Manual de Usuario

Versión 1.0

Autor: Grupo Meridian
Contacto: grupo.meridian@example.com

21 de septiembre de 2025

Prefacio

El **Proyecto MERIDA** nació de una necesidad personal: la búsqueda de un entorno de escritura que me hiciera sentir realmente cómodo. La ausencia de ciertos pequeños detalles en otros procesadores de texto fue lo que impulsó su desarrollo.

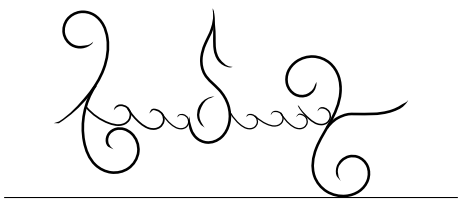
En sus primeros pasos, **MERIDA** fue concebido como un simple editor de texto que organizaba sus archivos mediante etiquetas. Con el tiempo, esta idea inicial fue creciendo, incorporando nuevas funciones, y seguirá evolucionando con cada versión.

MERIDA no pretende reemplazar nada, ni competir con nadie. Solo quiere ayudar, acompañarte en el proceso creativo y llenar esos huecos que, como escritor o creador, quizá también has sentido.

Publicar esta primera versión me llena de orgullo. Este es mi primer programa "real", y verlo funcionar me hace sentir que todo el esfuerzo valió la pena. Gracias por darle una oportunidad, por usarlo, por explorarlo... y tal vez, por mejorarlo. Cada sugerencia, cada corrección y cada línea de código que alguien más aporte hará que **MERIDA** crezca.

Agradezco sinceramente a quienes descarguen, prueben y exploren **MERIDA**. También ofrezco disculpas anticipadas por los posibles fallos que puedan encontrar y agradezco de corazón a quienes decidan colaborar con el proyecto. **MERIDA** es y será siempre un software libre y de código abierto.

Sea cual sea el futuro de este proyecto, sé que lo acompañaré mientras exista. Espero que tú también encuentres en **MERIDA** algo que te inspire a crear.



DGonzak

Índice general

Parte I

Proyecto MERIDA

Capítulo 1

Introducción

1.1. Propósito del proyecto

El **Proyecto MERIDA** fue creado para agrupar una serie de componentes digitales cuyo propósito principal es ayudar a la producción escrita en general (teniendo un enfoque más acercado a la escritura literaria). Estos componentes, también llamados módulos o, incluso, «programas» están construidos bajo un solo objetivo, ayudar con diferentes herramientas al proceso de escritura. Es así que el proyecto MERIDA surge como una agrupación de software destinado a la redacción.

1.2. Acerca de esta guía

La presente guía de usuario tiene como objetivo la explicación, cabal y detallada, sobre los distintos componentes del **proyecto MERIDA**. En esta guía se podrá encontrar explicaciones de uso e incluso, si llega a ser necesario, explicaciones sobre el código mismo del programa.

Como el proyecto MERIDA se construye mediante actualizaciones que añaden nuevos componentes cada cierto tiempo, la guía también se presentará mediante diferentes versiones. Cada versión corresponderá a la versión estable que se presente con el programa principal del proyecto MERIDA, y contendrá, por lo tanto, información sobre los componentes antiguos y nuevos respecto a la versión del programa. Es así que la guía se estructura mediante capítulos y secciones. Cada componente principal del proyecto obtiene un capítulo separado y propio, donde se explican a detalles todo su funcionamiento e integración con el proyecto en general.

Por otro lado, al final de la guía, se encontrará la parte de los **anexos**. En ella se podrá encontrar elementos adicionales al contenido principal de la guía que sirvan como información extra a la misma. Se podrán encontrar, por ejemplo, la sección de **preguntas frecuentes** o **solución a problemas comunes.** Estos elementos tienen carácter opcional, y no son estrictamente elementales para el proyecto. Aún así, se recomienda ampliamente su lectura, a fin de obtener el máximo conocimiento respecto al proyecto.

1.3. Público objetivo y requisitos mínimos

El público objetivo del proyecto MERIDA es, en verdad, variado. Al ser una agrupación de diferentes herramientas centradas en la redacción, tanto literaria como académica, es de esperarse que el público objetivo sea de estos ámbitos, es decir, escritores de novelas o de artículos. Sin embargo, la utilidad que el proyecto puede ofrecer no se limita a estos dos centros culturales. Cualquier persona puede usar el programa a fin de servirse de sus utilidades.

Ahora bien, en cuanto a requisitos del sistema para poder ejecutar el programa MERIDA, son realmente pocos. Para su ejecución, al menos en la versión en que es presentada esta guía, el proyecto MERIDA hace uso de los siguientes componentes:

- Python 3.11
- GTK4

Por supuesto, además se hace uso de otras librerías, propias de python, presentadas a continuación[1]:

```
greenlet==3.2.1
Markdown==3.8
platformdirs==4.3.8
pycairo==1.28.0
PyGObject==3.52.3
python-dateutil==2.9.0.post0
pytz==2025.2
six==1.17.0
SQLAlchemy==2.0.40
typing_extensions==4.13.2
```

Ahora bien, la primera versión de MERIDA (V1) ofrece la posibilidad de instalación mediante paquete únicamente en entornos GNU-Linux, más propiamente, en entornos Debian Linux[2]. Para la instalación en dicho sistema se tiene el paquete .deb ubicado en la página de github del proyecto. Una vez descargado, se debe correr el siguiente comando en terminal:

sudo dpkg -i merida_1.0-1_all.deb

Si se tiene problemas en la instalación debido a la falta de **dependencias**, siempre puede usar el siguiente comando justo después de haber intentado instalar el paquete .deb:

sudo apt-get -f install

[1]Las siguientes librerías son extraídas desde el archivo *requirements.txt* ubicado en el código fuente. Por ende tenga en cuenta que pueden variar en versión y número en futuras versiones. Se recomienda ampliamente revisar el archivo del código fuente siempre.

[2]El entorno de desarrollo para la primera versión de MERIDA fue, de hecho, Debian 13 Trixie

Con todo ello, debería ser capaz de instalar el paquete *.deb* en sus sistema operativo. Cabe aclarar que cualquier otro método para instalar paquetes *.deb* es igualmente válido. Se intentará incluir más métodos de instalación en futuras versiones.

Ahora bien, en cuanto a los sistemas operativos **Windows** aún no se tiene soporte oficial de paquetes, es decir, no hay un paquete *.exe* oficial para instalar MERIDA en Windows. Se tiene planeado, sin embargo, publicar dicho paquete en futuras versiones si es posible.

Ahora bien, si se desea, puede descargarse el código fuente del programa directamente desde la página de github. Si desea ejecutar MERIDA desde el código fuente tenga en cuenta que debe contar con todas las dependencias mencionadas anteriormente y hacerlo en un entorno virtual. Por supuesto, el entorno virtual debe tener acceso a GTK4 para que MERIDA pueda funcionar correctamente. Además, tenga en cuenta que el programa usará las carpetas del sistema (.config, .local/share, .cache y .local/state) para guardar configuraciones, archivos temporales, archivos recurrentes y logs en tales ubicaciones.

Capítulo 2

Estructura General del Proyecto MERIDA

2.1. Módulos actuales y Futuras integraciones

La presente versión del proyecto MERIDA (V.1) presenta como primer modulo el **Editor Merida**, componente importante para el proyecto. El editor presenta la característica más relevante del proyecto, la escritura. Se presente como un modulo simple que permite la creación de documentos.

Este primer módulo representa el primer paso del proyecto MERIDA, consistente en otorgar la oportunidad de escritura oportuna para el público objetivo. Sin embargo, no será el único. El módulo siguiente a salir, según los planes previstos por el Grupo Meridian, será la **linea de tiempo Merida**, que ayudará con la creación de lineas de tiempo útiles para investigación y/o planeación de contenido literario. Así mismo se planea integrar el **sistema de instalación de módulos** para MERIDA. Este es un sistema que permitirá la contribución directa de los usuarios, ya que podrán crear sus propios módulos e instalarlos en el programa. También se añadirán mejoras al editor Merida a fin de ofrecer más variedad de herramientas a los usuarios.

Se espera poder publicar la segunda versión de MERIDA a finales de 2026.

2.2. Flujo de Trabajo e Integración de Componentes

2.2.1. Consideraciones Generales

El proyecto MERIDA adopta una visión modular, lo que permite su apertura a colaboradores que deseen contribuir al mismo. Cada módulo se diseña considerando el contexto global del programa, para posteriormente integrarse por completo. Esta integración es fundamental, ya que asegura que el módulo pueda interactuar con el resto de los componentes y herramientas de MERIDA.

En cuanto a sus versiones, el proyecto se divide en dos grandes categorías:

- **Versiones principales**: incluyen la totalidad del programa MERIDA (su base funcional).
- **Versiones de componentes**: corresponden a cada módulo de forma independiente.

El ciclo de vida de un módulo comienza con su desarrollo inicial. Una vez finalizada su primera versión, entra en la **fase de implementación**, en la cual debe integrarse completamente con las demás herramientas del programa.

Al concluir la implementación, se libera una nueva versión principal que ya incluye el módulo. En caso de futuras actualizaciones, el módulo debe pasar nuevamente por la fase de implementación para garantizar que sus nuevas características se integren correctamente al ecosistema del proyecto. Esta metodología asegura un control preciso sobre la evolución del programa y de sus componentes.

El proyecto MERIDA también busca fomentar la **oportunidad de desarrollo**, brindando a cualquier interesado la posibilidad de contribuir en un entorno organizado. Para ello, es esencial seguir el flujo de trabajo definido, lo que asegura versiones claras y consistentes del programa y sus módulos.

Finalmente, cada módulo debe contar con su propia guía de usuario antes de integrarse en la versión principal. Dicha guía debe seguir la estructura básica descrita en este documento (Parte II). La autoría recae en los propios desarrolladores del módulo, quienes conocen mejor su funcionamiento. Sin embargo, se realizará una revisión de redacción para garantizar calidad y coherencia antes de su inclusión en la guía oficial. Documentos con errores ortográficos, gramaticales o de coherencia no serán aceptados para su implementación en la guía oficial de MERIDA hasta su correspondientes corrección[1].

2.2.2. Flujo de Trabajo del Programa

El flujo de trabajo describe la secuencia de eventos que tienen lugar desde el inicio de MERIDA hasta la carga completa de sus módulos y herramientas. Esto asegura que el programa esté listo para su uso y que todos los componentes estén correctamente inicializados.

Pantalla de comprobación inicial (Splash)

La primera etapa del flujo de trabajo es la **pantalla de comprobación inicial**[2]. Esta es una pequeña ventana que aparece cada vez que el programa se inicia.

La función principal de la pantalla de comprobación es, valga la redundancia, comprobar elementos críticos para el buen arranque y funcionamiento del programa MERIDA. Estos elementos pueden dividirse en dos:

[1] Se aclara que ninguna persona puede modificar contenido propio de un módulo sin consentimiento del autor o autores involucrados. Se debe limitar, así, a realizar corrección gramatical, ortográfica y demás elementos relacionados estrictamente a la forma y no al contenido.

[2] En el código como tal, dentro del archivo main.py, la pantalla de comprobación inicial recibe el nombre de «Splash»

- **Carpetas:** De las cuales el programa hace uso para guardar datos importantes (configuraciones, archivos temporales, resguardos para las bases de datos, etc). Las carpetas pueden dividirse, a su vez, en *carpetas principales* y *carpetas secundarias*.
- **Recursos:** De los cuales el programa necesita para poder funcionar correctamente. Por ahora, esta «sección» se refiere a la verificación de existencia de las bases de datos propias del sistema.

Ahora bien, cada «mensaje» que la pantalla de comprobación muestre es guardado en un *log de inicio*. Si se desea, se puede revisar esto log en la siguiente ruta (solo para sistemas Debian Linux):

/.local/state/MERIDA/logs/startup

Es decir, de forma predeterminada, el log se guarda en la carpeta de logs, de MERIDA, en la subcarpeta **startup**. Dentro de esta subcarpeta se podrán encontrar los logs divididos por inicio de programa, siendo configurados para que solo se mantengan los diez logs más nuevos.

Una vez que la pantalla de comprobación inicial se cierre, el programa ejecuta la pantalla principal dando bienvenida al usuario mediante la superposición de una imagen personalizable[3]. Se espera que para ese entonces, el programa este completamente listo para trabajar.

2.3. Estructura de carpetas y archivos

2.3.1. Descripción general del árbol de directorios

El programa MERIDA se organiza en una estructura de carpetas y archivos diseñada para optimizar su funcionamiento y facilitar tanto su mantenimiento como su futura ampliación. Cada elemento del directorio base cumple un propósito específico dentro del sistema: ya sea en la ejecución del programa, en la gestión de bases de datos o en el soporte de recursos auxiliares.

En esta sección se ofrece una descripción detallada de la función de cada carpeta y archivo, con el objetivo de que el lector comprenda la lógica de organización que sustenta al proyecto.

Es importante señalar que la estructura aquí presentada corresponde al código fuente original de MERIDA, es decir, a la versión destinada al desarrollo. Por lo tanto, no refleja las modificaciones que se realizan para generar paquetes ejecutables o instalables en distintos sistemas operativos. Si el usuario descarga directamente el código fuente, podrá verificar esta misma organización.

Por último, conviene destacar que, aunque la disposición general pueda variar ligeramente durante el empaquetado, las carpetas y archivos principales conservan su estructura básica. De este modo, si se presenta algún problema en la ejecución o instalación, el usuario podrá rastrear y localizar con facilidad los elementos relevantes siguiendo los nombres y jerarquías expuestas en esta sección.

[3]Esta imagen, que recibe el nombre de overlay, será explicada en las sección de **Ajustes de Estilo**

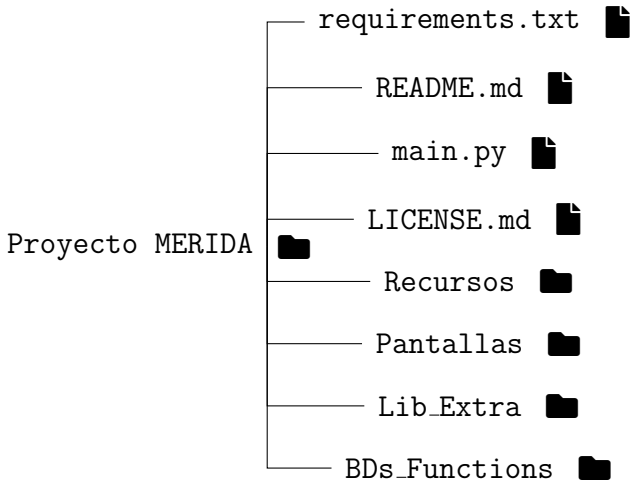


Figura 2.1: Estructura general del directorio del programa MERIDA

El primer nivel del directorio está compuesto por una serie de carpetas y archivos. Los elementos más importantes de este nivel son las cuatro carpetas mostradas en la Figura 2.1. Cada una de estas carpeta contiene una parte importante del sistema. Comenzaremos con la carpeta **Recursos**.

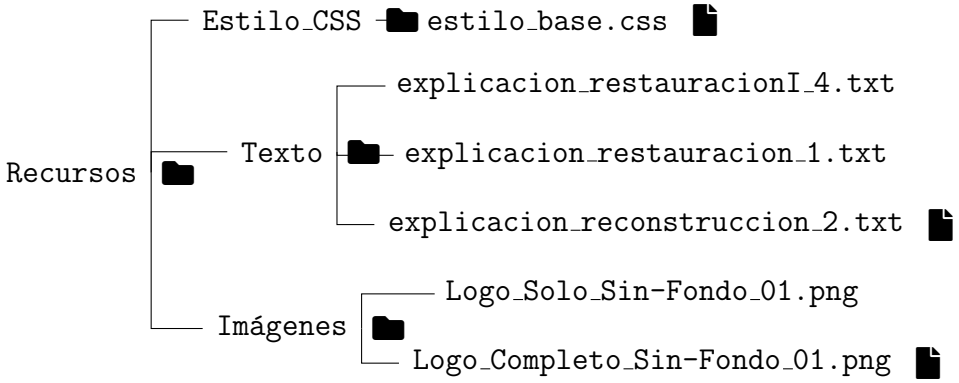


Figura 2.2: Contenido interno de la carpeta RECURSOS del directorio principal del programa MERIDA

La carpeta **RECURSOS** contiene los archivos que el programa necesita para mostrar información tanto visual como textual y, además, contiene los archivos css que el programa usa para definir su estilo (colores, etc). Dentro de la carpeta podemos encontrar tres subcarpetas:**Texto**, **Imágenes** y **Estilo_CSS**. Y tal como se cabría esperar, la carpeta texto contiene archivos txt que contienen explicaciones a mostrar en el programa. Por otro lado, la carpeta de imágenes contiene archivos de imagen (svg, png o jpeg) que el programa utiliza para mostrar logotipos y elementos decorativos similares. Por último, la carpeta de estilo css contiene los archivos que el programa usa para definir colores y elementos decorativos similares.

La utilidad de esta carpeta radica en ofrecer un espacio propio para mantener organizado los recursos especiales que el programa necesita.

La carpeta de **Pantallas** contiene todos los archivos que conforman la interfaz visual del programa MERIDA, y es por lo tanto, una parte importante de su funcionamiento.

Podemos encontrar dos términos importantes en esta carpeta: Pantalla y Vista. Las pantallas

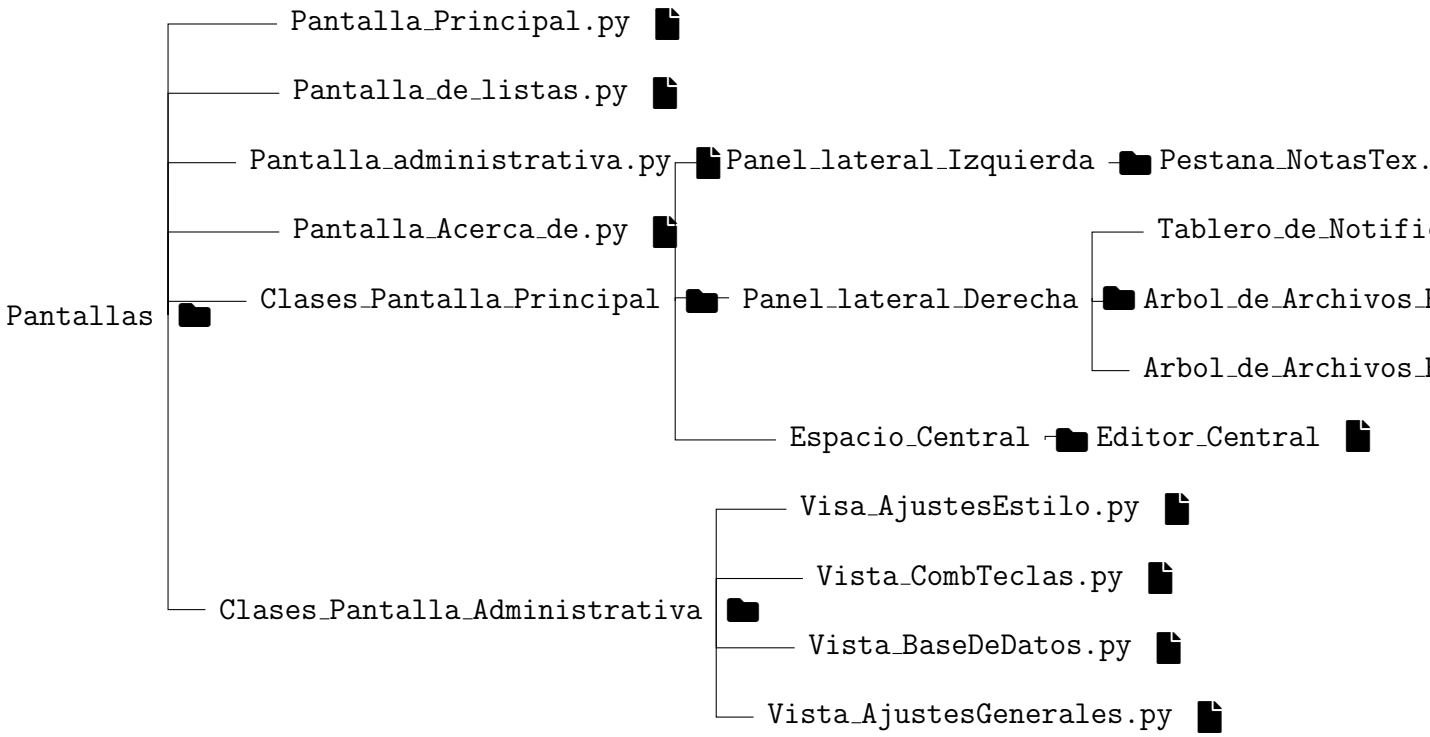


Figura 2.3: Contenido interno de la carpeta PANTALLAS del directorio principal del programa MERIDA

pueden llegar a contener diferentes vistas. Por ejemplo, la pantalla administrativa se encarga de posicionar y redirigir.[a] sus respectivas vistas, es decir, a cada sección de ajustes del programa. Se espera que este significado y uso se mantenga en futuros avances.

Ahora bien, los archivos encontrados en el directorio de **Pantallas** (ver Figura 2.3) corresponden a cada parte visual del programa. A continuación, se explican cada archivo y subdirectorio correspondiente (ver tabla 2.1).

Directorio:Pantallas/	
Archivo/Subdirectorio	Explicación
Pantalla_Principal.py	Archivo de ejecución principal que se encarga de mostrar la pantalla principal del programa (reconocible por ser la que contiene el EDITOR.)
Pantalla_de_listas.py	Archivo encargado de mostrar el contenido de la base de datos BD_TAGS. Esta base de datos contiene las **etiquetas** y las **direcciones de carpeta** de las mismas. También permite su modificación.
Pantalla_administrativa.py	Archivo encargado de ordenar y «llamar» a las vistas de ajustes (como combinaciones de teclas, bases de datos, etc.). Dicho de otro modo, es la pantalla que contiene los **ajustes** y **preferencias** del programa.
Pantalla_Acerca_de.py	Pantalla de créditos e información relevante del programa MERIDA.
Clases_Pantalla_Principal	Subdirectorio que contiene tres carpetas, cada una dedicada a un «espacio» propio de la pantalla principal. Estas son: *Espacio_Central*, *Panel_lateral_Derecha* y *Panel_lateral_Izquierda*. Como se puede suponer, cada carpeta contiene los archivos que permiten mostrar los elementos respectivos a cada espacio. Es decir, en la carpeta del Espacio Central, se encontrara, por ejemplo, el archivo correspondiente al **EDITOR** del programa que se encuentra ubicado, valga la redundancia, en la parte central. Y siguiendo esta lógica, se puede comprender los demás archivos de este subdirectorio.
Clases_Pantalla_Administrativa	Subdirectorio que contiene los archivos relevantes para las pantalla administrativa. Estos archivos muestran cada apartado de la zona «ajustes» o «preferencias» del programa.

Tabla 2.1: Cuadro de explicación del directorio de **Pantallas** del programa MERIDA

Por otro lado, la carpeta **Lib_Extra** contiene los archivos de «utilidad extra» que el programa necesita para realizar ciertas funciones. Al contrario de lo que debería creerse, esta carpeta es muy importante para el funcionamiento del programa ya que si sufre algún cambio, el comportamiento de ciertas funciones del programa podría verse afectado.

En el directorio de **Lib_Extra** (ver Figura 2.4) se encuentra todo archivo que ayude al funcionamiento del programa. Se tienen, hasta ahora, cuatro archivos en este directorio, cada uno con su función propia y relevante. Tenemos, así, al archivo **Tags_Formato.py** que permite dar «estilo» al texto ingresado en el editor. Quizás, contrario a lo que se podría creer a primera vista, este archivo no involucra directamente a la funcionalidad de etiquetas del programa. En palabras más simples, este archivo contiene funciones que crean tablas y aplican estilo (como color, tamaño, etc) al texto ingresado a secciones específicas del editor.

Por otro lado, tenemos al archivo **Rutas_Gestion.py**. Su función es contener en un solo lugar la «gestión» de las rutas que el programa usa para guardar archivos temporales, recurrentes, logs y recursos. Dado que el programa MERIDA tiene como objetivo, quizás lejano, llegar a ser multiplataforma, se incorporó prontamente una gestión de rutas. Si uno abre el archivo, en algún editor de texto o código, podrá ver que se manejan las rutas mediante la identificación

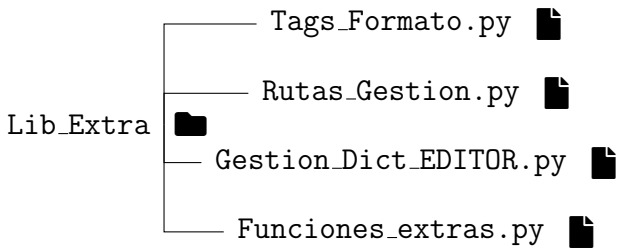


Figura 2.4: Contenido interno de la carpeta Lib_Extra del directorio principal del programa MERIDA

del sistema operativo donde corre el programa, teniendo como opción a Linux y a Windows[4]. Esto permite que se puede controlar más fácilmente cada ruta y se integre de mejor manera con el sistema operativo.

El archivo **Gestion_Dict_EDITOR.py** es un archivo de ayuda bastante específico. Se encarga de gestionar la modificación, registro o eliminación del contenido de un diccionario importante para el funcionamiento correcto del Editor Merida. Más allá de esta utilidad, este archivo no es usado por ninguna otra parte del programa.

Por último tenemos al archivo **Funciones_extras.py**. Este contiene funciones auxiliares que son usadas por todo el programa, de tal manera que un cambio en su código podría afectar varias partes del programa a la vez. Se recomienda usarlo si se quiere añadir funciones «generales» al programa en sí. Aunque también esta la opción de crear archivos separados si se desea.

Por último, vamos a explicar el directorio relacionado a las funciones de las bases de datos. No obstante, en esta sección tocaremos tan solo la explicación respectiva a la organización de archivos y carpetas del directorio, pues las bases de datos y sus archivos, serán explicadas con mayor detalle en el capitulo de **Preferencias del programa MERIDA** de este manual.

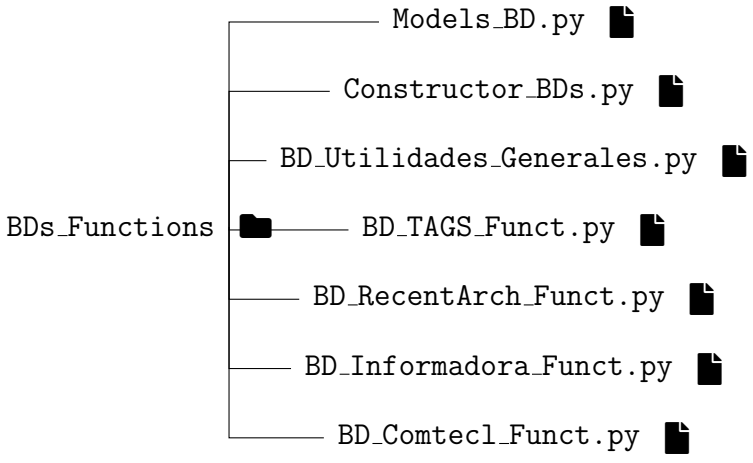


Figura 2.5: Contenido interno de la carpeta BD del directorio principal del programa MERIDA

El directorio de las funciones de las bases de datos(ver Figura 2.5) contiene todos los archivos que ofrecen funcionalidades al manejo de bases de datos del programa. Cabe decir que este directorio no se encuentran las propias bases de datos, ya que el objetivo de este directorio es

[4]Cabe destacar que el programa, actualmente, solo se encuentra disponible en Linux, aunque algunas partes, como la gestión de rutas, tengan como segunda opción a Windows.

almacenar únicamente los que otorguen funcionalidades a las bases de datos respectivas y/o estén directamente relacionados con ellas.

Cada base de datos tiene su archivo de «funciones». Este archivo, siempre representado con la terminación «**Funct**» al final de su nombre, contiene las funciones que se usan para trabajar con la base de datos a la que pertenecen[5]. Por otro lado, los archivos Models_BD.py, Constructor_Bds.py y BD_Utilidades_Generales.py son archivos de utilidad para trabajar con las bases de datos en **general**. Es decir, ofrecen utilidades que involucran a todas las bases de datos del **sistema** del programa. Ayudan, por ejemplo, a la construcción y creación de bases de datos o a la creación de respaldos.

Como se ha podido observar, el directorio del proyecto MERIDA está pensado para la expansión. Si, por ejemplo, se desea añadir una nueva pantalla al programa, basta con crear sus carpetas y archivos y añadirlos al código correspondiente. Y dado que la estructura de archivos del programa tiene está característica, se invita a los desarrolladores a **no cambiar la estructura general** de los directorios del programa sin previo consenso. Cambiarlo drásticamente, significaría modificar gran parte del código del programa y esto no debe hacerse a menos que haya motivos reales e indispensables delante.

2.3.2. Archivos temporales (tmp) y su función

Una vez terminada la explicación del directorio principal del proyecto, pasaremos a explicar la función que los archivos temporales tienen en el proyecto MERIDA.

Los archivos temporales son elementos de «uso y eliminación» que el programa genera para ciertos procesos. Estos archivos siempre tienen la extensión **.tmp** y se guardan en la ruta predeterminada de archivos temporales. Esta ruta es gestionada por el archivo **Rutas_Gestion** (Ver Figura 2.4). En GNU-Linux, se usa la siguiente ruta: **/.cache/MERIDA/Archivos_Temporales/**.

Siguiendo esta idea, podemos decir que los archivos temporales son usados generalmente para compartir datos de forma externa (es decir, fuera del código) entre diferentes funciones del programa.

Por ejemplo, hay una función que permite la reconstrucción de una base de datos. Este proceso involucra diversos pasos a cumplir, y uno de ellos es la creación de un archivo temporal. Este archivo temporal contiene únicamente un valor que el programa revisa cada vez que se inicia. Si este valor es positivo, se inicia la pantalla de «**Reinicio por reconstrucción de base de datos**», para luego eliminar dicho archivo. Esta es una de las aplicaciones que tiene el uso de los archivos temporales en MERIDA.

Ahora bien, generalmente los archivos tmp son simples archivos de texto donde se ingresan valores que podrán ser leídos y aplicados. Sin embargo, nada impide crear archivos temporales json, o .md. La única regla es que estén siempre en el directorio correspondiente y sean eliminados en cuanto su vida útil finalice.

Por otro lado, existen funciones especiales dentro del archivo **Funciones_Extras.py** (ver Figura 2.4) que permiten crear, leer y modificar un archivo temporal de MERIDA. Estas funciones

[5]El proyeco MERIDA usa SQLAlchemy para crear e interactuar con sus bases de datos. Algo que hay que tomar en cuenta para entender las razones detrás de los archivos existentes en el directorio BDs_Functions.

trabajan con una ruta predeterminada, por lo que conviene revisar la parte superior del archivo de funciones extras y ver la ruta. Dicha ruta está diseñada para trabajar con el directorio de archivos temporales ya mencionado.

Como conclusión, los archivos temporales sirven para compartir datos entre funciones del programa que, de otro modo, no podrían. Tienen una vida útil pequeña, puesto que son eliminados casi de inmediato.

2.3.3. Archivos recurrentes y datos que persisten entre ejecuciones

A diferencia de los archivos temporales, los **archivos recurrentes** son elementos *persistentes* que el programa usa en funciones específicas. Esto quiere decir que cada archivos recurrente contiene datos que deben ser guardados y permanecer entre ejecuciones del programa.

Por ejemplo, son muy utilizados para guardar opciones de las preferencias del programa. Se guardan, por ejemplo, el intervalo para realizar copias de seguridad automáticas, guardar la zona horaria, número de archivos a mantener, tema oscuro o claro, etc. Estos datos son utilizados por el programa cada vez que este se inicia para realizar validaciones o similares.

Claro que los archivos recurrentes pueden llegar a tener otras funciones, pero por el momento solo permiten guardar datos para ser utilizados en el siguiente inicio del programa. Y tal cual como sucedía con los archivos temporales, **Funciones_Extras.py** contiene herramientas para crear, leer y eliminar un archivos recurrente.

La ruta predeterminada para este tipo de archivos sigue la misma lógica de organización que con los archivos temporales, es decir, es gestionada por el archivo **Rutas_Gestion**.En el caso propio de los archivos recurrentes se usa la siguiente ruta en sistemas Linux: **/.config/MERIDA/Archivos_Recurrentes/**. Estos archivos generalmente son del tipo JSON, aunque las funciones que los crean y modifican (ubicadas en el archivo de **Funciones_Extras.py**) permiten la creación de archivos recurrentes txt.

Capítulo 3

Preferencias del programa MERIDA

3.1. Ajustes Generales

Bajo el nombre de **Preferencias** es que se puede encontrar el botón que permite entrar a la ventana de ajustes del programa. Este botón se encuentra junto con las demás opciones, en el **menú** del mismo programa. Una vez abierta la ventana de ajustes, se podrá ver dos elementos a recalcar:la barra lateral izquierda y el contenido en sí, donde se encuentran las opciones de ajustes.

En la barra lateral izquierda se pueden ver las diferentes **secciones** que tiene el programa para ser ajustado según las preferencias del usuario. Por el momento, los ajustes y secciones son muy pocos, casi inexistentes, dado la versión del proyecto. Por supuesto, en futuras actualizaciones, más y más ajustes se irán añadiendo. Pero por ahora se tiene únicamente las siguientes secciones de ajustes: **Ajustes Generales**, **Combinaciones de Teclas** y **Bases de Datos**. La primera será tratada en la presente sección de este capítulo, las otras en sus respectivas secciones.

Tal como su nombre lo indica, los **Ajustes Generales** almacenan todos los ajustes o preferencias de tipo **general**. Esto puede traducirse a ajustes que afecten todo el programa en su totalidad o en aquellos que sean más necesarios dado su importancia y búsqueda de los usuarios.

Uno de esos ajustes generales es la **Zona Horaria**. Este ajuste permite seleccionar una zona horaria que será utilizada para ajustar fechas (como las de las bases de datos) y así poder mostrarlas correctamente. Se irán añadiendo más zonas horarias en las siguientes versiones del programa.

Otro ajuste importante es el relacionado con la **Ruta Predeterminada**. Más específicamente, el poder modificarla. La ruta predeterminada es la ruta «obligada» que el programa usa para guardar los «documentos» generados por el editor si es que no se ha especificado otra **etiqueta**. Si se desea cambiar esta ruta (algo que, personalmente, no recomiendo) puede hacerlo desde este ajuste. Asegúrese de insertar correctamente la nueva ruta y guardarla.

Por otro lado, existen ajustes generales relacionados a las **Bases de Datos**. Aunque estas tengan sus propios ajustes en su respectiva sección, aquellos que se encuentran en la pestaña de ajustes generales están centrados en la configuración rápida y simple de algunas opciones del programa para con las bases de datos. Uno de estos ajustes, y por el momento el único,

es la **Configuración de Respaldo Automático**. Esta opción permite configurar dos datos importantes para la realización de los respaldos de bases de datos. El primero es el **número de archivos a mantener**, que permite definir el número de archivos que se mantendrán en la carpeta de respaldos. Esto quiero decir que, si por ejemplo, lo definimos en 3, el programa mantendrá siempre los 3 archivos más recientes de la carpeta de respaldos y eliminará todos los demás. Por otro lado, la segunda opción permite configurar el **intervalo** que el programa usará para decidir si se debe realizar un respaldo o no. Se divide en días, meses y años; y dichos ajustes son revisados cada vez que el programa se inicia. Para guardar todos los ajustes, no hay que olvidar que se debe presionar el botón **Guardar configuración de guardado automático**.

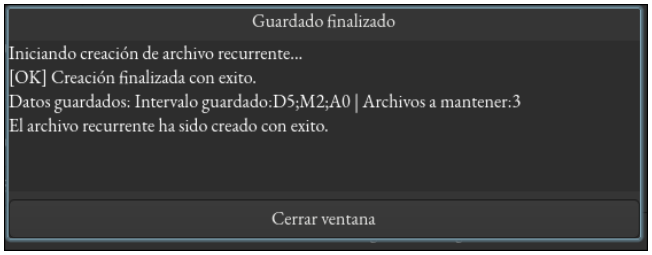


Figura 3.1: Ventana de **Modificación** de los Atajos de Teclado (Captura de Pantalla)

Si todo salió bien, se presentará una pequeña ventana que mostrará los pasos que se siguieron para guardar los ajustes (ver Figura 3.1). En dicha ventana se mostrará también los datos guardados.

Por último, siempre al final, se encuentran los opciones para **Restaurar Valores Predeterminados**. En dicho apartado existen las opciones para restaurar ciertos ajustes a su estado predeterminado. Por el momento solo existen dos de estas opciones:**Restaurar combinaciones de teclas predeterminadas** y **Restaurar la Ruta Predeterminada de Fábrica**. Conviene informar que estas opciones eliminarán cualquier modificación en las bases de datos respectivas a cada valor que el usuario haya hecho[1].

3.2. Combinaciones de Teclas

El programa permite la modificación de todas sus combinaciones de teclas (también llamadas **atajos de teclado**) que fueron registradas por el grupo encargado del proyecto MERIDA. Esto es posible gracias a la sección de **Combinaciones de Teclas** de la pantalla de preferencias o ajustes del programa. Una vez el usuario se encuentre en dicha sección, podrá ver, mediante una tabla, todas las combinaciones de teclas disponibles para modificar según su gusto particular.

Dicha tabla dispone de cuatro apartados (ID, Nombre, Combinación de tecla y Descripción de Acción) cuyo contenido es extraído de una base de datos propia. El único elemento modificable de esta tabla es la combinación de teclas como tal, dejando los demás campos (ID, Nombre y Descripción de Acción) a decisión del creador correspondiente. Para dicha modificación, tan solo es necesario *clikear* en el botón inferior de nombre **Modificar**. Una vez hecho esto, se abrirá una pequeña ventana que le pedirá insertar la nueva modificación de la combinación anterior (Ver Figura 3.2).

[1]Se usa **BD_TAGS** para la ruta predeterminada y **BD_ComTecl** para las combinaciones de teclas.

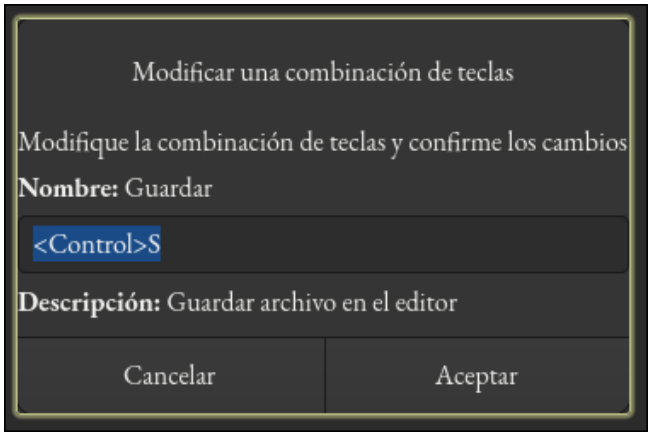


Figura 3.2: Ventana de **Modificación** de los Atajos de Teclado (Captura de Pantalla)

Ahora bien, hay un aspecto muy importante que explicar antes de que se puede modificar una combinación de teclas. El programa MERIDA se encuentra construido sobre GTK 4 y Python. Por tanto, se debe tener en cuenta las instrucciones sobre este tema para así poder modificar la combinación correctamente. Si el usuario ingresa una combinación de teclas incorrecta, el programa le notificará de su error mediante otra ventana en cuyo contenido se le aconsejará seguir el formato mostrado en algunos ejemplos (Ver Figura 3.3).

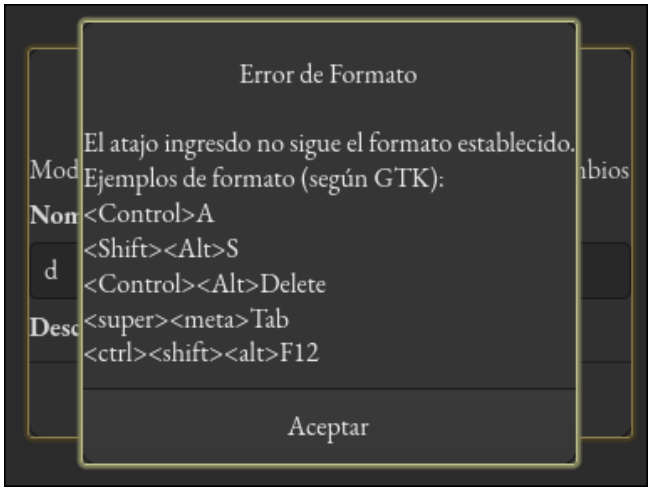


Figura 3.3: Ventana de **error** al insertar una combinación de teclas incorrecta (Captura de Pantalla)

Como se puede ver en la Figura 3.3, se ofrece una serie de ejemplos correctos de las combinaciones. En palabras más sencillas, tan solo es necesario ingresar los caracteres *menor que* y *mayor que*. Estos mismos deben contener el identificador correspondiente a la tecla que se desea utilizar como *disparador* de la acción seleccionada. Como tal, se considera ideal el mensaje de error mostrado en la Figura 3.3, ya que se considera lo suficientemente explicativo para cualquier persona.

De todas maneras, si se tiene problemas al encontrar el identificador correcto para una tecla, se recomienda buscarlo en la documentación oficial de GTK4.

3.3. Bases de Datos

Las bases de datos del programa **MERIDA** constituyen un componente esencial para su funcionamiento. De hecho, la sección de **Ajustes de Bases de Datos** ha sido una de las que más tiempo de desarrollo ha requerido. Esta sección se encuentra al final de la pantalla de administración, bajo el nombre **Bases de Datos**. Al igual que ocurre con la sección de combinaciones de teclas, el usuario encontrará una tabla que muestra todas las bases de datos registradas en el sistema.

La tabla principal está compuesta por cinco columnas: **ID**, **Nombre**, **Descripción**, **Última actualización registrada** y **Estado**. Estas y otras propiedades se explican en detalle más adelante. A la derecha de la tabla se presentan, en disposición vertical, las opciones que el programa ofrece para manipular las bases de datos. Antes de describir estas opciones, conviene comprender el funcionamiento general de esta pantalla.

Si el usuario hace *clic derecho* sobre una fila de la tabla, se abrirá un formulario que muestra todas las propiedades asociadas a la base de datos seleccionada. Cabe destacar que toda la información de gestión de bases de datos se almacena en una base de datos central denominada **BD_Informadora**, la cual juega un papel fundamental: allí se guardan los metadatos necesarios para llevar a cabo las distintas acciones relacionadas con las bases de datos.

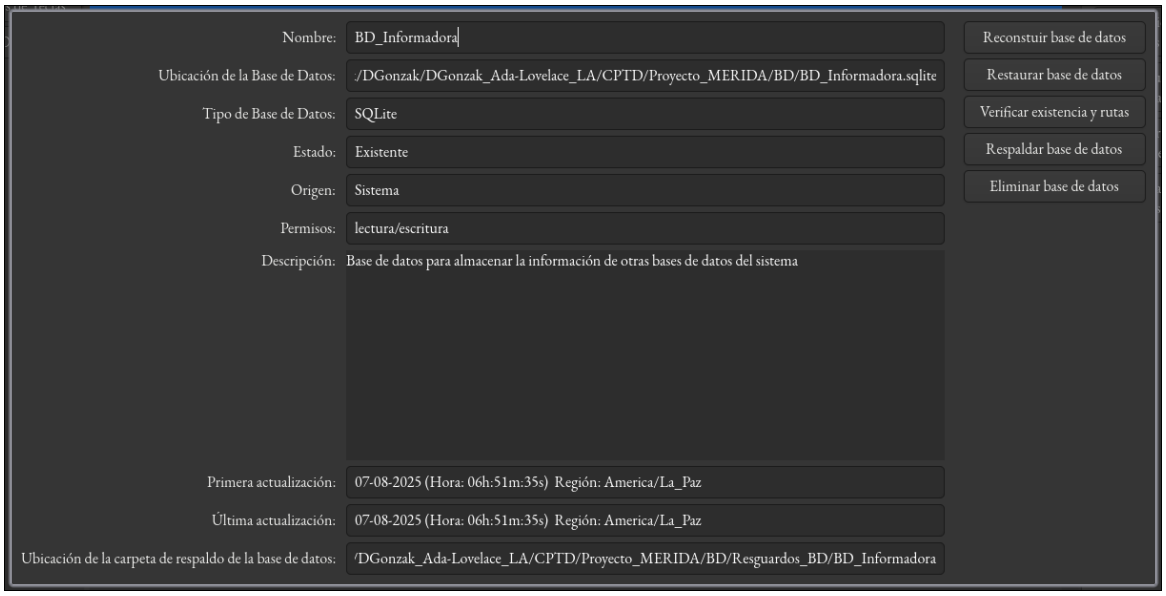


Figura 3.4: Formulario de propiedades de una base de datos registrada en MERIDA (captura de pantalla).

3.3.1. Propiedades de una base de datos en MERIDA

Propiedad: Nombre e ID

Tanto el **Nombre** como el **ID** deben ser únicos dentro de la base de datos **BD_Informadora**. No se permite el uso de espacios ni caracteres especiales. Por convención, todos los nombres de bases de datos comienzan con el prefijo **BD_**.

Propiedad: Ubicación

Indica la ruta en el sistema de archivos donde se encuentra el archivo binario de la base de datos (por ejemplo, un archivo `.sqlite`).

Propiedad: Tipo de Base de Datos

Muestra el tipo de motor utilizado. Aunque el sistema puede identificar otros motores, actualmente MERIDA trabaja principalmente con **SQLite**.

Propiedad: Estado

Indica si la base de datos está disponible. Puede ser:

- **Existente**: cuando el archivo binario ha sido localizado y validado.
- **Inexistente**: cuando la validación falla o el archivo no se encuentra en la ruta indicada.

Propiedad: Origen

Clasifica la base de datos según su procedencia: **Sistema**, **Usuario**, **Externa** o **Por extensión**. Actualmente, el sistema solo trabaja de manera completa con bases de datos de tipo **SISTEMA**, es decir, aquellas que vienen preinstaladas con MERIDA. Las funciones como restauración, reconstrucción y respaldo se aplican únicamente a este tipo de bases de datos.

Propiedad: Permisos

Aunque esta propiedad aún no está implementada, en el futuro permitirá restringir modificaciones a determinadas bases de datos según la configuración del usuario. Por el momento, todas las bases de datos poseen permisos de **lectura/escritura**, aunque el cambio de este parámetro no tiene efecto práctico.

Propiedad: Descripción

Campo de texto que ofrece una breve explicación o nota descriptiva de la base de datos.

Propiedades: Primera y Última actualización

Indican, respectivamente, la fecha en que la base de datos fue registrada en el sistema y la fecha de su última modificación. Se muestran siguiendo el formato de fecha configurado en la zona horaria del sistema.

Propiedad: Carpeta de respaldo

Muestra la ruta absoluta donde se almacenan las copias de seguridad de la base de datos. De forma predeterminada, todas las bases de datos comparten la misma carpeta de respaldo.

3.3.2. Opciones de gestión de bases de datos en MERIDA

Las propiedades descritas anteriormente se utilizan en distintos procesos internos del sistema. Aunque es técnicamente posible modificarlas de forma manual[2], no se recomienda hacerlo, especialmente con las bases de datos de origen **SISTEMA**.

A la derecha de la tabla de bases de datos se encuentra un conjunto de botones que permiten ejecutar diferentes operaciones:

- **Registrar base de datos**: añade un nuevo registro en **BD_Informadora**. Aunque las bases de datos de usuario aún no tienen funcionalidad completa, esta opción prepara el sistema para futuras versiones.
- **Operaciones grupales**: ejecuta procesos que afectan a todas las bases de datos de origen **SISTEMA**. Permite realizar tareas de **reconstrucción** (eliminar y recrear bases de datos) y **respaldo** (generar copias de seguridad).
- **Actualizar vista**: recarga manualmente la tabla de bases de datos para mostrar los cambios recientes.
- **Insertar BDs predeterminadas**: añade a **BD_Informadora** los registros de las bases de datos que forman parte del sistema predeterminado.
- **Restaurar desde respaldo**: permite recuperar bases de datos (archivos binarios y registros) desde la carpeta de respaldos.

Cuando se accede al formulario de propiedades de una base de datos individual (Figura 3.4), aparecen además dos opciones adicionales:

- **Verificar existencia y ruta**: comprueba si la base de datos se encuentra en la ubicación indicada y actualiza su estado a **existente** en consecuencia.
- **Eliminar base de datos**: elimina el registro de **BD_Informadora** o el archivo binario correspondiente (este último, también elimina el registro en BD_Informadora)

3.3.3. Archivos de restauración y respaldo

Al generar una copia de seguridad de una base de datos, MERIDA crea siempre dos archivos: uno con extensión `.sqlite` y otro con extensión `.sql`.

[2]Por ejemplo, editando directamente los registros en la base de datos INFORMADORA

- El archivo **.sqlite** es una copia exacta del archivo original.
- El archivo **.sql** contiene instrucciones en texto plano para recrear la base de datos desde cero.

En los procesos de restauración, el programa utiliza primero el archivo `.sqlite` por su rapidez y simplicidad. Si este no está disponible o se encuentra dañado, recurre al archivo `.sql`. Aunque nada impide al usuario utilizar cualquiera de los dos, se recomienda priorizar siempre el uso de las copias `.sqlite`.

En los procesos de respaldo, por otro parte, el programa nombra ambos tipos de archivos según la fecha de creación y una estructura específica. Cambiar la estructura significaría dañar procesos de restauración, ya que el programa utiliza estas fechas para realizar el **respaldo automático**. La estructura sigue el siguiente orden:

[nombre_BD]_respaldo_[tiempo_creacion]

En esta estructura, nombre_BD es el nombre de la base de datos de la cual se creó el respaldo y tiempo de creación es el momento exacto en el cual se creó el respaldo (ignorando segundos). Siguiendo esta estructura de nombramiento, se obtiene el siguiente resultado de ejemplo:

BD_Informadora_respaldo_2025-08-11_07-30.sql

Cualquier cambio en esta estructura, podría tener consecuencia en los procesos de restauración del programa. Como última nota sobre las bases de datos, cabe mencionar que si el programa detecta la inexistencia de respaldos en la ruta de resguardos predeterminada (esto es, en sistemas Debian Linux: /.local/share/MERIDA/BDs/Resguardos_BD) mandará un aviso en la pestaña **avisos** del **panel lateral derecho** de la interfaz principal (Ver Figura 3.5).

Ahora bien, para crear respaldos solo se debe ir a la pantalla de preferencias, sección base de datos y en la opción **Operaciones Grupales de Bases de Datos** se podrá crear respaldos para todas las bases de datos que aparezcan luego de realizar la búsqueda correspondiente. Una vez hecho esto, el mensaje no debería volver a mostrarse y se podrá **limpiar el tablero** de avisos.

3.4. Ajustes de Estilo

La sección que permite la personalización «visual» del programa MERIDA se encuentra bajo el nombre de **Ajustes de Estilo** en la pantalla de preferencias. En dicha sección se encuentran, asimismo, las opciones para personalizar la apariencia del programa. Por desgracia, dado la versión de MERIDA, se ofrecen opciones limitadas; a saber, tan solo dos: **Selección de Tema** y **Selección de Fondo de Pantalla**.

Además, se recomienda ampliamente, guardar los cambios cada vez que se modifique alguna opción. De lo contrario, los nuevos valores no se aplicarán.

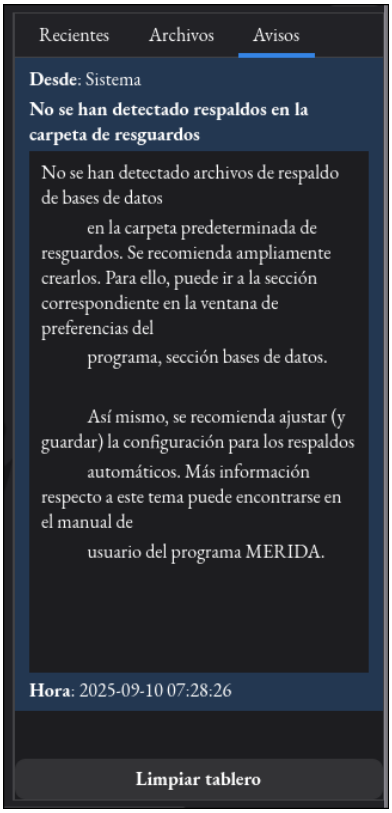


Figura 3.5: Aviso de detección de inexistencia de respaldos (Captura de Pantalla)

3.4.1. Selección de Tema

Tal como se podría esperar, se dispone del tema **claro** y del tema **oscuro**. Se podrá comprobar ambos temas accionando los botones según el nombre de la opción que se desee, o marcando directamente la casilla inferior si se desea que siga el tema propio del sistema.

En un futuro se incluirán opciones más avanzadas para poder definir su propio tema personalizado.

3.4.2. Selección de Fondo de Pantalla

El programa MERIDA (V.1) permite al usuario seleccionar un fondo de pantalla de su elección. Para ello se dispone un espacio propio donde se podrá elegir la imagen png o jpeg que se prefiera. El programa guardará la ruta absoluta y directa a la imagen y la usara en cada inicio de sesión. Si la imagen no es válida de alguna forma (no existe, esta dañada, etc) el programa usara la imagen predeterminada.

Por el momento solo se permita la elección de imagen para la **pantalla de carga** que se refiere a la imagen que aparece justo después de que la *pantalla de comprobación inicial* desaparezca. Esta pantalla de carga (que en realidad es un overlay de GTK4) desaparece cuando el programa ha finalizado de ajustar la interfaz según valores predeterminados o guardados desde la última sesión.

Ahora bien, se espera que en un futuro, las opciones de *fondo de pantalla* personalizados se

amplíen, permitiendo al usuario elegir varias imágenes para varias pantalla o, si lo prefiere, rotar por arranque según una carpeta de imágenes[3].

[3]Dichas funcionalidades se esperan poder publicarse con la versión 2 del programa MERIDA

Parte II

Componentes (Módulos) del Proyecto MERIDA

Capítulo 1

Editor Merida (Por DGonzak)

1.1. Introducción al Editor Merida

El **Editor Merida** es el primer componente a presentar del **Proyecto MERIDA**. Forma una elemento básico en su funcionamiento dada sus características que proporcionan al proyecto (programa) parte de su identidad. Como tal, el editor permite la generación de documentos/notas[1] cuyo objetivo principal es nutrir el proyecto de ideas y/o información que será utilizada por los demás componentes del proyecto MERIDA.

Al ser la herramienta con la cual todo se inicia, el editor adquiere una importancia fundamental en el desarrollo de un proyecto en MERIDA. Todo inicia en el editor, y desde su ahí, los proyectos personales solo van aumentando en características y tamaño. Por ende, el editor debe ser el primer componente en ser explicado y presentado con la primera versión del proyecto. A continuación explicaremos sus características y su uso previsto. Además, se mencionará sus vínculos con futuras herramientas y algunas características adicionales.

1.2. Uso previsto y filosofía del diseño

El Editor Merida tiene una filosofía de **simplicidad y utilidad inmediata**. Intenta ofrecer la posibilidad, tanto a escritores académicos o creativos, de poseer una herramienta siempre lista para ayudarlos en su proceso de creación. Intenta, además, ofrecer otras herramientas que potencien dicho proceso, teniendo mucha flexibilidad a la hora de crear contenido.

El diseño modular el editor permite su evolución futura, además de su integración con otros componentes del proyecto MERIDA. Fue pensado para ofrecer vínculos con las próximas herramientas, sin perder su foco principal: la escritura. Así, el editor merida busca ofrecer un entorno sencillo, sin distracciones no necesarias, con una estructura abierta, flexible, y una base que invita a la creación y mejoramiento de componentes.

[1] En este sentido, *documento* es cualquier **texto** creado por el editor

1.3. Interfaz de usuario y navegación

El **editor merida** se compone de tres secciones claramente diferenciables entre sí. Dejando de lado la zona superior y la zona inferior, que pertenecen a la interfaz en sí, es decir, en esas zonas se encontrarán elementos básicos de un programa, como ser: menús e información general. Ahora bien, cada zona del editor merida tiene una función clara asignada. En primer lugar tenemos a las dos **barras laterales**, cuyo protagonismo se debe a sus características utilitarias no solo para el componente del editor, sino que para las futuras actualizaciones. El editor como tal, se encuentra en la parte central, entre ambas barras laterales.

Comenzaremos, así, con las barras laterales para luego pasar a explicar el editor como tal. Como se dijo, estas barras son, o quizás mejor dicho, serán utilizadas por otros componentes del proyecto MERIDA. En cuanto al componente «**editor**», estas barras obtienen tres pestañas relacionadas, pero no limitadas, al editor en sí. En la barra lateral izquierda, se encuentra la pestaña de «**Notas**», que sirve para la generación de notas individuales relacionables al documento creado por el editor. Al lado derecho, es decir en la barra lateral derecha, se encuentran dos pestañas que sirven como navegación y están directamente relacionadas a la característica de «**etiquetas**» del editor. La primer pestaña sirve para ver los **archivos recientes** y la segunda para ver los archivos según **las carpetas extraídas o señaladas por las etiquetas**. Por otro lado, igualmente ubicada en el panel lateral derecho, se encuentra una última pestaña cuya función es más informativa que las otras. La pestaña de **avisos** sirve para que el programa pueda mostrar avisos importantes para su correcto funcionamiento. Por ejemplo, si se detecta que no hay resguardos en la carpeta de resguardos de bases de datos, se envía un aviso al iniciar el programa.

Ahora bien, como seguramente se podrá apreciar, todas las herramientas de las barras laterales pueden ser utilizadas por otros componentes, en especial aquellas que no están muy directamente relacionadas al editor. En este ámbito entran por ejemplo, las notas, explicadas a mayor detalle más adelante.

Ahora bien, en cuanto al editor en sí, ubicado en la parte central, entre ambas barras laterales, este se compone de tres elementos distintivos. El usuario se encontrará con tres campos de entrada, a saber, dos del mismo tamaña y el último de un tamaño mucho mayor. El primer elemento, es decir, el primer campo de entrada se usa para «**nombrar**» el archivo y el documento a crear. Es así que se llama **Campo de Título.**Este campo se usa para nombrar el archivo .merida, propio de todo el proyecto, y además otorga un identificativo propio al documento generados, como lo hace cualquier nombre en un archivo. El programa, esto es, la función de guardado del editor, utiliza el título para nombrar el archivo .merida, añadiéndole **barras bajas** a los espacios entre palabras.

Pasar de esto:

Título de Documento

A esto:

Título_de_Documento

Esto se hace para asegurar que el archivo .merida sea compatible en cualquier situación. Por otro lado, cabe destacar que la función de guardado no funcionará si alguno de los campos de entrada se encuentran vacíos. Esto se traduce a que, si el programa detecta que el campo de título se encuentra vació, no procederá con el guardado.

El segundo campo de entrada recibe el nombre **Campo de Etiquetas** y sirve para introducir las etiquetas anteriormente creadas en la pantalla de listas. La funcionalidad de etiquetas se explicará más adelante, ya que tiene su apartado propio. En este campo solo se deberá introducir etiquetas, y aunque en sí, el campo permita la introducción de texto, no es recomendable. El programa tiene pautas para detectar etiquetas en este campo, por lo que, en primer lugar, no deberían ser detectadas como tal. Más adelante se explicará el significado de los colores que adquieren las etiquetas si se siguen las pautas para detectarlas.

Por último, se encuentra el **Campo del Cuerpo** que sirve para almacenar el contenido del documento a generar. Esta es la parte primordial del editor, ya que alberga el contenido del documento. Sus características de funcionalidad, es decir, edición de documentos, serán tratas en su apartado correspondiente.

Estas tres partes conforman al editor en sí. Sin embargo, hablando de la interfaz del programa, se tiene dos secciones antes mencionadas. La primera es la zona del menú. Ahí se pueden encontrar las opciones para cambiar de pantalla y entrar a la sección de ajustes o de creación de etiquetas respectivamente. También se encontrará, en la parte superior justo al extremo opuesto de las opciones del menú, un pequeño botón que sirve para ocultar y mostrar las barras laterales. En la zona inferior se encuentra una **barra de progreso** y una **etiqueta de estado**. Ambos elementos sirven como métodos informativos. La barra de progreso para operaciones concretas, como el proceso de guardado y la etiqueta para información general. Con el paso del tiempo, es decir, a medida que el usuario vaya usando el programa, verá la utilidad de estas herramientas.

En general, la interfaz de usuario del proyecto MERIDA tiene como objetivo ser simple, pero a la vez, ofrecer incentivos de complejidad que motiven a las personas a interactuar con ella. Por lo tanto, es sumamente aconsejable que el usuario «pruebe» la interfaz a fin de familiarizarse con ella.

1.4. Funcionalidades Principales

1.4.1. Creación y Edición de Documentos

El **editor merida** tiene la funcionalidad de crear y editar documentos guardados en un archivo «.merida». Se podrá «cargar» este tipo de archivos, que no es más que un archivo .rar, desde la opción correspondiente en la opciones del menú. Como todo programa, si selecciona la opción **Abrir**, se le ofrecerá la opción de selección. Una vez cargado el archivo, los campos correspondientes (Campo de Título, etiquetas y cuerpo) serán llenados con la información encontrada en el archivo de origen.

Ahora bien, si usted desea crear un **nuevo documento**, tan solo debe comenzar a escribir en los campos de entrada vacíos. Una vez termine su creación, podrá guardarlo, creando así un nuevo archivo .merida. Recuerde que el título que haya puesto será el nombre del archivo .merida que se guardará según las direcciones de carpeta asociadas a las etiquetas que usted haya usado. Por

último, existe la opción de crear un documento **nuevo** en las opciones del menú. Esta opción **elimina** el contenido de los campos de entrada, a fin de otorgarle un espacio de trabajo limpio. Si usted no ha guardado su último trabajo, el programa no le dejará «limpiar» los campos de entrada a menos que usted así lo autorice en la ventana de confirmación.

Ahora bien, la edición de documentos se realiza mediante texto simple. El editor merida acepta algunas pautas para dar «estilo» a fragmentos de texto. Así, usted puede utilizar las mismas indicaciones que el tipado Markdown. No obstante, el editor aún no soporta la totalidad de este sistema de tipado. A continuación se nombran las pautas aceptadas:

Tipado	**Resultado**
**Texto en negrilla	****Texto en negrilla**
*_Texto en cursiva_*	*_Texto en cursiva_*
# Titulo	# Titulo
## Titulo medio	## Titulo medio
### Titulo pequeño	### Titulo pequeño
#### Titulo pequeño en cursiva	#### *Titulo pequeño en cursiva*

Tabla 1.1: Tipado aceptado por el editor merida

Como se puede ver en la Tabla 1.1, el editor merida acepta los comandos básicos de Markdown para dar estilo al texto. Por desgracia, aún no se tiene soporte para la totalidad de los comandos de Markdown. Se piensa, sin embargo, que dado el objetivo del editor, ofrecer una herramienta de **escritura rápida**, el estilo dado es más que suficiente. Si se desea un acabado más profesional, se recomienda ampliamente el uso de LaTeX.

Ahora bien, el editor merida no soporta aún otras características de interés para la escritura académica. A saber, la secuencia numérica, un sistema de referencias, soporte de tablas o cuadros y, quizás la más importante, el sentido de páginas. El editor merida no sigue la visualización por páginas; ofreciendo en cambio un lienzo infinito donde usted podrá escribir cuanto desee. Sin embargo, se prevee añadir estas funcionalidades en un futuro, acercando el proyecto MERIDA al ámbito académico. No obstante, como se mencionó hace poco, para ámbitos más profesionales, es mucho más útil hacer uso de LaTeX, dado su enfoque y trayectoria tan amplia.

1.4.2. Sistema de Etiquetas

Las etiquetas forman parte fundamental del funcionamiento de editor merida. Fueron la principal idea que motivó su creación, por lo que, dada su importancia, es momento de explicar su funcionamiento. Las etiquetas son un elemento direccional que el editor merida usa para saber donde guardar los documentos creados en él. Es decir, las etiquetas llevan asociadas, en una base de datos propia, las direcciones de carpeta designadas por el usuario para guardar archivos .merida de genero similar. Así, las etiquetas son indicaciones de guardado, y se pueden crear tantas como uno desee.

Las etiquetas se usan para señalar el lugar de guardado. Cada etiqueta se ingresa al **Campo de Etiquetas** según un formato específico que permite al programa reconocerlas como tal y seguir sus instrucciones de guardado. Para ello es necesario seguir dicho formato, ya que de

lo contrario, el programa será incapaz de reconocer las etiquetas. Sin embargo, trataremos en primer lugar su creación.

El editor merida posee una ventana específica para gestionar la creación de etiquetas. Se puede acceder a ella desde las opciones del menú, en la sub-sección de «**Administrar**» y, finalmente, haciendo click en la opción **Ver listas**. Aparecerá una ventana nueva que contendrá una tabla dividida en dos columnas, etiquetas y direcciones de carpetas.

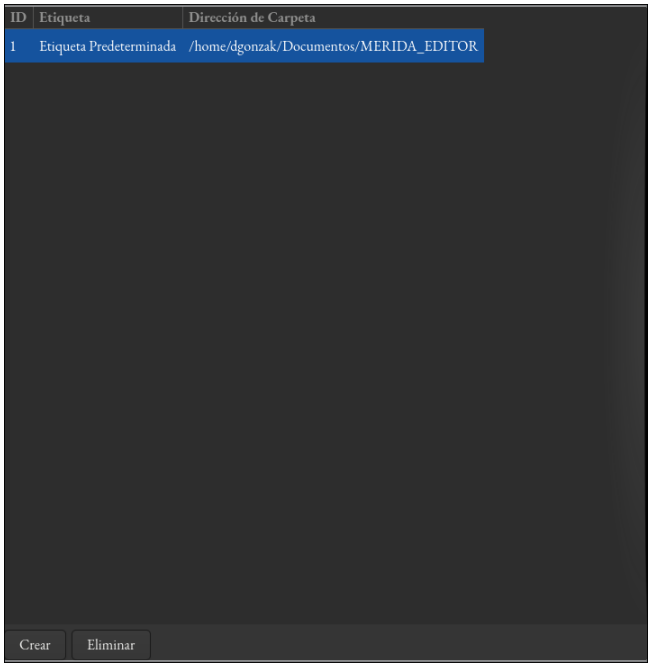


Figura 1.1: Ventana de **listas** para gestión de etiquetas (Captura de Pantalla)

Tal como se puede observar en la figura 1.1, de forma predeterminada, siempre existirá la **Etiqueta Predeterminada** creada en la ventana de listas. Esta etiqueta sirve cuando no se ha ingresado ninguna etiqueta en el campo correspondiente. Como tal, la etiqueta no debe ser usada de modo recurrente, ya que la idea del editor es que el usuario cree y use sus propias etiquetas. Por otro lado, la etiqueta predeterminada no puede eliminarse. Si usted la elimina, esperemos por accidente, en el siguiente reinicio del programa se volverá a crear.

Ahora bien, la creación de etiquetas es sumamente sencilla. Basta con dar click en el botón **Crear** de la ventana de listas (Ver Figura 1.1) ubicado en la parte inferior de la misma. Al lado de este, además, se encuentra en botón de **Eliminar**, que permite la destrucción de la etiqueta y su dirección de carpeta relacionada.

Una vez haya usted hecho click en el botón de crear, se le abrirá una ventana pequeña que le permitirá crear una etiqueta e ingresar una dirección de carpeta que quiera relacionar con ella.

Usted podrá ingresar el nombre identificativo de la etiqueta en el primer campo disponible. El nombre no debe repetirse, es decir, debe ser único en toda la base de datos. De lo contrario, el programa no le dejará crear la correspondiente etiqueta. Una vez establecido el nombre de la etiqueta, usted tendrá dos opciones para relacionarla con una carpeta. Siempre es más recomendable **crear** una carpeta nueva para facilitar el reconocimiento general de sus archivos. No use carpetas que ya tengan contenido, ya que aunque esto este «permitido» por el programa, puede generar confusiones. Bien, la primera opción es ingresar la ruta de **forma manual**, escribiéndola directamente en el campo de entrada **Ruta de Carpeta** (Ver Figura 1.2). La

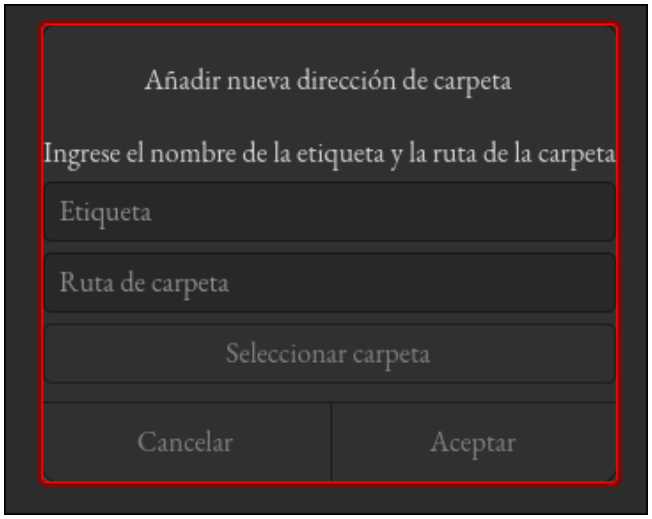


Figura 1.2: Ventana de **creación** de una etiqueta

segunda opción es seleccionarla utilizando el propio seleccionador de archivos de su sistema. Dando click en el botón **Seleccionar carpeta** podrá usted navegar de forma «gráfica» hasta la carpeta deseada y seleccionarla. Se recomienda ampliamente el uso de la segunda opción.

Ahora bien, el programa intenta comprobar que la ruta de carpeta ingresada evidentemente exista. Para ello buscará en su sistema la ruta, y solo si está existe y carece de permisos administrativos[2] permitirá su creación directa. Si el programa identifica que la ruta no existe, le preguntará si desea crear dicha ruta. Dependerá de usted esta última parte.

Una vez terminada la creación de la etiqueta, y seleccionada la ruta de su carpeta, deberá hacer click en el botón de aceptar de la ventana. La etiqueta se guardará en la base de datos y ahora estará lista para su uso. Podrá evidenciar su registro identificándola en la **pantalla de listas** anteriormente mencionada (Ver Figura 1.1) .

Por otro lado, la utilización de las etiquetas es bastante sencilla. En el editor, es decir, en los campos de textos descritos en la sección **1.3. Interfaz de Usuario y navegación**, existe un campo de entrada específico para el uso de las etiquetas del programa. Todas las etiquetas deben ir en ese campo de entrada siguiendo un formato de tipeo específico. Una vez el foco del programa termine «saliendo» del campo de entrada, el programa intentará identificar las etiquetas siguiendo las pautas de formato establecidas por su creador. Si la etiqueta ha sido correctamente insertada y creada en la base de datos, el programa la reconocerá y añadirá su ruta de carpeta a los árboles de navegación ubicados en la **barra lateral derecha**. Desde ahí, haciendo doble click, usted podrá navegar entre los diferentes archivos **.merida** que la carpeta contenga.

El formato para insertar correctamente las etiquetas es el siguiente:

[_Etiqueta_]

Como se puede ver en el ejemplo, el formato consiste en aplicar dos corchetes (abierto y cerrado) y dos guiones bajos antes de insertar la etiqueta, que debe estar en el medio de estos caracteres.

[2]El programa es incapaz de crear y funcionar con rutas de carpetas que necesiten permisos administrativos (sudo, por ejemplo) para escritura o lectura de archivos. Use siempre carpetas que no requieran estos permisos.

El programa es capaz de identificar espacios en el nombre de la etiqueta, por lo que no es necesario que usted coloque guiones bajos entre el propio nombre de la etiqueta.

Correcto: [_Etiqueta con Espacio_]
No Recomendable: [_Etiqueta_con_Espacio_]

Ahora bien, el programa posee una pequeña identificación por colores que le ayudará a saber si la etiqueta está o no está registrada en la base de datos. Si el programa logra identificar correctamente (según el formato ingresado) la etiqueta, y además, logra convalidar su existencia en la base de datos, «pintará» la etiqueta de color **verde**. De lo contrario, si no existe en la base de datos, la «pintará» de rojo (Ver Figura 1.3). Si usted intenta guardar teniendo una etiqueta en verde y una en roja, el programa detectará **Etiquetas en rojo**, por lo que le preguntará si desea crear una ruta para dichas etiquetas. Si acepta, será llevado a la pantalla de listas. Si no acepta, el guardado continuará siguiendo únicamente las rutas de las etiquetas comprobadas, es decir, que estén en verde. Si, por otro lado, solo hay etiquetas en rojo, el programa insertará automáticamente la **etiqueta predeterminada** y, por consiguiente, se guardará en la dirección predeterminada.

Figura 1.3: Comprobación por colores de la existencia de una etiqueta en la base de datos.

Dado el funcionamiento de las etiquetas en el editor merida, se recomienda siempre tener una dirección donde guardar. En realidad, la etiqueta predeterminada solo es un recurso de seguridad, donde si, por casualidad, no inserta una etiqueta propia, para no perder los datos, se guarda en la dirección predeterminada.

1.4.3. Guardado y Estructura de Archivos

El **Proyecto MERIDA** posee su propia estructura para guardar los documentos generados con el programa. Esta estructura ha sido pensada para ser lo más simple posible, además de «entenderse a la primera». Como tal, el proyecto guarda todo en un archivo .zip, cuya extensión es cambiada por .merida. Sin embargo, al final, sigue siendo un archivo .zip. En un futuro se prevé mejorar el sistema de guardado, es decir, su estructura. Por el momento, no obstante, se presenta la siguiente estructura dentro del módulo del editor merida, ya que aún no se sabe como funcionará este sistema en otros módulos. Por estas razones es probable que la estructura de archivos para el guardado de texto se modifique y cambie de ubicación. Tal vez, sea presentada en la siguiente versión de este manual afuera de la parte de **módulos** del proyecto.

Aclarados estos puntos, comencemos con la explicación. El editor utiliza la creación de archivos ZIP para guardar el contenido ingresado por el usuario y, posteriormente, cambiar la extensión a .merida. El cambio de extensión se justifica en dos simples razones, asegurar que el programa solo abra archivos con esa extensión y distinguir de mejor manera los archivos de guardado dentro de un posible entorno con múltiples tipos de archivos. Ahora bien, dentro de estos archivos existe una estructura que permite al programa cargar la información correctamente.

En primer lugar, antes de entrar en la explicación detallada de la estructura mencionada, hay que explicar el como guarda el programa los datos y cuando. Cuando se inicia con un nuevo documento, el programa comienza a supervisar cualquier cambio hecho en cualquier campo de entrada.Si se realiza algún cambio, la **etiqueta de estado** notifica al usuario con el siguiente mensaje:

Documento modificado. No guardado.

Esta etiqueta señala que, obviamente, ha habido algún cambio detectado por el programa. Si el usuario intenta salir del programa con «cambios no guardados», se le pedirá que confirme dicha acción. Si acepta, es decir, si desea salir con cambios no guardados, el programa se cerrará. Por el contrario, si cancela, volverá a la pantalla principal y deberá guardar su información.

Ahora bien, pasemos a la estructura interna de un archivo .merida. La estructura es sí, dado que es la primera versión del proyecto, es bastante simple. Como el único módulo del proyecto es el editor, tan solo se encontrarán tres archivos dentro del .merida. Estos son los siguientes:

1.contenido.md
2.etiquetas.txt
3.notas.json

Como se puede ver, la estructura es sencilla de entender. El primer archivo es de tipo markdown (md) y contiene el contenido del **Campo del Cuerpo** que el usuario ha ingresado. El segundo archivo es de tipo texto simple (txt) y almacena todas las etiquetas de su correspondiente campo. Y por último, el tercer archivo es de tipo json y almacena todas las notas que se crearon en el documento. Cualquiera puede descomprimir el archivo merida y ver el interior de sus archivos además de modificarlos. Sin embargo, no deben eliminarse, pues el programa esta diseñado para reconocer estos tres archivos, y sus nombres, y cargarlos al editor si existen.

También hay que recalcar que no hay ningún tipo de jerarquía entre los archivos, ya que al programa le interesa únicamente que existan. En un futuro, a medida que el proyecto vaya avanzando y, por ende, se vuelva más complejo, la estructura se irá, probablemente, jerarquizando. Por el momento, esta es toda la estructura y funcionalidad del guardado.

1.5. Características Adicionales

1.5.1. Sistema de Notas (NotasTex)

El editor merida permite la creación de textos cortos cuya intención en «guardar» la idea de forma sencilla y ordenada. La función central radica en la creación de estos textos, sin embargo, el propio editor pone a la disposición del usuario una capa más pequeña de la edición de textos. Es así que nace la particularidad que permite, en palabras generales, «**tomar notas de las notas**».

Ubicada en la barra lateral izquierda, en la pestaña cuyo nombre dicta **Notas**, es donde se pueden crear y editar las conocidas **NotasTex**. Estas notas están pensadas para almacenar datos extras, siendo lo ideal, de longitud corta. Otorgan más flexibilidad al editor central de textos, ya que permiten más opciones de redacción. Además, las notas serán un medio interesante para explotar relaciones que el editor merida tenga con otros módulos (componentes) del **Proyecto MERIDA**.

Para crear las notas tan solo basta con hacer click en el botón **Crear Nota** de su correspondiente pestaña en la barra lateral izquierda. Así se generará una nueva nota y el usuario podrá editarla siguiendo las reglas de tipado descritas para el editor central (Ver Tabla 1.1). Cuando se crea una nueva nota, ya sea mediante el botón o mediante la combinación de teclas establecida, el programa pondrá su «**atención**» en la nueva nota generada. Es decir, el foco estará en la nueva nota. Todo cambio realizado en alguna nota, se marcará como cambio no guardado, por lo que, al igual que los campos de entrada, el programa no le permitirá cerrarlo sin antes confirmar su deseo explícitamente.

Las notas son guardadas en un archivo de tipo json. Y son cargadas en orden junto con el cuerpo del texto y sus etiquetas. Para eliminar una nota, simplemente presione sobre el botón en la esquina superior del **contorno de la nota**. Al presionarlo, la nota se eliminará. Hay que advertir que las notas no se eliminaran si tienen **cambios no guardados** detectados por el programa. Si no tienen cambios no guardados, el programa te dejara eliminar la nota sin confirmación previa.

Como se puede ver, las notas son una característica opcional, pues se puede trabajar perfectamente sin ellas. Sin embargo, si se usan, ofrecen un mejor control de datos e información que no sea estrictamente necesaria para el texto central creado por el editor. En un futuro, las **notastex** tomarán un papel más activo, pues serán usadas para la conexión, en algunos casos, con otros componentes.

1.5.2. Conexiones Previstas con Futuras Herramientas

El editor aún se encuentra, en el momento en que son escritas estas lineas, en una fase temprana de su desarrollo. Desde luego, funciona para su objetivo principal, pero es razonable creer que con el tiempo aumentará su complejidad y utilidad. Por ello, se prevé distintas conexiones con los demás módulos del proyecto. Sin ir más lejos, las **notastex** tienen pensado establecer cierta comunicación de datos con otros elementos del proyecto. La siguiente versión del **Proyecto MERIDA** tiene planeado incorporar la linea de tiempo al programa.

La linea de tiempo ayudará a los creadores a establecer puntos cronológicos de sus obras, de tal manera que dispongan de una vista exacta de los acontecimientos ocurridos. Las notastex, muy probablemente, otorguen la facilidad de anotación para dichos eventos históricos. Todo inicia con el editor, y se va expandiendo más y más. Un punto principal del **Proyecto MERIDA** es, justamente, la creación de vínculos entre las distintas partes que lo componen. En lo ideal, todo componente del proyecto debe estar vinculado con el resto, de tal forma que todo junto responda a lo mismo, ayudar a crear.

Parte III

Anexos

Apéndice A

Preguntas Frecuentes (FAQ)

A.1. ¿Qué es un archivo .merida?

La extensión *.merida* representa a los archivos predeterminados del proyecto MERIDA. Mediante estos archivos, el programa es capaz de cargar y guardar la información creada por los módulos del proyecto. Por otro lado, cabe decir que el archivo .merida es en realidad nada más que un archivo comprimido al que se le ha cambiado la extensión. Según nuestras pruebas, es posible abrir los archivos .merida con cualquier programa que descomprima este tipo de archivos: RAR, ZIP, etc.

A.2. ¿Cómo se crean las etiquetas?

La creación de etiquetas ha sido detallada en la sección **1.4.2. Sistema de Etiquetas**, propia del capítulo 1 de la segunda parte de esta guía de usuario. Sin embargo, para explicarlo rápidamente, las etiquetas se crean usando el siguiente formato: [_**Etiqueta**_]. Deben ser *escritas* en su correspondiente campo de entrada.

A.3. ¿Qué pasa si cierro sin guardar?

En dado caso, el programa debería haber sido capaz de detener el cierre. El programa tiene una función que le impide cerrarse si detecta **Documento modificado. No guardado**. En tal caso, aparecerá una ventana emergente que le preguntará si desea continuar aún habiendo cambios sin guardar.

A.4. ¿Puedo abrir archivos .merida en otro programa?

Hablando técnicamente, si. Los archivos *.merida* son simples archivos comprimidos cuya extensión ha sido cambiada para facilitar la identificación. Usted puede abrirlos con cualquier

programa que trabaje con archivos comprimidos. Por otro lado, los **archivos internos** de los .merida, son de conocimiento común, es decir, cualquier programa de computadora cuyo foco sea la escritura, puede abrirlos. De cualquier manera, siempre es recomendable realizar una **copia de seguridad** regularmente.

A.5. ¿Qué hacer si no se reconoce una etiqueta?

En cuyo caso el programa sea incapaz de reconocer una etiqueta, es altamente recomendable revisar si está correctamente escrita, además de correctamente registrada en la base de datos que el programa usa para las etiquetas.

Apéndice B

Solución de problemas comunes

B.1. MERIDA no inicia

Verifique que todas las dependencias estén instaladas en su sistema. Puede comprobarlo con:

```
sudo apt install python3 python3-gi python3-cairo gir1.2-gtk-4.0
```

Si falta alguna dependencia, instálela y vuelva a intentar iniciar el programa. Recuerde, así mismo, las dependencias mencionadas en la sección **1.3. Público objetivo y requisitos mínimos** presentada en este manual.

Así mismo, si no puede determinar el error o problema que impide a MERIDA iniciarse, pruebe ejecutando el programa desde la termina y desde el archivo ejecutable:

1. Vaya a la dirección donde esta el archivo ejecutable de merida: /usr/bin/[1]
2. Busque el archivo **merida**
3. En la terminal, intente ejecutar dicho archivo con el siguiente comando: ./merida
4. Debería ver un mensaje de error al intentar ejecutar merida en la pantalla de la terminal que explicaría la razón de su no ejecución.

B.2. Error: No se encontraron las carpetas de MERIDA

Si al iniciar aparece un mensaje de error relacionado con carpetas faltantes, revise en primer lugar el log de inicio de MERIDA:

1. Los logs de inicio de ubican en la siguiente ruta: **/.local/state/MERIDA/logs/startup/** o, también, en **/.cache/MERIDA/logs/**

[1]Advertencia: La ruta: /usr/bin es una ruta protegida del sistema. No modifique ni elimine nada sin conocimiento suficiente para hacerlo. Tenga en cuenta que solo intentamos «correr» el programa MERIDA.

2. Una vez dentro de la carpeta de los logs, revise el archivo que tengan, en su nombre, la fecha de creación más reciente.

3. Dentro del archivo encontrará una serie de «registros» que le indicarán si las carpetas correspondientes pudieron crearse o no. De forma predeterminada, MERIDA revisa la disponibilidad de cada carpeta necesaria en cada inicio del programa; si no existe dicha carpeta, MERIDA intenta crearla automáticamente. Si hay problemas relacionados a este tema, se deben encontrar en los logs de inicio.

B.3. Bases de datos dañadas o vacías

Use la opción «Reconstrucción de bases de datos» en la sección de preferencias. Esto eliminará las bases de datos corruptas y las recreará con los valores predeterminados. Vaya a: 1.Administrar, 2.Preferencias, 3.Bases de datos, 4.Operaciones Grupales de bases de datos y 5.Iniciar la reconstrucción de las bases de datos (previa búsqueda).

También puede optar por reconstruir una sola base de datos desde la ventana de información individual de cada base de datos. Para ello vaya a: 1.Administrar, 2.Preferencias, 3.Bases de datos, 4.Click derecho en la base de datos de su preferencia (en la tabla mostrada) y 5.Reconstruir base de datos.

Ahora bien, si el error aún persiste, pruebe a eliminar directamente las bases de datos (haga un respaldo antes si corresponde). Para ello busqué la opción de «Eliminar base de datos» en la ventana de información individual mencionada anteriormente. Luego, entre las dos opciones que se muestran, seleccione «Eliminar Base de Datos». Esta opción eliminará el archivo del sistema operativo. Luego puede restaurarlo desde su copia de seguridad o, si la base de datos estaba dañada, simplemente vuelva a iniciar el programa. MERIDA debería ser capaz de recrear la base de datos si detecta que no existe en su carpeta predetermina (/.local/share/MERIDA/BDs/).

B.4. Problemas con respaldos automáticos

Si el sistema no está creando respaldos automáticos:

- Revise que el intervalo de respaldo esté configurado correctamente. Para ello puede ir a la sección de Preferencias del programa, sección Ajustes Generales, y revisar si sus datos aparecen en la sección de Configurar Respaldo Automático. Si no, vuelva a configurarlos y guárdelos. Debería poder visualizar la ventana de confirmación de registro de dichos datos(Ver Figura 3.1). Opcionalmente, puede ir a revisar directamente dicho archivo en: /.config/MERIDA/Archivos_Recurrentes/Archivo_Recurrente_Spinn.json
- Así mismo, compruebe los permisos de escritura en la carpeta: `/.local/share/MERIDA/BDs/Resgua`

B.5. El fondo de pantalla no se carga

Asegúrese de que la imagen esté en formato `.png` o `.jpg`, y que no haya sido movida o eliminada. Si el problema persiste, seleccione un nuevo fondo desde Ajustes de Estilo.

Apéndice C

Contribuciones y contacto

El proyecto MERIDA es de código abierto y está diseñado para crecer gracias a la colaboración de la comunidad.

C.0.1. Equipo de Desarrollo Actual

Actualmente el desarrollo de MERIDA está a cargo de:

- **Grupo Meridian** – Equipo principal de desarrollo y documentación.
- **DGonzak** – Desarrollador principal y autor de la versión inicial del manual de usuario, así como único miembro actual del Grupo Meridian y, claro, creador de MERIDA.

C.0.2. Contacto

Si deseas ponerte en contacto con el equipo de desarrollo, contribuir al código o reportar errores, puedes utilizar los siguientes medios oficiales[1]:

- **Correo electrónico principal:** dgonzak4@gmail.com
- **Repositorio oficial:** `https://github.com/DGonzak/Proyecto_MERIDA`
- **Reportes de errores y solicitudes:** Se recomienda utilizar el sistema de *issues* del repositorio para reportar errores o sugerir nuevas funciones.

C.0.3. Cómo Contribuir

Las contribuciones son bienvenidas. Para participar, sigue los siguientes pasos:

[1]Por el momento se usa el correo propio de **DGonzak**. Se espera que un futuro próximo, se cree un correo propio para el grupo meridian.

1. Haz un *fork* del proyecto y crea tu propia rama de desarrollo.
2. Documenta claramente los cambios realizados antes de enviar un *pull request*.
3. Si es posible, comunícate antes con algún miembro del grupo Meridian.

C.0.4. Agradecimientos

El equipo de MERIDA agradece a todos los colaboradores presentes y futuros que ayuden a mejorar este proyecto. Como muestra de afecto para con los colaboradores, se ofrece el siguiente espacio en la guía oficial de MERIDA.

Apéndice D

Licencia y Uso de Código

D.0.1. Licencia

El proyecto MERIDA se distribuye bajo la licencia **GNU General Public License v3 (GPLv3)**. Esta licencia garantiza que:

- El código fuente del programa estará siempre disponible para sus usuarios.
- Cualquier modificación o trabajo derivado debe mantenerse libre y abierto.
- Los usuarios pueden usar, copiar, distribuir y modificar el programa sin coste.

El texto completo de la licencia se incluye a continuación:

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.
```

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major

Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

```
    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
```

that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and

adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on

those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or

run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and

propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY

OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

```
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License.  Of course, your program’s commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
```

D.0.2. Uso de Código

El uso del código de MERIDA está permitido para:

- **Estudio y aprendizaje**: los usuarios pueden analizar el código para comprender su funcionamiento.
- **Modificación y mejora**: cualquier persona puede modificar el programa para adaptarlo a sus necesidades.
- **Redistribución**: se permite compartir el programa modificado o sin modificar, siempre que se conserve la misma licencia (GPLv3) y se cite la autoría original.

Nota importante: cualquier versión modificada que se distribuya públicamente debe incluir de forma clara la información de los autores originales, así como un registro de los cambios realizados.