Desarrollo de un Sistema de Automatización de Informes con AppSheet y Python

Juan Sebastián Martínez Cárdenas

Proyecto de Grado presentado en cumplimiento del requisito para optar por el grado de Ingeniero Electrónico

Dirigido por

Ing. PhD(c) Adolfo Ávila Barón

Noviembre de 2024

Facultad de Ingeniería Electrónica, División de Arquitectura e Ingenierías

Universidad Santo Tomas

ADVERTENCIA

La Universidad Santo Tomás no se hace responsable de las opiniones y conceptos expresados en el trabajo de grado, solo velará por qué no se publique nada contrario al dogma ni a la moral católica y porque el trabajo no tenga ataques personales y únicamente se vea el anhelo de buscar la verdad científica.

Capítulo III –Art. 46 del Reglamento de la Universidad Santo Tomás.

El trabajo de grado titulado Desarrollo de un Sistema de Automatización de Informes con AppSheet y Python, realizado por el estudiante Juan Sebastián Martínez Cárdenas, cumple con los requisitos exigidos por la Universidad Santo Tomás para optar al título de Ingeniero Electrónico.

Ing. PhD(c) Adolfo Ávila Barón

Docente Director de Trabajo de Grado

Contenido

Tabla de Figuras

Lista de tablas

1. Introducción

En el ámbito del mantenimiento de equipos biomédicos, la recopilación y gestión de datos ha sido tradicionalmente un proceso manual que depende en gran medida del uso de papel, lo cual no solo resulta ineficiente, sino que también genera un impacto ambiental considerable. En un contexto donde la digitalización y la sostenibilidad son prioridades, se vuelve imprescindible modernizar estos procesos hacia soluciones más eficientes y respetuosas con el medio ambiente. Este proyecto tiene como objetivo el desarrollo de un sistema integral que sustituya la gestión manual de datos por una plataforma digital, optimizando el proceso de mantenimiento de equipos biomédicos y eliminando la necesidad de utilizar papel, lo que permitirá una gestión más ágil, precisa y sostenible.

Para llevar a cabo esta transformación, se propuso la implementación de dos tecnologías clave: AppSheet, una plataforma sin código para el desarrollo de aplicaciones móviles, y Python, un lenguaje de programación versátil que facilitará la automatización de tareas. A través de la combinación de estas herramientas, se creó una aplicación móvil personalizada que permitirá a los técnicos de mantenimiento recopilar datos de manera eficiente y en tiempo real, mientras que los scripts en Python se encargarán de automatizar la generación de informes a partir de los datos almacenados en una base de datos en Excel. Este sistema no solo incrementa la precisión y eficiencia de los procesos de mantenimiento, sino que contribuirá significativamente a la reducción del uso de papel y al impacto ambiental asociado con las operaciones biomédicas. En consecuencia, el proyecto no solo responde a las necesidades operativas de las instituciones de salud, sino que también se alinea con los principios de sostenibilidad y responsabilidad ambiental que rigen la innovación tecnológica en la actualidad.

2. Planteamiento del problema

Actualmente, en la empresa, la gestión de la documentación de informes se realiza en formato físico, lo que presenta varios inconvenientes significativos. Entre ellos se incluyen el riesgo de daño y pérdida de información crítica para la institución, así como el acceso limitado a los documentos, lo que dificulta su consulta por parte del personal interesado. Este proceso requiere la búsqueda manual de los documentos y la presencia física para acceder a ellos, lo cual resulta en una tarea engorrosa y poco eficiente.

Otro aspecto relevante es que la falta de innovación tecnológica puede llevar a la empresa a estancarse en términos de productividad, impidiendo el desarrollo de áreas clave para su mejora continua. Esto limita el crecimiento y la capacidad de adaptación al contexto actual, en el que la demanda de procesos más rápidos y eficientes es cada vez mayor.

La falta de incorporación de innovaciones tecnológicas puede obstaculizar la competitividad de la empresa frente a otras entidades que ofrecen servicios similares. Sin estas herramientas, se dificultan la resolución de problemas inherentes a las operaciones diarias, lo que limita la eficiencia y productividad. Además, la ausencia de tecnología adecuada impide que los empleados puedan mejorar su desempeño, afectando negativamente la calidad del trabajo realizado y el rendimiento general de la empresa.

A partir de los problemas mencionados, surge la siguiente interrogante: ¿Cómo se podría diseñar un sistema que automatice la gestión de informes y de ser así como el sistema puede resolver las deficiencias identificadas?

3. Justificación

La empresa enfrenta desafíos significativos en su proceso actual de recopilación y entrega de informes de mantenimiento de equipos biomédicos, que se caracteriza por su enfoque manual y dependencia del papel. La digitalización de procesos es crucial para optimizar el uso de los recursos y el tiempo en la empresa, ya que reduce la ineficiencia asociada al manejo manual de documentos. Al adoptar tecnologías digitales, se disminuye significativamente el riesgo de errores en la documentación y la gestión de datos, mejorando la precisión y la fiabilidad de la información. Además, la digitalización permite a la empresa adaptarse rápidamente a cambios del mercado, elimina costos innecesarios relacionados con la impresión y el almacenamiento físico, y facilita un análisis más preciso y oportuno de los datos, lo que contribuye a una toma de decisiones más informada.

De acuerdo con Lozano Sequera y Reina Rodríguez (2020), las tecnologías de la información y las comunicaciones (TIC) se están consolidando como factores clave en la transformación de los sectores económicos. Elementos como los servicios en la nube, los nuevos medios de pago, los gestores financieros automatizados y los modelos de negocio innovadores están impulsando cambios significativos. En este contexto, las empresas que no han implementado procesos ágiles y eficientes de innovación enfrentan un creciente riesgo de obsolescencia.

En este sentido, la ingeniería electrónica y la programación ofrecen una solución integral al permitir la creación de una aplicación móvil personalizada para la recopilación de datos en campo, y la automatización de la generación de informes mediante scripts en Python. Esta transformación no solo optimizará la eficiencia operativa y reducirá costos a largo plazo, sino que también garantizará la precisión de los datos y facilitará una gestión más efectiva de los recursos. La centralización de los datos en una base de datos permitirá, además, un acceso más rápido, un almacenamiento más seguro y un análisis más eficiente de la información, lo que impulsará la productividad y competitividad de la empresa en el mercado.

4. Impacto social

La adopción de una solución digital para la gestión del mantenimiento de equipos biomédicos no solo repercute positivamente en la eficiencia operativa y la sostenibilidad ambiental, sino que también tiene un impacto social significativo. Al eliminar la dependencia del papel en la recopilación de datos, se reduce la carga administrativa sobre el personal médico y técnico, lo que les permite liberar tiempo y recursos para enfocarse en actividades de mayor valor, como la atención al paciente y la mejora de los servicios de salud.

Además, al agilizar y optimizar el acceso a la información sobre el estado de los equipos biomédicos, se fortalece la capacidad de los profesionales de la salud para tomar decisiones informadas y oportunas. Esto, a su vez, puede traducirse en una atención médica más eficiente y segura para la comunidad. En última instancia, la implementación de esta solución digital no solo transformará los procesos internos de gestión en el ámbito biomédico, sino que también contribuirá a mejorar la calidad y accesibilidad de los servicios de salud, beneficiando de manera directa a la sociedad en su conjunto.

5. Objetivos

5.1.Objetivo general

Desarrollar un sistema encargado de la gestión de información del mantenimiento de equipos biomédicos, empleando AppSheet para la recopilación de datos y Python para la generación automatizada de informes.

5.2.Objetivos específicos

Diseñar una aplicación móvil en AppSheet que recopile la información relacionada con las inspecciones, reparaciones y mantenimiento preventivo de equipos biomédicos.

Integrar la aplicación móvil con una base de datos en Excel para almacenar los datos recopilados de manera centralizada y accesible, garantizando la integridad y seguridad de la información.

Implementar scripts de Python para procesar los datos almacenados en la base de datos de Excel, para automatizar la generación de informes detallando el estado y la historia de mantenimiento de los equipos biomédicos.

6. Estado del arte

El desarrollo de sistemas de automatización de informes ha adquirido gran relevancia en diversas áreas profesionales, ya que permite optimizar procesos, reducir el margen de error humano y mejorar la eficiencia operativa. En este contexto, el presente proyecto se centra en el Desarrollo de un Sistema de Automatización de Informes con AppSheet y Python, con el objetivo de facilitar la generación, gestión y distribución de informes de manera automatizada y personalizada.

Este estado del arte tiene como propósito revisar las principales investigaciones y tecnologías relacionadas con la automatización de informes, específicamente aquellas que emplean herramientas como AppSheet y Python. A través de la revisión de estudios previos, se explorarán las metodologías empleadas en el desarrollo de sistemas similares, así como las ventajas y limitaciones de las herramientas seleccionadas. Además, se examinarán casos de uso y aplicaciones existentes que combinan estas tecnologías, con el fin de identificar las mejores prácticas y áreas de mejora que podrían aplicarse al proyecto. Este análisis no solo permitirá contextualizar el desarrollo del sistema en el panorama actual de la automatización, sino que también brindará innovaciones que resultan clave para la implementación de una solución eficiente y escalable a una problemática que se presente.

6.1. Desarrollo de un marco de diseño integrado utilizando scripts en Python para el modelado CAD paramétrico de acoplamientos de bridas:

Este artículo presenta una integración de AutoCAD con Python para optimizar el modelado CAD de una brida de acoplamiento. Las principales contribuciones incluyen el desarrollo de un sistema de diseño basado en conocimiento (KBE) y una interfaz gráfica de usuario (GUI) para facilitar el diseño paramétrico. Este enfoque mejora la personalización, reduce redundancias y satisface las necesidades específicas de diferentes máquinas con parámetros de acoplamiento de bridas. A través del uso de la API de AutoCAD, se genera un diseño eficiente y se crean dibujos 2D de fabricación. El estudio demuestra cómo el conocimiento aplicado (se refiere al aprendizaje de tipo maquina) puede impulsar el desarrollo de diseños personalizados.

Los investigadores concluyen que el desarrollado con la API Python-AutoCAD permite la parametrización y modificación del diseño de componentes con una interfaz gráfica de usuario dedicada (GUI), lo que resulta en una gama de diseños de acoplamiento de bridas basadas las técnicas de parametrización y los fundamentos diseño. Otra ventaja de esta síntesis de softwares permite que el tiempo de procesamiento necesario para la entrada GUI para el diseño personalizado de la brida y generación de dibujos 2D es aproximadamente 3 s para un 2.4 GHz lo que le otorga a el sistema informático una altamente eficiente para aplicaciones industriales. (Patil, K.R., Deshpande, S., Joshi, S, 2024)

6.2.Optimización del sistema de notificación y supervisión para el mantenimiento del aire acondicionado en la sks del sitio pt ppa

Esta investigación analiza el sistema actual de informes y monitoreo de mantenimiento de aire acondicionado (CA) en la SKS del sitio PPA de PT. Utilizando un enfoque cualitativo con un estudio de caso, se recopilaron datos mediante observación y revisión documental. El análisis cualitativo se realizó con el método de teoría fundamentada. Los resultados muestran que la optimización del sistema se logra mediante la creación de programas periódicos de inspección y mantenimiento, la asignación de códigos QR a las unidades de CA para facilitar la entrada de informes, y el uso de la aplicación AppSheet para registrar y supervisar los informes de mantenimiento. Además, se nombró un encargado del equipo eléctrico para supervisar el mantenimiento del CA y se realizó una capacitación para los administradores sobre los mecanismos de informes. Como resultado, el mantenimiento del CA puede ser monitoreado al 100%, y los reportes son más efectivos sin necesidad de intervención del administrador en cada entrada

Al finalizar el documento, los investigadores concluyen que la programación de inspecciones regulares, la asignación de códigos QR a las unidades de CA y el uso de la aplicación AppSheet facilitan la presentación de informes en tiempo real. Para corroborar estos resultados, se realizó una prueba posterior del mantenimiento del CA, que demostró que puede ser monitoreado al 100%, lo que reduce el uso de papel y mejora la eficiencia y efectividad de los informes, eliminando la necesidad de intervención administrativa en cada entrada. (NUGROHO, P. S, 2024).

6.3. Mejorar la eficiencia de la gestión de admisiones de estudiantes a través de la transformación digital: un estudio de caso utilizando una plataforma de desarrollo sin código.

Este estudio evaluó el impacto de las plataformas sin código, específicamente AppSheet, en la eficiencia de procesos complejos, centrado en el proceso de admisión universitaria. Al comparar los datos antes y después de la adopción de la plataforma, se observó una reducción significativa en el tiempo de presentación, procesamiento y los recursos humanos necesarios. La disminución en el personal y el tiempo de tramitación resalta el potencial de las plataformas sin código para mejorar la eficiencia y reducir costos.

Los investigadores concluyen que la adopción de plataformas sin código como AppSheet puede mejorar significativamente la eficiencia de procesos complejos, como el proceso de admisión universitaria. El estudio demostró que el uso de AppSheet redujo el tiempo de presentación, procesamiento y los recursos humanos necesarios para gestionar las admisiones, lo que resalta su capacidad para agilizar flujos de trabajo, automatizar tareas repetitivas y reducir costos. Sin embargo, el estudio tiene limitaciones, como un período de recopilación de datos corto y métricas limitadas. Se recomienda realizar más investigaciones para evaluar completamente la eficacia de estas plataformas y explorar posibles problemas éticos o legales asociados. (Do Dinh-Cuong; Nguyen The-Vinh; Bui Anh-tu, 2024)

6.4.Diseño e Implementación Móvil para Gestión de Solicitudes de Repuestos en Equipos Biomédicos y Registro de Dificultades Técnicas.

Este proyecto se enfoca en el desarrollo de una aplicación móvil para la gestión eficiente de solicitudes relacionadas con problemas técnicos o repuestos de equipos biomédicos. La aplicación fue desarrollada utilizando AppSheet y cuenta con una interfaz gráfica intuitiva que permite al usuario seleccionar el tipo de solicitud que desea realizar. Al seleccionar una opción, se activa un formulario que recopila toda la información necesaria para procesar la solicitud de manera eficaz. Posteriormente, la solicitud se guarda en una base de datos para su revisión posterior y, además, se envía una notificación al administrador para informar sobre la nueva solicitud ingresada.

El autor concluye que la aplicación facilita una gestión eficiente de las solicitudes y problemas técnicos, gracias a su interfaz fácil de usar y sus funciones específicas, que optimizan el flujo de trabajo. Asimismo, las notificaciones automáticas permiten un seguimiento oportuno, reduciendo los tiempos de respuesta y eliminando la dependencia de métodos de comunicación tradicionales, como llamadas telefónicas o mensajes de texto. (Claros Luis, 2024)

7. Marco teórico

7.1.Digitalización en el Mantenimiento de Equipos Biomédicos

La digitalización ha revolucionado numerosos aspectos de la industria, incluido el mantenimiento de equipos biomédicos. Tradicionalmente, este proceso se ha llevado a cabo utilizando formularios y documentación en papel, lo que puede resultar en ineficiencias operativas, errores de registro y un consumo excesivo de recursos. La transición hacia sistemas digitales ofrece una solución efectiva para abordar estos desafíos, permitiendo una recopilación de datos más eficiente, un seguimiento más preciso del estado de los equipos y una generación automatizada de informes. Esta transformación digital no solo mejora la eficiencia operativa, sino que también contribuye a la sostenibilidad ambiental al reducir el uso de papel y otros recursos. (Estrada Porras, 2017).

7.2.AppSheet como Plataforma para el Desarrollo de Aplicaciones Móviles:

AppSheet es una plataforma de desarrollo de aplicaciones móviles que permite a los usuarios crear aplicaciones personalizadas sin necesidad de conocimientos de programación. Utilizando una interfaz intuitiva basada en hojas de cálculo, los usuarios pueden diseñar y desplegar aplicaciones móviles que se adapten a sus necesidades específicas. En el contexto del mantenimiento de equipos biomédicos, AppSheet ofrece una solución ideal para la recopilación de datos en el campo. Los técnicos de mantenimiento pueden utilizar aplicaciones diseñadas en AppSheet para registrar datos de inspección, detalles de mantenimiento y otros registros relevantes de manera rápida y precisa, todo desde sus dispositivos móviles. (Petrovic, Radenkovic & Nejkovic, 2020).

7.3.Programación en Python para Automatización de Procesos:

Python es un lenguaje de programación versátil y poderoso que se utiliza ampliamente en la automatización de procesos, el análisis de datos y el desarrollo de aplicaciones. En el contexto del mantenimiento de equipos biomédicos, Python puede desempeñar un papel crucial en la generación automatizada de informes y el análisis de datos recopilados. Mediante el desarrollo de scripts y algoritmos personalizados, es posible procesar grandes cantidades de datos de mantenimiento, identificar patrones y tendencias, y generar informes detallados que faciliten la toma de decisiones informadas por parte de los técnicos y supervisores de mantenimiento. (Martel Solis, 2021).

7.4.Beneficios de la Integración entre AppSheet y Python:

La integración entre AppSheet y Python ofrece una solución completa para el mantenimiento de equipos biomédicos, combinando la facilidad de uso de la plataforma de desarrollo de aplicaciones móviles con la potencia y flexibilidad de la programación en Python. (Nejkovíc, Radenkovic & Petrovic, 2021) Al utilizar AppSheet para la recopilación de datos en el campo y Python para el análisis y la generación de informes automatizados, los usuarios pueden beneficiarse de un proceso de mantenimiento más eficiente, preciso y ambientalmente responsable. (Martel Solis, 2021) Esta integración permite a las organizaciones de atención médica optimizar sus operaciones de mantenimiento, garantizando el funcionamiento seguro y confiable de sus equipos biomédicos.

Plataformas de Desarrollo de Aplicaciones No Codificadas/AppSheet: Las plataformas de desarrollo de aplicaciones no codificadas, como AppSheet, han ganado popularidad debido a su capacidad para permitir a usuarios sin experiencia en programación crear aplicaciones móviles de manera rápida y sencilla. Según lo señalado por Chen et al. (2019), "estas plataformas utilizan interfaces visuales y herramientas de arrastrar y soltar para diseñar y personalizar aplicaciones, lo que las hace accesibles incluso para aquellos sin conocimientos técnicos avanzados" (p. 112).

Tabla 1. *Comparación con otros lenguajes de programación.*

Aspecto	*Python*	*C++*	*Java*
Sintaxis	Concisa y fácil de leer	Más compleja y detallada	Similar a C++, estructurada y clara
Flexibilidad	Altamente flexible debido a la naturaleza dinámica	Menos flexible debido a la tipificación estática	Moderadamente flexible
Aplicaciones embebidas	Utilizado en aplicaciones embebidas debido a su ligereza y flexibilidad	Utilizado en sistemas embebidos de alto rendimiento	Utilizado en sistemas embebidos con soporte para Java
Documentación	Documentación clara y extensa, con numerosos tutoriales y ejemplos	Buena documentación, aunque puede ser más técnica debido a su complejidad	Documentación exhaustiva y detallada y para grandes proyectos
Comunidad y soporte	Gran comunidad de desarrolladores y amplio soporte en línea	Comunidad robusta y amplio soporte en línea	Amplia comunidad y soporte
Desarrollo móvil	Popular en el desarrollo de aplicaciones móviles con frameworks	Utilizado en el desarrollo móvil, especialmente en juegos y aplicaciones de alto rendimiento	Fuerte en desarrollo móvil a través de Android SDK

Basándonos en los aspectos anteriormente discutidos, Se ha decidido utilizar Python para la automatización de los informes debido a su gran versatilidad y facilidad de uso. Python ofrece una combinación única de capacidad para manejar diversas tareas, desde el procesamiento de datos hasta la generación de informes, todo ello con una sintaxis clara y legible. Además, su amplia comunidad y la disponibilidad de numerosas bibliotecas y frameworks facilitan aún más el desarrollo ágil y eficiente de soluciones automatizadas.

7.5.Base de datos:

Una base de datos es un conjunto de datos almacenados en memoria externa, organizados mediante una estructura diseñada para satisfacer los requisitos de información de una empresa u organización, como una universidad o un hospital. Una base de datos se puede percibir como un gran almacén de datos que se define y se crea una sola vez, y que se utiliza al mismo tiempo por distintos usuarios. En una base de datos todos los datos se integran con una mínima cantidad de duplicidad. De este modo, la base de datos no pertenece a una sola área concreta, sino que se comparte por toda la organización. Además, la base de datos no sólo contiene los datos de la organización, también almacena una descripción de dichos datos. (Marqués, M. ,2009).

7.6.Bases de datos relacionales

Una base de datos relacional es un conjunto de una o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por campos en común que poseen las mismas características en ambas tablas, como por ejemplo el nombre de campo, tipo y longitud. A estos campos generalmente se les denomina campos identificadores (ID) o campos clave. A esta forma de construir bases de datos se le denomina modelo relacional.

Toda base de datos relacional está formada por uno o varios bloques de información llamados TABLAS (inicialmente denominados ficheros o archivos) que normalmente tendrán alguna característica en común. Una tabla o archivo de datos es un conjunto conexo de información del mismo tipo, por ejemplo, en una base de datos de una biblioteca, una tabla estará constituida por la información relativa a todos los libros de la misma, otra tabla contendrá información sobre los lectores, etc. Cada tabla está formada por registros. Un registro es la unidad elemental de información de la tabla o fichero. (Valderrey Sanz, P.,2015).

7.7.Bases de datos de modelos orientado a objetos

La orientación a objetos ofrece una mayor flexibilidad para abordar diversos requisitos, superando las limitaciones impuestas por los tipos de datos y los lenguajes de consulta de los sistemas de bases de datos tradicionales. Una característica clave de las bases de datos orientadas a objetos es la capacidad que brindan al diseñador para especificar tanto la estructura de objetos complejos como las operaciones que se pueden aplicar sobre estos objetos. Además, el creciente uso de lenguajes orientados a objetos en el desarrollo de aplicaciones ha sido otro factor importante para la creación de las bases de datos orientadas a objetos. (Valderrey Sanz, P.,2015).

7.8.Lenguajes de bases de datos

SQL (Structured Query Language): Es el lenguaje más comúnmente utilizado para gestionar bases de datos relacionales. Permite realizar consultas, actualizar registros, insertar y eliminar datos, y administrar la estructura de la base de datos. SQL es estándar en la mayoría de los sistemas de bases de datos como MySQL, PostgreSQL, SQL Server y Oracle.

PL/SQL (Procedural Language/SQL): Es una extensión de SQL utilizada por Oracle Database. Combina el poder de SQL con la programación estructurada y permite el uso de procedimientos, funciones, variables, excepciones y control de flujo, lo que le da mayor flexibilidad que SQL solo.

T-SQL (Transact-SQL): Es una extensión de SQL utilizada en Microsoft SQL Server. Además de las capacidades de SQL estándar, T-SQL incluye características adicionales de programación como variables, manejo de excepciones, bucles y lógica condicional, lo que permite crear procedimientos almacenados y funciones complejas.

NoSQL (Not Only SQL): Es un conjunto de tecnologías de bases de datos no relacionales que se utilizan para manejar grandes volúmenes de datos no estructurados o semi-estructurados. Existen varios tipos de bases de datos NoSQL, como bases de datos clave-valor, de documentos, de columnas y de grafos. Ejemplos de bases de datos NoSQL incluyen MongoDB, Cassandra, Redis y Neo4j.

Tomando en cuenta la anterior información se presentará una tabla comparativa para evaluar las ventajas y desventajas de usar AppSheet en el desarrollo de la aplicación, en comparación con los lenguajes de bases de datos previamente mencionados.

Tabla 2. *Comparación de bases de datos.*

Criterio	AppSheet	SQL & Derivados (PL/SQL, T-SQL, etc.)
Facilidad de uso	No requiere conocimientos de programación y es adecuado para usuarios sin experiencia técnica. Permite crear aplicaciones y formularios con un enfoque visual e intuitivo.	Los usuarios deben entender SQL y sus variantes (PL/SQL, T-SQL) para escribir consultas, definir estructuras y programar procesos de informes.
Recopilación de datos	AppSheet recopila información automáticamente basados en los datos en tiempo real, con un enfoque en la simplicidad y rapidez.	Los informes deben ser diseñados, creados y gestionados manualmente a través de consultas SQL complejas o procedimientos almacenados. Esto requiere más tiempo y esfuerzo.
Integración con otros sistemas	AppSheet se integra bien con diversas fuentes de datos como Google Sheets, Excel, bases de datos SQL, y otras aplicaciones sin necesidad de mucho código.	Las integraciones suelen requerir conexiones de bases de datos y más configuración manual. Depende del entorno de bases de datos utilizado (por ejemplo, Oracle, SQL Server).
Requerimientos de infraestructura	AppSheet es una plataforma en la nube, lo que significa que no requiere infraestructura propia ni mantenimiento, todo está gestionado en línea.	Las bases de datos SQL requieren servidores propios, administración de bases de datos, backups, y mantenimiento continuo.
Automatización	AppSheet permite la automatización de la generación de informes y otros procesos sin intervención manual continua.	Las consultas de informes y procesos automáticos deben ser creados y gestionados manualmente, lo que requiere conocimientos técnicos.
Mantenimiento	Al ser una plataforma gestionada, la mayor parte del mantenimiento (actualizaciones, gestión de infraestructura) es realizado por el proveedor.	El mantenimiento de bases de datos SQL, procedimientos, integraciones, y actualizaciones de software requiere personal dedicado y esfuerzos continuos.

Tras realizar la comparación y considerando el contexto del proyecto, se puede concluir que el uso de AppSheet es más eficiente. Esto se debe a que no se requiere un sistema tan complejo y robusto como SQL. Además, el objetivo es recopilar los datos de los equipos de manera práctica y rápida, lo que se logra de manera óptima mediante una aplicación móvil.

8. Desarrollo del proyecto

Para el desarrollo del proyecto, se emplean los siguientes dos softwares de diseño que permitirán completar el proyecto de manera efectiva. El primero es AppSheet, que se utilizará para crear una aplicación web que permitirá a los usuarios registrar los datos necesarios para la elaboración de los informes de las ESEs de salud. El segundo software es Python, que se encarga del procesamiento de los datos obtenidos mediante la aplicación. Python facilitará la generación de informes altamente personalizados, mejorando la visualización de los datos y proporcionando a la empresa una herramienta robusta para manejar una amplia variedad de información.

8.1.Hojas de Calculo

Antes de desarrollar la aplicación en AppSheet, es necesario utilizar hojas de cálculo para construir un marco de datos que incluya la información de los equipos de todas las ESEs con las que la empresa tiene un contrato. Además, será crucial extraer las características esenciales para los informes, tales como:

- La entidad de salud (ESE) a la que pertenece el equipo.
- La ubicación del equipo ya sea en una ambulancia o en un área específica de la entidad.
- El tipo de equipo (autoclave, tensiómetro, etc.).
- Una imagen del equipo.
- La marca del equipo.
- El modelo del equipo, incluyendo detalles como el número de serie, el número de inventario, el fabricante, el registro sanitario, la fecha de compra, la fecha de instalación, el vencimiento de la garantía y la clasificación del equipo (biomédico o industrial).

A continuación, se procede a revisar la documentación proporcionada por la empresa a sus clientes en los servicios de mantenimiento y calibración. Se observará que los documentos pueden agruparse según las actividades relacionadas, que son: Mantenimiento Preventivo, Mantenimiento Correctivo y Calibración.

En estas tres áreas se registran diferentes tipos de documentos. Por ejemplo, en el Mantenimiento Preventivo se incluyen los informes de mantenimiento correspondientes a las visitas realizadas a las ESEs y las hojas de vida de los equipos. En el Mantenimiento Correctivo,

se encuentra el reporte de mantenimiento correctivo, que es el documento generado cuando se presenta una falla en un equipo y se documenta su reparación. Finalmente, en la sección de Calibración, se registran los certificados de calibración.

8.1.1. Mantenimiento Preventivo:

En este apartado se recolectan los siguientes datos: imagen de la visita, observaciones, fecha del mantenimiento, nombre del ingeniero que realizó el mantenimiento, firma del encargado de la zona o de quien recibe el mantenimiento, y el número de cédula de dicha persona. Además, se ha identificado la necesidad de crear una base de datos adicional para cada tipo de equipo, con el fin de registrar información específica como si el equipo requiere calibración.

Adicionalmente, es necesario registrar la siguiente información para la hoja de vida del equipo:

Tabla 3. *Parámetros para las hojas de vida.*

Parámetros	
Periodicidad de la calibración	Humedad
Nivel de riesgo del tipo de equipo	Potencia
Clasificación biomédica	Presión
Rango de voltaje	Frecuencia
Rango de corriente	Temperatura
Rango de frecuencia	Peso
Rango de presión	Velocidad
Rango de potencia	Otros parámetros
Rango de peso	Móvil/Fijo
Rango de temperatura	Tiempo de trabajo
Rango de humedad	Protocolos
Otros parámetros relevantes	Recomendaciones del fabricante
Fuente de alimentación	Voltaje máximo y mínimo
Tecnología predominante	Corriente máxima y mínima

Es importante destacar que también se debe crear una lista de chequeo para cada equipo, la cual será gestionada en una base de datos en Excel. Cada equipo tendrá una lista de chequeo personalizada que incluirá los ítems previamente mencionados.

8.1.2 Mantenimiento Correctivo:

Para el mantenimiento correctivo, es necesario recolectar los siguientes ítems:

- Ingeniero que realizó el mantenimiento correctivo: Nombre del profesional encargado.
- Número de reporte: Identificador único del reporte de mantenimiento.
- Fecha del correctivo: Fecha en la que se realizó el mantenimiento correctivo.
- Falla reportada: Descripción del problema o falla del equipo.
- Informe de actividades: Detalle de las actividades realizadas durante el mantenimiento.
- Repuestos utilizados: Lista de repuestos empleados para la reparación.
- Quién recibe el mantenimiento correctivo: Nombre de la persona que recibió el mantenimiento.

8.2.Appsheet:

En el área de diseño de AppSheet, se exportan una variedad de bases de datos que desempeñan un papel crucial en la gestión eficiente de los datos recopilados por la aplicación. Estas bases de datos están cuidadosamente diseñadas para ser utilizadas en los procesos de mantenimiento correctivo, que se centran en la reparación de fallos y problemas inesperados, así como en el mantenimiento preventivo, que busca anticipar y evitar posibles fallos mediante intervenciones programadas. Al contar con estas bases de datos, se facilita una gestión integral y proactiva del mantenimiento, permitiendo una mejor planificación, seguimiento y ejecución de las tareas necesarias para asegurar el buen funcionamiento de los sistemas y equipos.

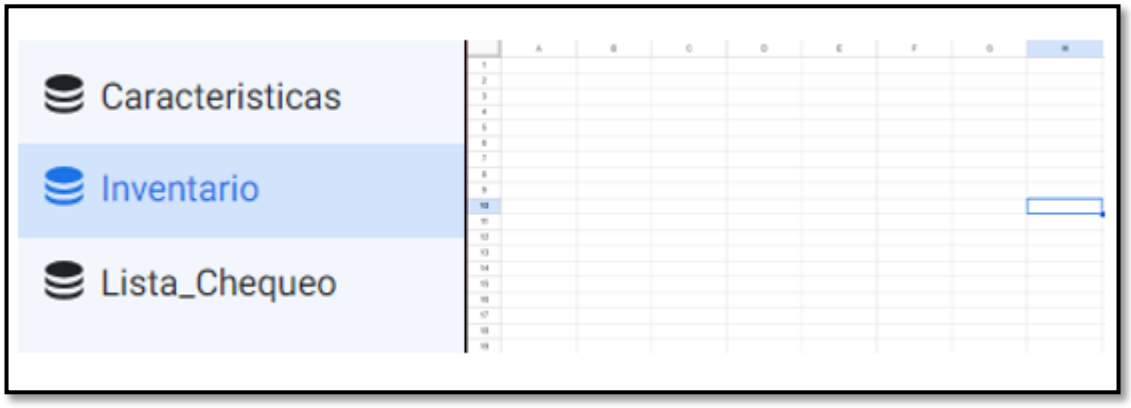


Figura 1. Appsheet (Lista de tablas).

La aplicación se crea a partir de un inventario que sirve como la base de datos principal para el mantenimiento correctivo y preventivo. En este proceso, se definen las columnas y sus respectivos tipos de datos, así como sus funciones específicas. Este enfoque asegura que cada elemento del inventario esté correctamente catalogado y que la aplicación pueda gestionar de manera efectiva tanto las intervenciones correctivas como las preventivas, facilitando así una administración más eficiente y organizada del mantenimiento.

Tabla 4. *Base de datos de Inventario.*

Parámetros	***Tipo de dato***
ID Equipo	Texto
ESE	Lista enumerada
Ubicación	Lista enumerada
Equipo	Lista enumerada
Foto	Imagen
Foto URL	Texto
Marca	Lista enumerada
Modelo	Texto
Serie	Texto

En la base de datos denominada "Inventario", se recopilan diversos datos, como se ilustra en la imagen adjunta. Entre esta información se incluyen detalles sobre el equipo que ha recibido mantenimiento correctivo o preventivo, las actividades realizadas y el ingeniero encargado de dichas tareas. Un aspecto clave de esta base de datos es la capacidad de aplicar filtros específicos en las columnas "VISITA" y "ACTIVIDAD V1, V2, V3, V4". Por ejemplo, al seleccionar "Primera Visita" y "Mantenimiento Preventivo" en la columna correspondiente, el sistema filtra y muestra únicamente los datos relevantes para esa combinación específica de visita y actividad. Esto facilita la visualización y el llenado de datos según los criterios seleccionados.

Tabla 5. *Mantenimiento preventivo.*

Parámetros	***Tipo de dato***
Inventario	Texto
Fabricante	Lista enumerada
Registro Sanitario	Lista enumerada
Fecha de Compra	Lista enumerada
Fecha de Instalación	Imagen
Vencimiento de Garantía	Texto
Tipo	Lista enumerada
Estado	Texto
Visita	Texto
Actividad v1	Enumeración
Actividad v2	Enumeración
Actividad v3	Enumeración
Actividad v4	Enumeración
Visita 1	Imagen
Visita 1 URL	Texto
Visita 2	Imagen
Visita 2 URL	Texto
Visita 3	Imagen
Visita 3 URL	Texto
Visita 4	Imagen
Visita 4 URL	Texto
Observaciones	Texto
Fecha v1	Fecha
Fecha v2	Fecha
Fecha v3	Fecha
Fecha v4	Fecha
Firma v1	Firma
Firma v1 URL	Texto
Firma v2	Firma
Firma v2 URL	Texto
Firma v3	Firma
Firma v3 URL	Texto
Firma v4	Firma
Firma v4 URL	Texto
CC v1	Texto
CC v2	Texto
CC v3	Texto
CC v4	Texto
Firma MTO 1	Lista enumerada
Firma MTO 2	Lista enumerada
Firma MTO 3	Lista enumerada
Firma MTO 4	Lista enumerada

Un aspecto destacado es que los campos que terminan en _Url están configurados mediante la siguiente expresión: “CONCATENAR("https://www.appsheet.com/template/gettablefileurl?appName=Mantenimiento-586870375-24-07-11-2&tableName=formulario&fileName= “, E2)”. Esta expresión genera una URL que apunta a la ubicación donde se almacenará la imagen tomada en la columna anterior. En esencia, al utilizar esta fórmula, se crea un enlace dinámico para cada archivo, facilitando el acceso y la gestión de las imágenes asociadas a las entradas del formulario.

Como se muestra en la imagen siguiente, al rellenar un informe para el mantenimiento preventivo, se debe completar la información requerida por el sistema. Cuando se llega al campo correspondiente a la "Visita" y se selecciona "Primera Visita", el filtro se activa para ajustar la configuración de la información subsiguiente. Esto incluye los campos de "Actividad", "Imagen" y "Observaciones", los cuales se actualizarán automáticamente para reflejar los datos relevantes de la visita seleccionada. Este proceso asegura que solo se muestre la información pertinente para la visita específica, facilitando la precisión y la eficiencia en la cumplimentación del informe.

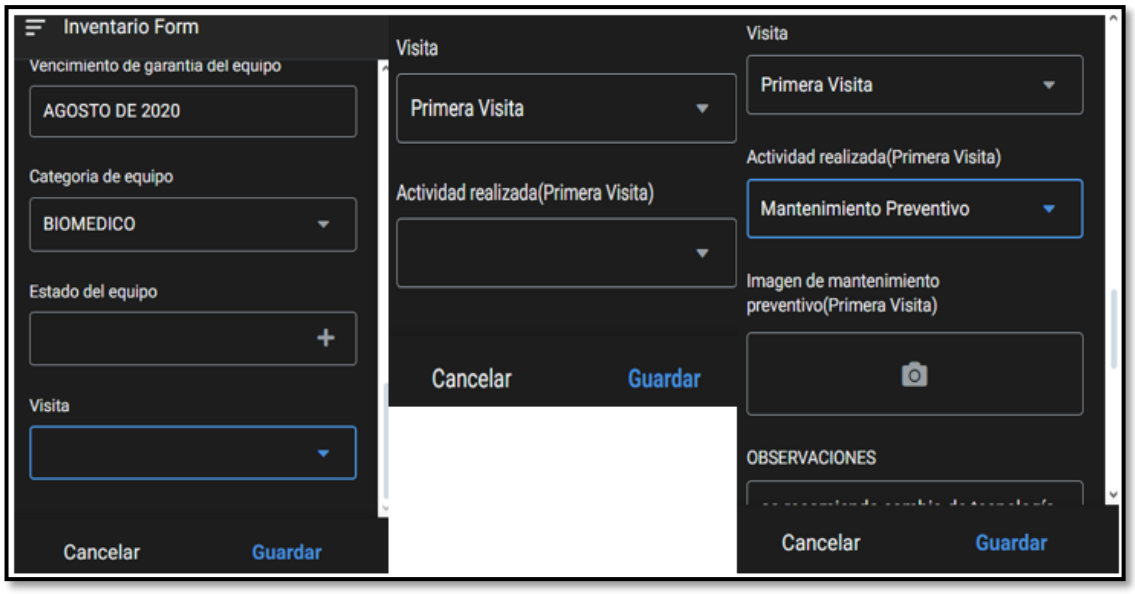


Figura 2. Appsheet informe de mantenimiento preventivo.

Este mismo proceso también se aplica al generar un informe para el mantenimiento correctivo. Sin embargo, en este caso se solicitará información diferente, como el ingeniero a cargo, el número de reporte, la fecha, entre otros datos específicos. Al seleccionar una visita, el filtro ajustará automáticamente los campos requeridos, como "Ingeniero a Cargo", "Número de

Reporte" y "Fecha", para que solo se muestre la información pertinente a la visita y el tipo de mantenimiento correctivo seleccionado.

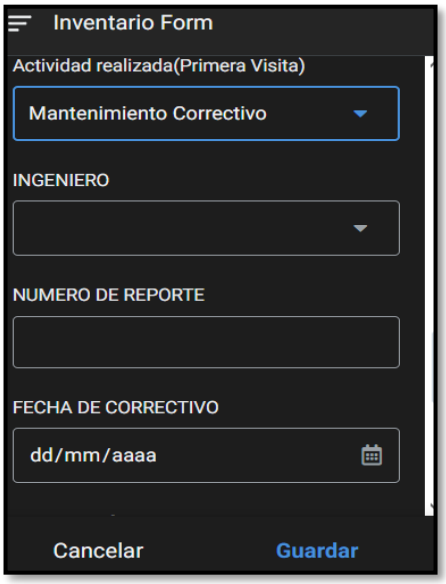


Figura 3. Appsheet informe mantenimiento correctivo.

El filtrado de la información se realiza mediante la programación de las columnas en la sección "Show If", utilizando una mediante condiciones lógicas, en las cuales se hace uso de la columna "VISITA" mediante la selección de "Primera Visita" y en la columna "ACTIVIDAD V1" se encuentra "Mantenimiento Preventivo", entonces se debe mostrar el campo correspondiente, en este caso, la casilla para tomar la imagen de la primera visita. Este enfoque asegura que solo se muestre la información relevante según la selección de visita y actividad, facilitando el proceso de ingreso de datos.

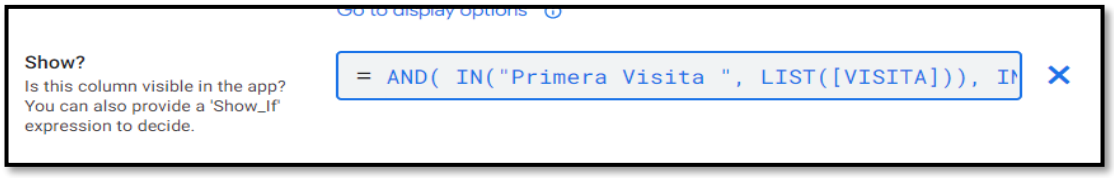


Figura 4. Appsheet restricción de visualización.

También se utiliza el condicional `OR` en el área de "Observaciones". A diferencia del condicional `AND`, que requiere que todas las condiciones especificadas se cumplan para mostrar el campo, el condicional `OR` permite que el campo se muestre si al menos una de las condiciones especificadas es verdadera. Esto proporciona mayor flexibilidad, ya que no es necesario cumplir todas las condiciones simultáneamente; con que se cumpla al menos una de ellas, el campo será visible.

```
OR(
  AND(
    IN("Primera Visita", LIST([VISITA])),
    IN("Mantenimiento Preventivo", LIST([ACTIVIDAD V1]))
  ),
  AND(
    IN("Segunda Visita", LIST([VISITA])),
    IN("Mantenimiento Preventivo", LIST([ACTIVIDAD V2]))
  ),
  AND(
    IN("Tercera Visita", LIST([VISITA])),
    IN("Mantenimiento Preventivo", LIST([ACTIVIDAD V3]))
  ),
  AND(
    IN("Cuarta Visita", LIST([VISITA])),
    IN("Mantenimiento Preventivo", LIST([ACTIVIDAD V4]))
  )
)
```

Figura 5. Appsheet Visualización de la visita.

Otro aspecto fundamental de la aplicación son las bases de datos "Lista de Chequeo" y "Características". La base de datos "Lista de Chequeo" se integra en la vista de la base de datos "Inventario" mediante un "Inline View". En esta configuración, la columna `id_equipo` está definida como un tipo de referencia (`REF`) a la tabla "Inventario", permitiendo que la lista de chequeo aparezca para su edición en el ítem correspondiente a cada equipo.

Cada ítem en la lista de chequeo puede tener opciones como "E" (Exitoso), "NE" (No Exitoso) o "NA" (No Aplica). Dependiendo de la visita seleccionada en la columna "VISITA" de la tabla "Inventario", se muestra la visita correspondiente para su edición. En este contexto, el código cambia para adaptarse a las necesidades específicas de cada visita, como se detalla a continuación.

Figura 6. Appsheet consulta para la visita.

Donde se indica que, al ser una referencia de [ID_EQUIPO], el sistema accede a la base de datos correspondiente y consulta la columna [VISITA]. Luego verifica si el valor de la columna es 'Primera Visita' para decidir si debe mostrarse.

En el área de vistas de la aplicación, es necesario detallar el proceso para configurar la opción de lista de chequeo en línea. Primero, se debe habilitar la opción para que se muestre en la vista principal. A continuación, se selecciona la lista de chequeo en línea y se activa la opción 'Enable' para permitir su edición. En la siguiente Figura se evidencia esto.

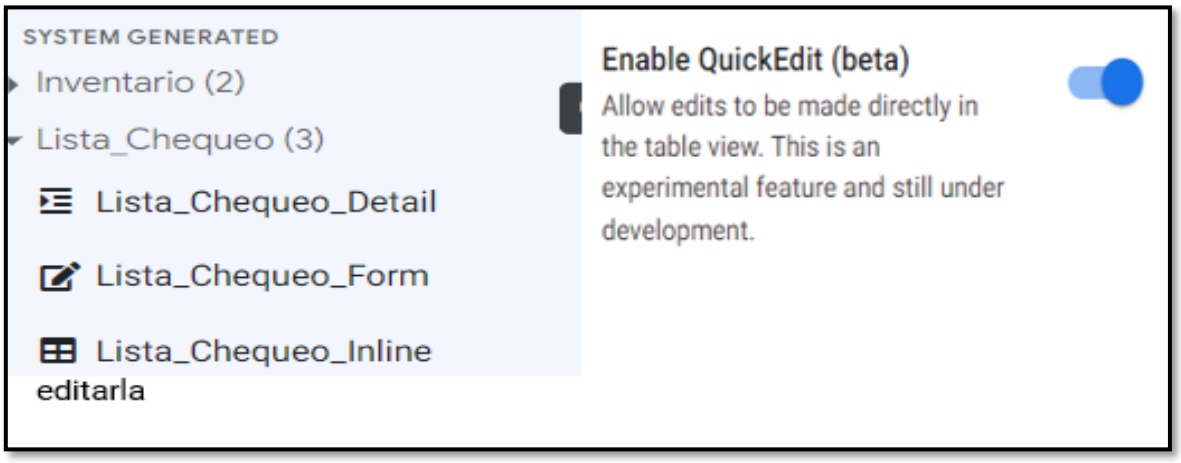


Figura 7. Appsheet configuración de lista de chequeo.

Finalmente, se crea una nueva regla en el formato para generar una alerta cuando falte algún dato por completar. Esta regla está diseñada para identificar campos vacíos y notificar al usuario sobre la información que necesita ser ingresada, asegurando así que todos los datos requeridos estén completos antes de finalizar el proceso.

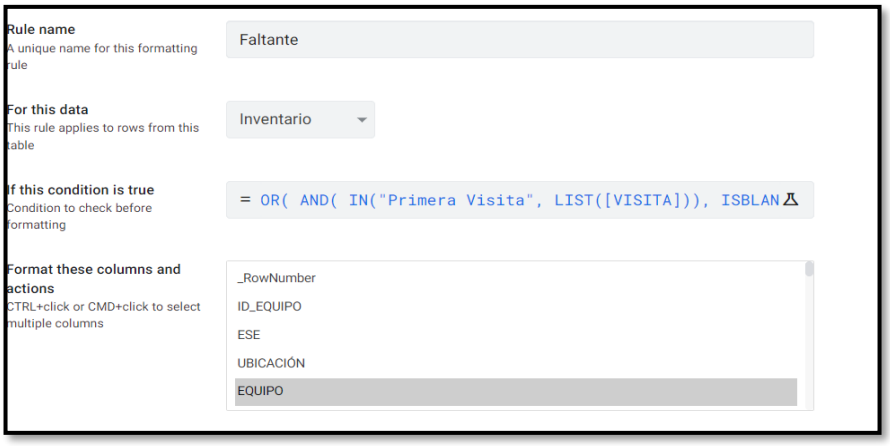


Figura 8. Appsheet campos vacíos.

Este código tiene el propósito de garantizar que en cada tipo de visita (Primera, Segunda, Tercera y Cuarta), todos los campos relevantes estén completados cuando la actividad registrada sea 'Mantenimiento Preventivo'. Si alguno de los campos necesarios está vacío (IsBlank) para una visita específica y la actividad es 'Mantenimiento Preventivo', la expresión devolverá TRUE. Esto indica que falta información y sugiere la necesidad de tomar una acción, como mostrar un mensaje de advertencia o bloquear el envío del formulario hasta que se completen todos los datos requeridos.

```
OR(
  AND(
    IN("Primera Visita", LIST([VISITA])),
    ISBLANK([VISITA 1]),
    [ACTIVIDAD V1] = "Mantenimiento Preventivo"
  ),
  AND(
    IN("Primera Visita", LIST([VISITA])),
    ISBLANK([FECHA V1]),
    [ACTIVIDAD V1] = "Mantenimiento Preventivo"
  ),
  AND(
    IN("Primera Visita", LIST([VISITA])),
    ISBLANK([FIRMA V1]),
    [ACTIVIDAD V1] = "Mantenimiento Preventivo"
  ),
  AND(
    IN("Primera Visita", LIST([VISITA])),
    ISBLANK([FIRMA MTO 1]),
    [ACTIVIDAD V1] = "Mantenimiento Preventivo"
  ),
  AND(
    IN("Primera Visita", LIST([VISITA])),
    ISBLANK([CC V1]),
    [ACTIVIDAD V1] = "Mantenimiento Preventivo"
  ),
  AND(
    IN("Segunda Visita", LIST([VISITA])),
    ISBLANK([VISITA 2]),
    [ACTIVIDAD V2] = "Mantenimiento Preventivo"
  ),
```

Figura 9. Appsheet rellenar campos según la visita.

8.3.Python:

En el siguiente apartado, se desarrollará el código destinado al ingreso de la información recopilada. Para ello, se utilizarán las siguientes bibliotecas:

- Tkinter para crear interfaces gráficas.
- Pandas para el manejo de dataframes.
- OS para interactuar con el sistema operativo.
- PIL para el procesamiento de imágenes.
- Docxtpl para la manipulación de archivos en formato Word.

Estas y otras bibliotecas se detallan en la imagen siguiente.

```
1  import tkinter as tk
2  from tkinter import ttk, messagebox, filedialog
3  import pandas as pd
4  import shutil
5  import os
6  import openpyxl
7  import requests
8  from docxtpl import DocxTemplate, InlineImage
9  from PIL import Image as PILImage
10 from io import BytesIO
11 from docx.shared import Mm
12 from datetime import datetime
13 from docx2pdf import convert
```

Figura 10. Python bibliotecas.

En el siguiente apartado del código, se declaran las rutas correspondientes a las plantillas de mantenimiento preventivo y correctivo, así como las imágenes extraídas de un archivo Excel y de una URL específica. Además, se establece la ruta de la base de datos de la cual se obtiene la información.

```
PLANTILLA_MANTENIMIETNO_PREVENTIVO = r'
    .\Inputs\Plantilla_Mantenimiento_Preventivo.docx'
PLANTILLA_MANTENIMIETNO_CORRECTIVO = r'
    .\Inputs\Plantilla_Mantenimiento_Correctivo.docx'
PLANTILLA_HOJA_DE_VIDA = r'.\Inputs\Plantilla_Hoja_de_Vida.docx'
IMAGENES_URL = r'.\Imagenes_URL'
IMAGENES_EXTRAIDAS = r'.\Imagenes_Extraidas'
BASE_DE_DATOS = 'Base_de_Datos.xlsx'

```

Figura 11. Python Rutas de las plantillas.

Este bloque de código establece variables globales que desempeñan un papel crucial en la gestión de la información. Cada variable está diseñada para almacenar dataframes que contienen datos relevantes de diferentes hojas de un archivo Excel, así como para definir rutas que facilitarán la carga y el almacenamiento de la información procesada. A continuación, se presentarán una serie de funciones encargadas del tratamiento de los datos y creación de informes.

```
Inventario_df = None
Lista_chequeo_df = None
Caracteristicas_df = None
Componentes_adicionales_df = None
ESES_df = None
SERVICIO_df = None
ruta_excel = None
output_path = None

```

Figura 12. Python Variables locales:

Se crea la función “seleccionar_excel” que permite al usuario seleccionar un archivo Excel mediante un cuadro de diálogo. Al ser seleccionado el archivo, la función carga datos de diversas hojas específicas del archivo en variables globales que almacenan dataframes. Estas hojas incluyen "Inventario", "Lista_Chequeo", "Caracteristicas", "Componentes_adicionales", "ESES" y "SERVICIOS". Si la carga de datos se realiza correctamente, los dataframes se actualizan con la información del archivo. En caso de que ocurra un error durante el proceso, como la falta de alguna hoja, la función muestra un mensaje de error para informar al usuario sobre el problema, asegurando así una interacción clara y eficiente.

```python
def seleccionar_excel():
    global Inventario_df
    global Lista_chequeo_df
    global Caracteristicas_df
    global Componentes_adicionales_df
    global ESES_df
    global SERVICIO_df
    global ruta_excel
    ruta_excel = filedialog.askopenfilename(
        title="Seleccionar archivo Excel",
        filetypes=[("Archivos de Excel", "*.xlsx *.xls")])
    if ruta_excel:
        try:
            Inventario_df = pd.read_excel(ruta_excel, sheet_name='Inventario')
            cargar_datos(Inventario_df)
            Lista_chequeo_df = pd.read_excel(ruta_excel, sheet_name
                ='Lista_Chequeo')
            Caracteristicas_df = pd.read_excel(ruta_excel, sheet_name
                ='Caracteristicas')
            Componentes_adicionales_df = pd.read_excel(ruta_excel, sheet_name
                ='Componentes_adicionales')
            ESES_df = pd.read_excel(ruta_excel, sheet_name='ESES')
            SERVICIO_df = pd.read_excel(ruta_excel, sheet_name='SERVICIOS')
```

Figura 13. Python función seleccionar Excel.

La función “cargar_datos” tiene como objetivo extraer y organizar información del dataframe “Inventario_df”. Primero, obtiene una lista única de municipios a partir de la columna 'ESE' del dataframe, eliminando duplicados y ordenando los valores. A continuación, se define una lista estática de visitas que incluye "Primera Visita", "Segunda Visita", "Tercera Visita" y "Cuarta Visita". Finalmente, la función asigna estas listas extraídas a los combobox correspondientes en la interfaz gráfica, permitiendo al usuario seleccionar entre los municipios disponibles y el tipo de visita. De esta manera, se facilita la interacción y selección de datos en la aplicación.

```python
def cargar_datos(Inventario_df):
    municipios = sorted(list(Inventario_df['ESE'].drop_duplicates()))
    visitas = ["Primera Visita", "Segunda Visita", "Tercera Visita", "Cuarta
        Visita"]
    combo_municipio['values'] = municipios
    combo_visita['values'] = visitas

```

Figura 14. . Python función cargar datos.

La función “seleccionar_carpeta” permite al usuario elegir una carpeta de destino para guardar documentos generados por la aplicación. Al ejecutarse, abre un cuadro de diálogo que solicita al usuario que seleccione una carpeta. Si el usuario selecciona una carpeta, la ruta se almacena en la variable global `output_path`, y se muestra un mensaje informativo que confirma la selección y especifica la ubicación donde se guardarán los documentos. Si el usuario cancela la selección, `output_path` se establece en `None`.

```
def seleccionar_carpeta():
    global output_path
    output_path = filedialog.askdirectory(title="Seleccione la carpeta de
        destino")
    if output_path:
        messagebox.showinfo("Carpeta seleccionada", f"Se guardarán los
            documentos en: {output_path}")
    else:
        output_path = None

```

Figura 15. Python función seleccionar carpeta.

La función `extraer_imagenes` se encarga de extraer imágenes de hojas específicas de un archivo Excel y guardarlas en un directorio determinado. Primero, define un diccionario llamado `configuracion_hojas`, que mapea los nombres de las hojas a las columnas que contienen los nombres de las imágenes. Luego, verifica si el directorio para almacenar las imágenes extraídas existe; si no, lo crea.

A continuación, itera sobre cada hoja definida en el diccionario, cargando los datos de la hoja correspondiente en un dataframe y abriendo el libro de trabajo con `openpyxl`. Luego, recorre las imágenes presentes en la hoja. Para cada imagen, determina la dirección de la celda a la que está anclada y obtiene el nombre correspondiente de la columna especificada en el dataframe. Si el nombre no está disponible, genera un nombre genérico basado en el índice de la imagen.

Finalmente, la imagen se guarda en el directorio especificado, utilizando la biblioteca PIL para manejar el formato y el almacenamiento de las imágenes extraídas. Esta función es útil para organizar y guardar visualmente información relevante contenida en el archivo Excel.

```python
def extraer_imagenes(file_path):
    configuracion_hojas = {
        'FIRMASS': 'INGENIERO',
        'ESES': 'ESE'
    }
    if not os.path.exists(IMAGENES_EXTRAIDAS):
        os.makedirs(IMAGENES_EXTRAIDAS)
    for sheet_name, columna_nombre in configuracion_hojas.items():
        df = pd.read_excel(file_path, sheet_name=sheet_name, engine='openpyxl'
            )
        workbook = openpyxl.load_workbook(file_path)
        sheet = workbook[sheet_name]
        for idx, image in enumerate(sheet._images):
            cell_address = image.anchor._from.row
            nombre_imagen = df.at[cell_address - 1, columna_nombre]
            if pd.notna(nombre_imagen):
                image_name = f"{nombre_imagen}.png"
            else:
                image_name = f"{sheet_name}_imagen_{idx + 1}.png"
            image_path = os.path.join(IMAGENES_EXTRAIDAS, image_name)
            image_stream = BytesIO(image._data())
            pil_image = PILImage.open(image_stream)
            pil_image.save(image_path)
```

Figura 16. Python función extraer imágenes.

La función “descarga_imagenes” tiene como propósito descargar imágenes desde URLs especificadas en un dataframe y guardarlas en un directorio determinado. Primero, filtra los datos para obtener solo aquellos que corresponden al “nombre_ese” proporcionado. Luego, verifica si el directorio para almacenar las imágenes existe; si no, lo crea.

A continuación, la función itera sobre cada fila del dataframe filtrado. Para cada fila, obtiene el identificador del equipo y revisa las columnas en busca de aquellas que contienen 'URL'. Si encuentra una URL válida, realiza una solicitud HTTP para obtener la imagen correspondiente. Si la solicitud es exitosa, la imagen se carga y se guarda en el directorio especificado, utilizando el identificador del equipo y el nombre de la columna para crear un nombre de archivo único.

```python
def descarga_imagenes(data, nombre_ese):
    datos_filtrados = data[data['ESE'] == nombre_ese]
    if not os.path.exists(IMAGENES_URL):
        os.makedirs(IMAGENES_URL)
    for index, row in datos_filtrados.iterrows():
        id_equipo = row.iloc[0]
        for columna in data.columns:
            if 'URL' in columna:
                image_url = row[columna]
                if pd.notna(image_url) and isinstance(image_url, str):
                    response = requests.get(image_url)
                    if response.status_code == 200:
                        img = PILImage.open(BytesIO(response.content))
                        image_name = f"{IMAGENES_URL}/{id_equipo}_{columna}.png"
                        img.save(image_name)

```

Figura 17. Python función descargar imágenes.

La función “agregar_datos_a_excel” se encarga de añadir nuevos datos a un archivo Excel existente o crear uno nuevo si no se encuentra. Primero, intenta abrir el archivo especificado utilizando “openpyxl”. Si el archivo no existe, se crea un nuevo libro de trabajo y se inicializa la hoja activa con una fila de encabezados predefinidos.

Luego, la función determina la siguiente fila vacía en la hoja para añadir los nuevos datos. Antes de agregar los datos, verifica que “datos_nuevos” sea una lista o tupla; si no es así, imprime un mensaje de error y termina la ejecución. Si los datos son válidos, se intenta añadirlos a la hoja mediante el método append. Finalmente, el libro se guarda de nuevo en el archivo especificado.

```python
def agregar_datos_a_excel(archivo_excel, datos_nuevos):
    try:
        libro = openpyxl.load_workbook(archivo_excel)
        hoja = libro.active
    except FileNotFoundError:
        libro = openpyxl.Workbook()
        hoja = libro.active
        hoja.append(['ID_EQUIPO', 'FECHA', 'PROTOCOLO', 'ACTIVIDAD', 'H_H',
            'H_P', 'REPUESTOS', 'OBSERVACIONES',
                     'EQUIPO_ACTIVO', 'NOMBRE', 'FIRMA'])
    fila_vacia = hoja.max_row + 1
    if not isinstance(datos_nuevos, (list, tuple)):
        print(f"Error: datos_nuevos debe ser una lista o tupla, pero se recibió
            {type(datos_nuevos)}")
        return
    try:
        hoja.append(datos_nuevos)
    except Exception as e:
        print(f"Error al agregar datos a la hoja de cálculo: {e}")
    libro.save(archivo_excel)
```

Figura 18. Python función agregar datos a Excel.

La función “convertir_a_fecha” transforma una cadena de texto que representa una fecha en un objeto de tipo datetime. Primero, la función normaliza el texto convirtiéndolo a mayúsculas y eliminando espacios innecesarios. Luego, define un diccionario que asocia nombres de meses en español con sus respectivos números.

Intenta convertir el texto directamente utilizando el formato %d %B %Y. Si esta conversión falla y lanza un “ValueError”, la función procede a buscar el nombre del mes en el texto. Si encuentra el mes, extrae el año y construye una cadena de fecha con el día fijo como "01", el número del mes correspondiente y el año. Luego, intenta convertir esta cadena a un objeto `datetime`. Si no se puede realizar ninguna conversión, la función retorna None.

```python
def convertir_a_fecha(texto):
    texto = texto.upper().strip()
    meses = {
        "ENERO": "01", "FEBRERO": "02", "MARZO": "03", "ABRIL": "04", "MAYO":
            "05", "JUNIO": "06",
        "JULIO": "07", "AGOSTO": "08", "SEPTIEMBRE": "09", "OCTUBRE": "10",
            "NOVIEMBRE": "11", "DICIEMBRE": "12"
    }
    try:
        return datetime.strptime(texto, '%d DE %B DE %Y')
    except ValueError:
        pass
    for mes, numero in meses.items():
        ano = ""
        if mes in texto:
            ano = texto.split()[-1]
            fecha_str = f"01/{numero}/{ano}"
            return datetime.strptime(fecha_str, '%d/%m/%Y')
    return None
```

Figura 19. Python función convertir a fecha.

La función “informe_mantenimiento” tiene como objetivo generar un informe detallado sobre el mantenimiento preventivo de equipos en un municipio específico. Comienza con la definición de una función interna, limpiar_nan, que maneja valores nulos, devolviendo el valor original si este es válido, o un valor por defecto en caso contrario.

Cuando el número de visitas (num_visit) es "Primera Visita", la función carga una plantilla de Word diseñada para informes de mantenimiento. Luego, filtra los datos del dataframe inventario para extraer únicamente la información relacionada con el municipio especificado. También se obtienen datos relevantes del dataframe eses, que incluye el nombre de la sede y la fecha de la visita.

A continuación, la función crea una lista de ubicaciones únicas dentro del municipio. Para cada ubicación, filtra los datos de los equipos y recorre cada equipo para extraer información esencial, como el ID, el tipo de equipo y las imágenes asociadas a la visita de mantenimiento. Durante la iteración, se verifica la existencia de imágenes de mantenimiento y se extrae el protocolo correspondiente del dataframe características. Además, se recopilan datos como la fecha de la visita y las observaciones, utilizando la función limpiar_nan para asegurar que la información sea válida y esté en el formato correcto. La función también verifica la fecha de vencimiento de la

garantía de cada equipo y evalúa su estado. Si la garantía está vigente, el equipo se clasifica como "en garantía"; de lo contrario, se toman las observaciones pertinentes para su registro.

Además de la información del equipo, se recogen otros datos relevantes, como el tiempo de trabajo y el estado funcional del equipo, así como el nombre del ingeniero responsable del mantenimiento. Todos estos datos se agrupan en una lista y se almacenan en la base de datos mediante la función agregar_datos_a_excel.

Una vez que se ha recopilado y estructurado toda la información, se crea un contexto para rellenar la plantilla de Word. Este contexto incluye la fecha, el nombre del municipio, el tipo de visita y los datos de los equipos. La plantilla se completa con esta información y se guarda en la ubicación especificada, generando un título adecuado para el archivo. Finalmente, el documento Word se convierte a formato PDF para facilitar su distribución y acceso. (Cabe destacar que este proceso se aplica también a las tres visitas restantes)

```
def informe_mantenimiento(inventario, nombre_municipio, num_visit, esess
    ,caracteristicass,output_path,base_de_datos):
    def limpiar_nan(valor, default=""):
        return valor if pd.notna(valor) else default
    if num_visit == "Primera Visita":
        docx_tpl = DocxTemplate(PLANTILLA_MANTENIMIETNO_PREVENTIVO)
        datos = []
        filt_Datos_Visita = inventario[(inventario['ESE'] == nombre_municipio
            )]
        ESE_INIT = esess[(esess['ESE'] == nombre_municipio)]
        nombre = " "
        fecha = " "
        nombre = ESE_INIT.at[0, 'NOMBRE CS']
        fecha = ESE_INIT.at[0, 'Fecha_V1']
        filt_ubicacion_list = sorted(list(filt_Datos_Visita['UBICACIÓN']
            .drop_duplicates()))
        for ubic in range(len(filt_ubicacion_list)):
            datos_list = []
            ubicacion = filt_ubicacion_list[ubic]
            filt_Datos_Equipo = filt_Datos_Visita[(filt_Datos_Visita['ESE'] ==
                nombre_municipio) & (filt_Datos_Visita['UBICACIÓN'] ==
                ubicacion)]
```

Figura 20. Python función informe mantenimiento 1.

```python
for index, row in filt_Datos_Equipo.iterrows():
    id_equipo = row['ID_EQUIPO']
    equipo = row['EQUIPO']
    inline_image_v1 = " "
    image_v1 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
        1_URL.png")
    if os.path.exists(image_v1):
        inline_image_v1 = InlineImage(docx_tpl, image_v1, width=Mm
            (50))
    protocolo = caracteristicass.loc[caracteristicass['EQUIPO'] ==
        equipo, 'PROTOCOLOS'].values
    if len(protocolo) > 0:
        protocolo_texto = protocolo[0]
    else:
        protocolo_texto = " "
    datos_dict = {}
    datos_dict = {
        'Equipo': limpiar_nan(row['EQUIPO']),
        'Modelo': limpiar_nan(row['MODELO']),
        'Serie': limpiar_nan(row['SERIE']),
        'Ubicacion': limpiar_nan(row['UBICACIÓN']),
        'Observaciones': limpiar_nan(row['OBSERVACIONES']),
        'Marca': limpiar_nan(row['MARCA']),
        'Protocolo': limpiar_nan(protocolo_texto),
```

Figura 21. Python función informe mantenimiento 2.

Para revisar el código completo se requiere ver el anexo C del documento

La función “hoja_vida” se encarga de generar una hoja de vida para equipos de un municipio específico, utilizando una plantilla de documento de Word. A través de la recopilación y organización de datos del inventario, se crea un informe que incluye información técnica y administrativa relevante sobre los equipos.

La función inicia cargando una plantilla de documento Word especificada por PLANTILLA_HOJA_DE_VIDA. Se define una función interna, limpiar_nan, que se utiliza para manejar valores nulos, devolviendo un valor por defecto si el dato es inválido. A continuación, se extraen datos fundamentales del dataframe esess, que incluye el nombre de la sede, NIT, correo electrónico y número de celular del municipio. Dependiendo del número de visita (num_visit), la función asigna la fecha correspondiente desde el dataframe esess. Este paso es crucial para documentar el contexto de la visita de mantenimiento.

La función verifica la existencia de imágenes relacionadas con el municipio, utilizando rutas específicas. Si las imágenes están disponibles, se insertan en el documento Word. Esto proporciona un aspecto visual que complementa la información escrita. A continuación, se filtran los datos del inventario para obtener únicamente la información relevante del municipio en cuestión. Se genera una lista de ubicaciones únicas, sobre las cuales se iterará para recolectar datos específicos de cada equipo.

Para cada equipo en una ubicación específica, se recogen datos detallados, tales como firmas de mantenimiento, nombres de responsables y fechas de las visitas. También se obtienen otros datos críticos como el protocolo de mantenimiento, recomendaciones del fabricante, y características del equipo (marca, modelo, serie, etc.). La función también maneja la inclusión de imágenes del equipo en el documento, si estas están disponibles. Esto es importante para dar una representación visual del estado del equipo.

Finalmente, se renderiza la plantilla con los datos del contexto y se guarda el documento en formato .docx. El nombre del archivo se genera de manera dinámica utilizando el nombre del municipio y la ubicación, garantizando que sea fácil de identificar. (Para ver el código completo revisar el anexo C)

```python
def hoja_vida(inventario,num_visit, nombre_municipio, esess, caracteristicass,
    lista_chequeo,servicio,componentess,output_path,base_de_datos):
    docx_tpl = DocxTemplate(PLANTILLA_HOJA_DE_VIDA)
    def limpiar_nan(valor, default=""):
        return valor if pd.notna(valor) else default
    ESE_INIT = esess[(esess['ESE'] == nombre_municipio)]
    nombre = ESE_INIT.at[0,'NOMBRE CS']
    nit = ESE_INIT.at[0,'NIT']
    email= ESE_INIT.at[0, 'E-MAIL']
    celular = ESE_INIT.at[0, 'CELULAR']
    fecha = " "
    if num_visit == "Primera Visita":
        fecha = ESE_INIT.at[0, 'Fecha_V1']
    elif num_visit == "Segunda Visita":
        fecha = ESE_INIT.at[0, 'Fecha_V2']
    elif num_visit == "Tercera Visita":
        fecha = ESE_INIT.at[0, 'Fecha_V3']
    elif num_visit == "Cuarta Visita":
        fecha = ESE_INIT.at[0, 'Fecha_V4']
    inline_image_logo = " "
    image_logo = os.path.join(IMAGENES_EXTRAIDAS, f"{nombre_municipio}.png")

```

Figura 22. Python función hoja de vida 1.

```python
    if os.path.exists(image_logo):
        inline_image_logo = InlineImage(docx_tpl, image_logo, width=Mm(10))
    inline_image_logo1 = " "
    image_logo1 = os.path.join(IMAGENES_EXTRAIDAS, f"{nombre_municipio}.png")
    if os.path.exists(image_logo1):
        inline_image_logo1 = InlineImage(docx_tpl, image_logo1, width=Mm(50))
    filt_Datos_Visita = inventario[(inventario['ESE'] == nombre_municipio)]
    filt_ubicacion_list = sorted(list(filt_Datos_Visita['UBICACIÓN']
        .drop_duplicates()))
    for ubic in range(len(filt_ubicacion_list)):
        ubicacion = filt_ubicacion_list[ubic]
        filt_Datos_Equipo = filt_Datos_Visita[(filt_Datos_Visita['ESE'] ==
            nombre_municipio) & (filt_Datos_Visita['UBICACIÓN'] == ubicacion)]
    datos = []
        for index, row in filt_Datos_Equipo.iterrows():
            ing1 = " "
            ing1 = row['FIRMA_MTO_1']
            ing2 = " "
            ing2 = row['FIRMA_MTO_2']
            ing3 = " "
            ing3 = row['FIRMA_MTO_3']
            ing4 = " "
```

Figura 23. Python función hoja de vida 2.

La función “informe_correctivo” comienza filtrando el inventario para obtener solo los datos correspondientes al municipio especificado. Se seleccionan las columnas relevantes, y se eliminan las filas con valores NaN en estas columnas. Si no hay datos válidos, la función finaliza sin generar un informe.

Para cada fila de datos filtrados, se procede a extraer información relevante:

- Inicialización de Variables: Se obtiene el nombre de la entidad prestadora de servicios (ESE) y se carga la plantilla del documento para el mantenimiento correctivo.
- Recopilación de Imágenes: Se intenta cargar imágenes asociadas al equipo y al ingeniero. Si las imágenes existen, se convierten en objetos InlineImage.

Luego se procede a crear un diccionario de datos que contengan la información relevante del mantenimiento, incluyendo:

- Detalles del Ingeniero: Firma y nombre del ingeniero.
- Datos del Reporte: Número de reporte, fecha, falla reportada, actividades realizadas, repuestos utilizados, y más.
- Información del Equipo: Incluye marca, modelo, serie e inventario.

Después se verifica el estado de la garantía del equipo. Se convierte la fecha de vencimiento a un formato de fecha y se compara con la fecha actual. Dependiendo del resultado, se determina si el equipo está en garantía o si se deben tomar otras observaciones. Se extrae el tiempo de trabajo del equipo desde el dataframe de características. Si no se encuentra información, se maneja adecuadamente el caso. Dependiendo del ingeniero asignado al mantenimiento, se asigna un nombre específico a la variable ingeniero, facilitando la personalización del reporte. Se prepara una lista con datos relevantes que se añaden a la base de datos, utilizando la función agregar_datos_a_excel.

Finalmente, se renderiza el documento con los datos recopilados y se guarda en el directorio especificado. El nombre del archivo se genera dinámicamente utilizando el nombre del equipo y otros detalles pertinentes, asegurando que sea fácilmente identificable.

```python
def informe_correctivo(inventario, nombre_municipio, caracteristicass,
    output_path, base_de_datos, esess):
    DATA = inventario[inventario['ESE'] == nombre_municipio]
    columnas_usadas = ['INGENIERO', 'NUMERO_DE_REPORTE',
        'FECHA_DE_CORRECTIVO']
    DATA_filtrado = DATA.dropna(subset=columnas_usadas)
    if DATA_filtrado.empty:
        return
    for index, row in DATA_filtrado.iterrows():
        ESE_INIT = esess[(esess['ESE'] == nombre_municipio)]
        nombre = ESE_INIT.at[0, 'NOMBRE CS']
        docx_tpl = DocxTemplate(PLANTILLA_MANTENIMIETNO_CORRECTIVO)
        id_equipo = row['ID_EQUIPO']
        inge = row['INGENIERO']
        image_fm = os.path.join(IMAGENES_URL,
            f"{id_equipo}_RECIBE_A_SATISFACION_URL.png")
        inline_image_f1 = None
        inline_image_inge = None
        if os.path.exists(image_fm):
            inline_image_f1 = InlineImage(docx_tpl, image_fm, width=Mm(50))
        image_inge = os.path.join(IMAGENES_EXTRAIDAS, f"{inge}.png")
        if os.path.exists(image_inge):
            inline_image_inge = InlineImage(docx_tpl, image_inge, width=Mm(50
                ))
```

Figura 24. Python función informe correctivo.

La función “generar_documento” se encarga de coordinar la generación de diferentes documentos basados en la información seleccionada por el usuario. Este proceso implica varias verificaciones y la ejecución de funciones específicas para asegurar que los documentos sean creados correctamente.

La función comienza verificando que los DataFrames necesarios, como Inventario_df, Caracteristicas_df, y otros, no sean None. Si alguno de estos no ha sido cargado, se muestra una advertencia al usuario para que seleccione primero un archivo Excel. Además, se verifica si se ha seleccionado una carpeta de destino para guardar los documentos generados; de no ser así, se informa al usuario mediante un mensaje de advertencia.

Después, se obtienen el municipio y el número de visita seleccionados por el usuario. Si alguno de estos no está seleccionado, se muestra otra advertencia, asegurando que se tenga toda la información necesaria para proceder.

Si todas las verificaciones son exitosas, la función procede a ejecutar varias tareas. En primer lugar, se eliminan las carpetas de imágenes temporales (IMAGENES_EXTRAIDAS y IMAGENES_URL) para evitar conflictos con datos anteriores. A continuación, se llaman a las funciones extraer_imagenes y descarga_imagenes, que son responsables de extraer y descargar las imágenes correspondientes a los equipos en el inventario del municipio seleccionado. Luego, se invocan la función informe_mantenimiento, hoja_vida e informe_correctivo, pasando los parámetros necesarios para la creación de cada uno de los documentos. Esto permite generar informes detallados y organizados para el municipio seleccionado.

Finalmente, se muestra un mensaje de éxito al usuario, informándole que los documentos han sido creados y guardados en la carpeta de destino especificada. Esto proporciona una confirmación visual de que el proceso se completó correctamente y que los informes están listos para ser utilizados.

```
def generar_documento():
    if Inventario_df is None or Caracteristicas_df is None or Lista_chequeo_df
        is None or Componentes_adicionales_df is None or ESES_df is None:
        messagebox.showwarning("Advertencia", "Por favor, seleccione un
            archivo Excel primero.")
        return
    if output_path is None:
        messagebox.showwarning("Advertencia", "Por favor, seleccione una
            carpeta de destino primero.")
        return
    nombre_municipio = combo_municipio.get()
    num_visit = combo_visita.get()
    if not nombre_municipio or not num_visit:
        messagebox.showwarning("Advertencia", "Por favor, seleccione un
            municipio y una visita.")
        return
    eliminar_carpeta(IMAGENES_EXTRAIDAS)
    eliminar_carpeta(IMAGENES_URL)
    extraer_imagenes(ruta_excel)
    descarga_imagenes(Inventario_df, nombre_municipio)
    informe_mantenimiento(Inventario_df, nombre_municipio, num_visit, ESES_df,
        Caracteristicas_df, output_path,BASE_DE_DATOS)
```

Figura 25. Python función generar documento.

La aplicación se presenta con una ventana principal titulada "Generador de Documentos" con dimensiones de 400x300 píxeles. Incluye un botón para seleccionar un archivo Excel necesario

para la generación de documentos, así como otro botón para elegir la carpeta de destino donde se guardarán los archivos generados.

Los usuarios pueden seleccionar un municipio y un número de visita mediante dos combobox, que permiten personalizar el contenido de los informes. Por último, hay un botón que, al ser presionado, ejecuta la función de generación de documentos.

```
root = tk.Tk()
root.title("Generador de Documentos")
root.geometry("400x300")
boton_seleccionar_excel = tk.Button(root, text="Seleccionar archivo Excel",
    command=seleccionar_excel)
boton_seleccionar_excel.pack(pady=10)
boton_seleccionar_carpeta = tk.Button(root, text="Seleccionar carpeta de
    destino", command=seleccionar_carpeta)
boton_seleccionar_carpeta.pack(pady=10)
tk.Label(root, text="Seleccione Municipio:").pack(pady=10)
combo_municipio = ttk.Combobox(root)
combo_municipio.pack()
tk.Label(root, text="Seleccione Visita:").pack(pady=10)
combo_visita = ttk.Combobox(root)
combo_visita.pack()
boton_generar = tk.Button(root, text="Generar Documento", command
    =generar_documento)
boton_generar.pack(pady=20)
root.mainloop()
```

Figura 26. Python interfaz gráfica.

8.4.Instalación de la App

Para crear un ejecutable de la aplicación previamente desarrollada, es necesario seguir una serie de pasos. A continuación, se detallan las instrucciones para llevar a cabo este proceso de manera efectiva.

8.4.1 Paso 1: Instalar Python y Configurar el PATH

8.4.1.1 Descargar Python:

- Visita el sitio oficial de Python: python.org/downloads.
- Descarga la versión más reciente de Python.

8.4.1.2 Instalar Python:

- Durante la instalación, asegúrate de seleccionar la opción "Add Python to PATH". Esta opción es esencial para que Python y sus herramientas se puedan utilizar desde la terminal (cmd).
- Completa la instalación.

8.4.1.3 Verificar la instalación de Python en el cmd:

- Abre el cmd (símbolo del sistema) y ejecuta el siguiente comando para confirmar que Python está correctamente instalado (python –version)

8.4.1.4 Confirmación de instalación:

- Si ambos comandos funcionan correctamente, significa que Python está instalado y configurado adecuadamente en tu sistema.

8.4.2 Paso 2: Agregar la Ruta de Scripts al PATH de Windows

Si Python no se añadió automáticamente al PATH, puedes hacerlo manualmente siguiendo estos pasos:

- Identificar la ruta de los Scripts de Python: La ruta será similar a C:\Users\<tu_usuario>\AppData\Local\Programs\Python\Python<versión>\Scripts
- Verifica que la carpeta "Scripts" contenga herramientas como pip.

8.4.2.1 Agregar al PATH:

- Abre el Panel de Control → Sistema → Configuración avanzada del sistema.
- Haz clic en Variables de entorno.
- En Variables del sistema, selecciona la variable Path y haz clic en Editar.
- Haz clic en Nuevo y pega la ruta hacia la carpeta "Scripts".
- Acepta todos los cambios y reinicia el cmd para que las modificaciones surtan efecto.

8.4.3 Paso 3: Preparar la Estructura de Carpetas y Archivos

8.4.3.1 Crear una carpeta principal:

Dentro de esta carpeta, organiza lo siguiente:

- Una carpeta inputs que contenga las plantillas de hoja de vida, mantenimiento correctivo y mantenimiento preventivo.
- El archivo .py del programa principal (por ejemplo, App.py).
- La base de datos necesaria para la aplicación.
- Un icono para el ejecutable en formato .ico.

Figura 27. Python creación de app 1. (carpeta de archivos).

8.4.4 Paso 4: Instalar PyInstaller

- Abre el símbolo del sistema (cmd) y ejecuta el siguiente comando para instalar PyInstaller: pip install pyinstaller
- Verifica que la instalación fue exitosa ejecutando: pyinstaller –versión

8.4.5 Paso 5: Crear el ejecutable (.exe) con PyInstaller

- Navega hasta la carpeta que contiene tus archivos utilizando el siguiente comando en el cmd: cd "C:\ruta\a\tu\carpeta\Proyecto"
- Ejecuta PyInstaller para crear el archivo ejecutable con el siguiente comando: pyinstaller -F -w -i icono.ico App.py
- F: Empaqueta todos los archivos en un único ejecutable.
- w: Evita que se abra la ventana de consola al ejecutar la aplicación (opcional para aplicaciones con interfaz gráfica).
- i icono.ico: Asigna un icono personalizado al ejecutable.
- App.py: Es el archivo Python que deseas convertir en ejecutable.
- PyInstaller creará una carpeta llamada dist, donde podrás encontrar el archivo App.exe.

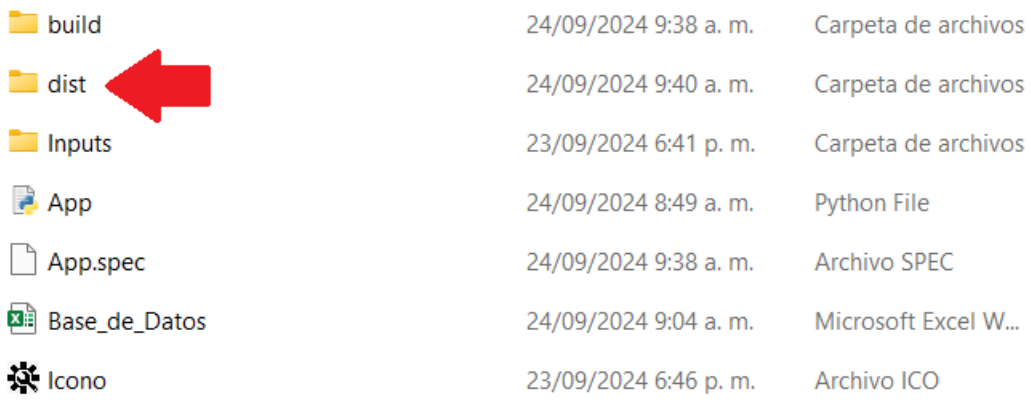


Figura 28. Python creación de app 2. (carpeta de archivos).

8.4.6 Paso 6: Instalar las bibliotecas necesarias

Ahora que tienes tu archivo ejecutable, es fundamental asegurarte de que todas las bibliotecas requeridas por tu aplicación estén instaladas. A continuación, se presenta una lista de las bibliotecas necesarias y los comandos para instalarlas.

8.4.6.1 Bibliotecas que debes instalar:

Para instalar todas las bibliotecas externas de una sola vez, ejecuta el siguiente comando en el símbolo del sistema (cmd):

- pip install pandas openpyxl requests docxtpl pillow docx2pdf python-docx pywin32
- pandas: Para la manipulación de datos en DataFrames. (pip install pandas)
- openpyxl: Para manejar archivos Excel (.xlsx). (pip install openpyxl)
- requests: Para hacer solicitudes HTTP. (pip install requests)
- docxtpl: Para manipular plantillas de documentos Word. (pip install docxtpl)
- Pillow: Para el manejo de imágenes. (pip install pillow)
- docx2pdf: Para convertir archivos Word a PDF. (pip install docx2pdf)
- pywin32: Para interactuar con las API de Windows, necesario para docx2pdf en Windows. (pip install pywin32)
- python-docx: Para trabajar con documentos de Word. (pip install python-docx)

8.4.7 Paso 7: Ejecutar el archivo ejecutable (.exe)

- Una vez creado el ejecutable, navega a la carpeta dist que contiene tu archivo .exe.
- Haz doble clic en el archivo App.exe para ejecutar tu aplicación

8.5.Plantillas

En esta sección se presentarán las diferentes plantillas que se utilizan para la elaboración de informes, tanto para el mantenimiento correctivo como preventivo, así como para la creación de hojas de vida de equipos. Se mostrará el formato que se empleará y los campos que la aplicación busca completar con la información recopilada a través de AppSheet.

8.5.1 Plantilla para la hoja de vida de un equipo

En la siguiente plantilla, se presentan las portadas que se pueden observar en la imagen. Los espacios que contienen la forma {{Ubicacion}} son los que la aplicación utilizará para completar la información. Cada uno de estos espacios incluye una palabra clave que facilitará su identificación y posterior modificación. En estas portadas, se busca incluir información sobre la ubicación del equipo, el municipio correspondiente y el logo de la entidad correspondiente.

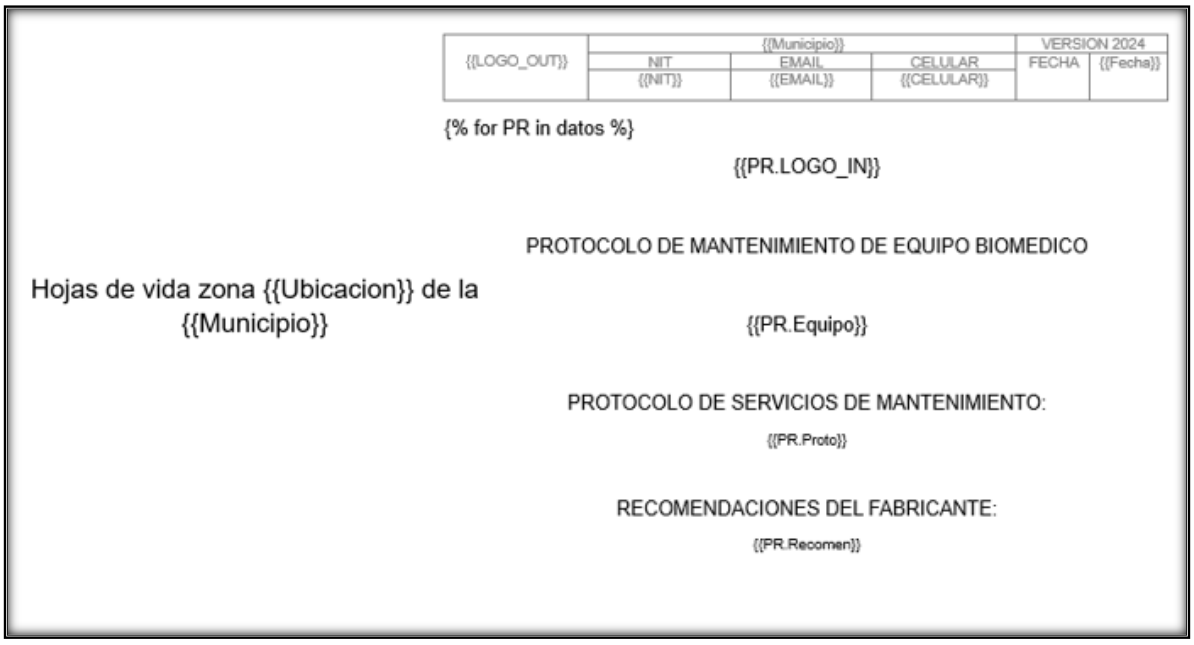
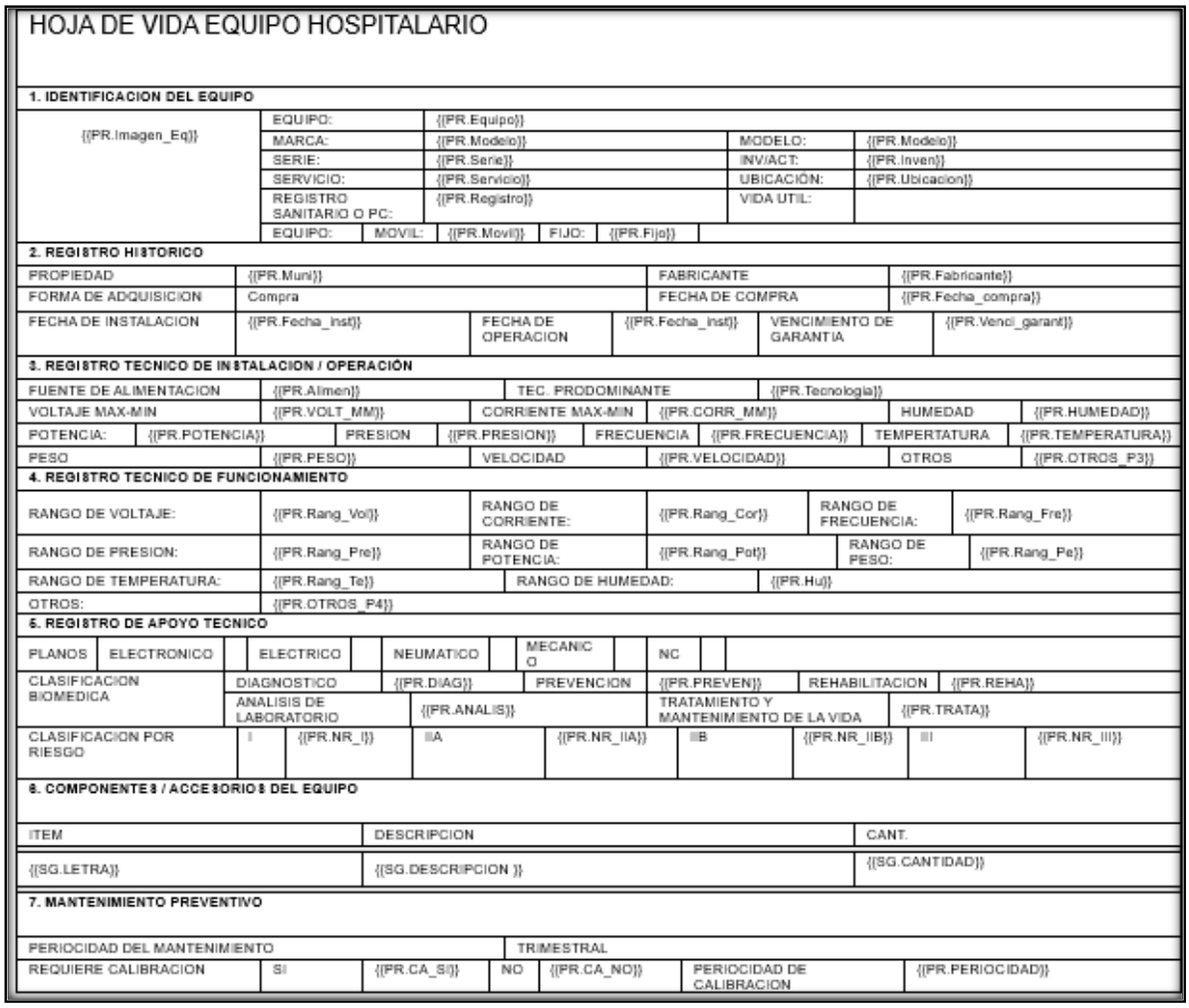


Figura 29. Plantilla hoja de vida 1.

En la siguiente sección de la plantilla se presenta el formulario correspondiente al equipo, donde se busca completar los siguientes campos de información: la identificación del equipo, el registro histórico, el registro técnico de operación, el registro técnico de funcionamiento, el registro de apoyo técnico, los componentes del equipo y, finalmente, la lista de chequeo del equipo. Esta última requiere las firmas del ingeniero encargado del mantenimiento y del responsable que recibe el equipo.



HOJA DE VIDA EQUIPO HOSPITALARIO

1. IDENTIFICACION DEL EQUIPO

{{PR.Imagen_Eq}}	EQUIPO:	{{PR.Equipo}}		
	MARCA:	{{PR.Modelo}}	MODELO:	{{PR.Modelo}}
	SERIE:	{{PR.Serie}}	INVIACT:	{{PR.Inven}}
	SERVICIO:	{{PR.Servicio}}	UBICACIÓN:	{{PR.Ubicacion}}
	REGISTRO SANITARIO O PC:	{{PR.Registro}}	VIDA UTIL:	
	EQUIPO:	MOVIL: {{PR.Movil}}	FIJO: {{PR.Fijo}}	

2. REGISTRO HISTORICO

PROPIEDAD	{{PR.Muni}}		FABRICANTE	{{PR.Fabricante}}	
FORMA DE ADQUISICION	Compra		FECHA DE COMPRA	{{PR.Fecha_compra}}	
FECHA DE INSTALACION	{{PR.Fecha_inst}}	FECHA DE OPERACION	{{PR.Fecha_inst}}	VENCIMIENTO DE GARANTIA	{{PR.Venci_garant}}

3. REGISTRO TECNICO DE INSTALACION / OPERACIÓN

FUENTE DE ALIMENTACION	{{PR.Almen}}		TEC. PRODOMINANTE	{{PR.Tecnologia}}			
VOLTAJE MAX-MIN	{{PR.VOLT_MM}}		CORRIENTE MAX-MIN	{{PR.CORR_MM}}		HUMEDAD	{{PR.HUMEDAD}}
POTENCIA:	{{PR.POTENCIA}}	PRESION	{{PR.PRESION}}	FRECUENCIA	{{PR.FRECUENCIA}}	TEMPERTATURA	{{PR.TEMPERATURA}}
PESO	{{PR.PESO}}		VELOCIDAD	{{PR.VELOCIDAD}}		OTROS	{{PR.OTROS_P3}}

4. REGISTRO TECNICO DE FUNCIONAMIENTO

RANGO DE VOLTAJE:	{{PR.Rang_Vol}}	RANGO DE CORRIENTE:	{{PR.Rang_Cor}}	RANGO DE FRECUENCIA:	{{PR.Rang_Fre}}
RANGO DE PRESION:	{{PR.Rang_Pre}}	RANGO DE POTENCIA:	{{PR.Rang_Pot}}	RANGO DE PESO:	{{PR.Rang_Pe}}
RANGO DE TEMPERATURA:	{{PR.Rang_Te}}	RANGO DE HUMEDAD:	{{PR.Hu}}		
OTROS:	{{PR.OTROS_P4}}				

5. REGISTRO DE APOYO TECNICO

PLANOS	ELECTRONICO		ELECTRICO		NEUMATICO		MECANICO		NC
CLASIFICACION BIOMEDICA	DIAGNOSTICO	{{PR.DIAG}}	PREVENCION	{{PR.PREVEN}}	REHABILITACION	{{PR.REHA}}			
	ANALISIS DE LABORATORIO	{{PR.ANALIS}}		TRATAMIENTO Y MANTENIMIENTO DE LA VIDA	{{PR.TRATA}}				
CLASIFICACION POR RIESGO	I	{{PR.NR_I}}	IIA	{{PR.NR_IIA}}	IIB	{{PR.NR_IIB}}	III	{{PR.NR_III}}	

6. COMPONENTES / ACCESORIOS DEL EQUIPO

ITEM	DESCRIPCION	CANT.
{{SG.LETRA}}	{{SG.DESCRIPCION }}	{{SG.CANTIDAD}}

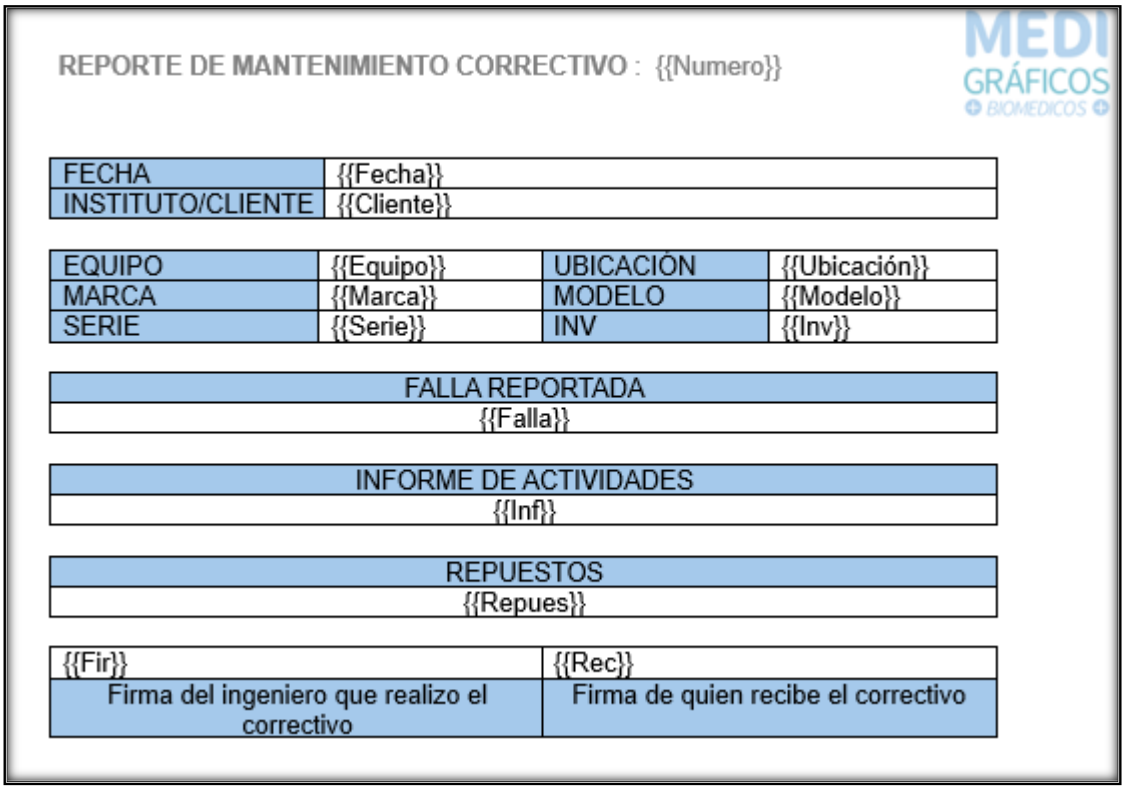
7. MANTENIMIENTO PREVENTIVO

PERIOCIDAD DEL MANTENIMIENTO	TRIMESTRAL					
REQUIERE CALIBRACION	SI	{{PR.CA_SI}}	NO	{{PR.CA_NO}}	PERIOCIDAD DE CALIBRACION	{{PR.PERIOCIDAD}}

Figura 30. Plantilla hoja de vida 2.

8.5.2 Plantilla para el mantenimiento correctivo de equipos

La siguiente plantilla corresponde al mantenimiento correctivo de los equipos y se compone únicamente de un formulario con el siguiente formato presentado. Para completar la información, se utiliza el mismo método que en la plantilla anterior, con un espacio clave que incluye una palabra clave para facilitar su llenado. Este formato recopila información sobre el equipo que recibió el mantenimiento, la falla presentada y el informe de la actividad realizada para devolverlo a circulación. Además, se registran los repuestos requeridos y las firmas del ingeniero y de la persona encargada de recibir el equipo.



REPORTE DE **MANTENIMIENTO CORRECTIVO** : {{Numero}}

MEDI GRÁFICOS BIOMEDICOS

FECHA	{{Fecha}}
INSTITUTO/CLIENTE	{{Cliente}}

EQUIPO	{{Equipo}}	UBICACIÓN	{{Ubicación}}
MARCA	{{Marca}}	MODELO	{{Modelo}}
SERIE	{{Serie}}	INV	{{Inv}}

FALLA REPORTADA
{{Falla}}

INFORME DE ACTIVIDADES
{{Inf}}

REPUESTOS
{{Repues}}

{{Fir}}	{{Rec}}
Firma del ingeniero que realizo el correctivo	Firma de quien recibe el correctivo

Figura 31. Plantilla mantenimiento correctivo.

8.5.3 Plantilla para el mantenimiento preventivo de equipos

Finalmente, se presenta la plantilla que recopila los informes de mantenimiento realizados a los equipos. En primera instancia, cuenta con dos portadas que indican los dispositivos médicos involucrados, el municipio donde se llevó a cabo el mantenimiento, la fecha de realización y, por último, la entidad correspondiente en la que se realizó el procedimiento.

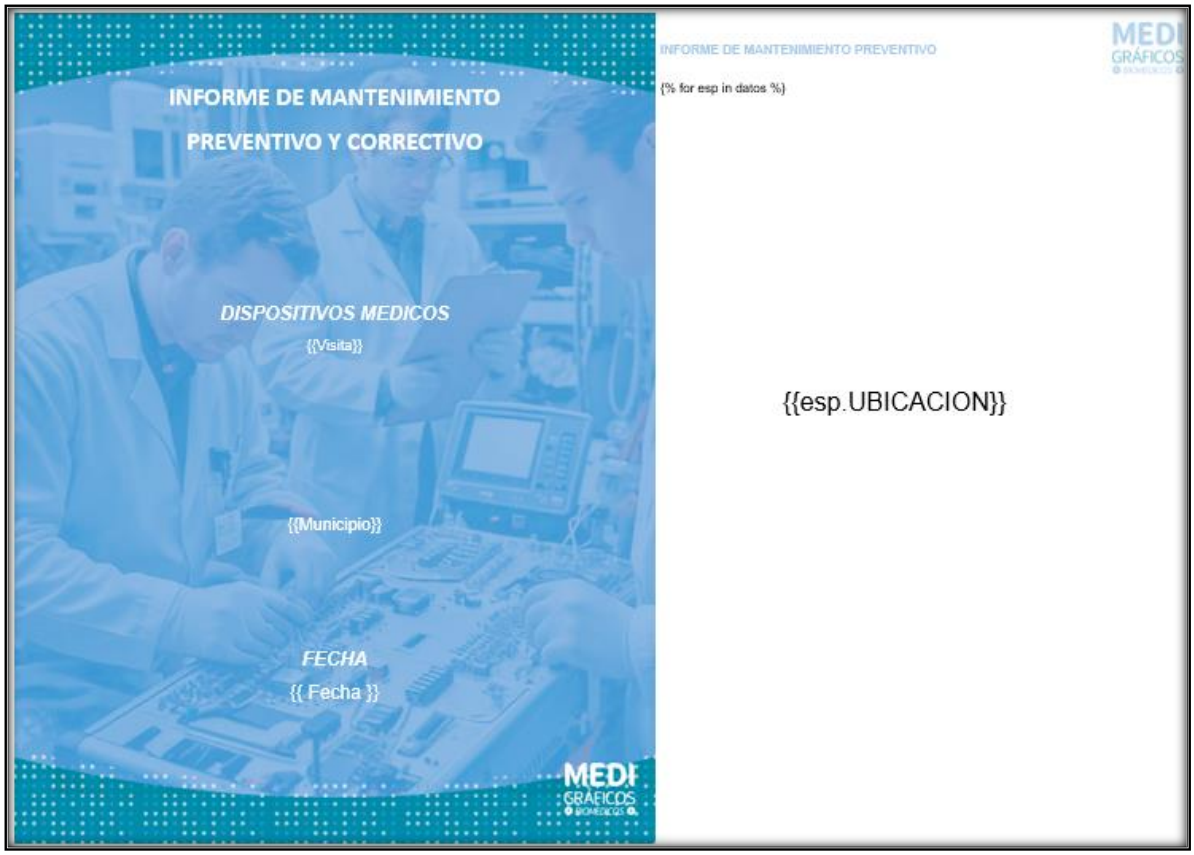
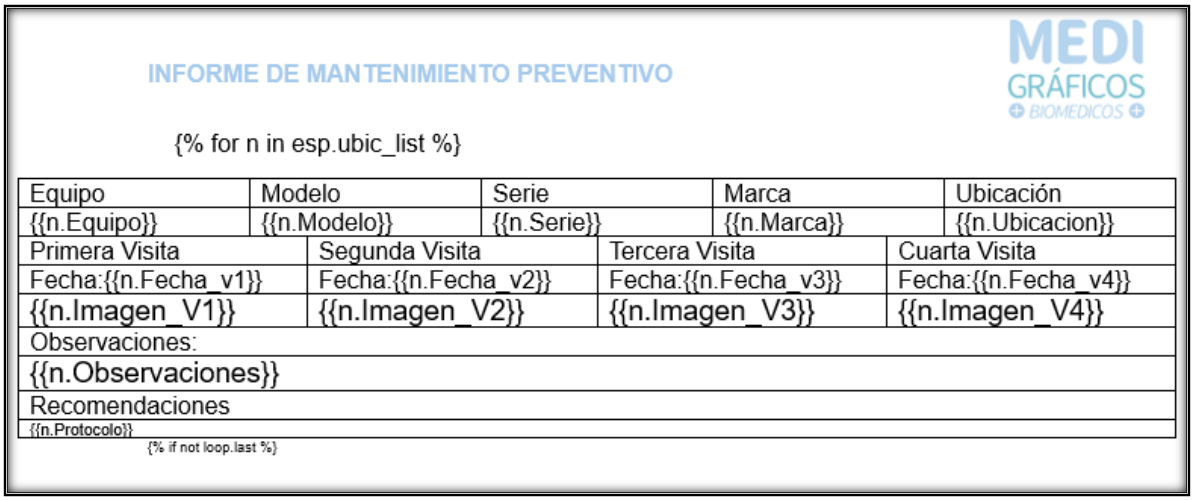


Figura 32. Plantilla mantenimiento preventivo 1.

En la siguiente sección de la plantilla se recopilan las evidencias de las cuatro visitas programadas al centro de salud. Como se observa en el formulario, se utiliza la misma lógica que en los formatos anteriores para completar la información. Lo más relevante de esta plantilla son las fotos y las fechas, ya que estas funcionan como evidencia de que se realizó el mantenimiento a los equipos.



INFORME DE MANTENIMIENTO PREVENTIVO

MEDI GRÁFICOS BIOMEDICOS

{% for n in esp.ubic_list %}

Equipo	Modelo	Serie	Marca	Ubicación
{{n.Equipo}}	{{n.Modelo}}	{{n.Serie}}	{{n.Marca}}	{{n.Ubicacion}}

Primera Visita	Segunda Visita	Tercera Visita	Cuarta Visita
Fecha:{{n.Fecha_v1}}	Fecha:{{n.Fecha_v2}}	Fecha:{{n.Fecha_v3}}	Fecha:{{n.Fecha_v4}}
{{n.Imagen_V1}}	{{n.Imagen_V2}}	{{n.Imagen_V3}}	{{n.Imagen_V4}}
Observaciones:			
{{n.Observaciones}}			
Recomendaciones			
{{n.Protocolo}}			

{% if not loop.last %}

Figura 33. Plantilla mantenimiento preventivo 2.

9. Análisis de resultados

En esta sección se presenta un compendio de los resultados obtenidos mediante la aplicación. A lo largo de este análisis, se destacarán los diferentes tipos de informes que la herramienta permite generar. Cada uno de estos informes ofrece una perspectiva única sobre los datos, lo que facilita una comprensión más profunda de los resultados. La diversidad en los informes no solo enriquece la interpretación de la información recopilada, sino que también brinda a los usuarios la flexibilidad necesaria para adaptarse a diferentes necesidades y contextos. A continuación, se explorarán las características y beneficios de cada tipo de informe, resaltando su importancia en la toma de decisiones y en la mejora continua de los procesos.

9.1.Hojas de vida de equipos

En esta plantilla, se incluye una portada que indica la zona donde se encuentran los equipos biomédicos, así como el centro de salud en el cual se recopiló la información. A continuación, se presenta una portada específica para cada equipo biomédico, la cual se generará mediante la aplicación. En estas dos portadas, se completarán campos reservados que pueden variar según la zona y el centro de salud mencionados en la primera portada. En la segunda portada, se incluirá el logo del centro de salud, junto con información relevante sobre el equipo en cuestión, el servicio de mantenimiento que se le proporcionará y las recomendaciones del fabricante.

Figura 34. Resultados hoja de vida 1.

A continuación, se presenta la hoja de vida del equipo, la cual se organiza en un formato que recopila toda la información relevante sobre el mismo. Esto incluye desde datos básicos, como la fotografía, marca, número de serie y modelo, hasta información más específica, como un registro histórico y un registro técnico de operación, así como otros detalles importantes relacionados con el equipo.

HOJA DE VIDA EQUIPO HOSPITALARIO

1. IDENTIFICACION DEL EQUIPO

EQUIPO:	Balanza de piso digital con tallímetro		
MARCA:	500 KL	MODELO:	500 KL
SERIE:	5000010000000000000	INV/ACT:	19
SERVICIO:	CONSULTA EXTERNA	UBICACIÓN:	ENFERMERÍA
REGISTRO SANITARIO O PC:	2013DM-0010061	VIDA UTIL:	

EQUIPO:	MOVIL:	X	FIJO:	

2. REGISTRO HISTORICO

PROPIEDAD	ESE CENTRO DE SALUD TOCA			FABRICANTE	HEALTH O METER
FORMA DE ADQUISICION	Compra			FECHA DE COMPRA	Junio de 2015
FECHA DE INSTALACION	Junio de 2015	FECHA DE OPERACION	Junio de 2015	VENCIMIENTO DE GARANTIA	junio de 2016

3. REGISTRO TECNICO DE INSTALACION / OPERACIÓN

FUENTE DE ALIMENTACION	BATERIA		TEC. PRODOMINANTE		ELECTRONICA		
VOLTAJE MAX-MIN	9 V DC		CORRIENTE MAX-MIN			HUMEDAD	95% HR
POTENCIA:		PRESION	NO APLICA	FRECUENCIA	NO APLICA	TEMPERTATURA	10 A 40 °C
PESO	6,89 KG		VELOCIDAD	NO PLICA		OTROS	

4. REGISTRO TECNICO DE FUNCIONAMIENTO

RANGO DE VOLTAJE:	NO APLICA	RANGO DE CORRIENTE:	NO APLICA	RANGO DE FRECUENCIA:	NO APLICA
RANGO DE PRESION:	NO APLICA	RANGO DE POTENCIA:	NO APLICA	RANGO DE PESO:	0-200 KG
RANGO DE TEMPERATURA:	NO APLICA	RANGO DE HUMEDAD:			
OTROS:					

5. REGISTRO DE APOYO TECNICO

PLANOS	ELECTRONICO		ELECTRICO		NEUMATICO		MECANICO		NC	
CLASIFICACION BIOMEDICA	DIAGNOSTICO	X	PREVENCION		REHABILITACION				
	ANALISIS DE LABORATORIO				TRATAMIENTO Y MANTENIMIENTO DE LA VIDA				
CLASIFICACION POR RIESGO	I	X	IIA		IIB		III		

6. COMPONENTES / ACCESORIOS DEL EQUIPO

ITEM	DESCRIPCION	CANT.

7. MANTENIMIENTO PREVENTIVO

PERIOCIDAD DEL MANTENIMIENTO			TRIMESTRAL			
REQUIERE CALIBRACION	SI	X	NO		PERIOCIDAD DE CALIBRACION	ANUAL

Figura 35. Resultados hoja de vida 2.

En la siguiente página del informe se detallan las cuatro visitas reglamentarias de mantenimiento preventivo que se deben realizar al equipo biomédico. Durante cada visita, es fundamental seguir una lista de chequeo que permita verificar que el equipo se encuentra en perfectas condiciones para llevar a cabo su labor. Posteriormente, se incluirán los espacios designados para las firmas del ingeniero a cargo del mantenimiento y del responsable de recibir dicho mantenimiento.

LISTA DE CHEQUEO 2024 CONTROL DE MANTENIMIENTO PREVENTIVO												
EQUIPO	MARCA	MODELO	SERIE	INVENTARIO	UBICACION
Balanza de piso digital con tallímetro	HEALTH O METER	500 KL	5000010000000000000	19	ENFERMERÍA

ITEM	VISITA 1			VISITA 2			VISITA 3			VISITA 4		
	E	NE	NA	E	NE	NA	E	NE	NA	E	NE	NA
Inspeccionar las condiciones ambientales en las que se encuentra el equipo				X			X			X		
Realice prueba de funcionamiento				X			X			X		
Efectuar la limpieza integral externa del equipo.				X			X			X		
Inspeccionar el sistema eléctrico o electrónico según corresponda.				X			X			X		
Verificar que se encuentre en cero, si no es así reajuste.				X			X			X		
Realice prueba de funcionamiento final				X			X			X		

	VISITA 1	VISITA 2	VISITA 3	VISITA 4
FECHA		2024-04-16	2024-07-05	2024-10-03
FIRMA Y SELLO INGENIERO QUE REALIZA EL MANTENIMIENTO		Yon Ali Torres Torres Ingeniero Electrónico USTA Tunja	Yon Ali Torres Torres Ingeniero Electrónico USTA Tunja	Yon Ali Torres Torres Ingeniero Electrónico USTA Tunja
FIRMA DEL ENCARGADO DEL ÁREA				
NOMBRE DEL ENCARGADO		Natalia Pulido	Natalia Pulido	Natalia Pulido
CÉDULA DEL ENCARGADO		1057607831.0	1057607831	1057607831.0
LISTA DE CHEQUEO Nº				EQ2667

Figura 36. Resultados hoja de vida 3.

Finalmente, se incluye un área destinada al registro histórico del mantenimiento, donde se deberá completar la información con la firma del ingeniero encargado de realizar el mantenimiento del equipo correspondiente. (Este proceso se repetirá con los demás equipos biomédicos que se encuentren en el centro de salud.)

HOJA DE VIDA EQUIPO HOSPITALARIO

EQUIPO	Balanza de piso digital con tallímetro	MARCA	HEALTH O METER	MODELO	500 KL
SERIE	5000010000000000000	INV/ACT	19		

8. REGISTRO HISTORICO DE MANTENIMIENTO

Fecha (aaaa/mm/dd)	Nº Reporte / Lista Chequeo/ Protocolo MTO	Actividad/ Novedad	H.H	H.P	Repuestos	Observaciones/ Acciones Relacionadas	Equipo Activo	Nombre completo (Responsable del MTO)	Firma (Responsable del MTO)
2024-04-16	EQ2667	Mantenimiento Preventivo	0.16	0.16				JOHN ALI	
2024-07-05	EQ2667	Mantenimiento Preventivo	0.16	0.16				JOHN ALI	
2024-10-03	EQ2667	Mantenimiento Preventivo	0.16	0.16				JOHN ALI	

CONVENCIONES: **INV./ACT= Numero de inventario o activo asignado el equipo por el prestador y debe concidir con el numero que reposa en el almacen. **PC= Permiso de comercializacion. ** VENC.= Fecha en la cual se vence la garantia establecida por el proveedor . **MAX.= Maximo. ** MIN.= Minimo. ** NC.= No cuenta. ** CANT.= Cantidad. ** H.H= Cantidad de horas que se gastaron en la intervencion realizada de ingeniero, tecnico, o personal similar, en la reparacion especifica del equipo. Se debe registrar en horas, ejemplo: 30 minutos es 0,5 horas. **H.P= Cantidad de horas que duro el equipo sin funcionar debido a la intervencion realizada. Adicionalmente se debe registrar el tiempo de parada en la consecucion de los repuestos, los insumos o accesorios y tramites admistrativos requeridos para la puesta en funcionamiento. **MTO.= Mantenimiento.

Figura 37. Resultados hoja de vida 4.

9.2.Reporte de mantenimiento

Al ingresar a la plantilla, se encuentra la siguiente portada. La aplicación se encarga de completar automáticamente la información relacionada con la visita correspondiente (en este caso, la cuarta), así como el centro de salud del reporte y, finalmente, la fecha en la que se realizó dicho reporte.

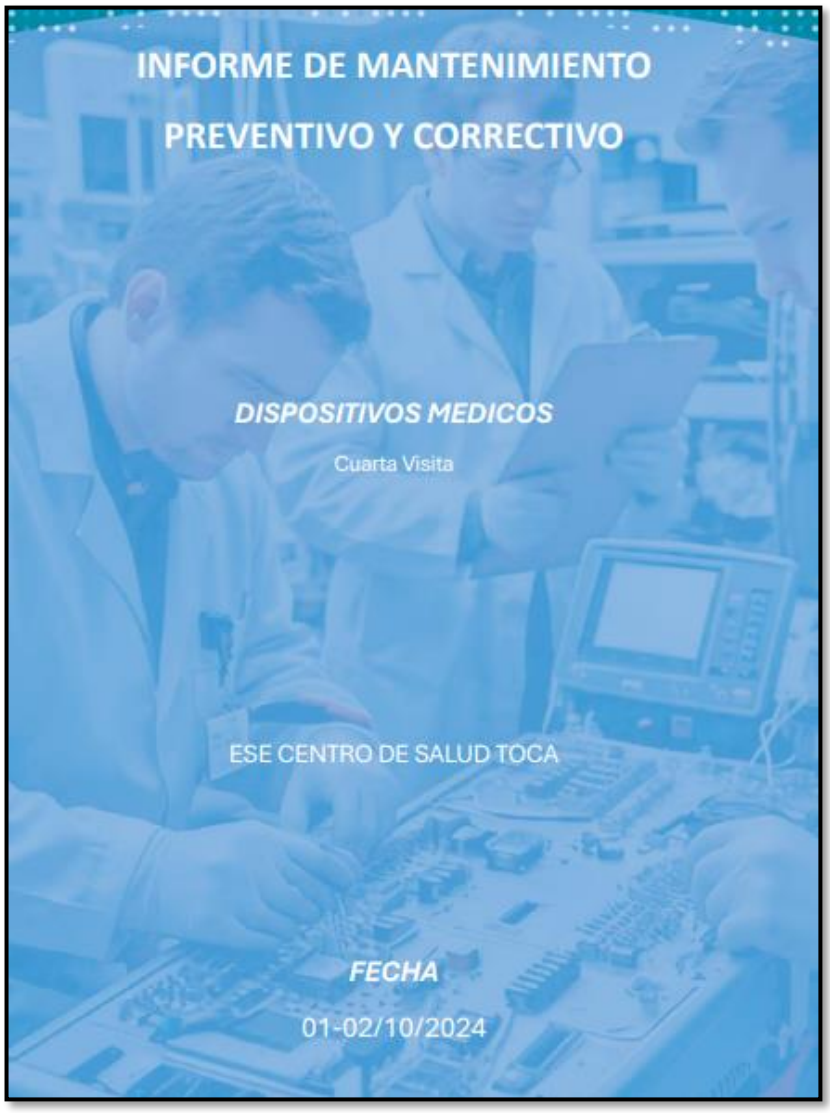


Figura 38. Resultados reporte de mantenimiento 1.

A lo largo del informe, se presentan una serie de portadas que indican las diferentes áreas del centro de salud. Dentro de cada sección, se incluyen formularios específicos para cada equipo de dicha área, los cuales recopilan información sobre el mantenimiento correspondiente a la visita en cuestión.

Dentro de las diferentes áreas del informe, se presenta el siguiente formato, el cual recopila información sobre la visita actual y las anteriores. Este formato incluye fotografías que sirven como evidencia del mantenimiento realizado, así como información general del equipo biomédico, que abarca el modelo, número de serie, marca, ubicación y nombre del equipo.

Equipo	Modelo	Serie	Marca	Ubicación
Desfibrilador (DEA)	BENEHEART C2	AE9-28004787	MINDRAY	ALMACÉN

Primera Visita	Segunda Visita	Tercera Visita	Cuarta Visita
Fecha:Fecha no disponible	Fecha:2024-04-15	Fecha:2024-07-05	Fecha:2024-10-02
			

Observaciones:
Recomendaciones
Verifique el estado y caducidad de la batería y electrodos, compruebe el exterior del Desfibrilador, si se observan fisuras u otros signos de daño. Limpie el exterior con un paño suave humedecido con agua jabonosa o alcohol, ajuste de conectores prueba en modo DEA, Revisión de impresora y cabeza térmica si tiene prueba de autotesty pruebas finales de funcionamiento.

Figura 39. Resultados reporte de mantenimiento 3.

10. Discusión de resultados

En esta sección, se analizan los resultados obtenidos en relación con los objetivos y requisitos establecidos para la aplicación. Se discuten los hallazgos más significativos y se evalúa el grado en que la aplicación ha cumplido con sus metas.

10.1. Cumplimiento de objetivos

10.1.1 Análisis de objetivo general

En este apartado, se evalúa si la aplicación ha cumplido con los objetivos propuestos. Para ello, se analizarán cada uno de ellos y se argumentará su cumplimiento, tal como se presentó en el análisis de resultados. En primer lugar, se considera el siguiente objetivo general:

- Desarrollar un sistema encargado de la gestión de información del mantenimiento de equipos biomédicos, empleando AppSheet para la recopilación de datos y Python para la generación automatizada de informes.

Considerando lo expuesto en el objetivo general, se puede afirmar que el proyecto cumple con estas estipulaciones. En el área de desarrollo, se ha utilizado AppSheet para crear una aplicación móvil que permite a los usuarios recopilar la información necesaria para los informes, la cual se almacena en un archivo de Excel para su posterior tratamiento.

Además, se emplea Python para procesar la información y generar el informe. Gracias a este potente software, es posible automatizar la creación de informes utilizando una plantilla predeterminada. Asimismo, se pueden realizar diversas modificaciones con solo unas pocas líneas de código, lo que ahorra considerable tiempo a los usuarios y permite corregir errores de manera eficiente.

10.1.2. Análisis de los objetivos específicos

Una vez analizado el objetivo general, se procedera a revisar los objetivos específicos del proyecto para garantizar el cumplimiento de todos los requisitos de la aplicación.

- Diseñar una aplicación móvil en AppSheet que recopile la información relacionada con las inspecciones, reparaciones y mantenimiento preventivo de equipos biomédicos.

Como se evidencia en la sección de desarrollo, específicamente en el área dedicada a la creación de la aplicación con AppSheet, esta ha sido diseñada con el propósito de que los usuarios recopilen la información necesaria para elaborar informes de mantenimiento correctivo o preventivo, así como hojas de vida de equipos. La aplicación incluye funciones de filtrado y organización de datos, lo que permite estructurar la información de manera adecuada.

- Integrar la aplicación móvil con una base de datos en Excel para almacenar los datos recopilados de manera centralizada y accesible, garantizando la integridad y seguridad de la información.

La aplicación de AppSheet opera a través de hojas de cálculo, las cuales se almacenan en la nube mediante la cuenta del usuario que crea la aplicación. Esto permite afirmar que la información está centralizada, ya que solo el usuario creador tiene acceso a ella. Además, proporciona una amplia accesibilidad, permitiendo el ingreso desde diferentes dispositivos. También se puede afirmar que esta información goza de buena integridad, ya que, al estar en la nube, está protegida contra la pérdida y las modificaciones no deseadas.

- Implementar scripts de Python para procesar los datos almacenados en la base de datos de Excel, para automatizar la generación de informes detallando el estado y la historia de mantenimiento de los equipos biomédicos.

Los scripts de Python permiten procesar de manera efectiva los datos almacenados en la base de datos de Excel, lo que cumple con el objetivo de automatizar la generación de informes sobre el estado y la historia de mantenimiento de los equipos biomédicos. Gracias a las bibliotecas de Python, es posible leer archivos CSV y formatos derivados, facilitando la manipulación de la información recopilada en la nube.

Para la creación de informes, solo se requiere una plantilla, y luego se puede desarrollar un script que complete automáticamente los espacios necesarios con texto e imágenes, estableciendo parámetros como tamaño, fuente y ubicación. Esta automatización no solo optimiza el proceso, sino que también asegura que los informes sean consistentes y precisos.

Además, como se menciona en la sección de Manual, Python permite crear un ejecutable en el escritorio que proporciona a los usuarios una interfaz interactiva, evitando la necesidad de interactuar con el código fuente. Esto mejora la comprensión y facilita el uso de la aplicación, lo que, en conjunto, respalda el cumplimiento del objetivo de automatizar la generación de informes detallados sobre el mantenimiento de los equipos biomédicos.

11. Conclusiones

Después del desarrollo del proyecto se puede concluir que:

- Se ha creado una aplicación móvil mediante AppSheet que permite recopilar información sobre el mantenimiento de equipos biomédicos en los diferentes centros de salud. Esta información puede ser posteriormente procesada para la elaboración de informes.

- La aplicación desarrollada no solo recopila datos en la plataforma móvil, sino que también registra la información en una base de datos en Excel, lo que proporciona una doble capa de seguridad al almacenar los datos en diferentes ubicaciones.

- Se ha complementado el diseño de la aplicación mediante un código en Python que permite procesar la información recopilada y generar diferentes tipos de informes según las necesidades del usuario. A través de la modificación de plantillas, esta funcionalidad facilita significativamente las tareas de los usuarios.

- Se ha creado un archivo ejecutable que permite acceder de manera inmediata a la interfaz gráfica de la aplicación. Esto facilita a los usuarios la generación de informes sin necesidad de interactuar con el código fuente, haciendo que el proceso de creación de informes sea intuitivo y sencillo.

- La implementación de una base de datos en Excel, junto con la recopilación de datos en la plataforma móvil, refuerza la seguridad y confiabilidad de la información. Al almacenar los datos en diferentes ubicaciones, se garantiza una mayor protección contra posibles pérdidas o errores en el almacenamiento.

- La automatización del proceso de generación de informes mediante la interfaz gráfica y el archivo ejecutable permite a los usuarios generar reportes de manera más rápida y eficiente. Optimizando el uso de recursos, ya que elimina la necesidad de intervención manual en el

procesamiento de datos, reduciendo tiempos de espera y aumentando la productividad en los centros de salud.

- También se concluye que el pasante no enfrentó dificultades durante el desarrollo de su pasantía, gracias a las habilidades adquiridas a lo largo de su formación, que le permitieron desempeñar todas las actividades encomendadas con éxito. Además, se destaca la valiosa experiencia que obtuvo, ya que le brindó la oportunidad de poner a prueba sus conocimientos en un entorno laboral real, lo que contribuye de manera significativa a su desarrollo profesional.

12. Referencias

Chen, J., Li, S., & Li, X. (2019). A low-code approach for mobile application development. IEEE Access, 7, 112-123.

Claros Luis. (2024). Diseño e Implementación Móvil para Gestión de Solicitudes de Repuestos en Equipos Biomédicos y Registro de Dificultades Técnicas.

Cristancho Lenis, C., & Lozano Hoyos, M. (2020). Desarrollo de un sistema de gestión de bases de datos en la nube para la optimización del proceso de integración de información de tecnología médica y validación a través de indicadores de cumplimiento.

Do Dinh-Cuong; Nguyen The-Vinh; Bui Anh-Tu. Enhancing student admissions management efficiency through digital transformation: A case study using a no-code development platform. International Research Journal of Science, Technology, Education, & Management (IRJSTEM), [s. l.], v. 4, n. 1, p. 1–8, 2024. DOI 10.5281/zenodo.10971980. Disponível em: https://research-ebsco-com.crai-ustadigital.usantotomas.edu.co/linkprocessor/plink?id=8e7fb811-0a30-368c-b507-f0b90c6d6968. Acesso em: 9 dez. 2024

Estrada Porras. (2017). Digitalización de la gestión del mantenimiento Unitecod SAS.

Fuaddi, M., & Subirosa, B. (2018). The Implementation and Testing of a Prototype for Web-Based Medical Equipment Maintenance System. In 2018 International Conference on Electrical Engineering and Informatics (ICELTICs) (pp. 159-163). Banda Aceh, Indonesia. DOI: 10.1109/ICELTICS.2018.8548828

Khalaf, A., Djouani, K., Hamam, Y., & Alayli, Y. (2017). Failure-cost mathematical model for medical equipment maintenance. In 2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME) (pp. 1-4). Beirut, Lebanon. DOI: 10.1109/ICABME.2017.8167525

Lavingia, K., Purohit, P., Dutta, V. et al. Advancements in automated testing tools for Android set-top boxes: a comprehensive evaluation and integration approach. Int J Syst Assur Eng

Manag 15, 2808–2817 (2024). https://doi-org.crai-ustadigital.usantotomas.edu.co/10.1007/s13198-024-02335-6

Martel Solis, L. K. (2021). Automatización de procesos para el registro de facturas en formato XML en una base de datos mediante Python.

Marqués, M. (2009). Bases de datos: (ed.). Castelló de la Plana, Spain: D - Universitat Jaume I. Servei de Comunicació i Publicacions. Recuperado de https://elibro.net/es/ereader/usta/51645?page=10.

Mora Pérez, L. D., Rivera Jaramillo, M. Y., & Otálora Rodriguez, S. F. (2019). Importancia de la innovación tecnológica en las empresas colombianas y su aporte en la productividad desde el año 2019. Revista de Investigaciones, 5(2), 45-56.

Nejković, V., Radenković, M., & Petrović, N. (2021). Ultramarathon Result and Injury Prediction using PyTorch. In 2021 15th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS) (pp. 249-252). Nis, Serbia. DOI: 10.1109/TELSIKS52058.2021

NUGROHO, P. S. Optimization of the Reporting and Monitoring System for Air Conditioning Maintenance at Pt Ppa Site Sks. Eduvest: Journal Of Universal Studies, [s. l.], v. 4, n. 3, p. 1138–1145, 2024. Disponível em: https://research-ebsco-com.crai-ustadigital.usantotomas.edu.co/linkprocessor/plink?id=9e393e5e-3519-302a-ba42-e8990d5f4b6d. Acesso em: 9 dez. 2024.

Petrović, N., Radenković, M., & Nejković, V. (2020). Data-driven mobile applications based on AppSheet as support in COVID-19 crisis. IcETRAN 2020, 1-6.

Pacheco, J. A., & Noriega, P. L. (s.f.). Desarrollo de un software para la gestión de acciones de cumplimiento de procesos de los equipos biomédicos de una institución prestadora de servicios de salud. Recuperado de http://hdl.handle.net/20.500.12749/23778

Patil, K.R., Deshpande, S., Joshi, S. et al. Developing an integrated design framework using python scripting for parametric CAD modelling of flange coupling. Int J Interact Des Manuf 18, 4451–4462 (2024). https://doi-org.crai-ustadigital.usantotomas.edu.co/10.1007/s12008-023-01694-x

Puntoni, V., Masselli, G. M. P., & Silvestri, S. (2021). An adaptation of Pareto's parametric distribution as a support tool for the analysis of maintenance costs of biomedical equipment. In 2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT) (pp. 32-36). Rome, Italy. DOI: 10.1109/MetroInd4.0IoT51437.2021

Valderrey Sanz, P. (2015). Gestión de bases de datos: (ed.). Madrid, Spain: RA-MA Editorial. Recuperado de https://elibro.net/es/ereader/usta/62469?page=13.

Anexo: Mantenimiento

En esta sección, se describirán las diversas actividades que llevó a cabo el pasante durante su pasantía en Medigráficos Biomédicos. A lo largo de este período, el pasante experimento un ambiente profesional que le permitió poner a prueba sus conocimiento y habilidades adquiridas. A su vez este asumió una serie de responsabilidades y tareas clave que contribuyeron a su desarrollo y al funcionamiento de la empresa.

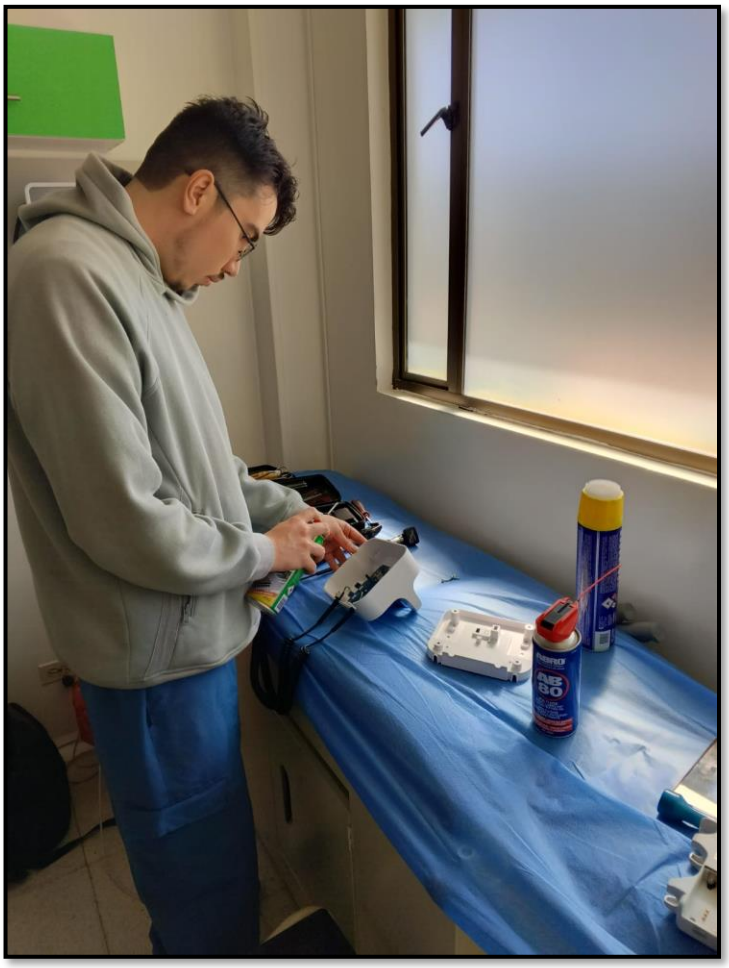


Figura 40. Anexo. Actividades del pasante, Mantenimiento preventivo.

Como se muestra en la Figura anterior, una de las responsabilidades del pasante consistía en llevar a cabo el mantenimiento preventivo de los equipos biomédicos. Para ello, primero recibió capacitación específica que le permitió adquirir las habilidades necesarias para desempeñar esta labor. Una vez completada su formación, el pasante pudo realizar las funciones habituales de un ingeniero dentro de la empresa, contribuyendo así al correcto funcionamiento y optimización de los equipos.

Figura 41. Anexo. Actividades del pasante, Mantenimiento Preventivo

Otra función importante que desempeñan los pasantes en la empresa es el mantenimiento correctivo de los equipos. Este proceso implica investigar la causa de un mal funcionamiento y, una vez identificada, proceder a reparar el desperfecto. Como se muestra en la imagen anterior, el pasante también estuvo a cargo de esta tarea, específicamente en el caso de una balanza digital para bebés, asegurando su correcto funcionamiento y contribuyendo así a la calidad de los servicios ofrecidos por la empresa.

Durante este proceso de pasantía, el pasante logró desempeñar las actividades asignadas de manera efectiva, sin encontrar mayores percances. Además, identificó un área de mejora dentro de la empresa relacionada con la creación de informes. Esto lo llevó a desarrollar un proyecto de investigación en el que colaboró en el diseño e implementación de soluciones tecnológicas innovadoras. También estuvo involucrado en el análisis de datos y en la elaboración de informes técnicos, lo que permitió tomar decisiones informadas sobre los productos y servicios ofrecidos. Su iniciativa y enfoque proactivo contribuyeron significativamente al desarrollo y optimización de los procesos en la empresa.

Finalmente, es importante destacar que el pasante proporcionó a la empresa una herramienta que facilita la generación automática de informes. Esta innovación no solo optimiza los procesos internos, sino que también contribuye al crecimiento y avance de Medigráficos Biomédicos como proveedor de servicios.

Tabla 6. *Lista de equipos para mantenimiento preventivo.*

Equipo	Cantidad	ESES	Imagen
Amalgamador	3	Tutaza, Cucaita , Tinjaca	
Autoclave	2	Berbeo , San eduardo	
Balanza de piso digital	4	Toca, Tutaza	
Balanza de piso digital con tallímetro	1	Toca	
Balanza pesa bebé digital	1	San eduardo	
Baño serológico	1	Toca	
Cavitron	3	Tutaza, Cucaita , Tinjaca	

Tabla 5

(Continuación) Lista de equipos para mantenimiento preventivo.

Centrífuga	1	Rondon	
Compresor	2	Cucaita , Tinjaca	
Concentrador de oxígeno	1	Susacon	
Contador de células	1	Rondon	
Contra ángulo	6	Tutaza, Cucaita , Tinjaca	
Data logger	1	Rondon	
Desfibrilador	1	Cucaita	

Tabla 5

(Continuación) Lista de equipos para mantenimiento preventivo.

Doppler fetal	1	Berbeo	
Electrocardiógrafo	1	Tutaza	
Equipo de órganos de pared	1	Mongua	
Fonendoscopio adulto/pediátrico	4	Cucaita,Berbeo	
Incubadora de pruebas biológicas	1	San eduardo	
Infantómetro	2	San eduardo, Berbeo	
Lámpara de fotocurado	3	Tutaza, Cucaita , Tinjaca	
Laringoscopio	1	La uvita	

Tabla 5

(Continuación) Lista de equipos para mantenimiento preventivo.

Microcentrífuga	1	Rondon	
Micromotor	5	Tutaza, Cucaita , Tinjaca	
Microscopio	1	Rondon	
Nebulizador	4	Susacon	
Negatoscopio	2	Rondon , San eduardo	
Pieza de mano	5	Tutaza, Cucaita , Tinjaca	
Pieza recta	2	Tutaza, Cucaita , Tinjaca	
Pulsioxímetro	2	Rondon	

Tabla 5

(Continuación) Lista de equipos para mantenimiento preventivo.

Succionador	3	Susacon, Tutaza	
Tensiómetro adulto	8	Cucaita, San eduardo	
Unidad odontológica	4	Rondon,Cucaita, Tinjaca, La Uvita	
Unidad odontológica portátil	1	Cucaita	

Anexo: Manual

MANUAL DE USO DE LA APLICACIÓN AUTOMATIZACION DE INFORMES

Realizado por: Juan Sebastián Martínez Cárdenas

Contacto: 3212898494

Este manual está diseñado para proporcionar una descripción detallada del funcionamiento de la aplicación, permitiendo a los usuarios completar todas las tareas de manera eficiente. A continuación, se detallan los pasos necesarios para navegar y utilizar las funciones principales de la herramienta.

Figura 42. Manual ejecutable.

Introducción al Funcionamiento

Una vez se ingresa a la aplicación, se recibirá al usuario mediante una interfaz gráfica que resulta clara e intuitiva permitiendo realizar todas las operaciones necesarias para generar informes. La pantalla inicial muestra cuatro opciones principales que deben completarse para generar un documento con éxito:

1. **Seleccionar archivo Excel**: Permite cargar los datos recopilados en un archivo de Excel.
2. **Seleccionar carpeta de destino**: Define el lugar donde se guardará el documento generado.
3. **Seleccionar municipio**: Especifica el lugar donde se realizó el mantenimiento, eligiendo entre una lista predefinida de entidades.
4. **Seleccionar visita**: Determina el tipo de visita (Visita 1, 2, 3 o 4).

El flujo de trabajo de la aplicación se basa en completar estos cuatro pasos de manera secuencial antes de proceder a generar el documento.

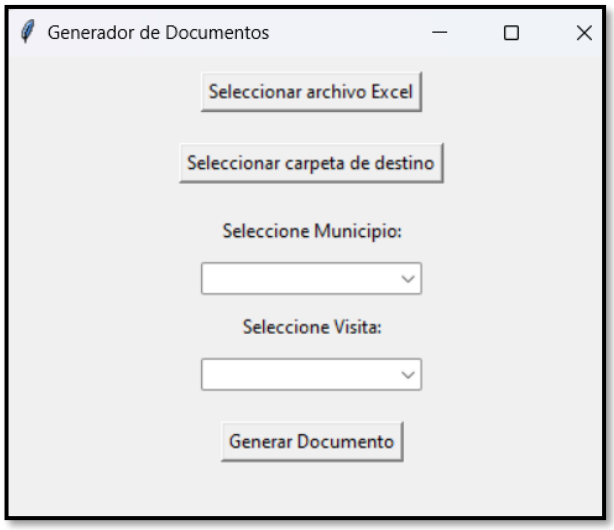


Figura 43. Manual pantalla de inicio.

Detalle de las Funciones

1. Seleccionar el Archivo Excel

Este es el primer paso en el uso de la aplicación.

1. Se hace clic en el botón **"Seleccionar archivo Excel"** en la interfaz principal.
2. Aparecerá una ventana emergente que permitirá buscar en tu dispositivo el archivo con los datos necesarios.

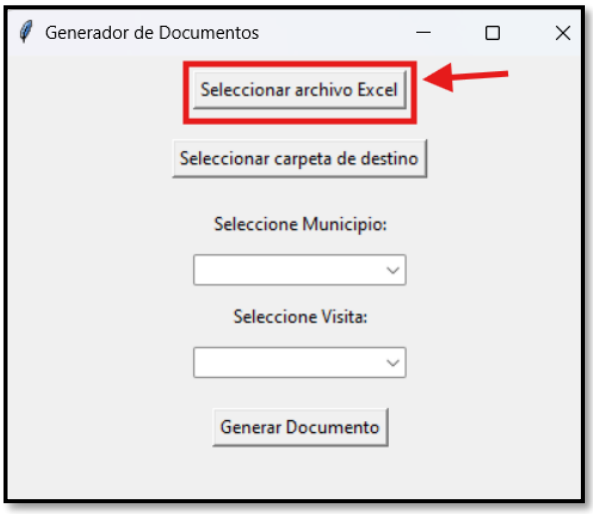


Figura 44. Manual seleccionar archivo 1.

Requisitos del Archivo

- El archivo debe tener la extensión **.xls** o **.xlsx**, formatos estándar de Excel.
- Se debe asegurar de que el archivo contiene la información relevante, organizada según las especificaciones del flujo de trabajo.

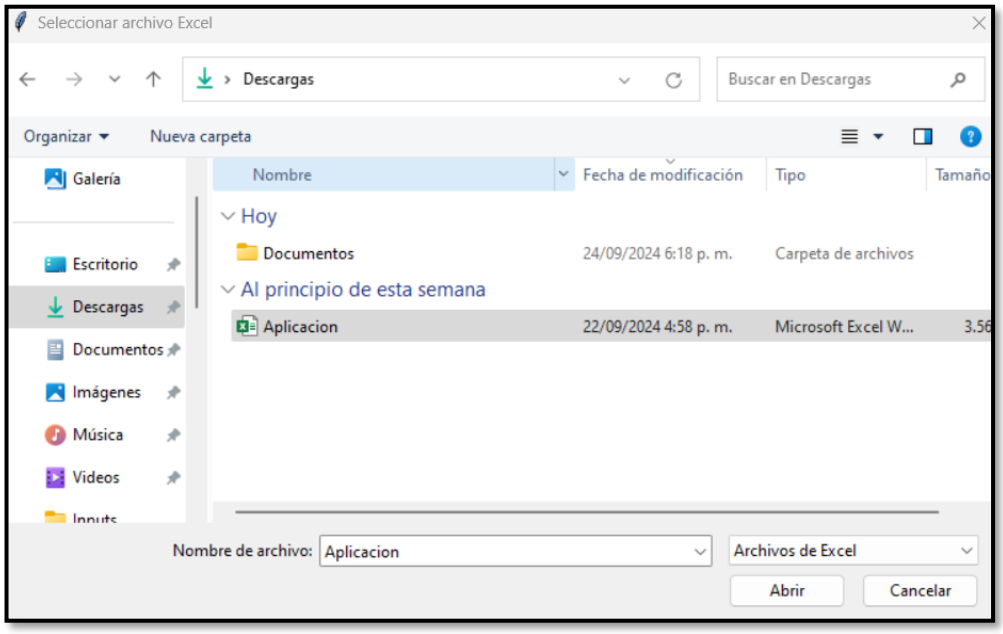


Figura 45. Manual seleccionar archivo 2.

Una vez cargado correctamente el archivo, la aplicación notificará que los datos están listos para ser procesados.

---

2. Seleccionar la Carpeta de Destino

Una vez que el archivo ha sido cargado, el siguiente paso es seleccionar la carpeta donde se guardará el informe generado.

1. Se hace clic en el botón **"Seleccionar carpeta de destino"**.

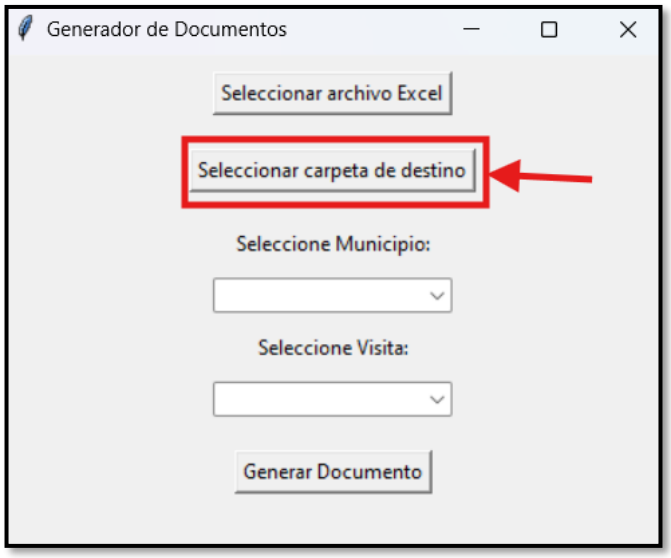


Figura 46. Manual seleccionar carpeta 1.

2. Aparecerá una ventana del explorador de archivos.

3. Se navegará por las carpetas del dispositivo y se selecciona la ubicación deseada.

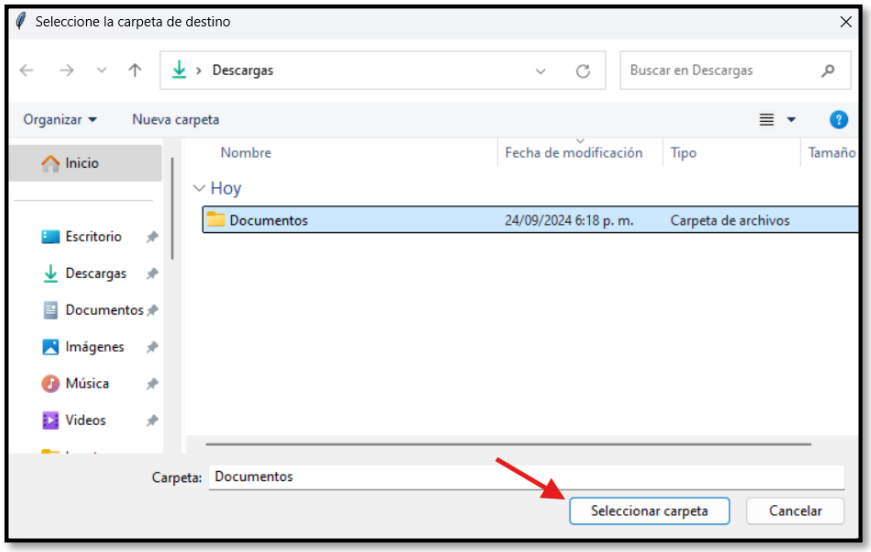


Figura 47. Manual seleccionar carpeta 2.

Recomendaciones

- Se debe elegir una carpeta de fácil acceso para facilitar la localización del documento generado.

- Se debe asegurar de tener permisos de escritura en la carpeta seleccionada.

Después de confirmar la selección, la aplicación recordará esta ubicación para guardar el archivo final.

---

3. Seleccionar el Municipio

El tercer paso consiste en especificar el municipio donde se realizó el mantenimiento.

1. Se hace clic en el desplegable correspondiente.

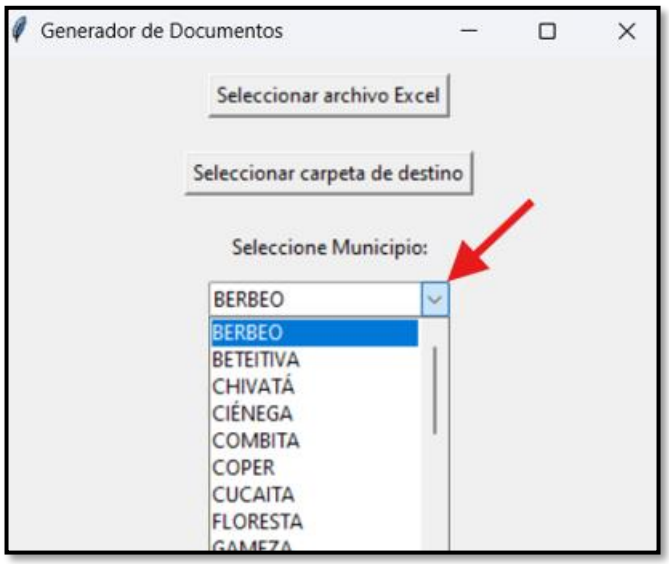


Figura 48. Manual seleccionar municipio.

2. Se vera una lista de municipios disponibles, organizada en orden alfabético.
3. Se debe selecciona la opción adecuada.

Información Adicional

- Esta opción permite una selección rápida y precisa de las entidades con las que la empresa tiene un contrato laboral.
- Si el municipio no aparece en la lista, Se debe comunicar con el administrador para actualizar la base de datos.

4. Seleccionar la Visita

El último campo por completar es la selección de la visita correspondiente.

1. Se hace clic en el botón de la cuarta opción: **"Seleccionar visita"**.

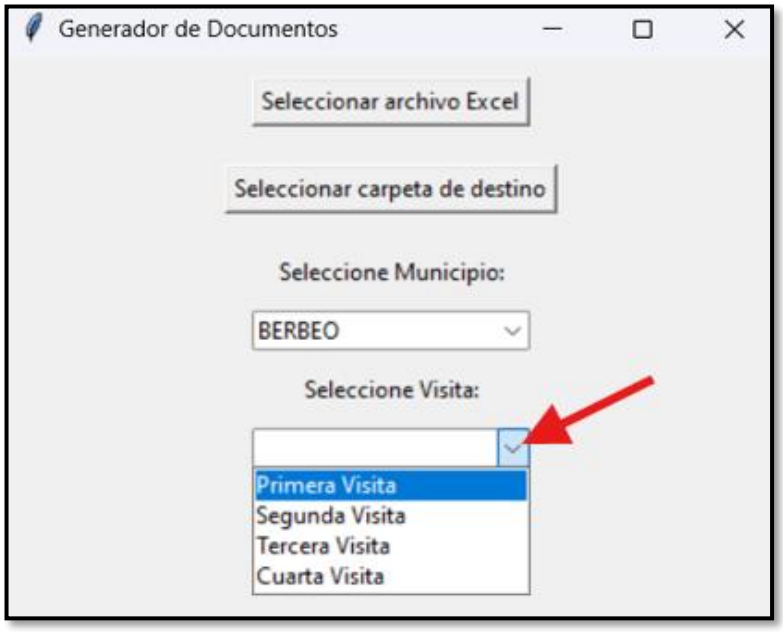


Figura 49. Manual seleccionar visita.

2. Se desplegarán las opciones: **Visita 1, Visita 2, Visita 3 y Visita 4**.
3. Se elige la visita adecuada según el contexto del mantenimiento.

Notas Importantes

- Se debe asegurar de que la selección corresponde al registro en el archivo Excel cargado.
- Una selección incorrecta puede afectar el contenido del informe final.

Generar el Informe Final

Paso Final

Una vez completados todos los campos, el último paso es generar el documento:

1. Se hace clic en el botón **"Generar documento"** en la interfaz principal.

Figura 50. Manual generar documento.

2. La aplicación comenzará a procesar la información cargada y creará un archivo Word o PDF, dependiendo de las configuraciones establecidas.
3. El archivo final será guardado en la carpeta seleccionada en el paso 2.

Verificación y Uso del Documento

- Una vez generado, se debe revisar el documento para asegurar de que toda la información es correcta.
- Si se deben realizar ajustes, se debe volver a abrir la aplicación, corregir los campos necesarios y regenerar el documento.

Resolución de Problemas Comunes

1. El archivo Excel no se carga

- Se asegura de que el archivo tenga la extensión correcta: .xls o .xlsx.
- Se verifica que el archivo no esté abierto en otra aplicación.

2. No se guarda el archivo generado

- Se comprueba que tienes permisos de escritura en la carpeta seleccionada.
- Se verifica que hay suficiente espacio en el disco.

3. Opciones no desplegables

- Se reinicia la aplicación si los menús no responden.
- Se debe asegurar de que la lista de municipios o visitas esté correctamente configurada.

Figura 51. Manual Prueba de un ingeniero.

Anexo: Códigos

Código para la función “informe_mantenimiento”

```
            'Protocolo': limpiar_nan(protocolo_texto),
            'Fecha_v1': pd.to_datetime(row['FECHA_V1']).strftime('%Y
                -%m-%d') if pd.notna(row['FECHA_V1']) else "  ",
            'Imagen_V1': inline_image_v1,
        }
        datos_list.append(datos_dict)
        fecha_vencimiento = None
        fecha_vencimiento = row.get('VENCIMIENTO_DE_GARANTIA', '')
        fecha_vencimiento_dt = convertir_a_fecha(fecha_vencimiento)
        fecha_actual = datetime.now()
        if fecha_vencimiento_dt is None:
            estado_garantia = row['OBSERVACIONES']
        else:
            if fecha_vencimiento_dt > fecha_actual:
                estado_garantia = "Equipo en garantía"
            else:
                estado_garantia = row['OBSERVACIONES']
        HH = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            equipo, 'TIEMPO DE TRABAJO'].values
        if len(HH) > 0:
            HH_texto = HH[0]
        else:
            HH_texto = " "
        funcional = ""
```

Figura 52. Código informe mantenimiento Parte 1.

```
            ESTADO = row['ESTADO']
            if ESTADO == 'Funcional':
                funcional = "Si"
            elif ESTADO == 'No Funcional':
                funcional = "No"
            MAN=row['FIRMA_MTO_1']
            ingeniero = ""
            if MAN == "ING YON":
                ingeniero = "JOHN ALI"
            elif MAN == "ING NEIDER":
                ingeniero = "NEIDER AP"
            new = [row['ID_EQUIPO'],row['FECHA_V1'],row['ID_EQUIPO']
                ,'Mantenimiento Preventivo',HH_texto,HH_texto," "
                ,estado_garantia,funcional,ingeniero,MAN]
            agregar_datos_a_excel(base_de_datos,new)
        datos_dict = {
            'UBICACION': ubicacion,
            'ubic_list': datos_list,
        }
        datos.append(datos_dict)
    context = {
        'Fecha': fecha,
        'Municipio': nombre,
        'Visita': num_visit,
```

Figura 53. Código informe mantenimiento Parte 2.

```
            'datos': datos,
        }
        docx_tpl.render(context)
        titulo = f"Reporte de Mantenimiento {nombre_municipio} {num_visit}
            .docx".upper().replace("_", " ")
        docx_path = os.path.join(output_path, f"{titulo}.docx")
        docx_tpl.save(docx_path)
        convert(docx_path)
    elif num_visit == "Segunda Visita":
        docx_tpl = DocxTemplate(PLANTILLA_MANTENIMIETNO_PREVENTIVO)
        datos = []
        filt_Datos_Visita = inventario[(inventario['ESE'] == nombre_municipio
            )]
        ESE_INIT = esess[(esess['ESE'] == nombre_municipio)]
        nombre = " "
        fecha = " "
        nombre = ESE_INIT.at[0, 'NOMBRE CS']
        fecha = ESE_INIT.at[0, 'Fecha_V2']
        filt_ubicacion_list = sorted(list(filt_Datos_Visita['UBICACIÓN']
            .drop_duplicates()))
        for ubic in range(len(filt_ubicacion_list)):
            datos_list = []
            ubicacion = filt_ubicacion_list[ubic]
```

Figura 54. Código informe mantenimiento Parte 3.

```
filt_Datos_Equipo = filt_Datos_Visita[(filt_Datos_Visita['ESE']
    == nombre_municipio) &
                                      (filt_Datos_Visita['UBICACIÓN'] ==
                                      ubicacion)]
for index, row in filt_Datos_Equipo.iterrows():
    id_equipo = row['ID_EQUIPO']
    equipo = row['EQUIPO']
    datos_dict = {}
    protocolo = caracteristicass.loc[caracteristicass['EQUIPO']
        == equipo, 'PROTOCOLOS'].values
    if len(protocolo) > 0:
        protocolo_texto = protocolo[0]
    else:
        protocolo_texto = ""
    inline_image_v1 = " "
    inline_image_v2 = " "
    inline_images_v1 = " "
    inline_images_v2 = " "
    image_v1 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
        1_URL.png")
    if os.path.exists(image_v1):
        inline_image_v1 = InlineImage(docx_tpl, image_v1, width
            =Mm(50))
```

Figura 55. Código informe mantenimiento Parte 4.

```
            image_v2 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                2_URL.png")
            if os.path.exists(image_v2):
                inline_image_v2 = InlineImage(docx_tpl, image_v2, width
                    =Mm(50))
            datos_dict = {
                'Equipo': limpiar_nan(row['EQUIPO']),
                'Modelo': limpiar_nan(row['MODELO']),
                'Serie': limpiar_nan(row['SERIE']),
                'Ubicacion': limpiar_nan(row['UBICACIÓN']),
                'Observaciones': limpiar_nan(row['OBSERVACIONES']),
                'Marca': limpiar_nan(row['MARCA']),
                'Protocolo': limpiar_nan(protocolo_texto),
                'Fecha_v1': pd.to_datetime(row['FECHA_V1']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V1']) else "Fecha no
                    disponible",
                'Fecha_v2': pd.to_datetime(row['FECHA_V2']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V2']) else "Fecha no
                    disponible",
                'Imagen_V1': inline_image_v1,
                'Imagen_V2': inline_image_v2,
            }
            datos_list.append(datos_dict)
            fecha_vencimiento = None
```

Figura 56. Código informe mantenimiento Parte 5.

```
fecha_vencimiento = row.get('VENCIMIENTO_DE_GARANTIA', '')
fecha_vencimiento_dt = convertir_a_fecha(fecha_vencimiento)
fecha_actual = datetime.now()
if fecha_vencimiento_dt is None:
    estado_garantia = row['OBSERVACIONES']
else:
    if fecha_vencimiento_dt > fecha_actual:
        estado_garantia = "Equipo en garantía"
    else:
        estado_garantia = row['OBSERVACIONES']
fecha_actual = datetime.now()
if fecha_vencimiento_dt is None:
    estado_garantia = row['OBSERVACIONES']
else:
    if fecha_vencimiento_dt > fecha_actual:
        estado_garantia = "Equipo en garantía"
    else:
        estado_garantia = row['OBSERVACIONES']

HH = caracteristicass.loc[caracteristicass['EQUIPO'] ==
    equipo, 'TIEMPO DE TRABAJO'].values
if len(HH) > 0:
    HH_texto = HH[0]
```

Figura 57. Código informe mantenimiento Parte 6.

```
            else:
                HH_texto = ""
            ESTADO = row['ESTADO']
            funcional = " "
            if ESTADO == 'Funcional':
                funcional = "Si"
            elif ESTADO == 'No Funcional':
                funcional = "No"
            ingeniero = " "
            MAN=row['FIRMA_MTO_2']
            if MAN == "ING YON":
                ingeniero = "JOHN ALI"
            elif MAN == "ING NEIDER":
                ingeniero = "NEIDER AP"

            new = [row['ID_EQUIPO'],row['FECHA_V2'],row['ID_EQUIPO']
                ,'Mantenimiento Preventivo',HH_texto,HH_texto," "
                ,estado_garantia,funcional,ingeniero,MAN]
            agregar_datos_a_excel(base_de_datos,new)
        datos_dict = {
            'UBICACION': ubicacion,
            'ubic_list': datos_list,
        }
        datos.append(datos_dict)
```

Figura 58. Código informe mantenimiento Parte 7.

```
        context = {
            'Fecha': fecha,
            'Municipio': nombre,
            'Visita': num_visit,
            'datos': datos,
        }
        docx_tpl.render(context)
        titulo = f"Reporte de Mantenimiento {nombre_municipio} {num_visit}
            .docx".upper().replace("_", " ")
        docx_path = os.path.join(output_path, f"{titulo}.docx")
        docx_tpl.save(docx_path)
        convert(docx_path)
    elif num_visit == "Tercera Visita":
        docx_tpl = DocxTemplate(PLANTILLA_MANTENIMIETNO_PREVENTIVO)
        datos = []
        filt_Datos_Visita = inventario[(inventario['ESE'] == nombre_municipio
            )]
        ESE_INIT = esess[(esess['ESE'] == nombre_municipio)]
        nombre = " "
        fecha = " "
        nombre = ESE_INIT.at[0, 'NOMBRE CS']
        fecha = ESE_INIT.at[0, 'Fecha_V3']
        filt_ubicacion_list = sorted(list(filt_Datos_Visita['UBICACIÓN']
            .drop_duplicates()))
```

Figura 59. Código informe mantenimiento Parte 8.

```
        for ubic in range(len(filt_ubicacion_list)):
            datos_list = []
            ubicacion = filt_ubicacion_list[ubic]
            filt_Datos_Equipo = filt_Datos_Visita[(filt_Datos_Visita['ESE']
                == nombre_municipio) &
                                                  (filt_Datos_Visita['UBICACIÓN'] ==
                                                  ubicacion)]
            for index, row in filt_Datos_Equipo.iterrows():
                id_equipo = row['ID_EQUIPO']
                equipo = row['EQUIPO']
                datos_dict = {}
                protocolo = caracteristicass.loc[caracteristicass['EQUIPO']
                    == equipo, 'PROTOCOLOS'].values
                if len(protocolo) > 0:
                    protocolo_texto = protocolo[0]
                else:
                    protocolo_texto = " "
                inline_image_v1 = " "
                inline_image_v2 = " "
                inline_image_v3 = " "
                inline_images_v1 = " "
                inline_images_v2 = " "
                inline_images_v3 = " "
```

Figura 60. Código informe mantenimiento Parte 9.

```
            image_v1 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                1_URL.png")
            if os.path.exists(image_v1):
                inline_image_v1 = InlineImage(docx_tpl, image_v1, width
                    =Mm(50))
            image_v2 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                2_URL.png")
            if os.path.exists(image_v2):
                inline_image_v2 = InlineImage(docx_tpl, image_v2, width
                    =Mm(50))
            image_v3 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                3_URL.png")
            if os.path.exists(image_v3):
                inline_image_v3 = InlineImage(docx_tpl, image_v3, width
                    =Mm(50))
            datos_dict = {
                'Equipo': limpiar_nan(row['EQUIPO']),
                'Modelo': limpiar_nan(row['MODELO']),
                'Serie': limpiar_nan(row['SERIE']),
                'Ubicacion': limpiar_nan(row['UBICACIÓN']),
                'Observaciones': limpiar_nan(row['OBSERVACIONES']),
                'Marca': limpiar_nan(row['MARCA']),
                'Protocolo': limpiar_nan(protocolo_texto),
                'Fecha_v1': pd.to_datetime(row['FECHA_V1']).strftime('%Y
```

Figura 61. Código informe mantenimiento Parte 10.

```
                -%m-%d') if pd.notna(row['FECHA_V1']) else "Fecha no
                    disponible",
                'Fecha_v2': pd.to_datetime(row['FECHA_V2']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V2']) else "Fecha no
                    disponible",
                'Fecha_v3': pd.to_datetime(row['FECHA_V3']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V3']) else "Fecha no
                    disponible",
                'Imagen_V1': inline_image_v1,
                'Imagen_V2': inline_image_v2,
                'Imagen_V3': inline_image_v3,
            }
            datos_list.append(datos_dict)
            fecha_vencimiento = None
            fecha_vencimiento = row.get('VENCIMIENTO_DE_GARANTIA', '')
            fecha_vencimiento_dt = convertir_a_fecha(fecha_vencimiento)
            fecha_actual = datetime.now()
            if fecha_vencimiento_dt is None:
                estado_garantia = row['OBSERVACIONES']
            else:
                if fecha_vencimiento_dt > fecha_actual:
                    estado_garantia = "Equipo en garantía"
                else:
                    estado_garantia = row['OBSERVACIONES']
```

Figura 62. Código informe mantenimiento Parte 11.

```
                fecha_actual = datetime.now()
                if fecha_vencimiento_dt is None:
                    estado_garantia = row['OBSERVACIONES']
                else:
                    if fecha_vencimiento_dt > fecha_actual:
                        estado_garantia = "Equipo en garantía"
                    else:
                        estado_garantia = row['OBSERVACIONES']

                HH = caracteristicass.loc[caracteristicass['EQUIPO'] ==
                    equipo, 'TIEMPO DE TRABAJO'].values
                if len(HH) > 0:
                    HH_texto = HH[0]
                else:
                    HH_texto = " "
                ESTADO = row['ESTADO']
                funcional = " "
                if ESTADO == 'Funcional':
                    funcional = "Si"
                elif ESTADO == 'No Funcional':
                    funcional = "No"
                ingeniero = " "
                MAN=row['FIRMA_MTO_3']
                if MAN == "ING YON":
```

Figura 63. Código informe mantenimiento Parte 12.

```
                ingeniero = "JOHN ALI"
            elif MAN == "ING NEIDER":
                ingeniero = "NEIDER AP"

            new = [row['ID_EQUIPO'],row['FECHA_V3'],row['ID_EQUIPO']
                ,'Mantenimiento Preventivo',HH_texto,HH_texto," "
                ,estado_garantia,funcional,ingeniero,MAN]
            agregar_datos_a_excel(base_de_datos,new)
        datos_dict = {
            'UBICACION': ubicacion,
            'ubic_list': datos_list,
        }
        datos.append(datos_dict)


    context = {
        'Fecha': fecha,
        'Municipio': nombre,
        'Visita': num_visit,
        'datos': datos,
    }
    docx_tpl.render(context)
    titulo = f"Reporte de Mantenimiento {nombre_municipio} {num_visit}
        .docx".upper().replace("_", " ")
```

Figura 64. Código informe mantenimiento Parte 13.

```
        docx_path = os.path.join(output_path, f"{titulo}.docx")
        docx_tpl.save(docx_path)
        convert(docx_path)
    elif num_visit == "Cuarta Visita":
        docx_tpl = DocxTemplate(PLANTILLA_MANTENIMIETNO_PREVENTIVO)
        datos = []
        filt_Datos_Visita = inventario[(inventario['ESE'] == nombre_municipio
            )]
        ESE_INIT = esess[(esess['ESE'] == nombre_municipio)]
        nombre = " "
        fecha = " "
        nombre = ESE_INIT.at[0, 'NOMBRE CS']
        fecha = ESE_INIT.at[0, 'Fecha_V1']
        filt_ubicacion_list = sorted(list(filt_Datos_Visita['UBICACIÓN']
            .drop_duplicates()))
        for ubic in range(len(filt_ubicacion_list)):
            datos_list = []
            ubicacion = filt_ubicacion_list[ubic]
            filt_Datos_Equipo = filt_Datos_Visita[(filt_Datos_Visita['ESE']
                == nombre_municipio) &
                (filt_Datos_Visita['UBICACIÓN'] == ubicacion)]
            for index, row in filt_Datos_Equipo.iterrows():
                id_equipo = row['ID_EQUIPO']
                equipo = row['EQUIPO']
```

Figura 65. Código informe mantenimiento Parte 14.

```
                datos_dict = {}
                protocolo = caracteristicass.loc[caracteristicass['EQUIPO']
                    == equipo, 'PROTOCOLOS'].values
                if len(protocolo) > 0:
                    protocolo_texto = protocolo[0]
                else:
                    protocolo_texto = " "
                inline_image_v1 = " "
                inline_image_v2 = " "
                inline_image_v3 = " "
                inline_image_v4 = " "
                inline_images_v1 = " "
                inline_images_v2 = " "
                inline_images_v3 = " "
                inline_images_v4 = " "
                image_v1 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                    1_URL.png")
                if os.path.exists(image_v1):
                    inline_image_v1 = InlineImage(docx_tpl, image_v1, width
                        =Mm(50))
                image_v2 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                    2_URL.png")
                if os.path.exists(image_v2):
                    inline_image_v2 = InlineImage(docx_tpl, image_v2, width
```

Figura 66. Código informe mantenimiento Parte 15.

```
            image_v3 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                3_URL.png")
            if os.path.exists(image_v3):
                inline_image_v3 = InlineImage(docx_tpl, image_v3, width=Mm
                    (50))
            image_v4 = os.path.join(IMAGENES_URL, f"{id_equipo}_VISITA
                4_URL.png")
            if os.path.exists(image_v4):
                inline_image_v4 = InlineImage(docx_tpl, image_v4, width=Mm
                    (50))
            datos_dict = {
                'Equipo': limpiar_nan(row['EQUIPO']),
                'Modelo': limpiar_nan(row['MODELO']),
                'Serie': limpiar_nan(row['SERIE']),
                'Ubicacion': limpiar_nan(row['UBICACIÓN']),
                'Observaciones': limpiar_nan(row['OBSERVACIONES']),
                'Marca': limpiar_nan(row['MARCA']),
                'Protocolo': limpiar_nan(protocolo_texto),
                'Fecha_v1': pd.to_datetime(row['FECHA_V1']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V1']) else "Fecha no
                    disponible",
                'Fecha_v2': pd.to_datetime(row['FECHA_V2']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V2']) else "Fecha no
                    disponible",
```

Figura 67. Código informe mantenimiento Parte 16.

```
                'Fecha_v3': pd.to_datetime(row['FECHA_V3']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V3']) else "Fecha no
                    disponible",
                'Fecha_v4': pd.to_datetime(row['FECHA_V4']).strftime('%Y
                    -%m-%d') if pd.notna(row['FECHA_V4']) else "Fecha no
                    disponible",
                'Imagen_V1': inline_image_v1,
                'Imagen_V2': inline_image_v2,
                'Imagen_V3': inline_image_v3,
                'Imagen_V4': inline_image_v4,
            }
            datos_list.append(datos_dict)
            fecha_vencimiento = None
            fecha_vencimiento = row.get('VENCIMIENTO_DE_GARANTIA', '')
            fecha_vencimiento_dt = convertir_a_fecha(fecha_vencimiento)
            fecha_actual = datetime.now()
            if fecha_vencimiento_dt is None:
                estado_garantia = row['OBSERVACIONES']
            else:
                if fecha_vencimiento_dt > fecha_actual:
                    estado_garantia = "Equipo en garantía"
                else:
                    estado_garantia = row['OBSERVACIONES']
            fecha_actual = datetime.now()
```

Figura 68. Código informe mantenimiento Parte 17.

```
            if fecha_vencimiento_dt is None:
                estado_garantia = row['OBSERVACIONES']
            else:
                if fecha_vencimiento_dt > fecha_actual:
                    estado_garantia = "Equipo en garantía"
                else:
                    estado_garantia = row['OBSERVACIONES']
            HH = caracteristicass.loc[caracteristicass['EQUIPO'] == equipo
                , 'TIEMPO DE TRABAJO'].values
            if len(HH) > 0:
                HH_texto = HH[0]
            else:
                HH_texto = " "
            ESTADO = row['ESTADO']
            funcional = " "
            if ESTADO == 'Funcional':
                funcional = "Si"
            elif ESTADO == 'No Funcional':
                funcional = "No"
            ingeniero = " "
            MAN=row['FIRMA_MTO_4']
            if MAN == "ING YON":
                ingeniero = "JOHN ALI"
            elif MAN == "ING NEIDER":
```

Figura 69. Código informe mantenimiento Parte 18.

```
            elif MAN == "ING NEIDER":
                ingeniero = "NEIDER AP"
            new = [row['ID_EQUIPO'],row['FECHA_V4'],row['ID_EQUIPO']
                ,'Mantenimiento Preventivo',HH_texto,HH_texto," "
                ,estado_garantia,funcional,ingeniero,MAN]
            agregar_datos_a_excel(base_de_datos,new)
        datos_dict = {
            'UBICACION': ubicacion,
            'ubic_list': datos_list,
        }
        datos.append(datos_dict)
    context = {
        'Fecha': fecha,
        'Municipio': nombre,
        'Visita': num_visit,
        'datos': datos,
    }
    docx_tpl.render(context)
    titulo = f"Reporte de Mantenimiento {nombre_municipio} {num_visit}
        .docx".upper().replace("_", " ")
    docx_path = os.path.join(output_path, f"{titulo}.docx")
    docx_tpl.save(docx_path)
    convert(docx_path)
```

Figura 70. Código informe mantenimiento Parte 19.

Código para la función “hoja_vida”

```
            ing4 = row['FIRMA_MTO_4']
            ID_EQ = row['ID_EQUIPO']
            nom1 = " "
            cc1 = " "
            nom1 = row['NOMBRE_V1']
            cc1 = row['CC_V1']
            nom2 = " "
            cc2 = " "
            nom2 = row['NOMBRE_V2']
            cc2 = row['CC_V2']
            nom3 = " "
            cc3 = " "
            nom3 = row['NOMBRE_V3']
            cc3 = row['CC_V3']
            nom4 = " "
            cc4 = " "
            nom4 = row['NOMBRE_V4']
            cc4 = row['CC_V4']
            fch1 = pd.to_datetime(row['FECHA_V1']).strftime('%Y-%m-%d') if pd
                .notna(
                row['FECHA_V1']) else ""
            fch2 = pd.to_datetime(row['FECHA_V2']).strftime('%Y-%m-%d') if pd
                .notna(
                row['FECHA_V2']) else ""
```

Figura 71. Código hoja vida Parte 1.

```
        fch3 = pd.to_datetime(row['FECHA_V3']).strftime('%Y-%m-%d') if pd
            .notna(
            row['FECHA_V3']) else ""
        fch4 = pd.to_datetime(row['FECHA_V4']).strftime('%Y-%m-%d') if pd
            .notna(
            row['FECHA_V4']) else ""
        EQUIPO = row['EQUIPO']
        protocolo = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'PROTOCOLOS'].values
        if len(protocolo) > 0:
            protocolo_texto = protocolo[0]
        else:
            protocolo_texto = ""
        recomendaciones = caracteristicass.loc[caracteristicass['EQUIPO']
            == EQUIPO, 'RECOMENDACIONES DEL FABRICANTE'].values
        if len(recomendaciones) > 0:
            recomendaciones_texto = recomendaciones[0]
        else:
            recomendaciones_texto = ""
        id_equipo = row['ID_EQUIPO']
        MARCA = row['MARCA']
        MODELO = row['MODELO']
        SERIE = row['SERIE']
        INV_ACT = row['INVENTARIO']
```

Figura 72. Código hoja vida Parte 2.

```
        UBICACION = row['UBICACIÓN']
        REGISTRO_SANITARIO =row['REGISTRO_SANITARIO']
        SERVICIO =servicio.loc[servicio['UBICACION'] == UBICACION,
            'SERVICIO'].values
        if len(SERVICIO) > 0:
            SERVICIO_texto = SERVICIO[0]
        else:
            SERVICIO_texto = ""
        MF = caracteristicass.loc[caracteristicass['EQUIPO'] == EQUIPO,
            'MOVIL/FIJO'].values
        if len(MF) > 0:
            MF_texto = MF[0]
        else:
            MF_texto = " "
        if MF_texto == 'FIJO':
            FIJO = "X"
            MOVIL = " "
        elif MF_texto == 'MOVIL':
            FIJO = " "
            MOVIL = "X"
        else:
            FIJO = " "
            MOVIL = " "

```

Figura 73. Código hoja vida Parte 3.

```
        inline_image_eq = " "
        image_eq = os.path.join(IMAGENES_URL, f"{id_equipo}_FOTO_URL.png"
            )
        if os.path.exists(image_eq):
            inline_image_eq = InlineImage(docx_tpl, image_eq, width=Mm(30
                ))
        FABRICANTE= row['FABRICANTE']
        Fecha_de_compra = row['FECHA_DE_COMPRA']
        Fecha_de_instalacion = row['FRECHA_DE_INSTALACION']
        Vencimiento_garantia = row['VENCIMIENTO_DE_GARANTIA']
        CB = caracteristicass.loc[caracteristicass['EQUIPO'] == EQUIPO,
            'CLASIFICACION BIOMEDICA'].values
        if len(CB) > 0:
            CB_texto = CB[0]
        else:
            CB_texto = " "
        if CB_texto == 'ANALISIS DE LABORATORIO':
            ANALIS = "X"
            PREVEN = " "
            DIAG = " "
            REHA = " "
            TRATA = " "
        elif CB_texto == 'PREVENCION':
            ANALIS = " "
```

Figura 74. Código hoja vida Parte 4.

```
            PREVEN = "X"
            DIAG = " "
            REHA = " "
            TRATA = " "
        elif CB_texto == 'DIAGNOSTICO':
            ANALIS = " "
            PREVEN = " "
            DIAG = "X"
            REHA = " "
            TRATA = " "
        elif CB_texto == 'REHABILITACION':
            ANALIS = " "
            PREVEN = " "
            DIAG = " "
            REHA = "X"
            TRATA = " "
        elif CB_texto == 'TRATAMIENTO Y MANTENIMIENTO DE LA VIDA':
            ANALIS = " "
            PREVEN = " "
            DIAG = " "
            REHA = " "
            TRATA = "X"
        else :
            ANALIS = " "
```

Figura 75. Código hoja vida Parte 5.

```
            PREVEN = " "
            DIAG = " "
            REHA = " "
            TRATA = " "
        NR = caracteristicass.loc[caracteristicass['EQUIPO'] == EQUIPO,
            'NIVEL DE RIESGO'].values
        if len(NR) > 0:
            NR_texto = NR[0]
        else:
            NR_texto = " "
        if NR_texto == 'I':
            NR_I = "X"
            NR_IIA = " "
            NR_IIB = " "
            NR_III = " "
        elif NR_texto == 'IIA':
            NR_I = " "
            NR_IIA = "X"
            NR_IIB = " "
            NR_III = " "
        elif NR_texto == 'IIB':
            NR_I = " "
            NR_IIA = " "
            NR_IIB = "X"
```

Figura 76. Código hoja vida Parte 6.

```
            NR_III = " "
        elif NR_texto == 'III':
            NR_I = " "
            NR_IIA = " "
            NR_IIB = " "
            NR_III = "X"
        else:
            NR_I = " "
            NR_IIA = " "
            NR_IIB = " "
            NR_III = " "
        Fuente = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'FUENTE DE ALIMETNACION'].values
        if len(Fuente) > 0:
            Fuente_texto = Fuente[0]
        else:
            Fuente_texto = ""
        TEC = caracteristicass.loc[caracteristicass['EQUIPO'] == EQUIPO,
            'TEC, PREDOMINANTE'].values
        if len(TEC) > 0:
            TEC_texto = TEC[0]
        else:
            TEC_texto = ""
```

Figura 77. Código hoja vida Parte 7.

```
        VOLT_MAX_MIN = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'VOLTAJE MAX- MIN'].values
        if len(VOLT_MAX_MIN) > 0:
            VOLT_MAX_MIN_texto = VOLT_MAX_MIN[0]
        else:
            VOLT_MAX_MIN_texto = ""
        CORR_MAX_MIN = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'CORRIENTE MAX- MIN'].values
        if len(CORR_MAX_MIN) > 0:
            CORR_MAX_MIN_texto = CORR_MAX_MIN[0]
        else:
            CORR_MAX_MIN_texto = ""
        HUMEDAD = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'HUMEDAD'].values
        if len(HUMEDAD) > 0:
            HUMEDAD_texto = HUMEDAD[0]
        else:
            HUMEDAD_texto = ""
        POTENCIA = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'POTENCIA'].values
        if len(POTENCIA) > 0:
            POTENCIA_texto = POTENCIA[0]
        else:
```

Figura 78. Código hoja vida Parte 8.

```
            POTENCIA_texto = ""
        PRESION = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'PRESION'].values
        if len(PRESION) > 0:
            PRESION_texto = PRESION[0]
        else:
            PRESION_texto = ""
        FRECUENCIA = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'FRECUENCIA'].values
        if len(FRECUENCIA) > 0:
            FRECUENCIA_texto = FRECUENCIA[0]
        else:
            FRECUENCIA_texto = ""
        TEMPERATURA = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'TEMPERATURA'].values
        if len(TEMPERATURA) > 0:
            TEMPERATURA_texto = TEMPERATURA[0]
        else:
            TEMPERATURA_texto = ""
        PESO = caracteristicass.loc[caracteristicass['EQUIPO'] == EQUIPO,
            'PESO'].values
        if len(PESO) > 0:
            PESO_texto = PESO[0]
        else:
```

Figura 79. Código hoja vida Parte 9.

```
            PESO_texto = ""
        VELOCIDAD = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'VELOCIDAD'].values
        if len(VELOCIDAD) > 0:
            VELOCIDAD_texto = VELOCIDAD[0]
        else:
            VELOCIDAD_texto = ""
        OTROS_P3 = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'OTROS_P3'].values
        if len(OTROS_P3) > 0:
            OTROS_P3_texto = OTROS_P3[0]
        else:
            OTROS_P3_texto = ""
        RANGO_VOLTAJE = caracteristicass.loc[caracteristicass['EQUIPO']
            == EQUIPO, 'RANGO DE VOLTAJE'].values
        if len(RANGO_VOLTAJE) > 0:
            RANGO_VOLTAJE_texto = RANGO_VOLTAJE[0]
        else:
            RANGO_VOLTAJE_texto = ""
        RANGO_CORRIENTE = caracteristicass.loc[caracteristicass['EQUIPO']
            == EQUIPO, 'RANGO DE CORRIENTE'].values
        if len(RANGO_CORRIENTE) > 0:
```

Figura 80. Código hoja vida Parte 10.

```
        if len(RANGO_PESO) > 0:
            RANGO_PESO_texto = RANGO_PESO[0]
        else:
            RANGO_PESO_texto = ""
        RANGO_TEMPERATURA = caracteristicass
            .loc[caracteristicass['EQUIPO'] == EQUIPO, 'RANGO DE
            TEMPERATURA '].values
        if len(RANGO_TEMPERATURA) > 0:
            RANGO_TEMPERATURA_texto = RANGO_TEMPERATURA[0]
        else:
            RANGO_TEMPERATURA_texto = ""
        RANGO_HUMEDAD = caracteristicass.loc[caracteristicass['EQUIPO']
            == EQUIPO, 'RANGO DE HUMEDAD'].values
        if len(RANGO_HUMEDAD) > 0:
            RANGO_HUMEDAD_texto = RANGO_HUMEDAD[0]
        else:
            RANGO_HUMEDAD_texto = ""
        OTROS_P4 = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'OTROS_P4'].values
        if len(OTROS_P4) > 0:
            OTROS_P4_texto = OTROS_P4[0]
        else:
            OTROS_P4_texto = ""

```

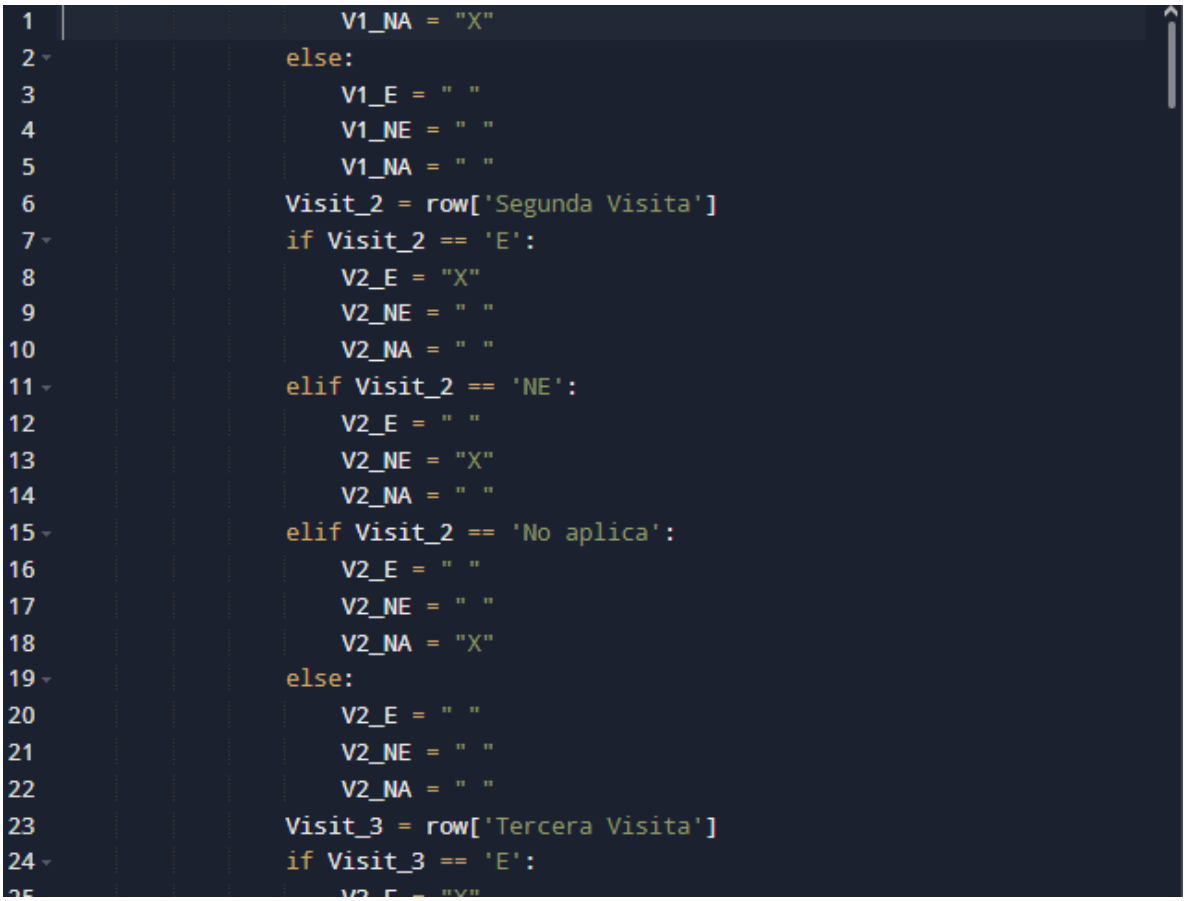
Figura 81. Código hoja vida Parte 11.

```
        CALIBRACION = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'CALIBRACION'].values
        if len(CALIBRACION) > 0:
            CALIBRACION_texto = CALIBRACION[0]
        else:
            CALIBRACION_texto = " "
        if CALIBRACION_texto == 'X':
            CA_SI = "X"
            CA_NO = " "
        else:
            CA_SI = " "
            CA_NO = "X"
        PERIOCIDAD_texto = " "
        PERIOCIDAD = caracteristicass.loc[caracteristicass['EQUIPO'] ==
            EQUIPO, 'PERIODICIDAD'].values
        if len(PERIOCIDAD) > 0:
            PERIOCIDAD_texto = PERIOCIDAD[0]
        else:
            PERIOCIDAD_texto = ""
        items_componentes = componentess[componentess['EQUIPO'] ==
            EQUIPO]
        lista_items_componentes = []
        contador = 0
        for idx, item row in items componentes.iterrows():
```

Figura 82. Código hoja vida Parte 12.

```
            letra = chr(65 + contador)
            item_componente = {
                'DESCRIPCION': item_row['DESCRIPCION'],
                'CANTIDAD': item_row['CANTIDAD'],
                'LETRA': letra,
            }
            lista_items_componentes.append(item_componente)
            contador += 1
        items_chequeo = lista_chequeo[lista_chequeo['ID_EQUIPO'] ==
            id_equipo]
        lista_items_chequeo = []
        for idx, row in items_chequeo.iterrows():
            Visit_1 = row['Primera Visita']
            if Visit_1 == 'E':
                V1_E = "X"
                V1_NE = " "
                V1_NA = " "
            elif Visit_1 == 'NE':
                V1_E = " "
                V1_NE = "X"
                V1_NA = " "
            elif Visit_1 == 'No aplica':
                V1_E = " "
                V1_NE = " "
```

Figura 83. Código hoja vida Parte 13.

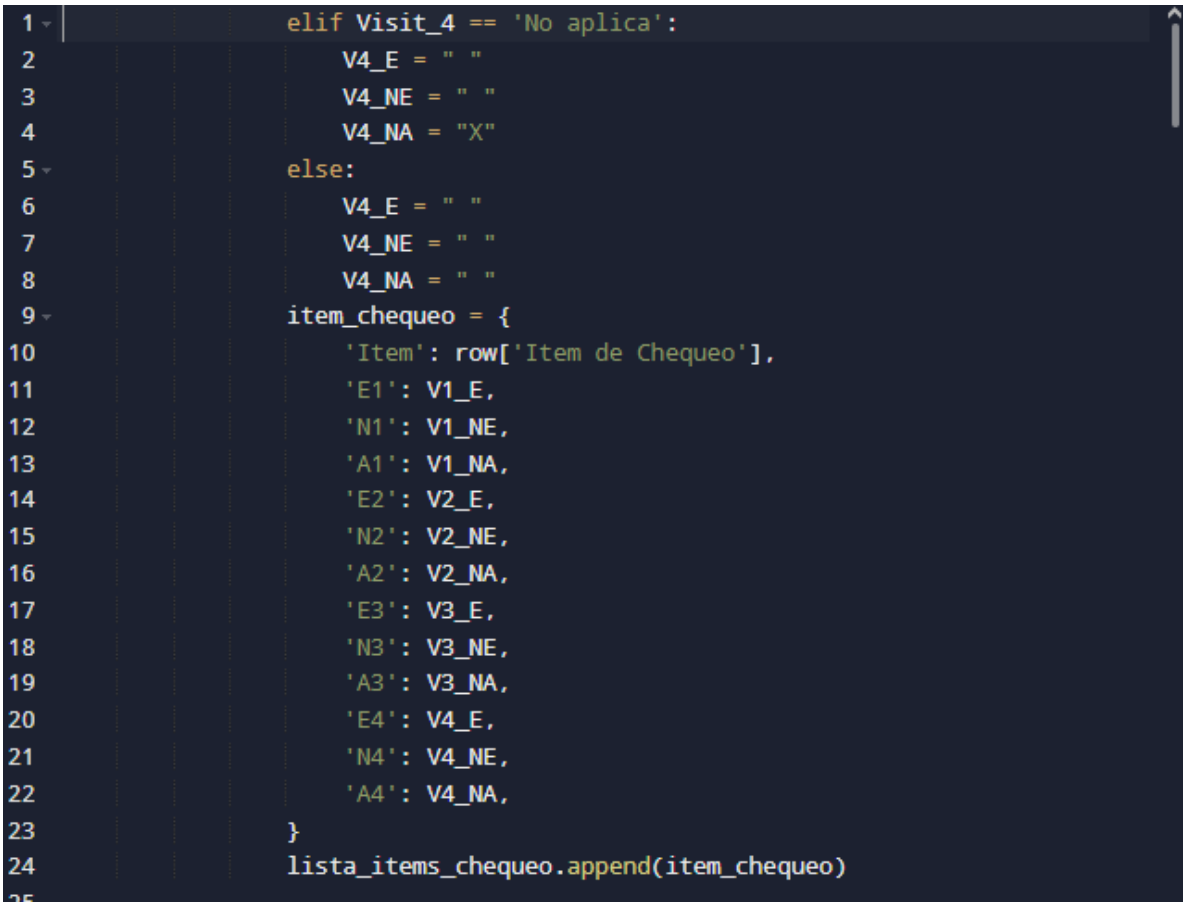


```
                V1_NA = "X"
            else:
                V1_E = " "
                V1_NE = " "
                V1_NA = " "
            Visit_2 = row['Segunda Visita']
            if Visit_2 == 'E':
                V2_E = "X"
                V2_NE = " "
                V2_NA = " "
            elif Visit_2 == 'NE':
                V2_E = " "
                V2_NE = "X"
                V2_NA = " "
            elif Visit_2 == 'No aplica':
                V2_E = " "
                V2_NE = " "
                V2_NA = "X"
            else:
                V2_E = " "
                V2_NE = " "
                V2_NA = " "
            Visit_3 = row['Tercera Visita']
            if Visit_3 == 'E':
```

Figura 84. Código hoja vida Parte 14.

```
                V3_E = "X"
                V3_NE = " "
                V3_NA = " "
            elif Visit_3 == 'NE':
                V3_E = " "
                V3_NE = "X"
                V3_NA = " "
            elif Visit_3 == 'No aplica':
                V3_E = " "
                V3_NE = " "
                V3_NA = "X"
            else:
                V3_E = " "
                V3_NE = " "
                V3_NA = " "
            Visit_4 = row['Cuarta Visita']
            if Visit_4 == 'E':
                V4_E = "X"
                V4_NE = " "
                V4_NA = " "
            elif Visit_4 == 'NE':
                V4_E = " "
                V4_NE = "X"
                V4_NA = " "
```

Figura 85. Código hoja vida Parte 15.



```
            elif Visit_4 == 'No aplica':
                V4_E = " "
                V4_NE = " "
                V4_NA = "X"
            else:
                V4_E = " "
                V4_NE = " "
                V4_NA = " "
            item_chequeo = {
                'Item': row['Item de Chequeo'],
                'E1': V1_E,
                'N1': V1_NE,
                'A1': V1_NA,
                'E2': V2_E,
                'N2': V2_NE,
                'A2': V2_NA,
                'E3': V3_E,
                'N3': V3_NE,
                'A3': V3_NA,
                'E4': V4_E,
                'N4': V4_NE,
                'A4': V4_NA,
            }
            lista_items_chequeo.append(item_chequeo)
```

Figura 86. Código hoja vida Parte 16.

```
        inline_image_f1 = " "
        inline_image_f2 = " "
        inline_image_f3 = " "
        inline_image_f4 = " "
        image_f1 = os.path.join(IMAGENES_URL, f"{id_equipo}_FIRMA V1_URL
            .png")
        if os.path.exists(image_f1):
            inline_image_f1 = InlineImage(docx_tpl, image_f1, width=Mm(25
                ))
        image_f2 = os.path.join(IMAGENES_URL, f"{id_equipo}_FIRMA V2_URL
            .png")
        if os.path.exists(image_f2):
            inline_image_f2 = InlineImage(docx_tpl, image_f2, width=Mm(25
                ))
        image_f3 = os.path.join(IMAGENES_URL, f"{id_equipo}_FIRMA V3_URL
            .png")
        if os.path.exists(image_f3):
            inline_image_f3 = InlineImage(docx_tpl, image_f3, width=Mm(25
                ))
        image_f4 = os.path.join(IMAGENES_URL, f"{id_equipo}_FIRMA V4_URL
            .png")
        if os.path.exists(image_f4):
            inline_image_f4 = InlineImage(docx_tpl, image_f4, width=Mm(25
                ))
```

Figura 87. Código hoja vida Parte 17.

```
        inline_image_ing1 = " "
        inline_image_ing2 = " "
        inline_image_ing3 = " "
        inline_image_ing4 = " "
        image_ing1 = os.path.join(IMAGENES_EXTRAIDAS, f"{ing1}.png")
        if os.path.exists(image_ing1):
            inline_image_ing1 = InlineImage(docx_tpl, image_ing1, width
                =Mm(25))
        image_ing2 = os.path.join(IMAGENES_EXTRAIDAS, f"{ing2}.png")
        if os.path.exists(image_ing2):
            inline_image_ing2 = InlineImage(docx_tpl, image_ing2, width
                =Mm(25))
        image_ing3 = os.path.join(IMAGENES_EXTRAIDAS, f"{ing3}.png")
        if os.path.exists(image_ing3):
            inline_image_ing3 = InlineImage(docx_tpl, image_ing3, width
                =Mm(25))
        image_ing4 = os.path.join(IMAGENES_EXTRAIDAS, f"{ing4}.png")
        if os.path.exists(image_ing4):
            inline_image_ing4 = InlineImage(docx_tpl, image_ing4, width
                =Mm(25))
        Historial_df = " "
        Historial_df = pd.read_excel(base_de_datos,sheet_name='Sheet')
        base = " "
```

Figura 88. Código hoja vida Parte 18.

```
        base=['ID_EQUIPO','FECHA','PROTOCOLO','ACTIVIDAD','H_H','H_P'
            ,'REPUESTOS','OBSERVACIONES','EQUIPO_ACTIVO','NOMBRE'
            ,'FIRMA']
        Data_Hist = " "
        Data_Hist = Historial_df.loc[Historial_df['ID_EQUIPO'] == ID_EQ,
            base]
        lista_items_historial = []
        for index, row in Data_Hist.iterrows():
            FM = row['FIRMA']
            Fecha = pd.to_datetime(row['FECHA']).strftime('%Y-%m-%d') if
                pd.notna(
            row['FECHA']) else "Fecha no disponible"
            inline_image_FM = ""
            image_FM = os.path.join(IMAGENES_EXTRAIDAS, f"{FM}.png")
            if os.path.exists(image_FM):
                inline_image_FM = InlineImage(docx_tpl, image_FM, width
                    =Mm(30))
            item_historial = {
                'F': Fecha if pd.notna(Fecha) else "",
                'P': row['PROTOCOLO'] if pd.notna(row['PROTOCOLO']) else
                    "",
                'A': row['ACTIVIDAD'] if pd.notna(row['ACTIVIDAD']) else
                    "",
                'H': row['H_P'] if pd.notna(row['H_P']) else "",
```

Figura 89. Código hoja vida Parte 19.

```
                'HP': row['H_P'] if pd.notna(row['H_P']) else "",
                'R': row['REPUESTOS'] if pd.notna(row['REPUESTOS']) else
                    "",
                'O': row['OBSERVACIONES'] if pd.notna
                    (row['OBSERVACIONES']) else "",
                'E': row['EQUIPO_ACTIVO'] if pd.notna
                    (row['EQUIPO_ACTIVO']) else "",
                'N': row['NOMBRE'] if pd.notna(row['NOMBRE']) else "",
                'I': inline_image_FM
            }
            lista_items_historial.append(item_historial)
        datos_dict = {
            'LOGO_IN': inline_image_logo1,
            'LOGO_IN1': inline_image_logo,
            'Equipo': limpiar_nan(EQUIPO),
            'Recomen': limpiar_nan(recomendaciones_texto),
            'Proto': limpiar_nan(protocolo_texto),
            'Marca': limpiar_nan(MARCA),
            'Modelo': limpiar_nan(MODELO),
            'Serie': limpiar_nan(SERIE),
            'Inven': limpiar_nan(INV_ACT),
            'Ubicacion': limpiar_nan(UBICACION),
            'Registro': limpiar_nan(REGISTRO_SANITARIO),
            'Servicio': limpiar_nan(SERVICIO_texto),
```

Figura **90.** Código hoja vida Parte 20.

```
            'Movil': limpiar_nan(MOVIL),
            'Fijo': limpiar_nan(FIJO),
            'Imagen_Eq': inline_image_eq,
            'Muni': limpiar_nan(nombre),
            'Fabricante': limpiar_nan(FABRICANTE),
            'Fecha_compra': limpiar_nan(Fecha_de_compra),
            'Fecha_inst': limpiar_nan(Fecha_de_instalacion),
            'Venci_garant': limpiar_nan(Vencimiento_garantia),
            'Alimen': limpiar_nan(Fuente_texto),
            'Tecnologia': limpiar_nan(TEC_texto),
            'VOLT_MM': limpiar_nan(VOLT_MAX_MIN_texto),
            'CORR_MM': limpiar_nan(CORR_MAX_MIN_texto),
            'HUMEDAD': limpiar_nan(HUMEDAD_texto),
            'POTENCIA': limpiar_nan(POTENCIA_texto),
            'PRESION': limpiar_nan(PRESION_texto),
            'FRECUENCIA': limpiar_nan(FRECUENCIA_texto),
            'TEMPERATURA': limpiar_nan(TEMPERATURA_texto),
            'PESO': limpiar_nan(PESO_texto),
            'VELOCIDAD': limpiar_nan(VELOCIDAD_texto),
            'OTROS_P3': limpiar_nan(OTROS_P3_texto),
            'Rang_Vol': limpiar_nan(RANGO_VOLTAJE_texto),
            'Rang_Cor': limpiar_nan(RANGO_CORRIENTE_texto),
            'Rang_Fre': limpiar_nan(RANGO_FRECUENCIA_texto),
            'Rang_Pre': limpiar_nan(RANGO_PRESION_texto),
```

Figura 91. Código hoja vida Parte 21.

```
            'Rang_Pot': limpiar_nan(RANGO_POTENCIA_texto),
            'Rang_Pe': limpiar_nan(RANGO_PESO_texto),
            'Rang_Te': limpiar_nan(RANGO_TEMPERATURA_texto),
            'Rang_Hu': limpiar_nan(RANGO_HUMEDAD_texto),
            'OTROS_P4': limpiar_nan(OTROS_P4_texto),
            'ANALIS': limpiar_nan(ANALIS),
            'PREVEN': limpiar_nan(PREVEN),
            'DIAG': limpiar_nan(DIAG),
            'REHA': limpiar_nan(REHA),
            'TRATA': limpiar_nan(TRATA),
            'NR_I': limpiar_nan(NR_I),
            'NR_IIA': limpiar_nan(NR_IIA),
            'NR_IIB': limpiar_nan(NR_IIB),
            'NR_III': limpiar_nan(NR_III),
            'Componentes': lista_items_componentes,
            'CA_SI': limpiar_nan(CA_SI),
            'CA_NO': limpiar_nan(CA_NO),
            'PERIOCIDAD': limpiar_nan(PERIOCIDAD_texto),
            'ItemsChequeo': lista_items_chequeo,
            'fch1': limpiar_nan(fch1),
            'fch2': limpiar_nan(fch2),
            'fch3': limpiar_nan(fch3),
            'fch4': limpiar_nan(fch4),
            'f1': inline_image_f1,
```

Figura 92. Código hoja vida Parte 22.

```
            'f2': inline_image_f2,
            'f3': inline_image_f3,
            'f4': inline_image_f4,
            'ing1': inline_image_ing1,
            'ing2': inline_image_ing2,
            'ing3': inline_image_ing3,
            'ing4': inline_image_ing4,
            'nom1': limpiar_nan(nom1),
            'nom2': limpiar_nan(nom2),
            'nom3': limpiar_nan(nom3),
            'nom4': limpiar_nan(nom4),
            'cc1': limpiar_nan(cc1),
            'cc2': limpiar_nan(cc2),
            'cc3': limpiar_nan(cc3),
            'cc4': limpiar_nan(cc4),
            'id_equipo': limpiar_nan(ID_EQ),
            'HISTORIAL': lista_items_historial,
        }
        datos.append(datos_dict)
    context = {
        'Municipio': nombre,
        'Ubicacion':ubicacion,
        'NIT': limpiar_nan(nit),
        'EMAIL': limpiar_nan(email),
```

Figura 93. Código hoja vida Parte 23.

```
        'CELULAR': limpiar_nan(celular),
        'Fecha': limpiar_nan(fecha),
        'LOGO_OUT': inline_image_logo,
        'datos': datos,
    }
    docx_tpl.render(context)
    titulo = f"Hoja de vida {ubicacion} {nombre_municipio}".upper().replace
        ("_", " ")
    docx_path = os.path.join(output_path, f"{titulo}.docx")
    docx_tpl.save(docx_path)
    convert(docx_path)
```

Figura 94. Código hoja vida Parte 24.