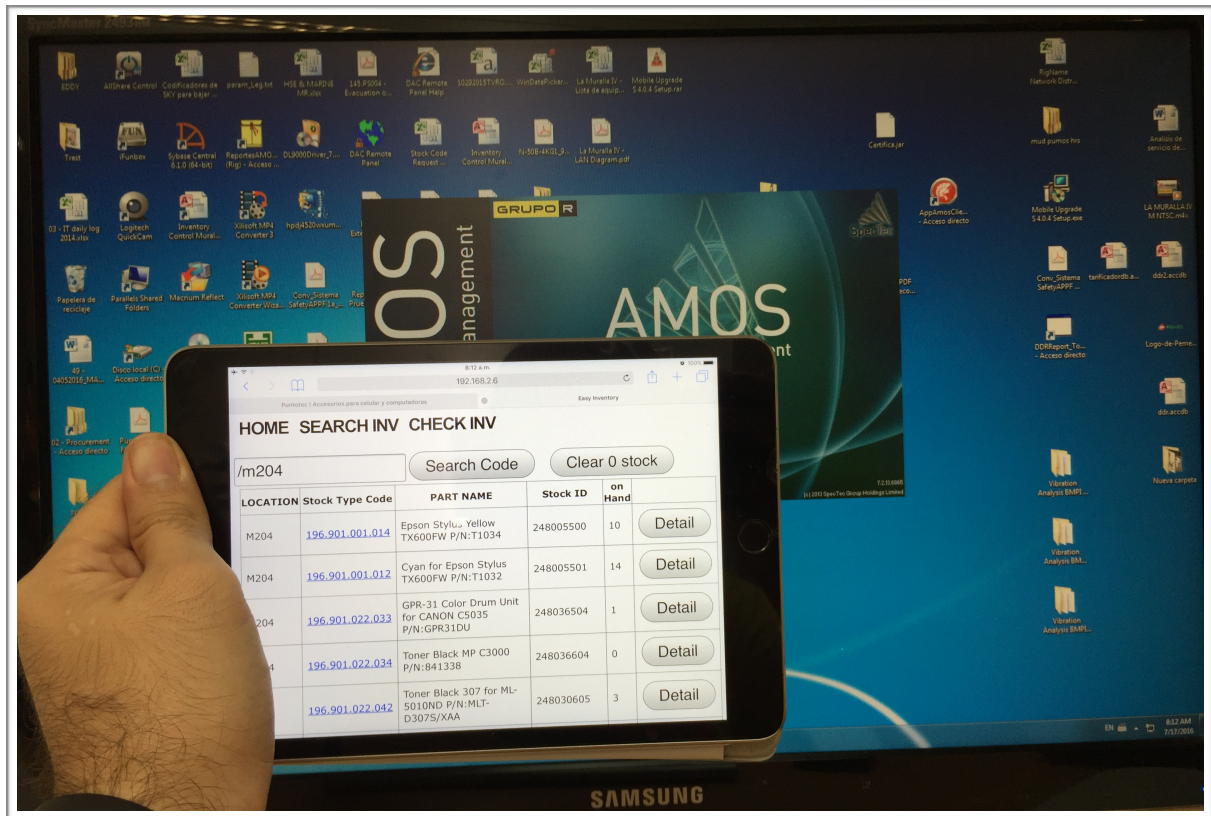


AMOS Easyinv



Manual de Desarrollo

Marco Cantu

Jul 2016

Introducción

Desarrollo y Estructura del programa

Desarrollo

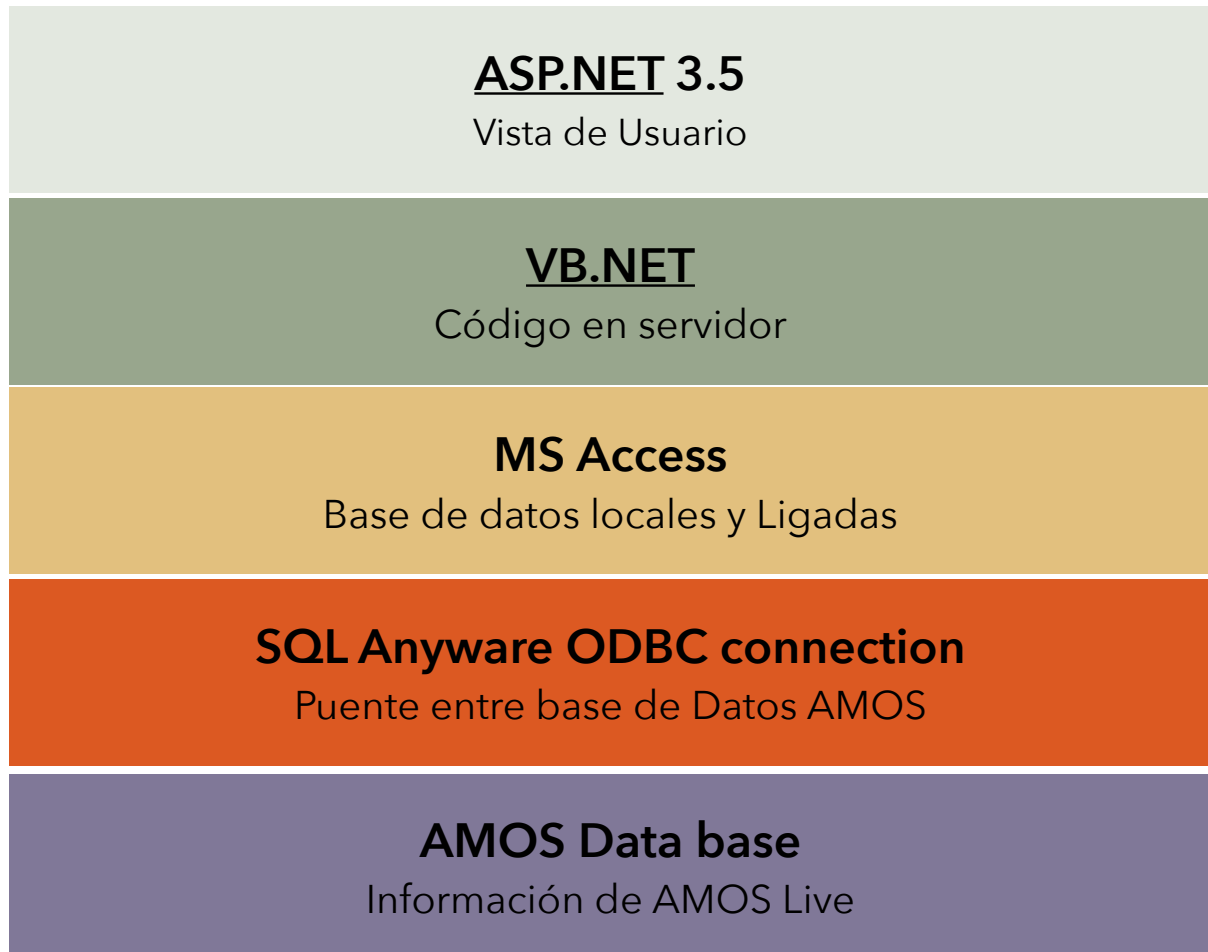
En este manual veremos como esta desarrollado el sistema Easyinv el cual consta de una estructura y un ambiente para su funcionamiento. en las siguiente paginas veremos desde como se estructura, ambiente para su funcionamiento, librerías y programas necesarios.

Indice

Desarrollo y Estructura	4
Requerimientos de ambiente de Desarrollo	5
Estructura del Programa	6
Estructura de librería EasyinvCore.dll	8
Flujo de Datos en Clases	9
Estructura de la Base de datos	12
VISTAS	13
Estructura de Objetos	14
Conectividad a la Base de Datos	18
Auto definición de Base de Datos	19
Consulta de Información	
ADO - Acceso a Datos por Objetos	21
Usando nuestra clase QueryBuilder	22
Obteniendo Consultas de la base de datos	23
Llenando un Gridview con resultados.	24

Desarrollo y Estructura

Estructura de Desarrollo



La estructura del desarrollo esta basada en paginas ASP.NET y con código por parte del servidor Visual basic.Net , este mismo consulta las base de datos que están en Microsoft Access, estas mismas necesitan una conexión ODBC a traves de los controladores de SQL Anyware para la conexión de la base de datos de AMOS.

Requerimientos de ambiente de Desarrollo

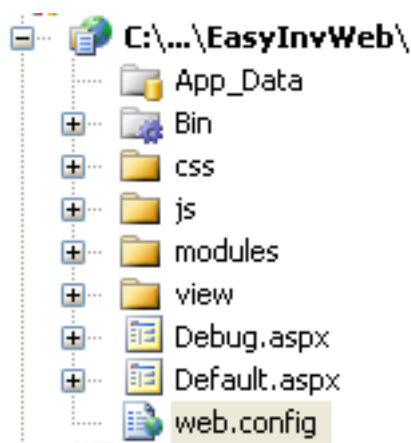
- Visual Studio 2008 o superior
- Framework .NET 3.0 o superior
- MS Access Database Engine (para 64bits o 32 bits)
- SQL Anyware ODBC Drivers (para 64bits o 32 bits)
- Servicio de Base de Datos AMOS a travez de ODBC
- Servidor WEB IIS 6.0 o superior con Framework 3.0 o superior
- Librería de jQuery 3

Fuentes en :

<https://github.com/marcocantugea/EasyinvGR>

Estructura del Programa

El ambiente web utiliza una estructura estándar MVC, el cual esta modulado en la siguiente Manera:



1.- css

Aquí se encuentra los archivos CSS que constan del diseño de la aplicación

2.- js

Aquí se encuentra las librerías en Javascript de JQuery 3.0

3.- Modules

en esta carpeta esta cada modulo desarrollado dentro de la aplicación. estos requieren de la carpeta view el cual contiene las secciones de encabezado, anexos y pie de las paginas.

4.- View

en esta carpeta encontramos las secciones de Encabezado, Menú y Pie de pagina.

5.- Web.config

En este archivo contiene la configuración global del sistema. en la sección de "appSettings" encontraremos las siguientes instrucciones.

Estas variables son requeridas y estas apuntan a las bases de datos, así también

```
<appSettings>
  <add key="DB-AMOSBRIGE" value="Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Documents and Settings\...
  <add key="DB-EASYINV" value="Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Documents and Settings\...
  <add key="domain" value="http://localhost:2615/EasyInvWeb"/>
</appSettings>
```

como la variable "domain" que es necesaria para re-direccionar fotos y mas archivos relacionados con el sistema.

EN LA CARPETA "BIN" SE ALOJARAN TODAS AQUELLAS LIBRERIAS REQUERIDAS POR LA APLICACION.

LIBRERIAS REQUERIDAS

- EASYINVCORE.DLL

Esta librería contiene toda la estructura interna del programa la cual veremos en el siguiente capítulo.

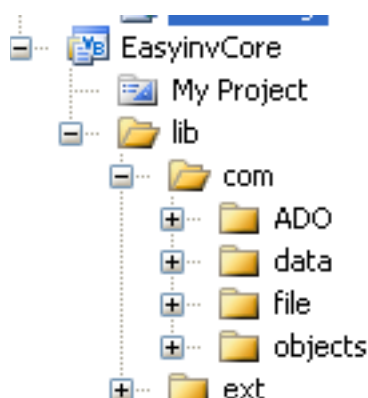
- MESSAGINGTOOLKIT.QRCODE.DLL

Esta librería es necesaria para generar el código de barras QR que necesita la aplicación.

Estructura de librería EasyinvCore.dll

Esta librería contiene todos los métodos y flujos de datos para que la aplicación funcione. Esta desarrollada bajo los estándares de POO (Programación Orientada a Objetos).

A continuación veremos la estructura veremos como se estructura



1.- Lib

Carpeta principal donde se aloja la lógica del programa

2.- com

Estándar para modular las clases dentro de la librería

3. ADO (Access Data Objects)

por su acrónimo en ingles (Access Data Objects) Acceso a Objetos de Datos, aquí tendremos las clases que se dedican a obtener la información de la base de datos y regresar objetos dentro de la aplicación.

en los siguientes capítulos veremos los métodos y funciones que utiliza el sistema para consultar la información.

4. data

En esta carpeta se encuentra el controlador de conexiones de la base de datos. el próximos capítulos veremos como funciona este controlador.

5. file

En esta carpeta se almacenan todas la clases que interactuar en manejar archivos, en los siguientes capítulos veremos funciones donde ocupa el sistema para reducir imagenes y guardar estas en archivos.

6.objects

En esta carpeta tenemos nuestros objetos relacionados a nuestra aplicación, en los siguientes articulo veremos como interactuar con los objetos y como estos se relacionan con las bases de datos.

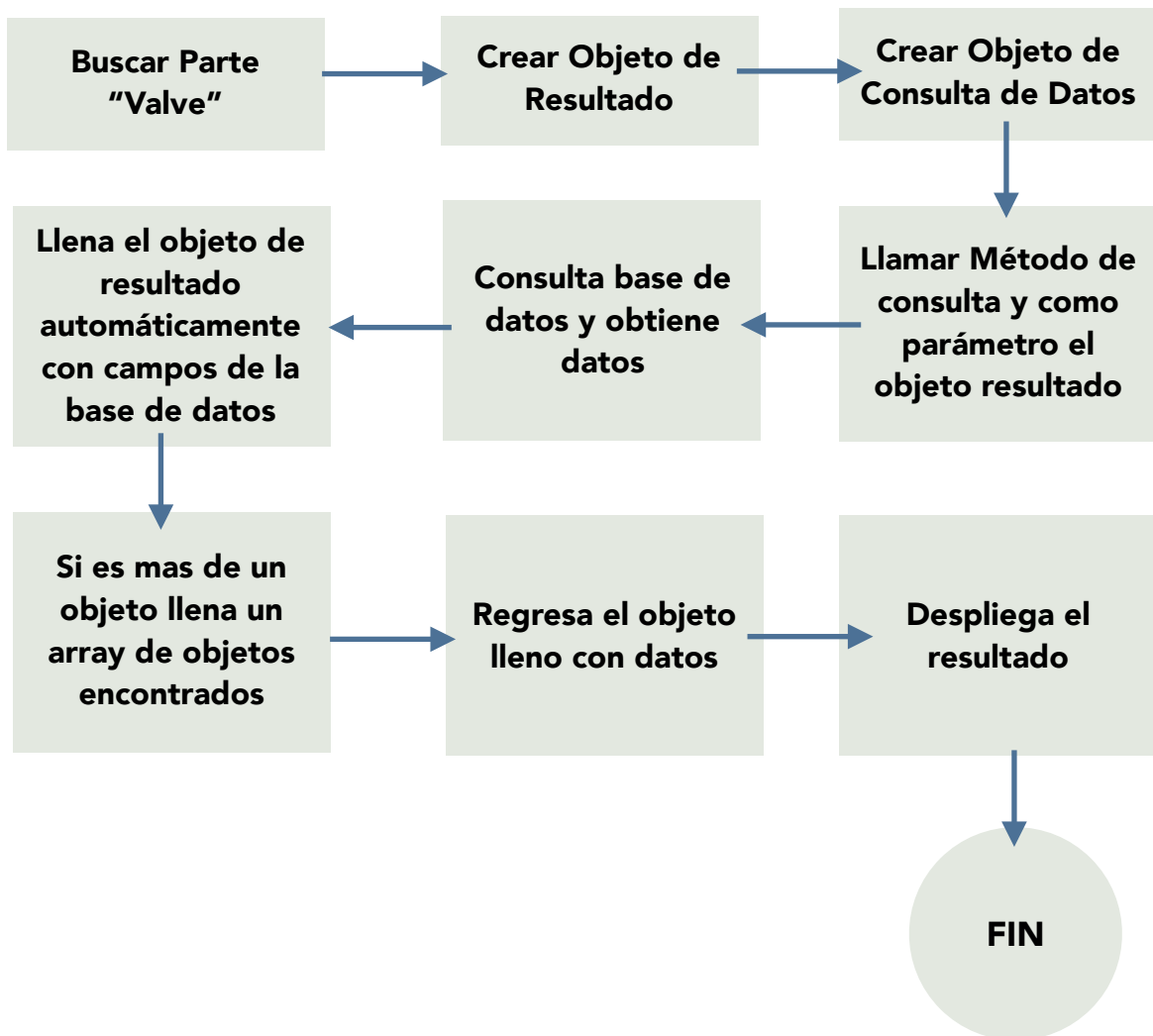
7. ext

En esta carpeta se almacenan las clases externas como Generador deCodigo de Barras QR y otras. estas sirven como puente para llamar a funciones externas de la aplicación.

Flujo de Datos en Clases

Ahora veremos como es el flujo de datos entre la aplicación web y librerías y base de datos.

Para este ejemplo queremos obtener la consulta de cierta parte de la base de datos para esto se realiza el siguiente flujo.



A continuación veremos un ejemplo de como aplicar este concepto.

1. palabra a busca seria "Valve"
2. Creamos el objeto de resultado.

Dim results As New EasyinvCore.com.objects.StockTypeCollection

3. Creamos el objeto de consulta de información

Dim _ADOEasyinv As New EasyinvCore.com.ADO.ADOEasyInv

4. Llamamos El método para buscar la información y como parámetros requeridos necesitamos para ese método la palabra a buscar y el objeto de resultado.

_ADOEasyinv.GetStockInventoryByLocation("Valve",results)

El metodo GetStockInventoryByLocation internamente llena el objeto "results" con objetos tipo "stockinventory" estos los podrás encontrar dentro de la estructura de la librería en la carpeta "objects", Mas adelante hablaremos a detalle de como esta compuestos estos objetos y como esta relacionados con la base de datos.

5. El objeto "results" ahora contiene la información necesaria que consulto en la base de datos. este puede ser usado para desplegar la información.

6. Despliegue de información

Para el despliegue de información, en .Net se utiliza el bucle "for each", para ver cada objeto. en este caso el objeto "result " tiene dentro de el otros objetos que son tipo "stockinventory" estos dentro cuenta con la información de la base de datos.

para consultarlos usamos

```
For each item as Easyinv.lib.com.objects.stockinventory in result.items
```

```
    ' Desplegamos el valor del objeto que contiene result
```

```
    Response.Write(item.STOCKTYPEID)
```

```
    Response.Write(item.PARTNAME)
```

```
Next
```

de esta forma desplegamos en pantalla la información que se obtiene en la base de datos.

Ejemplo completo.

```
' Objeto de resultado de consulta a base de datos
```

```
Dim results As New EasyinvCore.com.objects.StockTypeCollection
```

```
'Objeto de consulta a base de datos
```

```
Dim _ADOEasyinv As New EasyinvCore.com.ADO.ADOEasyInv
```

```
'Metodo de consulta a la base ddatos busca el inventario de una palabra
```

```
_ADOEasyinv.GetStockInventoryByLocation("Valve",results)
```

```
'Despliegue de informacion.
```

```
For each item as Easyinv.lib.com.objects.stockinventory in result.items
```

```
    ' Desplegamos el valor del objeto que contiene result
```

```
    ' Imprime en pantalla el stocktypeid
```

```
    Response.Write(item.STOCKTYPEID)
```

```
    'imprime en pantalla partname
```

```
    Response.Write(item.PARTNAME)
```

```
Next
```

Estructura de la Base de datos

El sistema cuenta con 2 base de datos en Microsoft Access:

amosbridge.accdb







Esta cuenta la conexión mediante ODBC a las tablas ligadas del sistema amos. aquí solo están las tablas ligadas por si en algún momento cambia los datos fuentes.

Esta se utiliza solo para consultas directas a la base de datos de AMOS

easyinv.accdb

Aquí contiene también tablas ligadas a el sistema de AMOS y ademas cuenta con las tablas para almacenar las capturas de inventario y la dirección de las fotos de las partes

las tablas son las siguientes

 inventory_capture	inventory_capture
 stocktypephotos	es la información que se guarda al momento de capturar inventario
 spectwosuite_LOCATION	stocktypephotos
 spectwosuite_STOCKLOCATION	es la tabla donde almacena la relación
 spectwosuite_STOCKTYPE	del archivo físico y la llave primaria del
 spectwosuite_STOCKUNIT	la tabla de AMOS Stoktype

spectwosuite_LOCATION

tabla donde esta la información del catalogo de la localización de almacenes

spectwosuite_STOCKLOCATION

tabla donde dice la cantidad de partes y la relación de la tabla locations

spectwosuite_STOCKTYPE

tabla donde se almacena la información de las partes

spectwosuite_STOCKUNIT

tabla donde esta la relación entre la unidad actual y las partes

VISTAS

Queries



q_inventoryinputs

q_inventoryinputs

Obtiene la locación de almacén y la fecha en que se capturo el reporte



q_maincatalogparts

q_maincatalogparts

Obtiene el catalogo de partes



q_onhand_stocktype

q_onhand_stocktype

Obtiene la cantidad de inventario que existe

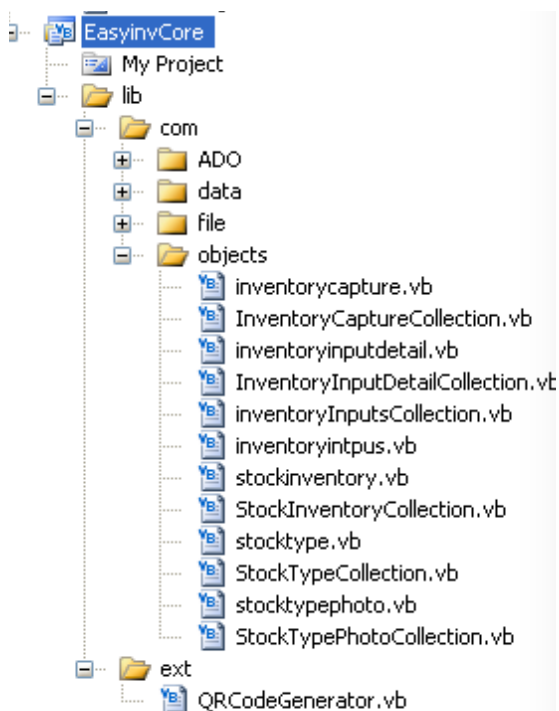


q_stockwarehose

q_stockwarehose

Obtiene el detalle del lo que existe en almacén.

Estructura de Objetos



Los objetos son parte esencial del flujo de la información ya que con ellos solo se puede acceder a la información de la base de datos y asegurar que la información no sea directamente modificada en la base de datos sin haber sido procesada.

Esta se divide en objetos únicos y en conjuntos de objetos almacenados en colecciones.

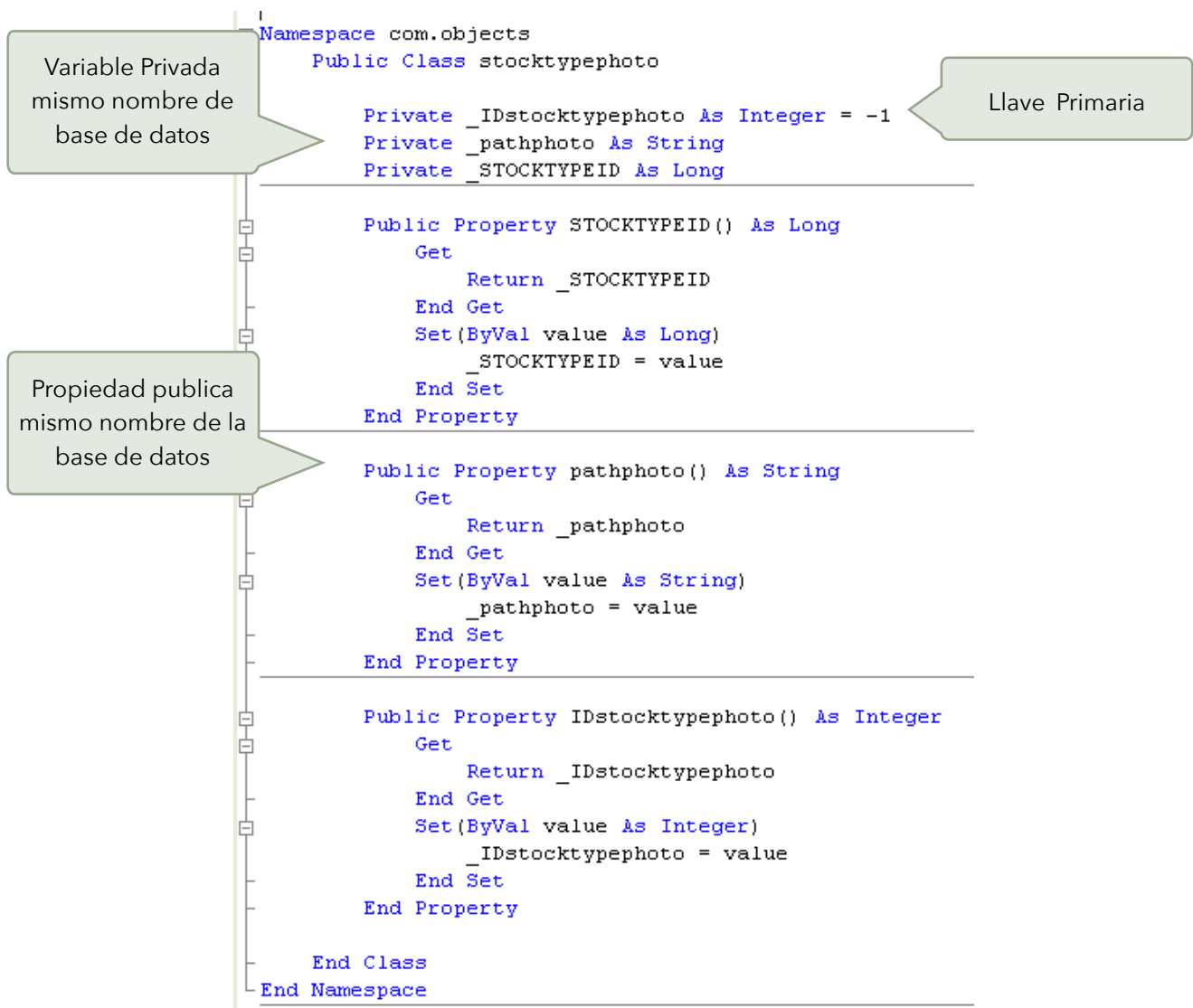
los objetos únicos son aquellos que **no contiene la Palabra "Collection"** y los que contienen la palabra mencionada

anteriormente son objetos que obtienen conjunto de objetos de los que se relacionan al momento de recolectar estos mismos.

Todos los objetos deben de contener la siguiente estructura.

1. Variable privada para el almacenamiento en memoria del valor
2. Declarar la variable y propiedad con el mismo nombre que en la base de datos
3. Las llaves primarias se deben de declarar con el valor -1

a continuación veremos un ejemplo del objeto Stocktypephoto.vb



la definición del objeto es con la finalidad que la clase ADO lea las propiedades y las relacione con la base de datos automáticamente.

la llave primaria se define "-1" para que cuando se realice alguna inserción de datos o actualización el objeto identifique que este campo no debe de ser llenado o actualizado.

Los objetos de Colección son aquellos que almacenan objetos para su consulta, en caso de que deseamos hacer alguna operación que esta misma sea calculada por el programa no por el servicio de la base de datos.

Estas clases implementan las interfaces de IEnumerable, IEnumerator y ICollection esto con el fin de cualquier componente de ".NET" procese la colección de datos sin tener que hacer componentes DATASET o objetos ARRAY mas adelante veremos un ejemplo.

Dentro de la estructura de una colección de datos vemos el ejemplo de la siguiente clase.

```
Namespace com.objects
Public Class StockTypePhotoCollection
    Implements IEnumerable, IEnumerator, ICollection

    Private position As Integer = -1
    Private _Items As New List(Of stocktypephoto)

Public ReadOnly Property Items() As List(Of stocktypephoto)
    Get
        Return _Items
    End Get
End Property

Public ReadOnly Property CurrentPosition() As Integer
    Get
        Return position
    End Get
End Property

Public Sub Add(ByVal Item As stocktypephoto)
    If Not IsNothing(Item) Then
        _Items.Add(Item)
    End If
End Sub

Public Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements ICollection.CopyTo
End Sub

Public ReadOnly Property Count() As Integer Implements ICollection.Count
    Get
        Return _Items.Count
    End Get
End Property

Public ReadOnly Property IsSynchronized() As Boolean Implements ICollection.IsSynchronized
    Get
        Return False
    End Get
End Property
```


En la imagen se nota la variable “_Items” el cual almacenara todos los objetos recolectados tipo “stocktypephoto” esta no puede ser consultada directamente, para esto se declaro la propiedad “ITEMS()” con la cual puede ser solo de lectura.

Para agregar nuevos objetos a nuestra colección de datos usamos el método declarado “ADD()” el cual requiere un objeto “stocktypephoto” para agregar a la colección de datos.

a continuación veremos como llenar esta colección

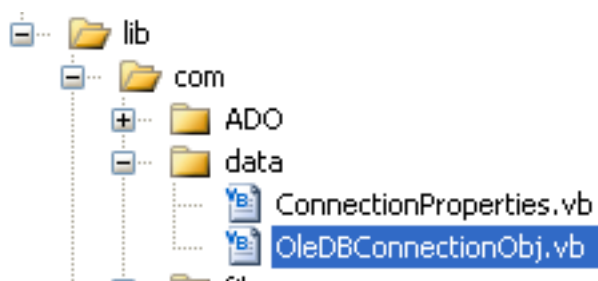
```
Public Sub LlenarCollection()  
    'Declaramos nuestro objeto de stocktypephoto  
    Dim newobj As New EasyinvCore.com.objects.stocktypephoto  
    'asignamos valores al objeto  
    newobj.STOCKTYPEID = 240938938  
    newobj.pathphoto = "esta es una prueba"  
  
    'creamos nuestra coleccion de datos  
    Dim newcollection As New EasyinvCore.com.objects.StockTypePhotoCollection  
    'agregamos el objeto a la coleccion  
    newcollection.Add(newobj)  
  
End Sub
```

De la forma anterior agregamos objetos a nuestra colección de datos. Ahora veremos como leer nuestra colección, para esto usaremos un bucle “For each” el cual nos ayuda a ver cada objeto dentro de nuestra colección

```
'Leer coleccion de datos  
For Each objeto As EasyinvCore.com.objects.stocktypephoto In newcollection.Items  
    'imprime en pantalla el valor de stocktypeid  
    Console.WriteLine(objeto.STOCKTYPEID.ToString)  
Next
```

Conectividad a la Base de Datos

El programa usa las siguientes clases para la consulta a la base de datos.



La clase "OleDbConnectionObj" contiene las propiedades y los métodos para la conectividad a la base de datos.

Para definir una base de datos requiere un objeto "ConnectionProperties" esta clase contiene los detalles de la conectividad de la base de datos. como continuación se muestra.

```
Namespace com.data
    Public Class ConnectionsProperty
        Dim _Name As String
        Dim _ConnectionString As String
        Dim _Connection As OleDbConnection
        Dim _Adap As OleDbDataAdapter
        Dim _Command As OleDbCommand

        Public Property Name() As String
            Get
                Return _Name
            End Get
            Set(ByVal value As String)
                _Name = value
            End Set
        End Property

        Public Property ConnectionString() As String
            Get
                Return _ConnectionString
            End Get
            Set(ByVal value As String)
                _ConnectionString = value
                Try
                    SetupConnection()
                Catch ex As Exception
                End Try
            End Set
        End Property

        Public ReadOnly Property Connection() As OleDbConnection
            Get
                Return _Connection
            End Get
        End Property
    End Class
End Namespace
```

en esta se almacena el nombre de la base de datos, cadena de conexión , el objeto de conexión, el adaptador a la base de datos y también se utilizara el objeto "Command" para hacer consultas a la base de datos.

Para crear una conexión a la base de datos se deberá heredar esa clase para poder usar los métodos de conexión

aquí veremos como se utiliza.

```
Public Class TestDB
    Inherits EasyinvCore.com.data.OleDbConnectionObj

    Public Sub createnewConnection()
        CreateConnection("DATABASE1", "Provider=Microsoft.ACE.OLEDB.12.0;Data So
        OpenDB("DATABASE1")
        CloseDB()
    End Sub

End Class
```

al momento de crear la clase vemos la parte de "Inherits" la cual hereda a la clase de OleDbConnectionObj y esta puede usar sus métodos de crear conexiones y abrir la base de datos.

Auto definición de Base de Datos

la clase de "OleDbConnectionObj" contiene un constructor el cual se encarga de que si existe un archivo de configuración "Web.config" o "App.config" este buscara en la sección de "appSettings" dentro del XML y buscara todas las llaves que empiecen con "DB-" y estas las tomara como configuración de la base de datos.

A continuación veremos un ejemplo de como se auto definen.

En nuestra aplicación web contamos con el archivo de WEB.CONFIG el cual contiene la siguiente info.



Veremos que en la sección de "appSettings" contamos con la definición de llaves "DB-AMOSBRIGE" y "DB-EASYINV". estas llaves al momento de usar nuestro objeto las tomara y las configurara como base de datos activas.

A continuacion veremos como realiza esto.

```

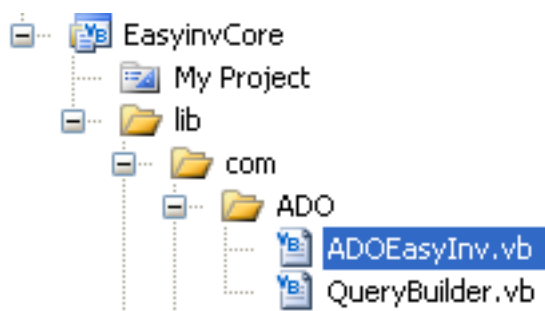
Namespace com.data
Public Class OleDbConnectionObj
Protected connections As New Collection
Protected connection As com.data.ConnectionsProperty

Public Sub New()
Dim cont As Boolean = False
For Each s As String In System.Configuration.ConfigurationSettings.AppSettings
If s.Contains("DB-") Then
cont = True
Dim con As New com.data.ConnectionsProperty
con.Name = s
con.ConnectionString = System.Configuration.ConfigurationSettings.AppSettings(s)
connections.Add(con, con.Name)
End If
Next
If Not cont Then
Throw New Exception("There no any Database configure please configure at least 1 database.")
End If
End Sub

```

Consulta de Información

ADO - Acceso a Datos por Objetos



Dentro de la aplicación contamos con la clase "ADOEasyInv" que esta contiene los métodos para consultar la base de datos y regresar objetos contruidos con la información solicitada.

En esta clase encontraremos las consultas a la base de datos, la inserción de datos , actualización de datos y eliminación de datos a nuestras tablas de la base de datos.

Veremos primero como funciona nuestra clase ADO. primero como ya explicamos requerimos heredar la clase "OleDbConnectionObj" para poder conectarnos a la base de datos.

Primero veremos como hacer una inserción de datos a la base de datos. para esto nos ayudara una clase llamada "QueryBuilder" que básicamente hace el puente entre nuestros Objeto y la base de datos. Por eso era importante que nuestros objetos tengan la misma nomenclatura que nuestra base de datos.

a continuacion veremos como hacer una inserción a nuestra base de datos.

Para este ejemplo haremos el ejemplo completo de como agregar un registro a la base de datos con el objeto de "stocktypephoto"

Usando nuestra clase QueryBuilder

esta clase es útil ya que nos ayuda a crear instrucciones SQL para inserción de datos, actualización de datos o borrar datos en base a objetos definidos dentro de nuestro sistema.

un ejemplo queremos crear una instrucción SQL para agregar a nuestra base de datos información tipo “stocktypephoto”

```
Public Function CreateQuery() As String
    ' creamos y llenamos nuestro objeto de stocktypephoto
    Dim newobj As New EasyinvCore.com.objects.stocktypephoto
    newobj.STOCKTYPEID = 22342342
    newobj.pathphoto = "photos/tmp/photo.jpg"

    ' creamos nuestro querybuilder
    ' al momento de crear nuestro objeto debemos decirle que es tipo stocktypephoto
    Dim querybuilder As New EasyinvCore.com.ADO.QueryBuilder(Of EasyinvCore.com.objects.stocktypephoto)
    ' tenemos que definir que instruccion vamos a crear
    querybuilder.TypeQuery = EasyinvCore.com.ADO.TypeQuery.Insert
    ' ahora debemos darle la informacion que tomara para hacer nuestro query
    querybuilder.Entity = newobj
    ' ahora tenemos que generar nuestra instruccion y tenemos que decirle que tabla de nuestra base
    ' de datos usara
    querybuilder.BuildInsert("stocktypephotos")

    ' regreamos nuestro query construido
    Return querybuilder.Query
End Function
```

En nuestra clase ADO veremos ya completo como se genera la instrucción y como se conecta a la base de datos y como ejecuta el comando.

```
Namespace com.ADO
    Public Class ADOEasyInv
        Inherits com.data.OleDbConnectionObj
    End Class
End Namespace

Namespace com.ADO
    Public Class ADOEasyInv
        Inherits com.data.OleDbConnectionObj

        Public Sub AddPhotoToStockType(ByVal photo As stocktypephoto)
            Dim qbuilder As New QueryBuilder(Of stocktypephoto)
            qbuilder.TypeQuery = TypeQuery.Insert
            qbuilder.Entity = photo
            qbuilder.BuildInsert("stocktypephotos")
            Try
                OpenDB("DB-EASYINV")
                connection.Command = New OleDb.OleDbCommand(qbuilder.Query, connection.Connection)
                connection.Command.ExecuteNonQuery()
            Catch ex As Exception
                Throw
            Finally
                CloseDB()
            End Try
        End Sub
    End Class
End Namespace
```

Obteniendo Consultas de la base de datos

Para las consultas de la base de datos (instrucción SQL "SELECT") usamos el método común, en este caso la aplicación llena la colección de objetos y regresa un objeto lleno de objetos.

A continuación veremos un ejemplo de como automáticamente se llena una colección de objetos, con las siguientes instrucciones automáticamente crea y llena los objetos en una colección

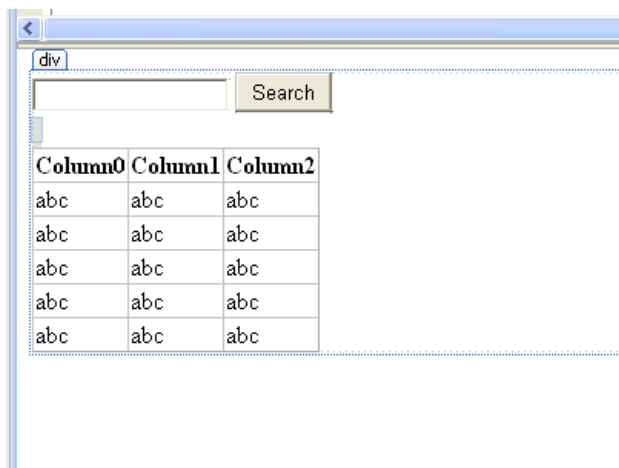
```
Public Sub GetPhotosStock(ByVal stocktypeid As Long, ByVal result As StockTypePhotoCollection)
    Try
        OpenDB("DB-EASYINV")
        Dim query As String
        query = "select * from stocktypephotos where stocktypeid=" & stocktypeid.ToString & " "
        connection.Command = New OleDb.OleDbCommand(query, connection.Connection)
        connection.Adap = New OleDb.OleDbDataAdapter(connection.Command)
        Dim dts As New DataSet
        connection.Adap.Fill(dts)
        If dts.Tables.Count > 0 Then
            If dts.Tables(0).Rows.Count > 0 Then
                For Each row As DataRow In dts.Tables(0).Rows
                    Dim photo As New stocktypephoto
                    For Each member In photo.GetType.GetProperties
                        If member.CanWrite Then
                            If member.PropertyType.Name = "String" Or member.PropertyType.Name = "Int32" Or
                                If Not IsDBNull(row(member.Name)) Then
                                    If member.PropertyType.Name = "String" Then
                                        member.SetValue(photo, row(member.Name).ToString, Nothing)
                                    End If
                                    If member.PropertyType.Name = "Int32" Then
                                        member.SetValue(photo, Integer.Parse(row(member.Name)), Nothing)
                                    End If
                                    If member.PropertyType.Name = "Int64" Then
                                        member.SetValue(photo, Long.Parse(row(member.Name)), Nothing)
                                    End If
                                    If member.PropertyType.Name = "DateTime" Then
                                        member.SetValue(photo, row(member.Name), Nothing)
                                    End If
                                End If
                            End If
                        End If
                    Next
                    result.Add(photo)
                Next
            End If
        End If
    Catch ex As Exception
        Throw
    Finally
        CloseDB()
    End Try
End Sub
```

lo que esta de azul , crea un objeto, y este mismo obtiene las propiedades y va asociando la propiedad del objeto con la base de dato y lo va llenando.

Llenando un Gridview con resultados.

A continuacion veremos un ejemplo de como llenar un componente "GridView" con nuestras colecciones y usando los métodos de ADO.

a nuestra pagina agregamos un nuevo gridview



a nuestro código en el evento de cargar la pagina agregamos la siguiente instrucción.

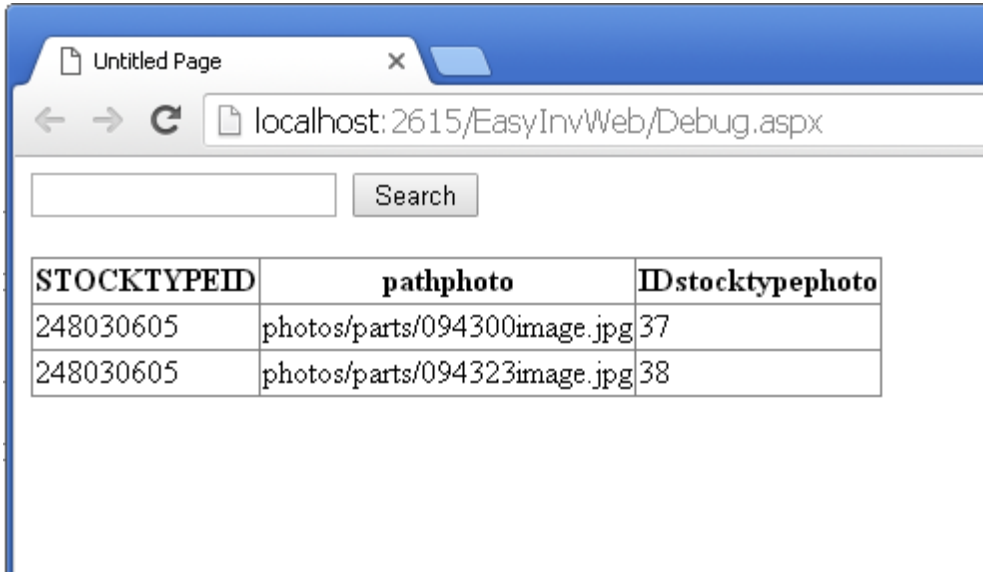
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    'declaramos nuestra variable de resultado la cual va a obtener
    'objetos de tipo stocktypephoto
    Dim resultado As New EasyinvCore.com.objects.StockTypePhotoCollection
    'declaramos nuestro objeto ADO
    Dim _ADOEasyinv As New EasyinvCore.com.ADO.ADOEasyInv

    'buscamos las fotos del stockunitid = 248030605
    _ADOEasyinv.GetPhotosStock(248030605, resultado)

    'asociamos los datos del resultado con el gridview
    'debemos regresar los objetos con la propiedad items
    GridView1.DataSource = resultado.Items
    'se imprime el gridview
    GridView1.DataBind()

End Sub
```


Nuestro Resultado sera :



The screenshot shows a web browser window with a single tab titled 'Untitled Page'. The address bar displays 'localhost:2615/EasyInvWeb/Debug.aspx'. Below the address bar is a search bar with a 'Search' button. The main content area contains a table with three columns: 'STOCKTYPEID', 'pathphoto', and 'IDstocktypephoto'. The table has two data rows.

STOCKTYPEID	pathphoto	IDstocktypephoto
248030605	photos/parts/094300image.jpg	37
248030605	photos/parts/094323image.jpg	38