**1. Creación de la instancia del temporizador**

Private WithEvents Timer1 As New Timer()

Se declara el temporizador Timer1 con el manejador de eventos WithEvents para capturar el intervalo de tiempo cuando el temporizador alcanza el intervalo especificado.

**2. Constructor del Formulario**

Public Sub New()

InitializeComponent()

' Configurar el temporizador

Timer1.Interval = 5000 ' 3600000 ms = 1 hora

Timer1.Start() ' Iniciar el temporizador

End Sub

Se crea el constructor New() donde se inicializan los componentes del formulario mediante la función InitializeComponent() y se configura el temporizador en el tiempo deseado, por defecto esta en 5 segundos ósea 5000 ms.

**3. Evento Timer1\_Tick**

En las siguientes líneas de código, se crea la función Timer1\_Tick que captura el evento del timer1, la cual, se ejecutará cuando el timer1 llegue al tiempo especificado en el constructor.

Posteriormente, se inicializan las variables que permitirán llevar el proceso de verificación de las fechas de vencimiento.

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

Dim targetDate As DateTime

Dim connection As New OleDbConnection(connectionString)

Dim query As String = "SELECT ID, NOMBRE, DESCRIPCION, STOCKACTUAL, FECHA\_CADUCIDAD FROM PRODUCTO"

Dim adapter As New OleDbDataAdapter(query, connection)

Dim dt As New DataTable()

Dim unidad As Integer

**Entre las variables declaradas están:**

* + targetDate: variable para almacenar la fecha de caducidad de cada producto.
  + connection: objeto de tipo OleDbConnection para conectarse a una base de datos mediante una cadena de conexión (connectionString).
  + query: la consulta SQL que selecciona los productos con su fecha de caducidad, stock actual, etc.
  + adapter: un adaptador de datos que ejecutará la consulta SQL y llenará los datos en un DataTable (dt).
  + unidad: variable para almacenar el stock actual del producto.

**4. Conexión y llenado de datos**

Try

connection.Open()

adapter.Fill(dt)

Se intenta abrir la conexión a la base de datos usando la función connection.Open() y se ejecuta la consulta SQL mediante el adapter.Fill(dt), llenando el DataTable (dt) con los resultados de la consulta.

**5. Recorriendo los productos**

For Each row As DataRow In dt.Rows

targetDate = row("Fecha\_caducidad")

unidad = row("STOCKACTUAL")

Se recorre cada fila del DataTable (que representa un producto). Para cada fila, se obtienen los valores de las columnas Fecha\_caducidad y STOCKACTUAL.

**6. Verificación de la fecha de caducidad**

If DateTime.Now >= targetDate And unidad > 0 Then

MessageBox.Show("El producto: " & row("NOMBRE") & " tiene fecha de caducidad " & row("FECHA\_CADUCIDAD") & " para " & row("STOCKACTUAL") & " unidades.")

Timer1.Stop() ' Detenemos el temporizador después de la verificación

Else

' Si aún no ha llegado la fecha, mostramos la fecha actual

Console.WriteLine("Fecha actual: " & DateTime.Now)

End If

Se verifica si la fecha actual (DateTime.Now) es mayor o igual a la fecha de caducidad del producto (targetDate) **y** si el stock actual del producto es mayor que cero (unidad > 0). Si ambas condiciones son verdaderas, se muestra un mensaje con la información del producto (nombre, fecha de caducidad y stock). Después de mostrar el mensaje, el temporizador se detiene con Timer1.Stop(). Si la fecha de caducidad aún no ha llegado, simplemente se imprime la fecha actual en la consola.

**7. Manejo de excepciones**

Catch ex As Exception

MessageBox.Show("Error al conectar: " & ex.Message)

Finally

connection.Close()

End Try

Si ocurre un error durante la conexión a la base de datos o al ejecutar la consulta, se captura la excepción y se muestra un mensaje de error con la función MessageBox.Show(). Finalmente, en el bloque Finally, se asegura de cerrar la conexión a la base de datos con connection.Close(), independientemente de si ocurrió un error o no.