Universidad de las Ciencias Informáticas

Facultad 1

Herramienta web de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0

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

Autor: Beatriz García Jiménez

Tutores: Dra. Berta Irailis Yanes Watson

MSc. Nurisel Palma Pérez

Ing. Enrique Muschett Cortina

La Habana, julio de 2020
"Año 62 de la Revolución"

DEDICATORIA

A mis padres por su apoyo incondicional.

A todas las personas que de una forma u otra contribuyeron a la realización de este trabajo.

AGRADECIMIENTOS

A mis padres porque sin ellos nada de esto hubiese sido posible. Gracias por siempre apoyar mis decisiones y darme todo su cariño.

A mi novio Sergio, a mi familia, a mis amigos y a mis tutores, gracias por su apoyo.

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo **Beatriz García Jiménez**, con carné de identidad **96062107694** soy la autora principal del trabajo titulado **"Herramienta web de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes _____ del_____

_____

Beatriz García Jiménez

Autora

_____ _____

Dra. Berta Irailis Yanes Watson MSc. Nurisel Palma Pérez

Tutor Tutor

_____

Ing. Enrique Muschett Cortina

Tutor

RESUMEN

La visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo es necesaria para realizar inspecciones, auditorías informáticas, medir el desempeño de los equipos o diagnosticar posibles fallas. En la Distribución Cubana de GNU/Linux Nova Servidores 7.0, el proceso de visualización de los detalles técnicos se realiza de forma manual, a través de comandos de consola. La presente investigación tuvo como objetivo desarrollar una herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0. Para el desarrollo de la investigación se emplearon los métodos científicos: teóricos (Analítico-Sintético, Inductivo-Deductivo e Histórico-Lógico) y empíricos (Entrevista y Observación). En la implementación de la propuesta de solución se utilizaron los *Framework* de desarrollo *Flask* y *Bootstrap*, como herramienta de modelado *Visual Paradigm* y como lenguajes de programación *Python*, *Jinja2*, *HTML*, *CSS* y *JavaScript*. Se aplicaron los patrones de diseño: Experto, Controlador y Bajo Acoplamiento; y se definió el patrón arquitectónico en capas (Capa de Presentación y Capa de Lógica del negocio). La metodología Variación de *AUP* para la UCI, guió el proceso de desarrollo de *software*. Se desarrolló una herramienta web que permite la visualización de los detalles técnicos del equipo. Los recursos de *hardware* que se visualizan son: *CPU*, memoria, almacenamiento, placa base, red y monitor; y los recursos de *software* son: sistema operativo, los servicios instalados y los procesos en ejecución. La herramienta permite al usuario exportar la información en formatos PDF o HTML e imprimirla.

Palabras claves: detalles técnicos, *hardware***,** herramienta web, *software*.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La Política Integral para el Perfeccionamiento de la Informatización de la Sociedad en Cuba, propuesta por el Ministerio de Comunicaciones, está asociada al cumplimiento del Lineamiento 108 de la Política Económica y Social aprobada por el 7mo Congreso del Partido (Partido Comunista de Cuba, 2017). La misma asegura la sostenibilidad y soberanía tecnológica. Ella contempla la implementación de políticas específicas, entre ellas: la realización del Programa Nacional de Informatización como parte del Plan Nacional de Desarrollo Económico Social hasta el 2030 y la ejecución del Sistema Nacional de Seguridad Tecnológica. Esta última, incluye el siguiente principio, directamente relacionado con esta tesis:

- Implementar soluciones y la infraestructura que garanticen la seguridad tecnológica en el proceso de informatización del país (Ministerio de Comunicaciones, 2017).

En este contexto y como parte de la política del Estado Cubano de migrar a *software* libre, las empresas cubanas avanzan en la informatización de sus procesos de gestión y productivos. Es importante durante esta etapa, el despliegue de aplicaciones y servicios. Entre estos servicios se encuentran: el correo corporativo, la página web de la empresa, las actualizaciones en línea del antivirus, los sistemas de bases de datos para la gestión de la contabilidad y los recursos humanos, entre otros. Por otra parte, para el equipo de especialistas en informática de la empresa, se convierte en una tarea de suma importancia la gestión de la seguridad y los controles de accesos, así como el monitoreo de las redes para detectar intrusos e intentos de accesos no autorizados a los datos de la empresa. Estos requerimientos obligan a decidir qué tecnología usar para garantizar seguridad, eficiencia en los servicios y disminuir los costos de mantenimiento de los sistemas a partir de los recursos con los que cuente la empresa.

Con la mirada puesta en la soberanía tecnológica del país, en garantizar la seguridad informática y facilitar la informatización de las empresas, con un conjunto de herramientas ajustadas a las necesidades del entorno empresarial cubano, surge el proyecto Nova Servidores. Esta distribución de GNU/Linux es desarrollada en la Universidad de las Ciencias Informáticas (UCI), por el Centro de *Software* Libre (CESOL), el cual es líder en los procesos de migración hacia tecnologías libres y de código abierto.

Un sistema informático como Nova Servidores está compuesto por dos subsistemas que reciben los nombres de *software* y *hardware*; el primero consiste en la parte lógica de una computadora (programas, aplicaciones, entre otros), y el segundo es la parte física (elementos como la placa base, la memoria,

entre otros). Ambos conforman el conjunto de recursos tecnológicos (intangibles y físicos) que emplea Nova Servidores con el propósito de satisfacer las peticiones de las aplicaciones clientes.

Según el diccionario de la lengua española (RAE), el concepto de Cliente – Servidor se refiere a un modelo de comunicación que vincula a varios dispositivos informáticos a través de una red (RAE, 2019a). El cliente realiza peticiones al servidor, y el servidor se encarga de satisfacer dichos requerimientos. Este modelo permite realizar un empleo eficiente de los recursos de cómputo de las empresas, las cuales destinan los recursos más potentes de *hardware* para realizar las funciones de servidores; mientras que el resto de las computadoras se utilizan para desplegar las aplicaciones clientes.

Nova Servidores está orientada a servidores que usan estándares abiertos. Entre sus características se encuentran la configuración fácil e intuitiva a través de la herramienta *nova-manager*, destinada a la administración de los servicios telemáticos y la compatibilidad con el *hardware* que predomina en las empresas cubanas; el que, en muchos casos, sufre de la obsolescencia tecnológica. Entre los módulos incluidos en la distribución se encuentran servidores web, proxy, antivirus, correo, entre otros (UCI, 2019).

Debido a la importancia de las aplicaciones que corren en los servidores y al valor de los datos que en ellos se almacenan, se aplican medidas de seguridad física e informática en los locales donde son instaladas las computadoras; y se designa un personal encargado de la administración de los recursos. La instalación de programas es una de las tareas más comunes de la administración de servidores, se necesita conocer toda la información de los recursos de *hardware* y *software*, con el objetivo de verificar si se cumplen los requisitos mínimos que requieren los programas.

En ocasiones, la información que se extrae de los recursos del equipo es necesaria para realizar inspecciones, auditorias informáticas, medir el desempeño de los equipos o diagnosticar posibles fallas. Es muy común que los especialistas de servicios técnicos, cuando se solicita su apoyo, pidan información detallada sobre los recursos de *hardware* y *software* del servidor. Ante de la ocurrencia de violaciones a la seguridad informática, las comisiones investigadoras suelen realizar informes sobre el estado de los recursos de los servidores.

En el proceso de elaboración de la investigación se aplicaron los métodos de entrevista y de observación. El método de entrevista se realizó mediante un cuestionario (ver Anexo 1) y el método de observación mediante la guía de observación a los clientes (ver Anexo 2). El objetivo fue conocer el modo en que los clientes realizan el proceso de visualización de los detalles técnicos de los recursos de *hardware* y

software para Nova Servidores 7.0. Como resultado se detectaron las siguientes deficiencias que se deben superar para perfeccionar dicho proceso:

- El proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* se realiza de forma manual, a través de comandos de consola.
- Los administradores se deben desplazar hacia cada uno de los puestos físicos para poder realizar la visualización de los detalles técnicos de los recursos de *hardware* y *software* de cada equipo, o realizar una conexión remota por SSH (Intérprete de Órdenes Seguro, del inglés *Secure SHell*) a cada servidor.
- Los comandos de consola que se aplican entregan un exceso de datos, que dificulta su entendimiento por parte de los usuarios que no tienen una alta experticia en *hardware* y *software*.
- Se dificulta la elaboración de los informes con la información recopilada para cada equipo inspeccionado.
- Se dificulta medir el desempeño de los recursos fundamentales (Unidad Central de Procesamiento, memoria, red, procesos y servicios).

A partir de lo antes expuesto se plantea el siguiente **problema de investigación**: ¿Cómo agilizar el proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0?

Para darle solución a este problema se define como **objeto de estudio**: Proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para servidores GNU/Linux.

Se define como **campo de acción**: Proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.

El **objetivo general** de la investigación es: Desarrollar una herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.

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

1. Elaborar el marco teórico referencial de la investigación teniendo en cuenta el proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para servidores GNU/Linux.
2. Diseñar una herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.
3. Implementar la herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.
4. Evaluar la herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.

Como **preguntas científicas** se plantean:

1. ¿Cuáles son las tendencias actuales relacionadas con el proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para servidores GNU/Linux?
2. ¿Cuáles son los aspectos a tener en cuenta para realizar el diseño de una herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0?
3. ¿Qué componentes son necesarios implementar en el desarrollo de una herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0?
4. ¿Qué pruebas de *software* aplicar para la evaluación de la herramienta web para visualizar los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0?

Para el desarrollo de la investigación se emplearon los siguientes **métodos científicos**:

Métodos Teóricos:

El método **analítico-sintético** consiste en descomponer la información por partes, con el objetivo de examinarla y sintetizarla para extraer los elementos fundamentales (Mora, 2017; Rodríguez et al., 1996). Este método permitió el estudio de diferentes fuentes bibliográficas para extraer los recursos más importantes que se incluyen en las herramientas analizadas para la visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo.

El método **inductivo-deductivo** es una forma de razonamiento mediante el cual se extrapola un conocimiento de casos particulares a uno más general o a partir de conocimientos generales se permite inferir casos particulares (Mora, 2017; Rodríguez et al., 1996). Este método permitió definir los detalles técnicos que se deben incluir en la herramienta a desarrollar. Mediante el método inductivo, a partir del estudio de la herramienta más referenciada seleccionar un grupo de recursos que deben ser incluidos en la nueva herramienta para Nova Servidores 7.0. El método deductivo permitió seleccionar, a partir de las herramientas analizadas, los recursos que se visualizan con más frecuencia para ser incluidos en la nueva herramienta.

El método **histórico-lógico** estudia la esencia de los acontecimientos en los diferentes períodos de la historia, analizando las leyes generales del funcionamiento y su desarrollo (Mora, 2017; Rodríguez et al., 1996). Este método permitió verificar cómo evolucionan teóricamente las herramientas analizadas para seleccionar las técnicas, los lenguajes y la plataforma, que se van a utilizar en la herramienta a desarrollar.

Métodos Empíricos:

El método **entrevista** se utilizó para conocer el proceso actual de visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo y cuáles son las deficiencias que se deben superar en la propuesta de solución. Para su aplicación se utilizó el **cuestionario de preguntas** (ver Anexo 1).

El método **observación** se utilizó en el análisis del proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo. Para su aplicación se utilizó la **guía de observación** (ver Anexo 2).

El presente trabajo, está estructurado en 3 capítulos, distribuidos de la siguiente forma:

Capítulo 1: Fundamentación teórica del proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para servidores GNU/Linux

Se abordan los principales conceptos y definiciones de los recursos y detalles técnicos asociados al problema de investigación planteado. Se define el proceso actual de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0. Se caracterizan las herramientas existentes en el mundo que responden al problema a resolver y por qué no son factibles. Se definen las

tecnologías a utilizar en el desarrollo de la investigación y la metodología de desarrollo que va a guiar todo el proceso de construcción de la herramienta.

Capítulo 2: Análisis y diseño de la herramienta web para Nova Servidores 7.0

Se define la propuesta de solución, se extraen los requisitos funcionales y no funcionales de la herramienta a desarrollar, se modela la estructura y funcionamiento de la solución a través de los diagramas de clases de diseño y paquetes. Además, se seleccionan los patrones de diseño que serán usados para resolver la situación problemática.

Capítulo 3: Implementación y pruebas de la herramienta web para Nova Servidores 7.0

Se describen los estándares de codificación usados en el desarrollo de la herramienta. Se analizan y verifican los resultados obtenidos a partir de las pruebas aplicadas a la herramienta.

CAPÍTULO 1: Fundamentación teórica del proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para servidores GNU/Linux

El presente capítulo aborda aspectos sobre la fundamentación teórica de la investigación. Contiene un análisis sobre cómo se lleva a cabo el proceso de visualización de los detalles técnicos actualmente. Se realiza un estudio de las herramientas informáticas existentes para el proceso de visualización de los detalles técnicos del equipo. Se describe la metodología de desarrollo de *software* a utilizar. Se analizan los distintos lenguajes, herramientas y tecnologías que serán empleados en el desarrollo de la solución. A continuación, se definen algunos conceptos asociados al problema de investigación planteado.

1.1 Conceptos asociados al dominio del problema

Recursos de *hardware* y *software*

Los recursos de *hardware* son un conjunto de componentes físicos de una computadora (por ejemplo: *CPU* (Unidad Central de Procesamiento, del inglés *Central Processing Unit*), Placa base, Red, entre otros) y los recursos de *software* son los programas que se ejecutan en la *CPU* (por ejemplo: Sistema operativo, Servicios, Procesos, entre otros) (Diccionario de Informática, 2019; Fernández, 2015).

Detalles técnicos

Según la RAE, el término detalle se define como: *"pormenor, parte o fragmento de algo"*; y el término técnico, *"perteneciente o relativo a las aplicaciones de las ciencias y las artes"* (RAE, 2019b-c). Por tanto, los detalles técnicos son las partes que nombran a las particularidades de algún objeto específico relacionado con la aplicación de las ciencias y las artes. Por ejemplo: fabricante de la *CPU*.

Herramienta de visualización de detalles técnicos de los recursos de *hardware* y *software*

La visualización consiste en transformar información en imágenes que faciliten la extracción de significado (Alcalde, 2015). Por tanto, una herramienta de visualización de detalles técnicos es un programa informático que encuesta al sistema operativo, mediante el uso de los comandos y traduce los datos obtenidos a una imagen visual o gráfica; con un nivel de usabilidad sencillo acorde a las personas no expertas y con la adaptación necesaria para que se visualice en distintos dispositivos electrónicos.

1.2 Definición de los recursos y detalles técnicos que los caracterizan

1.2.1 Equipo

Un equipo informático está formado por distintos dispositivos electrónicos que permiten la ejecución de programas informáticos (Diccionario de Informática, 2019). Por ejemplo: computadora.

- **Modelo del procesador:** Se refiere al modelo de la *CPU*; por ejemplo: *Intel(R) Core(TM) i5-7500 CPU*.
- **Distribución:** Se refiere a la personalización del sistema *GNU/Linux* que se encuentra instalada en el equipo; por ejemplo: Nova Servidores, *Ubuntu* o *Debian*.
- **Arquitectura:** Según la RAE, se define como: "*Estructura lógica y física de los componentes de una computadora"* (RAE, 2019d); por ejemplo: *64 bit* o *32 bit*.
- **Nombre de usuario:** Se refiere al nombre que se le asigna al usuario que inicia sesión en el equipo; por ejemplo: betty, administrador, entre otros.
- **Nombre del equipo:** Se refiere al nombre del dispositivo conectado a la red, que se le asigna una dirección *MAC* y permite realizar el seguimiento de los datos; por ejemplo: betty-B250M-E5V.
- **Fecha / Hora:** Se refiere a la fecha y hora exacta que se obtienen los detalles técnicos de la herramienta; por ejemplo: 10/02/2020 10:12.

1.2.2 Sistema Operativo (SO)

Según la RAE, se define como: "*programa o conjunto de programas que realizan funciones básicas y permiten el desarrollo de otros programas"* (RAE, 2019e). Son responsables de gestionar, coordinar las actividades y llevar a cabo el intercambio de los recursos de *hardware*. Los detalles técnicos a visualizar del sistema operativo son:

- **Plataforma:** Se refiere al tipo de sistema operativo; por ejemplo: *Mac OS, Windows* o *Linux*.
- **Distribución:** Se refiere a la personalización del sistema *GNU/Linux* que se encuentra instalada en el equipo; por ejemplo: Nova Servidores, *Ubuntu* o *Debian*.
- **Arquitectura:** Según la RAE, se define como: "*Estructura lógica y física de los componentes de una computadora"* (RAE, 2019d); por ejemplo: *64 bit* o *32 bit*.

- **Versión:** Se refiere a un número único que identifica la edición del sistema operativo. Utiliza el formato (MAYOR.MENOR.PARCHE); por ejemplo: *Ubuntu* 16.04.5, indica que es la edición número (16.x.x) de *Ubuntu*, con algunos cambios (x.04.x) de la edición anterior (x.03.x) y con algunas correcciones menores al *software* (x.x.5).
- **Versión *Kernel*[1]:** Se refiere a un número único que identifica la edición del *kernel*. Utiliza el formato (MAYOR.MENOR.PARCHE); por ejemplo: *4.15.0-47-generic*, indica que es la edición número (4.x.x), con algunos cambios (x.15.x) de la edición anterior (x.14.x) y es compatible (x.x.0) con la edición anterior. Además, es una versión preliminar porque se le añade un guion y una serie de identificadores después del valor PARCHE (x.x.*0-47-generic*).
- **Nombre clave de la distribución:** Se refiere a un nombre clave que se le asigna a la distribución; por ejemplo: *xenial*.
- **Fecha de compilación *Kernel:*** Se refiere a la fecha y hora exacta que se realizó la compilación del *Kernel*.

1.2.3 Unidad Central de Procesamiento (*CPU*)

Componente de *hardware* que interpreta las instrucciones contenidas en los programas, procesa los datos y produce resultados (STALLINGS, 2005). Los detalles técnicos a visualizar de la *CPU* son:

- **Fabricante:** Se refiere al nombre de la compañía fabricante de la *CPU*; por ejemplo: *GenuineIntel*.
- **Modelo del procesador:** Se refiere al modelo de la *CPU*; por ejemplo: *Intel(R) Core(TM) i5-7500 CPU*.
- **Frecuencia:** Se refiere al número de operaciones por unidad de tiempo; por ejemplo: 3.40 GHz
- **Familia:** Se refiere a la micro-arquitectura base del procesador siendo 6 para todos los procesadores desde *Pentium Pro* 1995 hasta la arquitectura *Haswell* del 2014, exceptuando *Netburst* que es 15.

[1] El **núcleo o *kernel*** constituye una parte fundamental del sistema operativo. Es el encargado de gestionar recursos, a través de servicios de llamada al sistema. Además, se encarga de decidir qué programa podrá hacer uso de un dispositivo de *hardware* y durante cuánto tiempo.

- **Cantidad de núcleos:** Se refiere al número de núcleos que posee el *CPU*; por ejemplo: 4.
- ***Cache* de código L1:** Se refiere a la capacidad de la memoria *cache* que almacena información sobre las operaciones a realizar (suma, resta, multiplicación, entre otras); por ejemplo: en los procesadores *Intel* es de 32K.
- ***Cache* de datos L1:** Se refiere a la capacidad de la memoria *cache* que almacena los datos que serán procesados; por ejemplo: en los procesadores *Intel* es de 32K.
- ***Cache* L2:** Se refiere a la capacidad de la memoria *cache* que almacena instrucciones y datos que utiliza el *CPU*. Hay una *cache* L2 por cada núcleo; por ejemplo: 256K.
- ***Cache* L3:** Se refiere a la capacidad de la memoria *cache* de mayor tamaño, que utiliza un bus de datos para comunicarse con cada núcleo; por ejemplo: 6144K.
- **Uso CPU (%):** Se refiere al porciento de uso de la *CPU*.

1.2.4 Memoria de Acceso Aleatorio (*RAM*)

Componente de *hardware* que se encuentra en dispositivos electrónicos. Su función básicamente es guardar información mientras el dispositivo está encendido (memoria volátil). Además, es utilizada por el sistema operativo y las aplicaciones para cargar las instrucciones (Chistensson, 2019; Rodríguez et al, 2003). Los detalles técnicos a visualizar de la *RAM* son:

- **Memoria total:** Se refiere a la capacidad de almacenamiento de la memoria *RAM*; por ejemplo: 31 GB.
- **Memoria disponible:** Se refiere a la capacidad de almacenamiento que se encuentra libre en el instante de la consulta; por ejemplo: 22 GB.
- **Memoria en uso:** Se refiere a la capacidad de almacenamiento que se encuentra ocupado en el instante de la consulta; por ejemplo: 9 GB.

1.2.5 Memoria de intercambio (*Swap*)

Es una partición del disco que se usa para guardar las imágenes de los procesos que no se mantienen en la memoria física. Los detalles técnicos a visualizar de la *Swap* son:

- ***Swap* total:** Se refiere a la capacidad total de almacenamiento definida para realizar el intercambio entre la memoria *RAM* y una memoria externa auxiliar; por ejemplo: 31 GB.
- ***Swap* disponible:** Se refiere a la capacidad de almacenamiento que se encuentra libre en el instante de la consulta; por ejemplo: 31 GB.
- ***Swap* en uso:** Se refiere a la capacidad de almacenamiento que se encuentra ocupado en el instante de la consulta; por ejemplo: 0 GB.

1.2.6 Almacenamiento

- **Particiones del sistema:** Se refiere a la división de una unidad de disco duro de gran capacidad, en dos o más unidades virtuales.

1.2.7 Placa base

Es la placa principal encargada de unir todos los componentes que se necesitan para poder procesar la información que el equipo recibe. Los detalles técnicos a visualizar de la placa base son:

- **Nombre de la placa base:** Se refiere al nombre que se le asigna a la placa base; por ejemplo: *B250M-E5V-CF*.
- **Fabricante:** Se refiere al nombre de la compañía que elabora la placa base; por ejemplo: *Gigabyte Technology Co*.

1.2.8 Red

Según la RAE, se define como: ‘‘*Conjunto de computadoras o de equipos informáticos conectadas entre sí y que pueden intercambiar información”*. Las redes se dividen en dos categorías: LAN (red de área local) y WAN (red de área amplia) (Chistensson, 2018; RAE, 2019f). Los detalles técnicos a visualizar de la red son:

- **Fabricante de la tarjeta de red *ethernet*:** Se refiere al nombre de la compañía que elabora productos para las comunicaciones en red; por ejemplo: *Realtek Semiconductor Co*.
- **Dirección IP:** Se refiere a una IP única que se le asigna a un equipo en una misma subred (PORATTI, 2003); por ejemplo: 192.168.5.107.

- **Máscara:** Se refiere a la dirección que define la estructura IP. Se divide en dos partes, la parte de la Red, que identifica la cantidad de bit en que se puede dividir toda la subred y la parte del Host, que define la cantidad de equipos que se puede tener por subred (PORATTI, 2003); por ejemplo: 255.255.255.0.
- **Difusión:** Se refiere a la dirección con la cual se comunica con todos los miembros de una subred (PORATTI, 2003); por ejemplo: 192.168.5.255.

1.2.9 Monitor

Según la RAE, se define como: ‘‘*En un equipo informático, dispositivo provisto de pantalla que permite visualizar la información”* (RAE, 2019g). Es un periférico de entrada/salida, si el monitor tiene pantalla táctil o si el usuario interactúa con el equipo usando el teclado y el *mouse*. Los detalles técnicos a visualizar del monitor son:

- **Fabricante:** Se refiere al nombre de la compañía que elabora productos relacionados con el monitor; por ejemplo: *The X.Org Foundation*.
- **Dimensiones:** Se refiere a las medidas de la pantalla en pixeles y en milímetros; por ejemplo: 1280 x 1024 pixeles y 339 x 271 milímetros.
- **Resolución:** Se refiere a la densidad de pixeles por pulgadas que posee el monitor; por ejemplo: 96 x 96.

1.2.10 Servicios

Un sistema operativo proporciona servicios a los programas y a los usuarios, a través de interfaces. Según *Free On-Line Dictionary Of Computing* (FOLDOC): *“Los servicios que proporciona un sistema operativo y su filosofía de diseño en general ejercen una gran influencia en el estilo de programación y sobre las culturas técnicas que afloran en torno a las máquinas en las que se ejecuta el sistema”* (FOLDOC, 2019). Los detalles técnicos a visualizar de los servicios son:

- **Servicios:** Se refiere a listar todos los servicios que se encuentran instalados en el equipo.
- **Estados:** Se refiere a determinar el estado de cada servicio: Activo (servicios en ejecución) o Inactivo (servicios detenidos).

- **Detalles:** Se refiere a mostrar los datos de un determinado servicio: estado, PID (identificador) y registros (detalles del tráfico del servicio: datos, fecha, hora y nombre del equipo que ejecuta el servicio).

1.2.11 Procesos

Es un flujo de ejecución de instrucciones en un entorno propio, y con todas las atribuciones de un programa. Los procesos se clasifican como procesos permanentes (se ejecuta indefinidamente en un equipo. Además, se ocupan de la gestión automatizada de las entradas y salidas de datos; y poseen escasa interacción con los usuarios), procesos concurrentes (se ejecutan simultáneamente en un mismo sistema) o hilos (flujos de ejecución de instrucciones independientes que tiene relaciones entre sí) (Pérez López; Ribas i Xirgo et al, 2004). Los detalles técnicos a visualizar de los procesos son:

- **Procesos:** Se refiere a listar todos los procesos que se encuentran en estado de ejecución.
- **PID:** Se refiere al identificador del proceso asignado a la tarea.
- **Usuario:** Se refiere al nombre del usuario encargado de la tarea.
- **%*CPU*:** Se refiere a la porción del porciento de uso de la *CPU* que emplea la tarea.
- **%MEM:** Se refiere a la porción del porciento de uso de la memoria física que emplea la tarea.
- **Tiempo:** Se refiere al tiempo estimado que toma la tarea en ejecutarse.

1.3 Proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0

En la actualidad, el proceso de visualización de los detalles técnicos en Nova Servidores 7.0, se realiza de forma manual. Los usuarios o administradores de red acceden al equipo o servidor y obtienen los detalles técnicos de los recursos de *hardware* y *software*, mediante líneas de comandos ejecutadas por consola. A continuación, en la Tabla 1 se describen algunos comandos utilizados.

Tabla 1: Comandos de Linux para obtener los detalles técnicos de los recursos de hardware y software del equipo. (Fuente: elaboración propia)

Recursos	Comandos de *Linux*	Descripción

Varios recursos	*lshw*	Visualiza los detalles técnicos del *hardware*.
CPU	*lscpu*	Visualiza los detalles técnicos de la *CPU*, como su velocidad, número de núcleos, fabricante, entre otros.
PLACA BASE	*cat /sys/devices/virtual/dmi/id/board_name*	Visualiza el nombre de la placa base.
	cat /sys/devices/virtual/dmi/id/board_vendor	Visualiza el fabricante de la placa base.
	lspci	Visualiza los detalles técnicos del bus PCI.
	lsusb	Visualiza los detalles técnicos del bus USB.
MEMORIA	*cat /proc/meminfo*	Visualiza los detalles técnicos de la memoria.
ALMACENAMIENTO	*df -h*	Visualiza los detalles técnicos de las particiones del sistema y sus puntos de montaje, así como el espacio utilizado y disponible en cada una.
	lsscsi	Visualiza los detalles técnicos de los dispositivos SCSI, como discos duros y unidades ópticas.

RED	*ifconfig*	Visualiza los detalles técnicos sobre las interfaces de redes actuales, como la dirección ip, máscara y difusión.
SO	*import platform* *platform.system()*	Visualiza la plataforma del sistema operativo.
	lsb_release -a	Visualiza la distribución del sistema operativo.
	cat /usr/lib/os-release	Visualiza la versión del sistema operativo.
	lsb_release -idc	Visualiza el nombre clave de la distribución.
	cat /proc/sys/kernel/version	Visualiza la fecha de compilación del *kernel*.
	lsmod	Visualiza los módulos cargados en el núcleo de *Linux*. Se listan los nombres, las dependencias con otros, el tamaño y el contador de uso del módulo.
MONITOR	*xdpyinfo*	Visualiza los detalles técnicos del monitor.
SERVICIOS	*service --status-all*	Visualiza los servicios instalados.
	service [nombre] status	Visualiza el estado de un servicio.

PROCESOS	*ps -aux*	Visualiza los procesos en ejecución.
EQUIPO	*import os* *os.environ['USER']*	Visualiza el nombre del usuario.
	import platform *platform.node()*	Visualiza el nombre del equipo.
	import datetime as dt *dt.datetime.today().strftime("%d/%m/%Y %H:%M")*	Visualiza la fecha / hora del instante de la consulta.

Este proceso requiere la memorización de los comandos y consume tiempo en la preparación de los usuarios. Se requiere contar con la documentación de consulta de los comandos y sus parámetros (STACK OVERFLOW DOCUMENTATION, 2019). Cada comando tiene diferentes modos de presentar los datos, lo que dificulta la creación de los informes con los datos sintetizando de cada uno de los equipos inspeccionados. Por otra parte, el exceso de datos que entregan los comandos de consola, hace difícil el proceso de comprensión de los conceptos por parte de los usuarios que no cuentan con una alta experticia.

En casos en que se requiera la inspección de un grupo de equipos ubicados en diferentes localizaciones geográficas, aun cuando estén enlazadas en la misma red, es necesario desplazarse a los lugares físicos para introducir los comandos por consola. Esto implica gastos de recursos de transportación y tiempo de los usuarios. Otra opción es realizar una conexión remota por SSH a cada servidor, para lo cual se requiere un mayor conocimiento informático y el entorno no es gráfico.

1.4 Estudio de herramientas de visualización de detalles técnicos

Algunas herramientas de escritorio que permiten visualizar los detalles técnicos de los recursos de *hardware* y *software* del equipo son: *Speccy, HardInfo, Aida64, HWinfo, HWMonitor* y *I-Nex.* A continuación, se describen cada una de estas herramientas.

Speccy

Es una herramienta *freeware*, desarrollada por *Piriform*. *Speccy* escanea automáticamente el equipo y muestra la información en una interfaz interactiva, completa e intuitiva para el usuario. El proceso de revisión del equipo toma menos de un minuto. Entre sus ventajas se encuentran la visualización en tiempo real de la temperatura de los componentes del equipo y el rendimiento de la GPU (Unidad de Procesamiento Gráfico, del inglés *Graphics Processor Unit*). Con el objetivo de detectar fallos antes de que ocurran muestra los niveles de temperatura de la *CPU*. La herramienta tiene la capacidad de mostrar información sobre el sistema operativo, *CPU, RAM*, placa madre, gráficos, discos duro, discos ópticos, audio, periféricos y red. La información puede ser exportada en formato XML o TXT. Posee una versión gratuita y otra de pago, esta última ofrece ventajas con respecto a la versión gratuita como: actualizaciones automáticas y soporte técnico exclusivo. La principal desventaja de usar *Speccy* es su disponibilidad solo para el sistema operativo *Windows*. Además, no permite generar informes personalizados por secciones (PIRIFORM, 2017; FILEOPT, 2017). De esta herramienta se puede tomar la visualización en tiempo real de las estadísticas sobre el uso de la red y la memoria. Además, las facilidades de exportación de los detalles técnicos hacia un fichero TXT.

HardInfo

Es una herramienta *freeware*, disponible para *Microsoft Windows* y *Linux*. *HardInfo* muestra la información del *hardware* y el sistema operativo en categorías bien organizadas que facilitan su comprensión. Posee una versión gratuita y otra de pago *HardInfo PRO.* Entre sus ventajas se encuentran las habilidades para explorar información de equipos remotos. Incluye un conjunto de pruebas para conocer el rendimiento del equipo (pruebas avanzadas de la *CPU*, almacenamiento HDD/SSD y adaptador de pantalla 3D). Permite la visualización en tiempo real del comportamiento de algunos recursos (monitorear el uso de la *CPU*, de la memoria *RAM* y el tráfico de la red). Además, la información puede ser exportada en formato HTML o TXT (BERLIOS.DE, 2019; Pereira et al, 2018; Ultimate Systems, 2018). De esta herramienta se puede tomar la capacidad de explorar los recursos de *hardware* y *software* de servidores remotos.

Aida64

Es una herramienta desarrollada por la compañía húngara *FinalWire*. *Aida64* proporciona a los usuarios información precisa y abundante del monitoreo de *hardware*. La herramienta está disponible para *Microsoft Windows, Android, iOS, Windows Phone, Chrome OS y Sailfish OS.* Entre sus ventajas se

encuentran la visualización en tiempo real del comportamiento de algunos recursos (monitorear el uso de la *CPU*, de la memoria del sistema y el disco). Incluye un conjunto de pruebas para conocer el rendimiento del equipo en ciertas aplicaciones (pruebas de esfuerzo y monitoreo de sensores). La información puede ser exportada en formato HTML, CSV o XML. Además, es un *software* comercial, disponible en cuatro versiones:

- **Extrema:** destinada a usuarios domésticos. Una licencia permite su uso en tres computadoras. Incluye una versión de prueba de 30 días.
- **Ingeniero:** destinada a ser usada en algunas empresas.
- **Negocios:** destinada a los administradores de sistema de empresas de gran tamaño.
- **Auditoría de red:** desarrollada para ser empleada en entornos de red con muchos equipos.

La principal desventaja de usar *Aida64* es que es un poco complejo para el usuario entender su funcionamiento (FINALWIRE, 2015). De esta herramienta se puede tomar la capacidad de visualizar los recursos de forma gráfica y sencilla; y las facilidades de exportación de los detalles técnicos hacia un fichero HTML.

HWinfo

Es una herramienta *freeware*, desarrollada por *REALiX* que posee una versión gratuita. *HWinfo* proporciona al usuario información completa y detallada del *hardware*. Facilita leer los datos y navegar a través de la interfaz. Entre sus ventajas permite leer todos los sensores de los componentes instalados y representarlos de manera gráfica, así como la interoperabilidad con otras herramientas. Además, la información puede ser exportada en múltiples tipos de informes en formato TXT, CSV, XML y MHTML; y permite generar informes personalizados. La herramienta muestra toda la información del *hardware* del equipo en diez secciones: *CPU*, placa madre, memoria, *bus*, adaptador de video, monitor, unidades, audio, red y puertos. La principal desventaja de usar *HWinfo* es su disponibilidad solo para el sistema operativo *Windows.* Además, se enfoca únicamente en el *hardware* y en mostrar la dirección *MAC* de la red (MALÍK, 2018). De esta herramienta se puede tomar la capacidad de monitorear en tiempo real el sistema y predicción de fallos. Además, permite generar informes personalizados por secciones.

HWMonitor

Es una herramienta *freeware*, desarrollada por la empresa *CPUID Software Development Kits*. *HWMonitor* facilita al usuario leer los datos y navegar a través de la interfaz. Permite al sistema la visualización de la temperatura de los componentes del equipo y la frecuencia de vueltas de los ventiladores. La herramienta muestra toda la información de los principales sensores del equipo, obteniendo datos relevantes sobre: voltajes, temperaturas y velocidad de los ventiladores. Además, el programa requiere derechos administrativos para su funcionamiento. Posee una versión gratuita y otra de pago *CPUID HWMonitor PRO*. La principal desventaja de usar *HWMonitor* es su disponibilidad solo para el sistema operativo *Windows.* Muestra información básica del equipo y no permite generar informes, solo en la versión *CPUID HWMonitor PRO* (CPUID, 2019). De esta herramienta se puede tomar la actualización de los datos en tiempo real.

I-Nex

Es una herramienta *freeware*, que posee una versión gratuita. *I-Nex* muestra información detallada y organizada de los siguientes componentes: *CPU,* GPU, placa madre, sonido, discos duro, *RAM*, red y *USB*. Además de mostrar información del sistema como el nombre del *host*, distribución de *Linux* que se utiliza y versión del *Kernel*, entre otras. Permite generar informes personalizados y exportarlo en formato *TXT*. También ofrece la opción de tomar capturas de pantallas directamente desde la herramienta (BARWALDT, 2017). De esta herramienta se puede tomar la capacidad de generar informes personalizados por secciones.

Existe un grupo de herramientas web para la administración de servidores que entre sus funcionalidades cuentan con módulos que permiten visualizar los detalles técnicos de los recursos de *hardware* y *software* del equipo. A continuación, se describen algunas de estas herramientas.

Webmin

Es una herramienta de configuración de sistema *GNU/Linux*, de código abierto, desarrollado por *Red Hat Enterprise Linux* (*RHEL*). *Webmin* permite configurar cuentas de usuario, de servicios como *Apache*, *DNS*, *PHP* o *MySQL;* y compartir archivos. Entre sus ventajas se encuentran la posibilidad de controlar varios equipos a través de una interfaz simple, o iniciar sesión en otros servidores *webmin* de la misma subred o red de área local. Además, la interfaz gráfica de *Webmin* es accesible a través de la web; y se puede entrar al servidor para realizar gestiones desde cualquier parte, usando los dispositivos móviles. La principal desventaja de la herramienta es que está orientada a usuarios especializados en la

administración de servidores, por tanto, su alcance supera los requerimientos del tipo de herramienta de visualización que debe brindar solución al problema planteado (Webmin, 2016). De esta herramienta se puede tomar la filosofía de permitir administrar un servidor sin tener que conocer los comandos básicos del sistema.

cPanel (Monitor de Recursos)

Es la herramienta de administración de sitios web más utilizada en el mundo. Tiene un módulo Monitor de Recursos, libre de costo, con la condición de que para adquirirlo se requiere una licencia de pago de la herramienta. El módulo permite monitorear los recursos que son utilizados por los sitios web hospedados en el servidor. Entre las características del módulo se encuentra la posibilidad de conocer el consumo de *CPU* y memoria en tiempo real, los errores de programación que se registraron en los sitios web y las consultas más lentas si se utiliza base de datos *MySQL*; de manera que se puede resolver problemas de forma rápida. La principal desventaja de *cPanel* es que no soporta distribuciones basadas en *Debian* y *Ubuntu* por definición; por lo tanto, no es posible utilizarlo en Nova Servidores que es una distribución basada en *Ubuntu* (Cpanel, 2020). De esta herramienta se puede tomar como patrón el diseño de la interfaz usando componentes gráficos.

CentOS Web Panel

CentOS (del inglés, *Community Enterprise Operating System*) es un panel de control *freeware*, desarrollado por *Red Hat Enterprise Linux* (*RHEL*) que posee versión gratuita. *CentOS Web Panel* facilita la gestión de los sitios web hospedados en el servidor. Permite ver el estado de los servicios, la información del *hardware* y el *software*, detalles de los discos y la configuración de la red. Además, posibilita la gestión de usuarios, de *DNS*, de servidor web, de la seguridad, de servidor de base de datos y de correo electrónico. Permite migrar la configuración de *cPanel*, restaurando fácilmente las bases de datos y otros archivos que se gestionaban con *cPanel*. La principal desventaja de *CentOS Web Panel* es su amplio alcance y que no soporta distribuciones basadas en *Ubuntu.* Además, solo puede ser instalado en distribuciones Linux con *CentOS* (CentOS Project, 2020). De este panel de control se puede tomar la experiencia relacionada con el monitoreo de los recursos.

A partir de la caracterización de las herramientas antes mencionadas se llega a la conclusión que no existe una herramienta web genérica, para cualquier tipo de servidor, que permita el proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo. El estudio

permitió definir un conjunto de características que fueron utilizadas como referencia para el diseño de la propuesta de solución. Entre ellas se encuentran el uso de cuadro de mando[2] para visualizar instrumentos gráficos que permitan un rápido entendimiento de los datos; la visualización de los recursos de forma gráfica y sencilla sin tener que conocer los comandos básicos del sistema; y el acceso a través de la web, lo que permite la visualización desde dispositivos móviles. A continuación, se describe la metodología de desarrollo de *software* que se empleará en la propuesta de solución.

1.5 Metodología de desarrollo

Una metodología de desarrollo de *software* es una colección de procedimientos, técnicas y herramientas que permiten a los desarrolladores de *software* implementar sistemas informáticos. En el epígrafe 1.5.1 se define la metodología de desarrollo de *software* empleada en la elaboración de la propuesta de solución.

1.5.1 Variación de *AUP* para la UCI

La metodología seleccionada para el desarrollo de la propuesta de solución es Variación de *AUP* (del inglés, *Agile Unified Process*) para la UCI, debido a que es la establecida por la universidad y se adapta al ciclo de vida definido para la actividad productiva. La misma cuenta con tres fases:

- **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. Se define el alcance del proyecto, estimaciones de tiempo, esfuerzo, costo y se decide si se ejecuta o no el proyecto.
- **Ejecución:** durante el desarrollo se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

El presente trabajo de diploma se desarrolla en la fase de ejecución; el cual transitará por las siguientes disciplinas propuestas en la metodología: requisitos, análisis y diseño, implementación, pruebas internas y pruebas de aceptación.

[2] Un **cuadro de mando** (del inglés, *Dashboard*) es una interfaz de usuario que utiliza gráficos para organizar la información en un modo simplificado y fácil de visualizar (KLIPFOLIO INC, 2020).

De la metodología seleccionada se escogió el escenario 4: Proyectos que no modelen negocio y solo pueden modelar el sistema con Historia de Usuario (HU). El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y poder implementarlos, probarlos y validarlos. Además, se recomienda en proyectos no muy extensos (Sánchez, 2015).

1.6 Herramientas, lenguajes y tecnologías

Para el desarrollo de la solución propuesta se realizó un estudio en cuanto a los lenguajes de programación, las tecnologías y las herramientas a utilizar, teniendo en cuenta las ventajas de cada una y que cumplan con las condiciones del problema planteado.

1.6.1 Modelado

Lenguaje de Modelado Unificado (*UML*) (versión 5.0)

Es un lenguaje para visualizar, especificar, construir y documentar un sistema. Está respaldado por el *OMG* (del inglés, *Object Management Group*). Se puede aplicar en el desarrollo de *software* con gran variedad de formas para dar soporte a una metodología de desarrollo de *software*. Un modelo UML está compuesto por 3 clases de bloques de construcción: los elementos (representaciones abstractas de cosas reales o ficticias), las relaciones y los diagramas; los cuales muestran diferentes aspectos de las entidades representadas (OMG, 2019; LUCID SOFTWARE INC., 2019).

Visual Paradigm (versión 8.0)

Se seleccionó como herramienta *CASE*[3], multiplataforma. Utiliza modelado UML para la representación de las etapas por las que transita un producto de *software*. Además, propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Visual Paradigm, 2019).

Pencil (versión 3.0.4)

Se seleccionó como herramienta gratuita y de código abierto para el diseño gráfico de los prototipos de interfaz de usuario. Además, permite diseñarlos de forma fácil y exportarlo en formatos *PNG, HTML* o

[3] Ingeniería de *Software* Asistida por Computadora (del inglés, *Computer Aided Software Engineering*). Aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el costo de las mismas en términos de tiempo y de dinero.

PDF. *Pencil* proporciona varias colecciones de formas para crear diferentes tipos de interfaces, desde páginas web hasta aplicaciones móviles (Pencil Project, 2019).

1.6.2 Lenguajes

Python (versión 3.5)

Lenguaje de programación creado por *Guido van Rossum*. Su implementación está bajo la licencia de código abierto *Python Software Foundation Licence*. A continuación, se presentan algunas de sus características:

- **Lenguaje interpretado:** No requiere de un compilador para ser ejecutado sino de un intérprete. El intérprete ejecuta el programa directamente, sin necesidad de generar previamente un ejecutable.
- **Lenguaje de alto nivel:** Consiste en una estructura sintáctica y semántica legible, que le permite una mejor comprensión al programador.
- **Multiplataforma:** Significa que puede ser interpretado en diversos sistemas operativos; por ejemplo: *Linux, Windows, Mac OS, Solaris*, entre otros.
- **Multiparadigma:** Permite adoptar varios estilos: programación orientada a objetos, programación imperativa y programación funcional.
- **Tipado dinámico:** Las variables no requieren ser definidas asignando su tipo de datos, se auto-asigna en tiempo de ejecución, según el valor declarado.

Se seleccionó *Python* como lenguaje de programación porque permite la programación orientada a objetos. *Python* en comparación con otros lenguajes de programación contiene una sintaxis sencilla, clara y limpia. Además, *Python* admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización de código (BAHIT, 2012).

Lenguaje marcado de hipertexto (*HTML*) (versión 5.0)

Es un lenguaje que se utiliza fundamentalmente en el desarrollo de páginas web. *HTML* (del inglés, *HiperText Markup Language*) se utiliza para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes. Los archivos desarrollados en *HTML* usan la extensión *.htm* o *.html*. Además, funciona por medio de etiquetas que describen la apariencia o función del texto enmarcado (MANZ, 2017).

Hojas de estilo en cascada (*CSS*) (versión 3.0)

Es un mecanismo simple que permite controlar la apariencia de una página web. *CSS* (del inglés, *Cascading Style Sheets*) describe cómo se muestra un documento en la pantalla y cómo se va a imprimir. Esta forma de descripción de estilos les permite a los desarrolladores web controlar el estilo y formato de múltiples páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en el *CSS* afectará a todas las páginas vinculadas a ese *CSS* en las que aparezca ese elemento (W3C, 2017).

JavaScript (versión correspondiente a la edición ECMA2019 de dicho estándar)

Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como aparición y desaparición de texto, animaciones, acciones que se activan al pulsar botones, entre otras (ZAKAS, 2005).

Jinja2 (versión 2.11.1)

Es un lenguaje de plantillas (del inglés, *templates*) basado en *Python*. Incluye la herencia e inclusión de plantillas y permite definir e importar macros dentro de las plantillas (RONACHER, 2017).

1.6.3 Entorno Integrado de desarrollo (*IDE*)

Pycharm (versión 2018.1.4)

Es el *IDE* de *Python*, desarrollado por la compañía *JetBrains*. Algunas de sus características son la compatibilidad con varios sistemas operativos y que se emplea mayormente para la programación en el lenguaje *Python*. *PyCharm* incluye varias funcionalidades como el análisis de código, la depuración gráfica e integración con sistemas de control de versión (JETBRAINS, 2019).

1.6.4 *Framework* de desarrollo

Flask (versión 1.1.1)

Es un micro - *framework* de desarrollo. Permite un control al detalle de la herramienta y una selección precisa de los componentes que se utilizan. Además, a través de extensiones puede incorporar otras funcionalidades que necesita el usuario; por ejemplo: integración de base de datos, interfaces de administración, herramientas de migración, entre otras. *Flask* es una plataforma web minimalista basada en *Jinja2* y *Werkzeug*. En la arquitectura Modelo – Vista – Controlador (MVC), *Flask* cubre el controlador y la vista; y no restringe al usuario la posibilidad de crear su propio modelo (DUPLAIN, 2013; GRINBERG, 2014; STACK OVERFLOW DOCUMENTATION, 2016).

Bootstrap (versión 4.1.3)

Es un *framework responsive*[4], multiplataforma y de código abierto, desarrollado por el equipo de *Twitter*. Se emplea para el diseño de sitios y aplicaciones web. Utiliza una malla o rejilla (del inglés, *grid*) de 12 columnas para realizar un diseño adaptable al tamaño del dispositivo en que se visualice. Además, incorpora otros complementos como formularios, botones, barras de navegación, entre otros. El sistema de rejilla se utiliza dentro de uno de los dos elementos contenedores que provee *Bootstrap*: *container* (centrado y con ancho fijo) o *container – fluid* (puede ocupar todo el ancho de la pantalla disponible) (BOOTSTRAP TEAM, 2019; GREY Li, 2019).

1.6.5 Biblioteca

jQuery (versión 3.3.1)

Es una biblioteca de *JavaScript*, multiplataforma, que agrupa las tareas comunes en métodos que se pueden llamar con una sola línea de código. Su propósito es hacer que sea más fácil el uso de *JavaScript* en el sitio web (JQUERY, 2020; THE JQUERY FOUNDATION, 2020).

1.6.6 Servidor de aplicaciones

Gunicorn (versión 20.0.4)

Es un servidor HTTP[5] para *Python*; compatible con varios *frameworks*, soporta WSGI[6] (del inglés, *Web Server Gateway Interface*) y *Flask*. El servidor *Gunicorn* es multiplataforma, gratuito, de código abierto y consume pocos recursos en ejecución (GUNICORN DEVELOPERS, 2019).

1.6.7 Herramienta de rendimiento

Apache Jmeter (versión 3.2)

Es una aplicación de código abierto diseñada para medir el rendimiento de las aplicaciones a partir de comportamientos funcionales. Puede ser utilizado para probar recursos estáticos y dinámicos, servicios

4 ***Framework responsive*****:** permite crear webs adaptables a diferentes tamaños de pantalla. Este tipo de diseño se basa en crear un único código, utilizar reglas y estilos *CSS* para adaptar los contenidos a los diferentes tamaños de pantalla.

5 **HTTP:** Protocolo de Transferencia de Hipertexto o del inglés, *Hipertext Transfer Protocol*.

6 **WSGI:** Es una especificación que describe cómo se comunica un **servidor web** con **una aplicación web**, y cómo se pueden llegar a encadenar diferentes aplicaciones web para procesar una solicitud / petición (o *request*).

web, simular una carga pesada en un servidor, grupo de servidores o para hacer un análisis gráfico del rendimiento. Es un *software* de escritorio escrito en *Java* y disponible para todas las plataformas (APACHE JMETER, 2017).

1.7 Conclusiones parciales

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

- El análisis del marco teórico referencial de la investigación sobre el proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0; y el estudio de los principales conceptos asociados al problema planteado, permitieron establecer las bases para el desarrollo de la investigación y conocer los detalles técnicos de los recursos del objeto de estudio.
- A partir de la caracterización de algunas herramientas para la visualización de detalles técnicos, se concluyó que no existe una herramienta web que cumpla con requerimientos del problema de investigación.
- Se demostró la necesidad de desarrollar una herramienta web que aproveche las mejores características que ofrecen las herramientas estudiadas, pues estas no resuelven la totalidad de las insuficiencias presentes en el proceso de visualización de los detalles técnicos para Nova Servidores 7.0. Algunas de las características son el uso de cuadro de mando, la gestión de los recursos de forma gráfica y sencilla; y el acceso a través de la web.
- En el desarrollo de la propuesta de solución se definió el empleo de herramientas informáticas libres para garantizar la soberanía tecnológica; y la utilización de la metodología de desarrollo de *software* Variación de *AUP* para la UCI.

CAPÍTULO 2: Análisis y diseño de la herramienta web para Nova Servidores 7.0

En el presente capítulo, se desarrolla el análisis y diseño de la propuesta de solución haciendo uso de los artefactos que propone la metodología Variación de *AUP* para la UCI, en el escenario 4. A continuación, se describe la propuesta de solución para darle cumplimiento al problema de investigación planteado.

2.1 Propuesta de solución

Se implementará una herramienta web con el nombre de “Dtecnico”, que le permitirá al usuario visualizar los detalles técnicos de los recursos de *hardware* y *software* de su equipo. Dtecnico mostrará información acerca de los siguientes componentes: Sistema Operativo, *CPU*, *RAM*, *Swap*, Almacenamiento, Placa Base, Red, Monitor, Servicios y Procesos. La información recogida por Dtecnico podrá ser exportada en formato PDF o HTML. Dicha herramienta permitirá perfeccionar el proceso de visualización al dar solución a las dificultades planteadas en el capítulo anterior; y es el resultado de la solución de diseño dada al problema de investigación. En el epígrafe 2.2 se muestra el modelo conceptual realizado para comprender el contexto del negocio.

2.2 Modelo conceptual

Un modelo conceptual contiene la descripción de cómo se relacionan los conceptos que intervienen en el negocio (LARMAN, 2003). A continuación, se representa el modelo conceptual que corresponde a la descripción del contexto del negocio de la propuesta de solución.

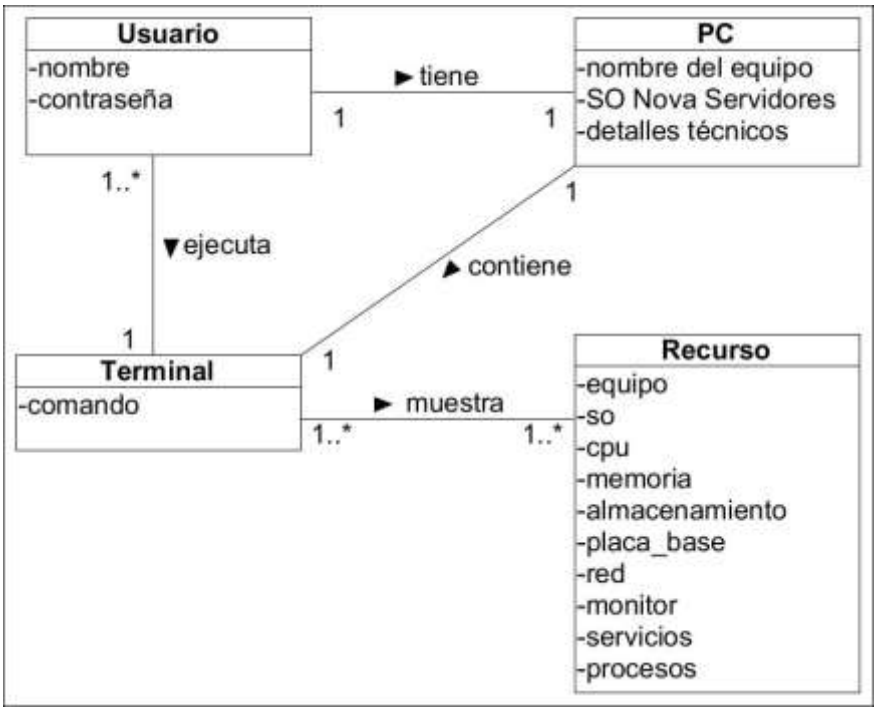


Figura 1: Modelo conceptual de la propuesta de solución.

(Fuente: elaboración propia)

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

PC: computadora que contiene la distribución cubana Nova Servidores 7.0 y cuenta con una terminal; con la computadora se puede visualizar los detalles técnicos de los recursos de *hardware* y *software* presente en su sistema.

Usuario: cliente que accede a la terminal para visualizar los detalles técnicos de los recursos de *hardware* y *software* de la computadora.

Terminal: consola de GNU/Linux en la que se escriben comandos para que se ejecuten.

Recurso: componente de la computadora que contiene detalles técnicos.

Con el objetivo de conocer las características y restricciones de los atributos representados en los conceptos de la Figura 1, se utilizó el diccionario de datos.

Diccionario de datos

Un diccionario de datos contiene información de los conceptos presentes en el contexto de un negocio determinado. En la Tabla 2 se presenta el diccionario de datos del concepto: “Recurso *CPU*”.

Tabla 2: Diccionario de datos del concepto Recurso CPU.

(Fuente: elaboración propia)

Descripción	**Recurso *CPU*:** Es uno de los componentes del equipo que contiene un conjunto de detalles técnicos.					
Atributos						
					Restricciones	
Nombre	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**	**Clases válidas**	**Clases no válidas**
Fabricante	Nombre de la compañía fabricante de la *CPU*	Cadena de caracteres	No	Sí	Cadena de caracteres	No aplica
Modelo del procesador	Identifica el modelo de la *CPU*	Alfanumérico	No	No	Caracteres alfanuméricos.	No aplica
Frecuencia	Número de operaciones por unidad de tiempo	Alfanumérico	No	No	No aplica	No puede tomar valores negativos
Familia	Identifica la micro-arquitectura base del procesador	Numérico	No	No	No aplica	No puede tomar valores negativos
Cantidad de núcleos	Número de núcleos que posee el *CPU*	Numérico	No	No	No aplica	No puede tomar valores negativos
Uso CPU (%)	Porciento del uso de la *CPU*	Numérico	No	No	No aplica	No puede tomar valores

						negativos
Capacidad de la *cache* de código L1	Capacidad de la memoria *cache* que almacena información sobre las operaciones a realizar (suma, resta, multiplicación, entre otras)	Alfanumérico	No	No	No aplica	No puede tomar valores negativos
Capacidad de la *cache* de datos L1	Capacidad de la memoria *cache* que almacena los datos que serán procesados	Alfanumérico	No	No	No aplica	No puede tomar valores negativos
Capacidad de la *cache* L2	Capacidad de la memoria *cache* que almacena instrucciones y datos que utiliza el *CPU*	Alfanumérico	No	No	No aplica	No puede tomar valores negativos
Capacidad de la *cache* L3	Capacidad de la memoria *cache* de mayor tamaño	Alfanumérico	No	No	No aplica	No puede tomar valores negativos

En el epígrafe 2.3 Requisitos se definen los requisitos a implementar, con el objetivo de conocer y mejorar las necesidades del cliente.

2.3 Requisitos

Los requisitos de un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. El objetivo principal es identificar y documentar todo, para comunicárselo de forma clara al cliente y a los miembros del equipo de desarrollo (Sánchez, 2015). A continuación, se presenta las actividades desarrolladas en esta disciplina, así como los productos de trabajo elaborados en el desarrollo de la investigación.

2.3.1 Fuentes para la obtención de requisitos

Las fuentes de obtención de requisitos utilizadas fueron:

- Modelo conceptual (ver Figura 1)
- Análisis de las herramientas existentes (ver epígrafe 1.4 Estudio de herramientas de visualización de detalles técnicos)
- Especialistas de CESOL

2.3.2 Técnicas de identificación de requisitos

En la obtención de los requisitos de la herramienta para la visualización de los detalles técnicos se emplearon como técnicas de identificación de requisitos la entrevista, la observación y el desarrollo de prototipos. Su definición y forma de aplicación se describen a continuación.

Entrevista

Es una técnica muy utilizada para obtener información cualitativa, como opiniones o descripciones subjetivas de actividades. Algunos aspectos importantes a tener en cuenta al realizar la entrevista es la preparación, ya que deben quedar bien definidos cuáles son los contenidos que se desean obtener. Además, se recomienda utilizar preguntas abiertas, donde los entrevistados puedan elaborar y dar detalles, más allá de simplemente responder “si” o “no” (Guerra, 2017). La técnica se aplicó a través de un **cuestionario de preguntas** realizado al cliente y a los especialistas del centro CESOL (ver Anexo 1) para obtener la información necesaria referente a los requerimientos de la herramienta a desarrollar.

Observación

Por medio de esta técnica se obtiene información sobre la forma en que se efectúan las actividades. La técnica se aplicó a través de una **guía de observación**, la cual permite observar la manera en que se llevan a cabo los procesos y verificar que realmente se sigan todos los pasos especificados. Los observadores experimentados saben qué buscar y cómo evaluar la relevancia de lo que observan (Guerra, 2017). Después de aplicada la entrevista y con la obtención de requisitos, se observó el proceso de visualización de los detalles técnicos llevado a cabo por los especialistas de CESOL (ver Anexo 2) lo que permitió refinar e incorporar nuevos requisitos, presentes en la Tabla 3.

Desarrollo de prototipos

Los prototipos suelen consistir en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación. Fomentan el desarrollo de ideas y permiten a los usuarios mejorar las especificaciones de requerimientos (Guerra, 2017). La técnica se aplicó a través del **prototipado de interfaz de usuario** para conocer cuáles son las características que debían tener las interfaces de la herramienta a desarrollar.

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

2.3.3 Especificación de requisitos de *software*

En la especificación de requisitos deben recogerse tanto las necesidades de clientes y usuarios como los requisitos que debe cumplir el *software* a desarrollar para satisfacer todas las necesidades. A continuación, se describirán los requisitos funcionales y no funcionales que se tuvieron en cuenta para elaborar la solución del problema planteado.

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de qué forma se debe comportar en situaciones particulares (SOMMERVILLE, 2005). En la Tabla 3 se muestran los requisitos funcionales que se identificaron en el desarrollo de la herramienta para la visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo.

Tabla 3: Listado de requisitos funcionales de la propuesta de solución.

(Fuente: elaboración propia)

Número	Requisito funcional	Descripción	Prioridad
RF.1	Mostrar detalles técnicos generales del equipo.	La herramienta permite al usuario visualizar los detalles técnicos generales del recurso EQUIPO.	Alta
RF.2	Mostrar detalles técnicos de la *CPU*.	La herramienta permite al usuario visualizar los detalles técnicos del recurso CPU.	Alta

RF.3	Mostrar detalles técnicos de la placa base.	La herramienta permite al usuario visualizar los detalles técnicos del recurso PLACA BASE.	Alta
RF.4	Mostrar detalles técnicos de la memoria.	La herramienta permite al usuario visualizar los detalles técnicos del recurso MEMORIA.	Alta
RF.5	Mostrar detalles técnicos del almacenamiento.	La herramienta permite al usuario visualizar los detalles técnicos del recurso ALMACENAMIENTO.	Alta
RF.6	Mostrar detalles técnicos de la red.	La herramienta permite al usuario visualizar los detalles técnicos del recurso RED.	Alta
RF.7	Mostrar detalles técnicos del sistema operativo.	La herramienta permite al usuario visualizar los detalles técnicos del recurso SO.	Alta
RF.8	Mostrar detalles técnicos del monitor.	La herramienta permite al usuario visualizar los detalles técnicos del recurso MONITOR.	Alta
RF.9	Listar servicios.	La herramienta permite al usuario visualizar los servicios instalados en el equipo, mostrando el nombre y el estado.	Alta
RF.10	Mostrar detalles técnicos de un servicio.	La herramienta permite al usuario visualizar los detalles técnicos de un determinado servicio.	Alta
RF.11	Listar procesos.	La herramienta permite al usuario visualizar los procesos en ejecución.	Alta
RF.12	Mostrar detalles técnicos de los	La herramienta permite al usuario visualizar	Alta

	procesos.	los detalles técnicos del recurso PROCESO.	
RF.13	Mostrar detalles de la herramienta.	Permite al usuario visualizar los detalles de la herramienta, como el nombre y versión de la herramienta, objetivo de su creación, a quién va destinada y el nombre del desarrollador.	Bajo
RF.14	Generar reporte sobre los detalles técnicos del equipo.	La herramienta permite generar reportes en formatos PDF o HTML, que contengan los detalles técnicos de cada recurso. Además, brinda la opción de seleccionar que recursos desea exportar.	Alta
RF.15	Imprimir reporte sobre los detalles técnicos del equipo.	La herramienta permite imprimir reporte sobre los detalles técnicos del equipo.	Alta
RF.16	Permitir autenticar usuario en la herramienta.	La herramienta permite al usuario autenticarse para acceder a los detalles técnicos de los recursos de *hardware* y *software* del equipo.	Alta
RF.17	Mostrar cuadro de mando con algunos detalles técnicos del equipo.	La herramienta permite al usuario visualizar, de forma dinámica, algunos detalles técnicos del equipo.	Alta

Requisitos no funcionales

Los requisitos no funcionales son cualidades y restricciones de los servicios o funciones ofrecidas por el sistema. La definición de los requisitos no funcionales de la propuesta de solución se realizó mediante el estándar de calidad ISO/IEC 25023:2017, propuesto en el producto de trabajo “Especificación de requisitos de *software*” del expediente de proyecto utilizado en la actividad productiva de la universidad

(SOMMERVILLE, 2005; NC, 2017). En la Tabla 4 se muestran los requisitos no funcionales que se identificaron en el desarrollo de la herramienta.

Tabla 4: Listado de requisitos no funcionales de la propuesta de solución.

(Fuente: elaboración propia)

Número	Requisito	Descripción
RNF.1	Requisito de funcionalidad	La herramienta web de visualización de detalles técnicos debe realizar todas las operaciones indicadas por el usuario en cada momento: mostrar los detalles técnicos de cada recurso, generar e imprimir reportes y permitir a los usuarios autenticarse.
RNF.2	Requisito de seguridad	A la herramienta de visualización de detalles técnicos solo tendrán acceso los usuarios autorizados.
RNF.3	Requisito de usabilidad	La herramienta web debe poseer un diseño "*Responsive*" para garantizar la adecuada visualización de los detalles técnicos en múltiples dispositivos: computadoras, tabletas y móviles.
RNF.4	Requisito de interoperabilidad	La herramienta web de visualización de los detalles debe ser compatible con Nova Servidores 7.0.
RNF.5	Requisito de confiabilidad	La herramienta web de visualización de los detalles técnicos debe ser capaz de recuperarse ante fallos que puedan ocurrir en el *software*.

Para describir las características de los requisitos funcionales definidos en el epígrafe 2.2.3, así como sus restricciones, se utilizó el producto de trabajo Historia de Usuario. En el epígrafe 2.2.4 se presenta un ejemplo de este artefacto de la ingeniería de *software*.

2.3.4 Historias de Usuario

Las Historias de Usuario especifican las tareas que debe realizar el sistema, y se utilizan para estimar sus tiempos de desarrollo. A continuación, en la Tabla 5 se define una de las historias de usuario de la propuesta de solución, las demás se encuentran descritas en el Anexo 3.

Tabla 5: RF.2 Mostrar detalles técnicos de la CPU.

(Fuente: elaboración propia)

Número: HU-02	**Requisito:** Mostrar detalles técnicos de la *CPU.*
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción "*CPU*", en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos de la *CPU*.

La información se obtiene con la consulta de los métodos ubicados en la clase *Cpu()*:

- *_kernel_amount()*, se visualiza la cantidad de núcleos.
- *_manufacturer()*, se visualiza el nombre de la compañía fabricante de la *CPU*.
- *_type()*, se visualiza el modelo del procesador.
- *_frequency()*, se visualiza la frecuencia.
- *_cpu_family()*, se visualiza la familia.
- *_cpu_percent(),* se visualiza el porciento del uso de la *CPU*.
- *_code_cache_l1()*, se visualiza la cache de código L1.
- *_date_cache_l1()*, se visualiza la cache de datos L1.
- *_cache_l2()*, se visualiza la cache L2.
- *_cache_l3()*, se visualiza la cache L3.

Observaciones: La herramienta debe mostrar un gráfico dinámico (en tiempo real) con el promedio del

porciento de uso de todos los *CPU*; y además un gráfico que refleje el porciento de uso de la *CPU* para cada núcleo.

Prototipo de interfaz de usuario:

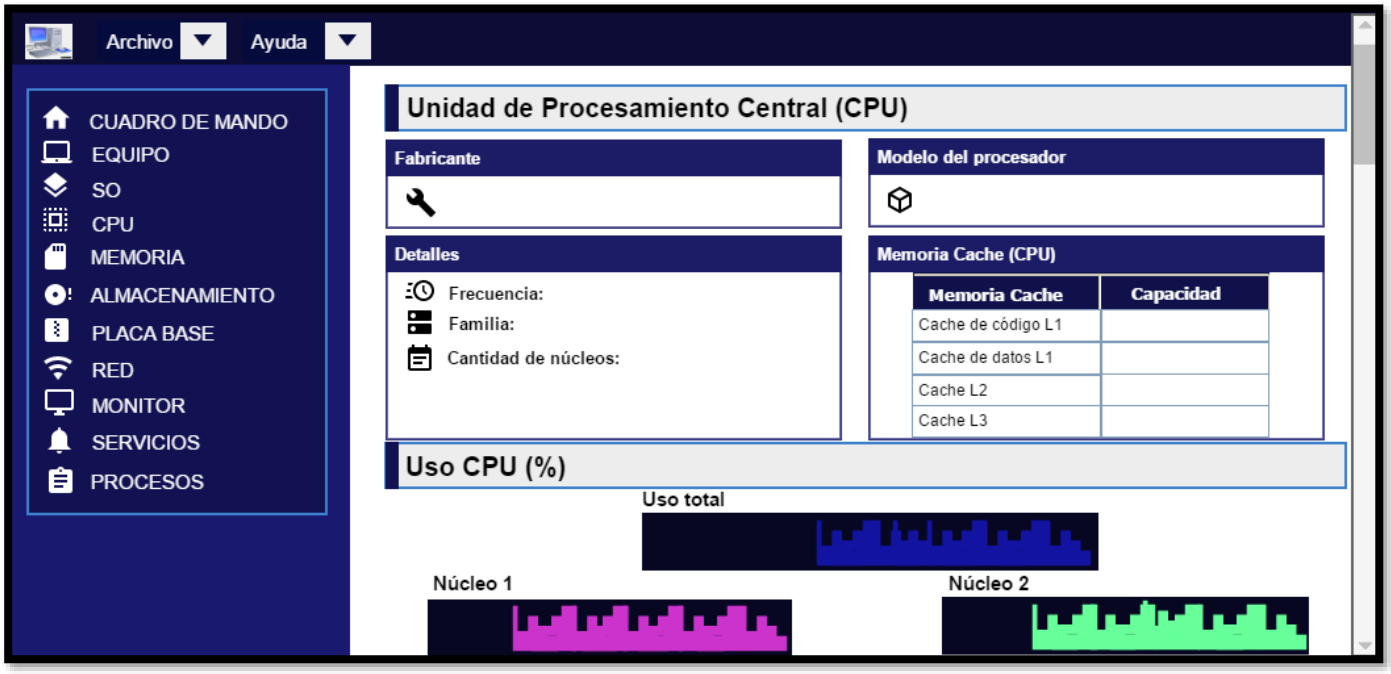


Figura 2: Prototipo de interfaz de usuario correspondiente al RF.2 Mostrar detalles técnicos de la CPU.

(Fuente: elaboración propia)

2.3.5 Validación de requisitos de *software*

El objetivo de la validación de requisitos es asegurar que todos los requisitos del *software* que se han establecido se correspondan con las necesidades de los clientes y usuarios. Además, que se hayan corregido las inconsistencias detectadas, omisiones y errores. Para validar los requisitos obtenidos se aplicó la técnica de prototipado de interfaz de usuario, la cual se describe a continuación.

Prototipado de interfaz de usuario

Es una técnica que permite al usuario entender fácilmente la propuesta de los ingenieros para resolver sus problemas de negocio. El objetivo principal de la creación de un prototipo de interfaz de usuario es poder

probar el diseño de la interfaz, incluyendo la capacidad de utilización antes de que empiece el desarrollo real (IBM CORP, 2018; PRESSMAN, 2010).

La identificación de las necesidades del cliente y las restricciones presentes en el contexto del negocio informatizado permitieron la realización del análisis y diseño de la propuesta de solución. En el epígrafe 2.4 se muestran las actividades desarrolladas en esta disciplina.

2.4 Modelo de Diseño

El modelo de diseño es una abstracción de la implementación y el código fuente del *software*. Su objetivo es representar mediante diagramas los aspectos relacionados con los requisitos no funcionales, utilizando las restricciones del lenguaje de programación seleccionado (SOMMERVILLE, 2005). A continuación, se describe el diagrama de clases de diseño con estereotipos web correspondiente a la agrupación de requisitos empleados en la propuesta de solución.

2.4.1 Diagramas de clases del diseño con estereotipos web

Los diagramas de clases del diseño tradicionales (DCD) describen gráficamente las especificaciones de las clases de *software* e interfaces en una aplicación (LARMAN, 2003). Las características particulares de los diagramas de clases web con respecto a los diagramas de clases tradicionales son el uso de estereotipos UML especiales, los cuales permiten modelar de forma certera la arquitectura y funcionamiento de aplicaciones de este tipo. A continuación, en la Figura 3 se muestra el diagrama de clases de diseño con estereotipos web correspondiente a la agrupación de requisitos.

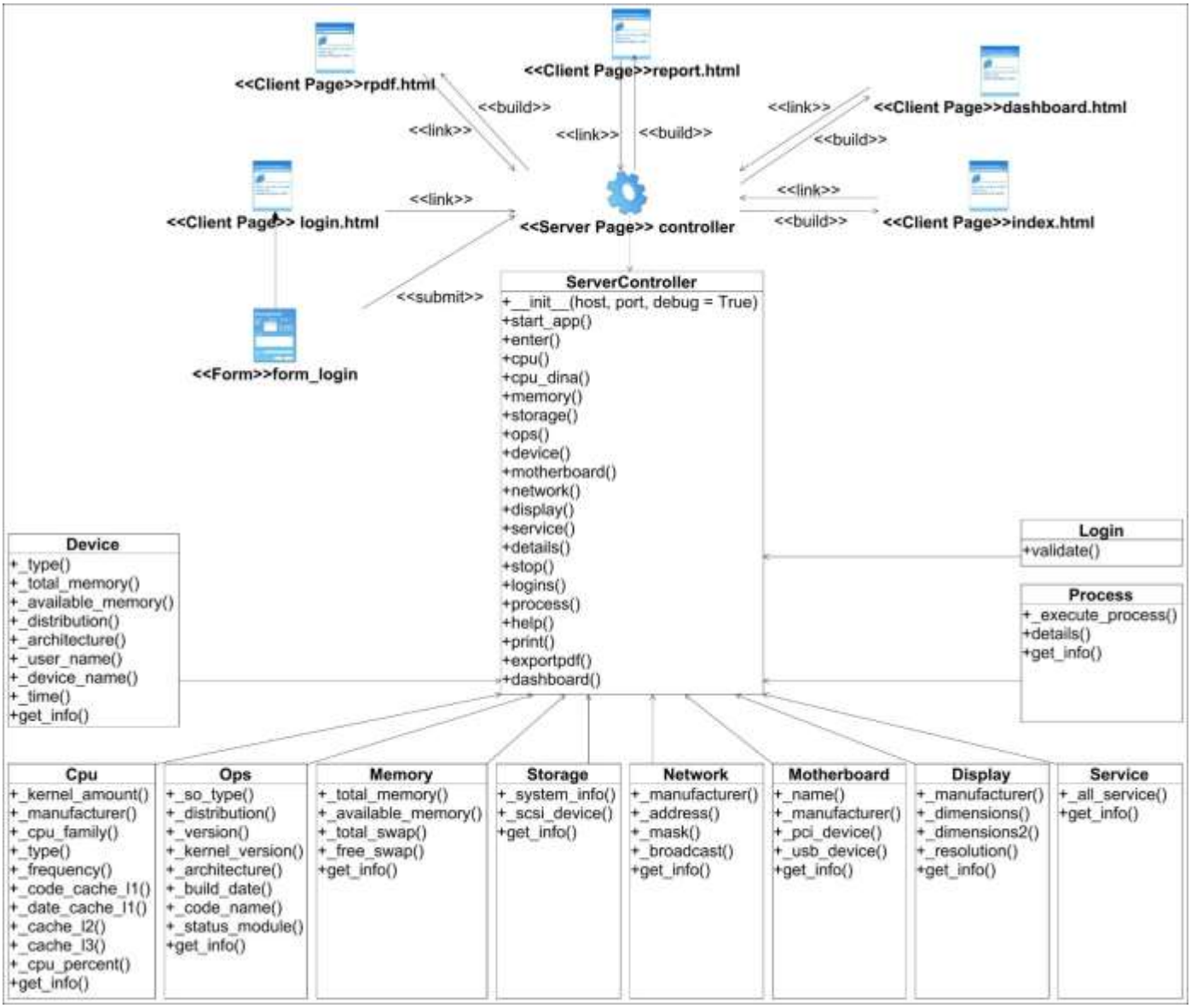


Figura 3: Diagrama de clases de diseño con estereotipos web correspondiente a la agrupación de los requisitos.

(Fuente: elaboración propia)

El diagrama de clases anterior está formado por las clases:

- **Clase interfaz:** define la relación que se establece entre la clase cliente, encargada de enviar peticiones al servidor de acuerdo a las acciones del usuario.

- **Clase controladora:** encargada de procesar la información utilizando un conjunto de servicios que ofrece el marco de trabajo *Flask* y que permiten mostrar los resultados en las vistas correspondientes utilizando el motor de plantillas *Jinja2*.

2.4.2 Patrones de diseño

Los patrones *GRASP* (Patrones Generales de *Software* para Asignar Responsabilidades) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades. Se clasifican como Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador, Polimorfismo, Fabricación pura, Indirección y Variaciones protegidas (LARMAN, 2003). Los patrones *GRASP* utilizados para el diseño de la herramienta son:

Experto: Este patrón plantea que se debe asignar una responsabilidad al experto en información. En otras palabras, la clase que cuente con la información necesaria para cumplir con una determinada responsabilidad (LARMAN, 2003). El patrón experto se ve reflejado en la clase *Login()*, este se encarga de encapsular la lógica referente al manejo de usuarios y contraseñas.

Controlador: Este patrón tiene como objetivo asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente el sistema global o a un escenario de casos de uso en el que tenga lugar el evento (LARMAN, 2003). La utilización de este patrón se evidencia en la clase controladora *ServerController()*, que se encarga de atender y ofrecer respuesta a cada una de las solicitudes del usuario relacionadas con el proceso de visualización de los detalles técnicos y de generar reportes.

Bajo Acoplamiento: Este patrón define cómo mantener bajas dependencias, bajo impacto al cambio e incrementar la reutilización. Asignar una responsabilidad de manera que el acoplamiento sea bajo. Un componente con bajo acoplamiento no mantiene fuertes dependencias con otros componentes (LARMAN, 2003). La utilización de este patrón se evidencia en las clases *Memory()*, *Network()* y *Display()*, las cuales para cumplir con sus funciones solo necesita relacionarse con la clase *ServerController()*.

A continuación, se define la arquitectura de la propuesta de solución.

2.5 Definición de la arquitectura

La arquitectura de *software* representa la estructura del sistema. Define los componentes que lo integran y las relaciones que existen entre ellos. Para el desarrollo de la herramienta se propone el patrón

arquitectónico en capas. Este patrón define una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Además, cada capa opera sólo con capas adyacentes (SOMMERVILLE, 2005, 2011; NC, 2017; IBM CORP, 2018; PRESSMAN, 2010). La arquitectura propuesta para el desarrollo de la herramienta de visualización de los detalles técnicos de los recursos de *hardware* y *software* del equipo identifica como capas:

- **Capa de Presentación:** representa la interfaz de navegación web de la herramienta.
- **Capa de Lógica del negocio:** representa las implementaciones concretas de cada uno de los procedimientos necesarios para desarrollar la herramienta.

2.5.1 Diseño arquitectónico

El diseño arquitectónico permite entender cómo se debe organizar el sistema y cómo se debe diseñar su estructura global. A continuación, se describe el diagrama de paquetes empleados en la propuesta de solución.

Diagrama de Paquetes

Los diagramas de paquetes se utilizan para representar las capas identificadas por el patrón arquitectónico empleado. A continuación, en la Figura 5 se muestra el diagrama de paquetes de la propuesta de solución.

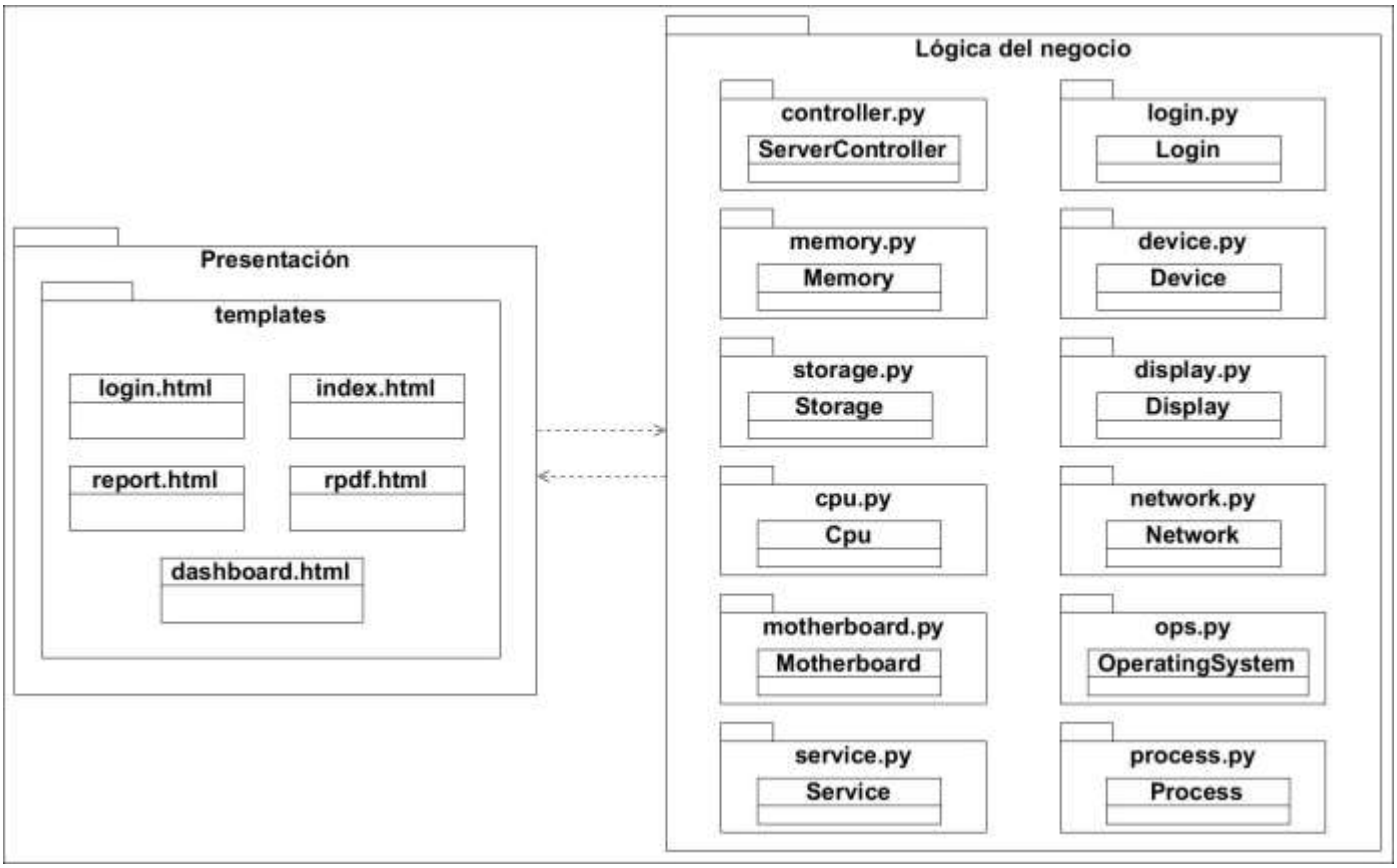


Figura 4: Diseño arquitectónico de la propuesta de solución.

(Fuente: elaboración propia)

Capa de Presentación: contienen las plantillas que corresponden a cada una de las vistas de la solución, estas son: *index.html* y *login.html*.

Capa de Lógica del negocio: contienen las clases que corresponden a la implementación de cada procedimiento: *Device(), Cpu(), Display(), Memory(), Motherboard(), Network(), OperatingSystem(), Storage(), Process()*, *Service()*, *Login()* y la clase controladora *ServerController().*

2.6 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Un diagrama de componentes representa las dependencias entre componentes de *software*, incluyendo componentes de código fuente, componentes del código binario y componentes ejecutables

(SCHMULLER, 2001). A continuación, en la Figura 6 se presenta el diagrama de componentes de la herramienta a desarrollar.

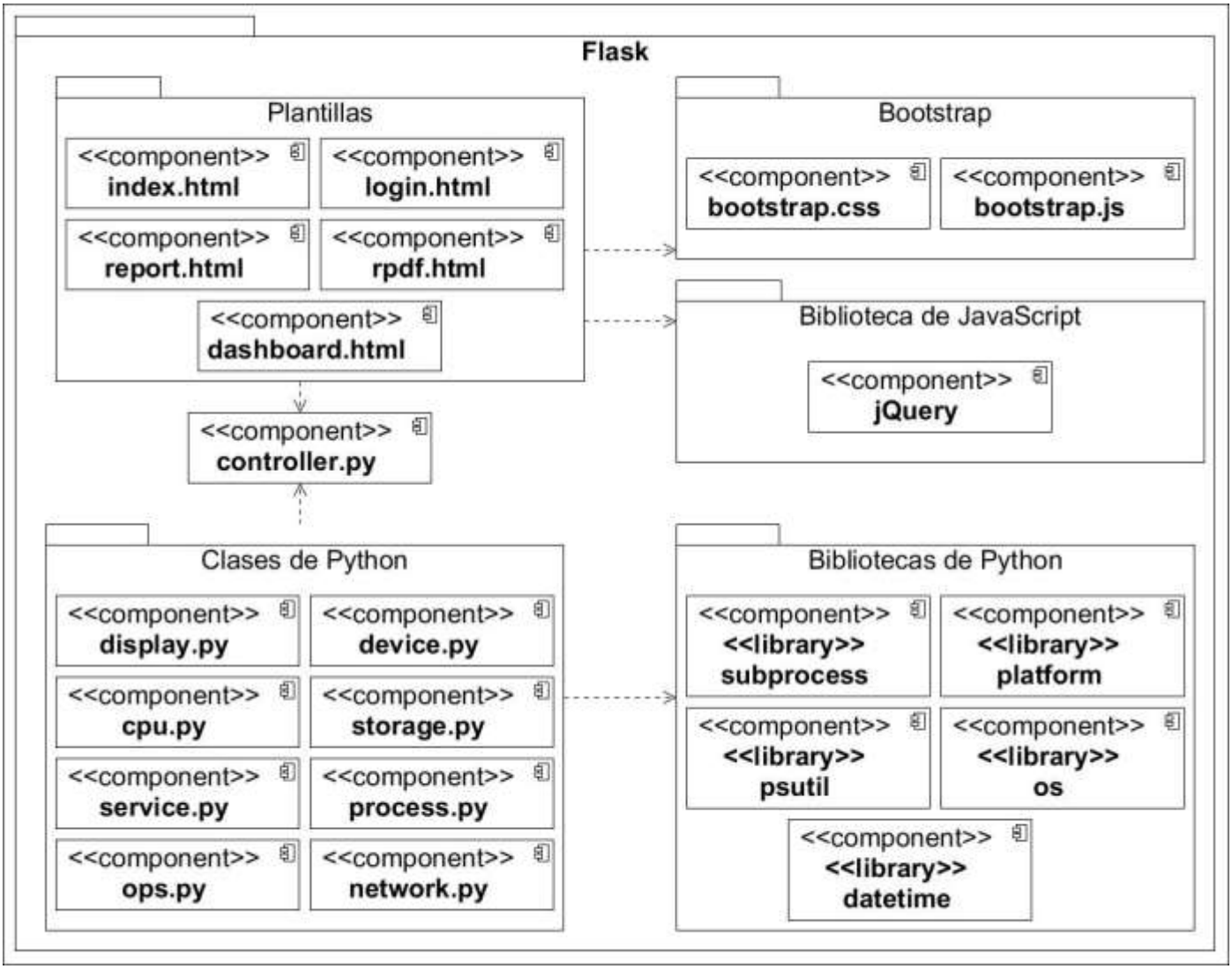


Figura 5: Diagrama de Componentes.

(Fuente: elaboración propia)

Descripción de los componentes

- **Plantillas:** Contiene las interfaces con las que interactúa el usuario.
- ***Bootstrap*:** Contiene los archivos *.css* (extensión de los archivos *CSS*) y *.js* (extensión de los archivos *JavaScript*) utilizados en las plantillas.

- ***Controller.py*:** Archivo que contiene todas las vistas de la herramienta.
- **Clases de *Python*:** Contiene las clases que utilizan las bibliotecas de *Python*.
- **Bibliotecas de *Python*:** Archivos de *Python* que permiten visualizar detalles técnicos del equipo.

2.7 Conclusiones parciales

Se desarrolló el análisis y diseño de la propuesta de solución haciendo uso de los artefactos que propone la metodología Variación de *AUP* para la UCI, en el escenario 4; y los resultados obtenidos arribando a las siguientes conclusiones:

- La especificación de los requisitos funcionales y no funcionales, contribuyeron a una mejor comprensión de los resultados que se pretenden obtener y guiaron la codificación de la herramienta.
- La definición de la arquitectura y los patrones de diseño GRASP propuestos, permitieron establecer las bases para fomentar las buenas prácticas de programación durante la fase de implementación.

CAPÍTULO 3: Implementación y pruebas de la herramienta web para Nova Servidores 7.0

En el presente capítulo, se describen los estándares de codificación que se emplean en el desarrollo de la herramienta; así como el diagrama de despliegue, y un ejemplo de la interfaz gráfica de usuario. Además, se realizan pruebas de *software* con el objetivo de descubrir y corregir los posibles errores de la herramienta de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.

3.1 Implementación

El objetivo de esta disciplina es construir el sistema informático, a partir de los resultados del Análisis y Diseño. Una implementación es la realización de una especificación técnica o algoritmos con un programa. Muchas implementaciones son realizadas según una especificación o un estándar (Sánchez, 2015; PRESSMAN, 2010). Durante la implementación se utilizaron los estándares de codificación descritos a continuación.

3.1.1 Estándares de codificación

Un estándar de codificación comprende un conjunto de reglas, normas o patrones con el objetivo de establecer uniformidad en el proceso de generación de código y la legibilidad del mismo. A continuación, se mencionan los estándares de codificación de *Python* utilizados para la implementación de la herramienta (VAN ROSSUM et al, 2016).

Indentación

- Se utilizaron 4 espacios por cada nivel de indentación.

Comentarios

- Se utilizan para describir cómo una sección de un código hace o ejecuta algo, poniendo énfasis en no comentar lo evidente.

Convenciones de nombres

- Los nombres de funciones y métodos están en letras minúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad.
- Se utilizó *CapWords* (palabras que comienzan con mayúsculas) para los nombres de las clases.

A continuación, se presenta un fragmento del código del controlador ServerController(), donde se evidencia el empleo de los estándares de codificación:

Figura 6: Estándares de codificación.

(Fuente: elaboración propia)

Los estándares de codificación definidos en la investigación permiten tener un estilo único en la implementación de la solución, facilita el estudio y entendimiento del código de la propuesta de solución y posibilita su fácil mantenimiento.

En el epígrafe 3.1.2 quedan evidenciados los resultados de la implementación de la herramienta web de visualización de detalles técnicos mediante el ejemplo de la interfaz gráfica de usuario mostrada en la Figura 8. Para su implementación se tuvo en cuenta las reglas de oro definidas por Roger Pressman para el diseño de interfaces gráficas de usuario.

3.1.2 Ejemplo de interfaz gráfica de usuario

La interfaz de usuario es una parte de un programa que gestiona la interacción con el usuario basándose en relaciones visuales como iconos, menús o formularios. Su principal objetivo es facilitar la comunicación

con la computadora (PRESSMAN, 2010). A continuación, se relacionan las reglas empleadas en el diseño e implementación de las interfaces gráficas de usuario:

Reglas de oro:

1. Dar control al usuario.
2. Reducir la carga de memoria del usuario.
3. Construcción de una interfaz consistente.

El cumplimiento de estas reglas permite desarrollar interfaces flexibles, en las que el usuario no realice acciones innecesarias y no deseadas, así como garantizar la interrupción de una secuencia de acciones en el instante requerido por el usuario. Brinda la posibilidad de desglosar la información de forma progresiva a través de una estructura organizada, orientar al usuario en las tareas a desarrollar y que esas tareas se realicen en el contexto adecuado (PRESSMAN, 2010).

En la Figura 7 se muestra la interfaz de usuario correspondiente a la vista de autenticación. En ella el usuario puede registrarse, con el nombre y la contraseña, para acceder a los detalles técnicos de los recursos de *hardware* y *software* del equipo.

Figura 7: Interfaz gráfica de usuario de la vista de autenticación.

(Fuente: elaboración propia)

En la Figura 8 se muestra la interfaz de usuario correspondiente a la vista de servicios. En ella se listan todos los servicios instalados en el equipo, el estado (Activo o Inactivo), los detalles técnicos de cada servicio. Además, permite filtrar los servicios por el nombre y ordenar la tabla de manera ascendente o descendente, por servicios o estados.

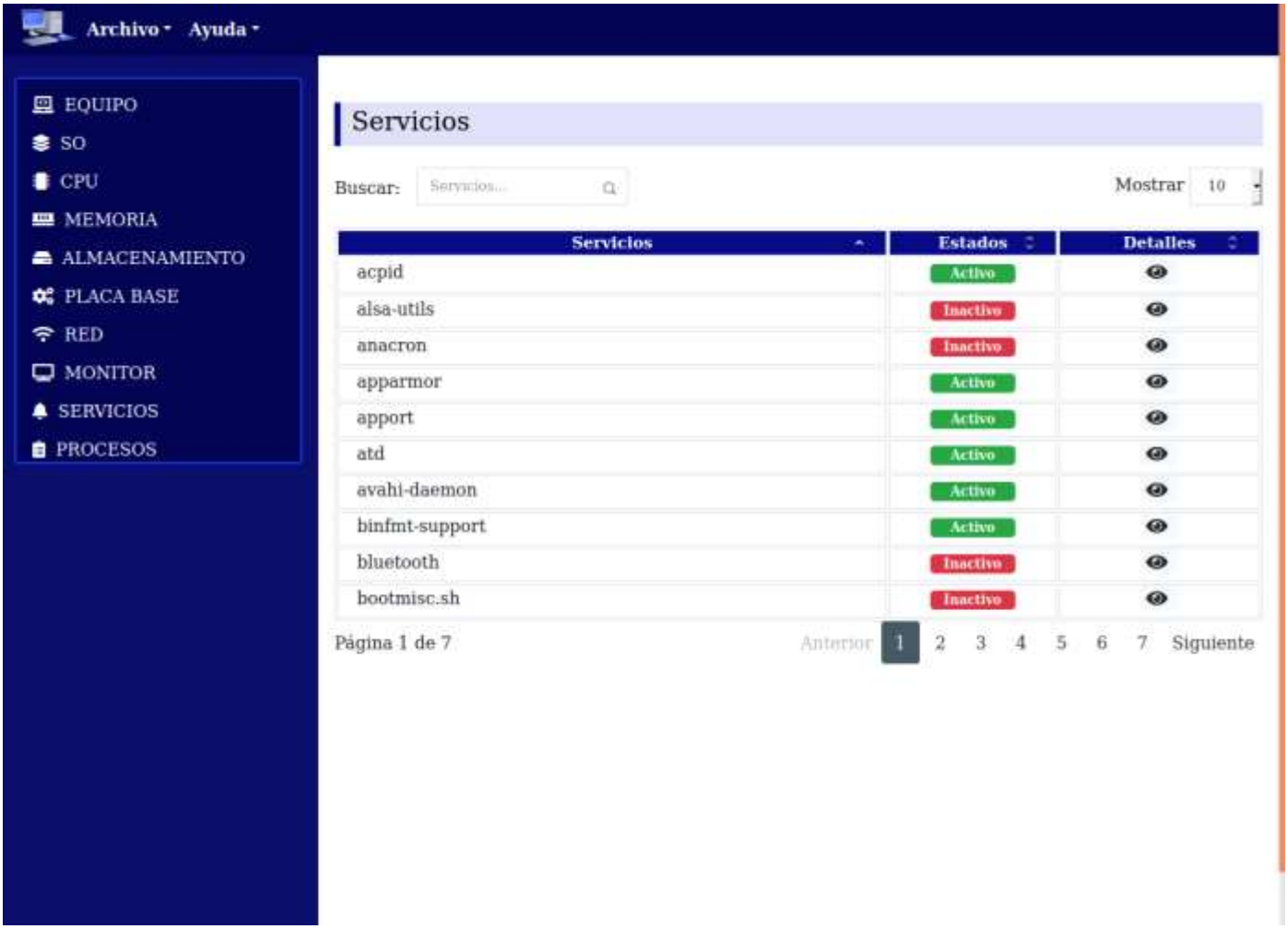


Figura 8: Interfaz gráfica de usuario de la vista de servicios.

(Fuente: elaboración propia)

En el epígrafe 3.2 se presenta el diagrama de despliegue de la herramienta web de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0. En él se detallan las características del entorno de ejecución de la herramienta.

3.2 Diagrama de despliegue

El diagrama de despliegue permite mostrar la arquitectura en tiempo de ejecución del sistema respecto al *hardware* y *software*. Se representa mediante nodos, con sus respectivos componentes, y las relaciones

que existen entre ellos (LARMAN, 2003). En la Figura 9 se presenta el diagrama de despliegue elaborado para la propuesta de solución.

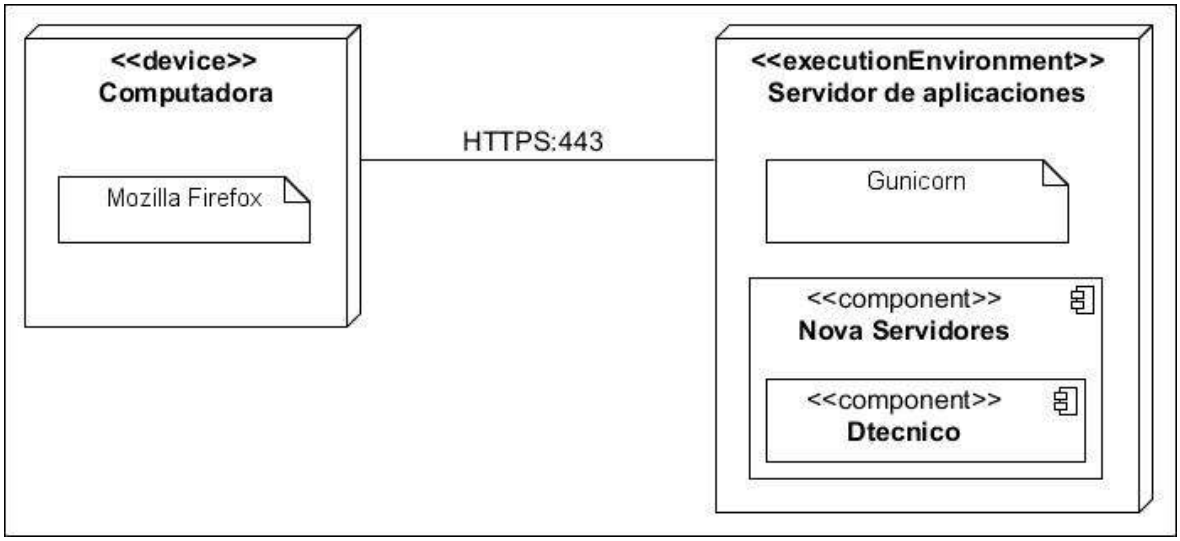


Figura 9: Diagrama de despliegue.

(Fuente: elaboración propia)

A continuación, se describen los nodos, componentes y protocolos de comunicación representados en el diagrama anterior.

Descripción de los nodos

- **Computadora:** estación de trabajo desde la cual se accede, a través de un navegador web, a la herramienta de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.
- **Servidor de aplicaciones:** dispositivo que proporciona los servicios de la herramienta. Para la propuesta de solución se utiliza *Gunicorn* en su versión 20.0.4.

Descripción de los componentes

- **Nova Servidores:** distribución cubana GNU/Linux para servidores.
- **Dtecnico:** herramienta de visualización de detalles técnicos de los recursos de *hardware* y *software* del equipo.

Descripción del protocolo

- **HTTPS:** protocolo de trasferencia de datos en su versión segura; utilizado en la herramienta de visualización de detalles técnicos para la transmisión de los datos entre el controlador y la vista.

3.3 Pruebas de *software*

Las pruebas de *software* consisten en la verificación del comportamiento de un programa en un conjunto finito de casos de prueba. El objetivo de las pruebas es descubrir posibles errores de implementación y calidad de un programa informático (PRESSMAN, 2010). A continuación, se describen los tipos de prueba de *software*, métodos y técnicas aplicados en la evaluación de la solución desarrollada.

3.3.1 Tipos de pruebas de *software*

Pruebas Unitarias

Las pruebas unitarias o de componente consisten en la ejecución de actividades que le permitan verificar la funcionalidad y la estructura (lógica interna) de cada elemento individualmente, una vez que ha sido codificado (PRESSMAN, 2010). Estas pruebas se realizan sobre cada sección del *software*, de manera independiente, con el objetivo de comprobar que está correctamente codificada.

Pruebas Funcionales

Las pruebas funcionales se centran en comprobar que el sistema funcione acorde a los requisitos funcionales y no funcionales, detectando posibles defectos derivados de errores en la fase de desarrollo. Mediante estas pruebas se demuestra que las funciones del *software* son operativas, la entrada se acepta de forma adecuada y se produce una salida correcta (PRESSMAN, 2010).

3.3.2 Métodos de prueba

Método de caja blanca

Las pruebas de caja blanca se centran en los detalles procedimentales del *software* (código fuente), para obtener los casos de prueba. Mediante este método, el ingeniero del *software* puede examinar el estado de la aplicación, en varios puntos, para determinar si el estado real coincide con el estado deseado o esperado (PRESSMAN, 2010).

Método de caja negra

Las pruebas de caja negra o pruebas funcionales se aplican sobre la interfaz del *software*. Mediante este método, el ingeniero del *software* puede validar que las salidas sean las esperadas. Además, se centra en encontrar las circunstancias en las que el sistema no se comporte conforme a las especificaciones establecidas (PRESSMAN, 2010).

3.3.3 Técnicas de prueba

Camino básico

La técnica del camino básico es un método de prueba de caja blanca que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Esta técnica debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, al tratar de confirmar que cada camino independiente sea ejecutado al menos una vez en el sistema (PRESSMAN, 2010).

Partición de equivalencia

La técnica de partición de equivalencia es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos, de los que pueden derivarse casos de prueba. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (PRESSMAN, 2010).

3.4 Aplicación de las pruebas de *software*

En el presente epígrafe se presentan los resultados obtenidos de las diferentes pruebas de *software* definidas en el epígrafe 3.3.1 Tipos de pruebas de *software*, empleando los métodos y técnicas descritos en los epígrafes 3.3.2 Métodos de prueba y 3.3.3 Técnicas de prueba, durante la ejecución de las disciplinas Pruebas internas y Pruebas de aceptación propuestas por la metodología de desarrollo de *software* variación AUP para la UCI.

3.4.1 Pruebas Internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas (Sánchez, 2015). Durante esta disciplina se aplicaron pruebas unitarias y funcionales, descritas a continuación.

Pruebas unitarias

Las pruebas unitarias se realizaron a través del método de prueba caja blanca y la técnica de camino básico. Para aplicar la prueba de camino básico, se debe realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del sistema. El procedimiento de mayor valor tiene una alta probabilidad de contener errores, además de que ofrece una medida del número de pruebas que deben diseñarse para validar la correcta implementación de una determinada función (PRESSMAN, 2010). En la Figura 10 se presenta el método *_distribution()* correspondiente al RF.7 Mostrar detalles técnicos del sistema operativo.

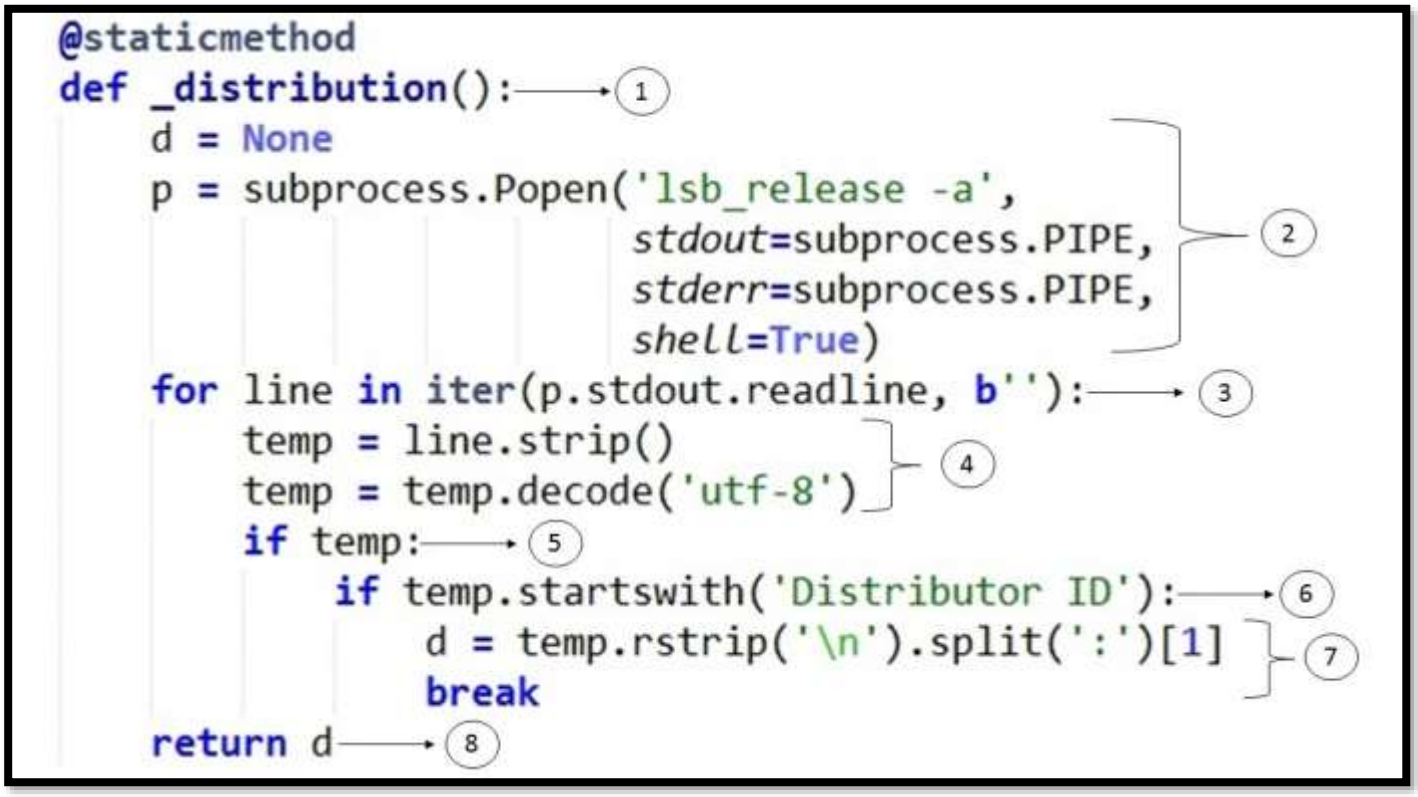


Figura 10: Método _distribution() correspondiente al RF.7 Mostrar detalles técnicos del sistema operativo.

(Fuente: elaboración propia)

1. Dibujar el grafo de flujo de la funcionalidad

Luego de numerar las líneas de código, se diseña el grafo que describe el flujo de control lógico de la funcionalidad, empleando nodos y aristas; como se representa en la Figura 11.

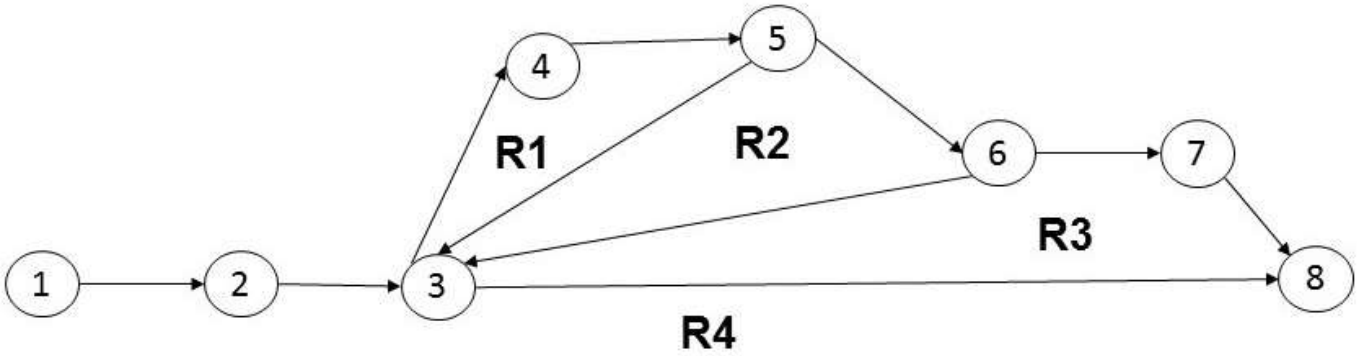


Figura 11: Grafo de flujo.

(Fuente: elaboración propia)

2. Determinar la complejidad ciclomática

La complejidad ciclomática de un grafo V(G) se puede calcular de tres formas diferentes:

V(G) = A-N+2 (Donde A es el número de aristas del grafo de flujo y N es la cantidad de nodos del grafo)

V(G) = 10-8+2

V(G) = 4

V(G) = P+1 (Donde P es el número de nodos predicados (nodos con más de una arista de salida) contenidos en el grafo)

V(G) = 3+1

V(G) = 4

V(G) = R (Donde R es el número de regiones (áreas delimitadas por nodos y aristas en el grafo))

V(G) = 4

3. Determinar los caminos linealmente dependientes

Tabla 6: Lista de rutas de la funcionalidad _distribution().

(Fuente: elaboración propia)

No. Rutas	Caminos
Ruta 1	1-2-3-4-5-6-7-8

Ruta 2	1-2-3-4-5-6-3-8
Ruta 3	1-2-3-4-5-3-8

4. Definir los casos de prueba para comprobar la ejecución

En el diseño de los casos de prueba se debe especificar los siguientes elementos:

- **Descripción:** contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.
- **Condición de ejecución:** se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del método.
- **Entrada:** se muestran los parámetros de entradas del método.
- **Resultados esperados:** se explica el resultado esperado de la ejecución del método.

En la Tabla 7 se presenta el diseño de caso de prueba de la Ruta 1 del conjunto de caminos básicos linealmente independientes correspondientes a la funcionalidad *_distribution()*:

Tabla 7: Caso de prueba para la Ruta 1 del método _distribution().

(Fuente: elaboración propia)

Caso de prueba unitaria	
No. Rutas: Ruta 1	**Ruta:** 1-2-3-4-5-6-7-8
Responsable de prueba: Beatriz García Jiménez	
Descripción: Método para extraer la distribución del SO, a partir de un comando de GNU/Linux.	
Condición de ejecución: El usuario selecciona la opción de visualizar los detalles técnicos del recurso SO.	
Entrada: comando de GNU / Linux: *lsb_release -a*.	
Resultado: Muestra la distribución del SO.	

Evaluación de la prueba: Satisfactoria.

Luego de realizadas las pruebas unitarias se obtuvo como resultado que el flujo de trabajo de las funcionalidades de la herramienta de visualización de detalles técnicos del equipo es correcto; pues se comprobó que cada sentencia del código fuente se ejecuta al menos una vez.

Pruebas funcionales

Las pruebas funcionales se realizaron a través del método de caja negra y la técnica de partición de equivalencia. Para la aplicación de la técnica de partición de equivalencia se diseñó el caso de prueba **Autenticar usuario** por historias de usuarios; en el que se definen las clases de equivalencia válidas y no válidas para cada campo de entrada de la herramienta.

Tabla 8: Caso de prueba de interfaz de usuario: Autenticar usuario.

(Fuente: elaboración propia)

Caso de prueba de interfaz de usuario	
Número: CDPHU16	**Nombre:** Autenticar usuario.
Probador: Beatriz García Jiménez	
Descripción: Prueba a la funcionalidad Autenticar usuario.	
Condición de ejecución: El usuario debe estar autenticado.	
Entrada / Pasos de ejecución: 1. El usuario accede a la interfaz de autenticación de la herramienta. 2. El usuario llena los campos usuario y clave del formulario Entrar. 3. El usuario presiona el botón Entrar de la interfaz. 4. La herramienta muestra la interfaz principal integrada por un cuadro de mando con los detalles técnicos de los recursos de *hardware* y *software* del equipo.	

Tabla 9: Resultados del Caso de prueba de interfaz de usuario: Autenticar usuario.

(Fuente: elaboración propia)

Escenarios	Resultados esperados	Evaluación
EC1. El usuario introduce correctamente los datos.	La herramienta autentica al usuario y entra en la interfaz principal mostrando el cuadro de mando con los detalles técnicos de los recursos de *hardware* y *software* del equipo.	Satisfactoria
EC2. El usuario introduce datos incorrectos.	La herramienta no autentica al usuario y muestra el siguiente mensaje: “Datos incorrectos”.	Satisfactoria
EC3. El usuario deja campos vacíos.	La herramienta no autentica al usuario y muestra los siguientes mensajes: “Usuario o clave incorrecta”.	Satisfactoria

En la Figura 12 se presenta un gráfico con los resultados de las pruebas funcionales.

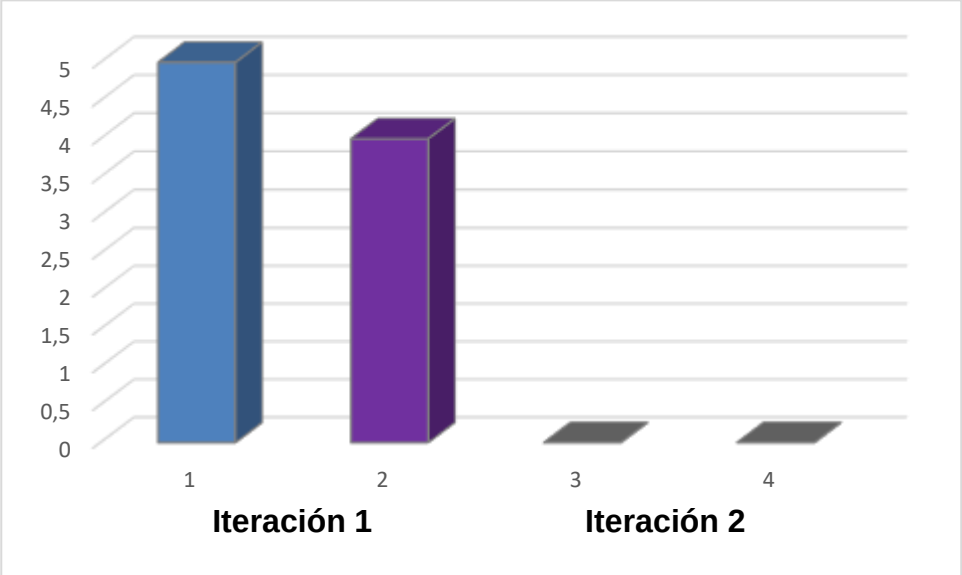


Figura 12: Resultados de las pruebas funcionales.

(Fuente: elaboración propia)

Luego de realizadas las pruebas funcionales a partir de 2 iteraciones; en la primera se identificaron 9 no conformidades, de ellas 4 fueron errores ortográficos y 5 de validación. Durante la segunda iteración no se identificaron no conformidades.

Pruebas de rendimiento

Las pruebas de rendimiento son un conjunto de pruebas no funcionales que se realizan, para determinar la velocidad de ejecución de una tarea concreta en un sistema bajo condiciones particulares de trabajo (PRESSMAN, 2010).

Los **objetivos** de estas pruebas son:

- Validar y verificar atributos de la calidad del sistema: uso de los recursos, escalabilidad y fiabilidad.
- Comparación de sistemas para encontrar cuál de ellos funciona mejor.
- Determinar qué componentes del sistema provocan que el conjunto presente rendimientos bajos.

Tipos de Pruebas de Rendimiento

Prueba de Carga: prueba de rendimiento que se realiza para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada (PRESSMAN, 2010).

Objetivos:

- Mostrar los tiempos de respuesta de todas las transacciones importantes.
- Localizar los “cuellos de botella” de una aplicación.

Pruebas de Estrés: Prueba de rendimiento que se realiza para observar el comportamiento de una aplicación bajo una cantidad de peticiones extrema (PRESSMAN, 2010).

Objetivos:

- “Romper” la aplicación.
- Determinar cómo rendirá la aplicación si la carga real supera a la carga esperada.

Resultados de las pruebas de rendimiento

La herramienta desarrollada fue sometida a varias pruebas de rendimiento para medir su funcionamiento. Estas pruebas estuvieron enfocadas en el análisis del comportamiento de los tiempos de respuesta de cada vista de la herramienta.

Para la realización de las pruebas de carga y estrés se utilizó la herramienta *Apache Jmeter*, descrita en el epígrafe 1.6.7 Herramienta de rendimiento. Las pruebas se realizaron desde una computadora con 31GB de RAM, microprocesador Intel Core i5 con 3.40 GHz y sistema operativo Ubuntu 16.04.6. A continuación, se describen las variables que miden el resultado de las pruebas de carga y estrés realizadas a la herramienta.

- **Muestra:** Cantidad de peticiones realizadas.
- **Media:** Tiempo promedio en milisegundos en el que se obtienen los resultados.
- **Min.:** Tiempo mínimo que demora un hilo en acceder al sistema.
- **Max.:** Tiempo máximo que demora un hilo en acceder al sistema.
- **Desviación estándar:** Promedio con respecto a la media aritmética.
- **% Error:** Por ciento de error de los servicios que no se llegaron a cargar de manera satisfactoria.
- **Rendimiento:** El rendimiento se mide en cantidad de solicitudes por segundo.
- **KB/s:** El rendimiento se mide en cantidad de kilobytes[7] por segundo.

Como se muestra en las Tablas 10 y 11, se simularon las peticiones realizadas al sistema por un total de 100 y 500 usuarios simultáneamente, obteniéndose los siguientes resultados:

Tabla 10: Resultados obtenidos a partir de las pruebas de carga y estrés por 100 usuarios.
(Fuente: elaboración propia)

[7] Unidad de almacenamiento de información equivalente a 1024 bytes.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Están...	% Error	Rendimiento	Kb/sec
EQUIPO	100	2883	194	6109	1428,43	0,00%	6,4/sec	69,46
SO	100	4459	290	6693	1507,61	0,00%	5,1/sec	42,10
CPU	100	1735	230	2359	594,45	0,00%	4,8/sec	59,05
MEMORIA	100	1115	16	1678	466,77	0,00%	4,6/sec	44,73
ALMACENAMIENTO	100	1207	40	1708	471,07	0,00%	4,3/sec	61,81
PLACA BASE	100	1169	17	1731	466,59	0,00%	4,0/sec	30,29
RED	100	2607	587	3877	688,15	0,00%	3,6/sec	28,23
MONITOR	100	1391	188	1969	407,14	0,00%	3,6/sec	27,59
SERVICIOS	100	18335	16130	20179	1151,90	0,00%	2,3/sec	411,58
PROCESOS	100	525	195	1946	422,96	0,00%	4,0/sec	825,33
Total	1000	3542	16	20179	5122,74	0,00%	21,8/sec	1008,30

Las pruebas realizadas muestran que el sistema es capaz de responder a 1000 peticiones de 100 usuarios conectados simultáneamente, en un tiempo promedio de 3542 milisegundos (3.5 segundos aproximadamente) con 0% de error.

Tabla 11: Resultados obtenidos a partir de las pruebas de carga y estrés por 500 usuarios.

(Fuente: elaboración propia)

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estánd...	% Error	Rendimiento	Kb/sec
EQUIPO	500	12676	212	55617	6975,49	26,20%	7,8/sec	67,22
SO	500	16864	1807	102230	6934,37	35,20%	3,9/sec	23,52
CPU	500	10632	1222	102228	8255,92	21,80%	3,8/sec	38,27
MEMORIA	500	7417	56	48259	7194,35	16,20%	3,9/sec	33,19
ALMACENAMIENTO	500	7124	813	54611	7698,06	14,00%	4,1/sec	51,66
PLACA BASE	500	7572	279	55182	7225,45	15,80%	4,3/sec	28,44
RED	500	9339	1230	51436	6688,54	12,20%	4,3/sec	30,28
MONITOR	500	9643	1076	49559	6763,17	16,00%	4,6/sec	30,88
SERVICIOS	500	47994	4430	100979	19090,31	12,20%	3,6/sec	573,50
PROCESOS	500	5183	265	21015	5620,77	6,00%	3,9/sec	1455,99
Total	5000	13444	56	102230	14962,60	17,56%	28,2/sec	1675,77

Con el objetivo de analizar el comportamiento del sistema en condiciones extremas, se realizó una prueba de estrés a 5000 peticiones de 500 usuarios, donde respondió en 13444 milisegundos (13 segundos aproximadamente) como tiempo promedio, aunque no fue capaz de responder el 17.56% de las peticiones realizadas. Demostrando que el sistema no responde de forma correcta para esta cantidad de peticiones con los recursos definidos anteriormente.

3.4.2 Pruebas de aceptación

En la disciplina de Prueba de aceptación se realizan las pruebas de *software* finales antes del despliegue del sistema. Su objetivo es verificar que el *software* está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el *software* fue construido (Sánchez, 2015). Las pruebas de aceptación se derivan de las historias de usuarios, donde cada una se convierte en un caso de

prueba en el cual el cliente especifica los puntos a probar. A continuación, en la Tabla 12 se muestra un caso de prueba para la funcionalidad **Mostrar detalles técnicos de la *CPU*.**

Tabla 12: Caso de prueba de aceptación: Mostrar detalles técnicos de la CPU.

(Fuente: elaboración propia)

Caso de prueba de aceptación	
Número: HU2CP2	**Historia de usuario:** Mostrar detalles técnicos de la *CPU.*
Responsable de prueba: Beatriz García Jiménez	
Descripción: Prueba para comprobar el proceso de visualización de detalles técnicos del recurso *CPU*.	
Condiciones de ejecución: El usuario tiene que estar autenticado.	
Entrada / Pasos de ejecución: 1. El usuario accede a la interfaz principal de la herramienta. 2. El usuario activa la opción **CPU** del menú de navegación.	
Resultado esperado: Muestra los detalles técnicos del recurso *CPU*, un gráfico dinámico (en tiempo real) con el promedio del porciento de uso de todos los *CPU* y un gráfico que refleja el porciento de uso de la *CPU* para cada núcleo.	
Evaluación de la prueba: Prueba satisfactoria.	

Una vez terminado este proceso el cliente quedó satisfecho, pues todos los casos de prueba se ejecutaron de forma satisfactoria.

3.5 Conclusiones parciales

Se logró la informatización del proceso de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0, a través de la herramienta desarrollada y los resultados obtenidos arribando a las siguientes conclusiones:

- La elaboración del diagrama de componentes, permitió una mejor comprensión de la estructura de los componentes del sistema implementado.
- La aplicación de las pruebas unitarias permitió corroborar que el flujo de trabajo de las funcionalidades implementadas es el correcto, pues cada sentencia se ejecuta al menos una vez.
- La ejecución de las pruebas funcionales posibilitó identificar las deficiencias de la propuesta de solución y garantizar el cumplimiento de los requisitos definidos.
- Mediante las pruebas de aceptación se verificó que la herramienta está lista para ser usada por los usuarios.

CONCLUSIONES GENERALES

Con el desarrollo de una herramienta web de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0 se da cumplimiento al objetivo general planteado. Para lograr este resultado se puede arribar a las siguientes conclusiones:

- El análisis del marco teórico sobre el proceso de visualización de detalles técnicos de los recursos de *hardware* y *software* del equipo, así como el estudio de aplicaciones informáticas que permiten visualizar detalles técnicos, demostraron la necesidad de desarrollar una herramienta web que contribuya con el proceso de visualización de detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0.
- Los 17 requisitos funcionales y los 5 no funcionales definidos permitieron cumplir con las necesidades del cliente. El análisis y diseño propició la elaboración de una herramienta web para la visualización de detalles técnicos del equipo. El uso de los patrones de diseño *GRASP* y el patrón arquitectónico en capas 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 implementada.
- El desarrollo de una herramienta web que visualiza los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0, permitió cumplir con las exigencias del cliente y las restricciones del diseño elaborado.

REFERENCIAS BIBLIOGRÁFICAS

ALCALDE, IGNASI. *Visualización de la información: de los datos al conocimiento*. Barcelona, España: Editorial UOC, noviembre 2015. ISBN: 978-84-9064-753-0.

APACHE JMETER. *Apache JMeter*. [En línea] 2017. [Citado el: 14 de marzo de 2020]. Disponible en: http://www.jmeter.apache.org.

BAHIT, EUGENIA. *Curso: Python para Principiantes*. [En línea] 2012. [Citado el: 6 de noviembre de 2019]. Disponible en: http://www.safecreative.org/work/1207302042960.

BARWALDT, ERIK. *Reading hardware information with I-Nex*. [En línea] 2017. [Citado el: 24 de noviembre de 2019]. Disponible en: http://www.linux-magazine.com/Issues/2017/201/I-Nex.

BERLIOS.DE. *HardInfo*. [En línea] 2019. [Citado el: 15 de enero de 2020]. Disponible en: https://www.berlios.de/software/hardinfo/.

BOOTSTRAP TEAM. *Bootstrap*. [En línea] 2019. [Citado el: 25 de enero de 2020]. Disponible en: http://www.getbootstrap.com.

CENTOS PROJECT. *CentOS Linux*. [En línea] 2020. [Citado el: 27 de enero de 2020]. Disponible en: https://www.centos.org.

CHRISTENSSON, P. *RAM Definition*. [En línea] 21 de junio de 2019. [Citado el: 11 de enero de 2020]. Disponible en: https://www.techterms.com/definition/ram.

CHRISTENSSON, P. *Network Definition*. [En línea] 19 de junio de 2018. [Citado el: 11 de enero de 2020]. Disponible en: https://www.techterms.com/definition/network.

CONSORCIO WORLD WIDE WEB (W3C). *Guía Breve de CSS*. [En línea] 2017. [Citado el: 10 de febrero de 2020]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.

CPANEL. *cPanel*. [En línea] 2020. [Citado el: 27 de enero de 2020]. Disponible en: https://www.cpanel.net.

CPUID. *HWMonitor PRO*. [En línea] 2019. [Citado el: 25 de enero de 2020]. Disponible en: https://www.cpuid.com/softwares/hwmonitor-pro-android.html.

CPUID. *HWMonitor*. [En línea] 2019. [Citado el: 25 de enero de 2020]. Disponible en: https://www.cpuid.com/softwares/hwmonitor.html.

DICCIONARIO DE INFORMÁTICA. *Definición de hardware*. [En línea] 2019. [Citado el: 5 de marzo de 2020]. Disponible en: https://www.abrirllave.com/diccionario-de-informatica/hardware.php.

DICCIONARIO DE INFORMÁTICA. *Definición de equipo informático*. [En línea] 2019. [Citado el: 6 de marzo de 2020]. Disponible en: https://www.abrirllave.com/diccionario-de-informatica/equipo-informatico.php**.**

DUPLAIN, RON. *Instant Flask Web Development*. Livery Place, Birmingham B3 2PB, UK: Editorial Packt Publishing Ltd., 2013. ISBN 978-1-78216-962-8.

FERNÁNDEZ, GREGORIO F. *Elementos de sistemas operativos, de representación de la información y de procesadores hardware y software.* Universidad Politécnica de Madrid. España: 2015, p. 22.

FILEOPT. *Speccy*. [En línea] 2017. [Citado el: 23 de noviembre de 2019]. Disponible en: https://www.fileopt.com/software/speccy.

FINALWIRE LTD. *AIDA64*. [En línea] 2015. [Citado el: 23 de noviembre de 2019]. Disponible en: https://www.aida64.com/corporate-solutions.

FOLDOC. *Definición de servicio*. [En línea] 2019. [Citado el: 12 de enero de 2020]. Disponible en: http://foldoc.org/operating+systems.

GREY, Li. *Bootstrap-Flask Documentation*. [En línea] 2019. [Citado el: 5 de diciembre de 2019]. Disponible en: https://bootstrap-flask.readthedocs.io/en/latest.

GRINBERG, MIGUEL*. Flask Web Development*. First Edition. Gravenstein Highway North, Sebastopol, United States of America: Editorial O'Reilly Media, Inc., 2014. ISBN 978-1-4493-7262-0.

GUERRA, CÉSAR A. *Obtención de Requerimientos. Técnicas y Estrategia*. [En línea] 2017. [Citado el: 18 de febrero de 2020]. Disponible en: https://www.sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

GUNICORN DEVELOPERS. *gunicorn*. [En línea] 2019. [Citado el: 14 de marzo de 2020]. Disponible en: https://www.gunicorn.org.

IBM CORP. *Artefacto. Prototipo de interfaz de usuario*. [En línea] 2018. [Citado el: 20 de febrero de 2020]. Disponible en:

https://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/core.base_rup/workproducts/rup_user_interface_prototype_7237E5AA.html.

JETBRAINS. *PyCharm*. [En línea] 2019. [Citado el: 9 de noviembre de 2019]. Disponible en: https://www.jetbrains.com/pycharm.

JQUERY. *What is jQuery?* [En línea] 2020. [Citado el: 14 de febrero de 2020]. Disponible en: https://www.w3schools.com/jquery/jquery_intro.asp.

KLIPFOLIO INC*. What is a data dashboard?* [En línea] 2020. [Citado el: 18 de marzo de 2020]. Disponible en: https://www.klipfolio.com/resources/articles/what-is-data-dashboard.

LARMAN, CRAIG. *UML y Patrones*. Segunda. Madrid, España: Editorial Pearson Educación, S.A., 2003. ISBN 84-205-3438-2.

LUCID SOFTWARE INC. Qué es un modelo de base de datos. [En línea] 2019. [Citado el: 5 de marzo de 2020]. Disponible en: https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos.

MALÍK, MARTIN. *HWinfo*. [En línea] 2018. [Citado el: 24 de noviembre de 2019]. Disponible en: https://www.hwinfo.com/about-software.

MANZ. *Lenguaje HTML - Documentación sobre desarrollo web*. [En línea] 2017. [Citado el: 10 de febrero de 2020]. Disponible en: https://lenguajehtml.com.

MINISTERIO DE COMUNICACIONES. *Política Integral para el Perfeccionamiento de la Informatización de la Sociedad en Cuba*. La Habana: 2017, p. 4-5.

MORA, E.C. de la C*. Herramienta para la administración de repositorios de la Distribución Cubana de GNU/Linux Nova.* En: Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas. UCI. La Habana: 2017.

OFICINA NACIONAL DE NORMALIZACIÓN (NC). *INGENIERÍA DE SOFTWARE Y SISTEMAS – REQUISITOS DE LA CALIDAD Y EVALUACIÓN DE SOFTWARE Y SISTEMAS (SQuaRE) – MEDICIÓN DE LA CALIDAD EL PRODUCTO DE SOFTWARE Y DEL SISTEMA*. [En línea] 2017. [Citado el: 3 de marzo de 2020]. Disponible en: http://www.nc.cubaindustria.cu.

OMG. *Introduction to OMG'S Unified Modeling Language*. [En línea] 2019. [Citado el: 8 de noviembre de 2019]. Disponible en: https://www.uml.org/what-is-uml.htm.

PARTIDO COMUNISTA DE CUBA. *Lineamientos de la Política Económica y Social del Partido y la Revolución para el período 2016-2021*. En: 7mo Congreso del Partido. La Habana: 2017, p. 19-20.

PENCIL PROJECT. *Pencil*. [En línea] 2019. [Citado el: 6 de marzo de 2020]. Disponible en: http://www.pencil.evolus.vn/Features.html.

PEREIRA F., LEANDRO A. *HardInfo - small system administration tool*. [En línea] 2018. [Citado el: 24 de noviembre de 2019]. Disponible en: https://www.linuxlinks.com/hardinfo.

PÉREZ LÓPEZ; RIBAS I XIRGO, et al. *Software Libre. Introducción al desarrollo del software*. Barcelona España: Editorial Eureca Media, SL, marzo 2004. ISBN: 84-9788-119-2. Primera edición, p. 183-193.

PIRIFORM LTD. *Speccy*. [En línea] 2017. [Citado el: 23 de noviembre de 2019]. https://www.ccleaner.com/speccy.

PORATTI, G. *REDES. La guía de referencia actual y definitiva*. Ciudad de Buenos Aires, Argentina: MP Ediciones S.A, 2003.

PRESSMAN, ROGER S. *Ingeniería del Software. Un enfoque práctico*. Séptima. Nueva York, E.U.A.: Editorial McGRAW-HILL INTERAMERICANA EDITORES, S.A., 2010. ISBN 978-607-15-0314-5.

RAE. *Definición de cliente-servidor*. [En línea] 2019a. [Citado el: 7 de marzo de 2020]. Disponible en: https://www.dle.rae.es/cliente.

RAE. *Definición de detalle*. [En línea] 2019b. [Citado el: 20 de marzo de 2020]. Disponible en: https://www.dle.rae.es/detalle.

RAE. *Definición de técnico*. [En línea] 2019c. [Citado el: 20 de marzo de 2020]. Disponible en: https://www.dle.rae.es/técnico.

RAE. *Definición de Arquitectura*. [En línea] 2019d. [Citado el: 12 de enero de 2020]. Disponible en: https://www.dle.rae.es/arquitectura?m=form.

RAE. *Definición de sistema*. [En línea] 2019e. [Citado el: 9 de enero de 2020]. Disponible en: https://www.dle.rae.es/sistema?m=form.

RAE. *Definición de Red*. [En línea] 2019f. [Citado el: 12 de enero de 2020]. Disponible en: https://www.dle.rae.es/red?m=form.

RAE. *Definición de Monitor*. [En línea] 2019g. [Citado el: 12 de enero de 2020]. Disponible en: https://www.dle.rae.es/monitor?m=form.

RODRÍGUEZ, G.; **BATISTA, G. et al**. *Metodología de la Investigación Educacional*. La Habana: Editorial Pueblo y Educación, 1996, p. 41-49.

RODRÍGUEZ, S.; JESÚS J. et al. *Introducción a la programación. Teoría y práctica*. San Vicente (Alicante), España: Editorial Club Universitario, 2003. ISBN 84-8454-274-2.

RONACHER, ARMIN. *Jinja2 Documentation*. [En línea] 2017. [Citado el: 13 de febrero de 2020]. Disponible en: https://buildmedia.readthedocs.org/media/pdf/jinja2/latest/jinja2.pdf.

SÁNCHEZ, TAMARA R. *Metodología de desarrollo para la Actividad productiva de la UCI*. Universidad de las Ciencias Informáticas. La Habana, Cuba: 2015.

SCHMULLER, JOSEPH. *Aprendiendo UML en 24 horas*. Madrid, España: Editorial Pearson Educación, S.A., 2001.

STACK OVERFLOW DOCUMENTATION. *Aprendizaje GNU/Linux*. [En línea] 2019. [Citado el: 12 de enero de 2020]. Disponible en: https://www.riptutorial.com/es/ebook/linux.

STACK OVERFLOW DOCUMENTATION. *Aprendiendo Flask*. [En línea] 2016. [Citado el: 12 de febrero de 2020]. Disponible en: https://www.riptutorial.com/flask.

STALLINGS, W*. Sistemas Operativos. Aspectos internos y principios de diseño.* Madrid, España: Editorial Pearson Educación, S.A. 2005. Quinta edición. ISBN 978-84-205-5796-0.

SOMMERVILLE, IAN. *Ingeniería de Software*. Séptima. Madrid, España: Editorial Pearson Educación, S.A., 2005. ISBN 84-7829-074-5.

SOMMERVILLE, IAN. *INGENIERÍA DE SOFTWARE*. Novena. México: Editorial Pearson Educación, S.A., 2011. ISBN 978-607-32-0603-7.

THE JQUERY FOUNDATION. *What is jQuery?* [En línea] 2020. [Citado el: 14 de febrero de 2020]. Disponible en: https://jquery.com.

UCI. *Nova Servidores*. [En línea] 2019. [Citado el: 23 de septiembre de 2019]. Disponible en: https://www.uci.cu/investigacion-y-desarrollo/productos/nova/nova-servidores.

ULTIMATE SYSTEMS. *Hardinfo*. [En línea] 2018. [Citado el: 24 de noviembre de 2019]. Disponible en: https://www.usro.net/products/hardinfo/#readmore.

VAN ROSSUM, G.; WARSAW, B., et al. *Style Guide for Python Code*. [En línea] 2016. [Citado el: 5 de marzo de 2020]. Disponible en: https://legacy.python.org/dev/peps/pep-0008.

VISUAL PARADIGM. *Visual Paradigm*. [En línea] 2019. [Citado el: 8 de noviembre de 2019]. Disponible en: http://www.visual-paradigm.com.

WEBMIN. *Webmin*. [En línea] 2016. [Citado el: 27 de enero de 2020]. Disponible en: http://www.webmin.com/.

ZAKAS, NICHOLAS C. *Professional JavaScript for Web Developers*. Canadá: Editorial Wiley Publishing, Inc., 2005. ISBN 978-0-7645-7908-0.

ANEXOS

Anexo 1: Entrevista realizada al cliente y a especialistas del centro CESOL

Objetivo: Conocer el proceso actual de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0

Preguntas:

1. ¿Cuáles son los aspectos fundamentales a tener en cuenta cuando se realiza el proceso de visualización de los detalles técnicos del equipo?

2. ¿Cuáles son los pasos a seguir para realizar el proceso de visualización?

Anexo 2: Guía de observación para el proceso de visualización de los detalles técnicos de los recursos de hardware y software para Nova Servidores 7.0

Observador: Beatriz García Jiménez

Lugar: Departamento de Servicios Integrales en Migración Asesoría y Soporte (SIMAYS) que pertenece al centro CESOL.

Objetivo: Identificar los requisitos fundamentales para la realización del proceso de visualización de los detalles técnicos de los recursos de *hardware* y *software* para Nova Servidores 7.0

Preguntas:

1. ¿Cómo se realiza el proceso de visualización de los detalles técnicos del equipo?

2. ¿Cuáles son los pasos a seguir?

3. ¿Cuál es el resultado?

4. ¿Cómo se realiza el proceso en el caso en que se requiera la inspección de un grupo de equipos ubicados en diferentes localizaciones geográficas?

Anexo 3: Historia de Usuario

Tabla 13: RF.1 Mostrar detalles técnicos generales del equipo.

(Fuente: elaboración propia)

Número: HU-01	**Requisito:** Mostrar detalles técnicos generales del equipo.
Programador: Beatriz García Jiménez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 25 horas
Riesgo en Desarrollo: No aplica	**Tiempo Real:** 25 horas

Descripción: El usuario activa la opción “EQUIPO”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos generales del equipo.

La información se obtiene con la consulta de los métodos ubicados en la clase *Device()*:

- *_type()*, se visualiza el procesador.
- *_total_memory()*, se visualiza la memoria total.
- *_available_memory()*, se visualiza la memoria disponible.
- Con la fórmula memoria total - memoria disponible, se visualiza la memoria en uso.
- *_distribution()*, se visualiza la distribución del sistema operativo.
- *_architecture()*,se visualiza la arquitectura.
- *_user_name()*, se visualiza el nombre del usuario
- *_team_name()*, se visualiza el nombre del equipo.
- *_time()*, se visualiza la fecha / hora del instante de consulta.

Observaciones: La herramienta debe mostrar un gráfico dinámico (en tiempo real) con el porciento de uso de la *CPU* y un gráfico de pastel representando la memoria total y la memoria en uso.

Prototipo de interfaz de usuario:

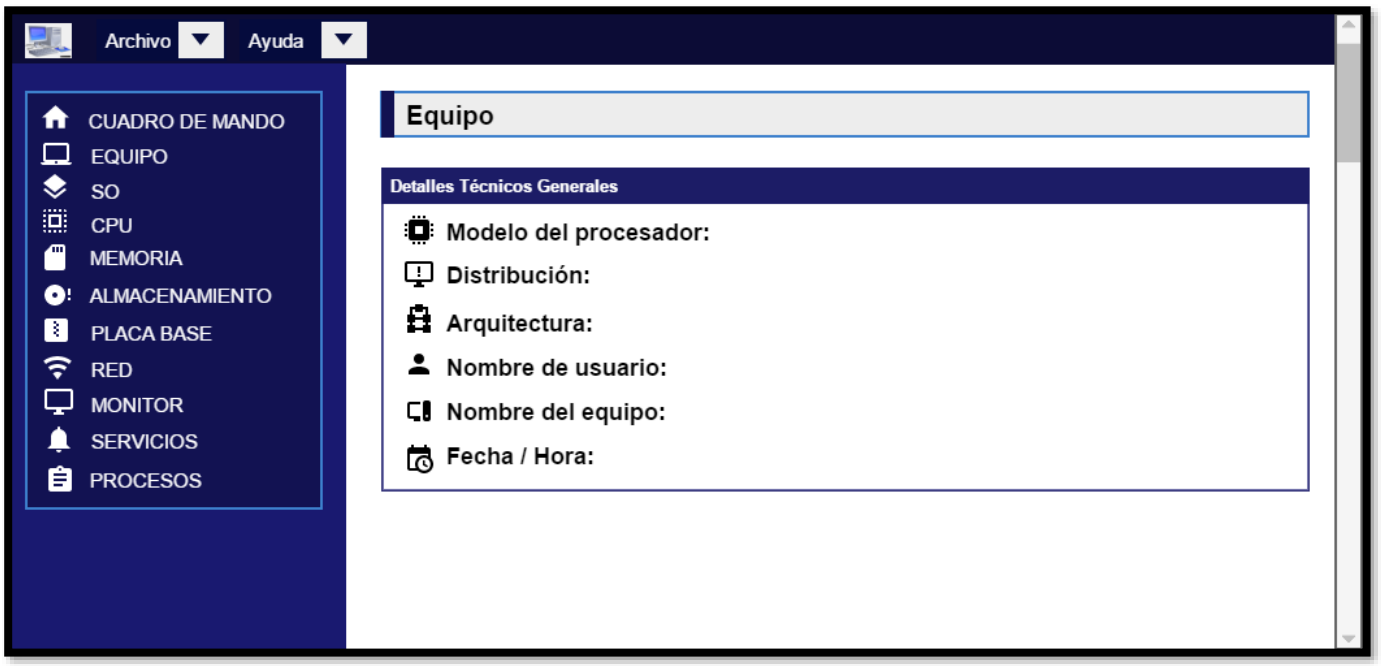


Figura 13: Prototipo de interfaz de usuario correspondiente al RF.1 Mostrar detalles técnicos generales del equipo.

(Fuente: elaboración propia)

Tabla 14: RF.3 Mostrar detalles técnicos de la placa base.

(Fuente: elaboración propia)

Número: HU-03	**Requisito:** Mostrar detalles técnicos de la placa base.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario activa la opción “PLACA BASE”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos de la placa base. Además, permite filtrar un determinado dispositivo por el nombre. La información se obtiene con la consulta de los métodos ubicados en la clase *Motherboard()*:	

- *_name()*, se visualiza el nombre de la placa base.
- *_manufacturer()*, se visualiza el fabricante de la placa base.
- *_pci_device()*, se visualizan los detalles de los dispositivos *PCI*.
- *_usb_device()*, se visualizan los detalles de los dispositivos *USB*.

Observaciones:

Prototipo de interfaz de usuario:

Figura 14: Prototipo de interfaz de usuario correspondiente al RF.3 Mostrar detalles técnicos de la placa base.

(Fuente: elaboración propia)

Tabla 15: RF.4 Mostrar detalles técnicos de la memoria.

(Fuente: elaboración propia)

Número: HU-04	**Requisito:** Mostrar detalles técnicos de la memoria.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1

Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “MEMORIA”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos de la memoria.

La información se obtiene con la consulta de los métodos ubicados en la clase *Memory()*:

Memoria de Acceso Aleatorio (*RAM*)

- *_total_memory()*, se visualiza la memoria total.
- *_available_memory()*, se visualiza la memoria disponible.
- Con la fórmula memoria total - memoria disponible, se visualiza la memoria en uso.

Memoria de intercambio (*Swap*)

- *_total_swap()*, se visualiza la memoria total.
- *_free_swap()*, se visualiza la memoria disponible.
- Con la fórmula memoria total - memoria disponible, se visualiza la memoria en uso.

Observaciones: Al posicionar el *mouse* sobre una porción del gráfico, la herramienta debe mostrar su porciento.

Prototipo de interfaz de usuario:

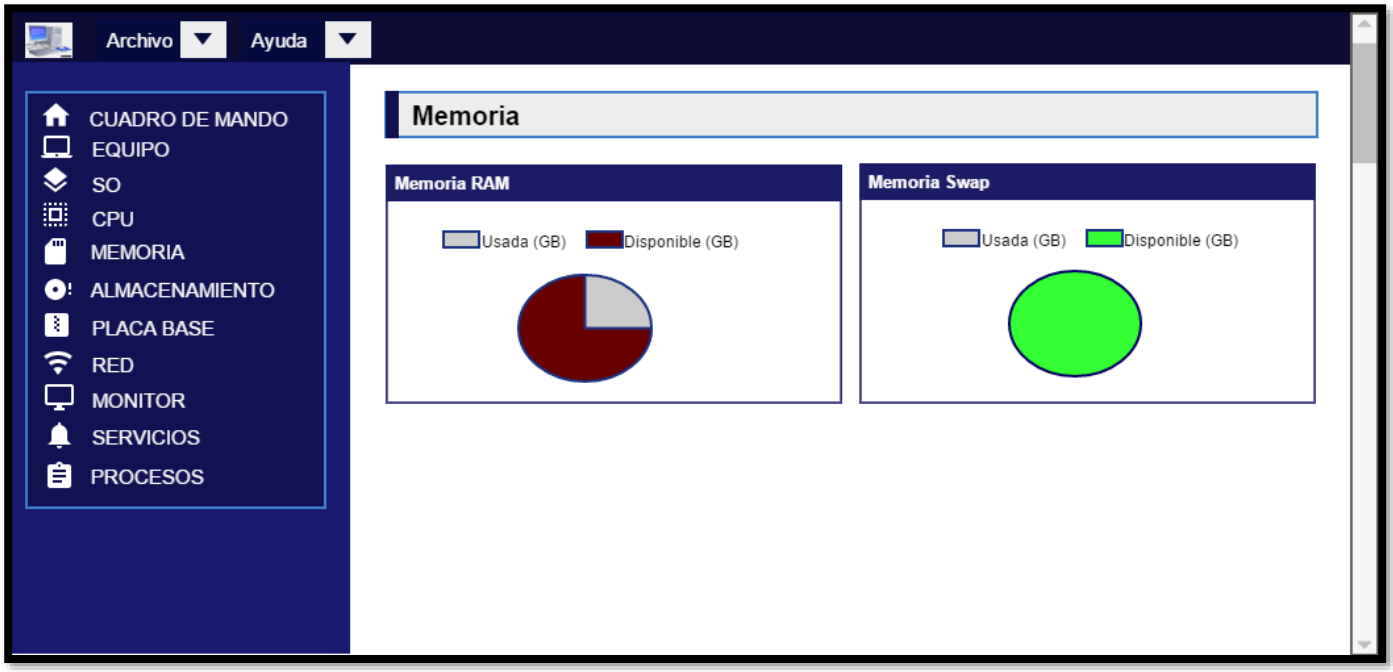


Figura 15: Prototipo de interfaz de usuario correspondiente al RF.4 Mostrar detalles técnicos de la memoria.

(Fuente: elaboración propia)

Tabla 16: RF.5 Mostrar detalles técnicos del almacenamiento.

(Fuente: elaboración propia)

Número: HU-05	**Requisito:** Mostrar detalles técnicos del almacenamiento.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario activa la opción “ALMACENAMIENTO”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos del almacenamiento. La información se obtiene con la consulta de los métodos ubicado en la clase *Storage()*:	

- *_system_info()*, se visualizan los detalles de las particiones del sistema, capacidad, uso, disponibilidad, porciento de uso y puntos de montaje.
- *_scsi_device()*,

Observaciones:

Prototipo de interfaz de usuario:

Figura 16: Prototipo de interfaz de usuario correspondiente al RF.5 Mostrar detalles técnicos del almacenamiento.

(Fuente: elaboración propia)

Tabla 17: RF.6 Mostrar detalles técnicos de la red.

(Fuente: elaboración propia)

Número: HU-06	**Requisito:** Mostrar detalles técnicos de la red.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas

Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “RED”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos de la red.

La información se obtiene con la consulta de los métodos ubicados en la clase *Network()*:

- *_manufacturer()*, se visualiza el fabricante.
- *_address()*, se visualiza la dirección ip.
- *_mask()*, se visualiza la máscara.
- *_broadcast()*, se visualiza la difusión.

Observaciones:

Prototipo de interfaz de usuario:

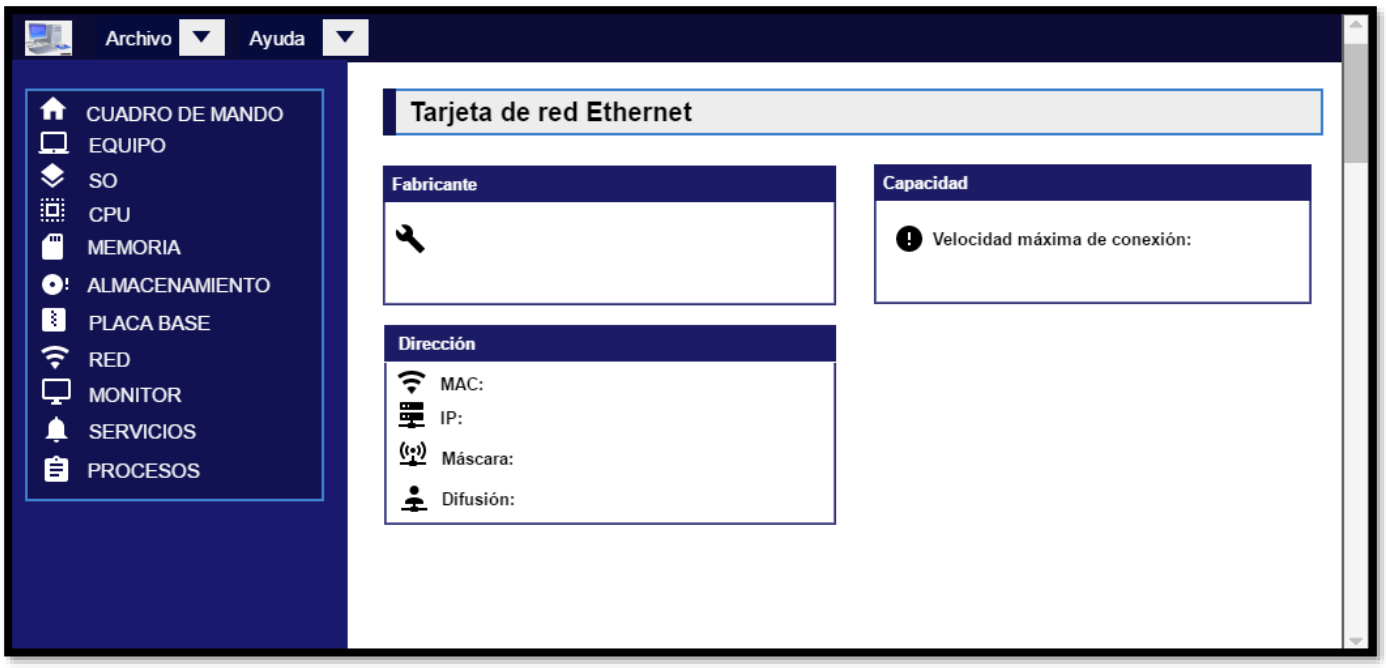


Figura 17: Prototipo de interfaz de usuario correspondiente al RF.6 Mostrar detalles técnicos de la red.

(Fuente: elaboración propia)

Tabla 18: RF.7 Mostrar detalles técnicos del sistema operativo.

(Fuente: elaboración propia)

Número: HU-07	**Requisito:** Mostrar detalles técnicos del sistema operativo.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “SO”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos del sistema operativo. Además, permite filtrar un determinado módulo por el nombre.

La información se obtiene con la consulta de los métodos ubicados en la clase *OperatingSystem()*:

- *_so_type()*, se visualiza la plataforma.
- *_distribution()*, se visualiza la distribución.
- *_version()*, se visualiza la versión del SO.
- *_architecture()*, se visualiza la arquitectura.
- *_kernel_version()*, se visualiza la versión del *kernel*.
- *_code_name()*, se visualiza el nombre clave de la distribución.
- *_build_date()*, se visualiza la fecha de compilación del *kernel*.

Observaciones:

Prototipo de interfaz de usuario:

Figura 18: Prototipo de interfaz de usuario correspondiente al RF.7 Mostrar detalles técnicos del sistema operativo.

(Fuente: elaboración propia)

Tabla 19: RF.8 Mostrar detalles técnicos del monitor.

(Fuente: elaboración propia)

Número: HU-08	**Requisito:** Mostrar detalles técnicos del monitor.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario activa la opción “MONITOR”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos del monitor. La información se obtiene con la consulta de los métodos ubicados en la clase *Display()*: • *_manufacturer()*, se visualiza el fabricante.	

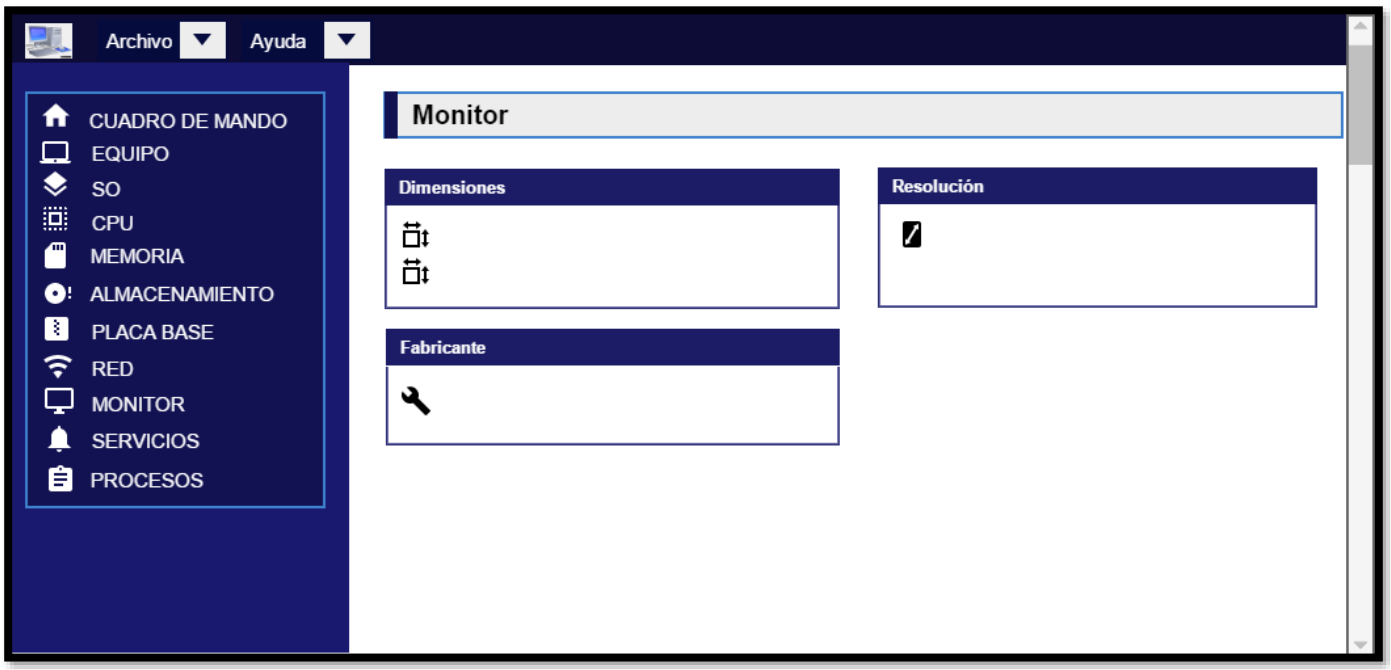
• *_dimensions()* y *_dimensions2(),* se visualizan las dimensiones. • *_resolution()*, se visualiza la resolución de la pantalla.
Observaciones:
Prototipo de interfaz de usuario:    *Figura 19: Prototipo de interfaz de usuario correspondiente al RF.8 Mostrar detalles técnicos del monitor.* *(Fuente: elaboración propia)*

Tabla 20: RF.9 Listar servicios.

(Fuente: elaboración propia)

Número: HU-09	**Requisito:** Listar servicios.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas

Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “SERVICIOS”, en el menú de navegación izquierdo, y la herramienta lista todos los servicios instalados en el equipo, mostrando el nombre y el estado. Además, permite filtrar un determinado servicio por el nombre.

La información se obtiene con la consulta de los métodos ubicados en la clase *Service()*:

- *_all_service()*, se listan todos los servicios instalados y se visualiza el estado de cada servicio (Activo o Inactivo).

Observaciones:

Prototipo de interfaz de usuario:

Figura 20: Prototipo de interfaz de usuario correspondiente al RF.9 Listar servicios.

(Fuente: elaboración propia)

Tabla 21: RF.10 Mostrar detalles técnicos de un servicio.

(Fuente: elaboración propia)

Número: HU-10	**Requisito:** Mostrar detalles técnicos de un servicio.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “SERVICIOS”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos de todos los servicios instalados.

La información se obtiene con la consulta de los métodos ubicados en la clase *Service()*:

- *details()*, se visualizan los datos de un determinado servicio: estado, PID (identificador) y registros (detalles del tráfico del servicio: datos, fecha, hora y nombre del equipo que ejecuta el servicio).

Observaciones: La herramienta debe mostrar una ventana que contiene los datos de un determinado servicio.

Prototipo de interfaz de usuario:

Figura 21: Prototipo de interfaz de usuario correspondiente al RF.10 Mostrar detalles técnicos de un servicio.

(Fuente: elaboración propia)

Tabla 22: RF.11 Listar procesos.

(Fuente: elaboración propia)

Número: HU-11	**Requisito:** Listar procesos.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario activa la opción “PROCESOS”, en el menú de navegación izquierdo, y la herramienta lista todos los procesos en ejecución. Además, permite filtrar un determinado proceso por el nombre. La información se obtiene con la consulta de los métodos ubicados en la clase *Process()*:	

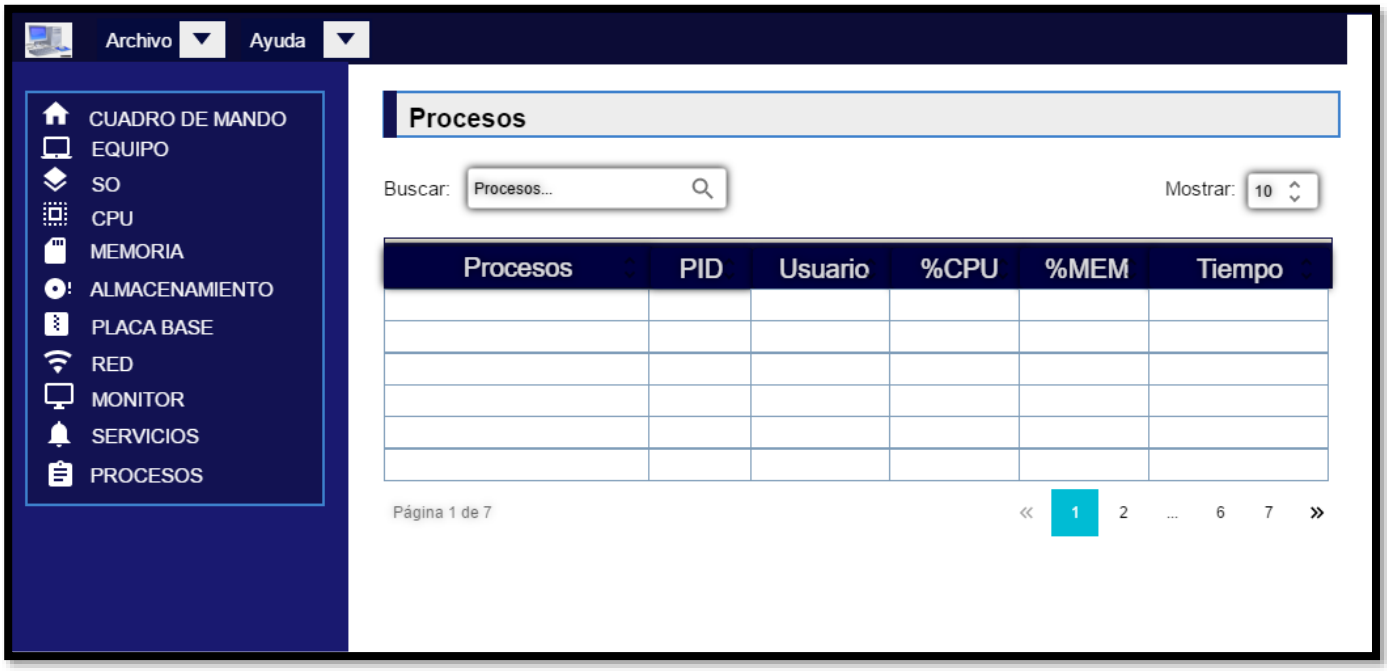
• *_execute_process(),* se listan los procesos en ejecución.
Observaciones:
Prototipo de interfaz de usuario: 

Figura 20: Prototipo de interfaz de usuario correspondiente al RF.11 Listar procesos.

(Fuente: elaboración propia)

Tabla 23: RF.12 Mostrar detalles técnicos de los procesos.

(Fuente: elaboración propia)

Número: HU-12	**Requisito:** Mostrar detalles técnicos de los procesos.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “PROCESOS”, en el menú de navegación izquierdo, y la herramienta muestra los detalles técnicos de todos los procesos en ejecución.

La información se obtiene con la consulta de los métodos ubicados en la clase *Process()*:

- *_execute_process(),* se listan los procesos en ejecución.

Observaciones:

Prototipo de interfaz de usuario:

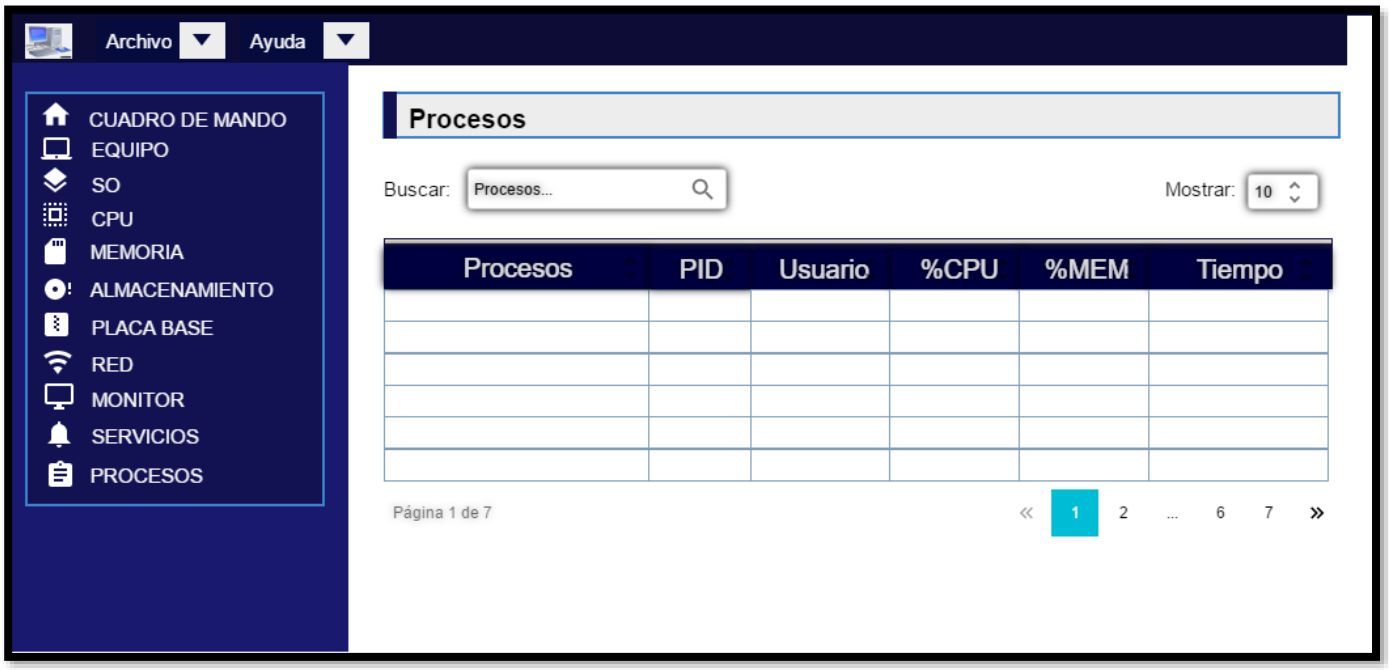


Figura 213: Prototipo de interfaz de usuario correspondiente al RF.12 Mostrar detalles técnicos de los procesos.

(Fuente: elaboración propia)

Tabla 24: RF.13 Mostrar detalles de la herramienta.

(Fuente: elaboración propia)

Número: HU-13	**Requisito:** Mostrar detalles de la herramienta.

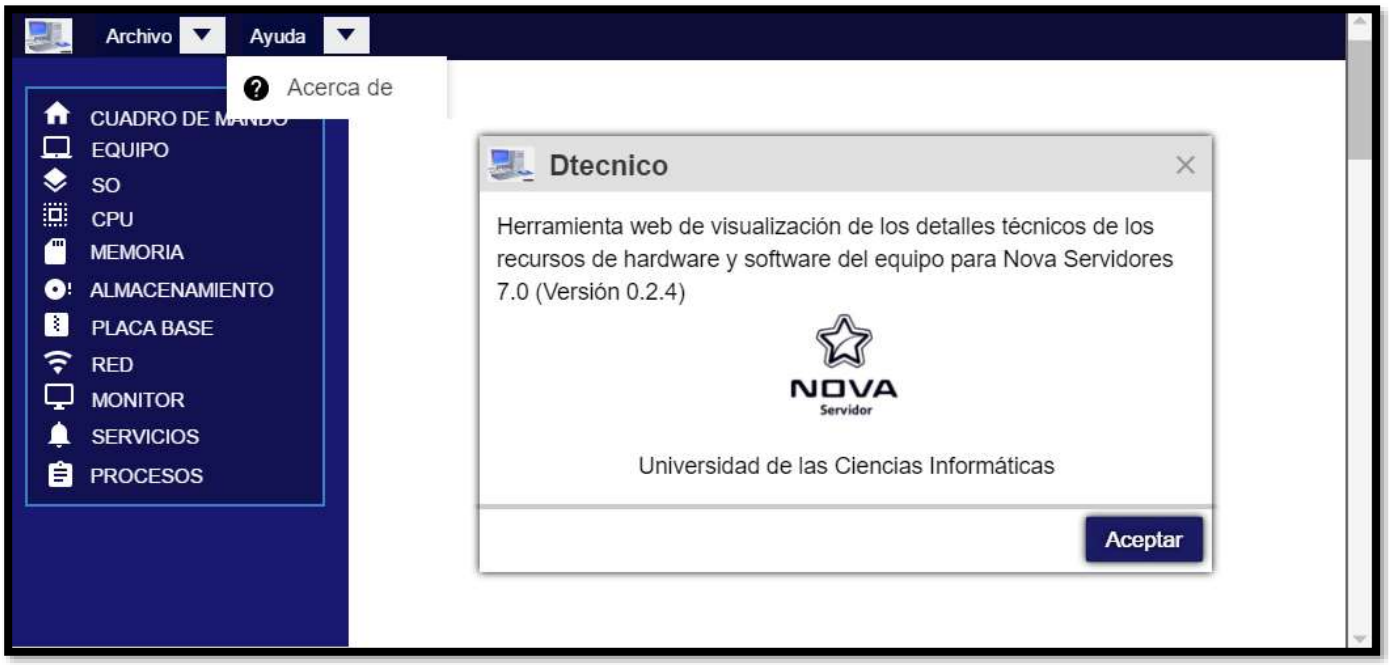
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario escoge la opción “Acerca de” para visualizar los detalles de la herramienta. Para ello, despliega la opción “Ayuda” en el menú de navegación y se abre una ventana que contiene: el nombre y la versión de la herramienta, el objetivo de su creación, a quién va destinada y el nombre del desarrollador.	
Observaciones: La herramienta debe mostrar una ventana con todos sus detalles.	
Prototipo de interfaz de usuario: 	

Figura 224: Prototipo de interfaz de usuario correspondiente al RF.13 Mostrar detalles de la herramienta.

(Fuente: elaboración propia)

Tabla 25: RF.14 Generar reporte sobre los detalles técnicos del equipo.

(Fuente: elaboración propia)

Número: HU-14	**Requisito:** Generar reporte sobre los detalles técnicos del equipo.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario despliega la opción “Archivo” en el menú de navegación, escoge la opción “Exportar” para generar reporte. Se abre una ventana que contiene dos posibilidades: 1. Generar reporte con todos los recursos. 2. Generar reporte personalizado. Los reportes se generan en formatos PDF o HTML.	
Observaciones: La herramienta debe notificar al usuario cuando la operación termine.	
Prototipo de interfaz de usuario:	

Figura 235: Prototipo de interfaz de usuario correspondiente al RF.14 Generar reporte sobre los detalles técnicos del equipo.

(Fuente: elaboración propia)

Tabla 26: RF.15 Imprimir reporte sobre los detalles técnicos del equipo.

(Fuente: elaboración propia)

Número: HU-15	**Requisito:** Imprimir reporte sobre los detalles técnicos del equipo.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario despliega la opción “Archivo” en el menú de navegación, escoge la opción “Imprimir”. Se abre una ventana que contiene una vista previa del documento, y presiona el botón “Imprimir” que se encuentra en extremo inferior derecho.	

Observaciones: La herramienta debe notificar al usuario cuando la operación termine.
Prototipo de interfaz de usuario:

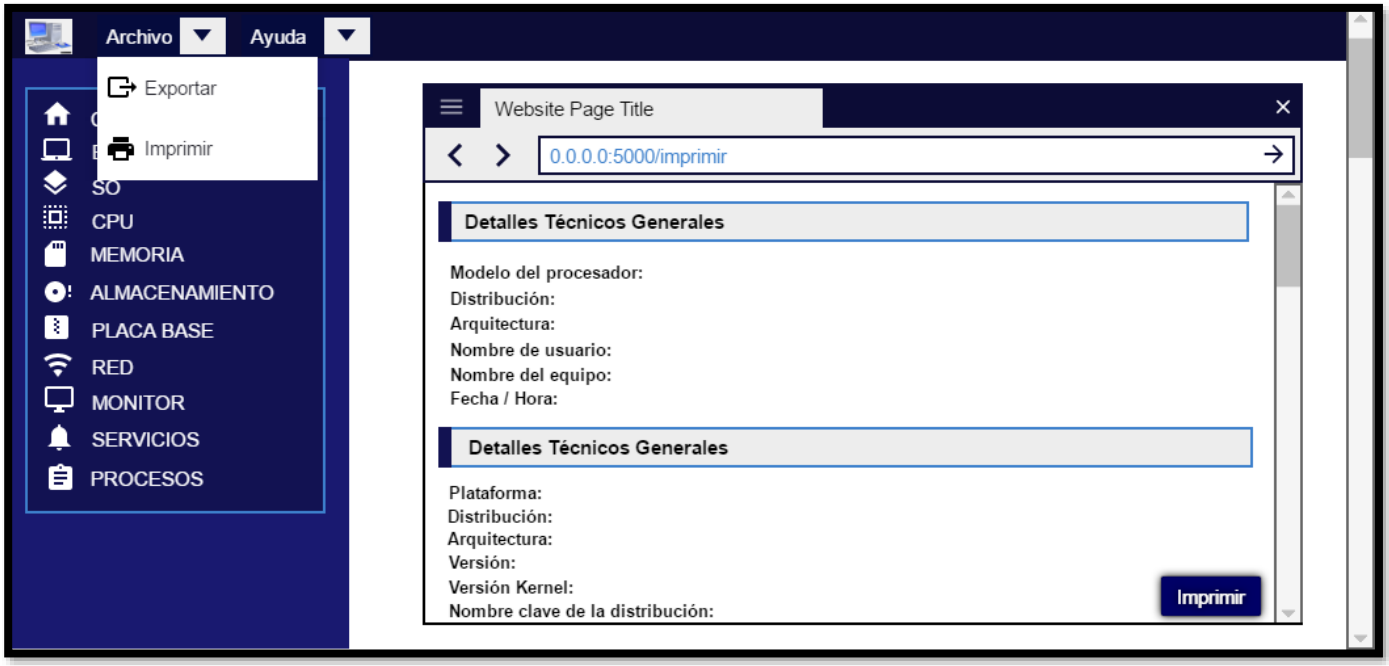


Figura 246: Prototipo de interfaz de usuario correspondiente al RF.15 Imprimir reporte sobre los detalles técnicos del equipo.

(Fuente: elaboración propia)

Tabla 27: RF.16 Permitir a los usuarios autenticarse en la herramienta.

(Fuente: elaboración propia)

Número: HU-16	**Requisito:** Permitir a los usuarios autenticarse en la herramienta.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas
Descripción: El usuario debe autenticarse para acceder a los detalles técnicos de los recursos de	

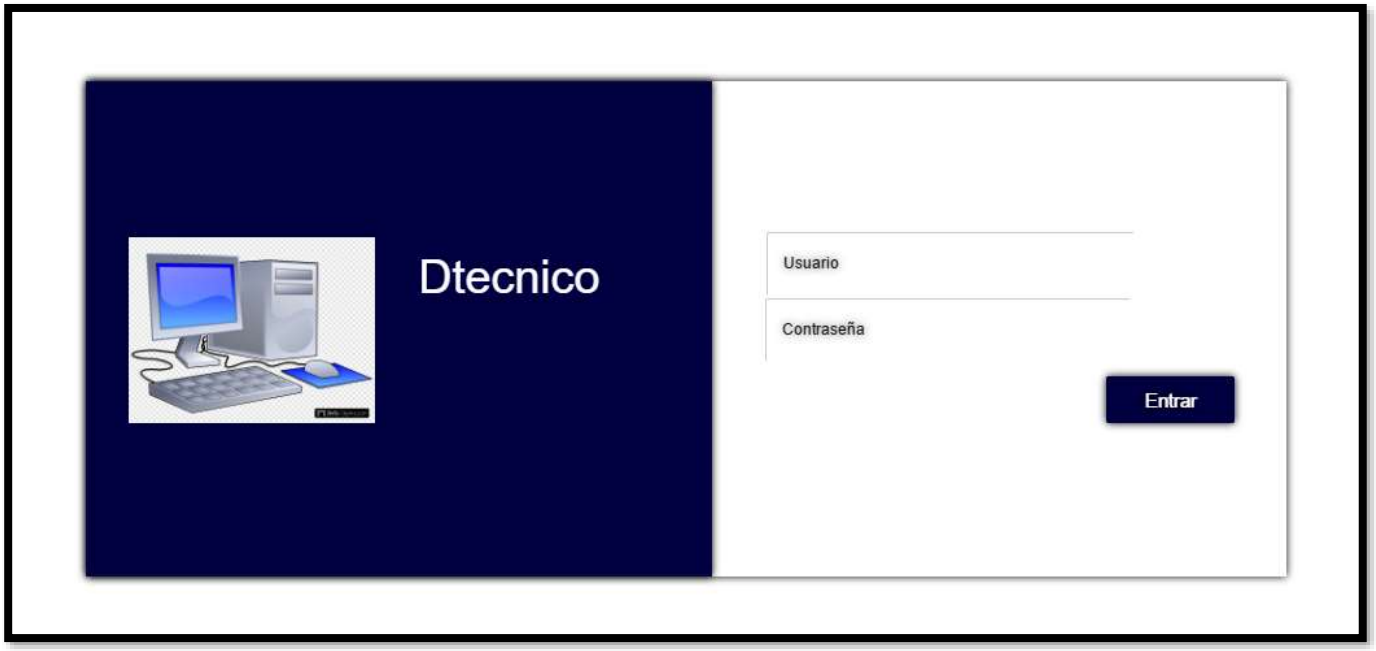
hardware y *software* del equipo. La información se obtiene con la consulta del método ubicado en la clase *Login()*: • *validate()*, se valida el usuario y la contraseña.
Observaciones:
Prototipo de interfaz de usuario:    *Figura 257: Prototipo de interfaz de usuario correspondiente al RF.16 Permitir a los usuarios autenticarse en la herramienta.* *(Fuente: elaboración propia)*

Tabla 28: RF.17 Mostrar cuadro de mando con algunos detalles técnicos del equipo.

(Fuente: elaboración propia)

Número: HU-17	**Requisito:** Mostrar cuadro de mando con algunos detalles técnicos del equipo.
Programador: Beatriz García Jiménez	**Iteración asignada:** 1

Prioridad: Alta	**Tiempo estimado:** 25 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 25 horas

Descripción: El usuario activa la opción “CUADRO DE MANDO”, en el menú de navegación izquierdo, o accede al icono de la parte superior izquierda; y la herramienta muestra de forma dinámica algunos detalles técnicos del equipo.

La información se obtiene con la consulta de los métodos ubicados en las clases *Cpu()* y *Memory()*:

- *_cpu_percent(),* se visualiza el porciento del uso de la *CPU*.

Memoria de Acceso Aleatorio (*RAM*)

- *_total_memory()*, se visualiza la memoria total.
- *_available_memory()*, se visualiza la memoria disponible.
- Con la fórmula memoria total - memoria disponible, se visualiza la memoria en uso.

Memoria de intercambio (*Swap*)

- *_total_swap()*, se visualiza la memoria total.
- *_free_swap()*, se visualiza la memoria disponible.
- Con la fórmula memoria total - memoria disponible, se visualiza la memoria en uso.

Observaciones: La herramienta debe mostrar algunos gráficos dinámicos (en tiempo real) representando:

- el promedio del porciento de uso de todos los *CPU*.
- el porciento de uso de la *CPU*.
- el porciento de uso de la *RAM*.
- la memoria total y la memoria en uso de la *RAM*, mediante un gráfico de pastel.
- la memoria total y la memoria en uso de la *Swap*, mediante un gráfico de pastel.

Prototipo de interfaz de usuario:

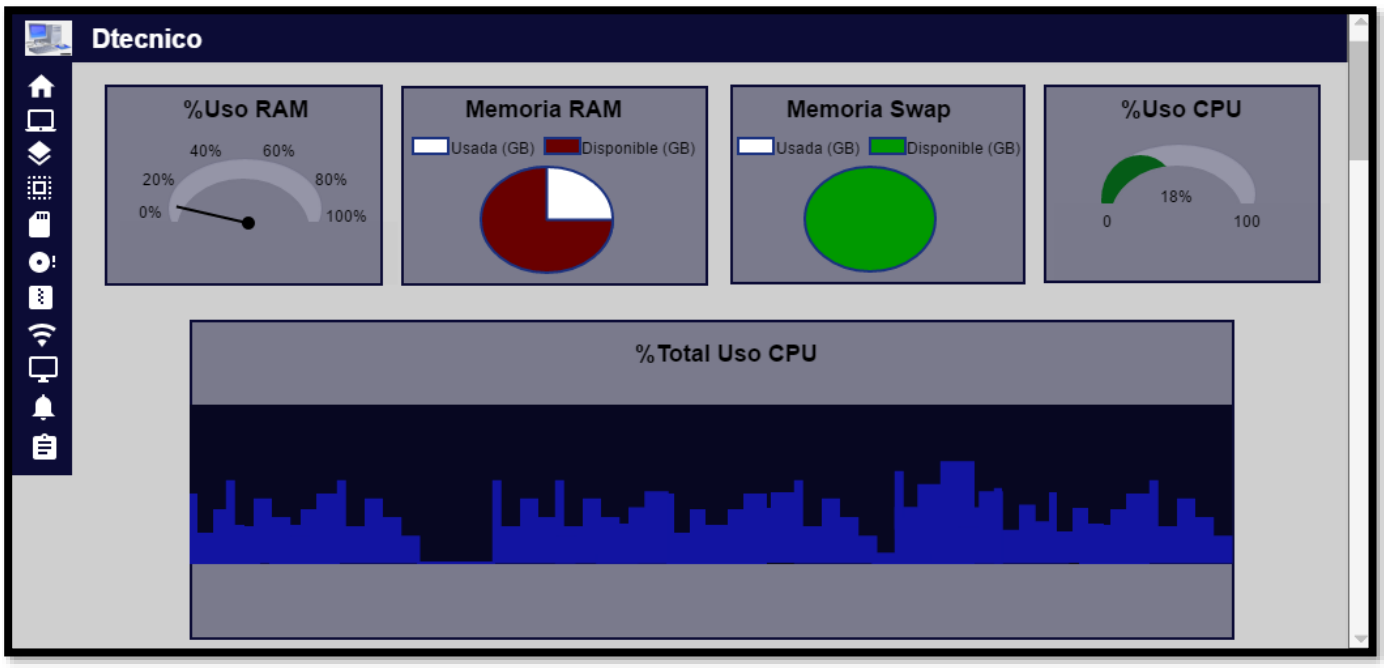


Figura 268: Prototipo de interfaz de usuario correspondiente al RF.17 Mostrar cuadro de mando con algunos detalles técnicos del equipo.

(Fuente: elaboración propia)