Universidad de las Ciencias Informáticas

Facultad 1

"Herramienta informática para la administración remota de tareas programadas en Nova 7.0"

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

Autora:

Sueyaile Toledo Ruiz

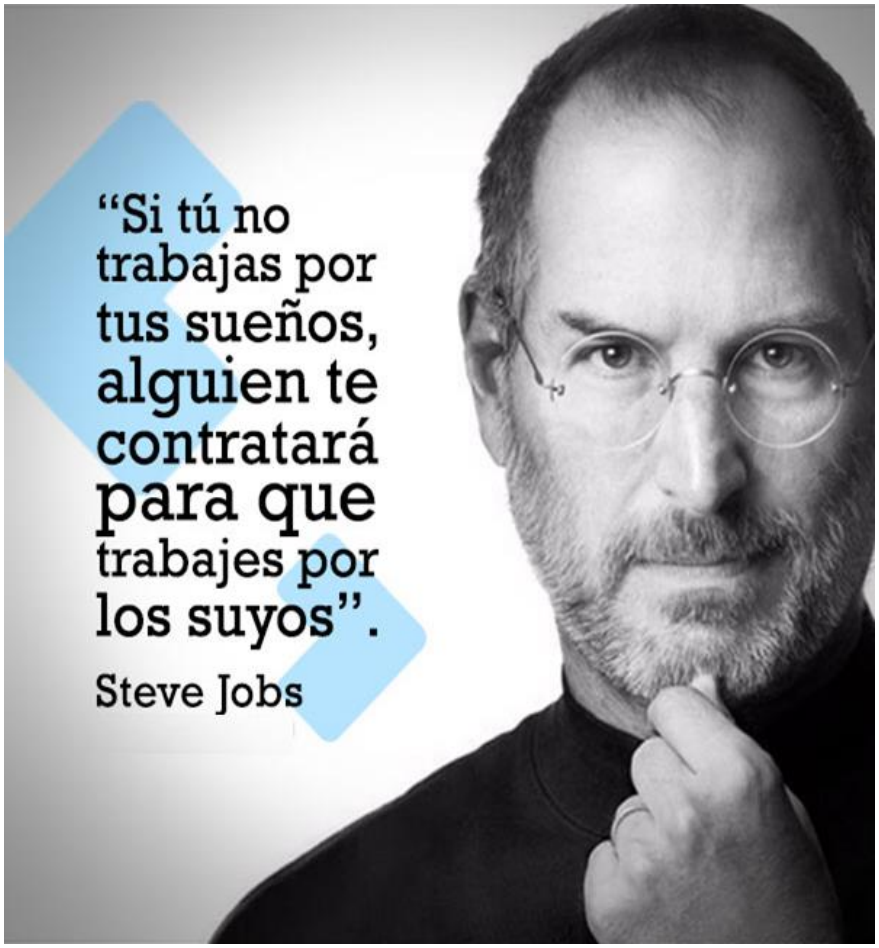
Tutores:

MSc. Anié Bermudez Peña

Ing. Enrique Muschett Cortina

La Habana, Octubre de 2020

"Año 62 de la Revolución"


“Si tú no trabajas por tus sueños, alguien te contratará para que trabajes por los suyos”.
Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaro que yo **Sueyaile Toledo Ruiz**, como autora principal del trabajo titulado **"Herramienta Informática para la administración remota de tareas programadas en Nova 7.0"** y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

Sueyaile Toledo Ruiz

Firma de la autora

_____

MSc. Anié Bermudez Peña

Firma del tutor

_____

Ing. Enrique Muschett Cortina

Firma de la tutora

DATOS DE CONTACTO

Autor: Sueyaile Toledo Ruiz

Correo electrónico: stoledo@estudiantes.uci.cu

Tutora: MSc. Anié Bermudez Peña

Especialidad de graduación: Ingeniera en Ciencias Informáticas, Máster en Gestión de Proyectos Informáticos. Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: abp@uci.cu

Tutor Enrique Muschett Cortina

Clasificación: Ingeniero en Ciencias Informáticas, Especialista B en Ciencias Informáticas del Centro de Software Libre. Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: emuschett@uci.cu

AGRADECIMIENTOS

Quisiera primero que todo darle las gracias a mi mamá por todo su cariño, por ser mi amiga mi compañera, mi padre. Le doy las gracias a Alberto por ser mi padre desde los cinco años de edad, a mi abuelo Guzmán y a mi tío Isbel por su cariño.

A mi abuela Liduvina por todo el cariño que una vez me dio en vida, y por cuidarme desde donde quiera que se encuentre.

A mis tutores Anié y Muschett, y por las horas dedicadas y a todas aquellas personas que de una forma u otra contribuyeron con su ayuda a que pudiera realizar con éxito este trabajo.

Agradezco a Ivaniet, por su paciencia y comprensión con cada una de mis dudas, a Yasiel P. Villazón por estar siempre ahí apoyándome con sus sugerencias en cuanto al desarrollo de la aplicación. A Nurisel por su voluntariedad cuando alguna duda surgía sobre metodología.

A Ponce por estar siempre ahí conteniéndome y aconsejándome, por su paciencia con cada uno de mis berrinches, por su cariño.

A Joel por compartir conmigo 3 años de esta experiencia que fue mi carrera y por ser mi apoyo y consejero en este tiempo.

A Aylin por ser mi amiga, por sus ideas locas, por cuidarme cuando estaba mal, por los buenos ratos compartidos.

A Indira por escucharme con paciencia cada vez que tengo un problema por ser mi apoyo en todo este tiempo.

A mis compañeros del aula con los que he compartido el transcurso de la carrera, por sus locuras y sus ideas.

A todos aquellos profesores que impartieron clases y me asistieron con sus conocimientos para ser hoy la persona que soy tanto profesionalmente, como socialmente.

Le agradezco a todas aquellas personas que, aunque no las mencione, estuvieron presentes en mi vida y me aportaron experiencias para ser la persona que soy hoy.

DEDICATORIA

A mi mamá, por ser mi apoyo, mí guía y por todo su esfuerzo y cariño para que siempre pudiera realizar mis sueños.

A mi padrastro Alberto, por ser mi padre y cuidarme desde los cinco años de edad.

A mi abuelo por todo su cariño.

A mi abuela Liduvina, que me cuida y me guía desde el cielo.

RESUMEN

En los últimos años el desarrollo de aplicaciones que facilitan el servicio de administración de tareas programadas en Linux ha ido creciendo paulatinamente, ejemplo de ello lo son las herramientas Webmin y Crontab-ui. Con el fin de asegurar la soberanía tecnológica e impulsar la innovación nacional, Cuba ha estado llevando a cabo la migración paulatina hacia Software Libre y las tecnologías de código abierto, proceso en el cual interviene con sus soluciones informáticas el Centro de Software Libre, perteneciente a la Universidad de las Ciencias Informáticas (UCI). Nova es una distribución cubana de GNU/Linux y una de estas soluciones, la cual permite realizar las acciones propias de cualquier sistema operativo. No obstante, el servicio de administración de tareas programadas en Nova se realiza de forma manual mediante la introducción de comandos en una interfaz de consola, lo cual se hace dificultoso para aquellos usuarios que no tengan dominio suficiente sobre los mismos. La presente investigación tuvo como objetivo implementar una herramienta informática para la administración remota de tareas programadas en Nova 7.0. Para su desarrollo se utilizó como guía en todo el proceso de desarrollo de software la metodología AUP ajustada para la UCI; así como diferentes tecnologías libres y la arquitectura N-Capas en su variante tres capas. Además, se realizaron pruebas para comprobar el correcto funcionamiento de la herramienta, demostrando que la misma cumple con todos los requisitos del cliente y un mejor manejo en la configuración de las tareas programadas.

Palabras clave: administración remota, Cron, Nova 7.0, tareas programadas.

ABSTRACT

In recent years the development of applications that facilitate the task management service scheduled in Linux has been growing gradually, an example of which are the Webmin and Crontab-ui tools. In order to ensure technological sovereignty and boost national innovation, Cuba has been carrying out the gradual migration to Free Software and open source technologies, a process in which the Free Software Center, which belongs to the University of Computer Science (UCI). Nova is a Cuban distribution of GNU / Linux and one of these solutions, which allows you to perform the actions of any operating system. However, the task management service programmed in Nova is done manually by entering commands in a console interface, which makes it difficult for those users who do not have sufficient mastery over them. The purpose of this research was to implement a computer tool for the remote management of scheduled tasks in Nova 7.0, for its development the AUP variation methodology for the UCI was used as a guide in the entire software development process; different free technologies and the N-Layers architecture was used in its three-layer variant. In addition, tests were carried out to verify the correct functioning of the tool, demonstrating that it complies with all customer requirements.

Keywords: *Cron, Nova 7.0, remote management, scheduled tasks.*

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con el creciente uso de las Tecnologías de la Información y las Comunicaciones (TIC), la información fluye sin fronteras, pues cada día es necesario que grandes volúmenes de datos puedan ser almacenados y gestionados, y Cuba no se encuentra excepto de ello (1). En la última década en nuestro país se ha estado realizando un arduo proceso de informatización de la sociedad, el cual ha ido encaminado y enfocado a los pilares fundamentales del proceso revolucionario: la salud, la educación y la cultura. Con el fin de asegurar la soberanía tecnológica e impulsar la innovación nacional, se lleva a cabo la migración paulatina hacia Software Libre y las aplicaciones de código abierto, debido a la necesidad que esto representa para la sociedad cubana en los aspectos políticos, económicos, sociales y tecnológicos (2).

Con el objetivo de apoyar la informatización del país, en la Universidad de las Ciencias Informáticas (UCI), fue creado el Centro de Software Libre (CESOL), líder en los procesos de migración hacia tecnologías libres y de código abierto en Cuba. El mismo realiza acciones como el desarrollo de aplicaciones de escritorio, herramientas de soporte a dicho proceso y la migración de los servicios telemáticos, los cuales utilizan generalmente técnicas de procesamiento de la información de forma remota. Dentro de dichos servicios telemáticos se encuentra el servicio de planificación de tareas (3).

El servicio de planificación de tareas permite realizar, con cierta periodicidad, determinadas acciones o tareas, algunas de ellas internas del Sistema Operativo, otras definidas por el administrador, e incluso algunas definidas por un usuario con los privilegios adecuados. La necesidad de este tipo de herramientas viene dada, en principio, tanto por el funcionamiento interno del sistema operativo como por la necesidad del administrador de garantizar un aceptable funcionamiento del sistema. En el caso de Linux, la planificación se realiza a través de Cron, que no es más que un servicio que administra los procesos que se ejecutan en segundo plano y en intervalos programados por los usuarios, resultando de gran utilidad para los usuarios que interactúan con este, debido a que favorece la gestión de tareas de forma desatendida (3).

El desarrollo de herramientas que ofrecen este servicio en Linux se ha ido incrementando progresivamente. Las tareas programadas permiten realizar acciones como copias de seguridad periódicas o controlar el sistema y ejecutar scripts personalizados, entre otras tareas. Existen diversas aplicaciones que implementan este servicio y se encuentran basadas en tecnologías libres, tomando como ejemplo: Webmin (4), Crontab-

ui (5) y Nova NAS (6). Estas herramientas brindan al usuario la posibilidad de crear, editar y eliminar tareas planificadas, mediante una interfaz gráfica amigable, sin necesidad de tener que acceder a la consola para su configuración. Las mismas poseen desventajas como: no brindan la posibilidad de administrar remotamente las tareas programadas o no abarcan todos los parámetros del cron.

Actualmente el servicio de administración de tareas programadas del sistema operativo Nova[1] se realiza de forma manual mediante la introducción de comandos a través de una interfaz de consola, la cual resulta poco amigable e intuitiva para los usuarios, provocando que el trabajo con la misma resulte complejo. Además, esto trae como consecuencia errores humanos en la configuración de las tareas, un mayor empleo de tiempo para aquellos usuarios que no dominen los comandos y el funcionamiento de los mismos; de igual manera ocurre con aquellos administradores que tienen que acceder de máquina en máquina cada vez que tenga que configurar las tareas a un conjunto de computadoras.

Para dar solución a la situación problemática antes mencionada se presenta como **problema a resolver:** ¿Cómo mejorar el proceso de administración remota de tareas programadas en la distribución cubana GNU/Linux Nova 7.0?

Por lo que se establece como **objetivo general:** desarrollar una herramienta informática para mejorar el proceso de administración remota de tareas programadas en Nova 7.0.

El **objeto de estudio de la investigación** se centra en el proceso de administración de tareas programadas en GNU/Linux enmarcando como **campo de acción** el proceso de administración de tareas programadas en Nova 7.0.

Para dar cumplimiento al objetivo general se determinaron los siguientes **objetivos específicos:**

- ✓ Elaborar el marco teórico de la propuesta de solución teniendo en cuenta las tecnologías asociadas a la administración remota de tareas programadas en GNU/Linux.
- ✓ Diseñar la herramienta para la administración remota de tareas programadas Nova 7.0.
- ✓ Implementar la herramienta para la administración remota de tareas programadas en Nova 7.0.
- ✓ Evaluar el correcto funcionamiento de la herramienta aplicando pruebas de software y encuestas de satisfacción a expertos, para garantizar la calidad de la misma.

[1] NOVA: distribución cubana de GNU/Linux.

Hipótesis: El desarrollo de una herramienta informática permitirá mejorar el proceso de administración remota de tareas programadas en Nova 7.0.

Variable independiente: Herramienta informática.

Variable dependiente: Proceso de administración remota de tareas programadas en Nova 7.0.

La Tabla 1 representa la operacionalización de la variable independiente la cual fue definida mediante la norma ISO-25010.

Tabla 1.Operacionalización de la variable dependiente.

Fuente: elaboración propia.

Dimensiones	Indicadores	Unidad de medida
Adecuación funcional	Completitud funcional	¿La herramienta le permite al usuario configurar una tarea programada de forma correcta? Sí No
Seguridad	Confidencialidad	¿La herramienta permite que solo el o los usuarios autorizados puedan configurar las tareas remotamente? Sí No

La operacionalización de la variable independiente (ver Tabla 2), se realiza mediante criterios definidos de acuerdo a las necesidades identificadas en el contexto del negocio y su evaluación se basa en el criterio de expertos.

Tabla 2. Operacionalización de la variable independiente.

Fuente: elaboración propia.

Dimensiones	Indicadores	Unidad de medida
Tiempo de configuración de las tareas programadas	Disminución del tiempo en la configuración de las tareas programadas	¿La configuración de las tareas se realiza mucho más rápido? Sí No
Errores humanos en la configuración de las tareas programadas	Disminución de errores en la configuración de las tareas programadas	¿Se cometen menos errores a la hora de realizar la configuración de las tareas? Sí No

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

Métodos teóricos:

- ✓ **Analítico-Sintético:** permite realizar un análisis del desarrollo y funcionamiento de las herramientas de administración remota de tareas programadas, para así sintetizar en sus principales aspectos, funcionalidades y características.
- ✓ **Histórico-Lógico:** permite hacer un análisis completo de la evolución de las diferentes herramientas de administración de tareas programadas existentes, seleccionando las más utilizadas, para así obtener las características e información necesaria para el diseño de la propuesta de solución.

Métodos empíricos:

- ✓ **Observación:** se realiza con el objetivo de constatar el estado actual de la problemática mediante la observación al proceso de seguimiento de la administración de tareas programadas en entornos libres.
- ✓ **Entrevista:** se realizan entrevistas al cliente y a especialistas del Centro CESOL para obtener la información necesaria referente a los requisitos de la herramienta a desarrollar.
- ✓ **Encuesta:** se realiza con el objetivo de recopilar información en cuanto al nivel de satisfacción de la propuesta de solución.

La estructura del documento de investigación consta de resumen, introducción, tres capítulos, conclusiones generales, referencias bibliográficas y anexos. La estructura de los capítulos se define a continuación:

Capítulo1. Fundamentación teórica sobre el proceso de administración de tareas programadas en GNU/Linux:

Se realiza un estudio de las herramientas de administración de tareas programadas existentes en GNU/Linux, para conocer sus características y obtener información acerca de cómo desarrollar una nueva herramienta basándose en las mismas. Se detallan los principales conceptos asociados, la metodología de desarrollo de software, las tecnologías, lenguajes y herramientas para el modelado de la solución.

Capítulo2. Análisis y diseño de la herramienta informática para la administración remota de tareas programadas en Nova 7.0:

Se presenta una propuesta de solución al problema planteado, son definidos los requisitos funcionales y no funcionales de la herramienta. Luego se realiza la descripción de los requisitos funcionales mediante las

Historias de Usuario, los prototipos de interfaz y los Casos de Pruebas. Finalmente se detalla el diseño arquitectónico y los patrones utilizados para la herramienta a desarrollar.

Capítulo3. Construcción y validación de la herramienta informática para la administración remota de tareas programadas en Nova7.0:

Se implementa la solución propuesta, se plantean los estándares de codificación y el diagrama de despliegue. Además, se confecciona el plan de pruebas y se realizan las pruebas necesarias para verificar que las funcionales implementadas dan solución a los requisitos planteados. Finalmente se realiza un análisis crítico sobre la importancia y aporte de la herramienta para la administración remota de tareas programadas en Nova 7.0.

CAPÍTULO I: Fundamentación teórica sobre el proceso de administración de tareas programadas en GNU/LINUX

1.1 Introducción

El presente capítulo contiene la fundamentación teórica de la investigación, además se realiza un estudio de las diferentes herramientas informáticas existentes para el servicio de administración de tareas programadas en GNU/Linux. Se definen los conceptos relacionados con dicho servicio y se define la metodología de desarrollo a utilizar para la elaboración de la propuesta de solución. Además, se describen las características del lenguaje de programación y otras herramientas empleadas en el desarrollo de la herramienta para la administración remota de tareas programadas en Nova 7.0.

1.2 Principales conceptos

Administración remota: se considera administración remota a la funcionalidad de algunos programas que permiten realizar ciertos tipos de acciones desde un equipo local y que las mismas se ejecuten en otro equipo remoto (7).

Tareas programadas: son aquellas que el usuario puede configurar para que cualquier proceso, programa o archivo se ejecute en el tiempo o fecha que se desee. En GNU/Linux las tareas pueden configurarse para ejecutarse de forma automática en un período de tiempo concreto y en las fechas indicadas. Un administrador del sistema puede utilizar las tareas programadas para realizar copias de seguridad periódicas o controlar el sistema y ejecutar scripts personalizados, entre otras tareas (3).

Cron: los *Cron Jobs* son el medio que permite a los usuarios de Linux/Unix establecer automáticamente instrucciones o scripts a una hora o fecha específica que normalmente es usado para comandos con tareas repetitivas administrativas, como respaldos, procesos regulatorios de información, entre otros (8).

Crontab: es un archivo de texto. Posee una lista con todos los scripts a ejecutar. Generalmente cada usuario del sistema posee su propio fichero *Crontab*. Se considera que el ubicado en la carpeta ***etc*** pertenece al usuario *root* (9).

1.3 Caracterización del proceso de administración de tareas programadas

La administración de tareas programadas en Linux se realiza mediante el programa *Cron*, el cual posee un editor de tareas llamados *crontab*. *Cron* permite programar una tarea (por ejemplo, un comando, programa o *Shell script*) para ejecutarlos periódicamente o una única vez. Se pueden programar las tareas de forma horaria, diaria, semanal, mensual, a una fecha y hora determinada (10). *Cron* es un demonio[2] lo que significa que solo requiere ser iniciado una vez, generalmente con el mismo arranque del sistema. El servicio de *Cron* se llama *crond*. En la mayoría de las distribuciones el servicio se instala automáticamente y queda iniciado desde el arranque del sistema (11).

La configuración del funcionamiento de *Cron* se encuentra dentro del directorio */etc*. Los ficheros más importantes implicados en el funcionamiento del servicio *Cron* son:

- ✓ El propio demonio de funcionamiento: *cron.d*
- ✓ El fichero de configuración (disponible para *root*): */etc/crontab*
- ✓ El fichero de inicio y parada del demonio: */etc/init.d/cron*
- ✓ La orden para la programación de tareas (disponible para los usuarios con suficientes privilegios): *crontab*
- ✓ El sistema de informes (logs) típico de los sistemas GNU/Linux: */var/log/Cron*

Para poder conocer el estado del servicio, así como iniciar o detener el mismo se deben ejecutar las órdenes correspondientes:

- ✓ Estado del demonio *Cron*: *#/etc/init.d/Cron status* o *service Cron status*
- ✓ Parar el demonio *Cron*: *# /etc/init.d/Cron stop* o *service Cron stop*
- ✓ Arranque del demonio *Cron*: *# /etc/init.d/Cron start* o *service Cron start* (12).

Hay al menos dos maneras distintas de usar *Cron*; la primera es en el directorio *etc*, donde se encuentran los siguientes directorios. De forma predeterminada, el paquete *Cron* incluye algunas tareas programadas que ejecutan:

- ✓ */etc/Cron.hourly/* una vez por hora;

[2] Demonio: es un servicio que se ejecuta de forma automática cuando se cumplan ciertas condiciones que lo desencadenan.

- ✓ */etc/Cron.daily/* una vez por día;
- ✓ */etc/Cron.weekly/* una vez por semana;
- ✓ */etc/Cron.monthly/* una vez por mes.

Si se coloca un archivo tipo script en cualquiera de estos directorios, entonces el script se ejecutará cada hora, cada día, cada semana o cada mes, dependiendo del directorio.

Como segundo modo de ejecutar o usar *Cron* es a través de manipular directamente el archivo */etc/crontab*. En la instalación por defecto de varias distribuciones de Linux, este archivo contiene varios parámetros como *shell, path, mail to y home*, los cuales aparecen especificados al inicio del archivo *crontab* (11).Cada usuario puede tener su propio archivo *Crontab*, de hecho, el */etc/Crontab* es el archivo *Crontab* del usuario *root,* cuando los usuarios normales (e incluso *root*) desean generar su propio archivo de *Crontab,* entonces se utilizará el comando *Crontab* (3).

```
#> cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

En la Tabla 3 se muestra una descripción de los primeros cinco campos que forman parte del archivo *crontab*.

Tabla 3. Descripción de las cinco primeras líneas del archivo *crontab*.

Fuente: elaboración propia.

Campo	**Descripción**
SHELL	Es el intérprete de órdenes para un sistema operativo bajo el cual se ejecuta el *cron*. Si no se especifica, se tomará por defecto el indicado en la línea */etc/passwd* correspondiente al usuario que esté ejecutando *cron*.

PATH	Contiene o indica la ruta a los directorios en los cuales *cron* buscará el comando a ejecutar. Este *path* es distinto al *path* global del sistema o del usuario.
MAIL TO	Es a quien se le envía la salida del comando (si es que este tiene alguna salida). *Cron* enviará un correo a quien se especifique en esta variable, es decir, debe ser un usuario válido del sistema o de algún otro sistema. Si no se especifica, entonces *cron* enviará el correo al usuario propietario del comando que se ejecuta.
HOME	Es el directorio raíz o principal del comando *cron*, si no se indica entonces, la raíz será la que se indique en el archivo */etc/passwd* correspondiente al usuario que ejecuta *cron.*
run-parts	Es un *script* que hace ejecutar los comandos del directorio especificado.

Luego de ser indicado estos parámetros vienen las líneas que ejecutan las tareas programadas propiamente. Los campos que forman estas líneas son 7 y están formados de la siguiente manera: minuto, hora, día del mes, mes, día de la semana, usuario y el comando.

En la Tabla 4 se muestra una descripción de los campos que forman parte del archivo *crontab*.

Tabla 4. Descripción de los campos que forman parte del archivo *crontab*.

Fuente: *(11)*.

Campo	Descripción
Minuto	Controla el minuto en que el comando será ejecutado, este valor debe de estar entre 0 y 59.
Hora	Controla la hora en que el comando será ejecutado, se especifica en un formato de 24 horas, los valores deben estar entre 0 y 23, 0 es medianoche.
Día del mes	Día del mes en que se quiere ejecutar el comando. Por ejemplo, se indicaría 20, para ejecutar el comando el día 20 del mes.
Mes	Mes en que el comando se ejecutará, puede ser indicado numéricamente (1-12), o por el nombre del mes en inglés, solo las tres primeras letras.
Día de la semana	Día de la semana en que se ejecutará el comando, puede ser numérico (0-7) o por el nombre del día en inglés, solo las tres primeras letras. (0 y 7 = domingo)
Usuario	Usuario que ejecuta el comando.

Comando	Comando, script o programa que se desea ejecutar. Este campo puede contener múltiples palabras y espacios.

Cron admite otros valores como # que se coloca al inicio de cada línea para indicar que es un comentario. Un asterisco * como valor en los primeros cinco campos, indicará inicio-fin del campo, es decir todo. Por ejemplo, un * en el campo de minuto indicará todos los minutos. También es posible especificar listas en los campos. Las listas pueden estar en la forma de 1 2 3 4 o en la forma de 1-4 que sería lo mismo. *Cron,* de igual manera soporta incrementos en las listas, que se indican de la siguiente manera: valor o lista / incremento.

Ejemplos de funcionamiento del *cron*:

0 22 * * * *root /usr/respaldodiario.sh*

0 23 * * 5 *root /usr/respaldosemanal.sh*

0 8,20 * * * sueyaile mail -s "sistema funcionando" stoledo@estudiantes.uci.cu

Las dos primeras líneas las ejecuta el usuario *root*. La primera línea ejecuta a las 10 de la noche de todos los días el script que genera un respaldo diario. La segunda ejecuta a las 11 de la noche de todos los viernes un script que genera un respaldo semanal. La tercera línea la ejecuta el usuario *sueyaile* y se ejecutaría a las 8 de la mañana y 8 de la noche de todos los días y el comando es enviar un correo a la cuenta *stoledo@estudiantes.uci.cu* con el asunto "sistema funcionando", una manera de que un administrador esté enterado de que un sistema remoto está activo en las horas indicadas, sino recibe un correo en esas horas, algo anda mal.

Creación, visualización, edición y eliminación del archivo *Crontab*

Para la creación de un archivo *Crontab* la forma más sencilla es utilizando el comando *crontab –e*. Este comando hace un llamado al editor de texto que se encuentra predeterminado para el entorno del sistema en el cual se está trabajando. Al ser creado un archivo *crontab,* este se almacenará por defecto en el directorio */var/spool/cron/crontabs* y recibirá el nombre del usuario que lo creó. Se puede crear, visualizar, modificar y eliminar un archivo *Crontab* para otro usuario, o para *root*, si se tienen privilegios de superusuario, lo que significa que el usuario tiene todos los permisos. Luego de ser creado el archivo se

añaden las líneas de comando descritas en la Tabla 4. Después se verifican los cambios en el archivo con el comando *crontab –l* [nombre_de_usuario]. Para la creación de un archivo *Crontab* para otro usuario se utiliza el comando *crontab –e* [nombre_de_usuario], donde nombre_de_usuario sería el usuario al cual se le está creando el archivo. Para verificar que para un usuario existe un archivo *crontab*, se utiliza el comando *ls –l* en el directorio */var/spool/cron/crontabs*. Cuando se elimina un archivo *crontab* se emplea el comando *crontab –r*, a este comando se le especifica el archivo *crontab* que se quiere eliminar, de lo contrario eliminará el archivo *crontab* al usuario que se encuentre trabajando en él (3)

Control del acceso al comando *Crontab*

Se puede controlar el acceso al comando *crontab*, se realiza mediante dos archivos en el directorio */etc/cron.d: cron.deny y cron.allow*. Estos archivos permiten que solo los usuarios especificados realicen tareas de comando *crontab,* como crear, editar, visualizar o eliminar sus propios archivos *crontab*. Los archivos *cron.deny y cron.allow* constan de una lista de nombres de usuario (un nombre de usuario por línea). Los archivos de control de acceso funcionan de manera conjunta como se indica a continuación:

- ✓ Si *cron.allow* existe, solo los usuarios indicados en este archivo pueden crear, editar, visualizar o eliminar archivos *crontab*.
- ✓ Si *cron.allow* no existe, todos los usuarios pueden ejecutar archivos *crontab,* excepto los usuarios indicados en *cron.deny.*
- ✓ Si *cron.allow* y *cron.deny* no existen, se necesitan privilegios de superusuario para ejecutar el comando *crontab*.

Los privilegios de superusuario son necesarios para editar o crear los archivos *cron. deny* y *cron.allow* (3).

1.4 Aplicaciones informáticas para la administración de tareas programadas en GNU/Linux

Actualmente existen varias herramientas cuyo principal objetivo es facilitar y optimizar la administración de tareas programadas en Linux. A continuación, se exponen las distintas soluciones estudiadas en el ámbito internacional y nacional.

1.4.1 Crontab-ui

Crontab-ui es una GUI (del inglés *Graphical User Interface* o Interfaz Gráfica de Usuario) basada en la web que no solo facilita la edición del *crontab*, sino que también le permite trabajar a través de un navegador

web desde cualquier máquina de la red. Además, brinda la posibilidad que todos los servidores Linux sin GUI también pueden beneficiarse de esto. *Crontab-ui* posee una interfaz mucho más amigable e interactiva que *Cron*, pero aún posee deficiencias. En primer lugar, al crear un nuevo trabajo, aún debe comprender qué valor numérico representan las entradas para Minuto, Hora, Día, Mes, Semana. El minuto, la hora y la semana concedidos deben explicarse por sí mismos (5).

1.4.2 Webmin

Webmin es una herramienta de configuración de sistemas Linux accesible vía web con una interfaz gráfica. Se caracteriza por ser modular, lo que significa que es posible añadir nuevas funcionalidades fácilmente. Brinda la posibilidad de configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, así como modificar y controlar muchas aplicaciones *open source* (código abierto). Entre los módulos de Webmin se encuentra Tareas Planificadas de *Cronb (Scheduled Cron Jobs*), ubicado en la categoría de sistema, es utilizado para programar tareas de *Cron* desde la interfaz gráfica. Se puede activar, desactivar, crear nuevas tareas *Cron*, y otras acciones desde Webmin (4).

1.4.3 Nova NAS

Nova NAS es una aplicación web cubana para la administración de un servidor de almacenamiento en la red (NAS), desplegada con Nova Servidores 2015, que gestiona los servicios y herramientas de forma centralizada y provee una interfaz intuitiva. Esta aplicación web provee una interfaz para la administración de tareas programadas. Algunas de sus principales características de Nova NAS son (6) :

- ✓ Desplegada sobre Nova Servidores 2015.
- ✓ Administración basada en la web.
- ✓ Gestión de tareas programadas, Nova NAS provee una interfaz para la administración de tareas programadas.
- ✓ Soporte para conexión segura.
- ✓ Administración remota (SSH, *Secure Shell o* Protocolo Seguro).
- ✓ Transferencia de archivos (FTP, Protocolo de Transferencia de Archivos, por sus siglas en inglés *FileTransfer Protocol*).
- ✓ Copias sincronizadas (*Rsync, Remote synchronize*).

- ✓ Compartición en la red (NFS, Sistema de archivos de red por sus siglas en *inglés Network File System*).

1.5 Análisis comparativo de aplicaciones informáticas para la administración de tareas programadas

A continuación, se muestra una comparación entre las aplicaciones informáticas para la administración de tareas programadas mencionadas anteriormente. Para ello se definieron una serie de indicadores, de acuerdo con los requisitos necesarios para el desarrollo de la propuesta de solución y el contexto del negocio, además de tenerse en cuenta las necesidades del cliente, ver Tabla 5.

Tabla 5. Definición de criterios y su clasificación.

Fuente: elaboración propia.

Clasificación / **Criterios para el análisis**	**Si cumple**	**No cumple**
Basada en software libre	Está basada en software libre	No está basada en software libre
Integración con Nova 7.0	Permite la integración con Nova 7.0	No permite la integración con Nova 7.0
Administración remota por protocolo SSH	Permite la administración de forma remota	No permite la administración de forma remota
Totalidad de parámetros del *cron*	Abarca todos los parámetros del *cron*	No abarca todos los parámetros del *cron*
Detección de cambios manuales en los ficheros de configuración del *cron*	Detecta todos los cambios realizados en los ficheros del *cron*	No detecta todos los cambios realizados en los ficheros del *cron*

En la Tabla 6 se muestra la comparación entre las aplicaciones informáticas para la administración de tareas programadas mencionadas anteriormente.

Tabla 6. Análisis comparativo entre aplicaciones de administración remota de tareas programadas.

Fuente: elaboración propia.

Criterios para el análisis	**Sistemas informáticos para la administración de tareas programadas**		
	Crontab-ui	Webmin	Nova NAS
Basada en software libre	Sí	Sí	Sí
Integración con Nova 7.0	Sí	Sí	Sí
Administración remota por protocolo SSH	No	Sí	Sí
Totalidad de parámetros del *cron*	Sí	Sí	No
Detección de cambios manuales en los ficheros de configuración del *cron*	Sí	No	Sí

Luego del estudio realizado se decide desarrollar una herramienta informática para administrar de forma remota las tareas programadas, debido a que las aplicaciones estudiadas no suplen las todas las necesidades para Nova 7.0. Se utilizarán algunas de las funcionalidades y características que emplean las aplicaciones antes mencionadas para la implementación de la nueva herramienta a desarrollar.

1.6 Metodología AUP adaptada para la UCI

La metodología de desarrollo de software AUP (*Agile Unified Process*, Proceso Unificado Ágil) para la UCI es una variación de AUP (13) que logra estandarizar el proceso de desarrollo de software en los proyectos productivos de la universidad, además de convertirse en la metodología rectora de su desarrollo productivo, ya que se adapta perfectamente al ciclo de vida de la actividad productiva de los diferentes centros de la institución (14). A continuación, se presentan las fases de AUP-UCI.

Inicio: durante esta fase se efectúan las actividades relacionadas con la planeación del proyecto. Se realiza un estudio inicial de la organización o cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto teniendo en cuenta los requisitos y la arquitectura. Durante el desarrollo se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

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

La presente investigación se desarrolla en la fase de Ejecución, pues la misma propicia que se ejecuten las actividades necesarias para realizar el software; además aborda las disciplinas: requisitos, análisis y diseño, implementación, pruebas internas y pruebas de aceptación.

Escenario para la disciplina Requisito

La metodología AUP en su variación para la UCI posee cuatro escenarios para modelar el sistema en los proyectos. Para la propuesta de solución se define a utilización del escenario 4 el cual se modela mediante historias de usuario, ya que es factible aplicarlo a proyectos bien definidos que ya hayan evaluado el negocio a informatizar, además el cliente siempre acompaña al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

1.7 Lenguajes y herramientas para el modelado

Las herramientas CASE (Ingeniería de Software Asistida por Computadora, en inglés *Computer Aided Software Engineering*) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas nos pueden ayudar en todos los aspectos en el ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras (15).

Visual Paradigm v8.0

Herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue (para metodologías de ciclo completo). El software de modelado UML agiliza la construcción de aplicaciones de calidad. Propicia además un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación hasta el análisis y el diseño, además permite la correcta realización de los artefactos propuestos por la metodología AUP adaptada para la UCI (16). Se seleccionó la herramienta *CASE Visual Paradigm for UML 2.0* (17) ya que es gratuita y se encuentra disponible para distribución Linux.

Lenguaje Unificado de Modelado v2.0

El Lenguaje de Modelado Unificado (UML, *Unified Modeling Language)* es un lenguaje que se centra en la representación gráfica de un sistema, por lo que permite construir, modelar y diseñar dicho sistema. Un modelo UML está compuesto por 3 clases de bloques de construcción: los elementos (representaciones abstractas de cosas reales o ficticias); las relaciones (relacionan los elementos entre sí); y los diagramas (son colecciones de los elementos con sus relaciones) (18).

1.8 Tecnologías para la implementación

Para el proceso de desarrollo de la herramienta de administración remota de tareas programadas para Nova 7.0 se han seleccionado un conjunto de herramientas y tecnologías que ayudan al diseño, la modelación e implementación de la solución propuesta, con el objetivo de obtener resultados satisfactorios. Las mismas fueron establecidas por la dirección del Centro CESOL permitiendo que estas tecnologías se adapten a las necesidades y características del desarrollador.

1.8.1 Lenguajes

El lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión (19). A continuación, se plantean los lenguajes utilizados para la construcción de la herramienta para la administración remota de tareas programadas en Nova 7.0.

HTML5

HTML (*HyperText Markup Language*, Lenguaje de Formato de Documentos para Hipertexto) es un lenguaje que se utiliza para el desarrollo de páginas web. Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. Permite scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son *JavaScript* y *PHP*. La versión 5 de *HTML* establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos (17). Se utiliza *HTML5* para la implementación de las interfaces de la herramienta (20).

CCS3

CSS3 (*Cascading Stylesheets*, Hojas de estilo en cascada) es un lenguaje de diseño gráfico creado por *W3C* (*World Wide Web Consortium*, Consorcio Mundial de la Red), para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web e interfaces de usuario escritas en *HTML* o *XHTML*. Puede ser aplicado a cualquier documento *.xml, .xhtml, .svg, .xul, .rss,* entre otros. También permite aplicar estilos no visuales, como las hojas de estilo auditivas. Se emplea *CCS3* para la implementación de las interfaces de la herramienta (21)

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, *JavaScript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con *JavaScript* se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Se utiliza *JavaScript* para la en la implementación de la lógica del negocio (22).

1.8.2 Marcos de trabajo

Un marco de trabajo o framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener (23).

Los marcos de trabajo definido para el desarrollo de la aplicación son:

Electron v8.0.0

Es marco de código abierto desarrollada por GitHub[3] para crear aplicaciones de escritorio utilizando *HTML*, *CSS* y *JavaScript* (24). Una de las ventajas principales del uso de *Electron* es que, dado que se basa en tecnologías web, es multiplataforma**,** lo que permite implementar aplicaciones para Linux, *MacOS* y *Windows*, con el mismo código. También cuenta con elementos nativos, como menús y notificaciones, así como herramientas de desarrollo útiles para la depuración y el informe de errores (24).

[3] GitHub: es una plataforma de desarrollo colaborativo de software para alojar proyectos(URL: https://github.com)

Alguna de las nuevas características que tiene la versión 8.0.0 ha añadido algunas nuevas características además de correcciones y mejoras al código del proyecto.

Vue.Js v4.2.2

Vue es un *framework open source* de JavaScript, el cual permite construir interfaces de usuarios de una forma muy sencilla. Una de las características más importantes de *Vue* es el trabajo con componentes. Un componente Vue es un elemento en el cual se encapsula código reutilizable. Dentro de un componente se pueden encontrar etiquetas *HTML*, estilos de *CSS* y código *JavaScript*. Los componentes permiten desarrollar proyectos estructurado en módulos y fáciles de escalar, pudiendo reemplazar un componente por otro de una forma muy sencilla (25).

Marco de diseño

Vuetify

Vuetify es la biblioteca de interfaz de usuario de Vue.js con lindos diseños de sus competentes y ha estado en desarrollo activo desde 2016 (26). *Vueify* es un framework que combina la potencia del propio *Vue.js* con la estética de *Material Design*[4]. Permite acelerar el desarrollo de aplicaciones complejas incorporando una gran cantidad de componentes "listos para usuar". Dispone de una enorme librería de componentes que incluye desde elementos como sencillos formularios como botones y *combobox*, a componentes más avanzados típicos de Android como "*cards*" o "*snackbars*" (27).

1.8.3 Entorno Integrado de Desarrollo (IDE)

Un IDE (*Integrated Development Enviroment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI) (28).

El *IDE* seleccionado por el proyecto de investigación para la construcción de la herramienta es:

Visual Studio Code v1.42.1

[4] Marial Design: es una normativa de diseño enfocado a las visualizaciones del sistema operativo Android, además de web y en cualquier plataforma.

Visual Studio Code es un editor de código ligero y potente que está disponible para los sistemas operativos de *Windows*, *Linux* y *MacOS*. Viene con soporte incorporado para *JavaScript* y *Node.js* y tiene un rico ecosistema de extensiones para otros lenguajes como*: C++, C#, Java, Python, PHP, GO*, entre otros. Posee tiempos de ejecución *.NET* y *Unity* (29).

Ventajas de *Visual Studio Code*:

- ✓ Brinda la posibilidad de configurar la vista de acuerdo con nuestras necesidades. De esta forma, podremos tener más de un código visible al mismo tiempo, las carpetas del proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades.
- ✓ Posee opciones de actualización en tiempo real del código desde la vista del navegador y compilación en vivo de los lenguajes que lo requieran.

1.8.4 Herramienta de control de versiones

Un sistema de control de versiones es una herramienta que registra todos los cambios hechos en uno o más proyectos, almacenando así versiones del producto en todas sus fases del desarrollo. Las versiones son como fotografías que registran su estado en ese momento del tiempo y se van guardando a medida que se hacen modificaciones al código fuente (30).

Git: fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente (31). Proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida. La rapidez en la gestión distribuida y el realmacenamiento periódico de paquetes son algunas de sus características más importantes.

1.9 Conclusiones del capítulo

En este capítulo se realizó un estudio relacionado con las herramientas de administración de tareas programadas en GNU/Linux, lo que permitió adquirir conocimientos que facilitan el desarrollo de la herramienta a implementar. De las herramientas estudiadas se hizo una descripción y se determinó que ninguna de ellas se ajusta a las necesidades de la distribución cubana Nova 7.0, por lo que se decide implementar una nueva herramienta para este sistema apoyándose en funcionalidades que ofrecen las soluciones estudiadas.

Se definieron todas las herramientas, tecnologías y lenguajes de modelación y programación que serán utilizadas para la creación de la herramienta y que fueron definidas por la dirección del proyecto. Todo este proceso está guiado por la metodología AUP en su variación para la UCI, teniendo en cuenta que se adecua a las características del proyecto.

CAPÍTULO II: Análisis y diseño de la herramienta de administración remota de tareas programadas en Nova 7.0

2.1 Introducción

En este capítulo se describe la propuesta a desarrollar, tomando como punto de partida los aspectos técnicos de la investigación referentes al análisis y diseño de la herramienta de administración remota de tareas programadas en Nova 7.0. Contiene un modelo conceptual que permite comprender el contexto del negocio. Se definen los requisitos funcionales y no funcionales a través de la especificación de requisitos de software y su descripción mediante historias de usuario. Se realiza el diagrama de clases del diseño, se describe la arquitectura y los patrones de diseño a utilizar en el desarrollo de la herramienta de la propuesta de solución.

2.2 Descripción del contexto de la propuesta de solución

La descripción del contexto del negocio se realizó mediante el desarrollo de un modelo conceptual, permitiéndole a la autora comprender los conceptos asociados al negocio informatizado y el funcionamiento del mismo.

2.2.1 Modelo conceptual

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se le denomina modelo conceptual, modelo de objetos del dominio y modelo de objetos de análisis. Al utilizar la notación *UML*, un modelo de dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Su principal objetivo es definir las interrelaciones de los objetos más importantes representados mediante clases. Además, desempeña un papel clave en la comprensión del entorno actual (32).

La Figura 1 muestra el modelo conceptual del negocio de la propuesta de solución.

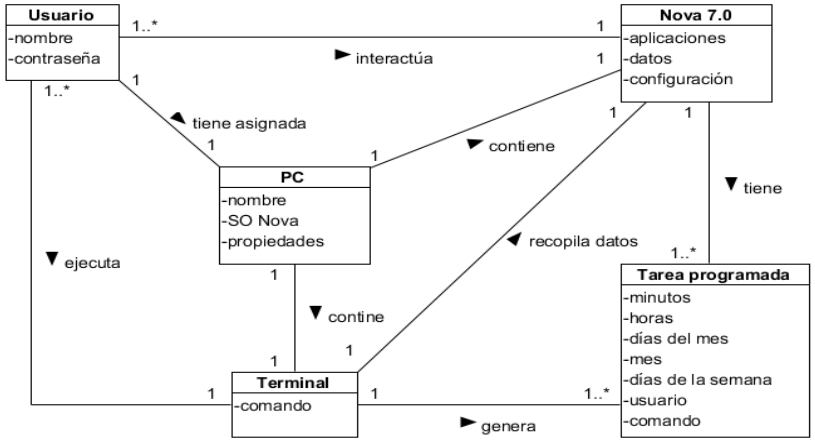


Figura 1. Modelo conceptual.

Fuente: elaboración propia.

A continuación, se muestran los conceptos del modelo conceptual, los cuales permiten una mejor comprensión del contexto del negocio.

Nova 7.0: distribución cubana de GNU/Linux. Cuenta con las funcionalides de cualquier sistema operativo de poder instalar aplicaciones, reproducir archivos multimedias y realizar acciones laborales.

PC: (*Personal Computer* o Computadra Personal) es un computador que contiene la distribución cubana de GNU/Linux que cuenta con una terminal, y posee Nova 7.0. Con este ordenador se puede realizar la programación de tareas automáticas para esta distribución.

Tarea programada: es aquella que el usuario puede configurar para que cualquier proceso, programa o archivo se ejecute en el tiempo o fecha que se desee. En GNU/Linux las tareas pueden configurarse para ejecutarse de forma automática en un período de tiempo concreto y en las fechas indicadas.

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

Usuario: es el cliente final del sistema operativo, el cual accede a la computadora para poder instalar paquetes y realizar las operaciones que necesiten ejecutarse con permiso de administrador del propio sistema.

A continuación, se muestran las características y restricciones de los atributos representados en los conceptos de la Figura 1.

2.2.2 Descripción del contexto de la herramienta a desarrollar

Con el objetivo de darle solución al problema planteado, el presente trabajo de diploma propone desarrollar una herramienta que cuente con las funcionalidades necesarias para administrar remotamente las tareas programadas en Nova 7.0. La herramienta a desarrollar permitirá mediante una interfaz gráfica amigable la configuración, planificación y edición de tareas personalizadas, las cuales se ejecutan a cada minuto y hora, así como semanalmente y mensualmente. Estas funcionalidades se podrán llevar a cabo debido a que la aplicación realizará conexiones seguras mediante el uso del protocolo *SSH* al servidor, accederá los ficheros de configuración y los modificará de acuerdo a las especificaciones del usuario.

2.3 Disciplina de requisitos

La disciplina de requisitos es una de las más importantes en el proceso de desarrollo de software; consiste en desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.

2.3.1 Fuentes de obtención para la disciplina de requisitos

Uno de los elementos más importantes del proceso de desarrollo del software es la obtención de los requisitos, debido a que ayuda a conciliar conflictos de intereses entre los involucrados y determinar qué tipo de software se desea desarrollar (33).

Se tienen como fuentes de obtención de requisitos:

- ✓ Modelo conceptual.
- ✓ Herramientas informáticas de administración de tareas programadas.
- ✓ Especialistas del centro CESOL.

2.3.2 Técnicas de identificación de requisitos

Las técnicas de identificación de requisitos de software permiten definir las necesidades de negocio de los clientes y los usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle, requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar (34).

En la obtención de los requisitos para la administración remota de tareas programadas para Nova 7.0 se utilizaron diferentes técnicas, las cuales se describen a continuación.

Entrevistas

La entrevista es de gran utilidad para obtener información cualitativa como opiniones o descripciones subjetivas de actividades. Es una técnica muy utilizada y requiere una mayor preparación y experiencia por parte del analista. La entrevista consiste en un intento sistemático de recoger información de otra persona a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada donde es muy importante la forma en que se plantea la conversación y la relación que se establece en la entrevista (35). La técnica se aplicó a través de entrevistas (Anexos 1 y 2) realizadas a especialistas del Centro CESOL (Anexo 4), para conocer las tecnologías y las características de la herramienta a desarrollar.

Observación

Por medio de esta técnica el analista obtiene información de primera mano sobre la forma en que se efectúan las actividades. Este método permite observar la forma en que se llevan a cabo los procesos y, por otro lado, verificar que realmente se sigan todos los pasos especificados. En muchos casos los procesos difieren del papel a la práctica. Los observadores experimentados saben qué buscar y cómo evaluar la relevancia de lo que observan (35). Se aplicó la técnica de observación para obtener información sobre las características y funcionalidades de las herramientas informáticas estudiadas para la administración de tareas programadas, ver Anexo 3.

Elaboración de prototipos de interfaz

Los prototipos consisten en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación pedida. Permiten a los usuarios experimentar para ver cómo este ayuda a su

trabajo, fomentan el desarrollo de ideas que desembocan en requisitos, mejoran las especificaciones de requisitos, identifican discrepancias entre los desarrolladores y los usuarios y permite formalizar la aceptación previa por parte del cliente de los requisitos del proyecto (35)**.** Se aplicó la técnica de elaboración de prototipos para detallar las características de las interfaces de la herramienta a desarrollar. Los prototipos desarrollados fueron revisados y aprobados por especialistas del Centro CESOL como representación visual de la propuesta de solución.

2.3.3 Especificación de requisitos de software

Los requisitos son capacidades y condiciones con las cuales debe ser conforme el sistema a desarrollar. El primer reto del trabajo de los requisitos es encontrar, comunicar y recordar lo que se necesita realmente, de manera que tenga un significado claro para el cliente y los miembros del equipo de desarrollo (36).

Requisitos funcionales

Los Requisitos Funcionales (RF) expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento) (37).

En la Tabla 7 el listado de los requisitos funcionales de la propuesta de solución. La complejidad de los requisitos funcionales se obtuvo mediante el producto de trabajo “Evaluación de requisitos”, propuesto en el expediente de proyecto 5.0 para la actividad productiva de la universidad.

Tabla 7. Listado de requisitos funcionales.

Fuente: elaboración propia.

Número	Requisito	Descripción	Complejidad	Prioridad
RF1	Conectarse remotamente	La herramienta permite conectarse remotamente a otra máquina.	Alta	Alta
RF2	Iniciar servicio *cron*	La herramienta permite iniciar el servicio *cron*, el cual se encuentra instalado automáticamente en el sistema.	Media	Alta
RF3	Reiniciar servicio *cron*	La herramienta permite reiniciar el servicio *cron*.	Media	Media
RF4	Detener servicio *cron*	La herramienta permite detener el servicio *cron*.	Media	Media

RF5	Crear tarea programada	La herramienta permite crear una tarea programada.	Alta	Alta
RF6	Mostrar tarea programada	La herramienta muestra una tarea programada con sus campos (nombre, minutos, horas, días de la semana, día del mes, mes, usuario y comando.	Media	Media
RF7	Listar tareas programadas	La herramienta muestra un listado con todas las tareas programadas que se encuentran creadas hasta el momento.	Media	Media
RF8	Editar tarea programada	La herramienta permite editar una tarea programada que ha sido seleccionada previamente.	Alta	Alta
RF9	Eliminar tarea programada	La herramienta permite eliminar una tarea programada que ha sido seleccionada previamente.	Media	Media

Requisitos no funcionales

Los requisitos no funcionales son propiedades que hacen al producto atractivo, usable, rápido y confiable. Se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar. Además, se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software (38).

La definición de los requisitos no funcionales de la propuesta de solución se realizó mediante el estándar de calidad ISO-25010 (*International Organization for Standarization*, Organización Internacional de Normalización), propuesto en el producto de trabajo “Especificación de requisitos de software” del expediente de proyecto utilizado en la actividad productiva de la universidad. A continuación, se presenta en la Tabla 8 el listado de los requisitos no funcionales de la propuesta de solución.

Tabla 8. Listado de requisitos no funcionales.

Fuente: elaboración propia.

Número	**Requisito**	**Descripción**
RNF1	Requisito de compatibilidad	La herramienta para administrar remotamente las tareas programadas debe ser compatible con Nova 7.0.

RNF2	Requisito de seguridad	A la herramienta solo tendrán acceso los usuarios administradores.
RNF3	Requisito de mantenibilidad	La herramienta debe hacer uso de los estándares de codificación definidos para la herramienta.
RNF4	Requisito de usabilidad 1	En la herramienta se deben visualizar todos los mensajes en idioma español. La tipografía debe ser uniforme, de un tamaño adecuado.
RNF5	Requisito de usabilidad 2	La herramienta debe ofrecer una interfaz amigable, fácil de operar. Las interfaces deben poseer un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios la utilización de la misma.
RN6	Restricciones del diseño e implementación	La herramienta se debe desarrollar utilizando el lenguaje del lado del cliente: HTML5, CSS3 y JavaScript., así como los marcos de trabajo Vuejs v 4.2.2 y Electron v.8.0, como marco de diseño Vuetify y como entorno de desarrollo Visual Studio Code v 142.

Se utilizaron las historias de usuario para describir las características de los requisitos funcionales definidos anteriormente.

2.3.4. Descripción de requisitos de software mediante Historias de Usuario

Para la descripción de los requisitos se elaboraron 9 Historias de Usuario. En la Tabla 9 se muestra la descripción de la historia de usuario correspondiente al RF 5: Crear tarea programada. El resto de las Historias de Usuario se encuentra en el Anexo 5.

Tabla 9. Descripción de la historia de usuario correspondiente al RF5 Crear tarea programada.

Fuente: elaboración propia.

Historia de Usuario

Número: 5	**Nombre del requisito:** Crear tarea programada
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 48 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 48 horas

Descripción:

La herramienta permite al usuario crear la planificación de una tarea programada. El usuario selecciona la opción “Lista de tareas” del menú izquierdo, luego selecciona la opción “Crear” que se encuentra en el menú superior, seguidamente se activan los criterios de selección de tiempo (minuto que admite un rango de (0-59), hora (1-24), día de la semana (Lunes-Domingo o 0-7), día del mes (1-31) y el mes (Enero-Diciembre o 0-12)), un campo para introducir el usuario y el comando que se va a ejecutar. Una vez llenados los campos se selecciona la opción “Guardar” en el menú inferior. En caso de que el usuario desee cancelar la creación de la tarea posee la opción “Limpiar” del menú inferior; y la opción “Atrás”, en el menú superior, la cual permite volver a la opción anterior.

Observaciones:

Para que se pueda crear una tarea es necesario que el servicio cron se encuentre iniciado.

Si la tarea fue creada correctamente debe añadirse a un listado de tareas programadas.

Prototipo de Interfaz

2.3.5 Técnicas de 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 del negocio, de los clientes y de los usuarios, que se han detectado las inconsistencias, omisiones y errores y que estos han sido corregidos (39).

A continuación, se describen las técnicas de elaboración de prototipos de interfaz y diseño de casos de pruebas.

Prototipo de interfaz de usuario

El prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un software que permite a clientes y usuarios entender más fácilmente la propuesta de los requisitos para resolver sus problemas de negocio. Está definido como la creación u obtención de modelos concretos o la aproximación de ideas mediante una serie de tecnologías para demostrar conceptos y probar opciones de diseño, posibilitando la validación y optimización de los mismos en un tiempo e inversión mínima (35). Los prototipos de interfaz realizados fueron analizados con un conjunto de especialistas del centro CESOL, lo que le permitió conocer las posibles deficiencias existentes, para poder realizar las correcciones necesarias antes de la implementación de funcionalidades de la herramienta.

Diseño de caso de prueba (DCP)

Es una parte de las pruebas de componentes y sistemas en las se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. Su objetivo es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requisitos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas (35)**.** La elaboración de los casos de pruebas permitió especificar los posibles datos de entradas y de salida de los requisitos funcionales para la herramienta de administración remota de tareas programadas para Nova 7.0. Fueron generados 9 casos de prueba en total, los cuales fueron revisados por especialistas del centro CESOL.

A continuación, se presenta el DCP que corresponde al RF 5 Crear tarea programada. Ver Tabla 10.

Caso de prueba correspondiente al RF 5 Crear tarea programada

Descripción general: la herramienta permite al usuario crear una tarea programada.
Precondiciones: El usuario debe estar autorizado en la herramienta.

Tabla 10. Caso de prueba RF5. Crear tarea programada.

Fuente: elaboración propia.

Escenario	**Descripción**	**Minutos**	**Horas**	**Días de la semana**	**Meses**	**Dia del mes**	**Usuario**	**Comando**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear tarea programada	El usuario selecciona la opción de "Crear" del menú superior, luego se	5	5	Martes	Enero	15	Sueyaile	/usr/respaldodiario.sh	Se crea satisfactoriamente la tarea programada	1.Seleccionar botón "Crear " 2 Llenar los campos para la

	activan los campos de selección de (minutos, horas, días de la semana, mes y días del mes), además de los campos usuario y comando. Una vez llenado los campos se selecciona el botón "Guardar" en el menú inferior.									configuración de la tarea. 3.Seleccionar botón "Guardar"
EC 1.2 Cancelar la creación de la tarea programada	En caso de que el usuario se arrepienta de crear la tarea, posee la opción de "Borrar", en el menú inferior y	NA	NA	NA	NA	NA	NA	NA	Se desactivan los criterios de selección	1.Seleccionar botón "Crear " 2. Seleccionar botón Borrar" 3. Seleccionar botón

	"Regresar", para volver a la vista anterior.									"Regresar".

Comentado [m1]: Ajustar tabla al contenido

En la Tabla 11 se muestra las descripciones de las variables tenidas en cuenta en los casos de pruebas.

Descripción de variables

Tabla 11.Descripción de las variables. Caso de prueba RF5 Crear tarea programada.

Fuente: elaboración propia.

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Minutos	campo de selección, checkbox, campo de texto.	No	El campo de texto solo admite valores numéricos, checkbox selecciona todos los minutos y el campo de selección permite seleccionar uno o varios minutos.
2	Horas	campo de selección, checkbox, campo de texto.	No	El campo de texto solo admite valores numéricos, checkbox selecciona todos las horas y el campo de selección permite seleccionar una o varias horas.
3	Días de la semana	campo de selección, checkbox, campo de texto.	No	El campo de texto solo admite valores alfa, checkbox selecciona todos los días de la semana y el campo de selección permite seleccionar uno o varios días de la semana.
4	Mes	campo de selección, checkbox, campo de texto.	No	El campo de texto solo admite valores alfas, checkbox selecciona todos los meses y el campo de selección permite seleccionar uno o varios meses.
5	Días del mes	campo de selección, checkbox, campo de texto.	No	El campo de texto solo admite valores numéricos, checkbox selecciona todos los días del mes y el campo de selección permite seleccionar uno o varios días del mes
6	Usuario	campo de selección	No	El campo de selección admite seleccionar un solo usuario

7	Comando	textarea	No	El textarea admite un string con el comando que se va a ejecutar.

2.4 Análisis y Diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. Los objetivos del análisis y diseño son: transformar los requisitos al diseño del futuro sistema, desarrollar una arquitectura para el sistema y adaptar el diseño para que sea consistente con el entorno de implementación (40). En las Figuras 4 y 5 se muestran dos ejemplos de vista del diseño correspondiente al requisito funcional RF 7 Crear tarea programada.

Figura 2. Vista del diseño correspondiente al RF 7 Crear tarea programada.

Fuente: elaboración propia.

Figura 3. Vista del diseño correspondiente al RF 7 Crear tarea programada.

Fuente: elaboración propia.

2.4.1 Patrones de diseño

Un patrón de diseño es una descripción de la comunicación entre objetos y clases, personalizada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades (41).

Patrones GRASP

Los patrones GRASP (*General Responsibility Assigment Software Patterns* o Patrones Generales de Software para la Asignación de responsabilidades) describen los principios fundamentales para asignar responsabilidades a los objetos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software (42). A continuación, se mencionan los patrones y donde fueron utilizados.

Experto: es empleado para asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir con una determinada responsabilidad (43). Este se encuentra presente en la clase *Config.js()* la cual maneja toda la información referente a los campos de las tareas programadas. Las clases *SelectJobs.vue()* y *ListJobs:js()* importan las etiquetas presentes en la clase

Config.js() y de esta manera implementar los atributos de las tareas programadas. En la Figura 4 se muestra la aplicación del patrón Experto en un extracto del código implementado.

config.js

```
export const APP_NAME = "ADMINISTACION REMOTA | SISTEMA"
export const APP_NAME_VALUE = "crontab-remote-admin"

export const REMOTE_FILE_CRON_CONF = "nano /etc/crontab"


export const MIN_OPTION =[
    '00','01', '02', '03', '04',
    '05', '06', '07', '08',
    '09', '10', '11','12',
    '13', '14', '15','16',
    '17', '18', '19','20',
    '21', '22', '23','24',
    '22', '23', '24','25',
    '26', '27', '28','29',
    '30', '31', '32','33',
    '34', '35', '36','37',
    '38', '39', '40','41',
    '42', '43', '44','45',
    '46', '47', '48','49',
    '50', '51', '52','53',
    '54', '55', '56','57',
    '58', '59', '60',
]

export const HOUR_OPTION = [
    '1', '2', '3', '4',
    '5', '6', '7', '8',
    '9', '10', '11','12',
    '13', '14', '15','16',
    '17', '18', '19','20',
    '21', '22', '23','24',
]
```

Figura 4. Patrón del diseño GRASP: Experto.

Fuente: elaboración propia.

Creador: ayuda a identificar qué clase debe ser responsable de la creación o instanciación de nuevos objetos o clases. Permite la visibilidad entre la clase creada y la clase creadora (43). Este patrón se evidencia en las clases *ListJobs.vue() y SelctJobs.vue()*, en ambas mediante el método *created()* el cual es responsable de la creación de las instancias de la clase *Cofig.js()*. A continuación, en la Figura 5 se muestra la aplicación del patrón Creador en un extracto del código implementado.

ListJobs.vue

```
23
24  export default {
25    data: () => ({
26      isOpenConfirm: false,
27      pending: false,
28      headers: [
29        { text: "Minutos", value: "m" },
30        { text: "Horas", value: "h" },
31        { text: "Días del mes", value: "dom" },
32        { text: "Meses", value: "mon" },
33        { text: "Dias de la semana", value: "dow" },
34        { text: "Usuario", value: "user" },
35        { text: "Comando", value: "comand" },
36      ],
37      job_list:[]
38
39    }),
40   created() {
41      this.pending = true;
42      CronConfController.getData(data => {
43        this.job_list = data;
44        this.pending = false;
45      });
46    },
```

Figura 5. Patrón del diseño GRASP: Creador.

Fuente: elaboración propia.

2.4.2 Arquitectura de la herramienta

El proceso de diseño de la arquitectura debe decidir cuál funcionalidad es la más importante a desarrollar. Además, define cuáles son los componentes básicos del sistema y cómo se relacionan entre ellos para implementar la funcionalidad (44).

El patrón de arquitectura en N capas, es el que se aplicará en este diseño ya que brinda una organización jerárquica tal que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. Este patrón es un sub-estilo dentro del estilo *llamada y retorno*, en el cual las restricciones topológicas del patrón pueden incluir una limitación que exige a cada capa operar solo con la adyacente y a elementos de una capa entenderse sólo con otros elementos de la misma capa (40).

Se define como arquitectura de la propuesta de solución la arquitectura n-capas, usando la variante de 3 capas, gracias a la facilidad que ofrece. Las capas son las siguientes:

- ✓ Presentación: es la capa encargada de la representación visual de la herramienta, informar sobre la situación de los procesos de negocio e implementación de las reglas de validación de interfaz. Las interfaces de usuario se comunican con la capa de Lógica de Negocio. Este caso está presente en las clases *layout.vue(), views,js().*

- ✓ Lógica de Negocio: es la capa donde se alojan mayoritariamente las clases. Interactúa con la capa Presentación para recibir y procesar las peticiones del usuario y enviar las respuestas a la capa de Acceso a Datos. Comprende la clase *ssh-connect.js.*
- ✓ Acceso a Datos: es donde son procesados los datos. Tiene asignada la responsabilidad de contener el código necesario para el manejo de datos.

A continuación, en la Figura 6 se representa la arquitectura empleada para el desarrollo de la solución.

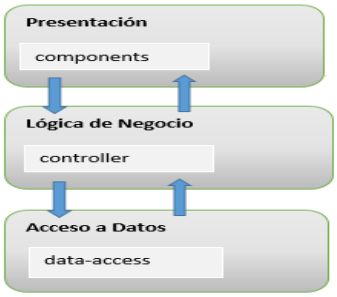


Figura 6. Arquitectura n-capas.

Fuente: elaboración propia.

2.4.3 Diagrama de clases

Los diagramas de clases del diseño permiten describir gráficamente las especificaciones de las clases del software. Muestran las clases (descripciones de objetos que comparten características comunes) que componen el sistema y como se relacionan entre sí. A continuación, se representa el diagrama de casos del diseño, ver Figura 7.

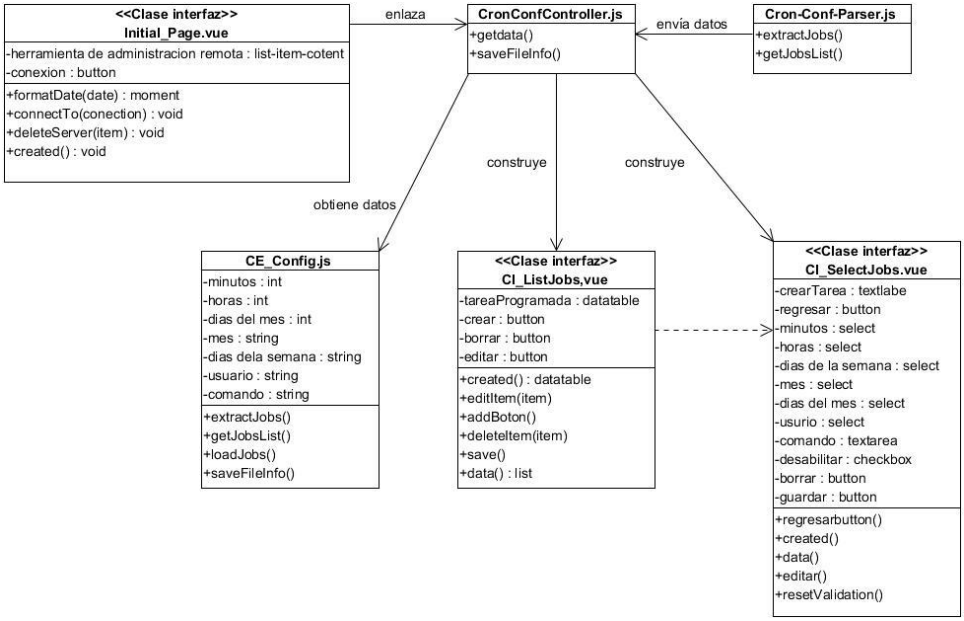


Figura 7. Diagrama de clases del diseño correspondiente al RF5 Crear tarea programada.

Fuente: elaboración propia.

El diagrama anterior está formado por las clases:

- ✓ Clases interfaces: definen todas las abstricciones para la interacción entre el usuario y la aplicación. Esta contiene las clases *ListJobs.vue(), SelectJobs.vue()* y *Main.vue().*
- ✓ Clase controladora: permite encapsular las funcionalidades necesarias para interactuar con las clases interfaces, las clases entidades y las clases de acceso a datos. En este caso está presente la clase *cron-conf-crontroller.js().*
- ✓ Clases de acceso a datos: permite la comunicación entre la clase controladora y la base de datos. Está formada por la clase *cron-cronf-parser.js()* y *server-datbase.js() y db.js().*

2.4.4 Modelo de datos

Un modelo de datos es un conjunto de herramientas conceptuales para la descripción de los datos y las relaciones entre ellos, su semántica y las restricciones de consistencia (45).

La persistencia de datos de la propuesta de solución se encuentra ubicada en el archivo *crontab* de Nova 7.0, registrada en la dirección *nano/etc/crontab* donde se encuentran los campos de configuración de la tarea programada (minutos, horas, día de la semana, día del mes, mes, usuario y comando).

Diagrama de paquetes

El diagrama de paquetes muestra las agrupaciones lógicas en que está dividido el sistema, así como las dependencias entre dichas agrupaciones**.** A continuación, en la Figura 8 se muestra el diagrama de paquetes de la propuesta de solución (42).

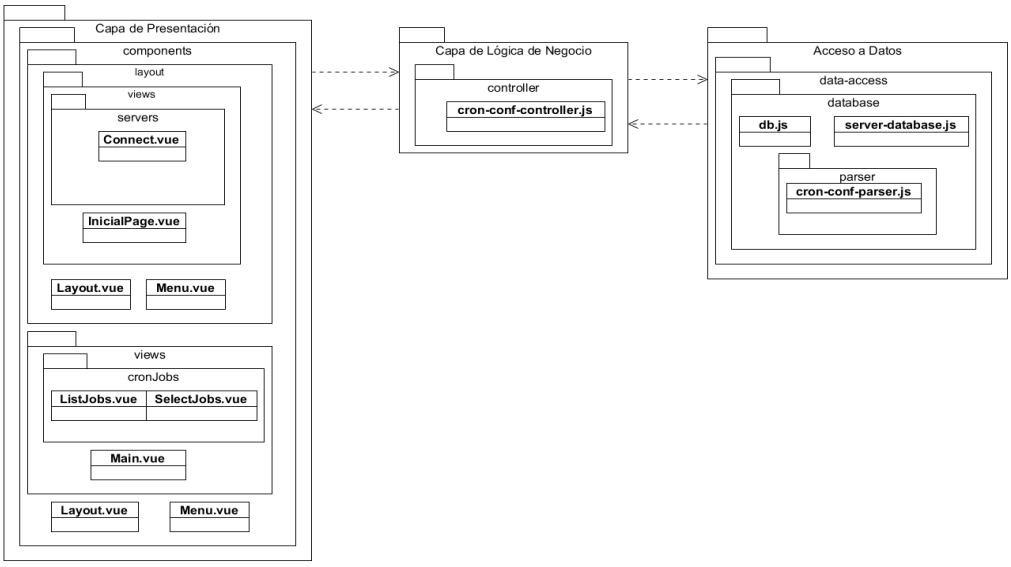


Figura 8. Diagrama de paquetes.

Fuente: elaboración propia.

2.5 Conclusiones del capítulo

El análisis y diseño de la herramienta de administración remota de tareas programadas para Nova 7.0 permitió identificar 9 requisitos funcionales y 5 no funcionales, de acuerdo a las necesidades del cliente. El empleo del patrón arquitectónico N-Capas posibilitó validar la organización y la comunicación entre los diferentes componentes de la propuesta de solución. El diagrama de clases propició modelar las diferentes clases, sus atributos y relaciones. El diagrama de paquetes evidenció la composición de las capas de la arquitectura de la propuesta de solución.

CAPÍTULO III: Construcción y validación de la herramienta informática para la administración remota de tareas programadas en Nova 7.0

3.1 Introducción

El presente capítulo contiene los productos de trabajos que se realizan en la disciplina de implementación y prueba, así como los estándares de codificación, diagrama de despliegue, diagrama de componentes y un ejemplo de la interfaz gráfica de usuario de la implementación de la propuesta de solución. Se documentan las pruebas realizadas y la evaluación del índice de satisfacción grupal sobre la herramienta de administración remota de tareas programadas en Nova 7.0.

3.2 Implementación

Una implementación es la realización de una especificación técnica o algoritmos como un programa, componente de software, u otro sistema de cómputo. También son una manifestación en el mundo real de las funciones de procesamiento y estructuras de la información. Muchas implementaciones son realizadas según una especificación o un estándar (34).

3.2.1 Estándares de codificación

Los estándares de codificación abarcan todos los aspectos generales del código fuente de una aplicación. Su importancia radica en la legibilidad del código para que sea entendible por todos los programadores de un proyecto como si este hubiese sido escrito por una sola persona. Cuando se añade código nuevo o se realiza mantenimiento del sistema, el estándar de codificación debe sugerir cómo trabajar con el código ya existente (46).

Para la propuesta de solución se tuvieron en cuenta los estándares de codificación para *Vue.js* definidos por una Guía de estilo de *Standard Library* (47). de la página oficial de Vue.js. A continuación, se presentan las principales prácticas empleadas en el desarrollo de la aplicación.

Cada componente es un archivo

Siempre que un compilador pueda concatenar archivos, cada componente debería estar en su propio archivo. Los nombres de las carpetas se escriben con minúscula y el de los componentes con mayúscula.

A continuación, en la Figura 9 se presenta un fragmento de la estructura del proyecto.

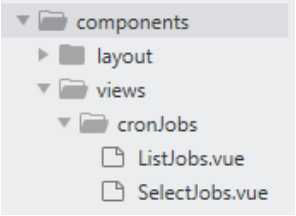


Figura 9. Estructura de los componentes.

Fuente: elaboración propia.

Comilla en los valores de los atributos

Los valores de atributos HTML no vacíos siempre deben estar dentro de comillas (simples o dobles). A continuación, en la Figura 10 se presenta un fragmento de código del componente *SelectJobs.vue*().

```
  <v-card :loading="pending" elevation="1" class="mx-auto" outlined>
<v-card-title>Minutos</v-card-title>
<v-card-text>
  <v-form ref="form" v-model="valid" lazy-validation>
    <v-row>
      <v-col>
        <v-select
          v-model="task.minutes" outlined
          :items="minOption"
          :menu-props="{ maxHeight: '200' }"
          label="Selecciona"
          multiple
          :disabled="pending || task.minutesAll"
        ></v-select>
      </v-col>
```

Figura 10. Valores de atributos HTML.

Fuente: elaboración propia.

Componente data

El componente *data* debe ser una función. Al usar la propiedad *data* en un componente (es decir, en cualquier lugar excepto en *new Vue*), el valor debe ser una función que devuelve un objeto. A continuación, en la Figura 11 se presenta un fragmento de código del componente *SelectJobs.vue()*.

```
data() {
  return {
    pending: false,
    valid: true,
    e3: [],
    task: {
      minutes: [],
      hours: [],
      dom: [],
      mon: [],
      dow: [],
      user: "",
      command: "",
},
    userOption: ["sue"],
    minOption: MIN_OPTION,
    hourOption: HOUR_OPTION,
    domOption: DOM_OPTION,
    monOption: MES_OPTION,
    dowOption: DOW_OPTION,
    commandOption: ""
  };
 }
}
```

Figura 11. Componente data.

Fuente: elaboración propia.

Identación

Se utilizan 4 espacios sin caracteres de tabulación o espacios en blanco.

Abreviaciones de directivas

Las abreviaciones de directivas (**:** para *v-bind* y @ para *v-on*) deben ser utilizadas siempre o nunca**.**

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

3.2.2 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, formularios o un puntero. Su principal objetivo es facilitar la comunicación con el ordenador (34). En la Figura 12 se muestra la interfaz de usuario correspondiente al RF 7 Listar tareas programadas.

Figura 12. Interfaz de usuario correspondiente al RF 7 Listar tareas programada.

Fuente: elaboración propia.

3.3 Diagrama de despliegue

Es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos de despliegue. Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema (48). Describen la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. A continuación, en la Figura 13 se muestra el diagrama de despliegue de la propuesta de solución.

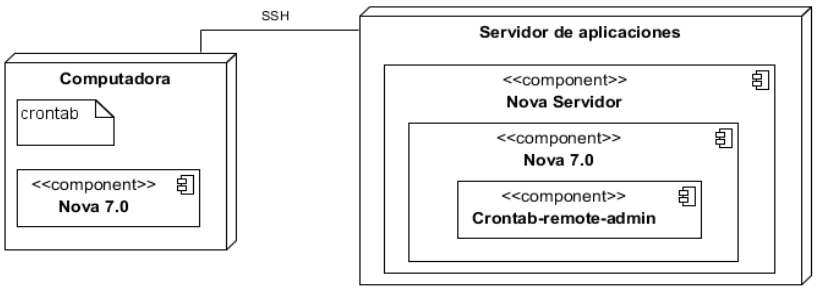


Figura 13. Diagrama de despliegue.

Fuente: elaboración propia.

A continuación, se muestra una explicación de los nodos y componentes representados en la Figura 16.

Descripción de los nodos:

- ✓ Computadora: estación de trabajo en la cual va a estar instalada la distribución cubana GNU/Linux Nova 7.0 y sobre la cual se va realizar la configuración de las tareas programadas.
- ✓ Servidor de aplicaciones: dispositivo en el cual se encuentra instalada la herramienta para administrar remotamente las tareas programadas.

Descripción de los componentes:

- ✓ Nova 7.0: distribución cubana de GNU/Linux Nova, que se encuentra como sistema operativo en las estaciones de trabajo (computadoras).
- ✓ Nova Servidor: distribución cubana GNU/Linux Nova para servidores.
- ✓ *Crontab-remote-Admin*: herramienta que permite administrar remotamente las tareas programadas del *cron.*
- ✓ *Crontab*: fichero de configuración manual del sistema operativo (Nova 7.0).

Descripción del protocolo:

- ✓ SSH (*Secure Shell,* Protocolo Seguro): protocolo que permite el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada (49).

3.4 Pruebas de software

Las pruebas de software consisten en la verificación del comportamiento de un programa en un conjunto finito de casos de prueba con diferentes ejecuciones. Consisten en una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación y calidad de un programa informático; probando el comportamiento del mismo (50). A continuación, se describen los tipos de pruebas, métodos y técnicas que fueran realizadas a la propuesta de solución.

3.4.1 Tipos de pruebas de software

Pruebas de unidad

Las pruebas de unidad descomponen las funciones del programa en comportamientos comprobables discretos que se pueden probar como unidades individuales. Están destinadas a verificar las unidades más pequeñas del software. Se aplican a las funcionalidades para verificar que los flujos de control y datos están cubiertos y funcionan tal como se espera (51)

Pruebas de funcionalidad

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 programación; dichas pruebas se llevan a cabo a través de la interfaz gráfica de la aplicación. Mediante estas pruebas se demuestra que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, además de mantener íntegra la información externa del sistema (34)

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 (14)

3.4.2 Métodos de pruebas

Método de caja blanca

El método de Caja blanca se enfoca en probar el sistema teniendo en cuenta la estructura interna del mismo. Verifica la correcta implementación de las unidades internas, las estructuras y sus relaciones y hacen énfasis en la reducción de errores internos (34).

Método de caja negra

Las pruebas de caja negra o pruebas funcionales se aplican sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. A través de estas pruebas se pretende demostrar que las funcionalidades del software son operativas; las entradas son aceptadas de manera correcta (43).

3.4.3 Técnicas de prueba

Camino básico

La técnica del camino básico tiene como objetivo comprobar que cada camino se ejecute de manera independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. 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 la herramienta (52).

Partición de equivalencia

La técnica de partición de equivalencia consiste en probar cada una de las funcionalidades de la herramienta, mediante la definición de casos de prueba que sean capaces de encontrar diferentes tipos de errores a nivel de interfaz. Es un conjunto de estados válidos y no válidos para las condiciones de entrada. 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 (34).

3.5 Aplicación de las pruebas de software

Este epígrafe muestra cómo se aplicaron las pruebas de software, métodos y técnicas que fueron mencionados en el epígrafe 3.4 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.

3.5.1 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 (14). Durante esta disciplina se aplicaron las pruebas unitarias y funcionales, las cuales se describen a continuación.

Pruebas unitarias

Las pruebas unitarias se realizaron mediante la aplicación del método caja blanca y la técnica del 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 de la herramienta. 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 (34).

Pruebas funcionales

Las pruebas funcionales se realizaron mediante el método de caja negra y partición de equivalencia. Para dicha técnica se retomaron los casos de pruebas realizados en el epígrafe 2.2.5 donde se definen las variables válidas y no válidas para cada entrada del sistema. La Figura 14 muestra un gráfico con los resultados de las pruebas funcionales.

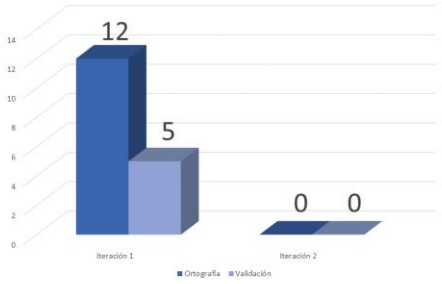


Figura 14. Resultado de las pruebas funcionales.

Fuente: elaboración propia.

Las pruebas funcionales se realizaron en 2 iteraciones, en la primera se identificaron 17 no conformidades, de ellas 12 de errores ortográficos y 5 de validación. Durante la segunda iteración no se identificaron no conformidades.

Pruebas de aceptación

El objetivo de esta prueba es encontrar los errores y las omisiones en la definición de los requisitos del sistema, debido a que los datos reales ejercitan el sistema en diferentes formas a partir de datos de prueba (53). A continuación, se muestran en la Tabla 12 el caso de prueba realizado por el cliente a la Historia de Usuario (HU) correspondiente al RF1 Conectar remotamente.

Caso de prueba de aceptación
Nombre de la historia de usuario: Conectar remotamente
Nombre de la persona que realiza la prueba: Yasiel Perez Villazón
Descripción de la prueba
Condiciones de ejecución: Tener instalado el protocolo SSH. Las credenciales para los campos de ip del servidor, usuario, y contraseña deben ser correctas.
Entrada/Pasos de ejecución: 1. Acceder a la interfaz principal herramienta 2. Seleccionar la opción Conectar del menú superior
Resultado esperado: La herramienta permite la conexión remota.
Evaluación de la prueba: Satisfactoria

Tabla 12. Caso de prueba de aceptación para la Historia de Usuario Conectar remotamente.

Fuente: elaboración propia.

Como resultado de la aplicación de las pruebas de aceptación se obtuvo una evaluación satisfactoria por parte del cliente. Lo que evidencia el cumplimiento de los objetivos propuesto para la realización de esta prueba.

Con la finalidad de demostrar el cumplimiento de los requerimientos de la propuesta de solución en la Tabla 13 se muestra la comparación entre las aplicaciones informáticas para la administración de tareas programadas estudiadas al comienzo de la investigación y la propuesta de solución.

Tabla 13. Análisis comparativo entre aplicaciones de administración remota de tareas programadas.

Fuente: elaboración propia.

Criterios para el análisis	**Sistemas informáticos para la administración de tareas programadas**			
	Crontab-ui	Webmin	Nova NAS	**Propuesta de solución**
Basada en software libre	Sí	Sí	Sí	**Sí**
Integración con Nova 7.0	No	Sí	Sí	**Sí**
Administración remota por protocolo SSH	Sí	Sí	Sí	**Sí**
Totalidad de parámetros del *cron*	Sí	Sí	No	**Sí**
Detección de cambios manuales en los ficheros de configuración del *cron*	Sí	No	Sí	**Sí**

Al comparar las aplicaciones estudiadas con la herramienta informática desarrollada en la presente investigación para administrar de forma remota las tareas programadas, se arroja como resultado que esta es superior en cuanto a los criterios tenidos en cuenta. De esta forma se denota el aporte de la propuesta de solución para cubrir necesidades de Nova 7.0.

3.6 Conclusiones del capítulo

Se logró la informatización del servicio de administración de tareas programadas en Nova 7.0 a partir de la herramienta desarrollada. La aplicación de las pruebas de unidad, funcionales y de aceptación permitieron detectar las no conformidades que afectaban el correcto funcionamiento la propuesta de solución, lo que posibilitó corregirlas a tiempo y verificar la calidad del producto.

CONCLUSIONES GENERALES

Con la realización de la presente investigación se brinda solución a los objetivos trazados, obteniéndose como principales resultados:

- ✓ El análisis del marco teórico referencial sobre la administración de tareas programadas en GNU/Linux y el estudio de aplicaciones informáticas que permiten este servicio demostraron la necesidad de implementar una herramienta que permita la administración remota de tareas programadas en Nova 7.0.
- ✓ Se definieron 9 requisitos funcionales y 3 no funcionales que permitieron satisfacer las necesidades del cliente y las partes interesadas. El análisis y diseño posibilitó el desarrollo de una herramienta para administrar de forma remota las tareas programadas en Nova 7.0.
- ✓ El análisis de la arquitectura y la aplicación de patrones de diseño permitieron comprender los aspectos relacionados con los requisitos del sistema, descomponer los trabajos de la implementación en partes creando así un punto de partida para la implementación.
- ✓ La aplicación de las pruebas permitió corregir las no conformidades detectadas, validando así todas las funcionalidades concebidas por el cliente y logrando una herramienta de escritorio con calidad y seguridad.
- ✓ Como resultado de la implementación se obtuvo una herramienta que necesita de pocos recursos y le permite realizar al usuario una mejor configuración de las tareas programadas en Nova 7.0 sin necesidad de acceder a la introducción de líneas de comando mediante la consola.

RECOMENDACIONES

A partir de la experiencia obtenida en el desarrollo de la investigación, la autora recomienda:

- ✓ Utilizar el contenido de la investigación como base de referencia para el desarrollo de futuras herramientas para administrar tareas programadas en otras versiones Nova y en otras distribuciones de GNU/Linux.
- ✓ La implementación de nuevas funcionalidades que permitan gestionar los archivos de configuración *Cron.deny* y *Cron.allow*.
- ✓ Crear nuevas funcionalidades que le brinde a la herramienta la tolerancia a fallos y la recuperación ante errores.

.

REFERENCIAS BIBLIOGRÁFICAS

1. Alejandro Rodríguez Toledo, Orlando Viera Pérez. *Módulo para la ejecucuión automática scripts en los clientes desde el servidor de GRHS.* La Habana : s.n., Módulo para la ejecucuión automática scripts en los clientes desde el servidor de GRHS.

2. Tamay, Yoel Miyares. *Administración del servicio antivirus desde la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST).* La Habana : s.n., 2014.

3. Alcolea, Wilbia Esther Mariño. *Módulo de planificación de tareas para la Herramienta de Migración y Administración de Servicios Telemáticos.* La Habana : s.n., 2017.

4. Webmin. [En línea] 2016. [Citado el: 2 de Diciembre de 2019.] http://www.webmin.com/intro.html..

5. Wallen, Jack. TechRepublic. How to installcrontab-ui for remote use. [En línea] Abril de 2015. https://www.techrepublic.com/article/how-to-install-crontab-ui-for-remote-use/.

6. Libre, Centro de Software. *Administración de un servidor de almacenamiento conectado a la red con Nova.* La Habana : s.n., 2015.

7. Alestra, Staff. Alestra Blog. *Herramientas de administración remota de desktop.* [En línea] 10 de Febrero de 2016. [Citado el: 19 de Octubre de 2019.] http://blog.alestra.com.mx/herramientas-de-administraci%C3%B3n-remota-de-desktop.

8. Fabricio, Salazar Zabala Alexis. *Sistema Administativo de ETFs (EXCHANGE TRADAED FUND'S) basado en servicios Cron Jobs para la empresa INDEXIQ, LL.de la ciudad de NEW YORK.* Ecuador : s.n., 2016.

9. Crespo, Adrian. Redeszone. *Cómo utilizar Cron y Crontab en Linux para programar tareas.* [En línea] 9 de Enero de 2017. [Citado el: 19 de Octubre de 2019.] https://www.redeszone.net/2017/01/09/utilizar-cron-crontab-linux-programar-tareas/.

10. Prieto, Raul Fernández. Raul Prieto Fernández. *¿Cómo programar tareas en GNU/Linux?* [En línea] 26 de Febrero de 2017. [Citado el: 2 de Diciembre de 2019.] https://www.raulprietofernandez.net/blog/gnu-linux/como-programar-tareas-en-gnu-linux.

11. González, Sergio Durán. LinuxTotal.com.mx. [En línea] 2015. [Citado el: 19 de Noviembre de 2019.] http://www.linuxtotal.com.mx/?cont=info_admon_006/..

12. Roland Mas, Raphael Hertzog. *El manual del Administrador de Debian.Debian Jessie desde el descubrimiento a la maestría.* 2016. ISBN.

13. Pabón, Perdo Elías. SMARTSOFT. *Metodologia de desarrollo Tradicional RUP.* [En línea] 31 de Julio de 2018. [Citado el: 19 de Noviembre de 2019.] https://smartsoftcolombia.com/portal/index.j.

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

15. Beltrán, Pedro. ¿Qué es una herramienta case? [En línea] [Citado el: 19 de Noviembre de 2019.] -- https://www.academia.edu/28037284/Qu%C3%A9_es_una_herramienta_CASE.

16. González, Laura Elena Varela. *Componente de cifrado de características biométricas utilizando plantillas cancelables.* La Habana : s.n., 2017.

17. *Visual Paradigm.* [En línea] 27 de Febrero de 2018. [Citado el: 2019 de Noviembre de 19.] https://www.visual-paradigm.com-aboutus/newsreleases/.

18. Lucid Software Inc. Lucidchart. *Qué es un modelo de base de datos.* [En línea] Diciembre de 2019. [Citado el: 2 de Diciembre de 2019.] https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos.

19. González, Anabel Valiente. *Entorno de Desarrollo Integrado para el lenguaje ensamblador.* La Habana : s.n., 2015.

20. Pérez, Julián y Gardey, Ana. Definiciones.de. *Definicion de HTML.* [En línea] Octubre de 2018. [Citado el: 28 de febrero de 2020.] https://definicion.de/html/.

21. W3C. W3C. *Cascading Style Sheets.* [En línea] [Citado el: 2 de Diciembbre de 2019.] https://www.w3.org/Style/CSS/specs.en.html.

22. Pérez, Javier Eguíluz. Introduccion a JavaScript. [En línea] [Citado el: 18 de febrero de 2020.] https://www.jesusda.com/docs/ebooks/introduccion_javascript.pdf.

23. *Marco de trabajo para gestionar actividades de calidad.* Aymara Marin Diaz, Yaimí Trujillo Casañola,Denys Buedo Hidalgo. 2, La Habana : Revista de Ciencias Informáticas, 2018, Vol. 12. ISSN.

24. *Electron Documentation.* [En línea] [Citado el: 19 de Noviembre de 2019.] https://www.electronjs.org/docs.

25. Pérez., Eduardo Ismael García. Programar es el futuro. *¿Qué es Vue.JS?* [En línea] 1 de Abril de 2019. [Citado el: 1 de Febrero de 2020.] https://codigofacilito.com/articulos/que-es-vue .

26. Vuetify. *Why vuetify.* [En línea] [Citado el: 18 de Febrero de 2020.] https://vuetifyjs.com/es-MX/introduction/why-vuetify.

27. LLamas, Luis. Luis LLamas. Ingienería, informática y diseño. *VUETIFY, ESTÉTICA MATERIAL DESIGN PARA TUS APPS EN VUEJS.* [En línea] 21 de Enero de 2019. [Citado el: 1 de Febrero de 2020.] https://www.luisllamas.es.

28. Pila, Luis Ernesto Zorrilla. *Portal web del Observatorio Tecnológico de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2016.

29. Visual Studio Code. [En línea] Enero de 2020. [Citado el: 17 de Febrero de 2020.] https://code.visualstudio.com/updates/v1_42.

30. andrearr. Hipertextual. *Qué es un sistema de control de versiones y por qué es tan importante.* [En línea] 30 de Abril de 2014.

31. CHACON, SCOTT y STRAUB, BEN. *Pro Git.* segunda edición. s.l. : Apress, 2014.

32. Alonso, Lexys Manuel Díaz. *Módulo para el monitoreo en tiempo real de clientes ligeros desde Nova-LTSP.* La Habana : s.n., 2017.

33. Pérez, Osiris. *Técnicas de obtención de requisitos.* La Habana : s.n., 2017.

34. Pressman, Roger S. *Ingeniería de software. Un enfoque práctico.* séptima edición. s.l. : McGrawHill, 2010. ISBN: 978-607-15-0314-5.

35. Cárdenas, Javier Piñeiro. *Herramienta web para la creación de personalizaciones de Nova Servidores.* La Habana : s.n., 2017.

36. Guzman, Adrian Alberto Thondique. *Componente para imputar datos en Pentaho Data Integration.* La Habana : s.n., 2015.

37. Martínez, Aimet Cabrera. *Módulo de procesamiento estadístico para el apoyo a la toma de decisiones del motor de búsqueda Orión.* La Habana : s.n., 2017.

38. Borrero, Marielsis Alina Hernández. *Aplicación informática para contribuir a la prevención de enfermedades profesionales en los informáticos.* s.l. : La Habana, 2016.

39. Pérez, Osiris. *Descripción de los procesos de validación y administración de requisitos.* Ciudad de La Habana : s.n., 2016.

40. Aquia, Shadet Asnay Ariosa. *Módulo para el cifrado de los datos informáticos almacenados en NovaNAS.* La Habana : s.n., 2019.

41. Torres, Yesenia de la Caridad Valdés. *Sistema para administración de documentos pdf en android(Sunsetdroid).* La Habana : s.n., 2017.

42. Palacios, Alejandro Campos. *Módulo de complementos para el servicio proxy de HMAST.* La Habana : s.n., 2017.

43. Rodríguez, Miosotis Toledo. *Módulo para la gestión de copias de seguridad en Nova 360.* La Habana : s.n., 2019.

44. Santiago, Felipe González y Gonzále, Yosel Lázaro Vera. *MÓDULO DE HMAST PARA LA ADMINISTRACIÓN Y MIGRACIÓN MÓDULO DE HMAST PARA LA ADMINISTRACIÓN Y MIGRACIÓN.* La Habana : s.n., 2016.

45. Francisco. Una Universidad Nacional Costa RIca . *Manual técnico BD Oracle.* [En línea] 2016. [Citado el: 19 de Febrero de 2020.] https://manual-tecnico-bd-oracle.readthedocs.io/es/latest/Modelo%20de%20datos.html.

46. Valderrama, Norlys Suárez. *Herramienta para detectar cambios en el código fuente del Sistema de Gestión para el Ingreso a la Educación Superior (SIGIES).* La Habana : s.n., 2017.

47. Gerchev, Ivaylo. Sitepoint.com. [En línea] 23 de Julio de 2019. [Citado el: 7 de Junio de 2020.] https://www.sitepoint.com/vue.js-tools-libraries.

48. SlideShare . *Diagrama de despliegue.* [En línea] 24 de Febrero de 2016. [Citado el: 18 de Marzo de 2020.] La interfaz de usuario es una parte de un programa que gestiona la interacción con el usuario basándose.

49. Raiola Networks. *Guía básica para SSH: Qué es, cómo instalarlo y cómo utilizarlo correctamente.* [En línea] 26 de Febrero de 2020. [Citado el: 7 de Junio de 2020.] https.//raiolnetworks.es/blog/ssh/.

50. ZAPATA, JAVIER. *Introducción a las pruebas.* 2013.

51. Pressman, Roger S. *Ingeniería de software: Un enfoque práctico.* Sexta Edición. Nueva York. : Estados Unidos: McGraw-Hill / Interamericana de México, 2005.

52. Sommerville, Ian. *Software Engineering.* novena edición. s.l. : Person Education Inc,2011, 2011.

53. Presman, Roger S. *Ingeniería de software: Un enfoque práctico.* 2002.

56. Márquez, Odalys Rosa Falcón. UCI . *Nova disponible para todos los internautas.* [En línea] 2 de Octubre de 2017. [Citado el: 19 de Noviembre de 2019.] https://www.uci.cu/universidad/noticias/nova.

ANEXOS

Anexo 1. Entrevista realizada a los especialistas del Centro de Software Libre (CESOL)

Objetivo: conocer el contexto del negocio, las tecnologías y las herramientas compatibles con dicho proyecto, así como obtener información sobre las características que debe tener la herramienta para poder ser integrada con Nova 7.0.

1. ¿Qué es Nova 7.0?
2. ¿Cuáles son las características de Nova 7.0?
3. ¿Qué sistemas se integran con Nova 7.0?
4. ¿Cómo se lleva a cabo actualmente el servicio de administración de tareas programadas?
5. ¿Qué características debe tener la herramienta a desarrollar?

Anexo 2. Encuesta realizada a los especialistas del Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI)

Objetivo: evaluar el nivel de satisfacción de los usuarios potenciales de la propuesta de solución.

El siguiente cuestionario permitió obtener información sobre criterios que ayudaron al desarrollo de la investigación. Marque con una X la opción que considere acertada, en otro caso responda brevemente.

1. ¿Actualmente Nova 7.0 satisface las necesidades para el servicio de administración remota de tareas programadas?

Sí__ No__ No sé __

2. ¿Considera necesario el uso de herramientas informáticas para administrar de forma remota las tareas programadas en Nova?

Sí__ No__ No sé __

3. Indique en qué medida le gustó el desarrollo de la herramienta de la propuesta de solución.

No me gusta mucho__ Me gusta poco__ Me gusta mucho __

Me es indiferente __ No sé __

4. ¿Qué beneficios considera usted que trae consigo el desarrollo de una herramienta para administrar de forma remota las tareas programadas en Nova 7.0?

Anexo 3. Guía de observación para obtener información acerca de aplicaciones informáticas que administran el servicio de tareas programadas en GNU/Linux

Observadora: Sueyaile Toledo Ruiz.

Lugar: Laboratorio de práctica profesional del centro CESOL.

Objetivo: identificar las características y funcionalidades principales para la administración de tareas programadas.

1. Información sobre aplicaciones informáticas que manejan el servicio de administración de tareas programadas.
   - ✓ Nombre de las aplicaciones.
   - ✓ Cuentan con interfaz gráfica.
   - ✓ Compatibilidad con Nova 7.0.
2. Características de las aplicaciones informáticas que permiten la administración de tareas programadas en GNU/Linux.
   - ✓ ¿Cómo realizan las aplicaciones el servicio de administración de tareas programadas?
   - ✓ ¿Cómo son las aplicaciones que emplean este servicio?
   - ✓ ¿Qué funcionalidades brindan estas aplicaciones para administrar las tareas programadas en GNU/Linux?

Anexo 4. Listado de especialistas a los que se le realizaron las entrevistas

Tabla 14. Listado de especialistas entrevistados del Centro CESOL.

Fuente: elaboración propia.

Nombre	**Nivel académico y científico**	**Rol**	**Cargo que ocupa**	**Años de antigüedad**

Yasiel Pérez Villazón	Máster	Programador		
Yosel Lázaro Vera González	Ingeniero	Programador		4
Ivaniet Díaz Romeu	Ingeniero			
Enrique Muschett Cortina	Ingeniero	Programador		
Lexys Manuel Díaz Alonso	Ingeniero	Programador		3
Nurisel Palma Pérez	Máster			

Anexo 5. Historias de Usuario correspondientes a los requisitos de software RF1, 2, 3, 4, 6, 7, 8 y 9

Tabla 15. Historia de usuario correspondiente al RF1 Conectar remotamente.

Fuente: elaboración propia.

Historia de Usuario	
Número: 1	**Nombre del requisito:** Conectar remotamente
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo en desarrollo: No aplica	**Tiempo real:** 1 hora
Descripción: La herramienta permite al usuario conectarse remotamente desde una computadora a otra. El usuario selecciona la opción de “conectar” que aparece en la pantalla de inicio de la herramienta y automáticamente se muestra una interfaz para introducir el IP, usuario y contraseña de la maquina ala que vas acceder.	
Observaciones: Tener instalado el protocolo SSH.	
Prototipo de Interfaz	


ADMINISTRACION REMOTA | SISTEMA
Conectado a 127.0.0.1
HERRAMINETA DE ADMINISTRACIÓN REMOTA
Módulo Si
Este módulo permi
del sistema y la cre
ÚLTIMAS CONEXIONES
iones recientes
Conectar al servidor
SERVIDOR
127.0.0.1
9/15
USUARIO
sue
CONTRASEÑA
Guardar parametros de conexión
CERRAR
CONECTAR
Nova AST © 2020
ADMINISTRACION REMOTA | SISTEMA
Conectado a 127.0.0.1
HERRAMINETA DE ADMINISTRACIÓN REMOTA
Módulo Sistema
Este módulo permite la administración
del sistema y la creación de tareas programdas
ÚLTIMAS CONEXIONES
127.0.0.1
Conectado como sue
Mon, Mar 9, 2020 3:07 PM
Nova AST © 2020

Tabla 16. Historia de usuario correspondiente al RF2 Iniciar servicio cron.

Fuente: elaboración propia.

Historia de Usuario	
Número:2	**Nombre del requisito:** Iniciar servicio *cron*
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo en desarrollo: No aplica	**Tiempo real:** 1 hora
Descripción: La herramienta permite al usuario iniciar el servicio *cron*. El usuario selecciona la opción “Iniciar” que aparece en la pantalla de inicio de la herramienta y automáticamente se muestra en pantalla el estado del servicio que indica si está activado o no.	
Observaciones: Para que se pueda iniciar el servicio *cron* se utiliza el comando *server cron start.*	
Prototipo de Interfaz	

Tabla 17. Historia de usuario correspondiente al RF3 Reiniciar servicio cron.

Fuente: elaboración propia.

Historia de Usuario	
Número: 3	**Nombre del requisito:** Reiniciar servicio *cron*
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo en desarrollo: No aplica	**Tiempo real:** 1 hora
Descripción: La herramienta permite al usuario reiniciar el servicio *cron*. El usuario selecciona la opción “Reiniciar” que aparece en la pantalla de inicio de la aplicación y automáticamente se muestra en pantalla el estado del servicio que indica si está activado o no.	

Observaciones:

Para que se pueda reiniciar el servicio *cron* se utiliza el comando *server cron restar.*

El servicio debe haber sido iniciado con anterioridad.

Prototipo de Interfaz

Tabla 18. Historia de usuario correspondiente al RF4 Detener servicio cron.

Fuente: elaboración propia.

Historia de Usuario	
Número: 4	**Nombre del requisito:** Detener servicio *cron*
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora

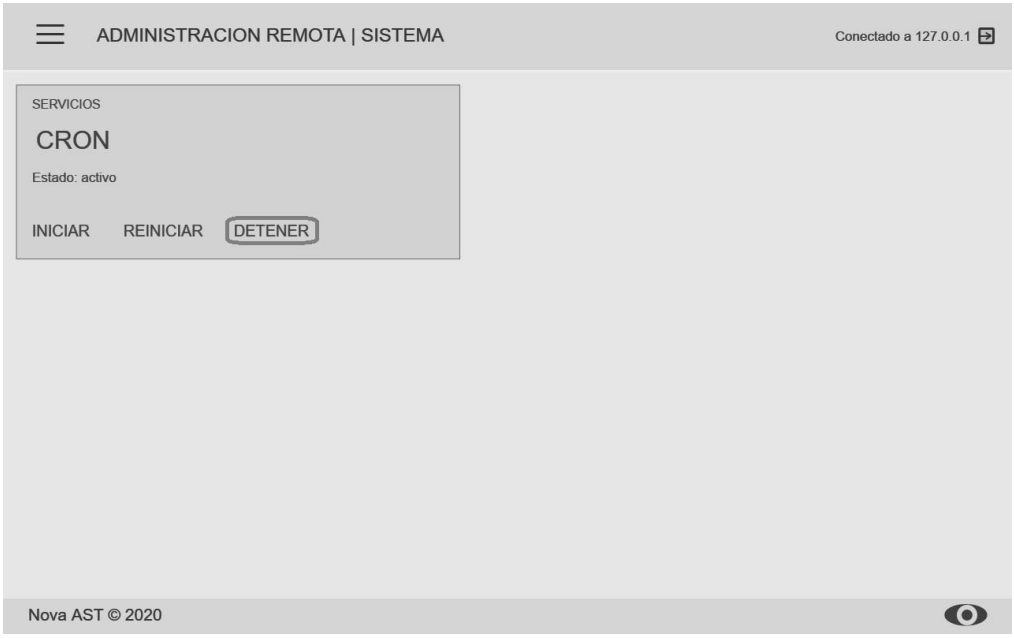
Riesgo en desarrollo: No aplica	**Tiempo real:** 1 hora
Descripción: La herramienta permite al usuario detener el servicio *cron*. El usuario selecciona la opción “Detener” que aparece en la pantalla de inicio de la aplicación y automáticamente se muestra en una interfaz con el estado del servicio que indica si está detenido o no.	
Observaciones: Para que se pueda detener el servicio *cron* se utiliza el comando *server cron stop.* El servicio debe haber sido iniciado con anterioridad.	
Prototipo de Interfaz 	

Tabla 19. Historia de usuario correspondiente al RF6 Mostrar tarea programada.

Fuente: elaboración propia.

Historia de Usuario

Número: 6	**Nombre del requisito:** Mostrar tarea programada
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 24 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 24 horas
Descripción: La herramienta muestra al usuario cada una de las tareas que han sido creadas hasta el momento con su respectiva información sobre el minuto, la hora, el día de la semana, el día del mes, el mes, el usuario y el comando que se ejecuta en la tarea.	
Observaciones: Tiene que existir al menos una tarea creada.	
Prototipo de Interfaz 	

Tabla 20. Historia de usuario correspondiente al RF7 Listar tareas programadas.

Fuente: elaboración propia.

Historia de Usuario	
Número: 7	**Nombre del requisito:** Listar tareas programadas
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 24 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 24 horas
Descripción: La herramienta muestra al usuario un listado de todas las tareas que han sido creadas hasta el momento con su respectiva información sobre el minuto, la hora, el día de la semana, el día del mes, el mes, el usuario y el comando que se ejecuta en la tarea.	
Observaciones: Tiene que existir al menos una tarea creada, sino muestra una lista vacía.	
Prototipo de Interfaz	

Tabla 21. Historia de usuario correspondiente al RF8 Editar tarea programada.

Fuente: elaboración propia.

Historia de Usuario	
Número: 8	**Nombre del requisito:** Editar tarea programada
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 24 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 24 horas

Descripción:

La herramienta le brinda al usuario la posibilidad de editar una tarea anteriormente creada. El usuario selecciona del listado de tareas, la tarea que desea editar y selecciona la opción ¨Editar´, que aparece al lado de la tarea y se encuentra representada por una figura de un lápiz. Una vez seleccionada esta opción se activan los campos de selección de tiempo (minuto, hora, día de la semana, día del mes, mes), además de los campos de usuario y comando. Si el usuario se arrepiente de editar la tarea aparece la opción “Regresar” para volver a la vista anterior. Una vez realizado los cambios el usuario puede seleccionar la opción de “Guardar” para cambiar los parámetros de la tarea programada.

Observaciones:

La tarea a editar debe existir.

Prototipo de Interfaz

Tabla 22. Historia de usuario correspondiente al RF9 Eliminar tarea programada.

Fuente: elaboración propia.

Historia de Usuario	
Número: 9	**Nombre del requisito:** Eliminar tarea programada
Programador: Sueyaile Toledo Ruiz	**Iteración:** 1
Prioridad: Alta	**Tiempo estimado:** 24 horas
Riesgo en desarrollo: No aplica	**Tiempo real:** 24 horas
Descripción: La herramienta le brinda al usuario la posibilidad de eliminar una tarea. El usuario selecciona del listado de tareas, la que desea eliminar y presiona la opción ¨Eliminar´, que aparece al lado de la tarea y se encuentra representada por una figura de un cesto de basura. Luego la herramienta muestra un mensaje de confirmación para eliminar o no la tarea seleccionada.	
Observaciones: La tarea a eliminar debe existir.	
Prototipo de Interfaz	


ADMINISTRACION REMOTA | SISTEMA
Conectado a 127.0.0.1
Lista de Tareas Programadas
CREAR
Minutos
Horas
Días del mes
Mes
Días de la semana
Usuario
Comando
Eliminar
15,20
12
25
enero,marrzo,oct...
lunes,martes,miercole...
ROOT
/usr/respald...
Rows per page:
10
Nova AST © 2020