UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

"Configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova"

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

Autor:

Erian Veliz Beltran

Tutor:

Ing. Javier Junquera Falcón
MSc. PA Yurisbel Vega Ortiz

Declaración jurada de autoría

Declaro por este medio que yo Erian Veliz Beltran, con carné de identidad 97111406980, ser el único autor de este trabajo de diploma titulado “Configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2022.

_____

Firma del autor

Erian Veliz Beltran

_____

Firma del tutor

Ing. Javier Junquera Falcón

_____

Firma del tutor

MSc. PA Yurisbel Vega Ortiz

Resumen

Como parte del proceso de desarrollo de software y la migración nacional a software libre surge Nova, el cual es un sistema operativo desarrollado en el centro de software libre de la UCI, CESOL. Como todo proyecto en desarrollo existen problemas que se van dando solución a medida que avanza el proyecto, uno de estos problemas es acercar el software libre al público inexperto, suavizando el tránsito de sistemas operativos más convencionales como Windows a otros como Linux desarrollados con software libre. Para cumplir con el objetivo de Nova de proveer un sistema como e intuitivo se desarrolló el presente trabajo de investigación, el cual tiene como meta desarrollar un módulo que permita agrupar las diferentes opciones de configuración del entorno de escritorio de Nova presentes en el centro de control de Gnome Shell. Para esto, se estudiaron herramientas que permitieran configurar el entorno de escritorio de Gnome Shell utilizadas en la actualidad, y se determinaron las características y cualidades que pudieran mejorar la configuración del entorno de escritorio de Gnome Shell. Además, se documentaron las tecnologías, herramientas, lenguajes a utilizar en la construcción del módulo y la definición de los elementos necesarios para el exitoso desarrollo de este, así como los artefactos requeridos por la metodología de desarrollo Variación del Proceso Unificado Ágil para la Universidad de las Ciencias Informáticas. Además, este módulo fue sometido a pruebas de software para verificar su calidad y correcto funcionamiento. El principal resultado de la investigación fue la obtención de un módulo que permite mejorar la configuración del entorno de escritorio de Nova desde el centro de control de Gnome Shell.

Índice

Índice de tablas

Índice de imágenes

Introducción

La sociedad actual está en un constante proceso de informatización y Cuba no es una excepción. En búsqueda de cumplir con los objetivos de la revolución y enfrentar la guerra no convencional a la que el país está involucrado abrió sus puertas La Universidad de Ciencias Informáticas el 23 de septiembre de 2002. Durante muchos años esta institución ha desarrollado muchos proyectos, uno de los más destacados es el sistema operativo Nova.

Un sistema operativo es un conjunto de programas que permite manejar la memoria, disco, medios de almacenamiento de información y los diferentes periféricos o recursos de una computadora, como son el teclado, el ratón, la impresora, la placa de red, entre otros. Los periféricos utilizan un driver o controlador y son desarrollados por los fabricantes de cada equipo. Existen diferentes sistemas operativos como Windows, Linux, MAS OS, en sus diferentes versiones. También los teléfonos y tabletas poseen un sistema operativo. Dentro de las tareas que realiza el sistema operativo, en particular, se ocupa de gestionar la memoria de nuestro sistema y la carga de los diferentes programas, para ello cada programa tiene una prioridad o jerarquía y en función de esta contará con los recursos de nuestro sistema por más tiempo que un programa de menor prioridad. (1)

GNU (acrónimo recursivo de **GNU is Not Unix**) es un sistema operativo de software libre, es decir, respeta la libertad de los usuarios. Este sistema operativo consiste en paquetes de GNU (programas publicados específicamente por el proyecto GNU) además de software libre publicado por terceras partes. El desarrollo de GNU ha permitido que se pueda utilizar un ordenador sin software que se pase por alto la libertad de los usuarios. GNU es un sistema operativo de tipo Unix, lo cual significa que se trata de una colección de muchos programas: aplicaciones, bibliotecas, herramientas de desarrollo y hasta juegos. Su desarrollo se inició en enero de 1984, se conoce como Proyecto GNU. En un sistema de tipo Unix, el programa que asigna los recursos de la máquina y se comunica con el hardware se denomina «kernel» (o núcleo). GNU se usa generalmente con un kernel llamado «Linux». Esta combinación es el sistema operativo GNU/Linux. Millones de personas usan

GNU/Linux, aunque muchos lo llaman erróneamente «Linux». El desarrollo del kernel propio de GNU, el GNU Hurd, se inició en 1990 (antes de que comenzara a desarrollarse el kernel Linux). Programadores voluntarios continúan desarrollando Hurd por tratarse de un proyecto técnico interesante. (2)

Un software libre como se había mencionado anteriormente es aquel que respeta la libertad de los usuarios y la comunidad, es decir que pueden copiar, distribuir, estudiar, modificar y mejorar el software. Con estas libertades los usuarios, tanto individuales como en forma colectiva, controlan lo que hace el programa. Cuando los usuarios no controlan lo que hace el programa, el programa controla al usuario. El programador controla el programa y, a través del programa, controla a los usuarios. Un programa que no es libre, llamado privativo, es por lo tanto un instrumento de poder injusto. Por tanto, el software libre es una cuestión de libertad, no de precio.

Nova es una distribución de GNU/Linux desarrollada por estudiantes y profesores de la Universidad de las Ciencias Informáticas, con la participación de miembros de otras instituciones, para apoyar la migración a tecnologías de software libre y código abierto en Cuba. Provee un sistema cómodo, enfocado al usuario final, garantizando una interacción intuitiva que persigue minimizar el cambio brusco al que se enfrentan las personas familiarizadas con sistemas Microsoft Windows. (2)

En el año 2010 se realizó una auditoría en los proyectos Nova Escritorio, Nova Ligero, Nova Base y Nova Servidores donde se detectaron un conjunto de desviaciones relacionadas con las áreas de gestión de la configuración de software, administración de requisitos, aseguramiento de la calidad y la planificación de los proyectos. (3)

Como todo proyecto en desarrollo, Nova ha ido actualizándose y adaptándose desde el año 2009 para el interés nacional. Nació para garantizar a la nación cubana independencia tecnológica, seguridad, adaptabilidad y sostenibilidad. La primera versión, presentada en la Feria Internacional Cubana de Informática, se basaba en la distribución de Gentoo, mientras comenzaba con la versión 2.1, publicada al año siguiente, tuvo lugar la transición definitiva a Ubuntu como base de trabajo. La versión 3.0, desde la interfaz similar a la de Windows 7, se presentó en la misma feria en 2013.

Situación problemática

En el Entorno de Escritorio de Nova actual no existe un módulo de configuración del escritorio, es decir una sección de la herramienta “configuración” donde se agrupen todas las opciones para personalizar el escritorio como, por ejemplo “Fondo de escritorio” o “Barra de tareas”, estas están dispersas por dicha herramienta lo que las hace difícil de encontrar o siquiera saber que existen, la mayoría en “Preferencias” o “Herramientas del sistema” los cuales tienen una gran cantidad de opciones extras que no se utilizan con frecuencia para dicha configuración, un ejemplo de esto es la herramienta de “Configuración de la sesión de escritorio”, además no cuenta con una opción para cambiar los temas de ventanas e iconos o el tamaño y estilo de las letras. Todo esto atenta contra la intención de Nova de proveer un sistema cómodo e intuitivo.

Problema de investigación

¿Cómo mejorar la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova?

Objeto de estudio

Proceso de configuración del entorno de escritorio en la distribución cubana de GNU/Linux Nova.

Campo de acción

Configuración del entorno de escritorio de Gnome Shell desde el centro de control en la distribución cubana de GNU/Linux Nova.

Objetivo general

Desarrollar un módulo que agrupe todas las herramientas para la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova.

Objetivos específicos

1. Elaborar el marco teórico de la investigación sobre el proceso de configuración del entorno de escritorio de Gnome Shell.
2. Diseñar un módulo que permita mejorar la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova
3. Implementar un módulo que permita mejorar la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova
4. Evaluar el módulo que permita mejorar la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova

Tareas de investigación

Para dar cumplimiento a los objetivos específicos expuestos anteriormente se plantean las siguientes tareas de investigación:

1. Estudio y análisis de las soluciones existentes para mejorar la configuración del entorno de escritorio.
2. Caracterización de la metodología, las tecnologías y las herramientas a utilizar para el desarrollo de la solución.
3. Levantamiento y definición de requisitos funcionales y no funcionales.
4. Diseño de la estructura y comportamiento de los componentes de la propuesta de solución.
5. Implementación de los componentes diseñados.
6. Validación de la propuesta de solución.

Preguntas científicas

En aras de encaminar la investigación y lograr el objetivo, se formularon las siguientes preguntas científicas:

1- ¿Cuáles son los principales fundamentos teóricos-metodológicos que sustentan el proceso de configuración de entorno de escritorio de Gnome Shell?
2- ¿Qué elementos se deben tener en cuenta para diseñar un módulo que permita mejorar la configuración del escritorio de Gnome Shell desde el Centro de Control de Nova?
3- ¿Qué componentes son necesarios implementar para obtener un módulo que permita mejorar la configuración del entorno de Gnome Shell desde el Centro de Control de Nova?
4- ¿Qué técnicas y pruebas de software se deben aplicar para evaluar el módulo para la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova?

Métodos de la investigación científica

Para el desarrollo se emplearon los siguientes métodos de investigación:

Métodos del nivel teórico:

1. **Análisis-síntesis:** se refiere a dos actividades complementarias en el estudio de realidades complejas. El análisis consiste en la separación de las partes de esas realidades hasta llegar a conocer sus elementos fundamentales y las relaciones que existen entre ellos. La síntesis se refiere a la composición de un todo por reunión de sus partes o elementos. Esta construcción se puede realizar uniendo las partes, fusionándolas u organizándolas de diversas maneras. Aplicando esta metodología puede extraer la lista de todas las posibles herramientas de configuración del

escritorio además del conocimiento que posee el público general sobre las herramientas de configuración del escritorio.

2. **Modelación:** La modelación es el proceso mediante el cual se crea una representación o modelo para investigar la realidad. (4) En este proyecto se realizó Las Historias de Usuarios, Diagrama de clases, Diagrama de componentes.

Métodos del nivel empírico:

1. **Observación:** Guía de observación para entender el funcionamiento de un módulo de configuración del escritorio.
2. **Entrevista:** Lista de preguntas para determinar cuál es el nivel de conocimiento del cliente sobre las opciones de configuración del escritorio.
3. **Encuesta:** Lista de preguntas dirigidas a un grupo de personas ajenas al proyecto para determinar cuáles son las opciones de configuración más utilizadas.

Estructura de la investigación

Capítulo 1: Fundamentación teórica, definición de conceptos, fundamentación de objeto de estudio y el campo de acción, análisis de sistemas homólogos, metodología de desarrollo de software, lenguaje y herramientas para el modelado de solución, tecnología de implementación.

Capítulo 2: Análisis y diseño, modelado del negocio, requisitos,

Capítulo 3: Implementación, pruebas y evaluación, estándares de implementación, diagrama de componentes, diagrama de despliegue, interfaz gráfica de usuario, pruebas de software, evaluación de objetivo de la investigación.

CAPÍTULO 1: Fundamentación teórica sobre el desarrollo un módulo de configuración del entorno de escritorio en la distribución cubana de GNU/Linux Nova.

1.1 Introducción

El presente capítulo aborda aspectos sobre la fundamentación teórica de la investigación. Contiene el estudio del proceso de desarrollo de un módulo de configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova. Se define la metodología de desarrollo de software, así como los lenguajes y herramientas empleados en el modelado y desarrollo de un módulo de configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova.

1.2 Definición de conceptos

GNOME es parte del proyecto GNU y parte del movimiento del software libre o código abierto. Es un sistema de escritorio similar a Windows que funciona en sistemas UNIX y similares a UNIX y no depende de ningún administrador de ventanas. (25)

Gnome Control Center es una interfaz gráfica que nos permite configurar diversos elementos de hardware (periféricos, gestión de energía, color, pantalla, sonido, teclado, etc.) y del sistema (notificaciones, compartir, fecha y hora, usuarios, privacidad, idioma…). (26)

Como **interfaz** designamos, en informática, la conexión física y funcional que se establece entre dos aparatos, dispositivos o sistemas que funcionan independientemente uno del otro. En este sentido, la comunicación entre un ser humano y una computadora se realiza por medio de una interfaz. (27)

1.3 Fundamentación del objeto de estudio y campo de acción

En este estudio se abordaron los entornos de escritorio de GNU/Linux, Gnome (en sus distintas versiones). A pesar de la existencia de otros entornos este es el que mayor

cantidad de usuarios acaparan en el mundo por los altos niveles de configuración que brinda, así como la gran cantidad de aplicaciones que tiene para incorporar funcionalidades y riqueza visual. Para cumplir los objetivos de este trabajo este entorno de desarrollo puede servir de base tecnológica, gracias a la flexibilidad de su arquitectura. Teniendo esto en cuenta, el entorno que servirá como base tecnológica para la realización de este trabajo es Gnome porque, a pesar de que KDE es una buena opción para el desarrollo, resulta imposible olvidar el objetivo principal del sistema operativo, la migración, por lo que se debe tener en cuenta que existen usuarios que ya están enmarcados en dicho proceso y que poseen cierto grado de experiencia en el uso de las aplicaciones de Gnome. Un cambio demasiado radical en la interfaz gráfica del sistema operativo dificultaría más la marcha hacia las alternativas libres. En cuanto a la versión que se debe utilizar la **3.4** porque ya no existe ningún proyecto de Gnome que se enfoque en el desarrollo de las versiones anteriores, por lo que en poco tiempo estas versiones estarán completamente obsoletas teniendo en cuenta el ritmo en el que se actualizan las tecnologías en GNU/Linux. Además, Gnome 3 fue aclamado por la comunidad de Nova como el entorno de escritorio para la próxima versión del sistema operativo, complaciendo así a una parte del auditorio en el proceso de migración.

1.4 Análisis de sistemas homólogos

A continuación, se presentan una lista de las opciones de configuración de escritorio perteneciente a distintas herramientas con una funcionalidad similar a la solución plateada.

Windows 11 utiliza el menú “configuración”, específicamente el módulo de “Personalización” para acceder a las opciones para modificar el entorno de escritorio el cual cuenta con:

Opciones	**Descripción**
Fondo	Permite modificar imagen de fondo, color, presentación
Colores	Color de énfasis, efectos de transparencia
Temas	Permite insertar, crear, administrar los temas
Pantalla de bloqueo	Permite modificar las imágenes de pantalla de bloqueo, aplicaciones, animaciones
Teclado táctil	Administrar los temas y el tamaño del teclado táctil
Inicio	Aplicaciones y elementos recientes, carpetas
Barra de tareas	Comportamientos de la barra de tareas, PIN del sistema

Fuentes	Instalar, administrar las fuentes
Uso de dispositivos	Selecciona todas las formas en que planeas usar el dispositivo para obtener sugerencias, anuncios y recomendaciones personalizados en las experiencias de Microsoft.

Tabla 1: Opciones de configuración de escritorio en Windows

La herramienta XFCE comprende una serie de funcionalidades que trabajan juntas para brindar las capacidades completas de un entorno de escritorio moderno. Se empaquetan de forma independiente y puede elegir entre varias opciones para crear su propio entorno de trabajo personalizado.

Del estudio de este sistema homologo se extrajeron las siguientes posibles opciones.

Opciones	**Descripción**
Fondo	permite cambiar la imagen del fondo de escritorio, así como el estilo, el color y cada cuanto minuto cambia la imagen del fondo
Menús	permite elegir si mostrar iconos en las aplicaciones del menú, incluir el menú de las aplicaciones al hacer clic derecho sobre el escritorio. Configurar el área de trabajo y mostrar el menú de la lista de ventanas al hacer clic central en el escritorio
Iconos	permite configurar el tamaño de los iconos y letras en el escritorio, así como elementos ocultos o miniaturas
Ajustes del gestor de ventanas	permite configurar el comportamiento de las ventanas
Estilo	configura el estilo del escritorio
Iconos	configura el estilo de los iconos
Tipo de letra	configura el tipo de letra, el suavizado y el orden de subpíxeles
Configuración	permite configurar el estilo de la barra de herramientas, sonido de eventos y si mostrar iconos en el menú o en los botones.
Área de trabajo	permite configurar los márgenes de la pantalla donde no se coloca ninguna ventana

Editor del menú	permite configurar los elementos que aparecen en el menú principal
Gestor de ventanas	permite configurar el estilo de las ventanas, el enfoque, y el anclaje de las ventanas
Notificaciones	Permite configurar el tema, la posición, opacidad y animaciones de las notificaciones.
Salvapantallas	permite habilitar o deshabilitar el salvapantallas o la pantalla de bloqueo, así como el tiempo antes de su aparición y la imagen o el tema

Tabla 2: Opciones de configuración de escritorio en XFCE

Después de GNOME, el entorno de escritorio Linux más utilizado es **KDE Plasma**. Este entorno de escritorio viene con varias herramientas y servicios para satisfacer casi cualquier demanda. Incluye aplicaciones como Dolphin, el administrador de sistemas de archivos predeterminado y más potente, y KGeoTag, un programa de geoetiquetado de fotografías.

Del estudio de este sistema homologo se extrajeron las siguientes posibles opciones.

Opciones	**Descripción**
Estilo global	permite una rápida configuración general para los que no desean dedicar mucho tiempo a personalizar su escritorio, así como la posibilidad de descargar nuevos temas globales.
Estilo de Plasma	permite modificar la apariencia de la barra de tareas, así como los otros menús
Estilo de las aplicaciones	que permite modificar los iconos y las barras de herramientas, así como el estilo de las aplicaciones de GNOME/GTK y descargar nuevos estilos de aplicaciones de GNOME/GTK 2 y 3.
Decoración de ventanas	que permite modificar el estilo de las ventanas, así como descargar nuevos estilos y cambiar los botones de la barra de título.
Colores	permite configuras los colores de las ventanas, así como descargar nuevos esquemas de colores.
Iconos	modifica los estilos de los iconos, carpetas y documentos.
Emoticonos	determina una serie de patrones de caracteres que al escribirlos los cambia por emoticonos.

Tipos de letra	permite configurar de forma exhaustiva el estilo de las letras en los menús, títulos de ventanas o iconos. Véase el tamaño, anchura, minúscula y mayúscula, suavizado de bordes, representación de subpíxeles, optimización de rende rizado
Gestión de tipos de letra	Agregar o eliminar los tipos de letras.
Cursores	permite modificar el estilo del cursor, el tamaño además de instalar nuevos estilos.
Comportamiento general	permite configurar la información emergente al seleccionar un archivo, aplicación o los cambios de estado. Permite acelerar o frenar la velocidad de las animaciones. Configura el comportamiento del clic y la barra de desplazamiento.
Bordes de la pantalla	configura acciones al arrastrar ventanas al borde de la pantalla.
Animación del clic del ratón	crea una animación cada vez que se pulsa un botón del ratón
Ayudante de desplazamiento	ayuda a localizar el centro de la pantalla cuando se mueve a una ventana.
Invertir	invierte el color del escritorio y de las ventanas.
Seguimiento del ratón	muestra un efecto de localización del cursor del ratón
Ampliación	amplia todo el escritorio
Espejo	una lupa de ventanas que parece una lente de ojo de pez
Lupa	amplia la sección de la pantalla que esta próxima al cursor del ratón
Borde de la pantalla	resalta el borde de la pantalla al aproximarse a él.
Contraste del fondo	mejora el contraste y la legibilidad de las ventanas semitransparentes
Desenfocar	desenfoca el fondo de una ventana semitransparente
Marcar con el ratón	permite dibujar líneas en el escritorio
Miniaturas laterales	muestra miniaturas de las ventanas en el borde de la pantalla
Puntos de contacto	visualizar puntos de contactos
Deslizar hacia atrás	desliza las ventanas hacia atrás cuando una ventana pasa a primer plano
Oscurecer inactivas	oscurece las ventanas inactivas
Cambiar tamaño de las ventanas	cambia el tamaño de las ventanas con un escalado rápido en lugar de actualizar su contenido
Cubo de escritorio	muestra cada escritorio virtual en la cara de un cubo
Presentar ventanas	reducir hasta que todas las ventanas abiertas se muestren una junto a otra
Rejilla de escritorio	reduce hasta que todos los escritorios se muestren en una rejilla
Mostrar FPS	muestra el rendimiento de Kwin en una esquina de la ventana
Mostrar pintado	resalta las áreas del escritorio que se han actualizado recientemente
Pantalla táctil	configura para provocar una acción al deslizar el dedo desde el borde de la pantalla hacia el centro

Bloqueo de pantalla	permite configurar si la pantalla se bloque y en qué tiempo además del aspecto visual de la pantalla de bloqueo, como la imagen de fondo y el reloj
Pantalla de inicio de sesión	permite configurar la pantalla de inicio de sesión
Sesión de escritorio	permite configurar el inicio, cierre y restauración de la sesión del escritorio
Pantalla de bienvenida	administrar la imagen de la pantalla de bienvenida

Tabla 3: Opciones de configuración de escritorio en KDE Plasma

MATE es la extensión de GNOME 2. Crea una experiencia de escritorio intuitiva y atractiva para Linux y otros sistemas operativos similares a Unix mediante el empleo de paradigmas clásicos.

Del estudio de este sistema homologo se extrajeron las siguientes posibles opciones.

Opciones	**Descripción**
Tema	permite configurar el tema del escritorio
Fondo	elige la imagen de fondo del escritorio, así como el estilo
Topografía	permite configurar el estilo, tamaño y fuente de las letras
Interfaz	permite elegir si mostrar iconos en los menús o en los botones
Ventana de inicio de sesión	permite configurar la apariencia de la ventana de inicio de sesión, así como nombre del equipo, opciones de accesibilidad, alimentación con batería, disposición del teclado, reloj y menú de apagado.
Salvapantallas	Permite determinar el tiempo antes que aparezca el salvapantallas y que imágenes aparecerán
Menú Principal	permite elegir qué elementos aparecerán en el menú principal
Notificaciones emergentes	permite elegir si aparecen notificaciones, el tema y la posición
Ventanas	permite configurar el comportamiento de las ventanas

Tabla 4: Opciones de configuración de escritorio en MATE

Resumen

Tras el análisis de sistemas homólogos y la realización de encuestas, se sustrajo la lista de opciones más usadas para la configuración del entorno de escritorio. Pese a que estas herramientas comparten cualidades similares a la propuesta de solución presentada, las versiones actuales no son compatibles con el sistema operativo Linux/Nova.

Opciones	Descripción
Fondo	permite modificar imagen de fondo, color, presentación
Tema	permite configurar el tema del escritorio, iconos y ventanas
Pantalla de bloqueo o Salvapantallas	permite modificar las imágenes de pantalla de bloqueo, aplicaciones, animaciones
Notificaciones emergentes	permite elegir si aparecen notificaciones, el tema y la posición
Tipos de letra	permite el estilo de las letras en los menús, títulos de ventanas o iconos. Véase el tamaño, anchura, minúscula y mayúscula, suavizado de bordes, representación de subpíxeles, optimización de rende rizado
Menú Principal	permite elegir qué elementos aparecerán en el menú principal
Ventanas	permite configurar el comportamiento de las ventanas
Cursores	permite modificar el estilo del cursor, el tamaño además de instalar nuevos estilos.

Tabla 5: Resumen de las opciones de configuración de escritorio

1.5 Metodología de desarrollo de software:

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito, comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado. (13)

En el desarrollo de la propuesta de solución se utiliza la metodología de desarrollo de software Variación de AUP para la UCI en su escenario 4. Esta metodología propone las siguientes fases:

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente

que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura, el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.

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

El desarrollo de la investigación se centrará en la fase de Ejecución y transitará por las siguientes disciplinas: requisitos, análisis, diseño, implementación, pruebas interna y pruebas de aceptación, definidas en la metodología. Porque la fase de inicio se basa en la gestión de proyecto, estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto, en la fase de cierre se analizan tantos los resultados del proyecto como su ejecución y las actividades formales de cierre del proyecto que no se realizan en la presente investigación.

Escenario No 4: Se escogió este escenario porque no es necesario modelar el negocio. El cliente, en este caso el centro de desarrollo CESOL, estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se eligió este escenario porque no es un proyecto muy extenso y que no es necesario empezar de desde cero porque existen algunas funcionalidades ya implementadas que permiten agregar otras, y las HU no poseen demasiada información.

Herramientas para el modelado de la solución

VisualParadim 8.0 es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma 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. (14)

1.6 Tecnologías de implementación:

Entorno de desarrollo:

Visual Studio Code 1.72.2 es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto.

Glade (o Glade Interface Designer, que significa 'Diseñador de interfaces Glade') es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y predeterminadamente no genera código fuente sino un archivo XML (ver sección GtkBuilder). La posibilidad de generar automáticamente código fuente fue descontinuada desde Glade versión 3.

1.7 Lenguaje de programación:

C es un lenguaje de programación de propósito general, originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell,1 como evolución del anterior lenguaje B, a su vez basado en BCPL.2:134. Al igual que B, es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la

eficiencia del código que produce y es el lenguaje de programación más popular para crear softwares de sistemas y aplicaciones. (16)

1.8 Conclusiones parciales

El estudio de los fundamentos teóricos-metodológicos permitió identificar los conceptos y sistemas homólogos referentes a las herramientas de configuración de entornos de escritorio, y esto permitió una visión más amplia de las funcionalidades que se desean para el entorno de escritorio de Nova. La metodología permitió identificar los diagramas necesarios para diseñar la propuesta solución y las herramientas necesarias, así como las técnicas de obtención e identificación de información.

Capítulo 2: Análisis y diseño del módulo de configuración del entorno de escritorio en la distribución cubana de GNU/Linux Nova.

2.1 Introducción

En el presente capítulo se aborda sobre el análisis y diseño del módulo de configuración del entorno de escritorio en la distribución cubana de GNU/Linux Nova. Se identifican sus características a través de los requisitos funcionales y no funcionales y se describen las historias de usuario que especifican cada requisito funcional. Además, se incluye la arquitectura del software, los patrones de diseño y los diferentes artefactos de ingeniería de software correspondiente a las funcionalidades.

2.2 Requisitos

Fuentes de obtención de requisitos

Se determino que los requisitos a utilizar durante esta propuesta de solución procedían de las siguientes fuentes:

- Análisis de sistemas homólogos
- Especialistas de CESOL

Técnicas de identificación de requisitos

Para poder identificar correctamente los requisitos dentro de la información extraída previamente se utilizaron las siguientes técnicas:

- Encuesta (ver anexos)
- Estudio de sistemas homólogos

Especificación de requisitos funcionales

Los requerimientos funcionales de un sistema son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.

Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser desempeñados por el sistema y otros requerimientos de negocio, cumplimiento, seguridad u otra índole. (17)

En la versión actual de Nova las opciones de configuración: Fondo, Notificaciones, Menú Principal, Ventanas y Cursores ya están programadas.

No.	Nombre	Descripción	Complejidad
RF_1	Agregar apariencia	Se debe crear una ventana apariencia para agrupar las opciones de configuración necesarias.	alta
RF_3	Agregar tema cursor	Se debe agregar una lista desplegable que permita elegir el tema del cursor	
RF_4	Agregar tema icono	Se debe agregar una lista desplegable que permita elegir el tema de los iconos del escritorio	alta
RF_5	Agregar pantalla de bloqueo	Se deba crear un menú, de configuración para poder seleccionar una imagen de salvapantallas.	baja
RF_6	Agregar ajuste para la pantalla de bloqueo	Dentro del menú para elegir la imagen de la pantalla de bloqueo se debe agregar	alta

		una opción para ajustar la proporción de dicha imagen	
RF_7	Agregar tipo de letra	Se debe crear una lista desplegable para cambiar el estilo de las letras en el escritorio	baja
RF_8	Agregar escalado de letra	Se debe agregar un factor de escalado que permita agrandar o disminuir el tamaño de letra	alta
RF_9	Agregar texto de interfaz	Se debe agregar al menú de estilo de letra una opción para elegir la apariencia del texto ASCII	alta
RF_10	Agregar texto del documento	Se debe agregar al menú de estilo de letra una opción para elegir la apariencia del texto en el escritorio y las ventanas.	alta
RF_11	Agregar texto monoespaciado	Se debe agregar al menú de estilo de letra una opción para elegir la apariencia de textos monoespaciados (es decir, donde todos los caracteres ocupan el mismo espacio)	alta

Tabla 6: Requisitos funcionales

Especificación de requisitos no funcionales

Los requerimientos no funcionales engloban características como rendimiento, facilidad de uso, presupuestos, tiempo de entrega, documentación, seguridad y auditorías internas, describen prestaciones, características y limitaciones que debe tener el sistema para alcanzar el éxito. (18)

Requisito de Interoperabilidad/Compatibilidad de software

RnF_1: Se utiliza el lenguaje de programación C.

Requisito de Eficiencia/Utilización de recursos

RnF_2: La aplicación debe utilizar el mínimo de recursos del sistema para garantizar la eficiencia.

Requisito Seguridad

RnF_3: Se debe publicar las actualizaciones de la aplicación en el repositorio, para que pueda ser descargada como actualización del sistema.

Requisito de Confiabilidad/Tolerancia a fallos

RnF_4: En caso de un error la aplicación debe mostrar una notificación con la información de este.

Prototipo de interfaz

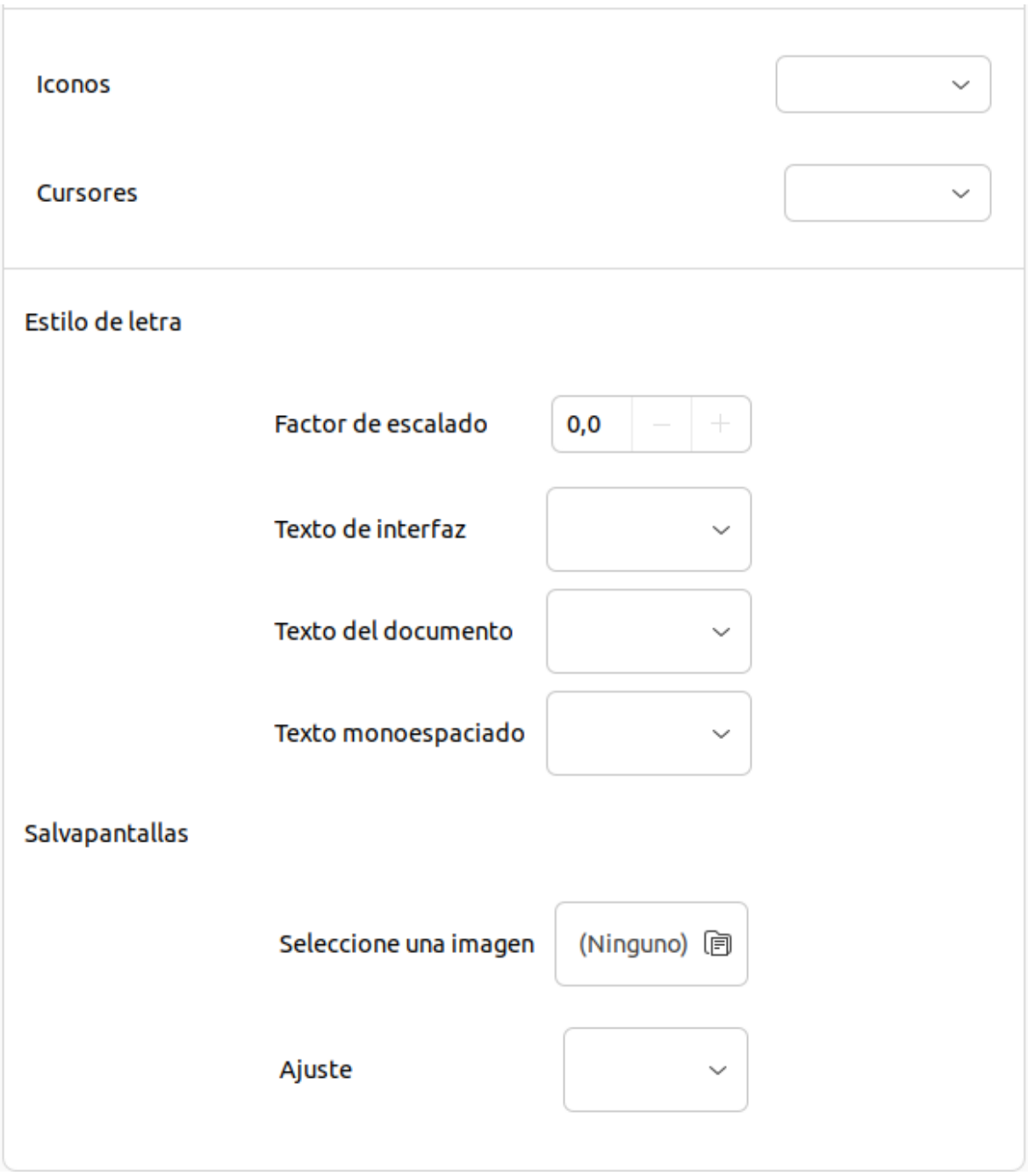


Imagen 1: prototipo de interfaz

2.3 Historias de usuarios (HU)

Historia de usuario	
Número: HU_1	**Nombre: Agregar apariencia**
Prioridad: Alta	**Iteración Asignada: 1**
Programador: Erian Veliz Beltran	**Tiempo Estimado: 10 horas**
Riesgo en Desarrollo: N/A	**Tiempo Real: 5 horas**
Descripción: Se debe crear una ventana apariencia para agrupar las opciones de configuración necesarias	

Tabla 7: Historias de usuarios de Agregar apariencia

Análisis y diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. Los objetivos del análisis y diseño son: transformar los requisitos al diseño del futuro sistema, desarrollar una arquitectura para el sistema y adaptar el diseño para que sea consistente con el entorno de implementación.

En este método de análisis y diseño se crea un conjunto de modelos utilizando una notación acordada como, por ejemplo, el Lenguaje Unificado de Modelado (UML). Aplica técnicas de modelado de objetos para diseñar soluciones que mejoren los procesos involucrados y para analizar los requisitos en diferentes contextos, por ejemplo, un sistema de negocio, un conjunto de módulos de software, entre otros.

Patrón Arquitectónico

Los patrones arquitectónicos son plantillas que describen los principios estructurales globales que construyen las distintas Arquitecturas de Software viables. Plantean una

organización estructural fundamental para un sistema de software, expresando un conjunto de subsistemas predefinidos, especificando responsabilidades y organizando las relaciones entre ellos. La selección de un patrón arquitectónico es además una decisión fundamental de diseño cuando se desarrolla un sistema de software. (19)

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

En este software fue desarrollado utilizando una arquitectura n-capas, en su caso se definieron 3 capas: Presentación, Controladora y Acceso a datos.

Presentación: En esta capa se presentan las opciones de configuración del escritorio y su estado actual a través de las interfaces de usuario o UI (User Interface). Permite interactuar con la aplicación. Se comunica con la capa Controladora. Las clases pertenecientes a esta capa son GtkComboBoxText, GtkFileChooserButton, GtkSpinButton, Apariencia_panel, CC_Panel.

Controladora: En esta capa se reciben los cambios hechos por el usuario en el panel de control y los aplica de forma inmediata. Envía la información a la capa de Acceso a datos y funciona como mediadora entre esta y la capa de Presentación. La clase ubicada en esta capa es ubuntu_panel.

Acceso a datos: En esta capa se guarda el estado del entorno de escritorio, el cual se sobrescribe al recibir nueva información de la capa superior. Es la encargada de proporcionarle la información del estado actual del entorno de escritorio cada vez que la capa controladora lo solicite. La clase ubicada en esta capa es ubuntu_gresource.

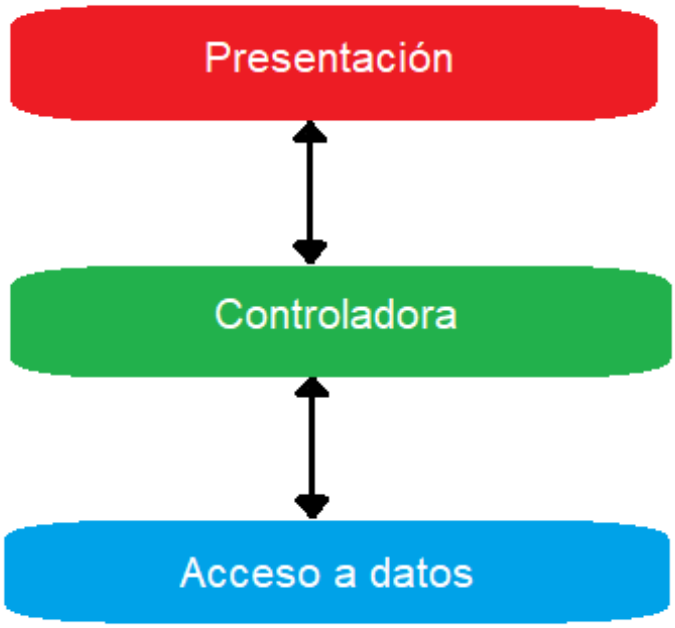


Imagen 2: Arquitectura de 3 capas

2.4 Diagrama de clases

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

Suelen utilizarse para explorar los conceptos de dominio, comprender los requisitos de los programas informáticos y describir diseños detallados. (20)

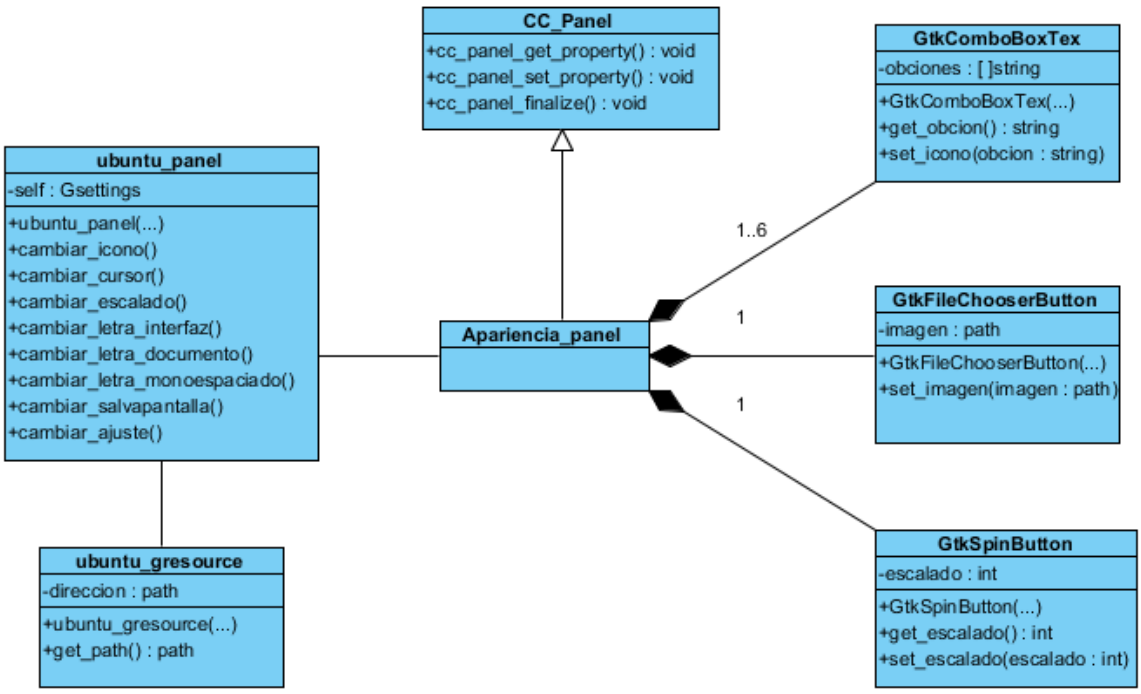


Imagen 3: Diagrama de clases

2.5 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Contribuyen a reutilizar diseño gráfico identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones.

General Responsibility Assignment Patterns (GRASP), son una serie de patrones que describen los principios fundamentales de la asignación de responsabilidades a objetos y son considerados una serie de buenas prácticas en el diseño de software.

Para la implementación de la propuesta de solución fueron aplicados los siguientes patrones de diseño:

Creador

El patrón Creador asigna a una clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes:

B agrega objetos de A.

B contiene objetos de A.

B registra instancias de objetos de A.

B utiliza más estrechamente objetos de A.

B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A). B es un creador de los objetos A. (28)

La clase Apariencia_panel contiene los datos de inicialización de la clase ubuntu_panel por lo que la clase Apariencia_panel es una clase creadora de ubuntu_panel. A demás la clase Apariencia_panel utiliza estrechamente objetos de las clases GtkComboBoxText, GtkSpinButton, GtkFileChooserButton, por lo que la clase Apariencia_panel es una clase creadora con respecto a estas otras tres clases.

Experto en información

El patrón Experto en información asigna a una clase la responsabilidad de experto en información, es decir, la clase que tiene la información necesaria para realizar las demás responsabilidades. (28)

Como se planteó anteriormente, la clase Apariencia_panel contiene los datos de inicialización de ubuntu_panel, por lo que Apariencia_panel es una clase Experto en información con respecto a la clase ubuntu_panel.

Alta cohesión

El patrón Alta cohesión asigna una responsabilidad de manera que la cohesión permanezca alta. En cuanto al diseño de objetos, la cohesión (o de manera más específica, la cohesión funcional) es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. Estos elementos pueden ser clases, subsistemas, etcétera. (28)

En la propuesta solución las clases GtkComboBoxText, GtkSpinButton y

GtkFileChooserButton, cumplen una sola función y es la de almacenar la información con la que el usuario interactúa en la interfaz.

Bajo acoplamiento

El patrón Bajo acoplamiento asigna una responsabilidad de manera que el acoplamiento permanezca bajo**.** El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de demasiados otros elementos; "demasiados" depende del contexto. Estos elementos pueden ser clases, subsistemas, sistemas, etcétera.
(28)
En la propuesta solución las clases GtkComboBoxText, GtkSpinButton y GtkFileChooserButton interactúan solamente con la clase Apariencia_panel, manteniendo el acoplamiento bajo.

2.6 Conclusiones parciales

La identificación adecuada de los requisitos permito definir los componentes necesarios para implementar un módulo que permita mejorar la configuración del entorno de escritorio.

La modelación conceptual a través de los diagramas de clases y patrones de diseño permitió implementar correctamente la propuesta solución con el mínimo de errores en el menor tiempo posible.

Capítulo 3: Implementación y pruebas del módulo para la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova.

3.1 Introducción

En este capítulo se muestran los diferentes artefactos que se utilizan para la implementación y pruebas del sistema, así como los estándares de codificación que debe seguir el equipo de desarrollo para una mejor entendimiento y organización del código, y de esta manera otorgarle validez tanto a los requisitos funcionales como no funcionales.

3.2 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño y cómo las clases, se implementan en términos de componentes, como fichero de código fuente y ejecutables. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (29).

3.3 Diagrama de componentes

Los diagramas de componentes son utilizados para estructurar el modelo de la implementación. Permiten modelar una vista estática del sistema, muestran la organización y las dependencias lógicas entre un conjunto de componentes del software, que pueden ser librerías, binarios, ejecutables y códigos fuentes.

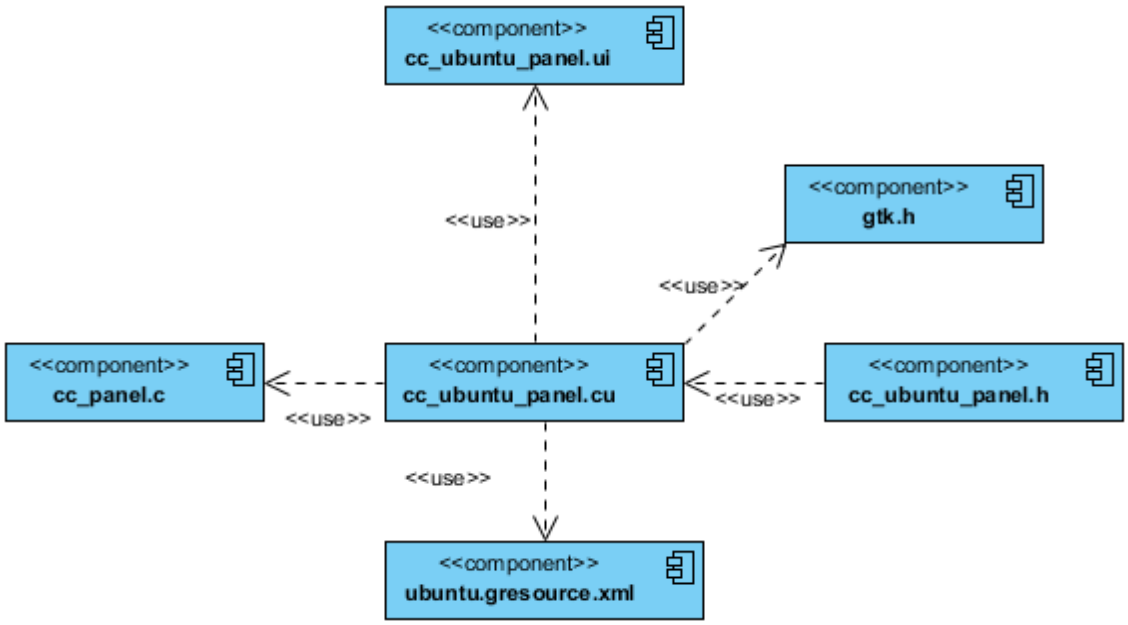


Imagen 4: Diagrama de componentes

3.4 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código el cual refleja un estilo armonioso, como si un único programador hubiese escrito todo el código de una sola vez. Cuando el proyecto de software incorpora código fuente previo, o cuando realiza el mantenimiento de un sistema de software el estándar de codificación debería establecer cómo operar con la base de código existente. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Para la propuesta de solución se utilizó los estándares de codificación para C basado en Pautas de Codificación C Departamento de electrónica, Sistemas e Informática.

Nombres de identificadores

- Deberán tener un nombre significativo para que, por su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que éstos tengan una longitud razonable. Si usa abreviaturas deben manejar la misma lógica en todo el programa.
- Evitar identificadores que comiencen con uno o dos caracteres de subrayado para evitar que se confundan con los que el compilador selecciona.
- Cada identificador de función, variable o procedimiento deberá ser precedido por la abreviación del tipo de dato de que es la variable, o si se trata de una función o procedimiento del tipo de dato que regresa.

Identificadores de variables

Comenzarán siempre con la primera letra minúscula correspondiente a su tipo de dato.

Para distinguir palabras dentro del nombre deberá emplearse una letra mayúscula o un guion bajo (_), sin mezclar ambas formas en un mismo programa.

Identificadores de punteros (apuntadores)

Su nombre deberá comenzar con la letra p.

Identificadores de variables dimensionadas (arreglos, matrices)

Su nombre deberá comenzar con las letras ar.

Identificadores de datos constantes

Serán declaradas en letras mayúsculas.

Identificadores de funciones

La primera letra deberá ser mayúscula.

Identificadores de tipos definidos por el usuario

La primera letra será mayúscula.

Organización Visual del Programa

Generales

- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.

Sangrías

- Las sangrías tendrán una longitud de tres espacios.
- Para las llaves que definen el cuerpo de una función, sangre un nivel.
- Sangre las instrucciones del cuerpo de cada estructura de control.
- Trate de evitar codificar más de tres niveles de sangrado.

Líneas y espacios en blanco

- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.
- Las declaraciones de datos dentro de una función deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.
- Deben incluirse espacios en ambos lados de los operadores binarios
- Es posible distribuir una instrucción grande sobre varias líneas. Si lo hace, seleccione puntos de ruptura que tengan sentido, como después de una coma en el caso de una lista, o después de un operador en el caso de una expresión larga.
- Sangre todas las líneas subsecuentes.
- Los operadores unarios (++, --, etc.) deben ponerse junto a sus operandos, sin espacios intermedios.
- Antes y después de cada estructura de control deberá poner una línea en blanco.

Paréntesis

- Para hacer más clara una expresión, es aceptable agregarle paréntesis innecesarios. Dichos paréntesis se llaman paréntesis redundantes.

3.5 Pruebas Funcionales

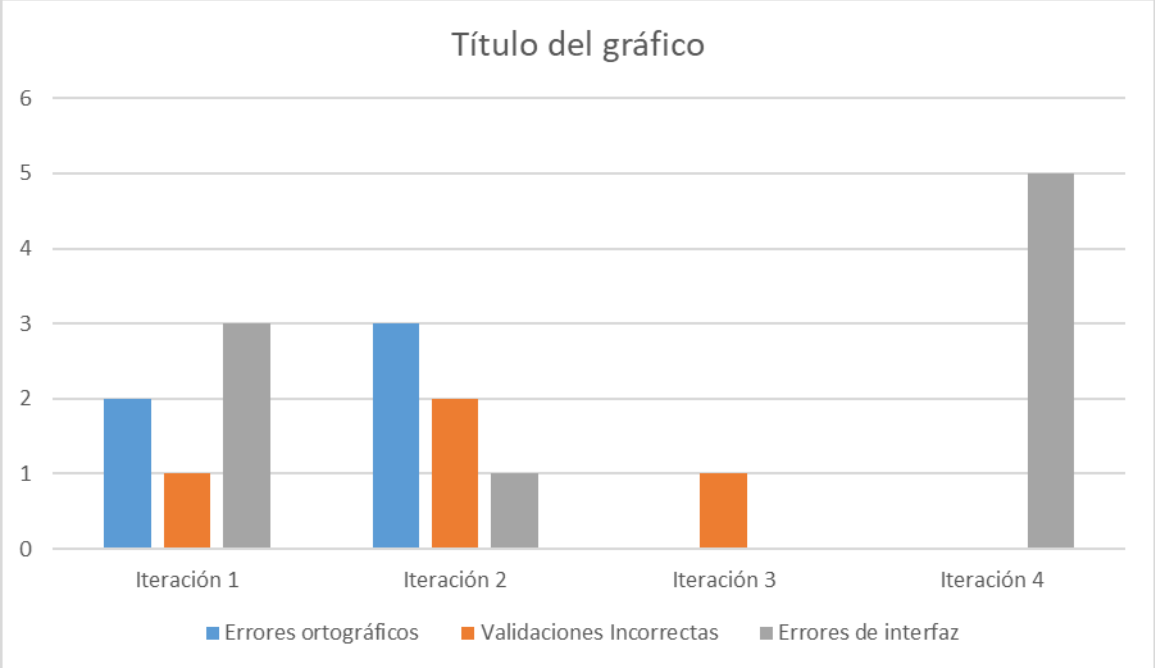
Las pruebas funcionales son aquellas que se llevan a cabo sobre la interfaz del software sin prestar atención al código, por lo que los casos de prueba son creados con el objetivo de demostrar que la entrada es aceptada de forma adecuada y que se produce una salida correcta. El diseño de esta prueba se realiza con la intención de detectar funciones incorrectas o ausentes, errores en accesos a bases de datos externas, errores de interfaz, errores de rendimiento, y errores de inicialización y de terminación. Dentro de la prueba se incluyen la técnica de partición de equivalencia que será la empleada en la validación. La partición de equivalencia es una técnica de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba.[23]

Tabla 8: Partición de equivalencia

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Respuesta del sistema	Flujo central
EC 1.1 Cambiar el tema de los íconos	El usuario elige un tema de entre los elementos de la lista desplegable	V Yaru-White	V Yaru-Dark	V Adwaita	V DMZ-Black	El sistema inmediatamente cambia el tema de los iconos	1.El usuario abre el módulo de apariencia. 2.El usuario despliega la lista de opciones. 3.El usuario selecciona una entre las opciones desplegadas

EC 1.2 Cambiar el tema de los iconos. El usuario no elige ninguno	El usuario despliega la lista de opciones, pero no realiza ningún cambio. Cliquea otra opción y la lista desplegable se cierra.	I Vacío	I Vacío	I Vacío	I Vacío	El sistema no realiza ningún cambio y cierra la lista desplegable.	1.El usuario abre el módulo de apariencia. 2.El usuario despliega la lista de opciones. 3.El usuario no selecciona ningún elemento de la lista

Durante el proceso de prueba se detectaron trece (13) no conformidades, de ellas cinco (5) corresponden a errores ortográficos y de idioma, cuatro (4) estaban relacionadas con las validaciones incorrectas y otras cuatro (4) no conformidades correspondientes a errores relacionados con la interfaz



3.6 Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia, el desempeño de un software involucra las operaciones del sistema bajo condiciones controladas y evalúa los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad. A continuación, se describen los tipos de pruebas de software aplicadas, así como los métodos y técnicas empleadas para la evaluación de la propuesta de solución.[24]

Prueba de caja blanca

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente (Ojeda, 2014).

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, 2005).

Para aplicar la prueba de camino básico, realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del sistema; una vez concluido este paso se selecciona el método con valor de complejidad más elevado. 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 R., 2010). En la imagen 5 se presenta el método rellenar_combo_box () correspondiente al RF4 Agregar tema icono.

1- **Dibujar el grafo de flujo de la funcionalidad**

En la imagen 6 se utilizó la notación de grafo de flujo para representar el código de la funcionalidad a probar

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) = 11-8+2 nodos del grafo
V(G) = 5

V(G) = P+1 Donde P es el número de nodos predicados (nodos con más de una
V(G) = 4+1 arista de salida) contenidos en el grafo
V(G) = 5

V(G) = R Donde R es el número de regiones (áreas delimitadas por nodos y aristas
V(G) = 5 en el grafo)

3- Determinar los caminos linealmente dependientes

Camino básico 1: 1,2,3,4,5,6,7,8

Camino básico 2: 1,2,3,8

Camino básico 3: 1,2,3,4,8

4- Definir los casos de prueba para comprobar la ejecución

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

- ✓ Descripción: contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.
- ✓ Condición de ejecución: se especifican los parámetros que deben 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 1 del conjunto de caminos básicos de linealmente independientes:

Diseño de caso de prueba para el camino 1	
Descripción	Método para buscar temas de iconos en los archivos y mostrarlos en una lista desplegable en la interfaz.
Condiciones	El usuario inicializa la Gnome Control Center.
Entradas	object: {CcUbuntuPanel *self}
Resultados	El usuario puede ver una lista desplegable con todos los temas de iconos guardados en los archivos.

Tabla 9: Caso de prueba

Luego de realizadas las pruebas se obtuvo como resultado que el flujo de trabajo de las funcionalidades del módulo para la configuración del entorno de escritorio de Gnome Control Center es correcto; pues se comprobó que cada sentencia del código fuente se ejecuta al menos una vez.

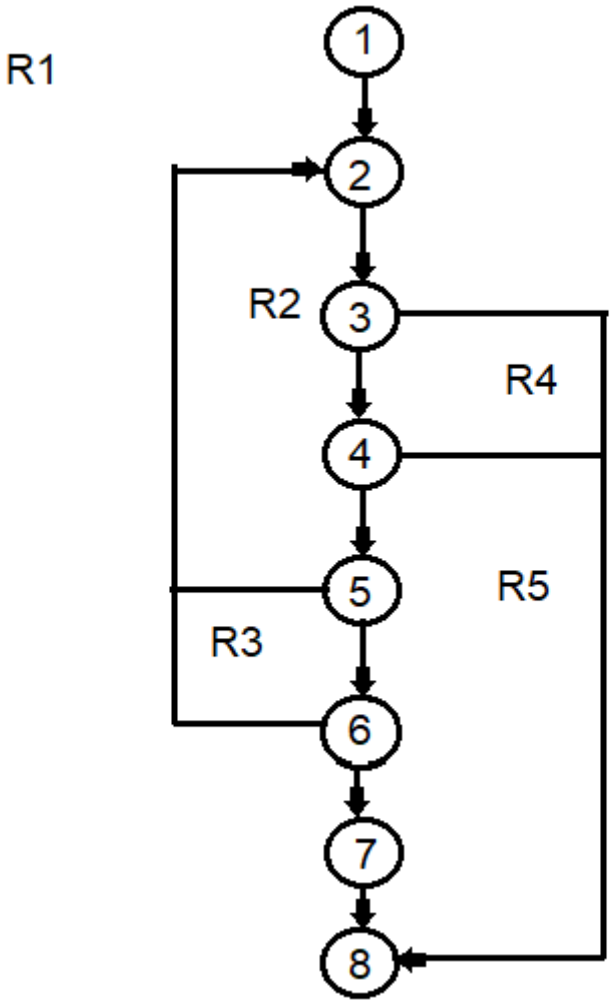


Imagen 6: Grafo de flujo

3.6 Conclusiones parciales del capítulo

Los estándares de codificación permitieron desarrollar un código reutilizable y de fácil comprensión en el desarrollo de la propuesta de solución. Una vez diseñadas, documentadas y ejecutadas las pruebas de software, se detectaron las no conformidades, lo que permitió verificar la calidad del producto.

Conclusiones generales

Como resultado de la presente investigación se obtuvo un módulo que permite mejorar la configuración del entorno de escritorio. En respuesta a las preguntas científicas plateadas se concluye:

¿Cuáles son los principales fundamentos teóricos-metodológicos que sustentan el proceso de configuración de entorno de escritorio de Gnome Shell?

- ✓ El estudio de los fundamentos teóricos-metodológicos permitió identificar los conceptos y sistemas homólogos referentes a las herramientas de configuración de entornos de escritorio, y esto permitió una visión más amplia de las funcionalidades que se desean para el entorno de escritorio de Nova.

¿Qué elementos se deben tener en cuenta para diseñar un módulo que permita mejorar la configuración del escritorio de Gnome Shell desde el Centro de Control de Nova?

- ✓ La metodología permitió identificar los diagramas necesarios para diseñar la propuesta solución y las herramientas necesarias, así como las técnicas de obtención e identificación de información.

¿Qué componentes son necesarios implementar para obtener un módulo que permita mejorar la configuración del entorno de Gnome Shell desde el Centro de Control de Nova?

- ✓ La identificación adecuada de los requisitos permito definir los componentes necesarios para implementar un módulo que permita mejorar la configuración del entorno de escritorio.
- ✓ La modelación conceptual a través de los diagramas de clases y patrones de diseño permitió implementar correctamente la propuesta solución con el mínimo de errores en el menor tiempo posible.

¿Qué técnicas y pruebas de software se deben aplicar para evaluar el módulo para la configuración del entorno de escritorio de Gnome Shell desde el Centro de Control de Nova?

- ✓ Los estándares de codificación permitieron desarrollar un código reutilizable y de fácil comprensión en el desarrollo de la propuesta de solución. Una vez diseñadas, documentadas y ejecutadas las pruebas de software y funcionales, se detectaron las no conformidades, lo que permitió verificar la calidad del producto.

Anexos

Preguntas de la entrevista:

¿Cuáles son las herramientas de configuración del entorno de escritorio que se usan en Nova?

¿Qué lenguaje de programación utiliza el centro de control Nova?

Preguntas de la encuesta:

Herramientas	Descripción	Frecuencia de uso 5(mucho)-1(poco\nunca)
Fondo	permite modificar imagen de fondo, color, presentación	5
Tema	permite configurar el tema del escritorio, iconos y ventanas	5
Pantalla de bloqueo o Salvapantallas	permite modificar las imágenes de pantalla de bloqueo	4
Notificaciones emergentes	permite elegir si aparecen notificaciones, el tema y la posición	1
Tipos de letra	permite el estilo de las letras en los menús, títulos de ventanas o iconos. Véase el tamaño, anchura, minúscula y mayúscula, suavizado de bordes, representación de subpíxeles, optimización de rende rizado	1
Menú Principal	permite elegir qué aplicaciones aparecerán en el menú principal	5
Ventanas	permite configurar el comportamiento de las ventanas	2
Cursores	permite modificar el estilo del cursor, el tamaño además de instalar nuevos estilos.	3

Guía de observación:

¿Cuáles herramientas de configuración son de nivel avanzado y puedan provocar fallos en la estabilidad del entorno de escritorio de Nova?

¿Qué herramientas son indispensables a la hora de desarrollar un módulo de configuración del escritorio?

Bibliografía

1. https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-sistema-operativo
2. https://www.gnu.org/home.es.html
3. https://www.researchgate.net/publication/317017785_Proceso_de_desarrollo_de_la_distribucion_cubana_de_GNULinux_Nova_Development_process_of_the_Cuban_distribution_of_GNU_Linux_Nova
4. https://innovacioneducativa.upm.es/competencias-genericas/formacion-evaluacion/analisis-sintesis
5. https://web.archive.org/web/20070818123652/http://primates.ximian.com/~miguel/gnome-history.html
6. https://es.wikipedia.org/wiki/Paquete_de_software
7. https://www.compuhoy.com/como-personalizo-mi-escritorio-en-linux
8. https://wiki.gnome.org/ThreePointZero/Plan
9. https://www.grulic.org.ar/node/10
10. https://web.archive.org/web/20010222011552/http://www.csdl.tamu.edu/~l0f0954/academic/cpsc610/hw2-3.htm
11. https://www.nova.cu/
12. https://es-academic.com/dic.nsf/eswiki/817057
13. http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software
14. https://geekflare.com/es/linux-desktop-environment
15. https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)
16. https://www.ecured.cu/Visual_Paradigm
17. http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html
18. https://ingenieriadesoftwareutmachala.wordpress.com/2017/01/20/requerimientos-funcionales-y-no-funcionales/
19. Trabajo de diploma “Arquitectura de referencia para el desarrollo de aplicaciones web del centro FORTES (XALIX 2.0)” por Hermes Rodríguez Quesada
20. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-clases/
21. 2022-2-3 https://jnjsite.com/los-patrones-del-diseno-software-grasp

22. https://www.adictosaltrabajo.com/2019/07/23/el-polimorfismo-es-la-clave
23. https://trycore.co/transformacion-digital/tipos-de-pruebas-funcionales
24. https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos
25. https://lamiradadelreplicante.com/2016/05/29/asi-sera-el-nuevo-gnome-control-center
26. https://www.compuhoy.com/que-es-gnome-linux-sistema-operativo-hoy
27. https://www.significados.com/interfaz
28. UML y Patrones 2da edición
29. LARMAN, 1999