Universidad de las Ciencias Informáticas

Módulo para la gestión de copias de seguridad en Nova 360

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

Autora: Miosotis Toledo Rodríguez

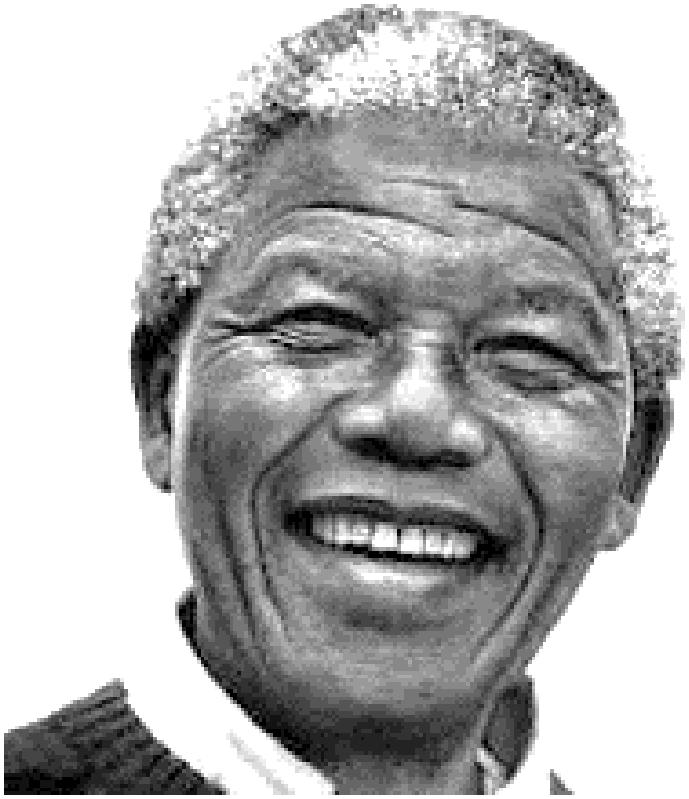
Tutores:

Ing. Ivaniet Díaz Romeu

Ing. Yosel Lázaro Vera Gonzáles

Co-tutor: Ing. Juan Alberto Pereira Delgado

La Habana, 24 junio del 2019



Siempre parece imposible hasta que se hace.

Nelson Mandela

Declaración de autoría

Declaro por este medio que yo **Miosotis Toledo Rodríguez**, con carné de identidad **96041107859** soy la autora principal del trabajo titulado "**Módulo para la gestión de copias de seguridad en Nova 360**" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

_____

Miosotis Toledo

Autora

_____ _____

Ing. Ivaniet Díaz Romeu

Tutor

Ing. Yosel Lázaro Vera González

Tutor

_____

Ing. Juan Alberto Pereira Delgado

Co-tutor

AGRADECIMENTOS

Al comenzar a escribir los agradecimientos me di cuenta que son muchas las personas a las que les debo el estar hoy aquí, obteniendo finalmente un título para mi futuro. No fue de música como una vez creímos, fue de informática como nunca imaginamos.

Por eso le agradezco a mi mamá, mi abuela, mi hermano y mi abuelo por ser mi ejemplo y mi sustento, por todo el apoyo que siempre me dieron en cada una de mis decisiones. Por sus ideas, pero sobre todo por estar siempre conmigo llevándome de la mano para que no perdiera el camino.

A Juan por los cuatro años y seis meses que me ha dedicado, por cada una de las ayudas que me ha prestado, por su paciencia ante cada uno de mis berrinches y lloraderas, por las sonrisas, las peleas y los consejos que siempre me ha dado.

A mis tutores Ivaniet y Yosel por las horas que me dedicaron sin importar el momento ni el día; siempre con sus opiniones y sugerencias bien recibidas y acertadas.

A René, por su ayuda incondicional en cuanto a bash y Linux se refiere, así como cada una de sus críticas, comentarios e ideas para mi aplicación.

A todos y cada uno de mis profesores durante estos cinco años por sus enseñanzas, hasta las más pequeñitas de ellas, pero en especial a Manolo, Yaima, Yanet, Israel, Yurisbel, Ponce, Delly.

A Daniela y Mariana, porque en ellas siempre encontré un abrazo amigo recordándome que nunca estaba lejos de casa.

A Lissandra, por tener y decir siempre las ideas más ocurrentes.

A Luisito, Leo, Carla, Carel y Mena, por las mejores horas de estudio.

A Ignacio solo por ser Superignac, mi amigo y a Gómez por todos sus comentarios.

Al Kuki por entretener nuestros turnos de clase con sus magistrales dones de actuación.

A mi papá y mis padrinos por ser únicos y sobre todo por el cariño que siempre me han dado.

A la familia de Matanzas, por todas sus palabras de ánimo.

A todas las personas que, aunque ahora no las mencione, de una forma u otra siempre han estado ahí para mí.

Agradecimientos

A todos y cada uno de ustedes, muchas gracias, simple y sencillamente por existir.

Gracias,

Miosotis.

DEDICATORIA

A mi mamá, la más soberana de todas las madres;

a mi abuela, la más poderosa de todas las abuelas;

a mi hermano, el mejor de todos los hermanos;

a mi abuelo, el más preocupado de todos los abuelos;

a mis tutores, los más dedicados de todos los tutores;

y al primer FI07 de la UCI, por ser mi grupo más querido.

RESUMEN

La información digital está en constante crecimiento y tal aumento exige grandes capacidades de almacenamiento. La computación en la nube, a través de Internet, ha facilitado el consumo de estos volúmenes de información mediante el uso de aplicaciones como OwnCloud y NextCloud. Cuba en los últimos años ha avanzado considerablemente en el uso de la Tecnologías de la Información y las Comunicaciones como parte de la estrategia de informatización de la sociedad. Con el objetivo de lograr la soberanía tecnológica, se desarrolla un proceso de migración hacia software libre en el que se emplean soluciones informáticas desarrolladas por el Centro de Software Libre, perteneciente a la Universidad de las Ciencias Informáticas. Nova 360 es una de estas soluciones; tiene como objetivos ofrecer servicios en la nube, permitir la sincronización con múltiples dispositivos de datos y compartir archivos de forma pública. Sin embargo, para realizar la gestión de sus copias de seguridad es necesario el uso de comandos en la terminal, lo cual representa una dificultad para aquellos usuarios que no tienen conocimientos avanzados de los comandos de Linux.

La presente investigación tuvo como objetivo desarrollar un módulo que gestione las copias de seguridad de Nova 360, para ello se utilizaron tecnologías en su mayoría libres, diferentes patrones de diseño y la arquitectura Modelo-Vista-Controlador. Con las pruebas aplicadas se comprobó el correcto funcionamiento de Salvas, demostrando que el mismo satisface las necesidades del cliente.

Palabras claves: copias de seguridad, información digital, Nova 360.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La información digital está en constante crecimiento y tal aumento exige disponer de grandes volúmenes de almacenamiento para protegerla, ya que es uno de los bienes más importantes con que cuentan los seres humanos para el desarrollo del conocimiento. La evolución de Internet ha venido acompañada de la posibilidad de consumir estos altos volúmenes de información situados en servidores ubicados en diversos lugares del planeta (MORRISON, 2016).

De acuerdo con un estudio presentado en la revista *Science* desde el 2007 el 99,9% de la información que se genera se encuentra en formato digital mientras que solo el 0,007% de la información del planeta está en papel (HILBERT, 2011); cabe destacar que en el 2016 se manejaban 2.5 quintillones de bytes en datos, lo que equivale a $2.5*10^9$ zettabytes (MARR, 2018). Una de las tecnologías informáticas que ha contribuido con este propósito es la computación en la nube; posibilitando el acceso y manejo de esta información desde cualquier dispositivo electrónico en cualquier región del mundo.

La computación en la nube es un modelo de prestación de servicios de negocios y tecnologías a través de una red informática. Se divide en tres categorías: el Software como Servicio *(*SaaS*, Software as a Service)*, siendo este un modelo de distribución de aplicaciones alojadas por un proveedor de servicio y puestas a disposición de los usuarios; Plataforma como Servicio *(*PaaS*, Platform as a Service)*, que es un conjunto de herramientas para abastecer al usuario de sistemas operativos y servicios asociados sin necesidad de descargas o instalación; e Infraestructura como Servicio *(*IaaS*, Infrastructure as a Service),* que favorece la tercerización de los equipos utilizados para apoyar las operaciones, incluido el almacenamiento, hardware, servidores y componentes de red (TWEEDIE, 2017).

El desarrollo de aplicaciones que ofrecen estos servicios ha tenido un gran auge en la comunidad de software libre, tomando como ejemplo: OwnCloud, OpenStack, NextCloud; las cuales representan una alternativa a aplicaciones propietarias como: Dropbox, iCloud, OneDrive o Google Drive. Estas herramientas ofrecen servicios de nubes personalizadas garantizando la seguridad de los datos almacenados. Permiten hacer copias de seguridad de su sistema de configuración y almacenamiento, las cuales pueden ser restauradas ante un fallo que comprometa la estabilidad del sistema y la pérdida total o parcial de su información (MAURYA, 2018).

La gestión de copias de seguridad permite crearlas, listarlas, restaurarlas, eliminarlas, así como definir, actualizar y eliminar planificaciones acerca de cuándo y en dónde se realizan estas copias de seguridad. Existen diversas aplicaciones basadas en tecnologías libres para la gestión de copias de seguridad tales como Bacula, Box Backup, Simple Backup, entre otras; y también privativas como Carbonite, Easus *Todo* Backup Free y CloudBerry Backup (DragonJar, 2018).

Cuba en los últimos años ha avanzado considerablemente en el uso de las Tecnologías de la Información y las Comunicaciones (TIC) para la gestión de los procesos de los diferentes Organismos de la Administración Central del Estado (OACE) y como política de la estrategia para la informatización de la sociedad cubana. Con el objetivo de garantizar la soberanía tecnológica, en estas entidades se desarrolla una migración hacia software libre. En ella se utilizan soluciones informáticas desarrolladas por el Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI). Entre estas soluciones se encuentran la distribución cubana GNU/Linux Nova (como sistema operativo), Nova Servidores (para brindar servicios telemáticos) y Nova 360 (ofreciendo servicios en la nube).

De acuerdo a una entrevista realizada a los miembros del proyecto que desarrolla Nova 360, se conoció que, esta es una solución informática que pertenece a la categoría de software como servicio del modelo computación en la nube. Es una personalización de Owncloud para las instituciones cubanas y posibilita la sincronización con múltiples dispositivos de datos, calendarios, tareas, archivos multimedia, galería de imágenes y otros. Permite compartir archivos entre usuarios o de forma pública. Su desarrollo está centrado en brindar una solución que integre diferentes tecnologías de Nova tales como: NovaDroid (personalización cubana para móviles con Android) y Nova Escritorio.

La entrevista aplicada permitió conocer además que actualmente los usuarios administradores de Nova 360 para realizar la gestión de copias de seguridad necesitan hacer uso de comandos en la consola del servidor que lo contiene, lo cual representa una dificultad para aquellos usuarios que no dispongan de los conocimientos necesarios en cuanto al uso de los comandos precisos, su sintaxis y funcionamiento. Asimismo, al tener menor experiencia deben emplear mayor tiempo en la escritura de las instrucciones para efectuar las copias de seguridad y la restauración de estas. El empleo de la consola también genera cierta resistencia en los usuarios ya que resulta poco amigable e intuitiva.

La situación descrita anteriormente provoca que este proceso no se pueda realizar con la sistematicidad requerida, ya que es una tarea engorrosa. Además, ante la ocurrencia de un fallo que comprometa la persistencia de los datos, ya no se podría contar con la información almacenada.

A partir de la situación problemática anteriormente planteada, se identificó el siguiente **problema de investigación:** ¿Cómo agilizar el proceso de planificación de respaldo y restauración de la información almacenada en Nova 360?

Se identifica como **objeto de estudio:** el proceso de gestión de copias de seguridad. Se propone como **campo de acción:** las aplicaciones informáticas para la gestión de copias de seguridad en servidores de almacenamiento en la nube.

Se define como **objetivo general**: Desarrollar un módulo para la gestión de copias de seguridad en Nova 360 que agilice el proceso de planificación de respaldo y restauración de la información que tiene almacenada.

Los **objetivos específicos** propuestos para el desarrollo de la investigación son:

1. Elaborar el marco teórico de la investigación sobre el proceso de gestión de copias de seguridad en servidores de almacenamiento en la nube.
2. Diseñar un módulo para la gestión de copias de seguridad en Nova 360.
3. Implementar un módulo para la gestión de copias de seguridad en Nova 360.
4. Evaluar el módulo para la gestión de copias de seguridad en Nova 360.

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

1. ¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre el proceso de gestión de copias de seguridad en servidores de almacenamiento en la nube?
2. ¿Cuáles son los aspectos a tener en cuenta para realizar el diseño de un módulo para la gestión de copias de seguridad de Nova 360?
3. ¿Qué componentes son necesarios implementar en el desarrollo de un módulo que gestione las copias de seguridad en Nova 360?

4. ¿Qué pruebas de software aplicar para la evaluación del módulo de gestión de copias de seguridad en Nova 360?

En el desarrollo de la investigación se utilizaron los siguientes **métodos de la investigación científica:**

Métodos teóricos:

Método **inductivo-deductivo:** permitió establecer enunciados a partir del conocimiento que se tiene sobre el proceso de gestión de copias de seguridad, y de la observación del comportamiento de las diferentes aplicaciones informáticas para la gestión de copias de seguridad.

Método **analítico-sintético:** se empleó para consultar la bibliografía especializada en cuanto al desarrollo, características y particularidades de las aplicaciones informáticas para la gestión de copias de seguridad, e identificar elementos claves que contribuyan a la solución del problema de investigación; permitiendo sintetizar conceptos.

Modelación: posibilitó modelar la representación estructural, las relaciones internas y las características que forman parte del módulo para la gestión de copias de seguridad de Nova 360, haciendo uso de diagramas para el análisis, el diseño y la implementación del software.

Métodos empíricos:

Entrevista: se realizó para tener un mayor conocimiento sobre el proyecto Nova 360, sus características y las que deben tener las aplicaciones con las que se relaciona, así como las tecnologías necesarias para el desarrollo de dicho proyecto.

Observación: se utilizó para monitorear cómo se realiza la gestión de copias de seguridad en aplicaciones informáticas pertenecientes al modelo de computación en la nube de la categoría software como servicio.

Encuesta: con el objetivo de recopilar información en cuanto al nivel de satisfacción de los usuarios potenciales de la solución desarrollada.

La presente investigación está estructurada en: resumen, introducción, tres capítulos, conclusiones, referencias bibliográficas y anexos. A continuación, se presenta una breve descripción de los capítulos:

Capítulo 1. Fundamentación teórica sobre el proceso de gestión de copias de seguridad en servidores de almacenamiento en la nube: en este capítulo queda reflejada la fundamentación teórica de la investigación sobre el proceso de gestión de copias de seguridad a través del estudio de sistemas

homólogos; además se definen la metodología de desarrollo de software, y los lenguajes y las herramientas para la modelación e implementación del módulo para la gestión de copias de seguridad.

Capítulo 2. Análisis y diseño del módulo para la gestión de copias de seguridad en Nova 360: el capítulo presenta las diferentes tareas desarrolladas y los productos de trabajo obtenidos en el análisis y diseño del módulo para la gestión de copias de seguridad de Nova 360. Se muestra el modelo conceptual elaborado para comprender el contexto del negocio informatizado. Se especificaron los requisitos y se elaboró el diseño del módulo a partir de lo establecido por la metodología de desarrollo de software Variación de AUP para la UCI.

Capítulo 3. Implementación y evaluación del módulo para la gestión de copias de seguridad de Nova 360: en este capítulo se muestran las tareas y los resultados obtenidos de la implementación, y la evaluación del módulo para la gestión de copias de seguridad de Nova 360. Contiene los estándares de codificación empleados, así como el diagrama de despliegue elaborado. Se presentan las pruebas de software aplicadas y la evaluación del índice de satisfacción grupal con respecto al módulo para la gestión de copias de seguridad en Nova 360.

CAPÍTULO 1: Fundamentación teórica sobre el proceso de gestión de copias de seguridad en servidores de almacenamiento en la nube

El presente capítulo aborda aspectos sobre la fundamentación teórica de la investigación. Contiene el estudio del proceso de gestión de copias de seguridad en aplicaciones informáticas, reflejado en un análisis sobre soluciones existentes para la realización de copias de seguridad. Se define la metodología de desarrollo de software utilizada para la elaboración de la propuesta de solución, así como los lenguajes y herramientas empleados en el modelado y desarrollo del módulo para la gestión de copias de seguridad en Nova 360.

1.1 Definición de conceptos

Copias de seguridad: salva, copia de respaldo o también *backup*, es una copia de los datos originales que se realiza con el fin de disponer de un medio para recuperarlos en caso de su pérdida (PAUPIER, 2017).

Demonio: también conocido como *daemon* (*Disks And Extensions MONitor,* Monitor de Discos y Extensiones) es un programa que se ejecuta como un proceso en segundo plano en vez de ser controlado directamente por el usuario, es un proceso no interactivo, comúnmente esperando que ocurran eventos y ofreciendo servicios. El primer *daemon* fue un programa que automáticamente hacía copias de seguridad en cintas de datos del sistema de archivos (Academic, 2017).

Fichero: también llamado archivo, es un conjunto organizado de unidades de información (bits) almacenados en un dispositivo. Ejemplos de ficheros de texto: .doc, .txt o .odt; de imagen: .jpg, .png, o .gif; de video: .mpg, .avi, .vob o .mkv; y de audio: .mp3, .mp4, o m4a (Concepto.de.com, 2018).

Módulo: se refiere a las aplicaciones que están por defecto o que han sido desarrolladas para Nova 360 (The OwnCloud Developers, 2018). Estas aplicaciones se encuentran dentro del directorio raíz de Nova 360 en la carpeta *apps/*.

1.2 Descripción del proceso de gestión de copias de seguridad en aplicaciones informáticas

El proceso de realizar una copia de seguridad consiste en la acción de leer y grabar en la ubicación original u otra alternativa la información almacenada en un sistema informático (Redessil.com, 2018). La manera en la que se realiza este proceso está condicionada por la aplicación que se vaya a utilizar para hacer la copia de seguridad de la información; también se debe tener en cuenta a qué se le va a hacer la copia. Estas

salvas pueden ser manuales o automáticas, teniendo en cuenta la configuración establecida por el usuario que la realizará, además se debe tener en cuenta el lugar donde se guardarán estas copias, ya sea en el mismo servidor o en una entidad externa.

En la actualidad existen aplicaciones informáticas que realizan copias de seguridad de la información que se puede tener acumulada en la computadora. Estas son almacenadas en nubes tales como DropBox, OwnCloud, Google Drive, NextCloud, entre otras; en la propia computadora o en soportes digitales, por ejemplo, CD, DVD y Blu-Ray. También existen aplicaciones que realizan copias de seguridad de la información que se encuentra presente en las nubes como por ejemplo archivos de audio, video y texto, o las copias de seguridad realizadas anteriormente a la computadora (JOHNSON, 2017). Estas últimas aplicaciones siguen tres pasos fundamentales: hacen una copia de seguridad de la configuración de la nube, luego una de la información que tiene el usuario guardada en la nube, y por último de la base de datos.

El principal motivo por el que existen las copias de seguridad es la necesidad de mantener segura toda la información que se maneja, y en caso de un fallo del sistema poder recuperar dicha información de alguna forma. Como complemento al proceso de copias de seguridad surge el proceso de restauración de los datos. El mismo se rige por los mismos pasos que el proceso de copia: restaurar la configuración de la nube, restaurar la información que se tenía almacenada, y por ultimo restaurar la base de datos (Redessil.com, 2018).

En el siguiente epígrafe se describen algunas aplicaciones que realizan copias de seguridad de los archivos almacenados en la nube.

1.3 Aplicaciones informáticas para la gestión de copias de seguridad

En la actualidad existen diversas aplicaciones dedicadas a la gestión de copias de seguridad, entre ellas están Bacula, rsync, Simple Backup, fwbackups y Box Backup. A continuación, se presenta una breve caracterización de cada una de ellas.

1.3.1 Bacula

Bacula es una colección de herramientas de respaldo, capaces de cubrir las necesidades de respaldo de equipos bajo redes IP. Se basa en una arquitectura Cliente-Servidor que resulta eficaz y fácil de manejar, dada la gama de funciones y características que brinda como copiar y restaurar ficheros dañados o perdidos. Además, debido a su desarrollo y estructura modular, Bacula se adapta tanto al uso personal como

profesional (RODRÍGUEZ, 2016). Bacula contiene varios componentes que le hacen única: es software libre, *open source*, gratuito y multiplataforma.

1.3.2 rsync

rsync es una de las herramientas más usadas en Linux al hablar de copia de seguridad pues es una de las herramientas por defecto en la mayoría de sus distribuciones. Permite hacer copias incrementales tanto local como remotamente. rsync brinda la posibilidad de actualizar sistemas de archivos completos; preservando los enlaces, dueños, permisos y privilegios; emplea rsh, ssh, o sockets directos para establecer las conexiones; y soporta conexiones anónimas. Es una herramienta simple de línea de comandos para hacer copia de seguridad y una de las grandes ventajas que tiene es que se pueden crear scripts simples y así crear copias de seguridad automatizadas (Archlinux, 2018).

1.3.3 Simple Backup

Simple Backup está enfocado en realizar copias de seguridad en escritorios. Puede hacer copias de archivos y directorios, y permite emplear expresiones regulares para propósitos de exclusión. Simple Backup utiliza archivos comprimidos, lo que no es la mejor solución a grandes cantidades de datos precomprimidos como por ejemplo archivos multimedia; sin embargo, incluye soluciones predefinidas que pueden ser usadas para hacer copias de seguridad a directorios, como */var*, */etc/*, */usr/local/*. Simple Backup no está limitado a copias predefinidas, también se pueden hacer copias personalizadas, manuales y programadas (DragonJar, 2018). Cuenta con una interfaz amigable para el usuario, pero no contiene una herramienta para restaurar la información, lo que puede representar un problema en caso de fallo del sistema o pérdida de la información.

1.3.4 Box Backup

Box Backup es una herramienta para hacer copias de seguridad completamente automatizada, siendo capaz de cifrarla y asegurarla. Utiliza dos demonios, uno para el cliente y otro para el servidor, así como una utilidad para restaurar. Las conexiones que establece son seguras debido a que emplea certificados SSL (*Secure Sockets Layer*) para autenticar los clientes. Box Backup es una solución de línea de comandos, multiplataforma, simple de configurar y usar; y cuenta con tres componentes importantes: *bbstored* (demonio del servidor), *bbackupd* (demonio del cliente), y *bbackupquery* (herramienta para consultar y restaurar las copias de seguridad) (DragonJar, 2018).

1.3.5 fwbackups

La herramienta fwbackups es una de las soluciones de Linux para hacer copias de seguridad. Es multiplataforma, tiene una interfaz amigable para el usuario, y puede hacer copias individuales o programadas. Permite realizar copias de seguridad tanto local como remotamente en formato .tar, .tar.gz, .tar.bz2 o con rsync. A través de esta herramienta se pueden obtener copias tanto de un solo archivo como de una computadora completa. A diferencia de muchas otras herramientas, fwbackups es fácil de instalar debido a que se encuentra en los repositorios de muchas distribuciones; además brinda la posibilidad de efectuar copias de seguridad incrementales o diferenciales para acelerar el proceso (DragonJar, 2018).

1.4 Análisis comparativo de aplicaciones informáticas para la gestión de copias de seguridad

En el siguiente epígrafe se efectuará un análisis comparativo de las aplicaciones informáticas para la gestión de copias de seguridad mediante el método QSOS (*Qualification and Selection of Open Source Software*, Calificación y Selección de Software de Código Abierto). Este modelo permite identificar si el software cumple con los requisitos técnicos, estratégicos y funcionales, mediante la comparación y clasificación de los diferentes productos según los criterios especificados (DÍAZ, et al., 2010).

1.4.1 Aplicación del método QSOS

El método cuenta con 4 fases para la evaluación y selección de software de código abierto:

Fase de Definición: durante esta fase se construye el marco referencial, para ello se clasifica la familia de software, los tipos de licencia, y los tipos de comunidad a la que pertenece. Se definió que se estudiarían aplicaciones de gestión de copias de seguridad que son de software libre, *open source* y multiplataforma.

Fase de Evolución: en esta fase se caracterizan los softwares analizados. En el epígrafe 1.3 está reflejada esta caracterización.

Fase de Calificación: esta fase consiste en la ponderación de los criterios definidos para la comparación de las aplicaciones de gestión de copia. En la siguiente tabla se establecen los criterios a tener en cuenta según las necesidades del cliente y las características del proyecto Nova 360. Para conocer las necesidades del cliente, así como las características del proyecto se realizó una entrevista a miembros del equipo de desarrollo del proyecto Nova 360, (ver Anexo 1).

Tabla 1. Definición de criterios y su ponderación

Fuente: elaboración propia

Puntuación / **Criterios para análisis**	**Cubierto** **1**	**No cubierto** **0**
Basada en software libre	Está basada en software libre	No está basada en software libre
Recomendada por la documentación oficial de OwnCloud	Está recomendada por la documentación oficial	No está recomendada por la documentación oficial
Integración con Nova 360	Permite la integración con Nova 360	No se integra con Nova 360
Interfaz visual	Cuenta con una interfaz visual	No cuenta con una interfaz visual
Copia de seguridad de ficheros	Permite las copias de seguridad de ficheros	No realiza copia de seguridad de los ficheros
Copias de seguridad de base de datos	Permite las copias de seguridad de base de datos	No realiza copia de seguridad de la base de datos
Restauración de ficheros	Permite la restauración de los ficheros	No restaura los ficheros
Restauración de base de datos	Permite la restauración de la base de datos	No restaura la base de datos

Fase de Selección: luego de creado el filtro en la tercera etapa se procede a aplicarlo con los datos proporcionados por las 2 primeras etapas, con el fin de formular consultas, comparaciones y la selección del producto. En la siguiente tabla se evidencia la comparación entre las aplicaciones informáticas para la gestión de copias de seguridad.

Tabla 2. Tabla comparativa de aplicaciones informáticas para la gestión de copias de seguridad

Fuente: elaboración propia

Criterios de análisis	**Sistemas informáticos para la gestión de copias de seguridad**				
	fwbackups	Bacula	Rsync	Box Backup	Simple Backup
Basada en software libre	1	1	1	1	1
Recomendada por la documentación oficial de OwnCloud	0	0	1	0	0
Integración con Nova 360	0	0	0	0	0
Interfaz visual	1	1	0	0	1
Copia de seguridad de ficheros	1	1	1	1	1
Copias de seguridad de base de datos	1	1	1	1	1
Restauración de ficheros	1	1	1	1	0
Restauración de base de datos	1	1	1	1	0

La comparación anterior evidencia que las aplicaciones estudiadas no cubren en su totalidad las necesidades existentes en Nova 360 para realizar la gestión de copias de seguridad; quedando reflejados estos requisitos en los criterios analizados y demostrando la importancia de desarrollar un módulo que aproveche las mejores funcionalidades de estas aplicaciones. El estudio realizado permitió definir requisitos funcionales y características que deben tener las aplicaciones de gestión de copias de seguridad, así como las limitaciones que poseen estas aplicaciones, tomadas en cuenta para el diseño de la propuesta de solución y así evitar errores en la gestión de copias de seguridad, fortaleciendo las funcionalidades del módulo para la gestión de salvas.

1.5 Nova 360

Nova 360 es una solución informática basada en OwnCloud, por lo que pertenece al modelo de computación en la nube. Permite la sincronización con múltiples dispositivos de datos, calendarios, tareas y otros, así como el intercambio de información entre Nova 360 y dispositivos con la distribución cubana GNU/Linux Nova o la personalización cubana de Android para móviles NovaDroid. En su estructura cuenta con 11 carpetas y 14 ficheros. Algunas de estas carpetas son: *config/* donde se encuentra la configuración del sistema; *data/* donde se almacenan las cuentas de los usuarios y la información que cada uno tiene guardada; y *apps/*, donde se encuentran todos los módulos de Nova 360. También cuenta con una serie de medidas de seguridad en el servidor tales como la encriptación del almacenamiento, monitoreo de red, y denegación y control de acceso a los servicios a través de la herramienta fail2ban. Nova 360 tiene como objetivo principal lograr la integración de NovaNas y Nova Unificado. El módulo para la gestión de copias de seguridad debe permitir planificar, realizar, eliminar y restaurar las copias de seguridad. Estas salvas se deben poder realizar de forma manual o automática y se debe poder almacenar en una ubicación local o en un servidor remoto definido por el usuario.

1.6 Metodología de desarrollo de software

Una metodología de desarrollo de software es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos para implementar sistemas informáticos. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de software a elegir las técnicas más apropiadas en cada momento del proyecto; así como a planificarlo, gestionarlo, controlarlo y evaluarlo (FERNÁNDEZ, 2013). En el epígrafe 1.6.1 se define la metodología de desarrollo de software empleada en la elaboración de la propuesta de solución.

1.6.1 Variación de AUP para la UCI

La metodología de desarrollo de software Variación AUP para la UCI es una variación de AUP (*Agile Unified Process*, Proceso Unificado Ágil) que logra 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 (RODRÍGUEZ, 2015).

Esta metodología cuenta con 3 fases:

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 (RODRÍGUEZ, 2015). De los 4 escenarios que propone la metodología para la descripción de requisitos se emplea el cuarto ya que es factible aplicarlo a proyectos bien definidos; también se ha de tener en cuenta que 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 de la solución

En el modelado de la propuesta de solución se utilizó el Lenguaje Unificado de Modelado (UML) y la herramienta Visual Paradigm v5.0 descritos a continuación:

Visual Paradigm v5.0

Visual Paradigm es una herramienta CASE (*Computing Aided Software Engineering*, Ingeniería de Software Asistida por Computación) es un software de modelado UML que permite analizar, diseñar, codificar, probar y desplegar. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Visual Paradigm Internationals, 2018).

Lenguaje Unificado de Modelado (UML) v8.0

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) (Lucid Software Inc., 2018).

1.8 Tecnologías para la implementación

Para efectuar la implementación de la propuesta de solución se utilizó PHP v7.0.8, JavaScript, HTML5, CSS3, JetBrains PhpStorm 2018.2.5, Apache v2.4, MySQL, Git. A continuación, se presenta una breve descripción:

1.8.1 Lenguajes

Un lenguaje de programación es un lenguaje diseñado para realizar procesos que pueden ser ejecutados por máquinas, como por ejemplo computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, y para expresar algoritmos con precisión (Definiciones.de, 2018). A continuación, se describen los lenguajes PHP v7.0.8, JavaScript, HTML5 y CSS3 utilizados para desarrollar la propuesta de solución.

PHP v7.0.8

PHP (*Hypertext Preprocessor*) es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. El código generado es interpretado por un servidor web con un módulo de procesador de PHP que genera el código HTML resultante. Para el presente trabajo se utilizó en su versión 7.0.8, la cual cuenta con mejoras de rendimiento y declaración de tipos de retorno en sus funciones (GONZÁLES, 2015).

Entre las ventajas de PHP están:

- ✓ Es un lenguaje multiplataforma.
- ✓ Es de código abierto.
- ✓ Tiene manejo de excepciones.

- ✓ Es potente, fácil de aprender.
- ✓ Permite el acceso a bases de datos y otras funcionalidades orientadas a la red.
- ✓ Dispone de abundante soporte en la Web.

JavaScript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web y aplicaciones de servidor, aunque también es usado en muchos entornos sin navegador. Es multi-paradigma, basado en prototipos, dinámico, y soporta estilos de programación funcional y orientada a objetos. Para definir cada una de sus versiones utiliza el estándar ECMAScript. Actualmente JS se encuentra en la versión correspondiente a la edición ECMA2019 de dicho estándar (Mozilla, 2018).

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 (PÉREZ, y otros, 2018).

CSS3

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 (W3C, 2018).

1.8.2 Entorno de desarrollo

Un entorno de desarrollo integrado (IDE*, Integrated Development Environment*) es una aplicación de software, que proporciona servicios integrales para facilitarle a los programadores el desarrollo de software. Consiste en un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de estos programas cuentan con auto-completado inteligente de código, lo que ayuda mucho en el desarrollo de aplicaciones (Aritmetrics, 2018). A continuación, se presenta Jetbrains PhpStorm 2018.2.5, IDE utilizado para la implementación de la propuesta de solución.

JetBrains PhpStorm 2018.2.5

JetBrains PhpStorm es un IDE multiplataforma para PHP basado en la plataforma IntelliJ IDEA de JetBrains. Proporciona un editor para PHP, HTML y JavaScript con análisis de código sobre la marcha, prevención de errores y refactorizaciones automatizadas para códigos PHP y JavaScript. Este editor es compatible con todas las funciones del lenguaje PHP (5.3, 5.4, 5.5, 5.6, 7.0, 7.1, 7.2) para proyectos modernos y heredados e incluye un editor de SQL (*Structured Query Language*, Lenguaje de consulta estructurada) completo con resultados de consultas editables. Con la versión 2018.2.5 del PhpStorm se corrigen varios errores en cuanto a las consultas SQL y se cuenta con la ventaja de añadir plantillas personalizadas (JetBrains, 2018).

1.8.3 Servidor de aplicaciones

Un servidor de aplicaciones es un programa de servidor en un equipo en una red distribuida que proporciona la lógica de negocio para un programa de aplicación. Gestiona la mayor parte o la totalidad de las funciones de lógica de negociación y de acceso a los datos de las aplicaciones. Los principales beneficios del empleo de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones (ROUSE, 2016). El ejemplo más común del uso de servidores de aplicación son los portales web que permiten el acceso, gestión y divulgación de la información de las empresas, brindar servicios tales como los servicios web, de manera segura y transparente, desde cualquier dispositivo. Durante la investigación se utilizó Apache v2.4 como servidor de aplicaciones.

Apache v2.4

Apache es un servidor web HTTP (*Hipertext Transfer Protocol*) gratuito, de código abierto, flexible, fácil de mantener y altamente configurable debido a su diseño modular. Es multiplataforma y presenta abundante documentación, adicionalmente tiene dentro de sus principales ventajas el funcionamiento con varios lenguajes, incluyendo PHP. Apache tiene amplia aceptación en la red desde 1996 convirtiéndose en el servidor HTTP más usado (Apache, 2018).

1.8.4 Sistema de gestión de base de datos

Se denomina Sistema de Gestión de Bases de Datos (SGBD) al software o conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista; además de proporcionar herramientas para añadir, borrar, modificar y analizar (MATO , 2014). Los usuarios pueden acceder a la información usando herramientas específicas de consulta y de generación de informes. Algunos ejemplos de SGBD son PostgreSQL, SQLite, Cassandra, MongoDB, MySQL, entre otros. Para la presente investigación se utilizó MySQL v5.7.11 como sistema de gestión de base de datos y MySQL Workbench para la administración, diseño y mantenimiento de dicho sistema de base de datos. A continuación, se presenta una breve caracterización.

MySQL v5.7.11

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual, (Licencia pública general y Licencia comercial), por Oracle Corporation. Es considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web. Gracias a su rendimiento, confiabilidad y facilidad de uso, se ha convertido en la principal opción de base de datos para aplicaciones web (Oracle Corporation, 2018).

MySQL Workbench

MySQL Workbench es una herramienta visual unificada para arquitectos de bases de datos y desarrolladores que integra desarrollo de software, administración y diseño de bases de datos, y modelado de datos, desarrollo de SQL y herramientas de administración integrales para la configuración del servidor, la administración de usuarios, las copias de respaldo, entre otras. Además, proporciona un conjunto de

herramientas para mejorar el rendimiento de las aplicaciones MySQL, así como su gestión y mantenimiento (Oracle Corporation, 2018).

1.8.5 Herramienta de control de versiones

Un sistema de control de versiones registra todos los cambios hechos en uno o más proyectos, guardando 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 (ANDREARR, 2014). Algunos de los sistemas de control de versiones más famosos son Subversion (también conocido como Svn), Mercurial y Git, este último es el que utiliza a lo largo de la investigación para el control de versiones. A continuación, se presenta una breve caracterización.

Git

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. Proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida. La rapidez en la gestión distribuida, la gestión de las ramas, la gestión eficiente de proyectos grandes y el realmacenamiento periódico de paquetes son algunas de sus características más importantes (CHACON, y otros, 2014).

Conclusiones del capítulo

El análisis del marco teórico sobre el proceso de gestión de copias de seguridad para Nova 360 y el estudio de los principales conceptos asociados al problema planteado, permitieron establecer las bases para el desarrollo de la investigación y conocer las características del objeto de estudio. A partir de la caracterización y comparación de varias aplicaciones para la gestión de copias de seguridad se demostró la necesidad de desarrollar un módulo que aproveche las mejores funcionalidades que estas aplicaciones ofrecen, pues estas no resuelven la totalidad de las insuficiencias presentes en Nova 360. En el desarrollo de la propuesta de solución se definió el empleo de herramientas informáticas en su mayoría libres para garantizar la soberanía tecnológica, y la utilización de la metodología de desarrollo de software Variación de AUP para la UCI.

CAPÍTULO 2: Análisis y diseño del módulo para la gestión de copias de seguridad en Nova 360

Este capítulo aborda los aspectos técnicos de la investigación referentes al análisis y diseño del módulo para la gestión de copias de seguridad de Nova 360. Contiene un modelo conceptual realizado para comprender el contexto del negocio informatizado. Se definen los requisitos a través de la especificación de requisitos de software, su descripción mediante historias de usuario, además de realizar la validación de los mismos. Se realiza el diagrama de clases del diseño, en el que se representan las restricciones de implementación de las funcionalidades del módulo a partir de los patrones del diseño. Se presenta el modelo de datos que describe la estructura de persistencia de la información y el diseño arquitectónico definido para la propuesta de solución.

2.1 Descripción del contexto de la propuesta de solución desarrollada

La descripción del contexto del negocio presentada en el epígrafe se realizó utilizando un modelo conceptual. El mismo le permitió a la autora de la investigación comprender el funcionamiento del dominio del negocio informatizado.

2.1.1 Modelo conceptual

Un modelo conceptual contiene la descripción de cómo se relacionan los conceptos que intervienen en el negocio y sirve para representar el problema de manera gráfica. Los conceptos que en este modelo aparecen, son una representación de elementos del mundo real y no de componentes de software (DALVIK, 2018). A continuación, se presenta el modelo conceptual que corresponde a la descripción del contexto del negocio de la propuesta de solución:

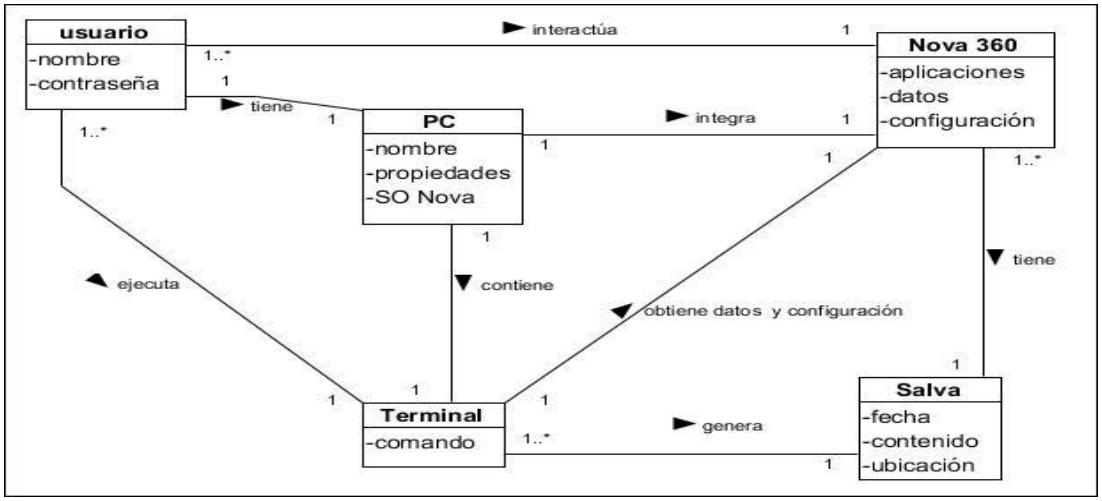


Figura 1. Modelo conceptual

Fuente: elaboración propia

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

Conceptos del contexto del negocio:

Nova 360: es un software que permite la sincronización de archivos, calendarios, tareas y notas. En su composición cuenta con las configuraciones, los datos que los usuarios han almacenado y con una base de datos.

PC: es un ordenador que contiene la distribución cubana de GNU/Linux que cuenta con una terminal, y se integra con Nova 360; con este ordenador se pueden realizar copias de seguridad de la información presente en su sistema.

Salva: es una copia de seguridad del sistema (datos, configuración y base de datos).

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

Usuario: es el cliente final del sistema y accede al sistema operativo para poder instalar los paquetes.

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

Diccionario de datos

Un diccionario de datos contiene las características lógicas de los datos que se van a utilizar en el sistema que se va a programar, incluyendo: nombre, descripción, alias, contenido y organización. Es un conjunto de información acerca de los conceptos presentes en el contexto de un negocio determinado (ÁLVARES, 2017). En la tabla 3 se presenta el diccionario de datos del concepto “Salva”.

Tabla 3. Diccionario de datos del concepto Salva

Fuente: elaboración propia

Descripción	Salva: Es una copia de seguridad del sistema (datos, configuración y base de datos).					
Atributos						
					Restricciones de clase	
Nombre	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Único?**	**Válidas**	**No válidas**
Fecha	Identificador de la salva	Numérico	No	Sí	No aplica	Caracteres numéricos
Contenido	Contiene el contenido de la salva	Alfanumérico	No	No	No aplica	No aplica
Ubicación	Lugar donde la salva es almacenada	Alfanumérico	No	No	Caracteres alfanuméricos	No aplica

Para conocer y mejorar el entendimiento de las necesidades del cliente, y la gestión de las funcionalidades y restricciones de diseño que debe cumplir la propuesta de solución se desarrolló la disciplina Requisitos.

2.2 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 (RODRÍGUEZ, 2015). A continuación, se presenta las actividades desarrolladas en esta disciplina, así como los productos de trabajo elaborados en el desarrollo de la investigación.

2.2.1 Fuentes para la obtención 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 (PÉREZ, 2017). En este proceso intervienen diferentes fuentes que permiten identificar los requisitos que forman parte de una aplicación informática. Durante esta etapa de la investigación se tuvieron en cuenta como fuentes de obtención de requisitos:

- ✓ Modelo conceptual
- ✓ Aplicaciones informáticas para la gestión de copias de seguridad (epígrafe 1.3 y 1.4).
- ✓ Especialistas del centro CESOL

Una adecuada comprensión de los requisitos favorece el desarrollo de los sistemas informáticos y que estos cumplan con las necesidades y expectativas del cliente. Para realizar este procedimiento existen diversas técnicas que guían al equipo de proyecto en el proceso de comunicación con el cliente (PRESSMAN, 2010). En el epígrafe 2.2.2 se definen técnicas utilizadas en el levantamiento de requisitos del módulo para la gestión de copias de seguridad en Nova 360.

2.2.2 Técnicas de identificación de requisitos

En la obtención de los requisitos del módulo para la gestión de las copias de seguridad en Nova 360 se emplearon como técnicas de identificación de requisitos la entrevista, la tormenta de ideas y el desarrollo de prototipos. Su definición y forma de aplicación se describen a continuación.

Entrevista

Es una técnica muy utilizada para recoger información de otra persona a través de una comunicación interpersonal que se realiza por medio de una conversación estructurada. Es importante la forma en que se plantea la conversación y la relación que se establece en la entrevista, ya que deben quedar bien definidos cuales son los contenidos que se desean obtener (GUERRA, 2018). La técnica se aplicó a través de una entrevista realizada a los miembros del proyecto Nova 360 (ver Anexo 1) para conocer cuáles son las tecnologías compatibles con el entorno de desarrollo, los sistemas con que se integra, así como los requerimientos y las características que debe tener una aplicación para ser instalada en dicho software.

Tormenta de ideas

Consiste en reuniones entre un grupo de personas donde como primer paso sugieren toda clase de ideas sin juzgar su validez, y después de recopilar todas las ideas se realiza un análisis detallado de cada propuesta (GUERRA, 2018). Se realizó una tormenta de ideas con los miembros del equipo de desarrollo de Nova 360, con el objetivo de concretar cuáles son las funcionalidades que no deben faltar en la propuesta de solución, ya que se habían identificado un primer conjunto de requisitos en donde no se habían tenido en cuenta todas las necesidades a cubrir. En ella se tuvieron en cuenta todos los planteamientos realizados.

Desarrollo de prototipos

Los prototipos suelen consistir en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación pedida. Fomentan el desarrollo de ideas que desembocan en requerimientos, además de permitir a los usuarios mejorar las especificaciones de requerimientos. En algunos casos también se utiliza como un medio para formalizar la aceptación previa del cliente de los requisitos del proyecto (GUERRA, 2018). Se aplicó la técnica de desarrollo de prototipos para conocer cuáles son las características que debían tener las interfaces del módulo desarrollado. Los prototipos elaborados fueron revisados por miembros del equipo de desarrollo, quienes consideraron que se debía contar con estos prototipos para la representación visual de la propuesta de solución.

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

2.2.3 Especificación de requisitos de software

La Especificación de Requisitos de Software (ERS) es una descripción completa del comportamiento y características del sistema que se va a desarrollar. Sus principales objetivos son ayudar a los clientes a describir claramente lo que se desea obtener de un determinado software y ayudar a los desarrolladores a entender qué quiere exactamente el cliente. La ERS incluyen un conjunto de requisitos funcionales y no funcionales que sirven para describir las necesidades de los clientes y de los usuarios, así como los requerimientos que debe tener el sistema para tener un mejor funcionamiento (MONFERRER, 2015).

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (RODRÍGUEZ, 2015). A continuación, se presenta el listado de los requisitos funcionales de la propuesta de solución:

Tabla 4. Listado de requisitos funcionales de la propuesta de solución

Fuente: elaboración propia

Número	Requisitos	Descripción	Complejidad
RF 1	Adicionar planificación de copia de seguridad.	El módulo permite al usuario crear la planificación de una copia de seguridad. En caso de que el usuario ya no desee crear la planificación cuenta con la opción de cancelar dicha planificación.	Alta
RF 2	Mostrar planificación de copia de seguridad	El módulo le debe mostrar al usuario un listado con todas las planificaciones creadas, así como mostrar los parámetros definidos por el usuario al marcar una de la lista.	Media

RF 3	Actualizar planificación de copia de seguridad	El módulo permite actualizar la planificación de una copia de seguridad seleccionada, o en caso que ya no desee actualizar la planificación selecciona la opción de cancelar.	Alta
RF 4	Eliminar planificación de copia de seguridad.	El módulo permite eliminar la planificación de la copia de seguridad que se encuentre seleccionada en ese momento.	Media
RF 5	Crear copia de seguridad.	El módulo permite crear una copia de seguridad a partir de la planificación previamente realizada. Aunque también se cuenta con la opción “Salva ahora”, la cual permite realizar una copia de seguridad instantáneamente, sin tener en cuenta las planificaciones anteriormente establecidas.	Alta
RF 6	Mostrar listado de copias de seguridad.	El sistema debe mostrar un listado con todas las copias de seguridad realizadas, su ubicación, nombre, tamaño y tipo (local o remota).	Media
RF 7	Eliminar copia de seguridad.	El sistema debe permitir eliminar una copia de seguridad realizada luego de haberla seleccionado en el listado de copias de seguridad.	Media
RF 8	Restaurar copia de seguridad.	El sistema debe permitir restaurar una copia de seguridad realizada luego de	Alta

		haberla seleccionado en el listado de copias de seguridad.	
RF 9	Adicionar ubicación	El módulo permite al usuario adicionar una ubicación para almacenar las copias de seguridad creadas. En caso de que el usuario ya no desee adicionar una ubicación puede cancelar la acción.	Media
RF 10	Mostar listado de ubicaciones.	El módulo le debe mostrar al usuario un listado con todas las ubicaciones creadas. Al marcar una de ellas se muestran los parámetros que el usuario había definido durante su creación.	Media
RF 11	Eliminar ubicaciones.	El módulo permite eliminar ubicación que se encuentre seleccionada en ese momento.	Alta
RF 12	Actualizar ubicación	El módulo permite actualizar la ubicación de una copia de seguridad seleccionada, o en caso que ya no desee actualizar la planificación puede cancelar la acción.	Alta

Requisitos no funcionales

Los requisitos no funcionales son cualidades y restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo sobre el proceso de desarrollo. Estas restricciones a menudo se aplican al sistema en su totalidad, normalmente apenas se aplican a las características o servicios individuales del sistema (SOMMERVILLE, 2011).

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 el listado de los requisitos no funcionales del módulo para la gestión de copias de seguridad en Nova 360.

Tabla 5. Listado de requisitos no funcionales de la propuesta de solución

Fuente: elaboración propia

Número	Requisito	Descripción
RnF 1	Requisito de funcionalidad	El módulo para la gestión de copias de seguridad en Nova 360 debe realizar todas las operaciones indicadas por el usuario en cada momento: adicionar, mostrar, actualizar y eliminar planificación de copia de seguridad; adicionar, mostrar, actualizar y eliminar ubicación de copia de seguridad; crear, mostrar, restaurar y eliminar copias de seguridad.
RnF 2	Requisito de interoperabilidad	El módulo para la gestión de copias de seguridad debe ser compatible con Nova 360.
RnF 3	Requisito de seguridad	Al módulo para la gestión de copias de seguridad en Nova 360 solo podrán acceder usuarios administradores.
RnF 4	Requisito de confiabilidad	El módulo para la gestión de copias de seguridad en Nova 360 debe ser capaz de recuperarse después de haberse producido un fallo de software o luego de la restauración de una copia de seguridad.
RnF 5	Requisito de usabilidad	El módulo para la gestión de copias de seguridad en Nova 360 debe mostrar sus interfaces en idioma español; mostrando en sus componentes una secuencia lógica de pasos a seguir para el usuario.
RnF 6	Requisito de mantenibilidad	El módulo para la gestión de copias de seguridad en Nova 360 debe estar codificado en el estándar definido por el proyecto de desarrollo

		de Nova 360, con el fin de facilitar el análisis para futuras modificaciones, o simplemente para dar mantenimiento al sistema.

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

2.2.4 Descripción de requisitos de software mediante Historias de Usuario

En la descripción de los requisitos se elaboraron 12 Historias de Usuarios. En la tabla 6 se muestra la descripción de la historia de usuario correspondiente al RF 1: Adicionar planificación de copia de seguridad.

Tabla 6. Descripción de la historia de usuario del RF.1 Adicionar planificación de copia de seguridad

Fuente: elaboración propia

HISTORIA DE USUARIOS	
Número: RF1	**Nombre del requisito:** Adicionar planificación de copia de seguridad.
Programador: Miosotis Toledo Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 25 horas
Riesgo en Desarrollo: No aplica	**Tiempo Real:** 25 horas
Descripción: El módulo permite al usuario crear la planificación de una copia de seguridad. El usuario selecciona la opción de “Adicionar planificación” del menú superior, luego se activan los criterios de selección de tiempo (minuto, hora, día de la semana, día del mes, mes), un campo para poner el nombre de la planificación y seleccionar la ubicación donde se realizará la copia de seguridad, la cual puede ser remota o local. En caso de que no desee seleccionar ninguna de las ubicaciones anteriores contará con un botón que le envía a la vista en la que crea una ubicación nueva. Una vez llenados los campos se selecciona la opción de “Crear” en el menú inferior. En caso de que el usuario se arrepienta en la creación de una planificación se cuenta con la opción “Cancelar” también en el menú inferior.	

Observaciones:

Prototipo elemental de interfaz gráfica de usuario:

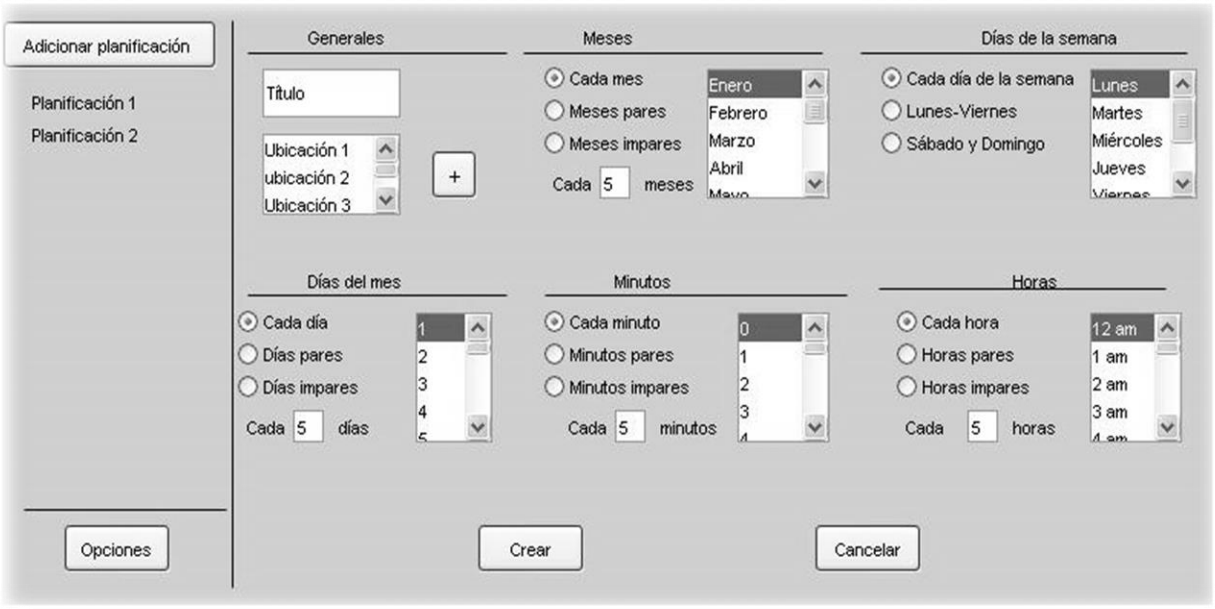


Figura 2. Prototipo de interfaz de usuario correspondiente al RF1. Adicionar planificación de copias de seguridad

Fuente: elaboración propia

2.2.5 Validación de requisitos de software

El objetivo de la validación de requisitos es asegurar que todos los requisitos del software que se han establecido se correspondan con las necesidades del negocio, de los clientes y de los usuarios, que se han detectado las inconsistencias, omisiones y errores y que estos han sido corregidos (PÉREZ, 2016). Para validar los requisitos obtenidos se aplicaron las técnicas de prototipado de interfaz de usuario y diseños de caso de prueba, los cuales se describen a continuación.

Prototipado 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 los clientes, usuarios finales y equipo de desarrollo explorar un diseño de interfaz de usuario alcanzable y adecuado que cumpla los requisitos, ayudando a reducir las distancias entre lo que es necesario y lo que es factible. El objetivo principal de la

creación de un prototipo de interfaz de usuario es poder probar el diseño de estas interfaces, incluyendo la capacidad de utilización antes de que empiece el desarrollo real. De este modo, se puede garantizar que se está construyendo el sistema correcto, antes de dedicar demasiado tiempo y recursos al desarrollo de la solución informática (IBM Corp., 2018).

Los prototipos de interfaz de usuario realizados de acuerdo a los requisitos obtenidos fueron analizados con el analista y jefe de proyecto de Nova 360. Esto le permitió a la autora de la investigación conocer las deficiencias existentes y que se pudieran hacer las correcciones necesarias antes del diseño e implementación de las funcionalidades del sistema. Posteriormente fueron presentados al cliente para que tuviera una noción de la propuesta de solución. El mismo emitió su valoración, las cuales fueron tomadas en cuenta en el desarrollo de la aplicación.

Diseño de caso de prueba (DCP): son una serie de acciones que se realizan para determinar una función o funcionalidad particular de su aplicación, su objetivo principal es crear casos de prueba para probar los posibles defectos que pudiera tener el sistema y demostrar que se satisfacen los requisitos obtenidos (HOOGENRAAD, 2018).

La generación de casos de prueba permitió especificar los posibles datos de entrada, así como las salidas de los requisitos funcionales del módulo para la gestión de copias de seguridad en Nova 360. En total se generaron 12 diseños de casos de pruebas, uno para cada requisito funcional, los cuales fueron revisados por la analista del proyecto y garantizó la corrección de las deficiencias encontradas y mejorar la calidad de las pruebas de software. A continuación, se presenta el DCP que corresponde al RF 1. Adicionar planificación de copia de seguridad.

Caso de prueba correspondiente al RF 1. Adicionar planificación de copia de seguridad

Descripción general: El módulo permite al usuario adicionar la planificación de una copia de seguridad.

Precondiciones: No aplica

Escenario	Descripción	Días del mes	Meses	Dias de la semana	Minutos	Horas	Respuesta del sistema	Flujo central
EC 1.1 Adicionar una planificación de una copia de seguridad	El usuario selecciona la opción de "Adicionar planificación" del menú superior, luego se activan los criterios de selección de tiempo (minuto, hora, día de la semana, día del mes, mes) para ser llenados. Una vez llenados los campos se selecciona la opción de "Crear" en el menú inferior.	5	5	Cada dia de la semana	5	5	Se añade la planificación a la lista de planificaciones creadas.	1. Seleccionar botón "Adicionar planificación". 2.Seleccionar botón "Crear".
EC 1.2 Cancelar la planificación	En caso de que el usuario se arrepienta de la creación de una nueva planificación se cuenta con la opción "Cancelar" también en el menú inferior.	NA	NA	NA	NA	NA	Se desactivan los criterios de selección	1. Seleccionar botón "Adicionar planificación". 2. Seleccionar botón "Cancelar"

Figura 3. Diseño de caso de prueba correspondiente al RF1. Adicionar planificación de copia de seguridad

Fuente: elaboración propia

Descripción de las variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Días del mes	Campo de texto, radio button, campo de selección	No	El campo de texto solo admite valores numéricos, en el campo de selección el usuario selecciona uno o varios días del mes.
2	Meses	Campo de texto, radio button, campo de selección	No	El campo de texto solo admite valores numéricos, en el campo de selección el usuario selecciona uno o varios meses.
3	Días de la semana	Radio button, campo de selección	No	El campo de texto solo admite valores numéricos
4	Minutos	Campo de texto, radio button, campo de selección	No	El campo de texto solo admite valores numéricos, en el campo de selección el usuario selecciona uno o varios minutos específicos.
5	Horas	Campo de texto, radio button, campo de selección	No	El campo de texto solo admite valores numéricos, en el campo de selección el usuario selecciona una o varioas horas especificas.

Figura 4. Descripción de las variables

Fuente: elaboración propia

La identificación de las necesidades del cliente, las restricciones presentes en el contexto del negocio informatizado y su evaluación por diferentes miembros del equipo de proyecto de Nova 360 permitieron la realización del análisis y diseño de la propuesta de solución, mostrado en el epígrafe 2.3.

2.3 Análisis y diseño

En esta disciplina los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Se modela el sistema, su forma y su arquitectura para que soporte todos los requisitos, incluyendo los requisitos no funcionales (RODRÍGUEZ, 2015). En los epígrafes siguientes se muestran las actividades desarrolladas en esta disciplina, así como los productos de trabajo elaborados en el proceso de desarrollo del módulo para la gestión de copias de seguridad en Nova 360.

2.3.1 Diseño de clases

El diseño de clases del módulo para la gestión de copias de seguridad en Nova360, se realizó mediante estereotipos web. En su desarrollo se tuvo en cuenta los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns*, Patrones de Asignación de Responsabilidad) e Inyección de dependencias. A continuación, se detallan los resultados obtenidos.

Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces: en él se muestran un conjunto de clases, interfaces, métodos, dependencias y relaciones. (HERNÁNDEZ, 2018). En la figura 5 se presenta el diagrama de clases del diseño que corresponde al RF1. Adicionar planificación de copia de seguridad.

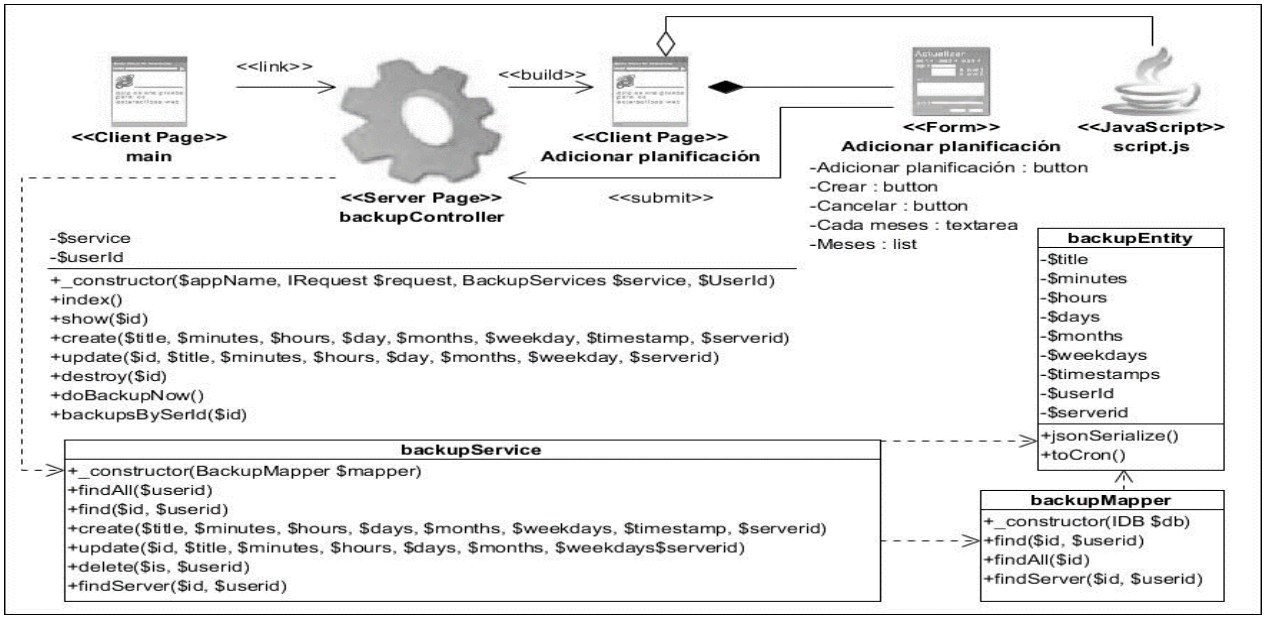


Figura 5. Diagrama de clases del diseño correspondiente al RF1. Adicionar planificación de copia de seguridad

Fuente: elaboración propia

El diagrama de clases anterior está formado por las clases:

- ✓ Clases interfaces: definen todas las abstracciones necesarias para la interacción entre los usuarios y la aplicación. Comprende las vistas *main.php* y Adicionar planificación*.*
- ✓ Clase controladora: permite encapsular las funcionalidades necesarias para interactuar con las clases interfaces, las clases entidades y las clases de acceso a datos. Este caso está presente en la clase *BackupController()*.
- ✓ Clases de acceso a datos: permite la comunicación entre la clase controladora y la base de datos. Se encuentra representada por la clase *BackupMapper()*.
- ✓ Clases persistentes: representan el almacenamiento de los datos que serán persistentes en el tiempo, aun cuando haya finalizado la ejecución de la aplicación. Está formada por la clase *BackupEntity()*.

Patrones de diseño de software

Un patrón de diseño provee un esquema para refinar componentes de un sistema de software y la forma en que se relacionan entre sí. Describe una estructura generalmente recurrente de comunicación de componentes que resuelve un problema de diseño general dentro de un contexto particular (ALMEIRA, y otros, 2007). En la figura 6 se representa el uso de los patrones de diseño GRASP e Inyección de dependencias:

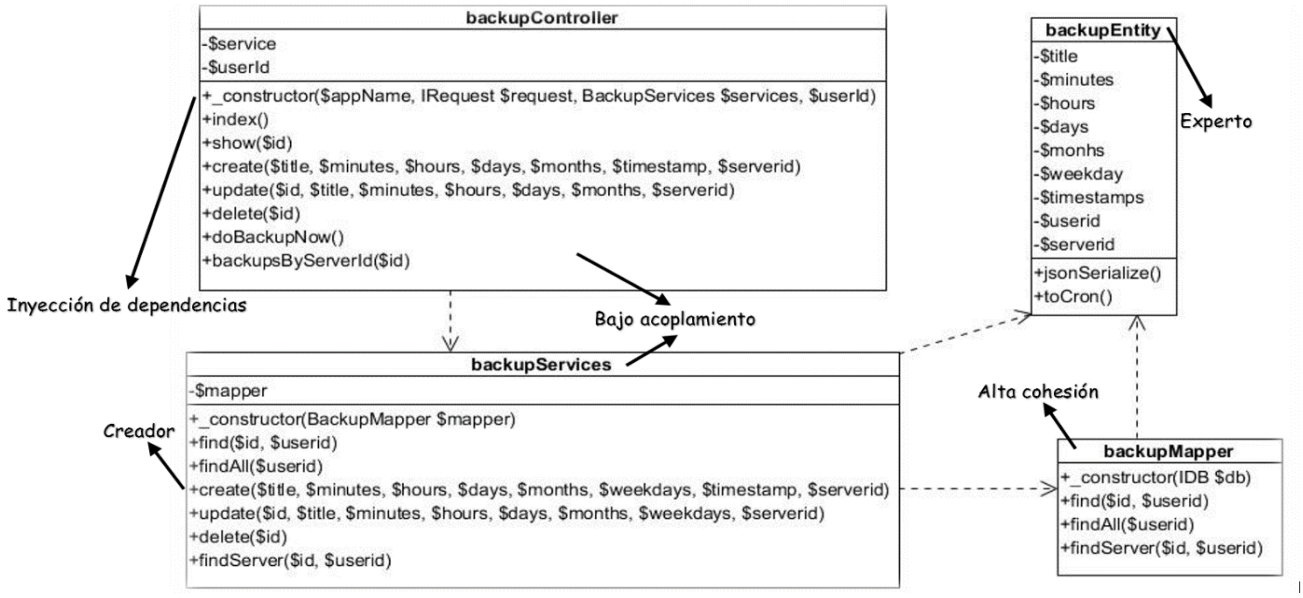


Figura 6. Patrones de diseño GRASP e Inyección de dependencias

Fuente: elaboración propia

Patrones GRASP:

Los patrones GRASP describen responsabilidades que están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. Entre estos patrones se encuentran el experto, el creador, el controlador, bajo acoplamiento, alta cohesión. A continuación, se describen los patrones que fueron empleados en el diseño de la propuesta de solución.

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. Con la aplicación del patrón experto se disminuye el acoplamiento y aumenta la cohesión del diseño. En las clases *BackupEntity(),*

ServerEntity() está empleado a través del método *jsonSerialize()*, presente en ambas clases durante la creación de los objetos *backup* y *server* respectivamente. Además, en estas clases se maneja toda la información referente a los objetos *backup* y *salva* mediante el uso de los métodos *get()*. En la figura 7 se presenta un ejemplo de la utilización de este patrón en la clase *ServerEntity()*.

```
class ServerEntity extends Entity implements JsonSerializable
{
    protected $name;
    protected $type;
    protected $route;
    protected $user;
    protected $pass;
    protected $userId;

    public function jsonSerialize()
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'type' => $this->type,
            'route' => $this->route,
            'user' => $this->user,
            'pass' => $this->pass
        ];
    }
```

Figura 7. 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. Este patrón se evidencia en las clase *BackupServices()* en el método *create()* el cual es responsable de la creación de las instancias de la clase *BackupEntity()*. De manera similar ocurre en la clase *ServerServices(),* en el método *create()* encargado de crear las instancias de la clase *ServerEntity().* En la figura 8 se presenta el uso de este patrón en la clase *BackupServices()*.

```
public function create($name, $type, $route, $user, $pass, $userId)
{
    $serverEntity = new ServerEntity();
    $serverEntity->setName($name);
    $serverEntity->setType($type);
    $serverEntity->setRoute($route);
    $serverEntity->setUser($user);
    $serverEntity->setPass($pass);
    $serverEntity->setUserId($userId);
    return $this->mapper->insert($serverEntity);
}
```

Figura 8. Patrón del diseño GRASP: Creador

Fuente: elaboración propia

Bajo acoplamiento: el objetivo de este patrón es lograr la menor dependencia entre clases, de manera que las modificaciones realizadas en algunas de ellas afecten lo menos posible el funcionamiento del sistema. Asigna responsabilidades de control de flujo del sistema a clases específicas. Un ejemplo donde se evidencia este patrón es entre las clases *BackupController()* y *BackupService()*, pues al existir una sola relación entre ambas, los cambios en una tienen poco influencia en el funcionamiento de la otra. Esta relación también se manifiesta entre las clases *ServerController()* y ServerServices(), así como entre las clases *BackupServices*() y *BackupEntity()*. El promedio de relaciones entre las clases es de 1.6, mientras que 3 es el máximo de relaciones entre clases que existe.

Alta cohesión: a través de este patrón se asegura la necesidad de mantener el sistema con un bajo nivel de complejidad, lo que quiere decir que el sistema debe realizar solo las operaciones necesarias para cumplir una tarea dentro de un área funcional. Este patrón queda evidenciado en cada una de las clases del diseño, ya que cada una responde con las funcionalidades que debe cumplir para realizar los trabajos correspondientes a cada clase. Un ejemplo se encuentra en la clase *BackupMapper()* pues ella maneja todas sus responsabilidades en función de realizar la tarea del acceso a los datos de la base de datos; en la clase *ServerMapper()* funciona de manera similar al ejemplo anterior.

Inyección de dependencias (DI, *Dependency Injection*) es un patrón de diseño de software que consiste en pasar a los componentes de software sus dependencias mediante los constructores, sus propiedades o métodos *setter* (LOCKHART, 2018). Este patrón se utiliza en las clases *BackupController()* y *ServerController()*, como se demuestra a continuación:

```
class BackupController extends Controller
{
    private $service;
    private $serverService;
    private $userId;

    use Errors;

    public function __construct($AppName, IRequest $request,
                                BackupServices $service, ServerServices $serverService, $UserId)
    {
        parent::__construct($AppName, $request);
        $this->service = $service;
        $this->serverService = $serverService;
        $this->userId = $UserId;
    }
```

Figura 9. Patrón del diseño Inyección de dependencias

Fuente: elaboración propia

2.3.2 Modelado de datos

Un modelo de datos se puede interpretar como un esquema que especifica las expresiones permitidas por el propio modelo, comunica las reglas y definiciones esenciales de los datos a los usuarios; además es un esquema lógico que describe la semántica a través de tablas representada por una tecnología de manipulación de datos tal como el lenguaje SQL (PÉREZ, y otros, 2012). El modelo de datos está formado por 3 elementos fundamentales:

- ✓ Entidades: objetos con existencia física o conceptual.
- ✓ Atributos: características que identifican o definen una entidad.
- ✓ Relaciones: dependencias o asociaciones entre las entidades.

La mayoría de los modelos de datos se pueden manifestar a través un diagrama entidad-relación, ya que representa las relaciones entre los objetos de datos (Lucid Software Inc., 2019). A continuación, se muestra el modelo de datos correspondiente a la propuesta de solución. En él se evidencian las entidades *backup_salva* y *backup_server* con sus atributos y relación:

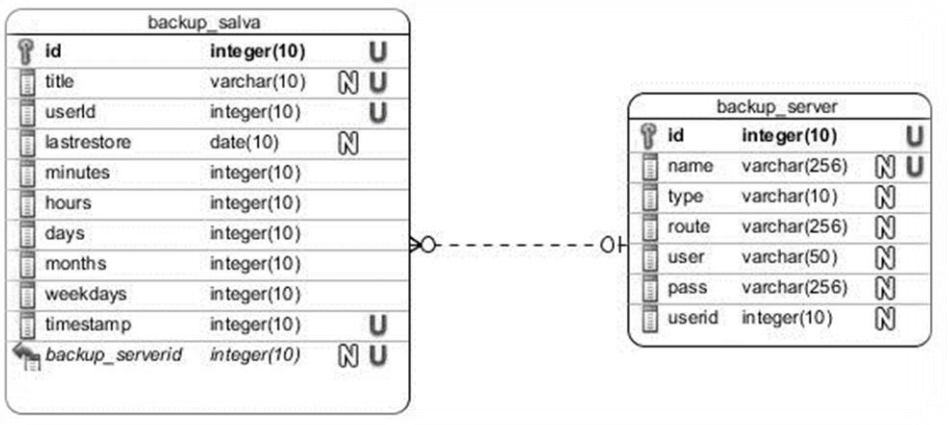


Figura 10. Modelo de datos de la propuesta de solución

Fuente: elaboración propia

2.3.3 Diseño arquitectónico

El diseño arquitectónico es el proceso creativo que intenta establecer una organización del sistema informático que satisfaga los requerimientos funcionales y no funcionales. Su objetivo es representar componentes que interactúen entre ellos y tener asignadas tareas específicas, ser flexible y extensible y representar las relaciones de control entre las partes (ALMEIRA, y otros, 2007). En la propuesta de solución se utiliza el patrón arquitectónico MVC (*Model-View-Controller*, Modelo-Vista-Controlador) descrito a continuación.

Patrón arquitectónico MVC

El patrón arquitectónico MVC divide una aplicación en tres componentes. El Modelo contiene la funcionalidad central y los datos, maneja el comportamiento y las relaciones entre ellos. Las Vistas son lo que utilizan los usuarios para interactuar con la aplicación, cada vista tiene una relación con el controlador, aun cuando varias vistas pueden estar asociadas al mismo modelo. Los Controladores manejan las entradas del usuario al realizar llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario (POTENCIER, 2018). En la siguiente figura se evidencia la estructura de la separación en capas propuesta por el patrón MVC.

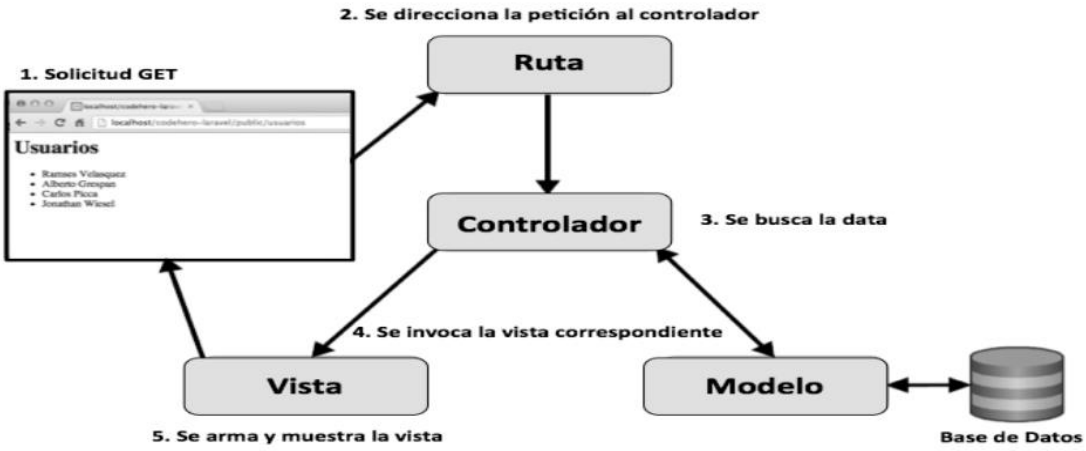


Figura 11. Patrón arquitectónico Modelo-Vista-Controlador

Fuente: (maestrosdelweb, 2016)

En la figura 12 se evidencia como fue empleado el patrón arquitectónico MVC en el módulo para la gestión de copias de seguridad de Nova 360. En ella se presenta la estructura del módulo para la gestión de copias de seguridad en Nova 360 en las siguientes capas:

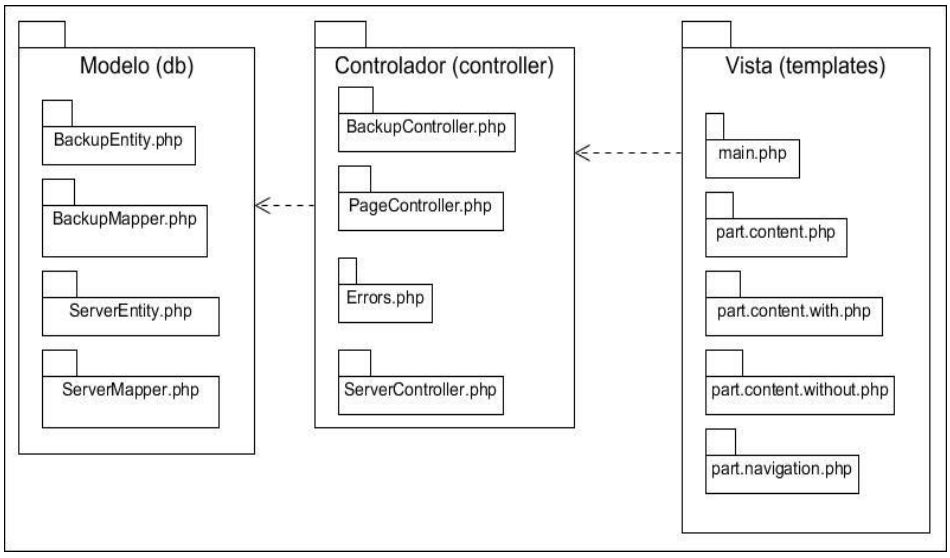


Figura 12. Diseño arquitectónico de la propuesta de solución

Fuente: elaboración propia

- ✓ Modelo: contiene las clases entidades *BackupEntity()* y *ServerEntity()*, y las clases de acceso a datos *BackupMapper()* y *ServerMapper().*
- ✓ Controlador: contienen los controladores para acceder al modelo e invocar las vistas correspondientes a cada uno de estos: *BackupController()*, *PageController(), Errors()* y *ServerController().*
- ✓ Vista: contiene las plantillas que corresponden a cada una de las vistas de la solución, estas son: *main.php*, *part.content.backup.php*, *part.content.with.backup.php*, *part.content.without.backup.php, part.navigation.backup.php*, *part.content.server.php*, *part.navigation.server.php*, *part.list.php*, *part.navigation.list.php* y *part.setting.php*.

Conclusiones del capítulo

El análisis y diseño del módulo para la gestión de copias de seguridad de Nova 360 permitió identificar 12 requisitos funcionales y 6 no funcionales, de acuerdo a las necesidades del cliente. El empleo del patrón arquitectónico MVC 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 empleo de los patrones del diseño GRASP e Inyección de dependencias le brindó una mayor calidad al software desarrollado. El diagrama entidad-relación garantizó el modelado de la base de datos mediante las entidades *backup_salva y backup_server*.

CAPÍTULO 3: Implementación y evaluación del módulo para la gestión de copias de seguridad en Nova 360

El capítulo contiene los productos de trabajos elaborados en las disciplinas de Implementación y pruebas, así como la evaluación del índice de satisfacción grupal sobre el módulo para la gestión de copias de seguridad en Nova 360. En él se presenta la especificación de los estándares de codificación utilizados durante la implementación de la solución; así como el diagrama de despliegue, y un ejemplo de la interfaz gráfica de usuario. Además, se realizan pruebas de software Unitarias, Funcionales e Integración con el objetivo de descubrir y corregir los posibles errores de módulo para la gestión de copias de seguridad en Nova 360.

3.1 Implementación

El objetivo de esta disciplina es construir el sistema informático, a partir de los resultados del Análisis y Diseño (RODRÍGUEZ, 2015). 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 (PRESSMAN, 2010). Durante la implementación se utilizaron los estándares de codificación descritos a continuación.

3.1.1 Estándares de codificación

Un estándar de codificación es un conjunto de reglas, normas o patrones que tienen como objetivo establecer uniformidad en el proceso de generación de un código. Son pautas de programación enfocadas a la estructura física del código para facilitar su lectura, comprensión y mantenimiento (OSORIA, 2013). Para la propuesta de solución se tuvo en cuenta los estándares de codificación para PHP definidos por el Manual del desarrollador de Owncloud (The OwnCloud Developers, 2018), ya que Nova 360 es una personalización de este para las instituciones cubanas. A continuación, se presentan los detalles:

Indentación

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

Tabuladores o Espacios

- ✓ Las estructuras de control deben tener un espacio entre la palabra reservada de la estructura y el signo de apertura de paréntesis para distinguir entre las llamadas de las funciones.
- ✓ Las funciones son llamadas sin espacio entre el nombre de la función, el signo del paréntesis y el primer parámetro; espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (;).

Estándares de nomenclatura

- ✓ Se utilizó *Upper Camel Case* (primera letra de cada palabra en mayúscula) para el nombre de las clases.
- ✓ Se utilizó *Lower Camel Case* (letra inicial de la primera palabra en minúscula y el resto de las palabras comienzan con mayúscula) para los nombres de los métodos de las clases, las variables y funciones.

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

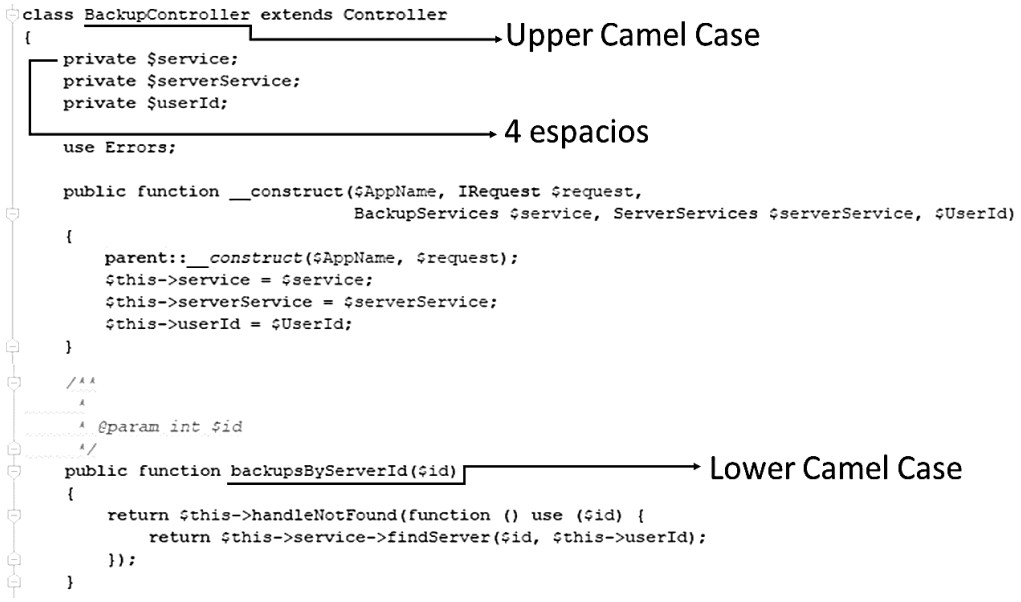


Figura 13. Estándares de codificación

Fuente: elaboración propia

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

En el epígrafe 3.1.2 quedan evidenciados los resultados de la implementación del módulo para la gestión de copias de seguridad de Nova 360 mediante el ejemplo de la interfaz gráfica de usuario mostrada en la figura 14. Para su implementación se tuvo en cuenta las reglas de oro definidas por Roger Pressman (2010) para el diseño de interfaces gráficas de usuario.

3.1.2 Ejemplo de interfaz gráfica de usuario

La interfaz de usuario es una parte de un programa que gestiona la interacción con el usuario basándose en relaciones visuales como iconos, menús, formularios o un puntero. Su principal objetivo es facilitar la comunicación con el ordenador (PRESSMAN, 2010). A continuación se relacionan las reglas empleadas en el diseño e implementación de las interfaces gráficas de usuario:

Reglas de oro (PRESSMAN, 2010)**:**

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

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

En la figura 14 se muestra la interfaz de usuario correspondiente a la gestión de la planificación de una copia de seguridad. En ella se puede adicionar una nueva planificación, se muestra el listado de las planificaciones realizadas y además brinda las opciones de eliminarlas y de cancelar la adición de la nueva planificación.

Figura 14. Interfaz gráfica de usuario para la planificación de copias de seguridad

Fuente: elaboración propia

En el epígrafe 3.2 se presenta el diagrama de despliegue del módulo para la gestión de copias de seguridad de Nova 360. En él se detallan las características del entorno de ejecución de la aplicación.

3.2 Diagrama de despliegue

El diagrama de despliegue es una estructura que muestra la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos de despliegue. En él están representados nodos que corresponden con los dispositivos de hardware o con algún entorno de ejecución de software. Estos nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria (LARMAN, 2010). En la figura 15 siguiente se presenta el diagrama de despliegue elaborado para la propuesta de solución:

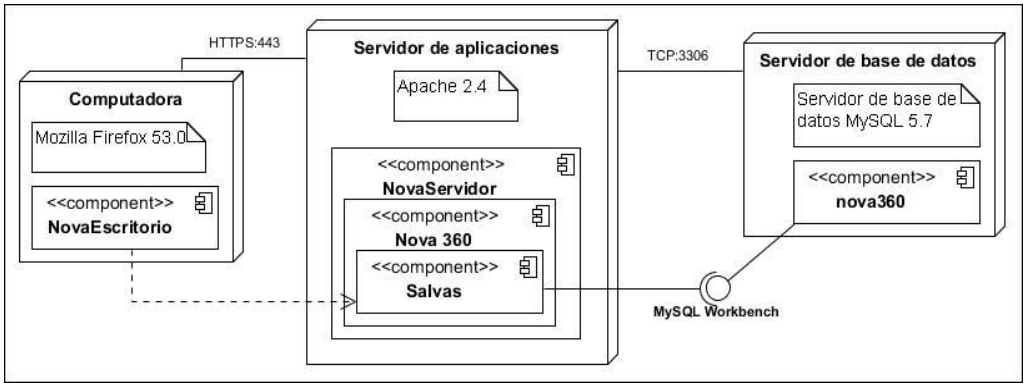


Figura 15. Diagrama de despliegue

Fuente: elaboración propia

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

Descripción de los nodos:

- ✓ Computadora: estación de trabajo desde la cual se accede al módulo para la gestión de copias de seguridad de Nova 360, a través de un navegador web.
- ✓ Servidor de aplicaciones: dispositivo que proporciona los servicios del módulo, y gestiona las funciones del negocio y el acceso a los datos. Para la propuesta de solución se utiliza Apache2 en su versión 2.4.
- ✓ Servidor de base de datos: permite el almacenamiento, modificación y extracción de la información de la base de datos de la aplicación desarrollada. Para estas operaciones se emplea MySQL en su versión 5.7.

Descripción de los componentes:

- ✓ Nova Escritorio: 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, necesaria para la ejecución de Nova 360.
- ✓ Nova 360: personalización de OwnCloud para las instituciones cubanas.

- ✓ Salvas: módulo para la gestión de copias de seguridad de Nova 360.
- ✓ nova360: base de datos de la aplicación para la gestión de copias de seguridad en Nova 360.
- ✓ MySQLWorbench: software a través del cual se puede manejar o gestionar la base de datos nova360.

Descripción de protocolos:

- ✓ HTTPS (*Hipertext Transfer Protocol,* Protocolo de Transferencia de Hipertexto): protocolo de trasferencia de datos en su versión segura; utilizado en el módulo de gestión de copias de seguridad para la transmisión de los datos entre los controladores y la vista de forma segura.
- ✓ TCP (*Transmission Control Protocol*, Protocolo de Transmisión de Control): protocolo de transporte por el que se envían los mensajes o corrientes de datos desde la base de datos hasta la aplicación. Además, a través de este protocolo se garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

3.3 Pruebas de software

Las pruebas de software consisten en la verificación del comportamiento de un programa en un conjunto finito de casos de prueba 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 (ZAPATA, 2013). A continuación, se describen los tipos de prueba de software, métodos y técnicas aplicados en la evaluación de la solución desarrollada.

3.3.1 Tipos de pruebas de software

Pruebas Unitarias

Las pruebas unitarias o de componente consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada (ZAPATA, 2013). Estas pruebas se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional, está correctamente codificado (OJEDA, 2014).

Pruebas Funcionales

Las pruebas funcionales se centran en comprobar que el sistema funcione acorde a los requisitos funcionales y no funcionales, detectando posibles defectos derivados de errores en la fase de 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 (PRESSMAN, 2010).

Pruebas de Integración

Las pruebas de integración consisten en la comprobación de que los elementos del software que interactúan entre sí, funcionan de manera correcta. Uno de sus objetivos es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. Se diseñan para descubrir errores o completitud en las especificaciones de las interfaces (OJEDA, 2014). Estas pruebas cuentan con dos estrategias de prueba: integración descendente en la cual se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal; y la integración ascendente que comienza la construcción y la prueba con los módulos de los niveles más bajos de la estructura del programa (PRESSMAN, 2010).

3.3.2 Métodos de prueba

Método de caja blanca

Las pruebas de caja blanca o pruebas estructurales se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente (OJEDA, 2014). Mediante este método de prueba, el ingeniero del software puede obtener casos de prueba que garanticen que se ejecuten por lo menos una vez todos los caminos independientes de cada módulo; que se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; que se realicen todos los bucles en sus límites y con sus límites operacionales; y efectúen las estructuras internas de datos para asegurar su validez (PRESSMAN, 2010).

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 (OJEDA, 2014).

3.3.3 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 el sistema (SOMMERVILLE, 2011).

Partición de equivalencia

La técnica de partición de equivalencia consiste en probar cada una de las funcionalidades del sistema, 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 (PRESSMAN, 2010).

Pruebas basadas en hilos

Esta técnica fue aplicada durante las pruebas de integración y su objetivo es integrar el conjunto de clases requeridas, para responder una entrada o suceso al sistema. Cada hilo se integra y prueba individualmente (PRESSMAN, 2010).

3.4 Aplicación de las pruebas de software

En el presente epígrafe se presentan los resultados obtenidos de las diferentes pruebas de software anteriormente definidas y aplicadas empleando los métodos y técnicas descritas en los subepígrafes 3.3.2 y 3.3.3 durante la ejecución de las disciplinas Pruebas internas y Pruebas de aceptación propuestas por la metodología de desarrollo de software variación AUP para la UCI.

3.4.1 Pruebas Internas

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

Pruebas unitarias

Las pruebas unitarias se realizaron a través del método de prueba 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 del sistema. El procedimiento de mayor valor tiene una alta probabilidad de contener errores, además de que ofrece una medida del número de pruebas que deben diseñarse para validar la correcta implementación de una determinada función (PRESSMAN, 2010). En la figura 16 se presenta el método *create()* correspondiente al RF1 Adicionar planificación de copia de seguridad.

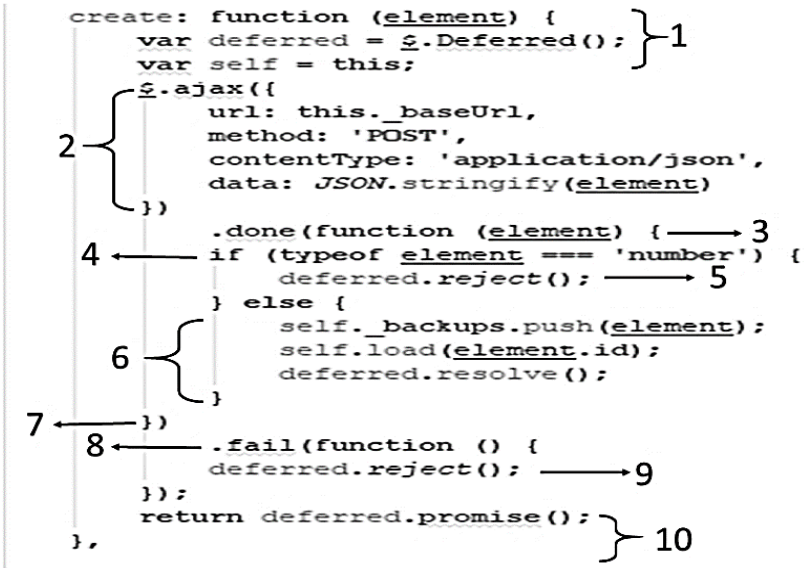


Figura 16. Método *create()* del correspondiente al RF1. Adicionar planificación de copia de seguridad

Fuente: elaboración propia

1. Dibujar el grafo de flujo de la funcionalidad

En la figura 17 se utilizó la notación de grafo de flujo para representar el código de la funcionalidad a probar:

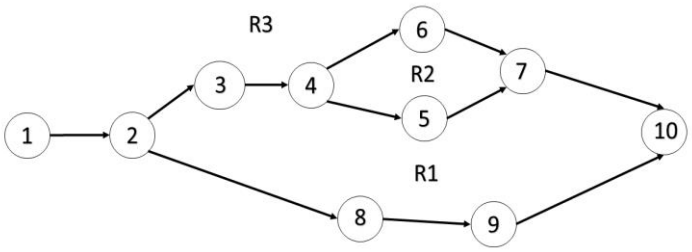


Figura 17. Grafo de flujo

Fuente: elaboración propia

2. Determinar la complejidad ciclomática

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

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

V(G) = 11-10+2 nodos del grafo

V(G) = 3

V(G) = P+1 Donde P es el número de nodos predicados (nodos con más de una

V(G) = 2+1 arista de salida) contenidos en el grafo

V(G) = 3

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

V(G) = 3 en el grafo)

3. Determinar los caminos linealmente dependientes

Camino básico 1: 1, 2, 3, 4, 5, 10

Camino básico 2: 1, 2, 3, 6, 7, 10

Camino básico 3: 1, 2, 8, 9, 10

4. Definir los casos de prueba para comprobar la ejecución

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

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

En la tabla se presenta el diseño de caso de prueba del camino 2 del conjunto de caminos básicos linealmente independientes correspondientes a la funcionalidad *create()*:

Tabla 7. Caso de prueba para el camino 2 del método *create()*

Fuente: elaboración propia

Diseño de caso de prueba para el camino 2	
Descripción	Método para crear una nueva programación de copia de seguridad.
Condiciones	El usuario selecciona la opción de adicionar una nueva programación de una copia de seguridad.
Entradas	object: { id: “111”, title: “new backup”, minutes: “25”, hours: “*”, days: “*/2”,

	months: “1-11/2”, weekdays: “0,6”, timestamp: “1553385464987”, serverid: “35”}
Resultados	Crea una nueva planificación con los valores definidos y se incluye en el listado de las planificaciones creadas.

Luego de realizadas las pruebas unitarias se obtuvo como resultado que el flujo de trabajo de las funcionalidades del módulo para la gestión de copias de seguridad en Nova 360 es correcto; pues se comprobó que cada sentencia del código fuente se ejecuta al menos una vez.

Pruebas funcionales

En las pruebas funcionales se tuvieron cuenta el método de caja negra y la técnica de partición equivalente. Para la aplicación de la técnica de partición equivalente se retomaron los casos de prueba diseñados en el epígrafe 2.2.5 “Validación de requisitos de software” en que se definen las clases de equivalencia válidas y no válidas para cada campo de entrada del sistema. En la figura 18 se presenta un gráfico con los resultados de las pruebas funcionales.

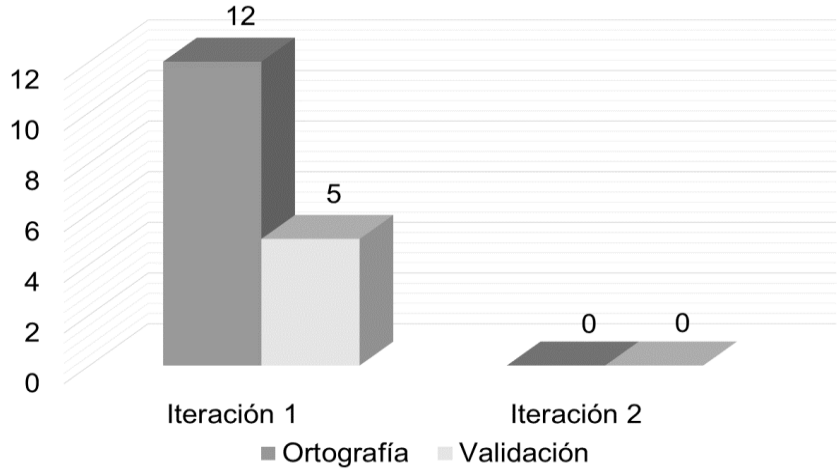


Figura 18. Resultados de las pruebas funcionales

Fuente: elaboración propia

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

Pruebas de integración

Para la prueba de integración se realizó el diseño de caso de prueba presentado en la figura 19 para comprobar la integración del módulo para la gestión de coipas de seguridad en Nova 360.

Descripción general

Permite instalar el módulo para la gestión de copias de seguridad en Nova 360.

Condiciones de ejecución

1. El código fuente del módulo para la gestión de copias de seguridad debe encontrarse en la carpeta *apps/* de Nova 360.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Instalar el módulo para la gestión de copias de seguridad desde Nova 360.	Instalar el módulo para la gestión de copias de seguridad.	Instala satisfactoriamente el módulo para la gestión de copias de seguridad y lo añade al menú de las aplicaciones disponibles de Nova 360.	1. Seleccionar la opción "Ajustes" en el menú superior derecho. 2. Seleccionar la opción "Aplicaciones" en la sección "Administración". 3. Seleccionar la opción "Mostrar aplicaciones inhabilitadas" en el menú "Gestión de apps". 4. Buscar el módulo "Backup" y seleccionar la opción "Activar".
EC 1.2 Instalar el módulo para la gestión de copias de seguridad desde Nova Servidor	Instalar el módulo para la gestión de copias de seguridad.	Instala satisfactoriamente el módulo para la gestión de copias de seguridad y lo añade al menú de las aplicaciones disponibles de Nova 360.	1. Abrir una terminal como "root" 2. Moverse a la carpeta del Nova360 y escribir : sudo -u www-data php occ app:enable backup
EC 1.3 Errores en el fichero info.xml del módulo para la gestión de copias de seguridad	Instalar el módulo para la gestión de copias de seguridad.	El módulo para la gestión de copias de seguridad no se instala.	1. Seleccionar la opción "Ajustes" en el menú superior derecho. 2. Seleccionar la opción "Aplicaciones" en la sección "Administración". 3. Seleccionar la opción "Mostrar aplicaciones inhabilitadas" en el menú "Gestión de apps".

Figura 19. Diseño de caso de prueba para la Prueba de Integración

Fuente: elaboración propia

Para instalar el módulo para la gestión de copias de seguridad en Nova 360, se debe garantizar que el código fuente se encuentre en la carpeta *apps/* de Nova 360. Luego se debe habilitar el módulo para que funcione correctamente. En la figura 19 se muestra el caso de prueba utilizado para corroborar la integración del módulo a la aplicación principal. Para desarrollar esta prueba se tuvieron en cuenta 3 escenarios que representan las posibles situaciones que se pueden presentar cuando se proceda a instalar la solución desarrollada. Como resultado se obtuvo que el módulo para la gestión de copias de seguridad se integró de manera correcta a Nova 360.

3.4.2 Pruebas de aceptación

En la disciplina de Prueba de aceptación se realizan las pruebas de software finales antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (RODRÍGUEZ, 2015). El cliente realizó pruebas funcionales en las que no se detectaron no conformidades y emitió un acta de aceptación de los productos de trabajo en total conformidad con la solución desarrollada (ver Anexo 4).

3.5 Evaluación de la satisfacción del cliente sobre la propuesta de solución

La satisfacción del cliente se ha convertido en uno de los objetivos más importantes dentro del proceso de desarrollo de software en la validación de cualquier investigación científica. La información brindada resulta muy útil para conocer las fortalezas y debilidades de la propuesta realizada (BARROSO RODRÍGUEZ, 2017). Con la finalidad de determinar el índice de satisfacción personal y grupal de los usuarios sobre el módulo para la gestión de copias de seguridad en Nova 360, se aplicó la Técnica de V. A. Iadov.

La técnica de V.A. Iadov está conformada por cinco preguntas: cuatro cerradas y una abierta, las cuales se relacionan a través de lo que se denomina el "Cuadro Lógico de Iadov". Constituye una vía para el estudio de la satisfacción en dependencia del campo donde se aplique, ya que los criterios que se utilizan se fundamentan en las relaciones que se establecen entre tres preguntas que se intercalan dentro de un cuestionario y cuya relación el sujeto desconoce (LÓPEZ, y otros, 2002). Estas preguntas permiten ubicar al o los encuestados en una escala de satisfacción tomando valores del 1 al 6 de la siguiente forma (BARROSO RODRÍGUEZ, 2017):

1. Clara satisfacción
2. Más satisfecho que insatisfecho

3. No definida
4. Más insatisfecho que satisfecho
5. Clara insatisfacción
6. Contradictoria

Para conocer el grado de satisfacción de los usuarios de la propuesta de solución se aplicó una encuesta (ver Anexo 2) a cinco especialistas del proyecto Nova 360. En la tabla 8 se evidencia la relación entre las preguntas cerradas del cuestionario.

Tabla 8. Cuadro lógico de Iadov

Fuente: elaboración propia

5. ¿Qué opina acerca de los beneficios que trae consigo la aplicación para la gestión de copias de seguridad en Nova 360?	2. ¿Actualmente Nova 360 satisface sus necesidades para la gestión de copias de seguridad?								
	No			No sé			Sí		
	3. ¿Considera usted que es necesario contar con una aplicación informática para la gestión de copias de seguridad en Nova 360?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	3	4
No me gusta	6	6	6	6	4	4	6	6	5
Me da lo mismo	2	3	6	3	3	3	6	6	4

El número de la interrelación de las tres preguntas indica la posición en la escala de satisfacción siguiente: clara satisfacción (A), más satisfecho que insatisfecho (B), no definida (C), más insatisfecho que satisfecho (D), clara insatisfacción (E) y contradictoria (F). A partir de la cantidad de respuestas en cada categoría se puede calcular el Índice de Satisfacción Grupal (ISG) utilizando la siguiente fórmula:

$ISG = \frac{\mathrm{A}(+1)+\mathrm{B}(+0.5)+\mathrm{C}(0)+\mathrm{D}(-0.5)+\mathrm{E}(-1)}{\mathrm{N}}$ Donde N es la cantidad total de respuestas.

El valor del ISG permite identificar las siguientes categorías grupales:

- ✓ Máxima insatisfacción: desde -1 hasta -0.49
- ✓ Más insatisfecho que satisfecho: desde -0.5 hasta -0.1
- ✓ No definido y contradictorio: 0
- ✓ Más satisfecho que insatisfecho: desde 0.1 hasta 0.49
- ✓ Máximo de satisfacción: desde 0.5 hasta 1

Los valores que se encuentran comprendidos entre -1 y -0.5 indican insatisfacción; los comprendidos entre -0.49 y +0.49 evidencian contradicción y los que están entre 0.5 y 1 indican que existe satisfacción (LÓPEZ, et al., 2002).

Resultados obtenidos

En la tabla 9 quedan reflejados los resultados obtenidos luego de la aplicación de la encuesta a especialistas del Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI):

Tabla 9. Resultados de la escala de satisfacción

Fuente: elaboración propia

Categorías grupales de satisfacción	N=5	Escala
Clara satisfacción	4	A
Más satisfecho que insatisfecho	1	B
Indefinido	0	C
Más insatisfecho que satisfecho	0	D

Máxima insatisfacción	0	E
Contradictorio	0	F

Cálculo del Índice de Satisfacción Grupal (ISG)

ISG = A (+1) + B (+0.5) / 5

ISG = (4 (+1) + 1 (+0.5)) / 5

ISG = 0.90

El valor obtenido del ISG fue de 0.90, lo que indica que entre los encuestados existe la mayor satisfacción con respecto al módulo de gestión de copias de seguridad de Nova 360. A través de las respuestas conseguidas, se pudieron ratificar los beneficios que trae consigo la solución desarrollada. Por lo que se puede afirmar que se cumplió el objetivo de la investigación.

Conclusiones del capítulo

Se logró la informatización de la gestión de copias de seguridad en Nova 360 a través del módulo desarrollado. La aplicación de las pruebas unitarias permitió corroborar que el flujo de trabajo de las funcionalidades implementadas es el correcto, pues cada sentencia se ejecuta al menos una vez. La ejecución de las pruebas funcionales posibilitó identificar las deficiencias de la propuesta de solución y garantizar el cumplimiento de los requisitos definidos. La realización de la prueba de integración demostró que el software desarrollado fue compatible con Nova 360. Mediante las pruebas de aceptación y la técnica de Iadov se confirmó la alta satisfacción del cliente con un ISG de 0.90 hacia el módulo para la gestión de copias de seguridad en Nova 360.

CONCLUSIONES GENERALES

La presente investigación documentó el proceso de desarrollo del módulo para la gestión de copias de seguridad en Nova 360. Una vez finalizado el mismo se pueden arribar a las siguientes conclusiones:

- ✓ El análisis del marco teórico sobre el proceso de gestión de copias de seguridad para Nova 360, así como el estudio de aplicaciones informáticas para la gestión de copias de seguridad, demostraron la necesidad de desarrollar una aplicación que contribuya con la gestión de copias de seguridad en Nova 360.
- ✓ Los 12 requisitos funcionales y los 6 no funcionales definidos permitieron cumplir con las necesidades del cliente. El análisis y diseño propició la elaboración de un módulo para la gestión de copias de seguridad en Nova 360. El uso de los patrones de diseño GRASP e Inyección de dependencias y el patrón arquitectónico MVC brindó una mayor calidad al software desarrollado.
- ✓ Se desarrolló un módulo que gestiona las copias de seguridad que se realizan en Nova 360 permitiendo cumplir con las exigencias del cliente y las restricciones del diseño elaborado.
- ✓ La aplicación de las pruebas de software permitió evaluar la solución desarrollada y garantizar el correcto funcionamiento del módulo implementado. La utilización de la técnica de Iadov posibilitó conocer la satisfacción de los usuarios hacia el módulo de gestión de copias de seguridad de Nova 360 con un ISG de 0.9.

RECOMENDACIONES

Se recomienda implementar un mecanismo que soporte el esquema de respaldo *Grandfather-Father-Son*, teniendo en cuenta que las copias mensuales (*Grandfather*), que consiste en hacer copias completas, se ejecutarán el primer y último día del mes; para las copias semanales (*Father*), las cuales son copias diferenciales se realizarán todos los sábados del mes; y las copias diarias (*Son*), copias incrementales se efectuaran los días de la semana comprendidos entre lunes y viernes.

REFERENCIAS BIBLIOGRÁFICAS

Academic. 2017. Enciclopedia universal. *Demonio (informatica).* [En línea] 2017. [Citado el: 5 de diciembre de 2018.] http://enciclopedia_universal.esacademic.com/20876.

ALMEIRA, ADRIANA y PÉREZ, VANINA. 2007. *Arquitectura de software. Estilos y Patrones.* Facultad de Ingeniería, Universidad Nacional De La Patagonia San Juan Bosco. Argentina : s.n., 2007. Tesina presentada a la para la obtención del título de Licenciatura en Informática.

ÁLVARES, SARA. 2017. Desarrolloweb.com. *Qué es un diccionario de datos.* [En línea] 2017. [Citado el: 12 de febrero de 2019.] https://desarrolloweb.com/faq/452.php.

ANDREARR. 2014. Hipertextual. *Qué es un sistema de control de versiones y por qué es tan importante.* [En línea] 30 de 4 de 2014. [Citado el: 10 de diciembre de 2018.] https://hipertextual.com/archivo/2014/04/sistema-control-versiones/.

Apache. 2018. The Apache Software Foundation. [En línea] 30 de abril de 2018. [Citado el: 31 de octubre de 2018.] https://www.apache.org/.

Archlinux. 2018. Rsync. [En línea] 21 de octubre de 2018. [Citado el: 5 de noviembre de 2018.] https://wiki.archlinux.org/index.php/Rsync_(Espa%C3%B1ol)#Como_herramienta_para_copias_de_seguridad.

Aritmetrics. 2018. Gosario digital. *¿Qué es un entorno de desarrollo?* [En línea] Octubre de 2018. [Citado el: 30 de octubre de 2018.] https://www.arimetrics.com/glosario-digital/entorno-de-desarrollo.

BARROSO RODRÍGUEZ, YADIRA. 2017. *Sistema de apoyo al diagnóstico a distancia de enfermedades géneticas basado en mapas cognitivos difusos.* Pinar del Río : Revista Ciencias Médicas de Pinar del Río, 2017. Vols. vol 21(6)810-819.

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

Concepto.de.com. 2018. Concepto.de. *Concepto de archivo.* [En línea] Noviembre de 2018. [Citado el: 20 de octubre de 2018.] https://concepto.de/archivo-informatico/.

DALVIK, JOSÉ. 2018. El Conspirador. *Qué es y para qué sirve un modelo conceptual.* [En línea] Diciembre de 2018. [Citado el: 10 de febrero de 2018.] https://www.elconspirador.com/2013/12/21/que-es-y-para-que-sirve-un-modelo-conceptual/.

Definiciones.de. 2018. Definiciones.de. *Concepto de lenguajes de programación.* [En línea] Octubre de 2018. [Citado el: 20 de noviembre de 2018.] https://concepto.de/lenguaje-de-programacion/.

DÍAZ, FRANCISCO, y otros. 2010. *Metodologías para la evaluación de herramientas Free/Open Source para pruebas de software.* Facultad de Informática, Universidad de La Plata. Buenos Aires, Argentina : s.n., 2010.

DragonJar. 2018. 10 extraordinarias herramientas para hacer copias de seguridad en Linux. [En línea] Octubre de 2018. [Citado el: 12 de diciembre de 2018.] https://www.dragonjar.org/herramientas-para-backup-linux.xhtml.

FERNÁNDEZ, YUSLEYDIS. 2013. *Metodología para desarrollar la distribuición cubana GNU/Linux Nova.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2013.

GONZÁLES, JOSÉ. 2015. *Desarrollo de sitios web con PHP y MySQL.* 2015.

GUERRA, CESAR ARTURO. 2018. Software Guru. *Obtención de requerimientos. Técnicas y Estrategias.* [En línea] Diciembre de 2018. [Citado el: 28 de febrero de 2019.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

HERNÁNDEZ, ENRIQUE. 2018. *El lenguaje unificado de modelado.* 2018.

HILBERT, MARTIN. 2011. Maping out the transition toward information societies: social nature, growth, and policies. [En línea] Febrero de 2011. [Citado el: 10 de octubre de 2018.] https://www.researchgate.net/publication/49826525_The_World's_Technological_Capacity_to_Store_Communicate_and_Compute_Information/download.

HOOGENRAAD, WIM. 2018. ITpedia. *Casos de prueba, ejemplos y mejores prácticas.* [En línea] Junio de 2018. [Citado el: 19 de marzo de 2019.] https://www.itpedia.nl/es/2018/06/01/testcases-voorbeelden-en-best-practices/.

IBM Corp. 2018. Artefacto. *Prototipo de interfaz de usuario.* [En línea] Diciembre de 2018. [Citado el: 10 de enero de 2019.]

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

JetBrains. 2018. PhpStorm. [En línea] 25 de Octubre de 2018. [Citado el: 25 de diciembre de 2018.] https://www.jetbrains.com/phpstorm/.

JOHNSON, KATE. 2017. SpinBackup. *Cloud storage vs. Cloud Backup.* [En línea] 2 de Diciembre de 2017. [Citado el: 25 de octubre de 2018.] https://spinbackup.com/blog/cloud-storage-vs-cloud-backup/.

LARMAN, CRAIG. 2010. *UML y patrones. Introducción al análisis y diseño orientado a objetos.* segunda edición. México : Prentice Hall, 2010.

LOCKHART, JOSH. 2018. Uniwebsidad. *PHP, la manera correcta.* [En línea] Diciembre de 2018. [Citado el: 20 de enero de 2019.] http://uniwebsidad.com/libros/php-correcto.

LÓPEZ, ALEJANDRO y GONZÁLES, VIVIANA. 2002. La técnica de Iadov. [En línea] abril de 2002. [Citado el: 11 de abril de 2019.] https://www.efdeportes.com/efd47/iadov.htm.

Lucid Software Inc. 2018. Lucidchart. *¿Qué es el lenguaje unificado de modelado (UML)?* [En línea] 2018. [Citado el: 25 de Octubre de 2018.] https://www.lucidchart.com/pages/es/qu%C3%A9-es-el-lenguaje-unificado-de-modelado-uml.

—. 2019. Lucidchart. *Qué es un modelo de base de datos.* [En línea] Enero de 2019. [Citado el: 23 de enero de 2019.] https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos.

maestrosdelweb. 2016. Patrón arquitectónico Modelo-Vista-Controlador. [En línea] 2016. [Citado el: 11 de abril de 2019.] http://www.maestrosdelweb.com/.

MARR, BERNARD. 2018. Forbes. *How Much Data Do We Create Every Day?* [En línea] 2018. [Citado el: 21 de abril de 2019.] https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#2e92032560ba.

MATO , ROSA MARÍA. 2014. *Sistemas de bases de datos.* La Habana : s.n., 2014.

MAURYA, RAJKUMAR. 2018. H2S MEDIA. *6 Best Open source personal Cloud software to setup cloud storage.* [En línea] 19 de Febrero de 2018. [Citado el: 24 de octubre de 2018.] https://www.how2shout.com/tools/best-open-source-personal-cloud-software-cloud-storage.html.

MONFERRER, RAÚL. 2015. *Especificación de requisitos según el estándar IEEE 830.* Ciudad de Valencia : s.n., 2015.

MORRISON, HIDEEKEL. 2016. Audiencia Electrónica. *Cifras del volúmen de datos que se mueven por internet.* [En línea] Octubre de 2016. [Citado el: 24 de octubre de 2018.] https://www.audienciaelectronica.net/2010/10/cifras-del-volumen-de-datos-que-se-mueven-por-internet/.

Mozilla. 2018. Moz://a. *MDN web docs.* [En línea] Octubre de 2018. [Citado el: 27 de noviembre de 2018.] https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript.

OJEDA, OLIVER. 2014. Técnicas de pruebas. [En línea] 2014. [Citado el: 23 de marzo de 2019.] https://es.slideshare.net/catalinocordero/unidad-10-tecnicas-de-pruebas.

Oracle Corporation. 2018. MySQL Workbench. [En línea] diciembre de 2018. [Citado el: 10 de enero de 2019.] https://www.mysql.com/products/workbench/.

—. 2018. Oracle. [En línea] 2018. [Citado el: 23 de dicembre de 2018.] http://www.oracle.com/us/products/mysql/overview/index.html.

OSORIA, DAYAMÍ. 2013. *Módulo para la extracción y representación de los metadatos en el Sistema de Gestión Documental de audio y video digitales TeVeo Plus V1.0.* Universidad de las Ciencias informáticas. La Habana : s.n., 2013. Trabajo de diploma para optar por el título de Ingeriero en Ciencias Informáticas.

PAUPIER, FABIEN. 2017. Appvizer. *Copia de Seguridad (Backup), Almacenamiento y Gestión de Archivos en la nube: ¿Cuál es la diferencia?* [En línea] 5 de Mayo de 2017. [Citado el: 28 de octubre de 2018.] https://www.appvizer.es/revista/it/copia-de-seguridad/copia-de-seguridad-backup-almacenamiento-y-gestion-de-archivos-en-la-nube-cual-es-la-diferencia.

PÉREZ, JULIÁN y GARDEY, ANA. 2012. Definiciones.de. *Definición de modelo de datos.* [En línea] 2012. [Citado el: 23 de marzo de 2019.] https://definicion.de/modelo-de-datos/.

—. 2018. Definiciones.de. *Definicion de HTML.* [En línea] Octubre de 2018. [Citado el: 26 de enero de 2019.] https://definicion.de/html/.

PÉREZ, OSIRIS. 2016. *Descripción de los procesos de validación y administración de requisitos.* Ciudad de la Habana : s.n., 2016.

—. 2017. *Técnicas de obtención de requisitos.* La Habana : s.n., 2017.

POTENCIER, FABIEN. 2018. Uniwebsidad. *El tutorial de Jobeet.* [En línea] 2018. [Citado el: 24 de marzo de 2019.] https://uniwebsidad.com/libros/jobeet-1-4/capitulo-4/la-arquitectura-mvc.

PRESSMAN, ROGER. 2010. *Ingeniería de software. Un enfoque práctico.* séptima edición. s.l. : McGraw-Hill, 2010. 978-0-07-337597-7.

Redessil.com. 2018. Redessil.com. *Copias de seguridad.* [En línea] 2018. [Citado el: 21 de octubre de 2018.] http://www.redessil.com/copias-de-seguridad.

RODRÍGUEZ, MARCOS. 2016. Viva Ubuntu. Tutoriales. *BACULA BACKUP: Instalación del servidor.* [En línea] 17 de Abril de 2016. [Citado el: 23 de noviembre de 2018.] http://vivaubuntu.com/bacula-backup-instalacion-del-servidor/.

RODRÍGUEZ, TAMARA. 2015. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : s.n., 2015.

ROUSE, MARGARET. 2016. SearchDataCenter. *Servidor de aplicaciones.* [En línea] 2016. [Citado el: 12 de noviembre de 2018.] https://searchdatacenter.techtarget.com/es/definicion/Computacion-en-la-nube.

SOMMERVILLE, IAN. 2011. *Software Engineering.* novena edición. s.l. : Person Education Inc., 2011.

The OwnCloud Developers. 2018. *OwnCloud Developer Manual.* 2018.

TWEEDIE, MITCHELL. 2017. WorkingMouse. *Software Licensing and Cloud Service Models.* [En línea] 5 de Diciembre de 2017. [Citado el: 21 de octubre de 2018.] https://workingmouse.com.au/innovation/software-licensing-and-cloud-service-models.

Visual Paradigm Internationals. 2018. Visual Paradigm. [En línea] 2018. [Citado el: 21 de noviembre de 2018.] http://www.visual-paradigm.com/.

W3C. 2018. W3C. *Cascading Style Sheets.* [En línea] Diciembre de 2018. [Citado el: 23 de enero de 2019.] https://www.w3.org/Style/CSS/.

ZAPATA, JAVIER. 2013. Introducción a las pruebas. [En línea] 2013. [Citado el: 19 de marzo de 2019.] https://pruebasdelsoftware.wordpress.com/author/javierzapatasanchez/.

ANEXOS

Anexo1: Entrevista realizada a los especialistas del equipo de desarrollo del proyecto Nova 360 de CESOL

Objetivo: conocer el contexto del proyecto Nova 360, las herramientas y tecnologías compatibles con el entorno de desarrollo de dicho proyecto, así como los sistemas que se integran con Nova 360 y las características que debe tener un módulo para ser instalado en dicho software.

1. ¿Qué es Nova 360?
2. ¿Cuáles son las características de Nova 360?
3. ¿Cuáles son los sistemas que se integran con Nova 360?
4. ¿Cree usted que es importante el uso de un módulo que permita la gestión de copias de seguridad en Nova 360? ¿Por qué?
5. ¿Qué características debe tener este módulo?

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

Objetivo: evaluar la satisfacción de los usuarios potenciales de la solución desarrollada.

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

1. ¿Considera usted necesario el uso de aplicaciones que permitan la gestión de copias de seguridad en las diferentes variantes de Nova?

   Sí__ No__ No sé__

2. ¿Actualmente Nova 360 satisface sus necesidades para la gestión de copias de seguridad?

   Sí__ No__ No sé __

3. ¿Considera usted que es necesario contar con una aplicación informática para la gestión de copias de seguridad en Nova 360?

   Sí__ No__ No sé __

4. Luego de haber interactuado con la aplicación para la gestión de copias de seguridad en Nova 360, refleje en qué medida le gusta la solución desarrollada:

   __Me gusta mucho
   __Me disgusta más de lo que me gusta
   __Me gusta más de lo que me disgusta
   __No me gusta
   __Me da lo mismo
   __No sé decir

5. ¿Qué opina acerca de los beneficios que trae consigo la aplicación para la gestión de copias de seguridad en Nova 360?

Anexo 3: Guía de observación para verificar la gestión de copias de seguridad en aplicaciones informáticas.

Observadora: Miosotis Toledo Rodríguez

Lugar: Laboratorio de práctica profesional del Centro de Software Libre (CESOL)

Objetivo: Identificar los elementos fundamentales para la gestión de copias de seguridad en aplicaciones informáticas.

1. Datos de las aplicaciones informáticas que permiten la gestión de copias de seguridad:
    - ✓ Nombre de las aplicaciones
    - ✓ Tipo de licencia
    - ✓ Cuenta con interfaz gráfica de usuario
    - ✓ Compatibilidad con diferentes sistemas operativos
2. Características de las aplicaciones informáticas que permiten la gestión de copias de seguridad:
    - ✓ ¿Cómo son las aplicaciones informáticas que permiten la gestión de copias de seguridad?
    - ✓ ¿Qué herramientas necesitan para ser instaladas?
    - ✓ ¿Cuáles son los pasos a seguir para instalar estas herramientas?
3. Impacto social de las aplicaciones informáticas que permiten la gestión de copias de seguridad.

Anexo 4: Acta de aceptación de los productos de trabajo

UCI Universidad de las Ciencias Informáticas

Acta de aceptación de productos de trabajo

ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En cumplimiento del **Convenio de colaboración** establecido entre el **Centro de Software Libre (CESOL)** y la estudiante **Miosotis Toledo Rodríguez** de la Facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto: **Módulo para la gestión de copias de seguridad en Nova 360**, se hace entrega del producto que se relaciona a continuación:

- Aplicación web para la gestión de copias de seguridad en Nova 360

La parte Cliente, luego de haber revisado el producto de trabajo relacionado anteriormente procede a firmar la aceptación de los mismos en total conformidad.

Entrega	**Recibe**
Nombre y apellidos: Miosotis Toledo Rodríguez	**Nombre y apellidos:** Ing. Hanny Valdés Hernández
Cargo: Estudiante Facultad 1	**Cargo:** Jefa del Departamento de Servicio de Integración en Migración Asesoría y Soporte
Firma:	**Firma:**

CESOL Facultad 1

Fecha: 02/04/2019