Universidad Rafael Landívar
Facultad de Ingeniería.
Ingeniería en informática y sistemas.
Programación Avanzada - Sección: 15
Catedrático: Cindy García Pérez

PROYECTO DE APLICACIÓN 2

Estudiantes: Héctor José Flores Pineda 1199923
Joaquín Raymundo Choc Salvador 1280423

Guatemala, 10 de noviembre de 2023

Introducción

Este proyecto de aplicación se centra en optimizar los procesos de gestión de inventario en una farmacia, aprovechando la tecnología digital para facilitar la organización y acceso a la información. Lo que ofrece un control más detallado sobre los movimientos que se realizan en el inventario, permitiendo agregar nuevos medicamentos, actualizar el inventario existente y consultar detalles por proveedor, nombre, categoría, entre otros criterios. Además, proporciona una interfaz intuitiva para visualizar y ordenar la información, lo que facilita la administración eficiente de los recursos disponibles. Brindando una solución efectiva para mantener un control preciso de los medicamentos en la farmacia, contribuyendo así a la operación eficiente del establecimiento en un contexto donde la gestión de inventario es esencial para su prosperidad.

Análisis

- **Entradas**

  - Class Medicamento:

    - ➔ string Nombre del Medicamento: Cadena de texto que representa el nombre del medicamento.
    - ➔ string Categoria: Cadena de texto que representa de qué tipo es el medicamento.
    - ➔ Int Dosis: Número entero que representa la cantidad de dosis recomendada para el medicamento.

  - Class Inventario

    - ➔ int Stock: Representa la cantidad de unidades que se tiene del medicamento en la farmacia.
    - ➔ string Caducidad: Cadena de texto que representa la fecha de vencimiento de un medicamento.
    - ➔ Proveedor proveedor: Representa el conjunto de datos de información sobre el proveedor del medicamento.
    - ➔ double Compra: Número decimal que representa el precio de compra del medicamento por parte del proveedor.
    - ➔ double Venta: Número decimal que representa el precio de venta del medicamento por parte de la farmacia.

  - Class Proveedor

    - ➔ string NombreProveedor: Cadena de texto que representa el nombre del proveedor de los medicamentos.
    - ➔ int NIT: Número entero de 9 dígitos que representa el Número de Identificación Tributaria.
    - ➔ string DireccionFiscal: Cadena de texto que representa la dirección fiscal del proveedor.
    - ➔ string Direccion: Cadena de texto que representa la dirección del proveedor.
    - ➔ int Teléfono: Número de 8 dígitos que representa el número telefónico del proveedor.
    - ➔ string Correo: Cadena de texto que representa el correo electrónico del proveedor.

  - string Búsqueda: Cadena de texto que representa el elemento a filtrar o a buscar dentro del listado de inventario, sea su nombre, principio activo, entre otros criterios.

- Parámetro Búsqueda: Criterio a utilizar en la búsqueda o filtrado de un medicamento.

- **Salidas**

  - Class Medicamento:

    - ➔ string Nombre del Medicamento: Cadena de texto que representa el nombre del medicamento.
    - ➔ int Registro: Número entero que representa el número de identificación único del medicamento
    - ➔ string Categoria: Cadena de texto que representa de qué tipo es el medicamento.
    - ➔ Int Dosis: Número entero que representa la cantidad de dosis recomendada para el medicamento.

  - Class Inventario

    - ➔ int Stock: Representa la cantidad de unidades que se tiene del medicamento en la farmacia.
    - ➔ string Caducidad: Cadena de texto que representa la fecha de vencimiento de un medicamento.
    - ➔ Proveedor proveedor: Representa el conjunto de datos de información sobre el proveedor del medicamento.
    - ➔ double Compra: Número decimal que representa el precio de compra del medicamento por parte del proveedor.
    - ➔ double Venta: Número decimal que representa el precio de venta del medicamento por parte de la farmacia.

  - Class Proveedor

    - ➔ string NombreProveedor: Cadena de texto que representa el nombre del proveedor de los medicamentos.
    - ➔ int NIT: Número entero de 9 dígitos que representa el Número de Identificación Tributaria.
    - ➔ string DireccionFiscal: Cadena de texto que representa la dirección fiscal del proveedor.
    - ➔ string Direccion: Cadena de texto que representa la dirección del proveedor.
    - ➔ int Teléfono: Número de 8 dígitos que representa el número telefónico del proveedor.
    - ➔ string Correo: Cadena de texto que representa el correo electrónico del proveedor.

- Otros

➔ Lista de Inventario: Listado que contiene todos los medicamentos registrados en la farmacia.
➔ Lista de Proveedores: Listado que contiene todos los proveedores registrados en la farmacia.
➔ double Precio Promedio: Cantidad decimal que indica el precio promedio del inventario disponible en la farmacia.
➔ Inventario.CSV: Informe de la información de los medicamentos en el inventario en un archivo.csv.
➔ Mensaje de error: Mensaje de formulario que indica que ocurrió o se presentó un error en la ejecución del programa.
➔ Mensaje de confirmación: Mensaje de formulario que indica la completa ejecución de un proceso o función.
➔ Formulario 1: Formulario adicional que brinda la opción al usuario de ingresar la información de un nuevo proveedor.
➔ Formulario 2: Formulario adicional que brinda la opción al usuario de actualizar la información de un medicamento registrado en el inventario.

- **Procesos**

  - **Declaración de variables globales**

    list<Inventario> Medicinas:

    - Tipo: list
    - Descripción: Listado que almacena todos los medicamentos ingresados en el inventario.

    list<Proveedor> proveedores:

    - Tipo: list
    - Descripción: Listado que almacena todos los proveedores registrados en la farmacia.

    static int Id:

    - Tipo: static int
    - Descripción: Variable entera que designa el número de identificación único para cada medicamento.

    bool CambioVerdadero:

    - Tipo: bool
    - Descripción: Variable booleana que indica si el cambio a realizar en la información en el inventario es válido o no.

bool Cerrar:

- Tipo: bool
- Descripción: Variable booleana que indica si es posible cerrar el formulario actual.
-

- **Funciones adicionales**

bool TxtBoxLlenos(Panel^ panel)

La función TxtBoxLlenos recibe un panel como entrada y verifica si todos los TextBox dentro de ese panel contienen texto. Utiliza un bucle para iterar a través de los controles del panel y verifica si cada uno de ellos es un TextBox. Si encuentra un TextBox vacío, la función retorna false. En caso contrario, si todos los TextBox están llenos, la función retorna true.

void VaciarTxtbox(Panel^ panel)

La función VaciarTxtbox se encarga de borrar el texto presente en todos los cuadros de texto (TextBox) contenidos dentro de un panel en una interfaz gráfica. Utiliza un bucle for each para recorrer los controles dentro del panel y verifica si cada uno es un TextBox. En caso afirmativo, realiza un casting dinámico para acceder a las propiedades específicas del TextBox y luego establece el texto como una cadena vacía, lo que resulta en la eliminación del contenido.

void LlenarTabla(list<Inventario> listado, DataGridView^% tabla)

La función LLenarTabla se encarga de llenar un control DataGridView con información proveniente de una lista de objetos de tipo Inventario. Primero, se inicializa un iterador para recorrer la lista. Luego, se itera a través de los elementos de la lista, agregando un nuevo renglón en la tabla por cada elemento.

Para cada renglón, se asignan valores a las celdas correspondientes en la tabla. Se extrae la información de cada objeto Inventario usando sus métodos Get, y se convierte al formato adecuado para ser asignado a las celdas del DataGridView. Se utilizan las propiedades Rows y Cells del control para acceder y asignar los valores.

void VaciarTabla(DataGridView^% tabla)

Esta función tiene como objetivo eliminar todas las filas de un control DataGridView proporcionado como argumento. Primero, verifica si la cantidad de filas en la tabla es mayor a cero con la condición tabla->Rows->Count > 0. Si esto es cierto, procede a un bucle que recorre las filas en orden inverso (i >= 0), y elimina cada fila usando tabla->Rows->RemoveAt(i). Esto garantiza que las filas se eliminen correctamente, incluso si el índice de la fila cambia durante la operación. En resumen, esta función proporciona una manera eficiente de vaciar completamente un DataGridView.

bool ValidarProveedor(String^ nit, String^ telefono, String^ correo)

La función ValidarProveedor verifica la validez de la información proporcionada para un proveedor. Primero, comprueba si la longitud del número de identificación tributaria (NIT) no es igual a 9 dígitos, o si la longitud del número de teléfono no es igual a 8 dígitos, o si el correo electrónico no contiene el carácter '@'. Si alguna de estas condiciones se cumple, la función devuelve “false”, indicando que la información proporcionada es inválida. De lo contrario, si todas las condiciones no se cumplen, la función devuelve true, lo que indica que la información del proveedor es válida.

- **Menú de Inicio**
  Se presenta un menú en el que se indica el nombre de la empresa Farmaceútica y un botón “Ingresar Proveedor”:

  ➔ Botón “Ingresar Proveedor”
  Al presionar el botón se muestra en un pantalla un formulario externo, en el que se presentan los campos de información del proveedor a llenar, además de un botón “Ingresar”.

  Eventos

  Evento button11_Click():

  En este evento implementa un ciclo while que se ejecuta mientras el número de elementos en la lista de proveedores no sea igual a 4. Dentro del ciclo, se llevan a cabo una serie de operaciones. En primer lugar, se inicializa una variable booleana que servirá para indicar si se encontró un proveedor con información duplicada. Luego, se crea una instancia del formulario externo y otra instancia de un objeto de tipo Proveedor. El formulario externo es mostrado en modo de diálogo, permitiendo al usuario ingresar información sobre un nuevo proveedor.

  Una vez que el formulario se cierra, la información ingresada en los campos de texto del formulario se recopila y se asigna a las propiedades del objeto Proveedor recién creado. A continuación, se inicia una iteración sobre la lista de proveedores existentes para verificar si existe algún duplicado en términos de NIT, nombre o número de teléfono.

  Si no se encuentra ningún proveedor repetido, el nuevo proveedor se agrega a la lista de Proveedores y se muestra un mensaje de confirmación. Y en tal caso se detecta un proveedor repetido, se muestra un mensaje de error indicando que se ha ingresado información duplicada.

  Finalmente, al salir del bucle, se realizan operaciones de interfaz gráfica para hacer visibles los botones en el menú principal y ocultar el botón “Ingresar Proveedor".

- **Form Nuevo Proveedor**

  Formulario que muestra todos los campos de información requeridos para el ingreso de un nuevo proveedor

  Eventos

  Evento Form1_ button1_Click():

  En este evento, al presionar el botón “Ingresar” del Form Nuevo Proveedor se ejecuta el siguiente algoritmo:

  En primer lugar, se verifica si alguno de los campos de texto (nombre, NIT, dirección, etc.) está vacío mediante una serie de condiciones “if” para determinar si el contenido de los campos no es igual a un espacio en blanco. Si se encuentra algún campo vacío, se muestra un mensaje de error indicando que un campo de información está sin completar. En caso contrario, se procede con el proceso de validación.

  La segunda parte de la validación se enfoca en verificar si el NIT, el número de teléfono y el correo electrónico ingresados son válidos utilizando la función ValidarProveedor(). Si la validación es exitosa, lo que indica que los datos cumplen con los requisitos establecidos, se establece la variable Cerrar como verdadera y se cierra el formulario actual.

  Sin embargo, si la validación de proveedor no es exitosa, se muestra un mensaje de error indicando que el NIT, el número de teléfono o el correo electrónico no han sido ingresados correctamente.

  Evento Form1_TxtBox_KeyPress():

  En este tipo de eventos, se verifica si el carácter presionado no es un dígito numérico utilizando la función Char::IsDigit. Si el carácter no es un dígito, pasa a la siguiente condición que verifica si el carácter no es el código ASCII para la tecla de retroceso (representado como 0x08). Si alguna de estas condiciones se cumple, se marca el evento como "manejado" al establecer e->Handled como verdadero. Esto indica que el evento ha sido tratado y el carácter no será procesado en el campo de texto.

  Evento Form1_FormClosing():

  En este evento se verifica si la variable booleana “Cerrar” es falsa, si lo es, muestra un mensaje en pantalla indicando que no es posible cerrar la ventana hasta que se complete el llenado de información, de lo contrario, se habilitará la opción de cerrar el formulario.

Funciones Get():

Las funciones Get() retornan los valores ingresados en los Textbox, con el fin de realizar el traspaso de información de un formulario adicional al formulario principal.

- **Menú Principal**
  Se presenta un menú en el que se muestran los botones de las funciones principales del programa:

  1. Botón “Nuevo Medicamento”

     Eventos

     Evento BotonNuevoMedicamento_Click()

     En este evento se interactúa con la lista de proveedores del programa.. Primero, se inicializa un iterador que se utilizará para recorrer la lista de proveedores.

     A continuación, se inicia un bucle que comienza asignando a “it” el primer elemento de la lista de proveedores y verifica si no ha alcanzado el final de la lista. Esto asegura que el bucle recorra todos los elementos de la lista.

     Dentro del bucle, se agrega el nombre de cada proveedor a un ComboBox. Esto se logra utilizando la función Add para añadir un nuevo elemento al ComboBox. El nombre de cada proveedor se obtiene a través de la función GetNombreP() aplicada al objeto de tipo Proveedor referenciado por “it”.

     Finalmente, después de completar el bucle, se cambia la pestaña que se muestra al usuario en la interfaz gráfica. Esto redirige al usuario a la pestaña número 1.

     Luego de presionar el botón se despliega una interfaz con los campos de información requeridos para el ingreso de un nuevo medicamento, además de presentar dos nuevos botones “Ingresar” y “Regresar”.

     Evento BotonIngresar1_Click:

     Comienza por verificar que todos los campos de texto estén llenos y que los ComboBoxes tengan opciones seleccionadas. Si este requisito se cumple, procede a inicializar instancias de las clases Inventario y Proveedor.

     Luego, se prepara una variable booleana para rastrear si se detecta un nombre de medicamento duplicado durante el proceso. A continuación, se incrementa el ID asociado a cada medicamento para asegurar un identificador único.

Los valores ingresados en los campos de texto y ComboBoxes se asignan a los atributos del objeto Inventario. Además, se busca y se asigna el proveedor correspondiente seleccionado en el ComboBox de proveedores.

Luego, se realiza un bucle que recorre la lista de medicamentos existentes para verificar si el nombre del medicamento es repetido.

Después, se verifica que los datos ingresados cumplan con los parámetros establecidos utilizando la función ValidarInfo(). En caso de que algún dato no cumpla con los criterios de validación o se encuentre un nombre repetido, se muestra un mensaje de error y se disminuye en uno el ID para simular que la entrada no fue creada.

Si los datos cumplen con los requisitos y no se encuentra un nombre duplicado, se muestra un mensaje de confirmación y se procede a vaciar los campos de texto y ComboBoxes en la pestaña actual. Finalmente, el nuevo medicamento es agregado a la lista de medicamentos.

En caso de que no se cumpla la condición inicial de campos llenos o ComboBoxes seleccionados, se muestra un mensaje de error indicando que un campo de información está vacío.

Evento BotonRegresar1_Click():

Vacía el Combobox con la información de Proveedores y dirige al usuario a la pestaña principal.

2. Botón “Actualizar Inventario”

En esta opción se despliega una pestaña en la que se muestra una tabla con la información actual de la lista de inventario de medicinas que se tiene hasta el momento, brindando la opción de seleccionar un elemento de la tabla para modificar la información del producto en cuestión.

Eventos

Evento BotonActualizarInventario_Click():

Primero, cambia la pestaña seleccionada a la pestaña número 2. Luego, llama a una función llamada LLenarTabla() que se encarga de poblar un DataGridView con información proveniente del listado del inventario

Evento TablaAct_DoubleClick():

Evento que ocurre cuando el usuario hace doble clic sobre una fila de la tabla de inventario. La función comienza extrayendo la información seleccionada en un DataGridView y la convierte en cadenas de texto para cada campo relevante de información.

A continuación, se descompone la fecha de caducidad en sus componentes individuales de año, mes y día. Esto es importante para trabajar con la fecha de manera más flexible y facilitar su manipulación.

Luego, se crea una instancia de otro formulario externo para editar la información del medicamento seleccionado. Rellena los campos del formulario con la información extraída del DataGridView.

Después, muestra el formulario al usuario mediante el método ShowDialog, una vez que el usuario ha realizado los cambios en la información del medicamento y ha confirmado la edición, se procede a verificar si la modificación es válida. Si es así, se obtiene el índice de la fila seleccionada en el DataGridView y se prepara un contador.

A continuación, se itera sobre la lista de medicamentos hasta que el contador coincida con el índice de la fila seleccionada.

Se actualizan los atributos del medicamento con la nueva información ingresada en el formulario de edición.

Se verifica si la información actualizada cumple con los requisitos establecidos mediante la función ValidarInfo(). Si todo está en orden, se reemplaza el medicamento en la lista de medicamentos con el medicamento actualizado y se muestra un mensaje de confirmación.

Si la información no cumple con los criterios de validación, se muestra un mensaje de error. Si no se puede actualizar la información, también se muestra un mensaje de error.

Finalmente, se eliminan las filas del DataGridView y se llenan nuevamente con la información actualizada de la lista de medicamentos.

Evento Regresar2_Click():

Primero, cambia la pestaña activa, seleccionando la pestaña número 2.

Después, se invoca la función LLenarTabla(). Esta función tiene como objetivo llenar un DataGridView llamado “TablaAct” con información específica de medicamentos proveniente de la lista de inventario de medicamentos.

Form2

Se despliega un formulario con la información del medicamento seleccionado en TextBox y ComboBox para que el usuario pueda modificar la información del medicamento

Eventos

- **Form2_BotonActualizar_Click():**

Evalúa si todos los campos de texto en un panel están llenos y si los ComboBox de categoría y proveedor tienen texto seleccionado.

Si esta condición se cumple, se asigna el valor true a la variable CambioVerdadero() y se procede a cerrar la ventana actual. Esto indica que se ha confirmado un cambio válido.

En caso de que la condición no se cumpla, se muestra un mensaje de error al usuario a través de un cuadro de diálogo. El mensaje advierte al usuario que debe llenar todos los campos de información para continuar.

- **Form2_BotonCancelar_Click():**

Este evento convierte la variable booleana CambioVerdadero igual a falso; y cierra el formulario externo.

- **Form2_GetStatusCambio():**

Función que retorna la variable booleana CambioVerdadero();

3. Botón “Consulta Información”

Esta opción despliega una pestaña en la que se muestra un menú en la que se le solicita al usuario ingresar en un Textbox, el nombre o principio activo del medicamento según el criterio seleccionado, mostrando la información del medicamento.

Eventos

- **Evento BotonConsultarInformación_Click()**:

En este evento, primero se establece que la pestaña activa se cambie a la tercera pestaña.

A continuación, se observan varias líneas que asignan texto específico a distintos elementos de la interfaz de usuario.

Estas líneas de código tienen el propósito de inicializar o resetear visualmente ciertos campos de información. En este caso, se establece un texto de guía compuesto por guiones bajos ("_") dentro de las etiquetas correspondientes. Esto puede indicar al usuario que estos campos están disponibles para ser llenados con información.

- **Evento Buscar_Click():**

El evento comienza creando un objeto de la clase Inventario llamado Búsqueda, que será utilizado para llevar a cabo las operaciones de búsqueda.

Se declara una variable tipo string para almacenar la información que el usuario desea buscar. También se crea un iterador que se utilizará para recorrer la lista de medicamentos.

Luego, se verifica si el campo de texto de búsqueda contiene información. En caso afirmativo, se procede con la búsqueda. Si el campo está vacío, no se realiza ninguna operación.

A continuación, se verifica qué opción de búsqueda ha seleccionado el usuario en la interfaz gráfica. Si la opción de búsqueda por nombre está seleccionada, se obtiene la información a buscar del campo de texto y se ejecuta la búsqueda utilizando el método BusquedaEncontradaNombre() del objeto Búsqueda.

Si se encuentra un medicamento con el nombre buscado, se actualizan las etiquetas de la interfaz gráfica con la información correspondiente del medicamento encontrado. En caso de no encontrar el elemento buscado, se muestra un mensaje de error mediante un cuadro de diálogo.

Si la opción de búsqueda por principio activo está seleccionada, se realiza una operación similar a la búsqueda por nombre, pero esta vez se utiliza el método BusquedaEncontradaPrincipio().

Si no se ha seleccionado ningún parámetro de búsqueda, se muestra un mensaje de error indicando que el usuario debe elegir una opción de búsqueda. Si el campo de búsqueda está vacío, se muestra un mensaje que indica al usuario que debe ingresar el nombre o el principio activo del medicamento para realizar la búsqueda. En todos los casos, el programa finaliza la ejecución del código después de mostrar el mensaje de error.

Evento Regresar3_Click():

Este evento dirige al usuario a la pestaña principal del programa, además de vaciar la información del medicamento encontrado y limpiar el Textbox de búsqueda.

Funciones

bool BusquedaEncontradaNombre(list<Inventario> Listado, string buscar)

En esta función se utiliza un bucle que recorre a través de la lista tomada como parámetro utilizando un iterador. Dentro del bucle, se verifica si el nombre del objeto en la posición de memoria del iterador coincide con el valor de la variable buscar. Para esto, se compara el nombre del objeto con el valor de la variable buscar.

Si se encuentra una coincidencia, se indica que se ha encontrado un elemento con el nombre buscado. En caso de que no se encuentre ninguna coincidencia después de recorrer toda la lista, se indica que no se ha encontrado ningún elemento con el nombre buscado.

bool BusquedaEncontradaPrincipio(list<Inventario> Listado, string buscar)

Realiza el mismo procedimiento que la función BusquedaEncontradaNombre(), la diferencia es que el parámetro de búsqueda empleado es el principio activo del medicamento.

4. Botón “Generar informe”

En esta opción se despliega un RichTextBox que presenta un informe del listado de inventario de medicamentos ordenados por nombre del medicamento, incluyendo la opción de “exportar” el informe a un archivo.csv, brindando al opción al usuario de escoger en qué directorio desea guardar el archivo.

Eventos

Evento GenerarInforme_Click()

Primero, se establece que la pestaña seleccionada en la interfaz gráfica sea la número 4;. Esto indica que se está dirigiendo al usuario a una sección específica de la interfaz gráfica relacionada con la generación de informes.

Luego, se crea un objeto llamado objeto de la clase Inventario. Esto servirá como instancia para acceder a los métodos de la clase.

A continuación, se declara una lista de objetos Informe de la clase Inventario, y se utiliza el método OrdenamientoPorNombre() para ordenar la lista de medicamentos por nombre. El resultado se almacena en la lista Informe.

Después, se inicia un bucle que recorre cada elemento de la lista de medicamentos en el inventario. Para cada medicamento en la lista, se realiza lo siguiente:

Se crea una cadena de texto empleando el método String::Format para formatear la información del medicamento de una manera legible.Esta cadena de texto se agrega al cuadro de texto utilizando el método AppendText, lo que significa que se agrega al final del texto existente en el cuadro de texto sin sobrescribirlo.

Evento Exportar_Click()

Este evento se encarga de facilitar la creación y escritura de un archivo CSV a partir de una interfaz de selección de carpeta. Primero, se crea un objeto de tipo FolderBrowserDialog.. Este objeto proporciona una ventana emergente que permite al usuario seleccionar una carpeta en su sistema de archivos.

Luego, se verifica si el usuario ha seleccionado una carpeta y ha confirmado su elección al presionar el botón "Aceptar" en la ventana de selección de carpeta. Si el usuario procede, se obtiene la ruta de la carpeta seleccionada y se almacena en la variable Ruta. A continuación, se convierte esta ruta en un formato de cadena de caracteres, se crea un archivo con extensión .csv en la carpeta seleccionada por el usuario. El archivo se denomina "Inventario.csv" y se abre en modo de escritura.

Después, se agrega una línea de encabezado al archivo CSV que enumera las columnas de datos. Posteriormente, se agrega la información del informe (previamente formateada como texto CSV válido) al archivo CSV. Finalmente, el archivo se cierra y se muestra un cuadro de diálogo informativo que confirma al usuario que el archivo ha sido guardado con éxito.

Evento Regresar4_Click():

Este evento retorna al usuario a la pestaña principal y borra el informe mostrado en el RichTextBox.

5. Botón “Precio Promedio”

Esta opción muestra al usuario una pestaña en la que se despliega la cantidad de medicamentos disponibles en el inventario de la farmacia y el precio total del inventario. Dando la opción de calcular el precio promedio de compra de los medicamentos en el inventario, mostrando el resultado en Quetzales.

Eventos

Evento PrecioPromedio_Click()

Este evento comienza instanciando un objeto llamado "Texto" perteneciente a la clase "Inventario", el cual se utilizará para llevar a cabo operaciones relacionadas con la administración del inventario.

Posteriormente, se selecciona la pestaña número 5 en la interfaz gráfica, para luego, actualizar el contenido de dos elementos en la interfaz:

Se actualiza el número total de medicamentos disponibles en el inventario. Esto se logra mediante la invocación del método TotalMedicamentos de la clase "Inventario", aplicado a la lista de medicamentos del inventario. El resultado se convierte a una cadena y se concatena con la etiqueta "unidades".

El segundo elemento, "PrecioTotal", se actualiza con el valor total de los precios de los medicamentos en el inventario. Esto se hace utilizando el método TotalPrecios de la clase "Inventario", también aplicado a la lista del inventario. El resultado se formatea como una cadena que representa un valor en moneda con dos decimales.

Evento CalcularPrecio_Click()

Este evento comienza con una variable entera que se tomará como un contador. Posteriormente, se utiliza el método TotalPrecios() para calcular la suma total de los precios de todos los medicamentos en el inventario

Luego, se calcula el promedio de precios por medicamento dividiendo la suma total de precios entre la cantidad total de medicamentos. El resultado se formatea como una cadena que representa un valor en moneda con dos decimales.

Finalmente, este valor calculado se asigna al texto de una etiqueta en la interfaz gráfica, proporcionando así una visualización del promedio de precios por medicamento.

Evento Regresar5_Click()

Este evento retorna al usuario a la pestaña principal y borra el texto que muestra el precio promedio de compra del inventario.

Funciones

int TotalMedicamentos(list<Inventario> Listado)

En esta función se declara una variable entera, que se utilizará para llevar la cuenta del total de existencias de medicamentos en el inventario.A continuación, se inicia un bucle que itera a través de una lista de objetos de la clase Inventario llamada Listado. El bucle se ejecutará mientras el iterador ”elem” no alcance el final de la lista.

Dentro del bucle, se accede al atributo Stock de cada objeto Inventario a través del iterador elem, y se suma su valor actual al contador.Después de completar el bucle y acumular todos los valores de existencias de los medicamentos en la lista, se retorna al valor final del contador.

int TotalPrecios(list<Inventario> Listado)

Esta función calcula la sumatoria total de los precios de compra de los medicamentos de la lista de inventario, por medio de la iteración de la lista de inventario, y sumando el precio de compra de cada medicamento por medio de una variable acumulativa.

6. Botón “Inventario del medicamento”

   Al seleccionar esta opción, se despliega una pestaña en la que se muestra un RichTextBox y un ComboBox con el nombre de todos los medicamentos en el inventario; que al seleccionar un nombre, se muestra en el RichTextBox la información que se tiene del inventario del medicamento seleccionado.

   Eventos

   Evento InventariodelMedicamento_Click()

   En este evento, se dirige al usuario a la pestaña 6 del menú, además se inicializa un iterador que recorre la lista de inventario, agregando los nombres de los medicamentos en la lista al ComboBox.

   Evento NameBox_SelectedIndexChanged()

   Este evento registra la opción seleccionada por el usuario en el ComboBox de nombre de medicamento; Primero se verifica que se esté seleccionando una opción, luego recorre la lista de inventario con un iterador hasta encontrar el nombre del medicamento seleccionado; con el medicamento encontrado se imprimen sus datos de inventario en el TextRichBox.

   Evento Regresar6_Click()

   Este evento dirige al usuario al menú principal, además de borrar los elementos añadidos al Combobox y el texto añadido en el RichTextBox.

7. Botón “Precio más alto”

   Esta opción despliega al usuario una pestaña en la que se presenta un ComboBox y un RichTextBox, en el que se le brinda la opción al usuario de escoger un Proveedor del ComboBox, mostrando la información del medicamento más caro del proveedor en el RichTextBox.

Eventos

Evento PreciomásAlto_Click()

Este evento direcciona al usuario a la pestaña 7 del programa, a su vez, añadiendo los nombres de los proveedores registrados en el ComboBox; por medio de un iterador que recorre la lista de Proveedores.

Evento ProveedorBox_SelectedIndexChanged()

Este evento evalúa la selección de una opción en el ComboBox de Proveedores, borrando cada vez el texto mostrado en el RichTextBox; si se ha escogido un nombre, guarda la opción seleccionada; y por medio de un iterador, busca en la lista de Proveedores, el nombre seleccionado. Al encontrar al Proveedor, encuentra el medicamento más caro con la función PrecioMasAlto().

Posteriormente, se muestra en el RichTextBox la información del medicamento más caro que proporciona el proveedor junto al precio de compra.

Evento Regresar7_Click

En este evento se limpia el texto mostrado en el RichTextBox, la selección del ComboBox y retorna al usuario a la pestaña principal.

Funciones

Inventario PrecioMasAlto(list<Inventario> listado, string proveedor)

Esta función tiene como objetivo encontrar el medicamento con el precio más alto de un proveedor en particular,

Primero, se declara una lista donde se guardaran todos los medicamentos que provengan del proveedor, además de una instancia de la clase Inventario; luego, se recorre la lista de Inventario añadiendo los elementos que contengan como atributo el nombre del Proveedor recibido como parámetro en la función, para agregar el “Medicamento” en la lista declarada inicial.

Después recorre la nueva lista, y compara el precio de cada elemento de la misma, retornando el elemento con el precio de compra más alto; para luego igualarlo a la instancia de la clase Inventario inicializada al inicio, retornando ese objeto al terminar la función.

8. Botón “Búsqueda y filtrado de medicamentos”

Esta opción despliega un conjunto de parámetros, un TextBox o ComboBox dependiendo del parámetro elegido y un Datagridview con la información del listado de Inventario de medicamentos.

Eventos

Evento BusquedayFiltrado_Click()

Este evento dirige al usuario a la pestaña 8 del programa, además de llenar la tabla con la información de la lista de Inventario.

Evento radioCategoria_CheckedChanged()

Este evento actúa cuando el usuario escogió el criterio de “Categoría”, haciendo invisible el TextBox y volviendo visible el ComboBox, añadiendo como opciones las categorías de medicamentos; además de borrar y volver a llenar la tabla de filtrado.

Evento radioProveedor_CheckedChanged()

Este evento actúa cuando el usuario escogió el criterio de “Proveedor”, haciendo invisible el TextBox y volviendo visible el ComboBox, añadiendo como opciones los nombres de los Proveedores provenientes del listado de Proveedores, además de borrar y volver a llenar la tabla de filtrado.

Evento ComboBoxFiltro_SelectedIndexChanged()

Este evento sucede al seleccionar una opción en el ComboBox, inicia declarando una lista de Inventario que se utilizará como listado de filtro, vaciando la información presentada en el Datagridview; luego evalúa si el criterio “Categoría” está seleccionado, si lo está, se ejecuta la función FiltradoCategoria() y se llena la tabla con la información de la lista de filtro modificada por la función; en cambio, si el criterio seleccionado es “Proveedor”, se ejecuta la función FiltradoProveedor() y se realiza el mismo procedimiento.

Evento radioName_CheckedChanged()

Este evento actúa cuando el usuario escogió el criterio de “Nombre”, haciendo invisible el ComboBox y volviendo visible el TextBox, además de borrar y volver a llenar la tabla de filtrado.

Evento radioPrincipio_CheckedChanged()

Este evento actúa cuando el usuario escogió el criterio de “Principio”, haciendo invisible el ComboBox y volviendo visible el TextBox, además de borrar y volver a llenar la tabla de filtrado.

Evento TextBoxfiltro_TextChanged()

Este evento sucede cuando se escribe un carácter en el TextBox de filtrado, el evento comienza declarando una lista de Inventario para el filtro y búsqueda, además de obtener la cadena de texto ingresada por el usuario.

Luego evalúa si el TextBox de filtrado está vacío, si no lo está, vaciará la tabla de filtro, y si el criterio de búsqueda es “Nombre” se ejecuta la función FiltradoNombre() y se llena la tabla de filtro con la lista de búsqueda actualizada; del mismo modo, si se selecciona el criterio de búsqueda “Principio”, se ejecutará la función FiltradoPrincipio(), se llena la tabla de filtrado con los medicamentos filtrados.

Evento Regresar8_Click

Este evento sucede al presionar el botón Regresar en la pestaña 8, lo que hace es redireccionar al usuario a la pestaña principal y vacía la tabla de filtrado.

Funciones

void FiltradoCategoria(list<Inventario> &listado, string categoria)

Esta función tiene como objetivo encontrar los medicamentos de la lista Inventario que concuerde con la misma “Categoría”, para ello, se recorre la lista de Inventario, y se agrega los elementos que coinciden con la búsqueda a una lista y esta se iguala a la lista tomada como parámetro.

void FiltradoProveedor(list<Inventario> &listado, string proveedor)

Esta función tiene como objetivo encontrar los medicamentos de la lista Inventario que concuerde con el mismo “Proveedor”, para ello, se recorre la lista de Inventario, y se agrega los elementos que coinciden con la búsqueda a una lista y esta se iguala a la lista tomada como parámetro.

void FiltradoNombre(list<Inventario> &listado, string buscador)

Esta función tiene como objetivo encontrar los medicamentos de la lista Inventario que concuerde con el nombre buscado en el TextBox de filtro, para ello, se recorre el listado de medicamentos y se convierte en minúsculas tanto el texto a buscar como el nombre de los medicamentos.

Luego, se verifica si la subcadena a buscar se encuentra dentro del nombre del medicamento; si se cumple la condición, agrega el medicamento a la lista inicializada, para igualarla a la lista tomada como parámetro.

void FiltradoPrincipio(list<Inventario> &listado, string buscador)

Esta función tiene como objetivo encontrar los medicamentos de la lista Inventario que concuerde con el principio activo buscado en el TextBox de filtro, para ello, se recorre el listado de medicamentos y se convierte en minúsculas tanto el texto a buscar como el principio activo de los medicamentos. Luego, se verifica si la subcadena a buscar se encuentra dentro del principio activo del medicamento; si se cumple la condición, agrega el medicamento a la lista inicializada, para igualarla a la lista tomada como parámetro.

9. Botón “Modificar Proveedor”

Al seleccionar esta opción, se despliega una pestaña con una interfaz parecida al Botón “Ingresar Proveedor”, siendo la diferencia la presencia de un ComboBox con el listado de Proveedores, ya que al seleccionar un proveedor, la información de proveedor se muestra en los textbox correspondientes.

Eventos

Evento ComboBoxActPro_SelectedIndexChanged()

Este evento sucede cada vez que se selecciona una opción en el ComboBox, mientras que la selección no sea nula; se recorrerá el listado de Proveedores, hasta que se encuentre el nombre seleccionado en el ComboBox. Al encontrar el proveedor, se muestra la información del proveedor en los TextBox correspondientes.

Evento ModificarProveedor_Click()

En este evento se redirige al usuario a la pestaña 9 de control “Menús”, además se recorre la lista de proveedores y se agrega a un ComboBox el nombre de todos los proveedores.

Evento BotónActualizarProveedor_Click()

La lógica comienza con una validación de los datos de entrada, asegurándose de que todos los campos requeridos estén completos y cumplan con los parámetros establecidos.

Una vez validados los datos, se procede a crear instancias de las clases Proveedor llamadas Objeto y Anterior. La información ingresada en los campos de texto y la selección de proveedores se asigna a las propiedades de “Objeto”.

Luego, se busca el proveedor específico que se desea actualizar dentro de la lista de proveedores. Si se encuentra, se copia su información actual antes de realizar la actualización en el objeto Anterior, lo que permite revertir los cambios en caso de algún problema.

Después, se actualizan las propiedades del proveedor con la nueva información ingresada. Para garantizar la integridad de la lista de proveedores, se realiza una verificación adicional para evitar que la nueva información coincida con datos únicos (nombre, NIT, número de teléfono) de otros proveedores en la lista. En caso de coincidencia, se muestra un mensaje de error y se restaura la información del proveedor a su estado anterior.

Si la actualización se lleva a cabo con éxito y no se encuentran coincidencias, se muestra un mensaje de éxito al usuario. Además, se actualiza la información relacionada en la lista de medicamentos que estén asociados al proveedor que se está actualizando.

La interfaz gráfica también se maneja de forma adecuada, limpiando los campos de texto y la selección del proveedor después de completar la actualización. En situaciones de error, se manejan excepciones, como la falta de coincidencia del proveedor a actualizar o la falta de información válida para la actualización.

Evento CancelarAct_Click()

En este evento se redirige al usuario al menú principal, y al mismo tiempo, se vacía todos los TextBox de la pestaña anterior, y el ComboBox con el listado de Proveedores.

10. Botón “Salir”

Al seleccionar esta opción, se cierra el formulario y termina el programa.

Eventos

Evento Salir_Click()

Este evento sucede al presionar el botón “Salir”, lo que realiza es el cierre del formulario principal, terminado el programa.

- **Restricciones**

  - **Proveedores**

- ➢ Se verifica que el NIT sea un número de 9 dígitos.
- ➢ Se verifica si el número de teléfono tiene 8 dígitos
- ➢ Se verifica si el correo está en el formato correcto..
- ➢ Se verifica que no exista información única repetida (Nombre,NIT,Teléfono)

- **Inventario**

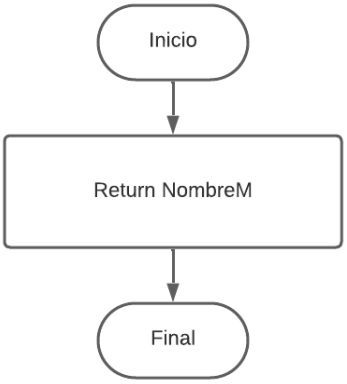
  - ➢ Se verifica que ningún campo de información esté vacío.
  - ➢ Se limita la entrada válida en los TextBox que involucran números, para que el usuario solo pueda ingresar números, no letras.
  - ➢ Se verifica que la dosis recomendada sea una cantidad mayor que cero.
  - ➢ Se verifica que el precio de compra y venta no sea menor que 0..
  - ➢ Se verifica que la fecha ingresada sea una fecha válida.

- **Otras Restricciones**

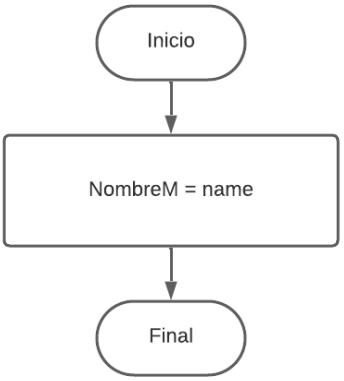
  - ➢ Se verifica que el campo de información de búsqueda esté lleno
  - ➢ Se verifica que el usuario solo pueda seleccionar la fila completa en la tabla de inventario.
  - ➢ Se verifica que el usuario haya elegido un criterio o parámetro en los procesos de búsqueda.

Diseño

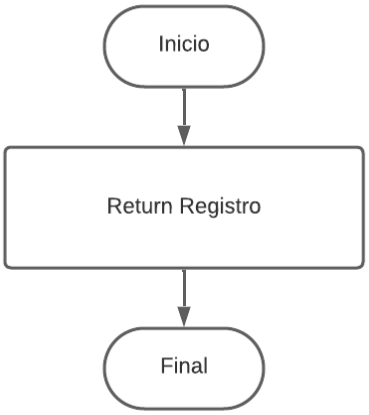
- std::string GetName();



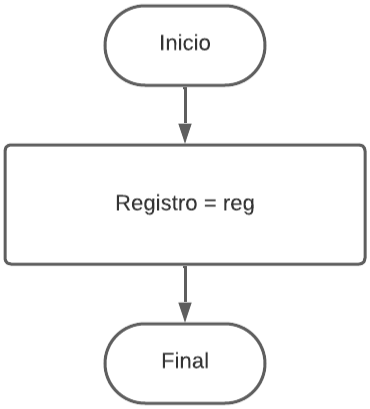
- void SetName(std::string Name);



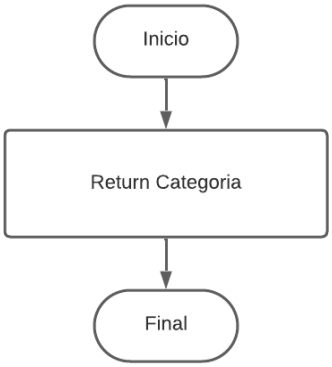
- int GetRegistro();



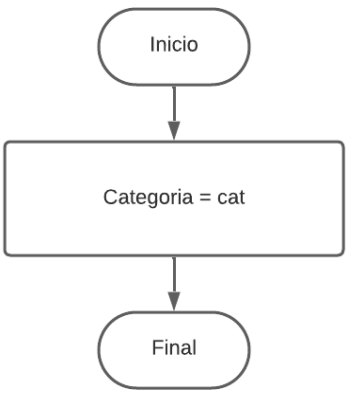
- void SetRegistro(int reg);



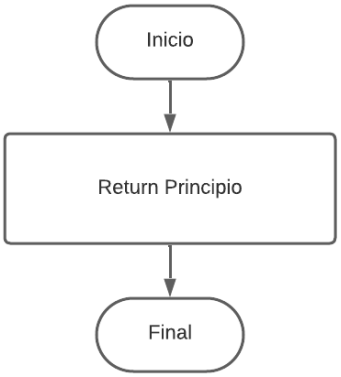
- std::string GetCategoria();



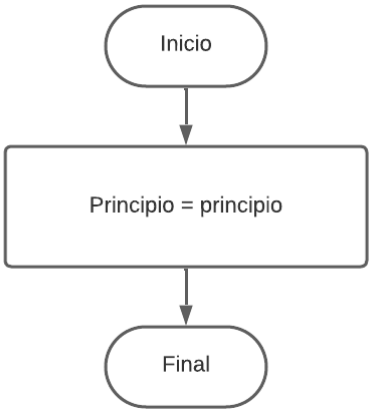
- void SetCategoria(std::string cat);



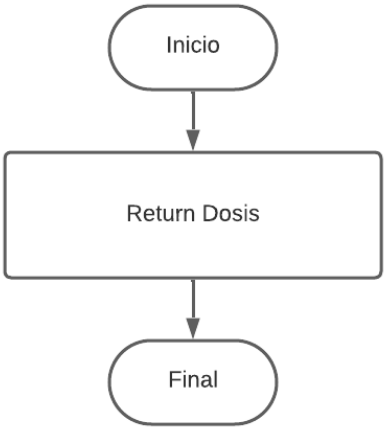
- std::string GetPrincipio();



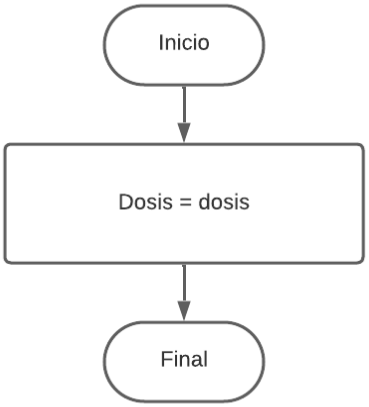
- void SetPrincipio(std::string principio);



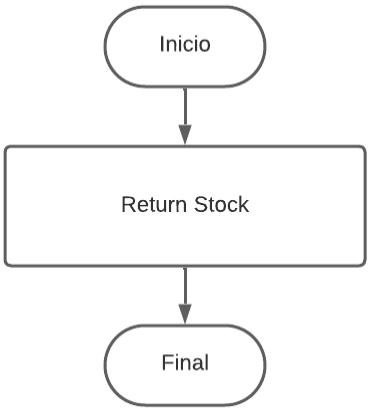
- int GetDosis();



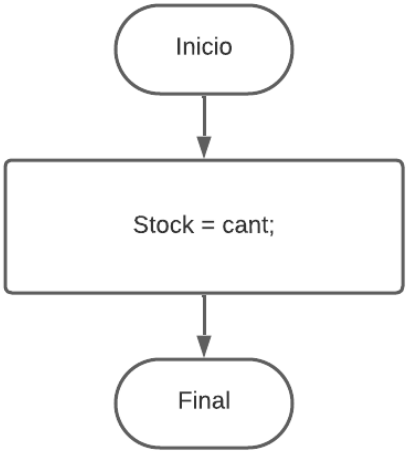
- void SetDosis(int dosis);



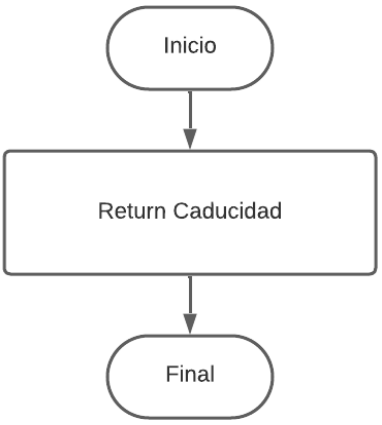
- int GetStock();



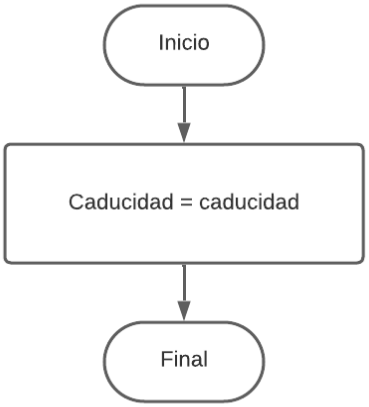
- void SetStock(int cant);



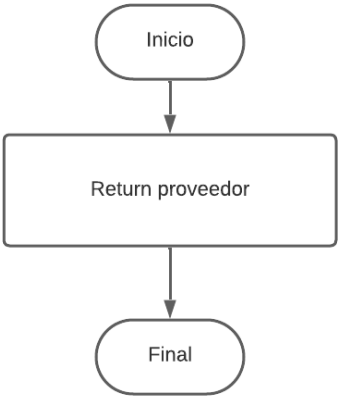
- std::string GetCaducidad();



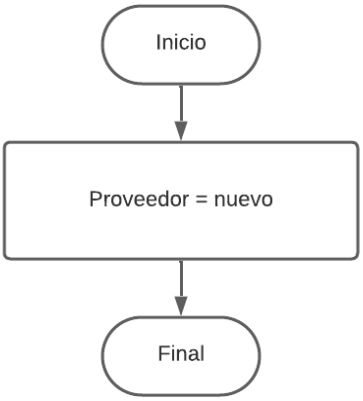
- void SetCaducidad(std::string caducidad);



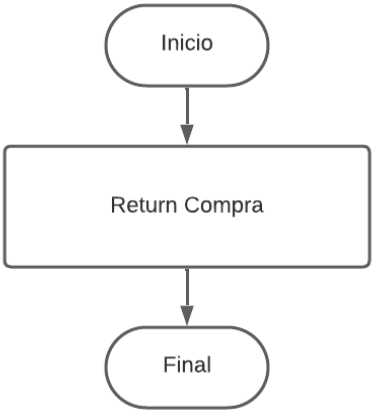
- Proveedor GetProveedor();



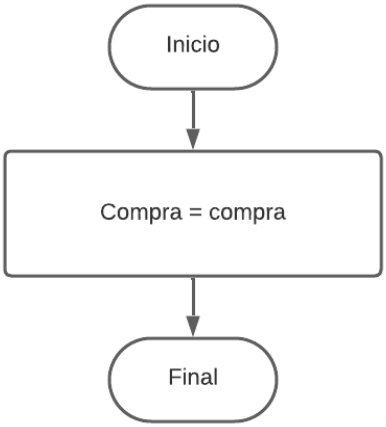
- void SetProveedor(Proveedor nuevo);



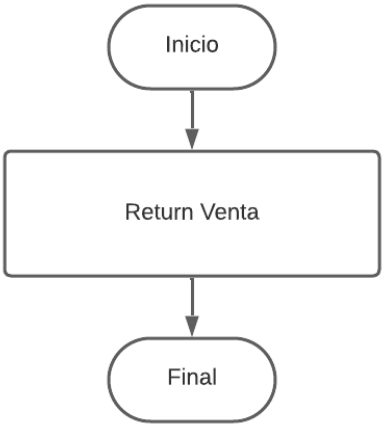
- double GetCompra();



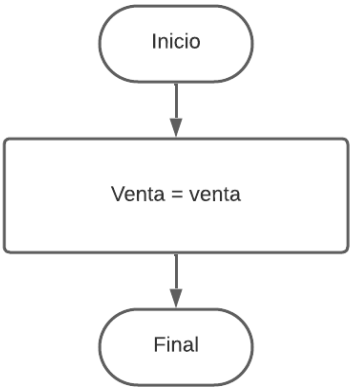
- void SetCompra(double compra);



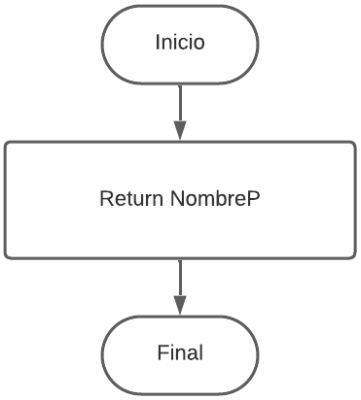
- double GetVenta();



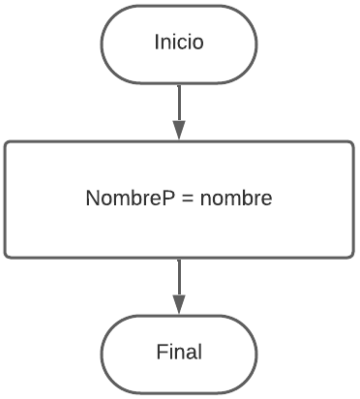
- void SetVenta(double venta);



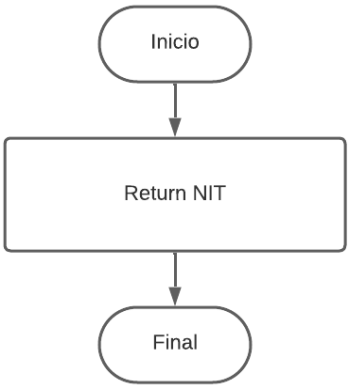
- std::string GetNombreP();



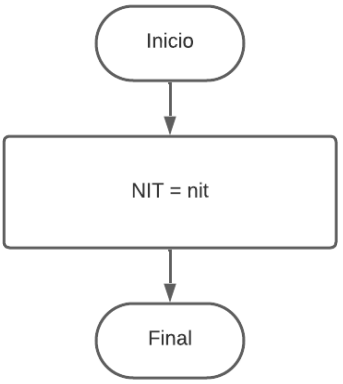
- void SetNombreP(std::string nombre);



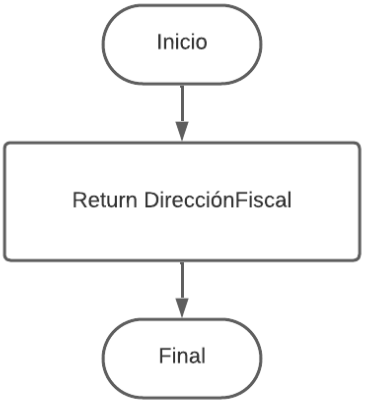
- int GetNIT();



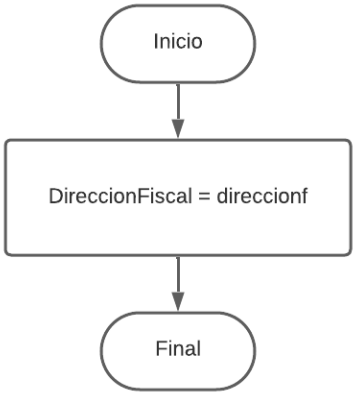
- void SetNIT(int nit);



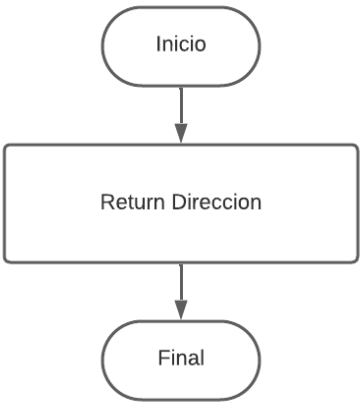
- std::string GetDireccionF();



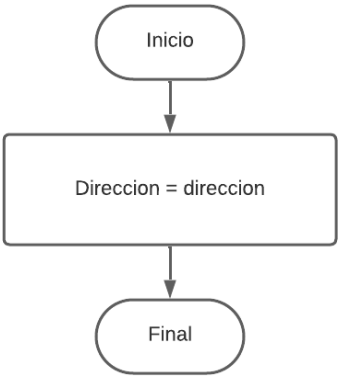
- void SetDireccionF(std::string direccionf);



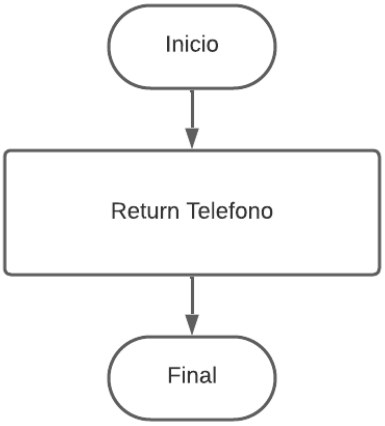
- std::string GetDireccion();



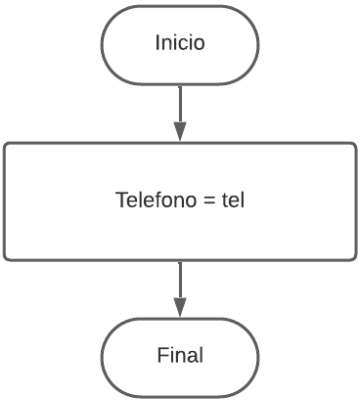
- void SetDireccion(std::string direccion);



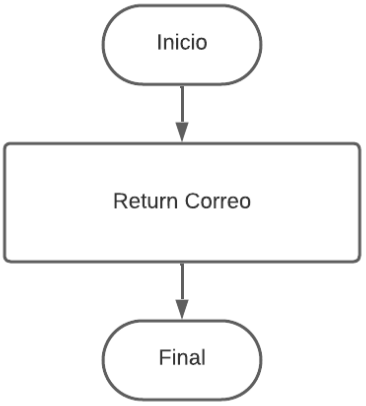
- int GetTel();



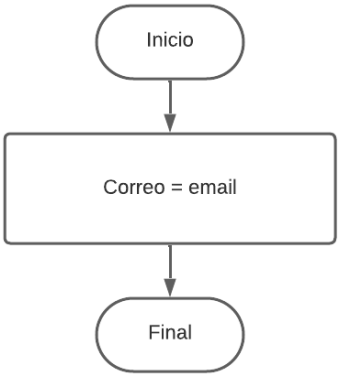
- void SetTel(int telefono);



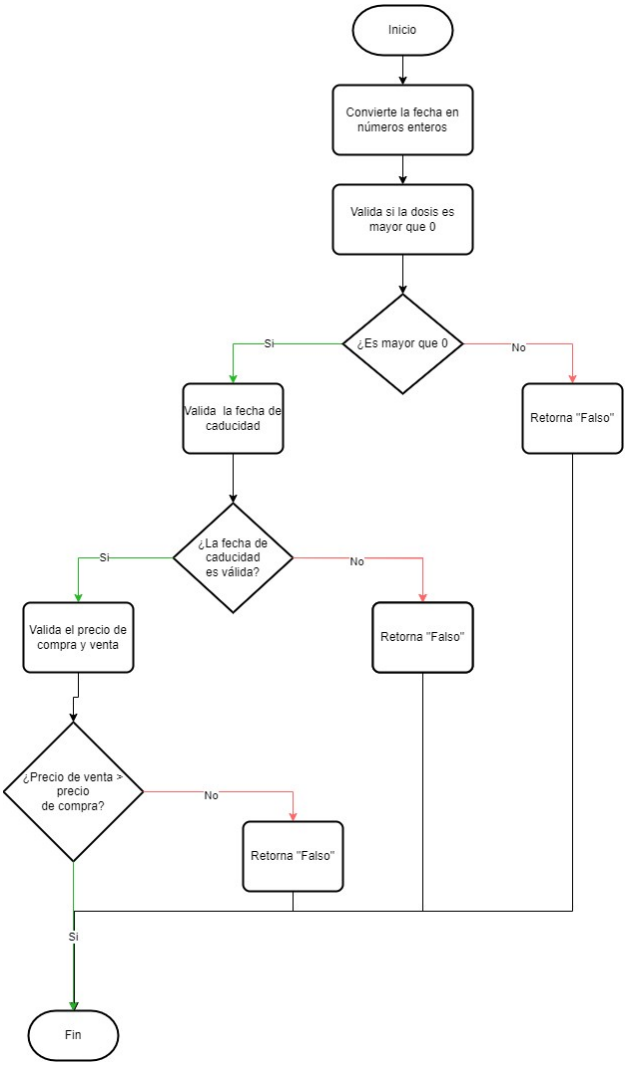
- std::string GetEmail();



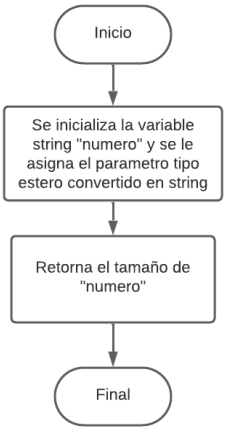
- void SetEmail(std::string email);



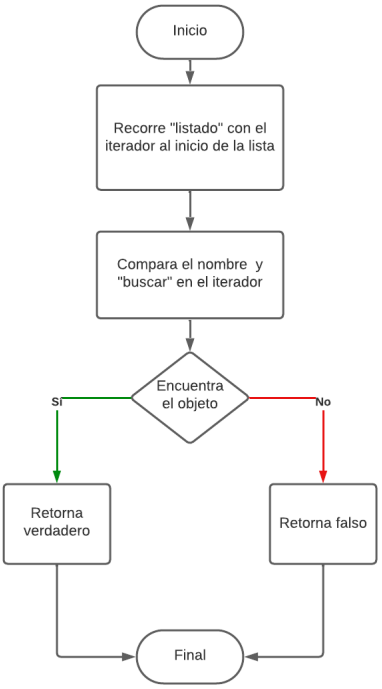
- bool ValidarInfo(int dosis, std::string año, std::string dia, std::string mes, double compra, double venta, int stock)



- int ContadorNum(int num);

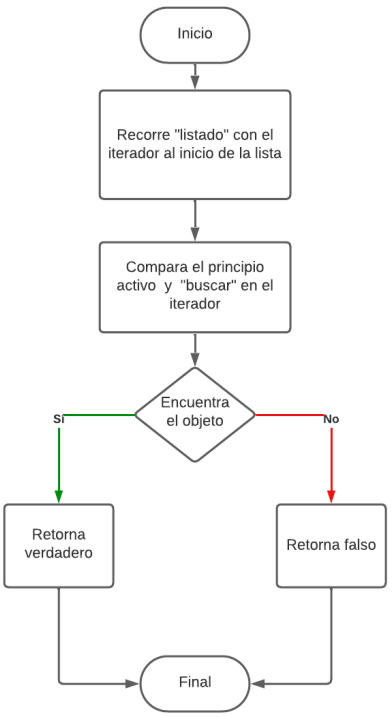


- bool BusquedaEncontradaNombre(std::list<Inventario> Listado, std::string buscar)

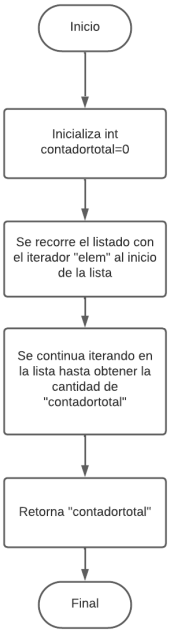


-

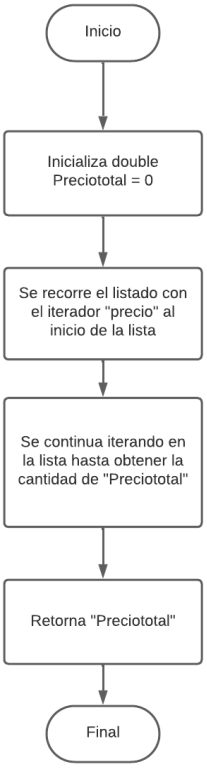
bool BusquedaEncontradaPrincipio(std::list<Inventario> Listado, std::string buscar)



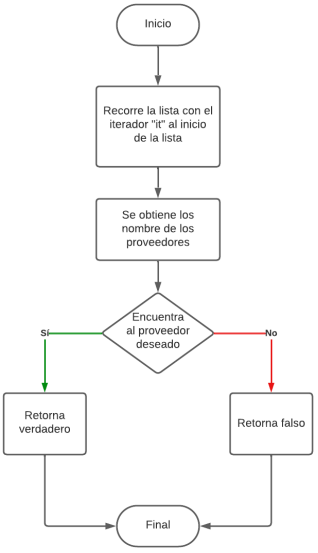
- int TotalMedicamentos(std::list<Inventario> Listado)



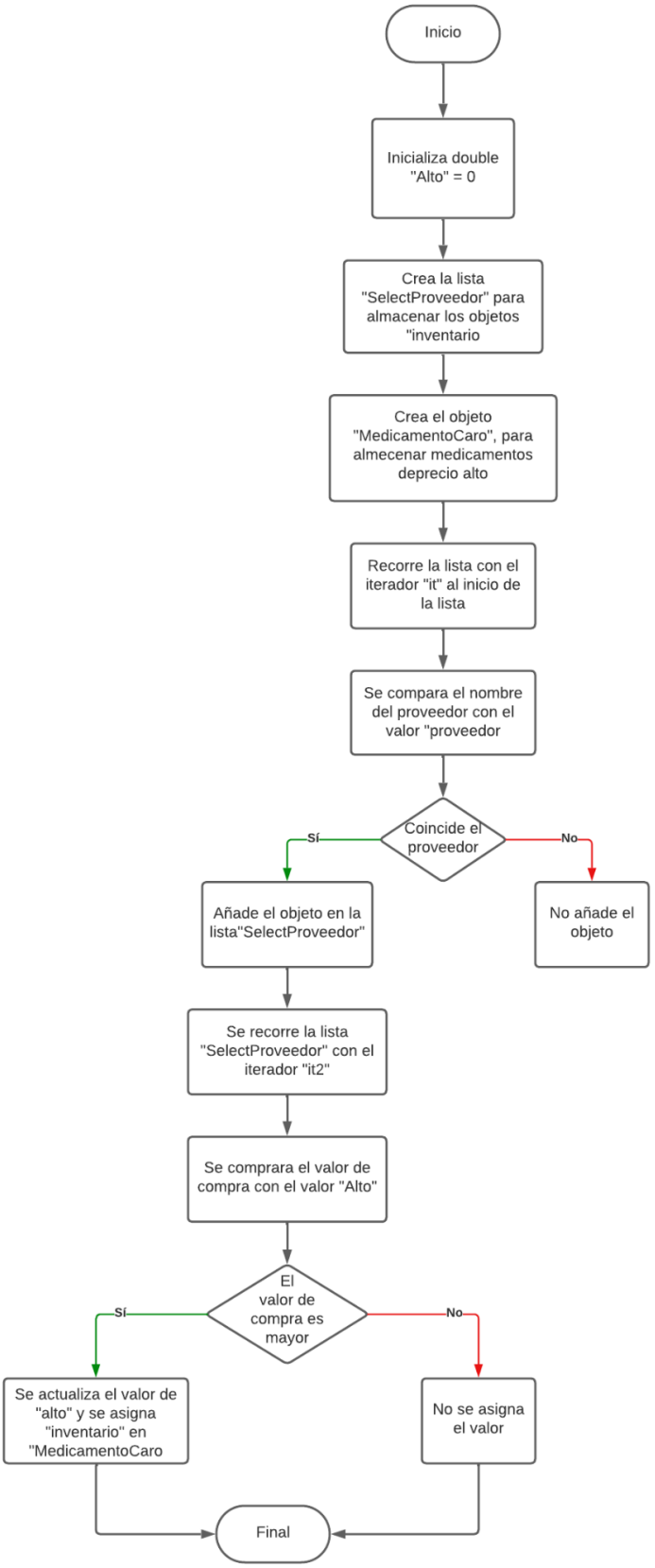
- double TotalPrecios(std::list<Inventario> Listado);



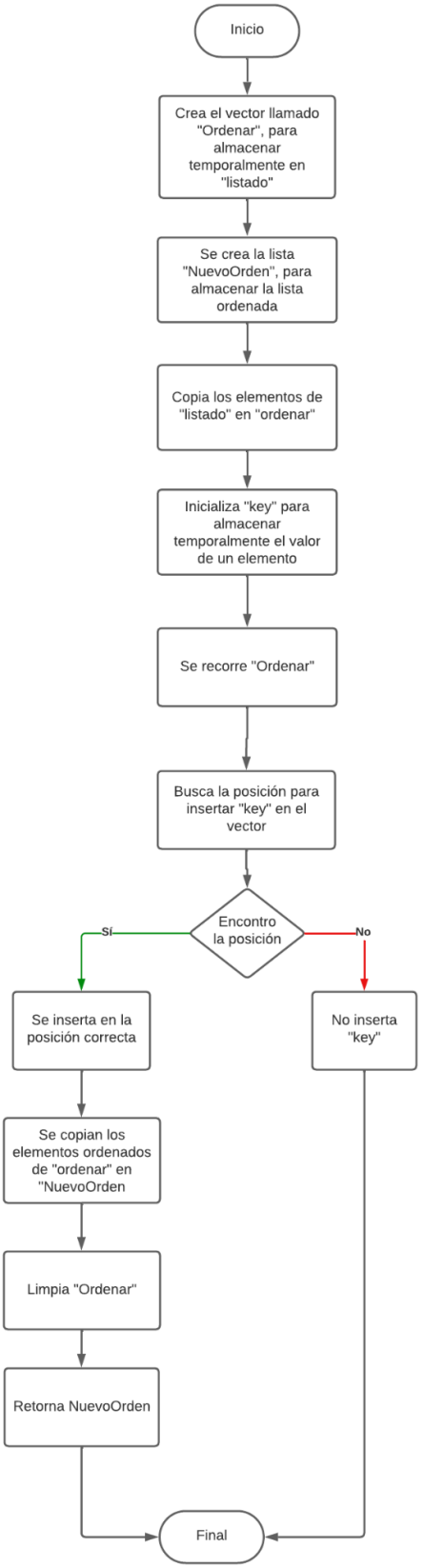
- bool ComprobarProveedor(std::list<Inventario> proveedores, std::string proveedor);



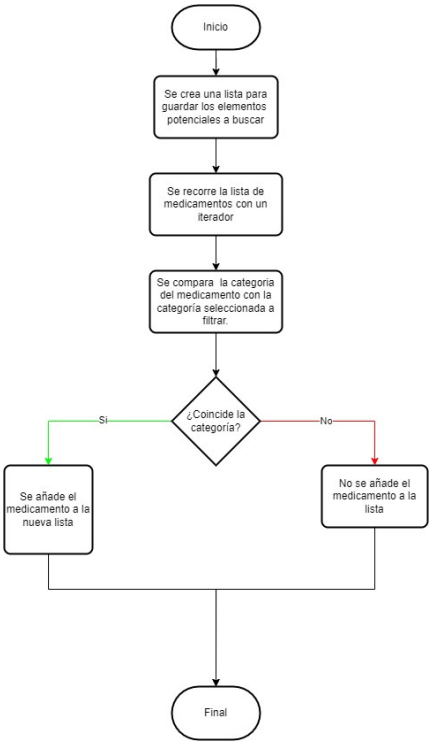
- Inventario PrecioMasAlto(std::list<Inventario> listado, std::string proveedor);



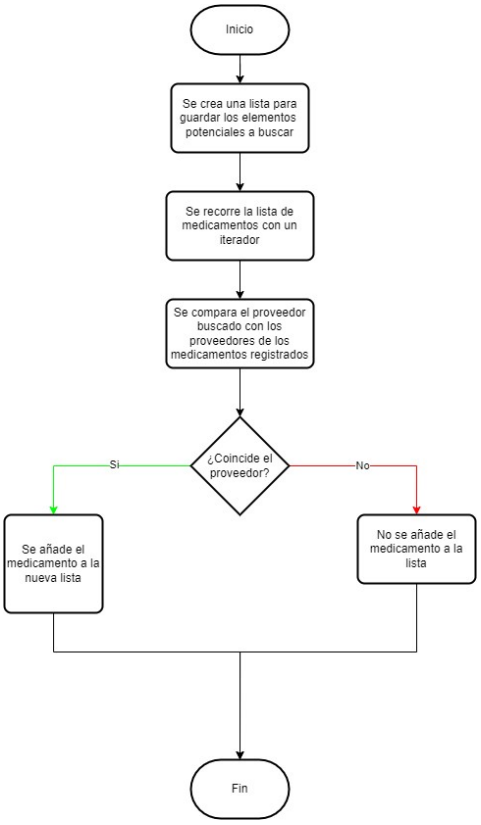
- list<Inventario> OrdenamientoPorNombre(std::list<Inventario> listado)



- void FiltradoCategoria(std::list<Inventario>& listado, std::string categoria)

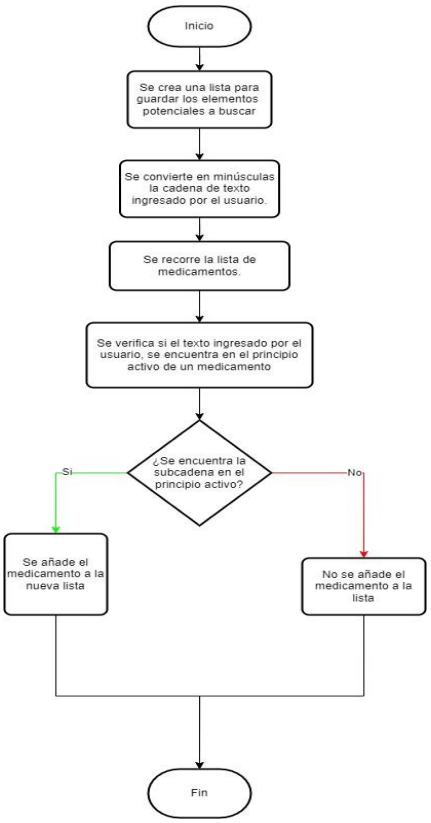


- void FiltradoProveedor(std::list<Inventario>& listado, std::string proveedor)

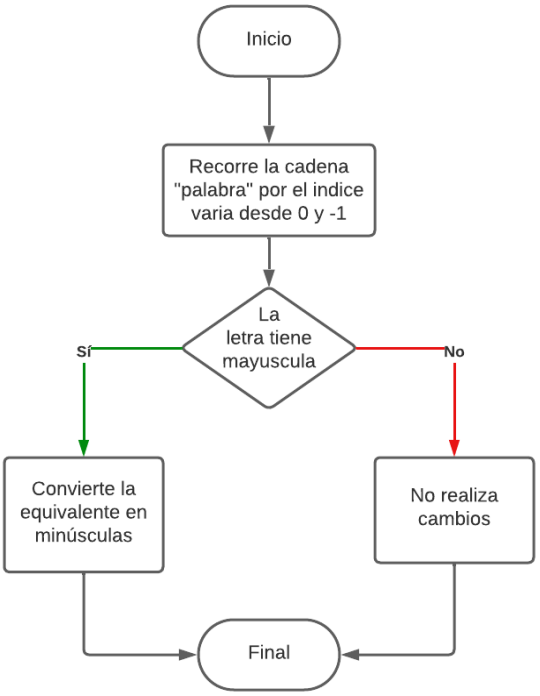


- void FiltradoNombre(std::list<Inventario>& listado, std::string buscador)

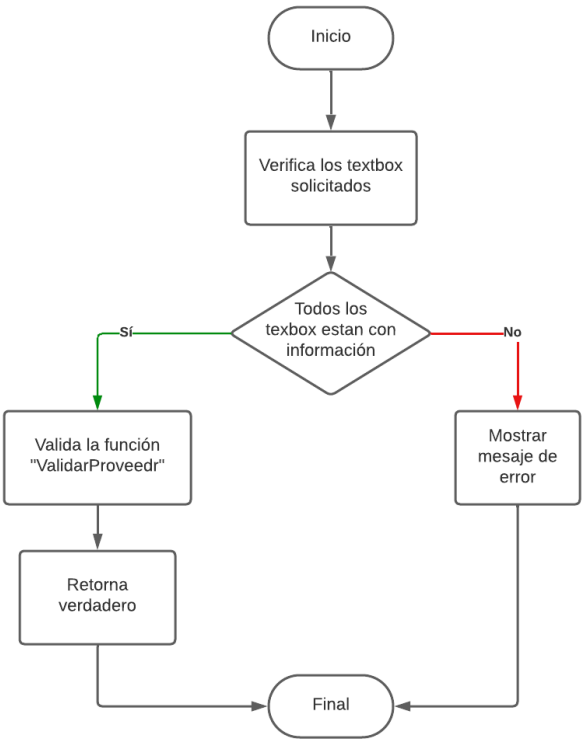
- void FiltradoPrincipio(std::list<Inventario>& listado, std::string buscador);



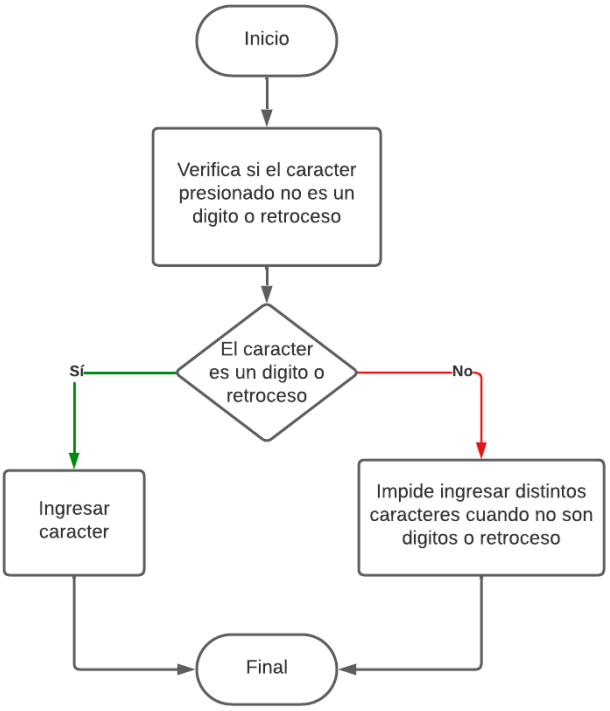
- std::string Minúsculas(std::string palabra);



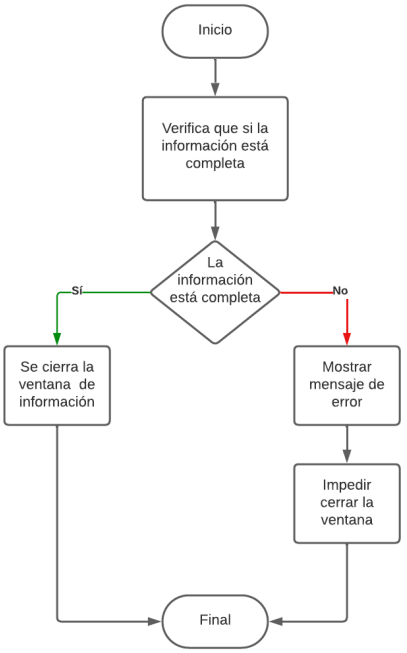
- Void Form1_ button1_Click(System::Object^ sender, System::EventArgs^ e)



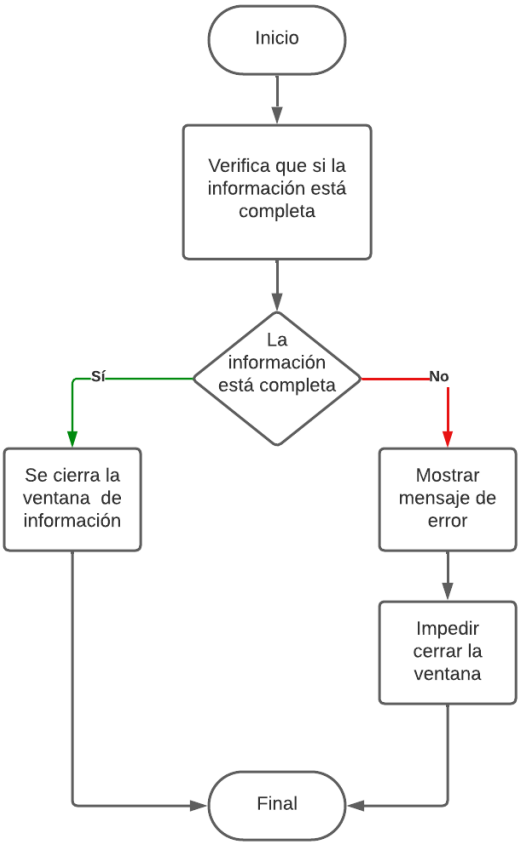
- VoidKeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e)



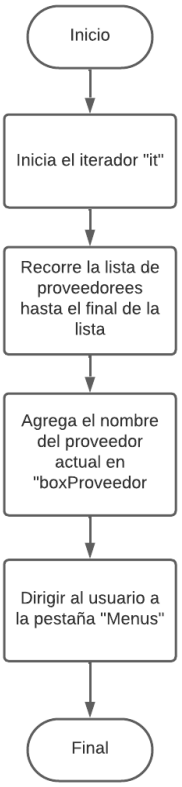
- Void MyForm1_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e)



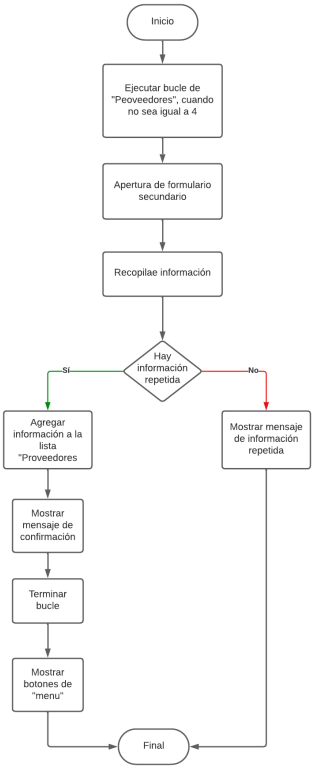
- Void Form2_BotonActualizar_Click(System::Object^ sender, System::EventArgs^ e)



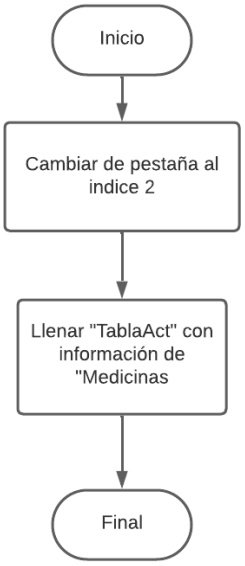
- Void BotonNuevoMedicamento_Click(System::Object^ sender, System::EventArgs^ e)



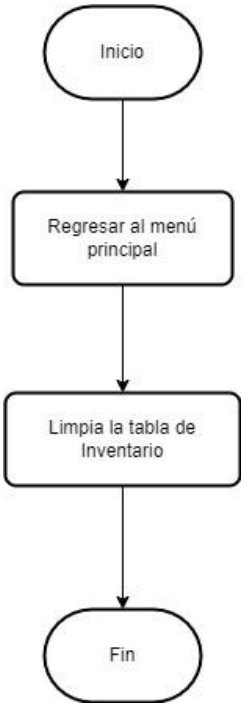
- Void button11_Click(System::Object^ sender, System::EventArgs^ e)



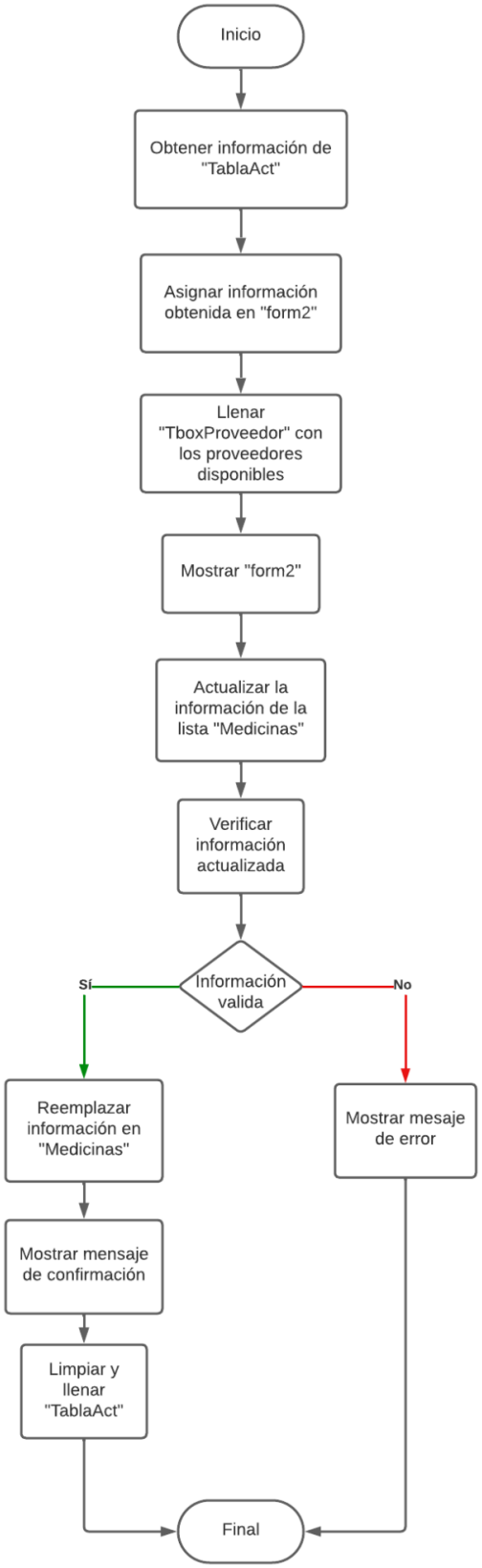
- Void BotonActualizarInventario_Click(System::Object^ sender, System::EventArgs^ e)



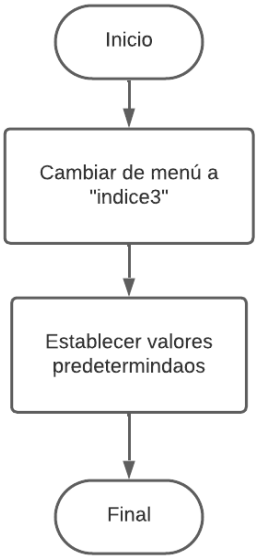
- Void Regresar2_Click(System::Object^ sender, System::EventArgs^ e)



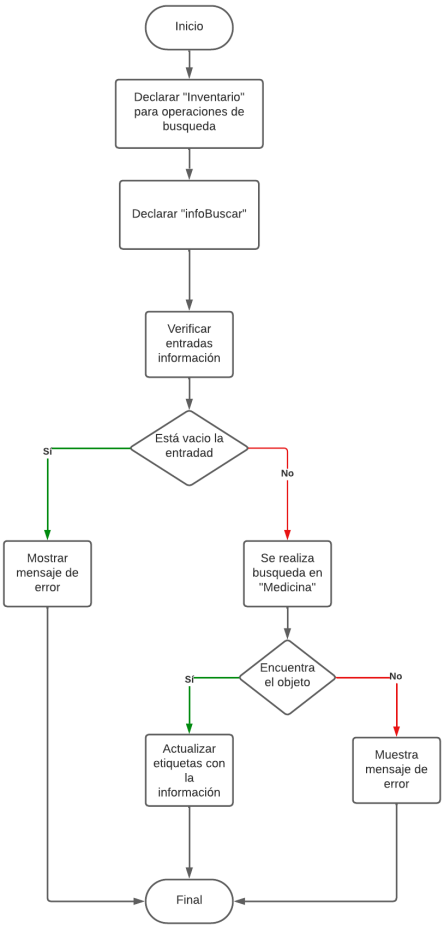
Void TablaAct_DoubleClick(System::Object^ sender, System::EventArgs^ e)



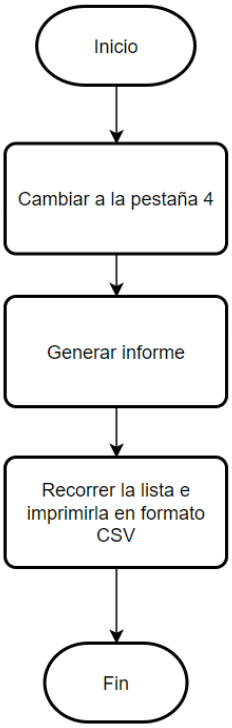
- Void BotonConsultarInformación_Click(System::Object^ sender, System::EventArgs^ e)



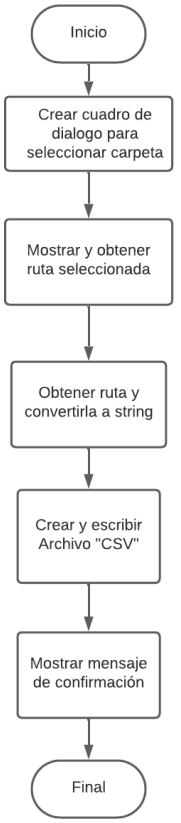
- Void Buscar_Click(System::Object^ sender, System::EventArgs^ e)



- Void GenerarInforme_Click(System::Object^ sender, System::EventArgs^ e)

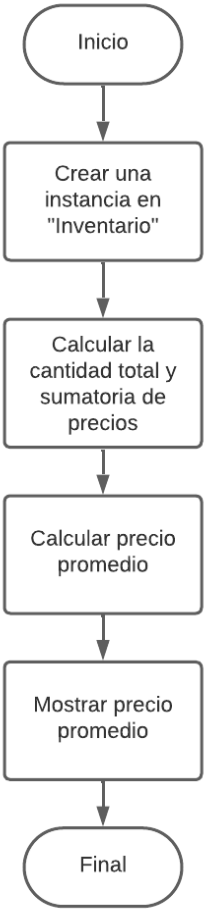


- VoidExportar_Click(System::Object^ sender, System::EventArgs^ e)

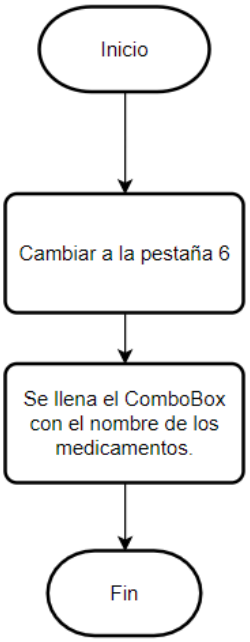


- PrecioPromedio_Click(System::Object^ sender, System::EventArgs^ e)

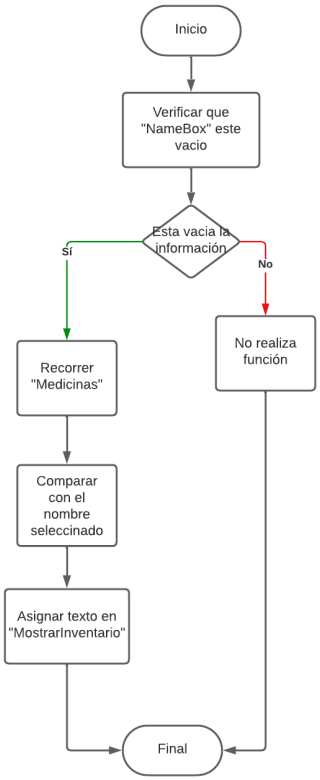
- CalcularPrecio_Click(System::Object^ sender, System::EventArgs^ e)



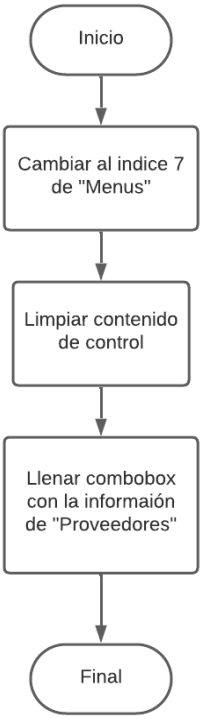
- Void InventariodelMedicamento_Click(System::Object^ sender, System::EventArgs^ e)



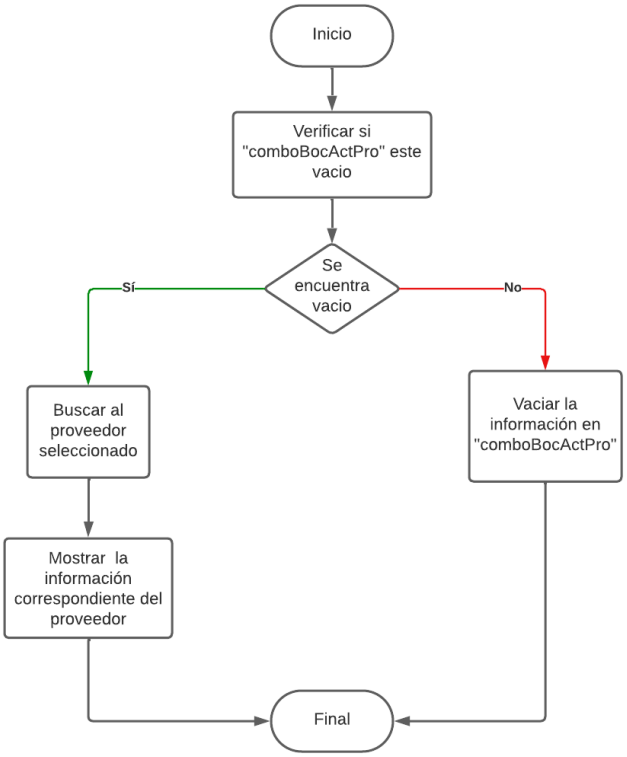
- Void NameBox_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e)



- Void PreciomásAlto_Click(System::Object^ sender, System::EventArgs^ e)



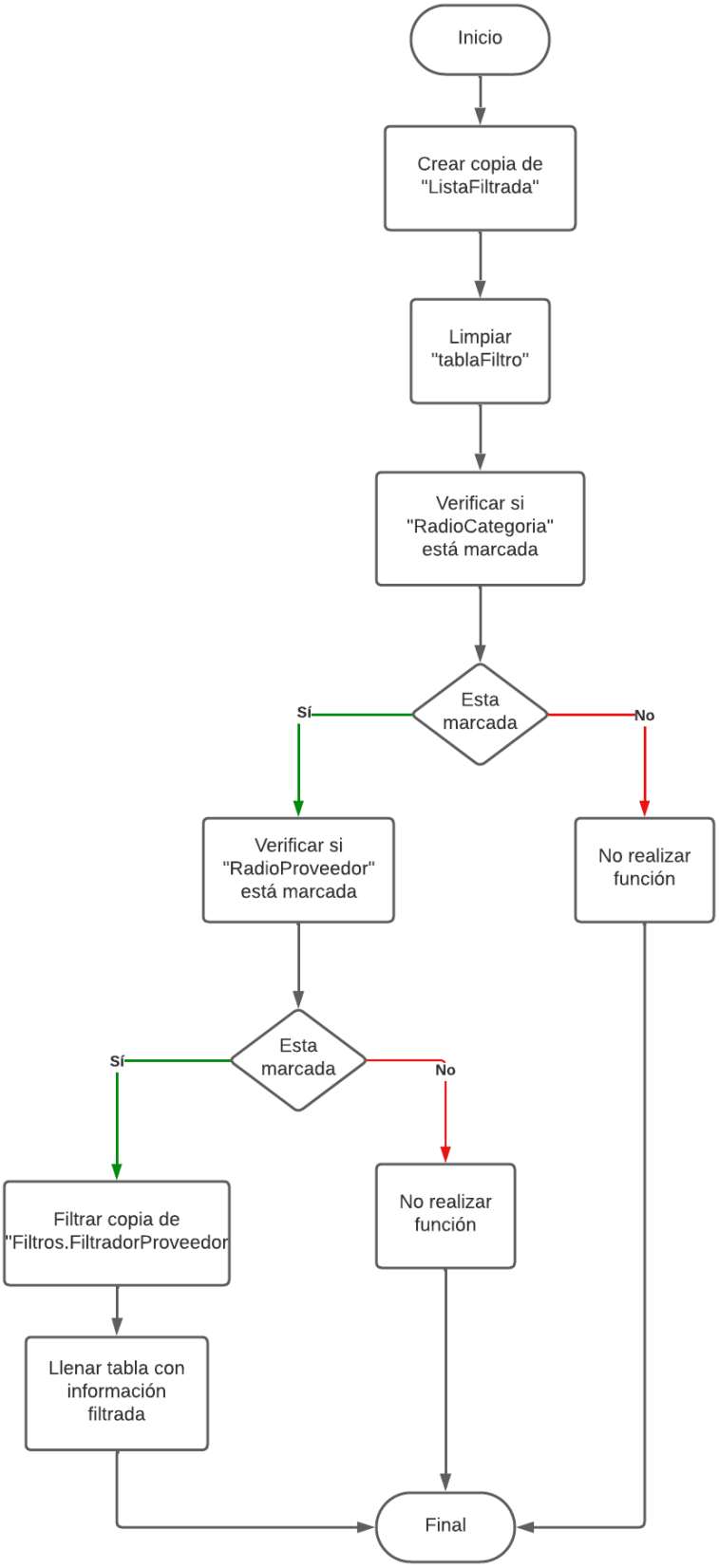
- Void ProveedorBox_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e)



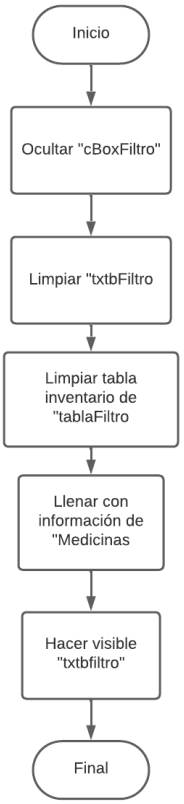
- Void BusquedayFiltrado_Click(System::Object^ sender, System::EventArgs^ e)

- Void radioProveedor_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

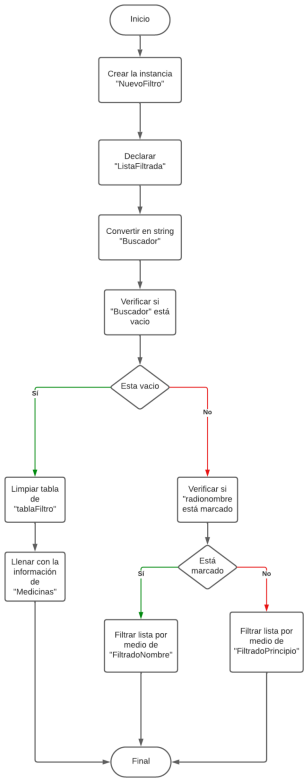
- Void ComboBoxFiltro_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e)



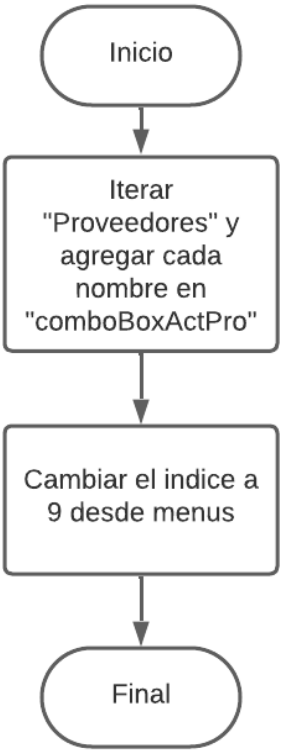
- Void radioName_CheckedChanged_CheckedChanged(System::Object^ sender, System::EventArgs^ e)



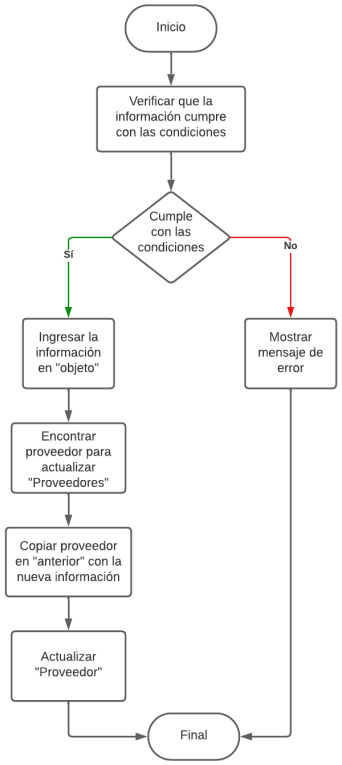
- Void TextBoxfiltro_TextChanged(System::Object^ sender, System::EventArgs^ e)



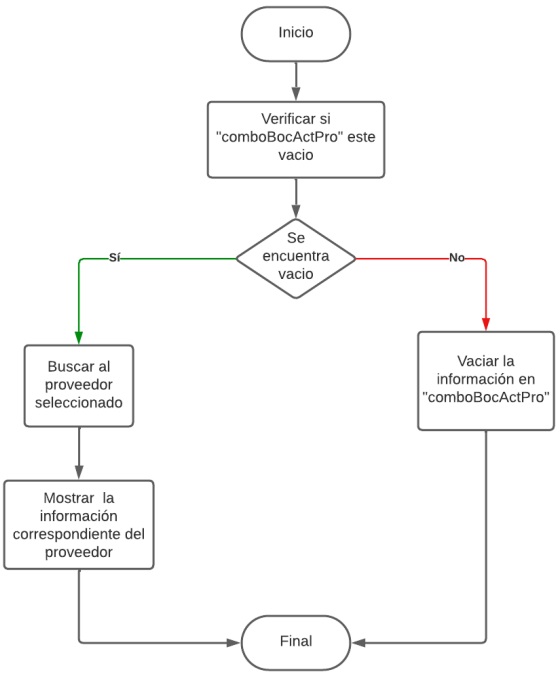
- Void ModificarProveedor_Click(System::Object^ sender, System::EventArgs^ e)



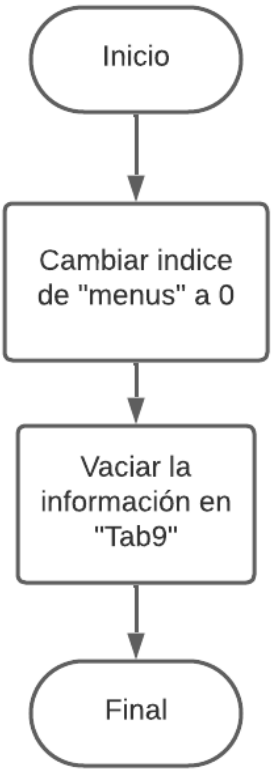
- Void BotónActualizarProveedor_Click(System::Object^ sender, System::EventArgs^ e)



- Void ComboBoxActPro_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e)



- Void.CancelarAct_Click(System::Object^ sender, System::EventArgs^ e)



- Void Salir_Click(System::Object^ sender, System::EventArgs^ e)

Conclusiones

El programa brinda a los usuarios una herramienta para la eficiente gestión y organización del inventario de medicamentos en una farmacia. Gracias a su capacidad avanzada de manejo de información, los usuarios pueden realizar tareas fundamentales como la actualización de datos de proveedores, asegurando la precisión y consistencia de la información. La interfaz intuitiva facilita la navegación y el acceso a funciones clave

El programa requiere que el usuario tenga un cierto conocimiento sobre los productos o el área farmaceútica, ya esto le permitirá utilizar el programa de forma más intuitiva y fácil; sin embargo, se contempla que existan ciertos errores en el ingreso y manejo de información; tal que, un usuario no tan conocedor en el área farmaceútica pueda utilizar el programa adecuadamente.

El programa proporciona estadísticas útiles sobre el inventario, como el total de medicamentos disponibles y el cálculo del precio promedio. Estas herramientas analíticas no solo facilitan la toma de decisiones informadas para los usuarios, sino que también les ofrecen una perspectiva completa y detallada de la salud de su inventario. La capacidad de obtener rápidamente información crucial contribuye a una gestión más efectiva y a la toma de decisiones estratégicas que impulsan el rendimiento general de la farmacia.

Recomendaciones

Se recomienda el uso de una librería para el manejo de directorios y archivos, si es posible, utilizar otra librería diferente de dirent.h, debido a que su instalación es compleja y, si no se conoce el procedimiento para su instalación, se puede derivar en problemas de compilación y ejecución.

Se recomienda el uso de “listas” y “colas” para el manejo de un listado ordenado de elementos, en este caso, las canciones y CD’s para representar un listado y una cola de reproducción; asimismo, el uso de “vectores” debido a la amplias opciones que ofrece para manipular los elementos dentro del vector, permitiendo ordenarlo según ciertos parámetros establecidos

Se recomienda el uso de punteros para referirse a un valor que debe permutar a lo largo del tiempo, modificando y obteniendo este valor por medio de funciones get y set, a su vez, utilizar parámetros por referencia en las funciones correspondientes, volviendo más eficiente la ejecución del programa.

Referencias

- **fstream:** Biblioteca que define distintas clases que admiten operaciones de iostream en archivos externos almacenados. Se utilizó está biblioteca para obtener y leer la ubicación de los archivos que se encuentren en el ordenador..

https://learn.microsoft.com/es-es/cpp/standard-library/fstream?view=msvc-170

- **iostream:** Biblioteca que declara los objetos que controlan la lectura y escritura, suele ser el único encabezado para realizar entradas y salidas. Se hizo uso de la biblioteca para operaciones de entradas y salidas en el proyecto

https://learn.microsoft.com/es-es/cpp/standard-library/iostream?view=msvc-170

- **msclr/marshal_cppstd:** Biblioteca que proporciona funciones para facilitar la interoperabilidad entre código administrado y código no administrado. Se utilizó para convertir los tipos de datos de C++ estándar (como string) y tipos de datos NET (como String^) y viceversa.

https://learn.microsoft.com/en-us/cpp/dotnet/overview-of-marshaling-in-cpp?view=msvc-170

- **list:** Biblioteca que proporciona una serie de funciones y métodos que permiten la manipulación de elementos en la lista, como la inserción y eliminación eficientes en cualquier posición, así como el acceso a los elementos mediante iteradores; se utilizó esta biblioteca para el manejo de información global de los medicamentos y proveedores, así como la capacidad de recorrer el listado de información.

https://learn.microsoft.com/es-es/cpp/standard-library/list-class?view=msvc-170

- **Windows:** Biblioteca que proporciona un conjunto de funciones y macros que permiten a los programadores interactuar con el sistema operativo Windows, se utilizó para la implementación de ventanas emergentes para la selección de directorio en el equipo del usuario.

https://www.aprendeaprogramar.com/mod/forum/discuss.php?d=738

- **vcclr:** Biblioteca que permite a los desarrolladores escribir código en C++ y utilizar características del Common Language Runtime (CLR) de Microsoft, que es el entorno de ejecución de .NET. Se utilizó esta biblioteca para la conversión entre Tipos Nativos de C++ y Tipos Gestionados de .NET.

https://learn.microsoft.com/es-es/cpp/dotnet/how-to-declare-handles-in-native-types?view=msvc-170

Anexo

- **Manual de usuario:**

Bienvenidos a la aplicación “Sistema de Información Farmacias”, a continuación se expondrá y se explicará las funcionalidades que ofrece la misma; del mismo modo, los pasos a seguir para utilizar correctamente la aplicación de música..

Menú de Inicio

Al iniciar la aplicación, se presenta un Menú de Inicio con una opción:

1. Ingresar Proveedor

Al seleccionar la opción (“Ingresar Proveedor”), se abrirá una ventana emergente en la que puede ingresar la información de los cuatro proveedores en la farmacia, es necesario que ingrese correctamente toda la información, debido a que si se detecta una información inválida o repetida se mostrará un mensaje de error, y se le notificará que tipo de error ocurrió en el ingreso de información; del mismo modo, si todos los campos fueron ingresados correctamente se mostrará un mensaje de confirmación de la nueva entrada.

*Nota: No podrá cerrar esta ventana hasta que ingrese los 4 proveedores correspondientes

Menú Principal

Al terminar el ingreso de información de los proveedores de medicamentos de la farmacia, se desplegará el menú principal con las siguientes opciones.

1. Nuevo Medicamento
2. Actualizar Inventario
3. Consultar Información
4. Generar Informe
5. Precio Promedio
6. Inventario del medicamento
7. Precio más alto
8. Búsqueda de Medicamentos
9. Salir

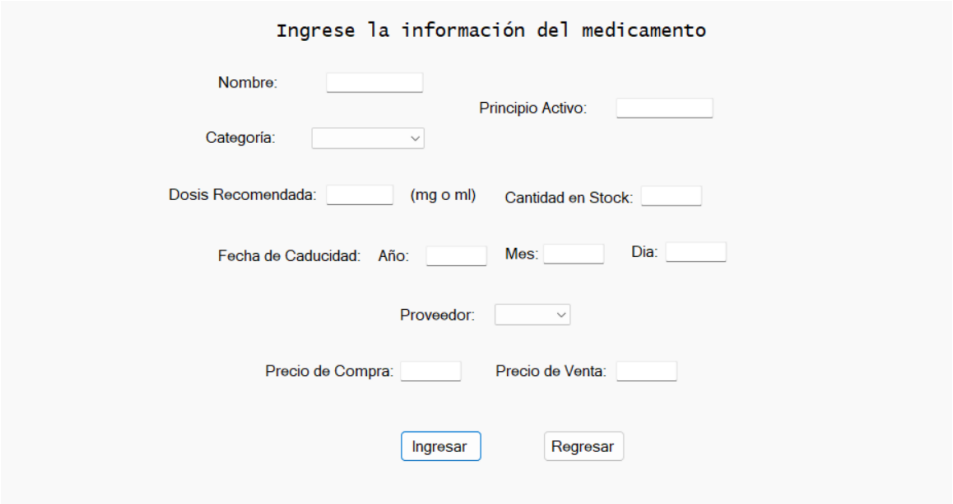
y una opción adicional:

- Modificar Proveedor

A continuación se detallan la función de cada opción del menú mostrado:

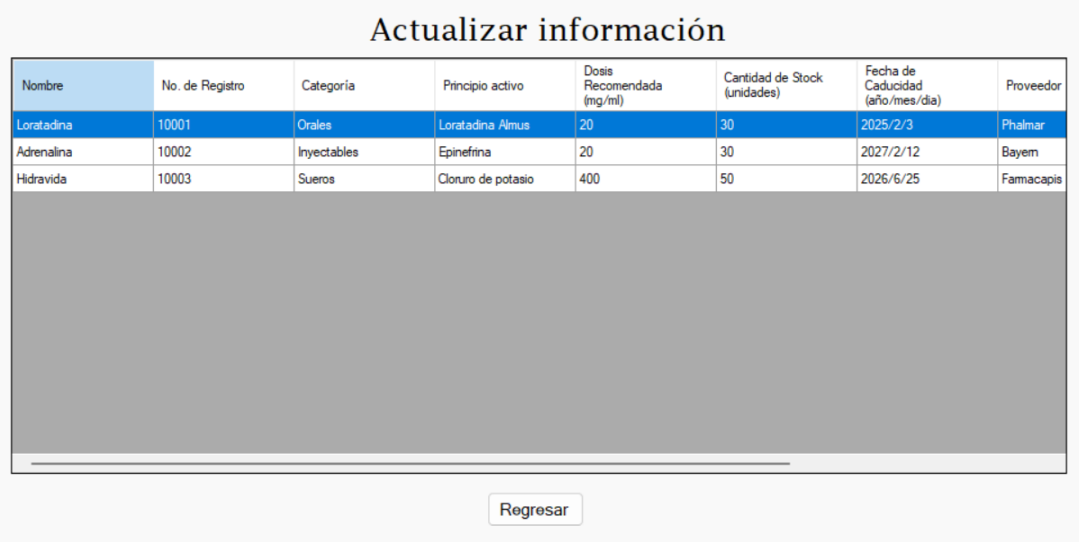
Nuevo Medicamento

Al seleccionar la opción “Nuevo Medicamento”, se desplegará una pestaña en la que se solicita toda la información sobre el medicamento a ingresar, se recomienda tener disponible la información del medicamento para ingresar correctamente todos los campos de información, al terminar de llenar todos los campos, presione el botón Ingresar para añadir el medicamento al inventario.

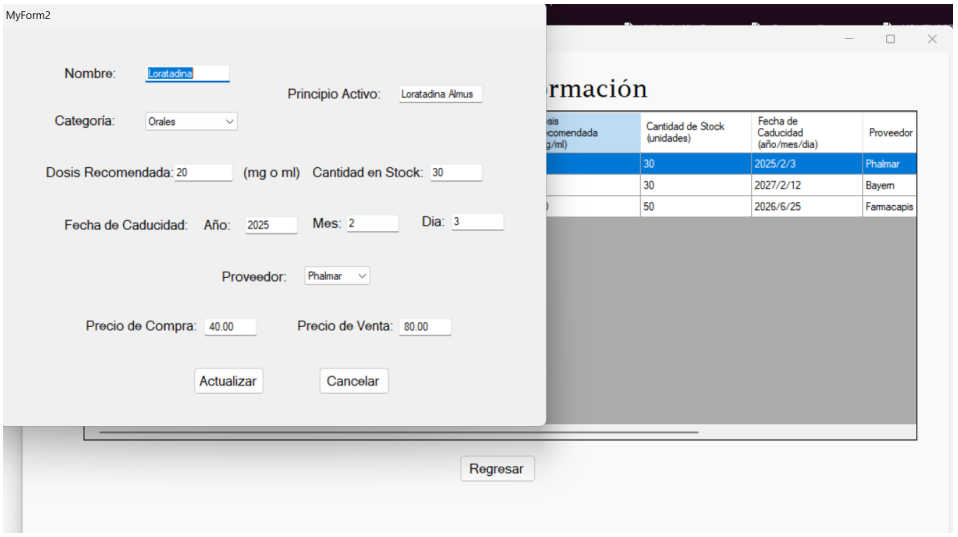


Actualizar Inventario

Al seleccionar esta opción, se mostrará en una tabla todos los medicamentos en el inventario, con la información más importante de cada medicamento.

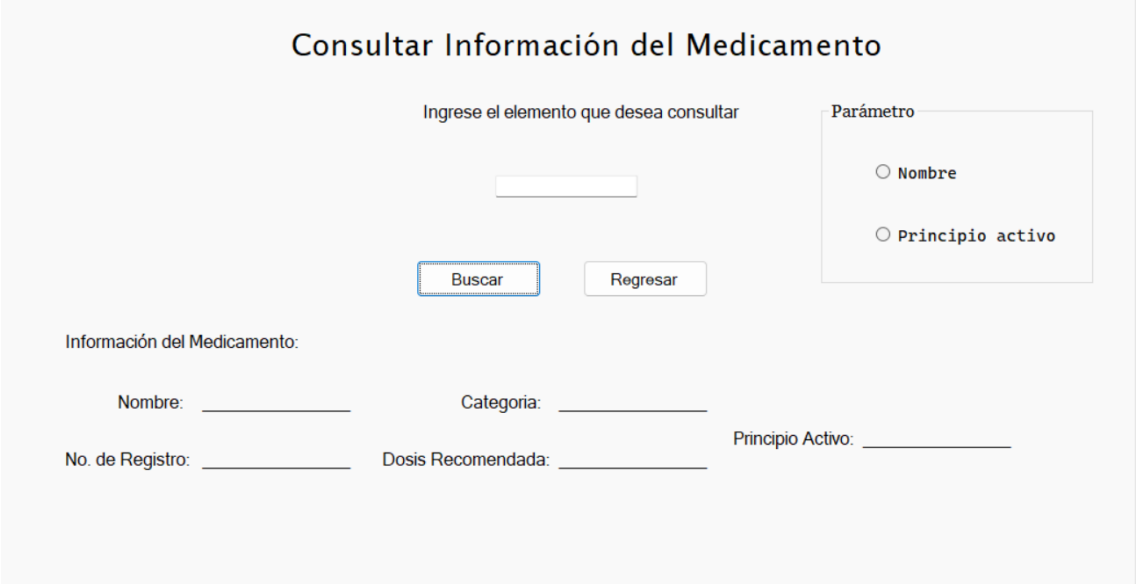


Al realizar un “doble click” sobre una fila del inventario, se abrirá una ventana emergente con toda la información del medicamento; desde esa ventana, es posible modificar la información de inventario del medicamento, dicha actualización se verá reflejada inmediatamente en la tabla de Inventario.

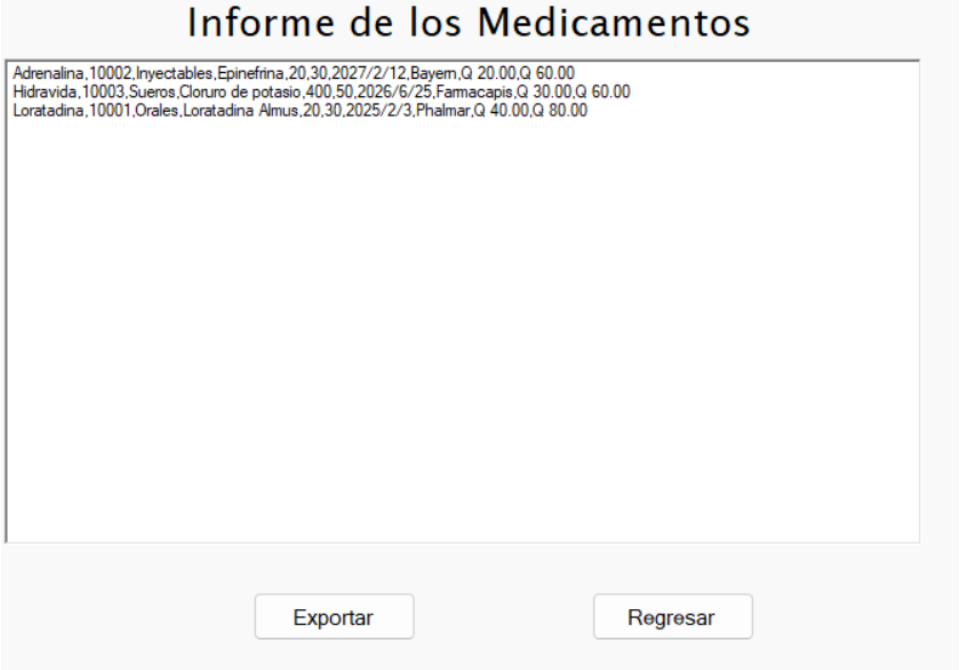


Consultar Información

Al seleccionar esta opción, se mostrará un menú para que el usuario pueda consultar la información de un medicamento por el nombre del mismo o su principio activo, es importante elegir un criterio de búsqueda antes de consultar la información, además, para que la búsqueda se complete correctamente debe ingresar el nombre o principio activo con el que se registró en el inventario.

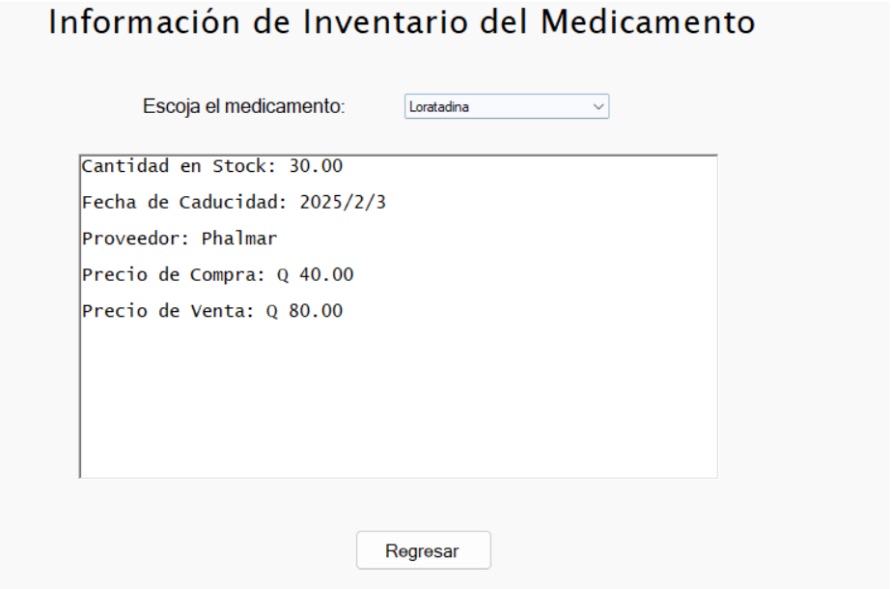


Generar Informe
Al seleccionar esta opción, se mostrará un informe de todos los medicamentos registrados en el inventario de la farmacia, con la opción de “Exportar” el informe realizado en un archivo CSV, con el cual se puede generar una tabla de información en el Programa Microsoft Excel.



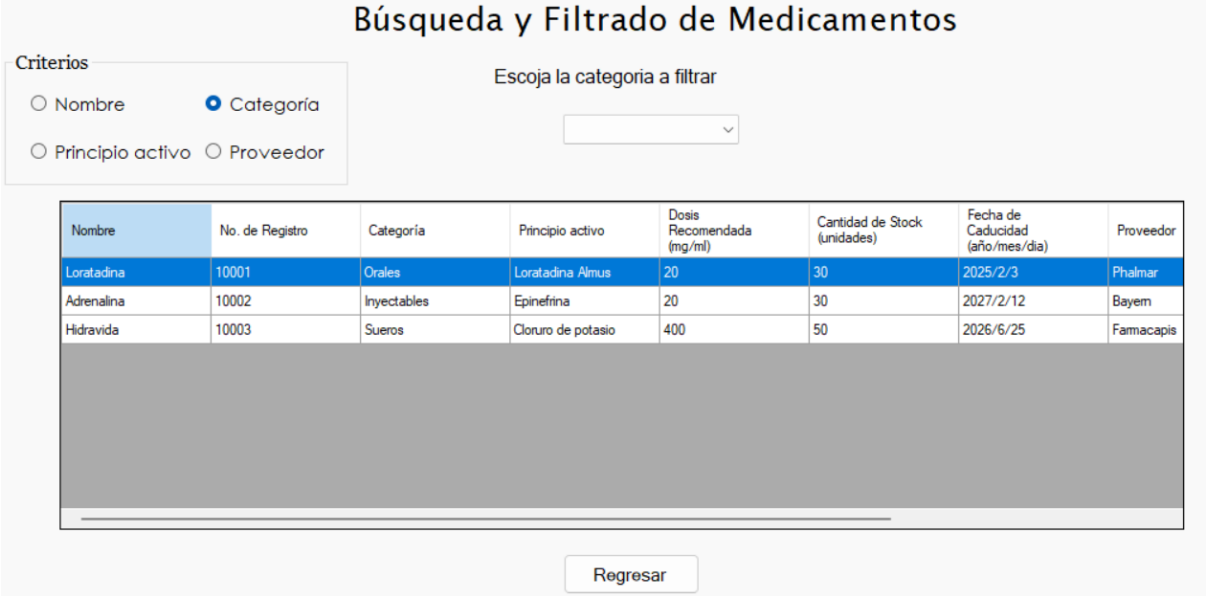
Precio Promedio
Al seleccionar esta opción, se mostrará en pantalla la cantidad de medicamentos disponibles en la farmacia y el precio total de compra de los medicamentos, al presionar el botón “Calcular” se mostrará en pantalla, el precio promedio de compra de cada medicamento disponible en el inventario.

Inventario del medicamento
Al seleccionar esta opción, se mostrará un menú en que se le solicita al usuario escoger un medicamento dentro de todos los medicamentos del inventario, al escoger un medicamento, se mostrará en pantalla toda la información de inventario del medicamento como la fecha de caducidad, la cantidad en Stock, etc



Precio más alto
Al seleccionar esta opción, el usuario será dirigido a un menú donde podrá escoger un proveedor de los proveedores registrados en la farmacia, al escoger un proveedor, se mostrará la información del medicamento más caro que proporciona el proveedor, así como su precio de compra.

Búsqueda del medicamento
Al seleccionar esta opción, el usuario será dirigido a un menú donde se mostrará la lista de inventario en una tabla y los diferentes criterios de filtrado que puede utilizar para realizar la búsqueda, al escoger el criterio de “Proveedor” o “Categoría”, se le proporcionará un cuadro de opciones para escoger qué proveedor o categoría desea filtrar; en el caso de los criterios de “Nombre” y “Principio Activo”, se mostrará un espacio para que el usuario pueda ingresar el medicamento a buscar, y de manera autónoma, se filtrarán los medicamentos según el texto que siga colocando el usuario.



Salir

Al seleccionar esta opción, el usuario saldrá de la aplicación y el programa se cerrará

Modificar Proveedor
Esta opción adicional tiene como función, modificar la información de un proveedor cuando suceda el caso de que se ingresó alguna información errónea o se desea actualizar la información del proveedor.