3 Frameworks

Además de los lenguajes de programación que definen cómo estará escrito el código, es importante destacar la importancia de los *frameworks* de desarrollo estos son las herramientas y librerías de código pre-escrito que facilitan a los desarrolladores las tareas cotidianas o de uso común para evitar ejecutarlas desde cero.

Django 4.1

Es un framework web de alto nivel escrito en Python que permite el desarrollo rápido de sitios web seguros y mantenibles. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago. Provee casi

todo lo que los desarrolladores necesitan, sigue principios de diseño consistentes y tiene una amplia y actualizada documentación (34).

Usa un componente basado en la arquitectura *"shared-nothing"* donde cada parte de la arquitectura es independiente de las otras por lo tanto puede ser reemplazado o cambiado si es necesario.

Bootstrap 5

Bootstrap es una biblioteca multiplataforma de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales.

Además de todas las características que ofrece el framework, su principal objetivo es permitir la construcción de sitios web responsive para dispositivos móviles, esto significa que las páginas están diseñadas para funcionar en desktop, tablets y smartphones, de una manera muy simple y organizada (35).

1.8 Herramientas para el modelado de la solución

La modelación de software es una técnica con la complejidad inherente a los sistemas. El uso de modelos ayuda al equipo de trabajo de desarrollo de software a visualizar el sistema de información a construir. Un modelo permite comprender mejor el sistema que estamos desarrollando: sus elementos y sus relaciones (36).

Lenguaje Unificado de Modelado (UML 2.5)

El lenguaje de modelado unificado (UML por sus siglas en inglés) es un estándar para la representación visual de objetos, estados y procesos dentro de un sistema, este puede servir de modelo para un proyecto y garantizar así una arquitectura de información estructurada; por otra parte, ayuda a los desarrolladores a presentar la descripción del sistema de una manera que sea comprensible para quienes están fuera del campo (37).

Visual Paradigm 15.2

Visual Paradigm es una herramienta de software diseñada para que los equipos de desarrollo de software modelen el sistema de información empresarial y gestionen los procesos de desarrollo. Admite lenguajes y estándares de modelado clave de la industria, como Lenguaje de modelado unificado (UML). Ofrece un conjunto completo de herramientas de software que las empresas necesitan para la captura de requisitos, análisis de procesos, diseño de sistemas, diseño de bases de datos, etc (38).

1.9 Herramientas de Desarrollo

PyCharm 2021.1.1

PyCharm es un IDE (Entorno de Desarrollo Integrado) multiplataforma dirigido principalmente a Python y al desarrollo web. Proporciona una finalización del código inteligente, inspecciones del código, indicación de errores sobre la marcha y arreglos rápidos, así como refactorización de código automática y completas funcionalidades de navegación.

El editor de código inteligente de PyCharm ofrece compatibilidad de primer nivel con Python, JavaScript, CoffeeScript, TypeScript, CSS, entre otros lenguajes de plantilla (39).

MySQL 8.0

MySQL es un sistema open source de administración de base de datos desarrollado y soportado por Oracle. MySQL permite almacenar y acceder a los datos a través de múltiples motores de almacenamiento, incluyendo InnoDB, CSV y NDB, también es capaz de replicar datos y particionar tablas para mejorar el rendimiento y la durabilidad. Los usuarios de MySQL no tienen que aprender nuevos comandos, pueden acceder a sus datos utilizando comandos SQL estándar (40).

MySQL está escrito en C y C++ y está disponible en más de 20 plataformas tales como Mac, Windows, Linux y Unix, soporta grandes bases de datos con millones de registros y admite muchos tipos de datos como float, double, char, varchar, binary, time, date, datetime entre otros. Por parte de la seguridad MySQL utiliza un sistema de privilegios de acceso y contraseñas encriptadas que permite la verificación basada en el host. Los clientes de MySQL pueden conectarse a MySQL Server utilizando varios protocolos, incluyendo sockets TCP/IP en cualquier plataforma. MySQL también admite una serie de programas cliente y de utilidad, programas de línea de comandos y herramientas de administración como MySQL Workbench (40).

1.10 Servidor de aplicaciones

Un servidor de aplicaciones es un programa de servidor en un equipo en una red distribuida que proporciona la lógica de negocio para un programa de aplicación. El servidor de aplicaciones se ve frecuentemente como parte de una aplicación de tres niveles, que consta de un servidor gráfico de interfaz de usuario (GUI), un servidor de aplicaciones (lógica empresarial) y un servidor de bases de datos y transacciones (41).

Apache 2.4.41

Apache es un servidor web HTTP (Hipertext Transfer Protocol) gratuito, de código abierto, flexible, fácil de mantener y altamente configurable debido a su diseño modular. Es

multiplataforma y presenta abundante documentación, adicionalmente tiene dentro de sus principales ventajas el funcionamiento con varios lenguajes (42).

Conclusiones del capítulo

Al finalizar el desarrollo de este capítulo se han esclarecido los conceptos más importantes sobre la temática y se ha alcanzado un nivel más alto de comprensión sobre la gestión y control de los dispositivos USB. El estudio de las herramientas homólogas permitió identificar funcionalidades que garantizan el control centralizado de los dispositivos USB. El análisis de las distintas herramientas y tecnologías de desarrollo permitió seleccionar las adecuadas para el desarrollo del sistema propuesto. Se decidió usar un *framework* para el desarrollo de la solución, en ambas partes del sistema, tanto *frontend* como *backend*.

CAPÍTULO 2: Análisis y diseño de la herramienta informática para la gestión centralizada de dispositivos USB

Introducción

En el presente capítulo se describe la herramienta que se desea implementar. Se definen los requisitos a través de la especificación de requisitos de software, su descripción mediante historias de usuario, además de realizar la validación de los mismos. Se realiza el diagrama de clases del diseño, en el que se representan las restricciones de implementación de las funcionalidades de la herramienta a partir de los patrones del diseño. Se presenta el modelo de datos que describe la estructura de persistencia de la información y el diseño arquitectónico definido para la propuesta de solución.

2.1 Descripción del contexto de la propuesta de solución desarrollada

La descripción del contexto del negocio presentada en el epígrafe se realizó utilizando un mapa conceptual. El mismo permite comprender el funcionamiento del negocio informatizado.

2.1.1 Mapa conceptual

El mapa conceptual es una representación que ayuda a entender un tema en específico al visualizar las relaciones entre las ideas y conceptos. Por lo general, las ideas son representadas en nodos estructurados jerárquicamente y se conectan con palabras de enlace sobre las líneas para explicar las relaciones (43). A continuación, se presenta el mapa conceptual que corresponde a la descripción del contexto del negocio de la propuesta de solución:

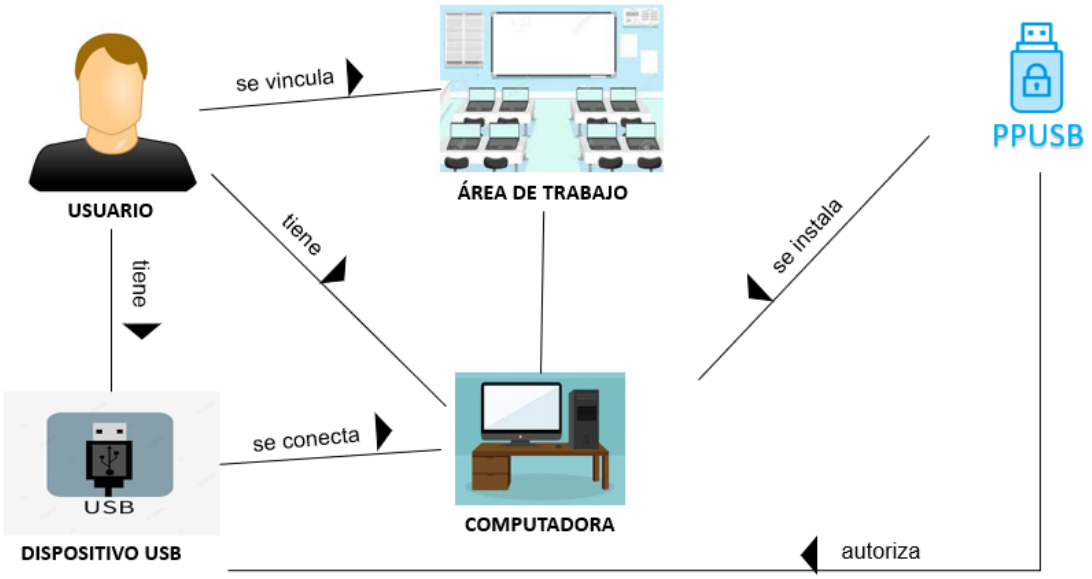

Figura 2 Mapa conceptual (Elaboración propia)

Las definiciones que se muestran a continuación permiten comprender el significado que tienen los conceptos representados en el mapa conceptual del contexto del negocio informatizado.

Conceptos del contexto del negocio:

Local: es el área de trabajo de una organización, el cual tiene un nombre además de tener asignado personas y computadoras.

Computadora: es un ordenador que tiene como sistema operativo Nova 8.

Dispositivo USB: es un dispositivo de almacenamiento conectado en las computadoras a través de un puerto.

PPUSB: es una aplicación instalada en las computadoras, tiene como objetivo controlar y dar acceso a los dispositivos USB conectados en ella.

Usuario: es un trabajador el cual está vinculado a un local y a una computadora.

2.2 Requisitos

La ingeniería de requisitos es la fase de un proyecto de software donde se definen las propiedades y la estructura del mismo; y que a la vez comprende el desarrollo y gestión de requisitos (44). A continuación, se presenta las actividades desarrolladas en esta disciplina, así como los productos de trabajo elaborados en el desarrollo de la investigación.

2.2.1 Fuentes para la obtención de requisitos

Una de las etapas más importantes dentro del desarrollo de software consiste en la definición de requerimientos, ya que es en esta etapa donde se plasman de forma clara y concisa todos los requisitos del usuario (45), existe un gran número de técnicas para obtener requerimientos. Durante esta etapa de la investigación se tuvieron en cuenta como fuentes de obtención de requisitos:

• Especialistas del centro CESOL.

• Aplicaciones informáticas para la gestión y control de dispositivos USB en la distribución GNU/Linux.

2.2.2 Técnicas de identificación de requisitos

Para el desarrollo de la propuesta de solución se emplearon como técnicas de identificación de requisitos: la entrevista, la tormenta de ideas y el desarrollo de prototipos. A continuación, se muestra su definición y su forma de aplicación.

Entrevista

Una entrevista es un intercambio de ideas u opiniones mediante una conversación con el fin de obtener información sobre un tema específico, participan dos roles: el entrevistador y el entrevistado (46). Esta técnica se aplicó a través de una entrevista realizada a los miembros del proyecto Nova 8 (Anexo 1), para conocer cuáles son las tecnologías compatibles con el entorno de desarrollo, así como los requerimientos y las características que debe tener una aplicación para ser instalada en dicho software.

Tormenta de ideas

La tormenta de ideas es una técnica cuyo objetivo principal es encontrar soluciones creativas a algún problema (47). Se realizó una tormenta de ideas con miembros del equipo de desarrollo de Nova 8 para concretar cuales son las funcionalidades que no deben faltar en la propuesta de solución, en esta se tuvieron en cuenta todos los planteamientos realizados.

Desarrollo de prototipos

Un prototipo es una versión preliminar del software para ofrecer al usuario una visión previa de cómo será el sistema, es considerado un desarrollo evolutivo porque va evolucionando hasta convertirse en el producto final. (48) Se aplicó esta técnica para conocer cuáles son las características que debían tener las interfaces de la herramienta desarrollada. Los prototipos elaborados fueron revisados por miembros del equipo de desarrollo, quienes consideraron que se debía contar con estos prototipos para la representación visual de la propuesta de solución.

La aplicación de las técnicas descritas permitió comprender con profundidad el problema de la investigación y facilitando la identificación de las funcionalidades y características de la propuesta de solución, definidas en el siguiente epígrafe.

2.2.3 Especificación de requisitos de software

La Especificación de Requisitos de Software es una actividad primordial en el desarrollo de software, orienta el comportamiento del sistema que se va a desarrollar, tiene como objetivo ayudar a los clientes a describir claramente que quieren lo que facilitará el trabajo del equipo de desarrollo, contiene un conjunto de requisitos funcionales y no funcionales, el lenguaje utilizado para su redacción debe ser informal, de forma que sea fácil de comprender por todas las partes involucradas en el desarrollo (49).

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que prestará el sistema, en otras palabras, es el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones (50). A continuación, se presenta el listado de los 8 requisitos funcionales de la propuesta de solución:

Tabla 2 Listado de requisitos funcionales (Elaboración propia)

Número	Requisitos	Descripción	Complejidad
RF 1	Autenticar usuario	Permite que un usuario se autentique en el sistema.	Media
RF 2	Adicionar local	Permite insertar un local a la base de datos.	Alta
RF 3	Eliminar local	Permite eliminar un local de la base de datos.	Media
RF 4	Modificar local	Permite modificar los datos de un local que se encuentre en la base de datos.	Alta
RF 5	Listar local	Permite mostrar los datos de un local.	Baja
RF 6	Listar computadoras	Permite mostrar una computadora.	Baja
RF 7	Mostrar listado de computadoras por local	El sistema debe mostrar un listado de todas las computadoras existentes en un local.	Media
RF 8	Mostrar listado de dispositivos USB por computadora	El sistema debe mostrar un listado de todos los dispositivos USB dada una computadora.	Alta

Requisitos no funcionales

Los requisitos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando, entre ellos están los referidos a atributos como la eficiencia, seguridad, usabilidad del sistema entre otros (50).

La definición de los requisitos no funcionales de la propuesta de solución se definió mediante el estándar de calidad ISO-25010 (International Organization for Standarization, Organización Internacional de Normalización), propuesto en el producto de trabajo "Especificación de requisitos de software" del expediente de proyecto utilizado en la actividad productiva de la universidad. A continuación, se presenta el listado de los 7 requisitos no funcionales de esta propuesta:

Tabla 3 Listado de requisitos no funcionales (Elaboración propia)

Requisitos	Número	Descripción
Requisito de funcionalidad	RNF 1	La herramienta para la gestión centralizada de dispositivos USB debe realizar todas las operaciones indicadas por el usuario tales como: insertar, mostrar, modificar y eliminar usuarios, computadoras y locales, además de mostrar listados de computadoras y dispositivos USB.
	RNF 2	En las computadoras del área de trabajo debe estar instalada la herramienta PPUSB.
	RNF 3	La herramienta debe permitir recibir datos de PPUSB.
Requisito de seguridad	RNF 4	A la herramienta solo podrán acceder los usuarios con usuario y contraseña válida.
Requisito de interoperabilidad	RNF 5	La herramienta para la gestión centralizada de dispositivos USB debe ser compatible con Nova 8.
Requisito de usabilidad	RNF 6	La herramienta para la gestión centralizada de dispositivos USB debe mostrar sus interfaces en idioma español; además de mantener una secuencia lógica de pasos a seguir para el usuario.

Requisito de mantenibilidad	RNF 7	La herramienta para la gestión centralizada de dispositivos USB debe estar codificado en el estándar definido por el proyecto de desarrollo de Nova 8, con el objetivo de facilitar el análisis para futuras modificaciones o para dar mantenimiento al sistema.

2.3 Descripción de requisitos de software mediante Historias de Usuario

Una historia de usuario es una explicación informal de una función de software, escrita desde la perspectiva del usuario final. Estas historias deben escribirse utilizando un lenguaje no técnico para brindar contexto al equipo de desarrollo (51).

A continuación, se muestran 8 historias de usuario generadas a partir de los 8 requisitos funcionales, las cuales se estructuran de la siguiente manera:

1. Número: a cada historia de usuario se le asigna un número para facilitar la identificación por parte del equipo de desarrollo.
2. Usuario: es la persona que accede al sistema.
3. Nombre: nombre descriptivo de cada historia de usuario.
4. Prioridad en el negocio: es el grado de prioridad que le asigna el cliente a la historia de usuario en dependencia del valor en el negocio. Los valores que puede tomar son (Alta, Media o Baja).
5. Riesgo en desarrollo: es el grado de complejidad que le asigna el equipo de desarrollo a la historia de usuario luego de analizarla. Los valores que toma pueden ser (Alto, Medio o Bajo).
6. Iteración asignada: número de la iteración en la cual será implementada la historia de usuario.
7. Tiempo estimado: estimación por el equipo de desarrollo para darle cumplimiento a la historia de usuario.
8. Tiempo real: es el plazo que el equipo de desarrollo tiene para cumplir la historia de usuario.
9. Programador responsable: es el programador encargado de desarrollar la historia de usuario.
10. Descripción: es la descripción sobre lo que debe hacer la historia de usuario.
11. Validación: requisito que debe cumplir para interactuar con la historia de usuario.

12. Prototipo elemental de interfaz gráfica: es el primer modelo que sirve como representación del producto final.

Tabla 4 Descripción de la historia de usuario del RF.1 Autenticar usuario (Elaboración propia)

Historia de Usuario	
Número: RF 1	**Usuario:** Administrador
Nombre: Autenticar usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Iteración asignada: 1	
Tiempo estimado: 10 horas	**Tiempo real: 10 horas**
Programador responsable: Roxatne M. Moreno Ramírez	
Descripción: El usuario introduce un usuario y una contraseña en un campo de texto, y luego presiona un botón (Aceptar) para enviar los datos.	
Validación: Los campos de usuario y contraseña deben estar correctos para poder acceder al sistema.	
Prototipo elemental de interfaz gráfica de usuario	

Figura 3 Prototipo de interfaz de usuario correspondiente al RF1 Autenticar usuario (Elaboración propia)

Tabla 5 Descripción de la historia de usuario del RF.2 Adicionar local (Elaboración propia)

Historia de Usuario	
Número: RF 2	**Usuario:** Administrador
Nombre: Adicionar Local	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 1	
Tiempo estimado: 14 horas	**Tiempo real: 10 horas**
Programador responsable: Roxatne M. Moreno Ramírez	
Descripción: El administrador presiona un botón (Añadir local), introduce los datos en un campo de texto y luego presiona un botón (Aceptar) para enviar los datos.	
Validación El usuario debe estar autenticado en el sistema.	
Prototipo elemental de interfaz gráfica de usuario	

Figura 4 Prototipo de interfaz de usuario correspondiente al RF2 Añadir local (Elaboración propia)

Tabla 6 Descripción de la historia de usuario del RF.3 Eliminar local (Elaboración propia)

Historia de Usuario	
Número: RF 3	**Usuario:** Administrador
Nombre: Eliminar Local	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Iteración asignada: 1	
Tiempo estimado: 6 horas	**Tiempo real: 6 horas**
Programador responsable: Roxatne M. Moreno Ramírez	
Descripción: El administrador podrá eliminar cualquier local de la base de datos al presionar un botón rojo (Eliminar), se visualizará en pantalla una alerta de seguridad, el administrador podrá presionar el botón (Aceptar) para eliminar definitivamente el local o podrá presionar el botón (Cancelar) para cancelar la operación.	
Validación: El usuario debe estar autenticado en el sistema.	
Prototipo elemental de interfaz gráfica de usuario	

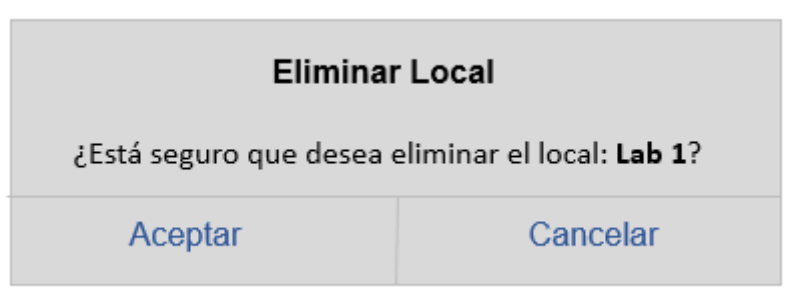

Figura 5 Prototipo de interfaz de usuario correspondiente al RF3 Eliminar local (Elaboración propia)

Tabla 7 Descripción de la historia de usuario del RF.4 Modificar local (Elaboración propia)

Historia de Usuario	
Número: RF 4	**Usuario:** Administrador
Nombre: Modificar Local	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta

Iteración asignada: 1	
Tiempo estimado: 18 horas	**Tiempo real: 15 horas**
Programador responsable: Roxatne M. Moreno Ramírez	
Descripción: El administrador podrá modificar cualquier dato de los locales existentes en la base de datos al presionar un botón verde (Modificar), se visualizará en pantalla todos los datos del local seleccionado, los cuales pueden ser modificados, el administrador presionará un botón (Aceptar) para enviar los datos y un botón (Cancelar) para cancelar la operación.	
Validación: El usuario debe estar autenticado en el sistema.	
Prototipo elemental de interfaz gráfica de usuario	

Figura 6 Prototipo de interfaz de usuario correspondiente al RF4 Modificar local (Elaboración propia)

Tabla 8 Descripción de la historia de usuario del RF.5 Listar local (Elaboración propia)

Historia de Usuario	
Número: RF 5	**Usuario:** Administrador
Nombre: Listar Local	

Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Iteración asignada: 1	
Tiempo estimado: 6 horas	**Tiempo real: 5 horas**
Programador responsable: Roxatne M. Moreno Ramírez	
Descripción: Cuando el administrador acceda al sistema, se visualizará en pantalla un listado con todos los locales de la base de datos.	
Validación: El usuario debe estar autenticado en el sistema.	
Prototipo elemental de interfaz gráfica de usuario	

Figura 7 Prototipo de interfaz de usuario correspondiente al RF5 Listar local (Elaboración propia)

Tabla 9 Descripción de la historia de usuario del RF.6 Mostrar listado de computadoras (Elaboración propia)

Historia de Usuario	
Número: RF 6	**Usuario:** Administrador
Nombre: Mostrar listado de computadoras.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Iteración asignada: 1	
Tiempo estimado: 8 horas	**Tiempo real: 8 horas**
Programador responsable: Roxatne M. Moreno Ramírez	

Descripción: El administrador podrá visualizar todas las computadoras conectadas en red de todos los locales al presionar un botón (Computadoras).
Validación: El usuario debe estar logueado en el sistema.
Prototipo elemental de interfaz gráfica de usuario

Figura 8 Prototipo de interfaz de usuario correspondiente al RF6 Mostrar listado de computadoras (Elaboración propia)

Tabla 10 Descripción de la historia de usuario del RF.7 Mostrar listado de computadoras por local (Elaboración propia)

Historia de Usuario	
Número: RF 7	**Usuario:** Administrador
Nombre: Mostrar listado de computadoras por local	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Iteración asignada: 1	
Tiempo estimado: 15 horas	**Tiempo real: 12 horas**
Programador responsable: Roxatne M. Moreno Ramírez	

Descripción: El administrador podrá visualizar todas las computadoras conectadas en red de un local al presionar un local del listado de locales.
Validación: El usuario debe estar logueado en el sistema.
Prototipo elemental de interfaz gráfica de usuario

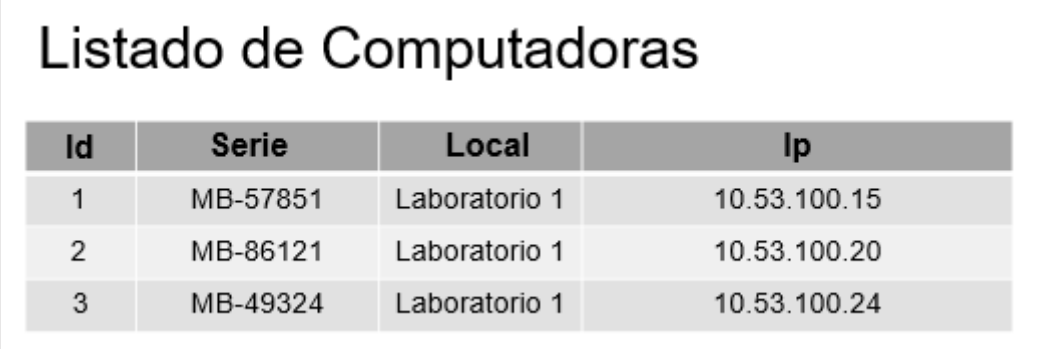

Figura 9 Prototipo de interfaz de usuario correspondiente al RF7 Mostrar listado de computadoras por local (Elaboración propia)

Tabla 11 Descripción de la historia de usuario del RF.8 Mostrar listado de dispositivo USB por computadora (Elaboración propia)

Historia de Usuario	
Número: RF 8	**Usuario:** Administrador
Nombre: Mostrar listado de dispositivos USB por computadora	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Iteración asignada: 1	
Tiempo estimado: 18 horas	**Tiempo real: 15 horas**
Programador responsable: Roxatne M. Moreno Ramírez	
Descripción: El administrador podrá ver un listado de todos los dispositivos USB con su estado (autorizado o denegado) al presionar un botón (Dispositivos).	
Validación: El usuario debe estar logueado en el sistema.	
Prototipo elemental de interfaz gráfica de usuario	

Figura 10 Prototipo de interfaz de usuario correspondiente al RF8 Mostrar listado de dispositivos por computadora (Elaboración propia)

2.4 Análisis y diseño

Esta disciplina es muy importante en el desarrollo de un software, si se considera necesario los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos. Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales (24).

2.4.1 Diseño de clases

El diseño de clases de la herramienta para la gestión centralizada de dispositivos USB, se realizó mediante estereotipos web. En su desarrollo se tuvo en cuenta los patrones de diseño GRASP (General Responsibility Assignment Software Patterns, Patrones de Asignación de Responsabilidad). A continuación, se detallan los resultados obtenidos.

Diagrama de clases del diseño

Un diagrama de clase es un tipo de diagrama UML que describe un sistema visualizando los diferentes tipos de objetos dentro de un sistema y los tipos de relaciones estáticas que existen entre ellos. También ilustra las operaciones y atributos de las clases (52).

A continuación, se presenta el diagrama de clases del diseño correspondiente al Requisito Funcional 1 Autenticar Usuario.

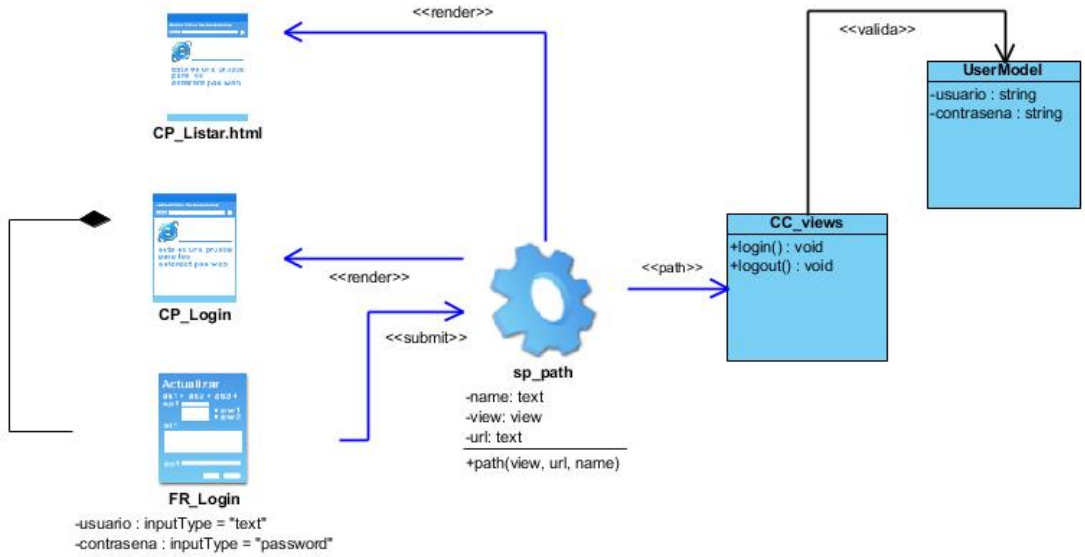

Figura 11 Diagrama de clase del RF 1 Autenticar usuario (Elaboración propia)

El diagrama de clases está compuesto por un server_page, 2 client_page, un formulario, una clase controladora y un modelo. Un usuario accede a través de una url a la client_page Login, la cual contiene un formulario en donde el usuario debe introducir los datos (usuario y contraseña), una vez introducidos estos datos son enviados a la clase controladora views a través de una url contenida en la server_page, la clase controladora validará los datos en la clase UserModel, si los datos son correctos la clase controladora a través de una url renderizará la client_page Listar.html, si son incorrectos se renderizará la client_page Login.

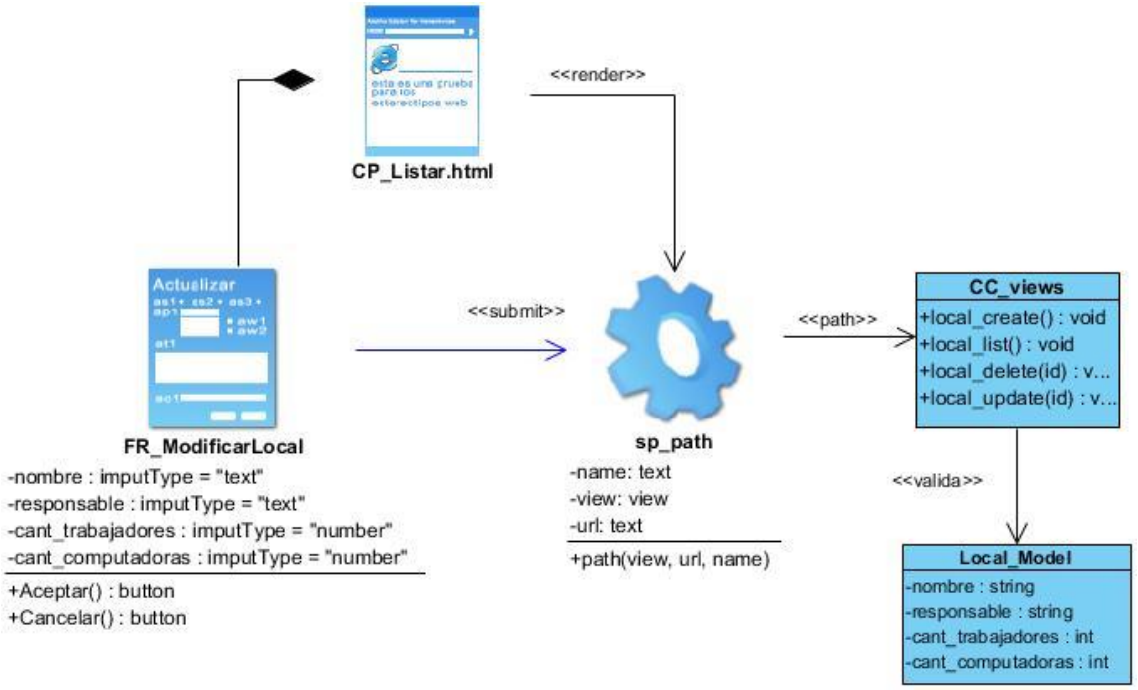

Figura 12 Diagrama de clase del RF 4 Modificar local (Elaboración propia)

El diagrama de clases está compuesto por un server_page, una client_page, un formulario, una clase controladora y un modelo. Un usuario accede a través de una url a la client_page Listar.html donde se visualiza un listado de locales, al seleccionar un local para modificar sus datos, la client_page contiene un formulario con los datos de los locales (nombre, responsable, cantidad_trabajadores, cant_computadoras), el usuario hace las modificaciones convenientes y se envían esos datos a través de una url a la clase controladora, esta guardará los datos en la clase Local_Model y la server_page renderizará la client_page Listar.html.

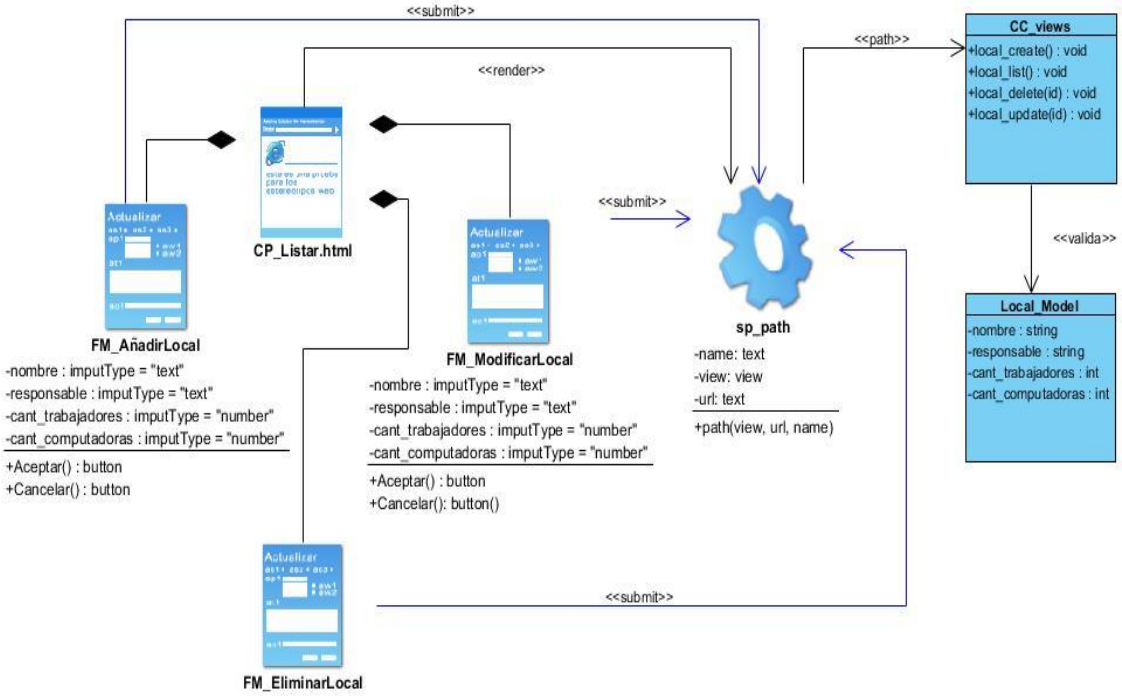

Figura 13 Diagrama de clase de los requisitos añadir, modificar y eliminar local (Elaboración propia)

El diagrama de clases está compuesto por un server_page, una client_page, tres formularios, una clase controladora y un modelo. El usuario se encuentra en la client_page Listar.html donde puede añadir, modificar y eliminar un local, en función de la operación que el usuario seleccione se visualizará el formulario correspondiente, estos datos pasarán a la clase controladora a través de una url, la clase controladora valida los datos en la clase Local_Model, luego de validados el server_page renderiza a la client_page Listar.html.

2.5 Arquitectura de software

Un diseño arquitectónico es un conjunto de patrones coherentes y abstracciones que proporcionan un marco definido y claro para interactuar con el código fuente del software (53).

Django es conocido como un Framework Modelo-Vista-Plantilla (MVT, por sus siglas en ingles), esta arquitectura tiene como principio que cada uno de los componentes esté separado en diferentes objetos, lo que quiere decir que los componentes no se pueden combinar dentro de una misma clase.

El patrón de diseño MVT es constituido por tres componentes:

- **Modelo:** contiene sólo los datos de aplicación más puros, como entidades u objetos de la base de datos, no contiene ninguna lógica o método para presentar dichos datos al usuario.

- **Vista:** existe entre el template y el modelo, es la encargada de tomar los datos del modelo, procesarlos y enviarlos a la plantilla.
- **Plantilla:** presenta los datos del modelo al usuario. La plantilla accede a los datos del modelo, pero no interpreta o manipula la información.

Las tres partes de MVT están interconectadas, la plantilla muestra los datos generados por la vista, la cual los extrae y modifica del modelo.

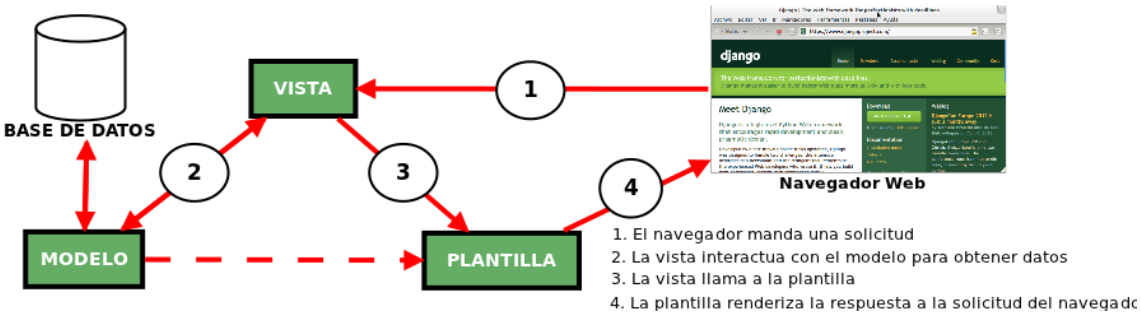

Figura 14 Funcionamiento de la arquitectura MVP (Sergio Infante Montero, 2012)

En la figura se presenta como fue empleado el patrón MVT en la herramienta para la gestión centralizada de dispositivos USB en GNU/Linux Nova. En ella se presenta la estructura de la herramienta en las siguientes capas:

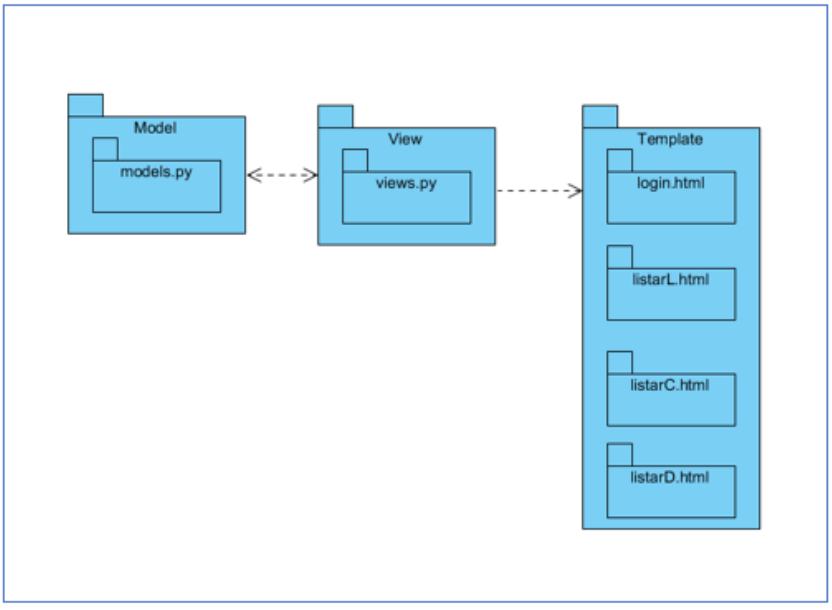

Figura 15 Diagrama de paquetes de la propuesta de solución (Elaboración propia)

2.6 Patrones de diseño

Un patrón de diseño describe problemas comunes de software y detalla la solución de este, la cual debe ser reutilizable. Son descripciones de comunicación entre objetos y

clases que pueden adaptarse para resolver un problema de diseño general en un contexto particular (54).

Los Patrones Generales de Software para la Asignación de Responsabilidades (GRASP) son utilizados para describir los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (55).

- **Experto:** con este patrón se mantiene el encapsulamiento de la información en una clase a la cual le será asignada toda la responsabilidad de creación de objetos e implementación de métodos acorde a la lógica del negocio que le corresponda.

Este patrón se evidencia en la clase Local.

```
class Local(models.Model):
    computadora = models.ForeignKey(Computadora, blank=True, on_delete=models.CASCADE, null=True)
    nombre = models.CharField(max_length=255)
    responsable = models.CharField(max_length=255)
    numero_trabajadores = models.PositiveIntegerField()
    cantidad_computadoras = models.PositiveIntegerField()

    def __str__(self):
        return self.nombre

    class Meta:
        db_table = 'Local'
        ordering = ['id']
```

Figura 16 Patrón de diseño GRASP: Experto (Elaboración propia)

- **Alta Cohesión:** la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase y al objetivo para lo cual fue creada. Con la Cohesión Lógica se realizará múltiples tareas relacionadas, pero, en tiempo de ejecución, sólo una de ellas será llevada a cabo.

Este patrón queda evidenciado en cada una de las clases del diseño, ya que cada una responde con las funcionalidades que debe cumplir para realizar los trabajos correspondientes a cada clase.

```python
@login_required
def local_update(request, pk):
    if request.method == 'POST':
        nombre = request.POST['nombre']
        responsable = request.POST['responsable']
        numero_trabajadores = request.POST['numero_trabajadores']
        cantidad_computadoras = request.POST['cantidad_computadoras']
        modificar = Local.objects.get(pk=pk)
        if modificar is not None:
            if nombre:
                modificar.nombre = nombre
            if responsable:
                modificar.responsable = responsable
            if numero_trabajadores:
                modificar.numero_trabajadores = int(numero_trabajadores)
            if cantidad_computadoras:
                modificar.cantidad_computadoras = int(cantidad_computadoras)
            modificar.save()
            return redirect('local_list')
        return redirect('local_list')
    return redirect('local_list')
```

Figura 17 Patrón de diseño GRASP: Alta cohesión (Elaboración propia)

- **Bajo Acoplamiento:** maneja la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Un ejemplo donde se evidencia este patrón es entre las clases Computadora y Local, pues al existir una sola relación entre ambas, los cambios en una tienen poca influencia en el funcionamiento de la otra.

```python
class Local(models.Model):
    nombre = models.CharField(max_length=255)
    responsable = models.CharField(max_length=255)
    numero_trabajadores = models.PositiveIntegerField()
    cantidad_computadoras = models.PositiveIntegerField()

    def __str__(self):
        return self.nombre

    class Meta:
        db_table = 'Local'
        ordering = ['id']


class Computadora(models.Model):
    usuario = models.OneToOneField(User, blank=True, on_delete=models.CASCADE, null=True)
    local = models.ForeignKey(Local, blank=True, on_delete=models.CASCADE, null=True)
    serie = models.PositiveIntegerField(unique=True)
    ip = models.CharField(max_length=50)

    def __str__(self):
        return 'MB' + str(self.serie)

    class Meta:
        db_table = 'Computadora'
        ordering = ['id']
```

Figura 18 Patrón de diseño GRASP: Bajo acoplamiento (Elaboración propia)

2.7 Modelo de datos

Una base de datos es utilizada para guardar toda la información de un sistema determinado para su posterior uso, especifica las expresiones permitidas por el propio modelo, comunica las reglas y definiciones esenciales de los datos a los usuarios. El modelo de datos está formado por 3 elementos fundamentales:

• Entidades: objetos con existencia física o conceptual.

• Atributos: características que identifican o definen una entidad.

• Relaciones: dependencias o asociaciones entre las entidades.

El modelo entidad relación representa la relación entre dos o más entidades, ayuda a construir una estructura lógica de base de datos (56).

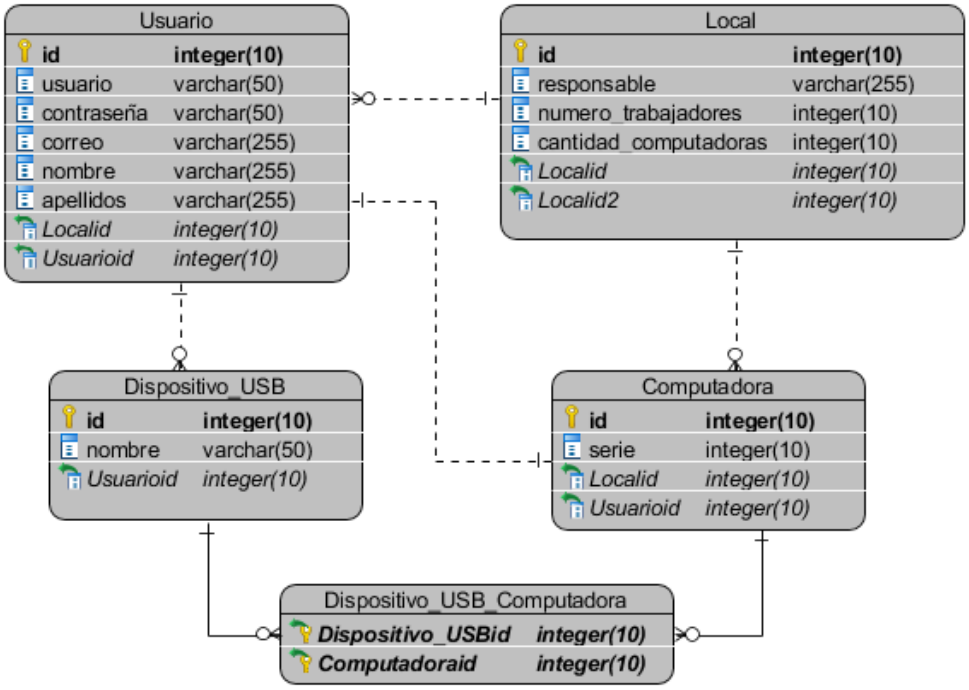

Figura 19 Modelo Entidad Relación (Elaboración propia)

Usuario: registra toda la información de los usuarios del sistema, se almacena en la base de datos los atributos que lo identifican como id, usuario, contraseña, correo, nombre y apellidos.

Local: registra toda la información de los locales del sistema, se almacena en la base de datos los atributos que lo identifican como id, responsable, cantidad de trabajadores y cantidad de computadoras.

Dispositivo USB: registra toda la información de los dispositivos USB del sistema, se almacena en la base de datos los atributos que los identifican como id, nombre y su estado.

Computadora: registra toda la información de las computadoras del sistema, se almacena en la base de datos los atributos que la identifican como id y serie.

Conclusiones del capítulo

Luego de definir las características del sistema se puede llegar a las siguientes conclusiones:

- El análisis de las herramientas para la gestión y control de dispositivos USB fue de gran utilidad para identificar las funcionalidades que se necesita en la propuesta y las restricciones que debe cumplir.
- Se definieron 8 requisitos funcionales y 7 requisitos no funcionales.

- Fue necesario realizar una estimación por HU para saber el tiempo y esfuerzo que puede llegar a necesitar el desarrollo de la propuesta.
- Se identificó el estilo arquitectónico y los patrones de diseño a utilizar en la propuesta, lo que permite realizar cambios futuros en el código sin generar gran impacto.
- Se generó el modelo de datos con el que contará el sistema de acuerdo a todas las funcionalidades y restricciones del mismo.

CAPÍTULO 3: Implementación y evaluación de la herramienta informática para la gestión centralizada de dispositivos USB

Introducción

Corresponde en este capítulo obtener los resultados de acuerdo a las disciplinas de implementación y pruebas del software. Se diseñan y aplican las pruebas para comprobar el correcto funcionamiento de la herramienta. Se muestran las conexiones del hardware y se realiza la validación de la propuesta.

3.1 Implementación de la propuesta de solución

En esta disciplina se lleva a cabo la implementación de las HU planificadas para cada iteración. El plan de iteraciones se revisa luego de concluir cada iteración con el objetivo de ajustarlo a los cambios aprobados. Para llevar a cabo la implementación de las HU se asignan tareas al equipo o persona que desarrollará la propuesta. Se debe emplear un lenguaje técnico ya que esta actividad cuenta con la participación de los desarrolladores.

Iteración No.1: Se implementa la HU de los Requisitos Funcionales Autenticar Usuario, Adicionar Usuario, Insertar Local, Eliminar Local y Listar Local quedando pendiente el Requisito Funcional Modificar Local.

Iteración No.2: Se culmina la implementación del Requisito Funcional Modificar Local y se implementan, además los Requisitos Funcionales Insertar Computadora, Modificar Computadora y Listar Computadora, quedando pendiente el Requisito Funcional Eliminar Computadora.

Iteración No.3: Se finaliza el Requisito Funcional Eliminar Computadora y se implementan los Requisitos Funcionales Mostrar listado de computadoras por local y Mostrar listado de dispositivos USB por computadora.

Como resultado de las iteraciones anteriores, se realizó la siguiente tabla que muestra el tiempo estimado y real de cada Historia de usuario y su respectiva iteración.

Tabla 12 Iteraciones por Historias de Usuario (Elaboración propia)

Historia de Usuario	Tiempo Estimado	Tiempo Real	Iteración
Autenticar usuario	4 horas	2 horas	1
Insertar local	4 horas	2 horas	1 y 2

Eliminar local	6 horas	4 horas	1 y 2
Modificar local	6 horas	4 horas	2
Mostrar local	6 horas	3 horas	2
Mostrar computadoras	6 horas	3 horas	3
Mostrar listado de computadoras por local	4 horas	2 horas	3
Mostrar listado de dispositivos USB por computadora	7 horas	4 horas	3

3.2 Estándares de Codificación

Los estándares de codificación son un conjunto de reglas que definen la apariencia del código y su estructura, facilitando que sea más legible y se pueda dar mantenimiento de forma más sencilla.

- La declaración de importación debe escribirse en líneas separadas.
- Utilizar 4 espacios para la sangría.
- De ser posible cada línea de código no debe exceder los 80 caracteres.
- Poner sufijo Exception a todas las excepciones.
- Poner sufijo Template a las vistas.
- Poner sufijo Model a los modelos.
- Poner el prefijo CC a las clases controladoras.
- Las variables usaran las letras en minúscula y separando las palabras con un guion bajo.
- Usar verbos para los métodos y sustantivos para atributos.
- Escribir comentarios para encontrar código más fácilmente.
- Los estilos van en archivos separados del código HTML.

En el desarrollo de la propuesta de solución se emplearon diferentes estilos que se muestran a continuación:

Definición de clases:

Para crear una clase se emplea la palabra reservada *class* seguido de un nombre escrito en minúscula, a excepción de la primera letra de cada palabra que se escribe en mayúscula y se usa el símbolo (:) como llave de apertura, se encuentra al final de la línea.

```
class Local(models.Model):
    nombre = models.CharField(max_length=255)
    responsable = models.CharField(max_length=255)
    numero_trabajadores = models.PositiveIntegerField()
    cantidad_computadoras = models.PositiveIntegerField()
```

Figura 20 Código Fuente - Definición de clases (Elaboración propia)

Definición de métodos:

Para crear un método se emplea la palabra reservada *def,* todas las funciones definidas en una clase deberán tener al menos un argumento.

```
def login(request):
    if request.method == 'POST':
        user = auth.authenticate(username=request.POST['usuario'], password=request.POST['contrasena'])
        if user is not None:
            auth.login(request, user)
            return redirect('local_list')
        else:
            return render(request, 'login.html', {'error': 'Usuario o Contraseña incorrecta'})
    else:
        return redirect('login_page')
```

Figura 21 Código Fuente - Definición de métodos (Elaboración propia)

Definición de variables:

Para crear una variable basta con poner un nombre, en estas no es necesario especificar el tipo de datos.

```
usuario = models.ForeignKey(Usuario, blank=True, on_delete=models.CASCADE, null=True)
local = models.ForeignKey(Local, blank=True, on_delete=models.CASCADE, null=True)
serie = models.PositiveIntegerField(unique=True)
```

Figura 22 Código Fuente - Definición de variables (Elaboración propia)

3.3 Diagrama de componentes

El diagrama de componentes muestra una vista de alto nivel de los componentes dentro de un sistema. Estos componentes pueden ser componentes de software como una base de datos o una interfaz de usuario; o componentes de hardware como un circuito o un dispositivo. Este diagrama está formado por tres elementos: componentes, interfaz y relación de dependencia (57).

A continuación, se muestra el Diagrama de Componentes generado por la presente investigación.

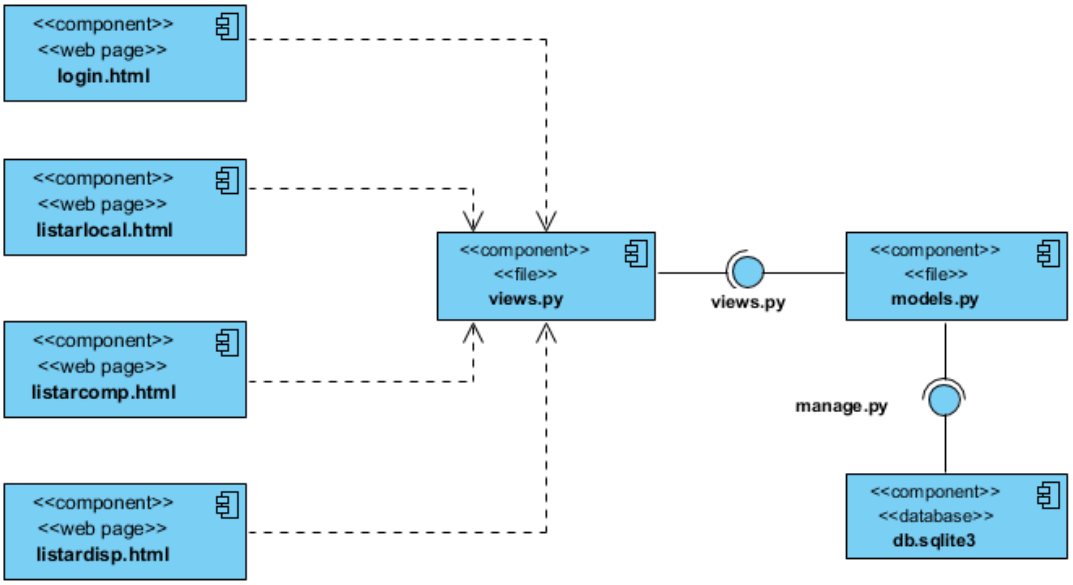

Figura 23 Diagrama de componentes (Elaboración propia)

3.4 Diagrama de Despliegue

El diagrama de despliegue es un tipo de diagrama UML que sirve para representar la relación de un sistema, utilizando nodos para realizar la expresión gráfica del mismo. Son de gran utilidad para representar sistemas de hardware y software y poder observar cómo se vería su relación de forma real (58).

En la figura se muestra el diagrama de despliegue que corresponde con la herramienta que se desea implementar.

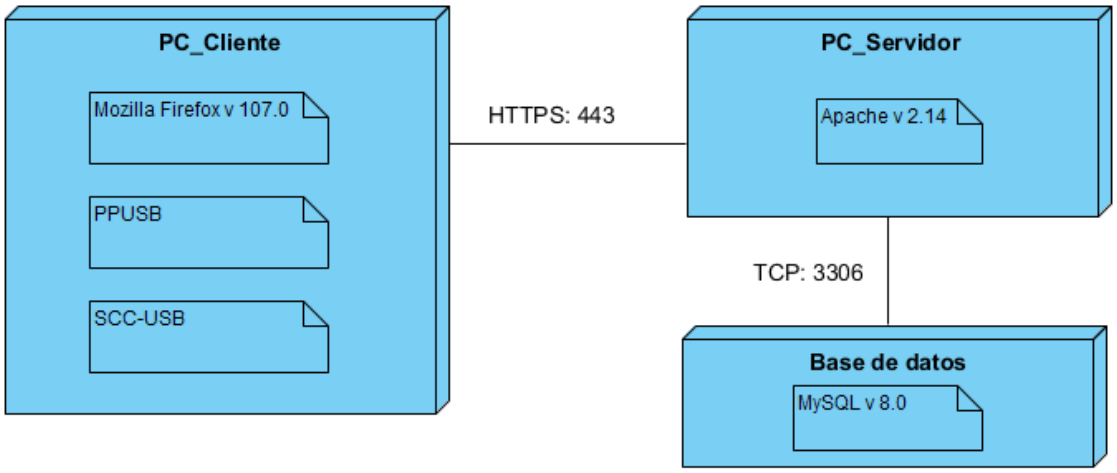

Figura 24 Diagrama de Despliegue (Elaboración propia)

En el diagrama anterior se muestra las distintas conexiones entre el hardware que se adecua a la herramienta que se desea implementar. El servidor es el host virtual donde se encuentra el servidor de la aplicación o sistema el cual gestiona la mayor parte de las funciones de acceso a los datos de la aplicación y la base de datos de la misma donde se almacena toda la información del sistema.

3.5 Pruebas de software

Las pruebas de software son un conjunto de procesos con los que se pretende probar un sistema en diferentes momentos para comprobar su correcto funcionamiento. Las pruebas de software, se definen como el proceso de aplicar unos pocos criterios de prueba de propósito general bien definidos a una estructura o modelo del software (59).

3.5.1 Estrategia de pruebas

Para la ejecución de las pruebas se emplearán los productos de trabajo donde se describió toda la información necesaria durante el desarrollo de la herramienta. Estos productos son:

- Especificación de requisitos
- Historias de Usuario

Para describir los tipos de pruebas que se le realizarán al sistema se tuvo en cuenta los requisitos adquiridos durante el diseño de la propuesta y las características de calidad identificadas en la norma (ISO/IEC 25010, 2011). En la tabla 6 se muestran los tipos de pruebas:

Tabla 13 Tipos de pruebas (Elaboración propia)

Tipos de prueba	Descripción
Funcionalidad	Se basa en el chequeo de los requisitos definidos durante el diseño de acuerdo a las funcionalidades presentes en el sistema.
Usabilidad	Se analiza el comportamiento de usuarios mientras realizan tareas en el sistema.
Seguridad	Analiza la seguridad del sistema en cuanto al acceso al mismo y a la información de personas autorizadas.
Portabilidad	Verifica que el producto pueda ser utilizado en diferentes entornos de hardware, software u otros.
Confiabilidad	Se encarga de verificar que los procesos de recuperación llevan al estado deseado y a su vez mantiene informados a los usuarios.

Mantenibilidad	Se analiza la facilidad con la que el producto puede ser modificado de forma efectiva debido a necesidades correctivas o evolutivas.

Las listas de chequeo son formatos generados para realizar acciones repetitivas que hay que verificar, son usadas en inspecciones o revisiones de artefactos generados en el proceso de producción de software (60).

Teniendo esto en cuenta fue necesario utilizar una lista de chequeo con el objetivo de analizar y comprobar que se cumplieran correctamente cada uno de los requisitos no funcionales con los que debe contar el sistema.

Los roles que intervinieron en el desarrollo de las pruebas son:

- Jefe de equipo
- Coordinador de prueba
- Desarrollador
- Probador

En la tabla 7 se muestra el cronograma de planificación de pruebas, además de la especificación en cada actividad.

Tabla 14 Cronograma de planificación de pruebas (Elaboración propia)

Actividades	Días	Responsable	Participantes
Elaborar la estrategia de prueba	Día 1	Coordinador de prueba	Coordinador de prueba
Diseñar la prueba	Día 2	Coordinador de prueba	Coordinador de prueba
Realizar el montaje del entorno de prueba	Día 3	Coordinador de prueba	Coordinador de prueba
Primera iteración de las pruebas	Día 4 y día 5	Coordinador de prueba	Probador
Corrección de defectos en los artefactos en prueba y actualización de los CP y artefactos de apoyo	Día 6	Jefe de equipo	Desarrollador
Segunda iteración de las pruebas	Día 7 y día 8	Coordinador de prueba	Probador

Evaluación de los resultados de las pruebas	Día 9	Coordinador de prueba	Probador

3.5.2 Método de caja negra

Es el método en el cual el elemento es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Estas pruebas son realizadas desde la interfaz gráfica (61).

Para desarrollar las pruebas de caja negra existen varias técnicas:

- Técnica de partición de equivalencia.
- Técnica de análisis y valor límite.

3.5.3 Técnicas de prueba

Partición de equivalencia

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada (62).

3.5.4 Aplicación de las pruebas de software

En el presente epígrafe se presentan los resultados obtenidos de las diferentes pruebas de software anteriormente definidas y aplicadas empleando los métodos y técnicas descritas en los subepígrafes 3.5.2 y 3.5.3.

2.5.6 Pruebas Funcionales

Las pruebas funcionales garantizan que las características y funcionalidades del software se comportan según lo esperado sin ningún problema. Valida principalmente toda la aplicación con respecto a las especificaciones mencionadas. Dentro de las ventajas de la aplicación de estas pruebas están: asegura de que el sitio web o aplicación está libre de defectos, garantiza el comportamiento esperado de todas las funcionalidades entre otras (63).

Para el desarrollo de estas pruebas la autora selecciona la técnica de caja negra o también llamadas pruebas de comportamiento las cuales permiten comprobar el comportamiento del sistema dado un conjunto de condiciones de entrada.

Con el objetivo de mejorar el margen de confianza, de que se encontrarán todos los defectos, se emplearon los Diseños de Casos de Prueba (DCP) para verificar las diversas funcionalidades del sistema, descritas en el formato de las Historias de Usuario (HU).

En el Diseño de Caso de Prueba (DCP) se describieron las variables pertenecientes a cada uno y los valores que las mismas deben tomar para la realización de las pruebas.

A continuación, se presenta un diseño de caso de prueba utilizado para comprobar que se cumplen los requerimientos del sistema y detectar cualquier error en la aplicación.

Los valores de las variables pueden ser: Válidos (color verde) e Inválidos (color rojo).

Descripción general

El sistema permite autenticar usuario.

Condiciones de ejecución

El usuario debe estar autenticado en el sistema con anterioridad.

RF Autenticar Usuario

Escenario	Descripción	Usuario	Contraseña	Respuesta del sistema	Flujo central
Autenticar usuario de forma correcta	El sistema autentica un usuario de forma correcta.	admin	123	El sistema autentica al usuario y le permite el acceso a las funcionalidades del sistema.	1-El usuario accede a la url del portal web. 2- El sistema muestra una interfaz con el formulario de autenticación. 3- El usuario introduce la información y presiona el botón: "Iniciar sesión".
Autenticar usuario de forma incorrecta	El sistema no autentica un usuario de forma incorrecta.	admin	admin123	El sistema no autentica al usuario y emite el emnsaje "Campos Inválidos"	
		ADMIN	123		
		.123	123		
	El sistema no autentica un usuario dejando campos vacíos.	admin			
			123		
Cerrar Sesión en el Sistema	El sistema desloguea al usuario de forma correcta.			Al usuario cerrar la sesión, se desloguea del sistema y este lo dirige a la portada del mismo.	El usuario autenticado selecciona el botón "Cerrar sesión"

Figura 25 Diagrama de Caso de Prueba RF Autenticar Usuario (Elaboración propia)

Descripción general

El sistema permite adicionar un local.

Condiciones de ejecución

El usuario debe estar autenticado en el sistema con anterioridad.

RF Adicionar local

Escenario	Descripción	Nombre	Responsable	Número de trabajadores	Cantidad de computadoras	Respuesta del sistema
Crear local de forma correcta	El sistema inserta un local de forma correcta.	Laboratorio 102	Carlos Sánchez Martínez	15	15	El sistema adiciona el local.
Crear local dejando campos vacíos.	El sistema no inserta un local dejando campos vacíos.					El sistema no adiciona al local y emite el mensaje "No se pudo añadir el local:Campos Vacíos"
Cancelar operación	El sistema no realiza ninguna operación.					No realiza ninguna operación y regresa a la página anterior.

Figura 26 Diagrama de Caso de Prueba RF Adicionar local (Elaboración propia)

Descripción general

El sistema permite modificar un local.

Condiciones de ejecución

El usuario debe estar autenticado en el sistema con anterioridad.

RF Modificar local

Escenario	Nombre	Responsable	Número de trabajadores	Cantidad de computadoras	Respuesta del sistema
Modificar usuario de forma correcta	Laboratorio 102	Carlos Sánchez Martínez	15	15	El sistema modifica el local.
Modificar usuario dejando campos vacíos.				10	El sistema modifica el local.
Modificar usuario con valores existentes.	valor existente				El sistema no modifica al local y emite el mensaje "No se pudo modifica el local: El nombre ya existe"
Cancelar operación					No realiza ninguna operación y regresa a la página anterior.

Figura 27 Diagrama de Caso de Prueba RF Modificar local (Elaboración propia)

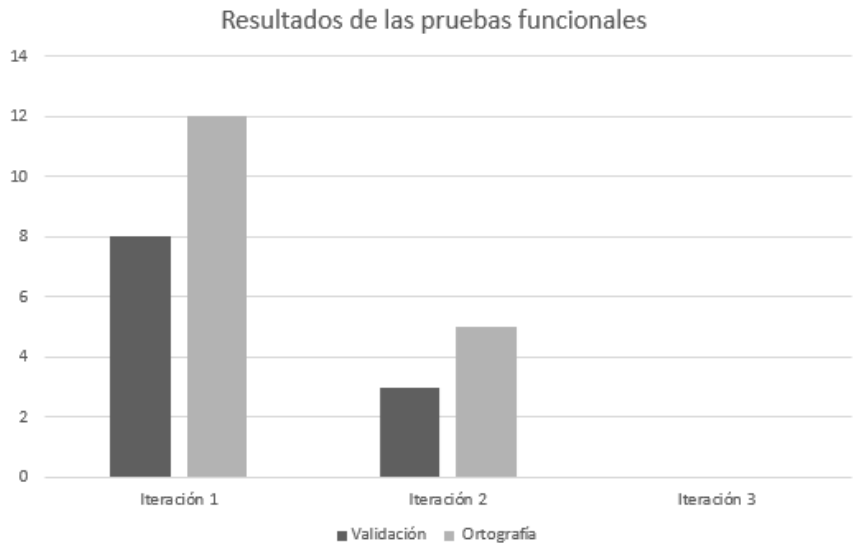

Figura 28 Resultado de las pruebas funcionales (Elaboración propia)

3.5.7 Pruebas de regresión

La prueba de regresión es un tipo de prueba que se realiza para verificar que un cambio de código en el software no afecta la funcionalidad existente del producto. Esto es para asegurarse de que el producto funcione bien con nuevas funciones, correcciones de errores o cualquier cambio en la función existente. Los casos de prueba ejecutados previamente se vuelven a ejecutar para verificar el impacto del cambio (64).

En el caso de la solución se realizaron las pruebas de regresión descritas al Gestionar Local y se arrojaron distintos resultados en tres distintas iteraciones:

Tabla 15 Especificación de Pruebas de Regresión (Elaboración propia)

Iteración	Error a Resolver	Problemas Encontrados
Iteración 1	No era posible modificar un local.	Al modificar un local no se modificaba el local correspondiente al identificador enviado.
	No era posible eliminar un local.	Al eliminar un local no se eliminaba el local correspondiente al identificador enviado.
	No era posible listar las computadoras.	Al listar las computadoras, se listaban los locales.

Iteración 2	Se corrigió el error 1 y 2 de la Iteración 1.	La interfaz de eliminación de local mostraba errores en el *responsive design.*
Iteración 3	Se corrigió el error 3 de la iteración 1 y el error encontrado en la Iteración 2.	No se encontraron problemas.

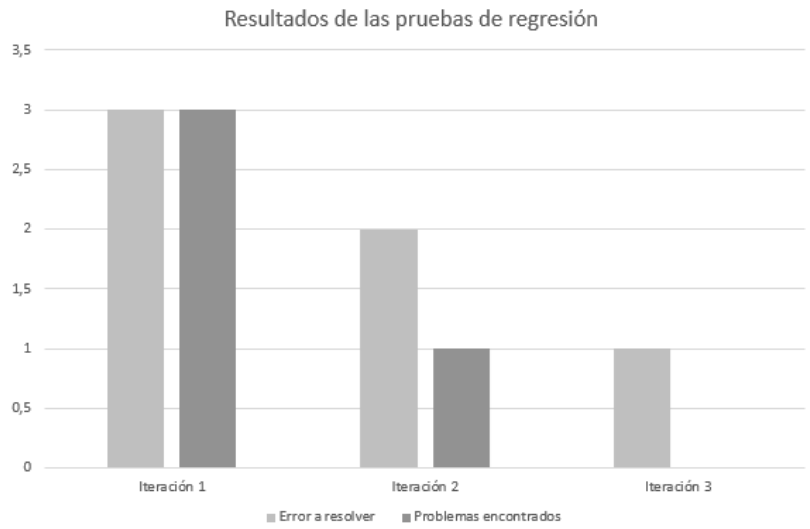

Figura 29 Resultados de las pruebas de regresión (Elaboración propia)

3.5.8 Pruebas de Usabilidad

Las pruebas de usabilidad refieren a un método para probar la funcionalidad de un sitio web, una aplicación u otro producto, y consisten en evaluar un producto o servicio probándolo con usuarios representativos reales mientras intentan completar tareas en él. El objetivo es identificar cualquier problema de usabilidad, recoger datos cualitativos y cuantitativos, así como determinar la satisfacción del participante con el producto (65). La usabilidad es la capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas según la ISO/IEC 25000 (66):

- Capacidad para reconocer su adecuación: capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.

- Capacidad de aprendizaje: capacidad del producto que permite al usuario aprender su aplicación.
- Protección contra errores de usuario: capacidad del sistema para proteger a los usuarios de hacer errores.
- Estética de la interfaz de usuario: capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- Accesibilidad: capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

Tabla 16 Lista de Chequeos para Pruebas de Usabilidad (Elaboración propia)

Elementos definidos por la metodología				
Número	**Indicador a evaluar**	**Evaluación**	**NP**	**Observación**
Visibilidad del sistema				
1	¿Cuándo se selecciona un icono se diferencia de los no seleccionados?	Si		
2	¿La página refleja la identidad de la organización?	Si		
3	¿El menú de navegación aparece en un lugar destacado?	Si		
5	¿Las imágenes se muestren con buena resolución?	Si		
Lenguaje común entre sistema y usuario				
7	¿El lenguaje es simple, con un tono adecuado?	Si		
8	¿La información que se presenta en la aplicación es fácil de	Si		

	entender y memorizar?			
9	¿La información es de rápida lectura, y con una disposición asequible?	Si		
10	¿Se utiliza siempre la misma nomenclatura para las mismas funciones?	Si		
11	¿Utiliza los conceptos establecidos para las funciones estándar? (“buscar" para las búsquedas, etc.)	Si		
Prevención de errores				
12	¿Existe suficiente espacio entre los elementos de acción (links, botones, etc.) para prevenir que el usuario haga click en el elemento incorrecto?	Si		
13	¿Hay ausencia de enlaces rotos o que no lleven a ninguna página?	Si		
14	¿Los botones de acción, (tales como “Enviar”) siempre son invocados por el	Si		

	usuario y no automáticamente invocados por el sistema cuando el último campo de un formulario ha sido lleno?			
15	¿El buscador (si existe) permite errores tipográficos y ortográficos (tildes)?	Si		
16	¿Se evita el contenido importante del sitio en ventanas emergentes?	Si		
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores				
17	¿En caso de errores de consistencia dentro del sitio, ¿se ofrece un mensaje de personalizado mediante una página explicativa?, (Por ejemplo: Error 404 para página inexistente)	Si		
18	¿El mensaje de error permite volver a la situación anterior?	Si		
19	¿El sitio está diseñado para necesitar el mínimo	Si		

	de ayuda y de instrucciones?			
Flexibilidad y eficiencia de uso				
20	¿Las páginas no requieren volver a escribir la información solicitada en páginas anteriores?	Si		
21	¿El cursor se desplaza adecuadamente en un formulario al presionar “tabulador”?	Si		
22	¿Las partes o secciones más importantes de los sitios son accesibles desde la página de inicio?	Si		

3.6 Pruebas de Aceptación

Las pruebas de aceptación se realizan para verificar que el software cumple las expectativas desde el punto de vista del cliente y de los usuarios finales. Para realizar una prueba de aceptación y decidir si se acepta el software, se necesitan criterios claros, estos son los llamados criterios de aceptación (67).

Los criterios de aceptación funcional indican cómo debe comportarse el software para ayudar a los usuarios a realizar su trabajo. Se refieren a las funciones, o características, que ofrece el software.

Los criterios de aceptación no funcionales se refieren a cómo el software hace lo que hace: aspectos como la accesibilidad, la facilidad de uso, las garantías de seguridad y privacidad, la velocidad, la fiabilidad y muchos más (67).

Tipos de Pruebas de Aceptación:

- Pruebas de Aceptación del Usuario (UAT).

- Pruebas de Aceptación Operativa (OAT).
- Pruebas de Aceptación de la Normativa.
- Pruebas de Aceptación Alfa y Beta.

El cliente realizó pruebas funcionales en las que no se detectaron no conformidades y emitió un acta de aceptación de los productos de trabajo en total conformidad con la solución desarrollada.

Conclusiones del capítulo

Con el desarrollo de este capítulo se abordó una serie de aspectos correspondientes a la implementación y la validación de la propuesta de solución. El empleo de los estándares de codificación permitió adoptar una estructura homogénea asegurando el fácil mantenimiento del sistema y la alta calidad del código. Con la representación del modelo de despliegue se pudo exponer como se relacionan los elementos de hardware del sistema. Se demostró que se requieren pruebas de software para señalar los errores que ocurren durante las fases de desarrollo a través estas se asegura de que las funcionalidades de la aplicación sean adecuadas y de que los clientes estén satisfechos con ella. Cuando el producto entregado es de calidad, ayuda a ganar la confianza de los clientes.

Conclusiones

Con el desarrollo de esta investigación se puede arribar a la conclusión que el análisis del diseño teórico sobre el proceso de gestión centralizada de dispositivos USB es de suma importancia para la elaboración de esta herramienta.

El trabajo con una metodología de desarrollo de software permitió reducir el nivel de dificultad, organizar las tareas, agilizar el proceso y mejorar el resultado final de la aplicación a desarrollar.

La definición de las tecnologías y herramientas de desarrollo maximizaron la calidad en todo el ciclo de vida al desarrollar la aplicación informática mejorando la productividad y automatizando los procesos de creación, normalización y generación de la base de datos.

Los 8 requisitos funcionales y los 7 no funcionales definidos permitieron cumplir con las necesidades del cliente. El análisis y diseño propició la elaboración de una herramienta para la gestión centralizada de dispositivos USB. El uso de los patrones de diseño GRASP y el patrón arquitectónico MVP brindó una mayor calidad al software desarrollado.

La aplicación de las pruebas de software permitió evaluar la solución desarrollada y garantizar el correcto funcionamiento de la herramienta para la gestión centralizada de dispositivos USB en GNU/Linux Nova, logrando la total satisfacción del cliente.

Recomendaciones

Se recomienda agregar la funcionalidad de exportar informes como PDF de toda la información sobre los dispositivos autorizados y denegados.

Bibliografía

1. globalbit. globalbit. *globalbit.* [En línea] 20 de julio de 2019. https://www.globalbit.co/2019/07/20/el-alcance-del-software-en-el-mundo-actual-y-su-impacto-en-el-futuro/.

2. [En línea] https://concepto.de/computadora/.

3. R, María Estela. 20 Ejemplos de Periféricos y su función. *20 Ejemplos de Periféricos y su función.* [En línea] 5 de febrero de 2021. https://www.ejemplos.co/20-ejemplos-de-perifericos-y-su-funcion/.

4. Editorial Etecé. Enciclopedia concepto. *Enciclopedia concepto.* [En línea] 5 de agosto de 2021. https://concepto.de/usb/.

5. Blanch, Alberto. Emprende con Arsys. [En línea] 5 de junio de 2018. https://www.arsys.es/blog/proteger-informacion-empresa.

6. Editorial Etecé. Enciclopedia Concepto. [En línea] 22 de octubre de 2021. https://concepto.de/malware/.

7. Universidad de las Ciencias Informáticas. [En línea] https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo.

8. humanos. [En línea] https://humanos.uci.cu/nova/.

9. *Sistema centralizado para la protección de puertos USB en Nova.* Luis Daniel Sierra Corredera, Juan Manuel Fuentes Rodríguez, Arletis Ortíz Rodríguez. La Habana, Cuba : s.n., 2018. Sistema centralizado para la protección de puertos USB en Nova. pág. 6.

10. Economía gestione a su favor. [En línea] febrero de 2014. https://economia.org/gestion.php.

11. Westreicher, Guillermo. Economipedia. [En línea] 07 de agosto de 2020. [Citado el: 20 de junio de 2022.] https://economipedia.com/definiciones/gestion.html.

12. Porto, Julián Pérez. Definición de . [En línea] 2021. [Citado el: 20 de junio de 2022.] https://definicion.de/gestion/.

13. Significados. [En línea] [Citado el: 10 de noviembre de 2022.] https://www.significados.com/centralizacion-y-descentralizacion/.

14. Real Academia Española. [En línea] [Citado el: 20 de junio de 2022.] https://dle.rae.es/centralizar.

15. Gardey, Julián Pérez Porto y Ana. Definición de. [En línea] 2014. [Citado el: 20 de junio de 2022.] https://definicion.de/centralizacion/.

16. Quiroa, Myriam. Economipedia. [En línea] 7 de diciembre de 2020. [Citado el: 17 de octubre de 2022.] https://economipedia.com/definiciones/areas-funcionales-de-una-empresa.html.

17. Manage Engine. [En línea] 2022. https://www.manageengine.com/latam/desktop-central/gestion-control-dispositivos-usb.html.

18. myservername.com. [En línea] 2022. https://spa.myservername.com/top-40-popular-j2ee-interview-questions.

19. [En línea] 2022. [Citado el: 23 de junio de 2022.] https://spa.myservername.com/cososys-endpoint-protector-review.

20. [En línea] 2022. https://spa.myservername.com/amazon-web-services-interview-questions-answers#Recommended_Reading.

21. EsGeeks. [En línea] [Citado el: 2022 de junio de 16.] https://esgeeks.com/usbguard-proteger-puertos-usb-en-linux/.

22. EsGeeks. [En línea] 4 de septiembre de 2018. [Citado el: 20 de junio de 2022.] https://esgeeks.com/usbguard-proteger-puertos-usb-en-linux/.

23. Santander Universidades. Becas Santander. [En línea] 21 de diciembre de 2020. https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html.

24. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana, Cuba : s.n., 2015.

25. Code Space. [En línea] 23 de julio de 2021. [Citado el: 13 de septiembre de 2022.] https://codespaceacademy.com/blog/diferencia-entre-desarrollador-backend-y-frontend/.

26. Coppola, Maria. Hubspot. [En línea] 19 de septiembre de 2022. [Citado el: 20 de junio de 2022.] https://blog.hubspot.es/website/frontend-y-backend.

27. MDN Web Docs. [En línea] 4 de agosto de 2022. [Citado el: 13 de septiembre de 2022.] https://developer.mozilla.org/es/docs/Web/HTML.

28. DesarrolloWeb.com. [En línea] [Citado el: 21 de junio de 2022.] https://desarrolloweb.com/home/html.

29. Mdn Web Docs. [En línea] 5 de septiembre de 2022. [Citado el: 13 de septiembre de 2022.] https://developer.mozilla.org/es/docs/Learn/CSS/First_steps/What_is_CSS.

30. Mdn Web Docs. [En línea] 3 de junio de 2022. [Citado el: 13 de septiembre de 2022.] https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

31. *Eloquent Javascript.* San Francisco, California : s.n., 2008.

32. Machuca, Fernando. Crehana. [En línea] 19 de mayo de 2022. [Citado el: 13 de septiembre de 2022.] https://www.crehana.com/blog/desarrollo-web/que-es-el-backend-y-como-usarlo/.

33. Qué es Python (desarrolloweb.com). [En línea] [Citado el: 5 de junio de 2022.] https://desarrolloweb.com/articulos/1325.php.

34. Introducción a Django - Aprende sobre desarrollo web | MDN (mozilla.org). [En línea] https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction.

35. Bootstrap: guía para principiantes de qué es, por qué y cómo usarlo. [En línea] 12 de abril de 2020. [Citado el: 13 de septiembre de 2022.] https://rockcontent.com/es/blog/bootstrap/.

36. Lago, Neybis. Saasradar. [En línea] 31 de mayo de 2022. [Citado el: 2022 de junio de 5.] https://saasradar.net/modelado-de-software/#abh_posts.

37. Krall, César. aprenderaprogramar.com. [En línea] [Citado el: 5 de junio de 2022.] https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163.

38. Visual Paradigm. [En línea] [Citado el: 2022 de junio de 5.] https://www.visual-paradigm.com/.

39. Funcionalidades - PyCharm (jetbrains.com). [En línea] [Citado el: 5 de junio de 2022.] https://www.jetbrains.com/es-es/pycharm/features/.

40. ComputerWeekly.es. [En línea] abril de 2021. [Citado el: 18 de noviembre de 2022.] https://www.computerweekly.com/es/definicion/MySQL.

41. Ferguson, Kevin. ComputerWeekly.es. [En línea] [Citado el: 5 de junio de 2022.] https://www.computerweekly.com/es/definicion/Servidor-de-aplicaciones-y-proveedor-de-servicios-de-aplicaciones.

42. B, Gustavo. Hostinger Tutoriales. [En línea] 23 de febrero de 2022. [Citado el: 5 de junio de 2022.] https://www.hostinger.es/tutoriales/que-es-apache/.

43. Lucidchart. [En línea] [Citado el: 10 de noviembre de 2022.] https://www.lucidchart.com/pages/es/que-es-un-mapa-conceptual.

44. Norén, Anders. Nat Apuntes. [En línea] 10 de enero de 2020. [Citado el: 23 de junio de 2023.] https://www.natapuntes.es/ingenieria-de-requisitos/.

45. Guerra, César Arturo. Software Guru. [En línea] [Citado el: 23 de junio de 2022.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

46. Editorial Etecé. Concepto. [En línea] Equipo editorial Etecé, 25 de septiembre de 2020. [Citado el: 1 de julio de 2022.] https://concepto.de/entrevista/.

47. Suárez, Jennifer Delgado. Rincón de la psicología. [En línea] [Citado el: 1 de julio de 2022.] https://rinconpsicologia.com/tormenta-de-ideas-tecnica-ejemplos/.

48. Felipe. HostingPlus. [En línea] 6 de julio de 2021. [Citado el: 1 de julio de 2022.] https://www.hostingplus.pe/blog/modelo-de-prototipos-que-es-y-cuales-son-sus-estapas/.

49. *La especificación de requerimientos de software desde la perspectiva de un nuevo.* Ramón Ventura Roque Hernández, Rebeca Díaz Redondo, Ana Fernández Vilas. 3, Ciudad Victoria, México : CienciaUAT, 2012, Vol. 6.

50. Requeridos Blog. Requeridos Blog. [En línea] 20 de abril de 2018. [Citado el: 25 de junio de 2022.] https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a.

51. Asana. [En línea] 21 de enero de 2022. [Citado el: 12 de julio de 2022.] https://asana.com/es/resources/user-stories.

52. Creately. [En línea] 10 de mayo de 2022. [Citado el: 12 de julio de 2022.] https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-clases/.

53. Adriana Sandra Almeira, Vanina Perez Cavenago. Silo.tips. [En línea] 2007. [Citado el: 7 de julio de 2022.] https://silo.tips/download/arquitectura-de-software-estilos-y-patrones#.

54. Rojas, Edson Bryan Castañeda. PUPC. [En línea] 26 de noviembre de 2016. [Citado el: 10 de julio de 2022.] https://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/7513.

55. Garzas, Javier. javiergarzas.com. [En línea] 5 de agosto de 2014. [Citado el: 10 de julio de 2022.] https://www.javiergarzas.com/2014/08/los-patrones-grasp.html.

56. Edrawsoft. [En línea] 2021. [Citado el: 12 de julio de 2022.] https://www.edrawsoft.com/es/er-diagram/.

57. Diagramas UML. [En línea] [Citado el: 21 de octubre de 2022.] https://diagramasuml.com/componentes/.

58. Plantilla Árbol Genealógico. [En línea] 2022. [Citado el: 8 de septiembre de 2022.] https://plantillaarbolgenealogico.net/diagramas/despliegue/.

59. Turrado, Jorge. CampusMVP.es. [En línea] 10 de marzo de 2020. [Citado el: 10 de septiembre de 2022.] https://www.campusmvp.es/recursos/post/que-son-las-pruebas-de-software.aspx.

60. 9001:2015, ISO. [En línea] 28 de junio de 2022. [Citado el: 17 de octubre de 2022.] https://www.nueva-iso-9001-2015.com/2022/06/checklist-para-controlar-sistema-gestion-de-calidad-sector-industrial/.

61. Marquez, Antonio. TesterModerno. [En línea] 15 de marzo de 2020. [Citado el: 22 de octubre de 2022.] https://www.testermoderno.com/caja-blanca-vs-caja-negra/.

62. [En línea] [Citado el: 14 de noviembre de 2022.] http://html.rincondelvago.com/metodos-de-prueba-caja-de-pandora.html.

63. LoadView . [En línea] 16 de octubre de 2020. [Citado el: 10 de noviembre de 2022.] https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/.

64. myservername.com. [En línea] [Citado el: 22 de octubre de 2022.] https://spa.myservername.com/what-is-regression-testing.

65. QuestionPro. [En línea] [Citado el: 22 de octubre de 2022.] https://www.questionpro.com/blog/es/pruebas-de-usabilidad/.

66. ISO 25000 calidad de software y datos. [En línea] https://iso25000.com/index.php/normas-iso-25000/iso-25010/23-usabilidad.

67. Digité. [En línea] [Citado el: 23 de octubre de 2022.] https://www.digite.com/es/agile/pruebas-de-aceptacion/.

ANEXOS

Anexo 1

Entrevista para obtener información sobre el proceso de gestión y control de dispositivos USB en Nova.

Nombre: _____ **Especialidad:** _____

¿Qué le gusta más de esta profesión? _____

¿Ha realizado estudios en esas temáticas? ¿En cuáles? _____

¿Dónde trabaja ahora? _____

¿Qué cargo ocupa? _____

¿Cuánto tiempo lleva ejerciendo esa responsabilidad? _____

¿Cuál es el mayor reto de su equipo de trabajo en la actualidad? _____

1. ¿Considera necesario el proceso de gestión y control de dispositivos USB? ¿Por qué?
2. ¿Conoce usted herramientas que gestionen y controlen los dispositivos USB? ¿Cuáles?
3. ¿Cuáles son los elementos más comunes en el proceso de gestión y control de dispositivos USB?
4. ¿Qué sabe usted sobre PPUSB? ¿Conoce su funcionamiento?
5. ¿Considera oportuno contar con una herramienta que permita controlar centralmente los dispositivos USB en Nova?
6. ¿Qué características debe tener esta herramienta?

Anexo 2

Encuesta a especialistas del Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI)

Objetivo: evaluar la satisfacción de los usuarios potenciales de la solución desarrollada. Especialista, le invito a responder el siguiente cuestionario con el fin de conseguir su colaboración en la presente investigación, se solicita que exprese en sus repuestas, criterios verídicos que guíen a la autora del trabajo. Marque en cada pregunta con una X en una sola opción y en el caso de la 5 responda brevemente. Muchas gracias por el tiempo brindado.

1. ¿Considera usted necesario el uso de aplicaciones que permitan la gestión y el control de los dispositivos USB en Nova?

   ____ Si ____ No ____ No sé

2. ¿Actualmente Nova cuenta con una herramienta que gestione y controle de manera central los dispositivos USB?

   ____ Si ____ No ____ No sé

3. ¿Considera usted que es necesario contar con una herramienta que gestione y controle de manera central los dispositivos USB en Nova?

   ____ Si ____ No ____ No sé

4. Luego de haber interactuado con la herramienta para la gestión y control de dispositivos USB, refleje en qué medida le gusta la solución desarrollada:

   __Me gusta mucho

   __Me gusta más de lo que me disgusta

   __Me da lo mismo

   __Me disgusta más de lo que me gusta

   __No me gusta

   __No sé decir

5. ¿Qué opina acerca de los beneficios que trae consigo la herramienta para la gestión centralizada de dispositivos USB en Nova?