Gestor de Inventario para Hogar o Laboratorio

Proyecto Final

Integrantes:

Samuel Acuña Garzón

Carolina García García

Algoritmos en sistemas electrónicos

Daniel Palomino

2025

# Resumen del sistema

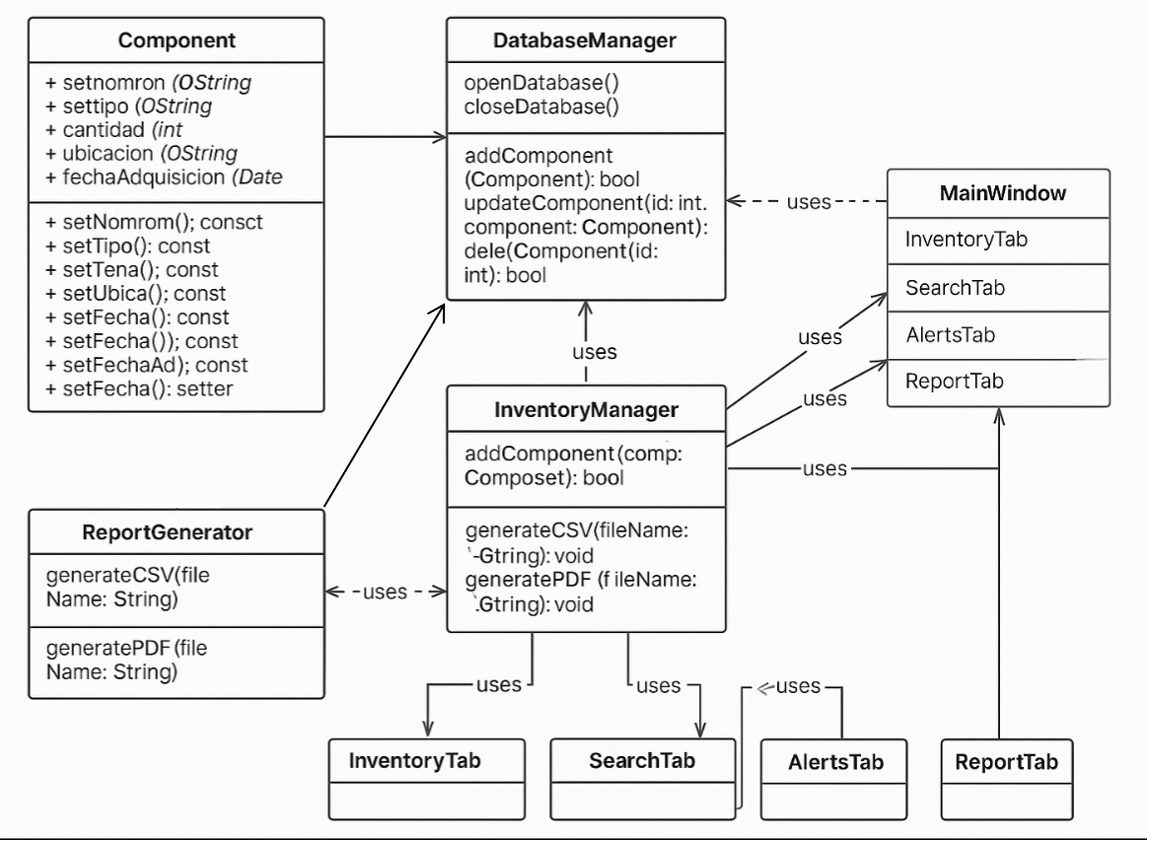
El sistema de inventario para administrar componentes electrónicos o dispositivos del hogar/laboratorio. Implementado en C++ con Qt y base de datos SQLite, orientado a dispositivos de bajo consumo como Raspberry Pi; este cuenta con un entorno grafico para visualizar el inventario, las alertas de stock bajo, los reportes y la búsqueda de componentes como a su vez se le permite al usuario interactuar con el entorno añadiendo editando y eliminando componentes.

# Metodología

Para este proyecto se realiza la implementación de 4 clases especificas: class Component, class Inventory Manager, class Database Manager, class Report Generator ,para desarrollar cada una de las funciones solicitadas en el resumen del sistema, el grupo de trabajo utilizo de intermediarios otras clases para que sean estas quienes interactuen con el usuario y envíen la información a cada una de las clases para que estas operen de la manera esperada, cabe aclarar que no se utilizó directamente el entorno grafico de qt (los archivos .ui) pero si se utilizaron mediante código en estas clases auxiliares mencionadas anteriormente junto con las funciones específicas de qt.

# Diseño, Implementación y Funcionamiento de clases

Para el diseño del gestor inventario se llegaron a usar dos tipos de clases; una de estas se enfoca en el funcionamiento interno del gestor, guardando y editando los datos registrados por el usuario en una base de datos, filtrar y buscar datos específicos, junto a la generación de reportes y el manejo de alertas de componentes escasos. El otro tipo de clases se encarga en la interfaz con la que interactúa el usuario; estos dos tipos de clases se relacionan y se ven conectadas entre sí, esto se ve reflejado en el siguiente UML:



## Clases Principales

Estas clases son las que se encargan del funcionamiento interno, estas son:

### Clase Component

La clase Component representa los elementos básicos del inventario. Cada componente cuenta con atributos como el nombre, tipo, cantidad, ubicación y fecha de adquisición. Esta clase incluye métodos de acceso (getters) y modificación (setters) que permiten interactuar con sus datos. Es una clase fundamental utilizada por otras entidades del sistema, como el generador de reportes (Report Generator).

### Clase Database Manager

El Database Manager es responsable de gestionar la conexión y operaciones básicas sobre la base de datos, incluyendo abrir y cerrar la conexión, así como insertar, actualizar o eliminar componentes. Esta clase es utilizada por el Inventory Manager para almacenar y recuperar los datos persistentes de los componentes.

### Clase Inventory Manager

Esta clase actúa como intermediaria entre la lógica de la interfaz y la base de datos. Administra las operaciones de alto nivel sobre los componentes, como su adición y la generación de reportes en formato CSV o PDF. Se apoya en Database Manager para la persistencia y en Report Generator para la elaboración de reportes. Es la clase central del sistema y es utilizada por todas las pestañas de la interfaz de usuario.

### Clase Report Generator

La clase Report Generator está encargada de la generación de reportes del inventario. Ofrece métodos para exportar los datos a archivos en formato CSV o PDF. Para esto, utiliza los datos de los objetos de tipo Component. Aunque no interactúa directamente con la interfaz gráfica, es invocada desde Inventory Manager para proporcionar esta funcionalidad.

## Clases Para la Visualización

Estas clases se enfocan en la interfaz gráfica donde interactúa el usuario, estando relacionadas con las clases principales, siendo estas:

### Clase Main Window

Main Window representa la ventana principal de la interfaz gráfica del usuario. Contiene diferentes pestañas que permiten interactuar con distintas funcionalidades del sistema, tales como la gestión del inventario, búsqueda de componentes, visualización de alertas y generación de reportes. Cada una de estas pestañas se comunica con Inventory Manager para acceder a la lógica del sistema y datos necesarios.

### Pestañas de la Interfaz

El sistema está compuesto por cuatro pestañas principales:  
 - Inventory Tab: Permite al usuario añadir y gestionar los componentes del inventario.  
 - Search Tab: Facilita la búsqueda de componentes mediante distintos criterios.  
 - Alerts Tab: Muestra alertas relacionadas con el inventario, como bajo stock.  
 - Report Tab: Permite generar reportes del inventario en distintos formatos.  
 Cada una de estas pestañas hace uso de Inventory Manager para llevar a cabo sus funciones.

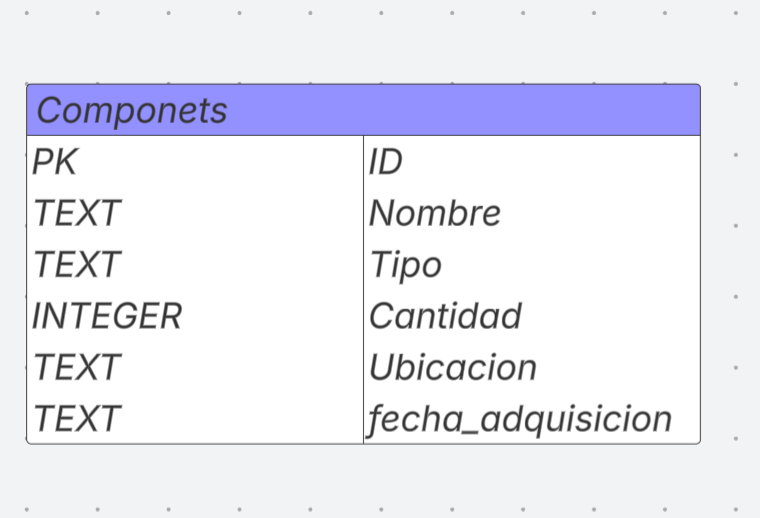
## Relaciones entre Clases

Las pestañas de la interfaz (InventoryTab, SearchTab, AlertsTab, ReportTab) usan a Inventory Manager.  
Inventory Manager usa a Database Manager para gestionar la persistencia de los datos.  
Inventory Manager también usa a Report Generator para crear reportes a partir de los objetos Component.  
Para finalizar, el Report Generator utiliza directamente a DatabaseManager para acceder a los datos necesarios para los reportes.

# Diseño de la base de datos

La aplicación utiliza una base de datos embebida SQLite para almacenar y presentar dentro del entorno grafico la información de los componentes electrónicos. Esta base de datos está integrada directamente en el sistema.

A continuación, se muestra el modelo de datos utilizado, representado en un **diagrama Entidad-Relación (ER)** Creado con Lucidchart. Esta aplicación se creó teniendo presente que todo se almacena en una única tabla llamada Componentes, que representa cada componente del inventario.



La explicación de cada uno se muestra a continuación

Id: Identificador único del componente (clave primaria)

Nombre: Nombre del componente agregado en el inventoryTab y enviado al InventoryManager

Tipo: Tipo del componente (e.g., resistencia, IC)

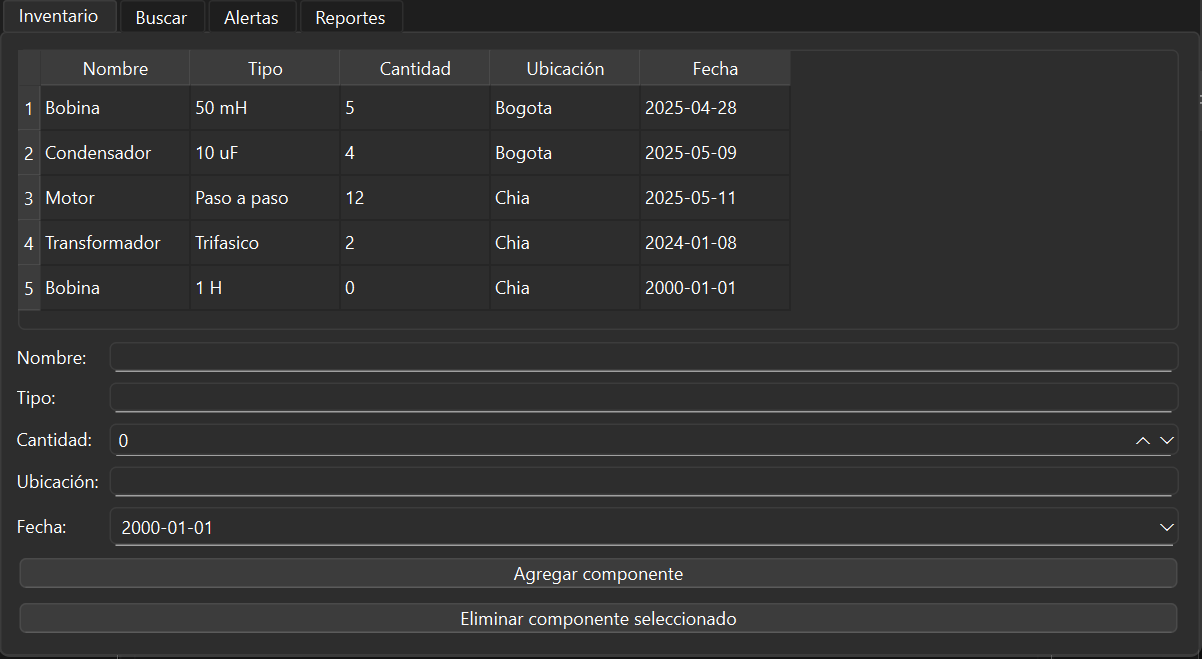
Cantidad: Número de unidades disponibles (con base en este dato se van a generar las alertas en caso de que la cantidad sea menor a 5 unidades)

Ubicación: Lugar físico donde se encuentra el componente

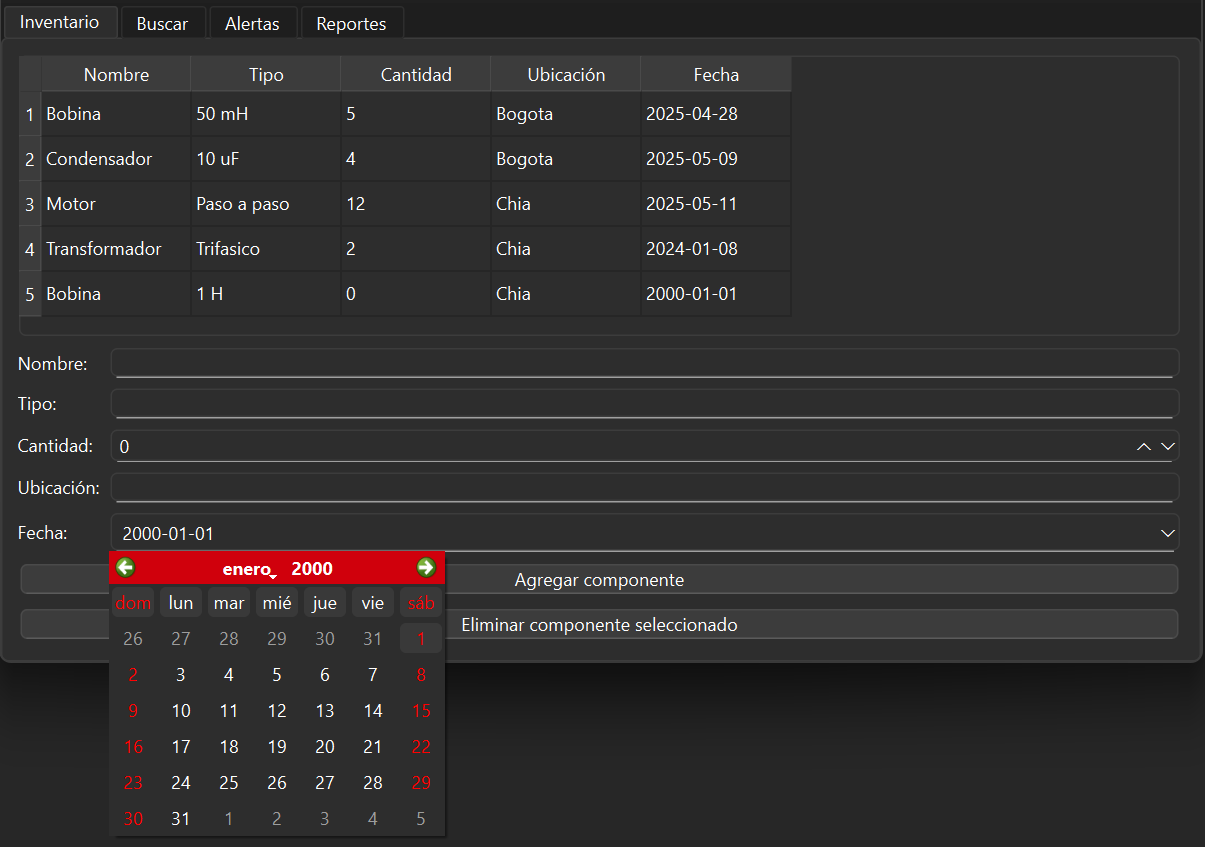
Fecha\_adquisicion: Fecha en la que se registró la entrada del componente en la base de datos

# Ejecución final de la aplicación

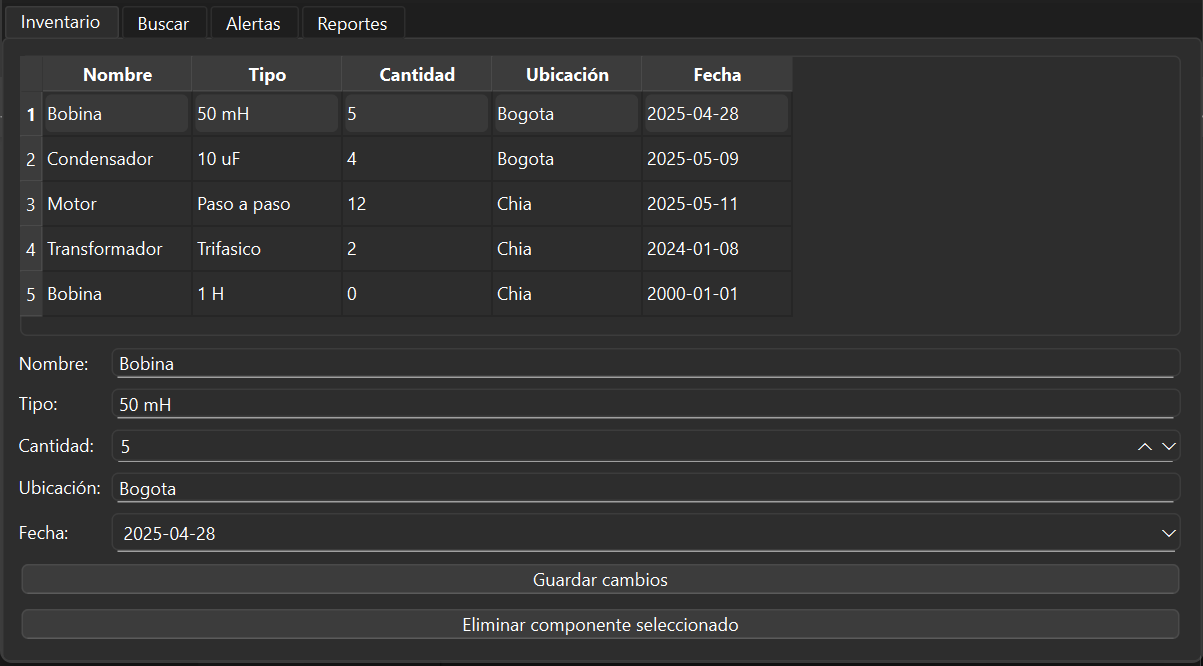
Al ingresar al Gestor de Inventario, nos vamos a encontrar con la siguiente página:



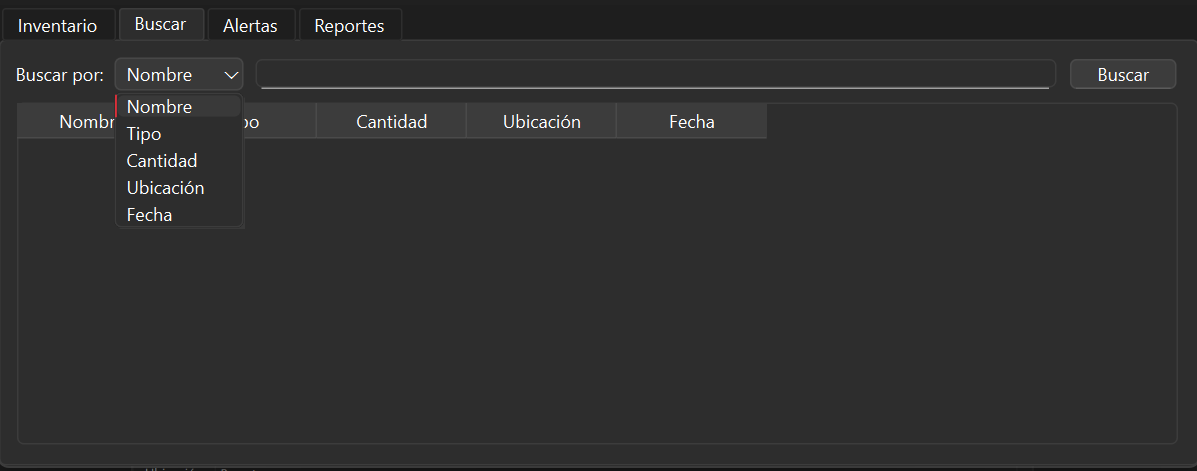
En esta, el usuario va a poder agregar componentes nuevos que él indique, agregándole el nombre, el tipo y la ubicación de forma escrita, la cantidad del componente por medio de las flechas ubicadas en la zona derecha de este atributo. Para en caso de la fecha, el usuario la puede agregar ya sea escrita o por medio de un calendario desplegable con la flecha que se encuentra en la zona derecha de este campo. Cuando el usuario registra el componente, este va a aparecer en la tabla de la parte superior de la pestaña del inventario.



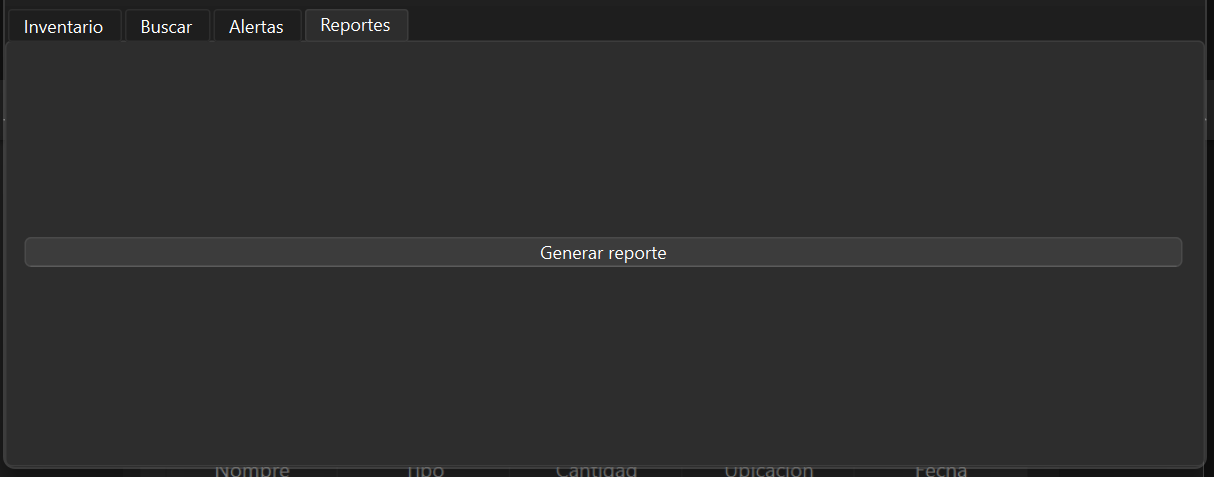
En esta parte, el usuario también la posibilidad modificar o eliminar componentes que ya se encuentran registrados, para esto se selecciona el componente y el usuario va a poder realizar los cambios necesarios.

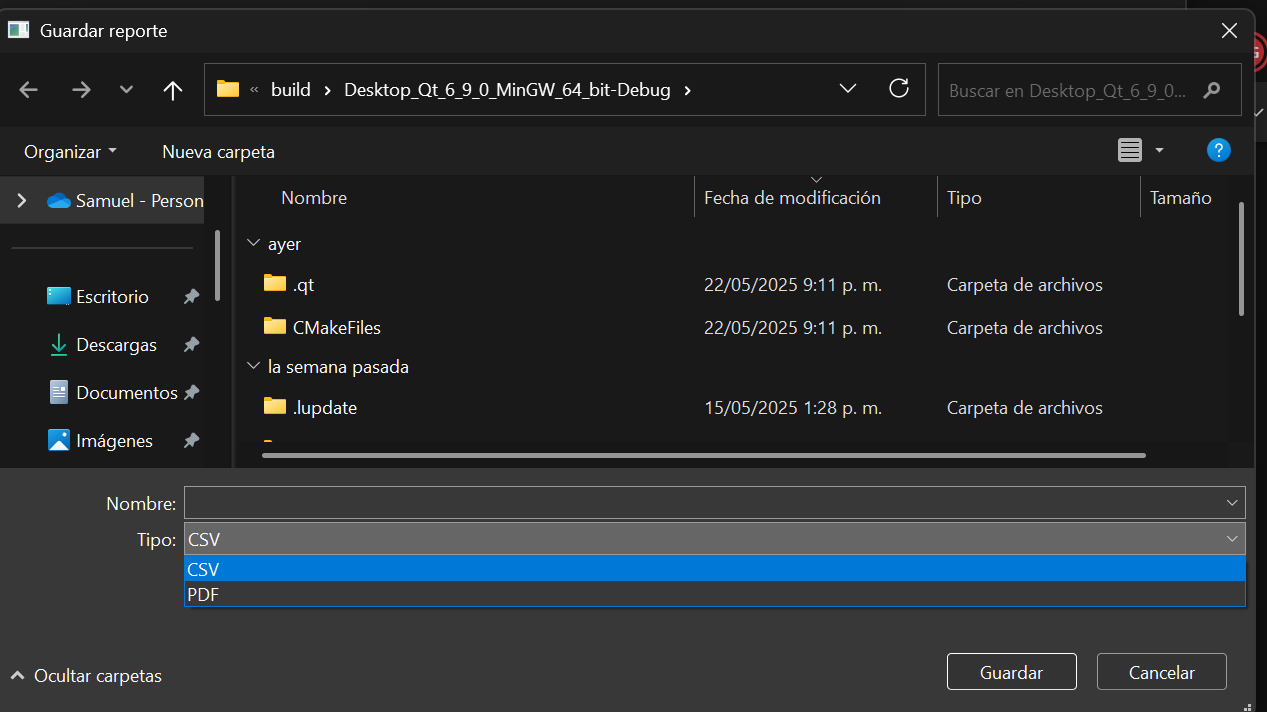


Pasando a la pestaña de buscar, en esta sección el usuario puede buscar y filtrar los componentes que él requiera, esto se hace a través de los atributos que tiene los componentes.

En la sección de alerta, se va a poder ver los datos cuya cantidad sea menor o igual a cinco unidades. Los datos que tengan una cantidad mayor a cinco no van a aparecer en esta sección.

Para finalizar, en la pestaña de reportes se encuentra un botón para generar y guardar un reporte en formato PDF o CSV, esto es dependiendo de lo que el usuario necesite.





# Repositorio:

<https://github.com/samuel2005wq/Proyecto_Inventario/tree/main>

# Conclusiones

* El desarrollo de la creación e implementación de esta aplicación de gestión de inventarios ha permitido integrar diferentes conocimientos brindados a lo largo del curso como C++, sqlt y a su vez se logró implementar a través del diseño orientado a objetos y el uso de patrones estructurados una solución funcional y completa de lo requerido en el proyecto.
* La implementación de funcionalidades dentro de la aplicación como la búsqueda avanzada, generar reportes y alertas por inventario bajo, demuestra el uso práctico del sistema en contextos reales como laboratorios, oficinas técnicas o almacenes de componentes electrónicos.
* Durante el desarrollo de la aplicación, se usaron herramientas como los diagramas UML para la planificación de las clases y sus interacciones y un modelo entidad-relación claro para el diseño de la base de datos embebida. Estos elementos no solo ayudaron a guiar el desarrollo, sino que también aseguran una comprensión más sencilla del sistema por parte del grupo de trabajo.