

MANUAL DE USUARIO BANCO DE APLICACIONES BASICAS PARA EL ESTUDIO DE LA PROGRAMACION ORIENTADA A OBJETOS BAJO ARQUITECTURA DE CAPAS

FECHA:

13/11/2015.

1. JUSTIFICACIÓN

Con el objetivo de garantizar el correcto funcionamiento de las aplicaciones que forman parte del banco de aplicaciones básicas para el estudio de POO bajo arquitectura de capas. Asegurando el cumplimiento de las tareas y funciones que deben desarrollar las mismas. Se detalla en el presente informe el modo de uso de las aplicaciones que hacen parte del banco de aplicaciones.

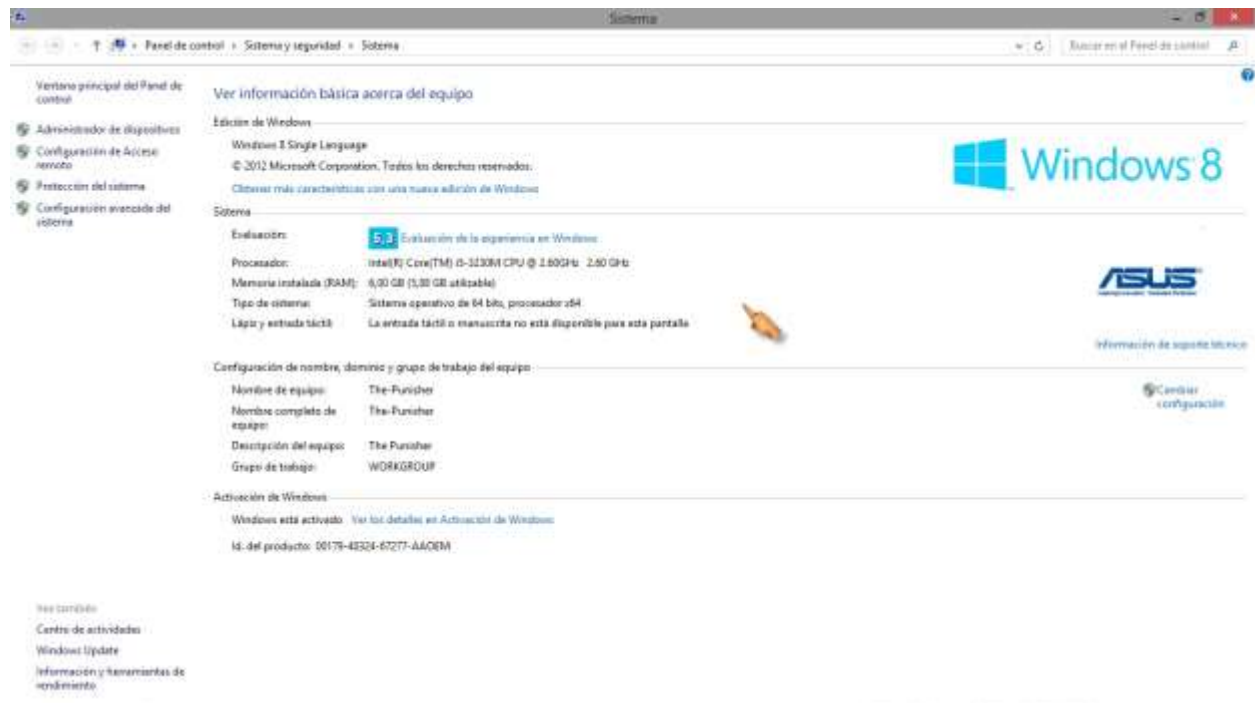
ESPECIFICACIONES TÉCNICAS RECOMENDADAS

Detalles del Sistema

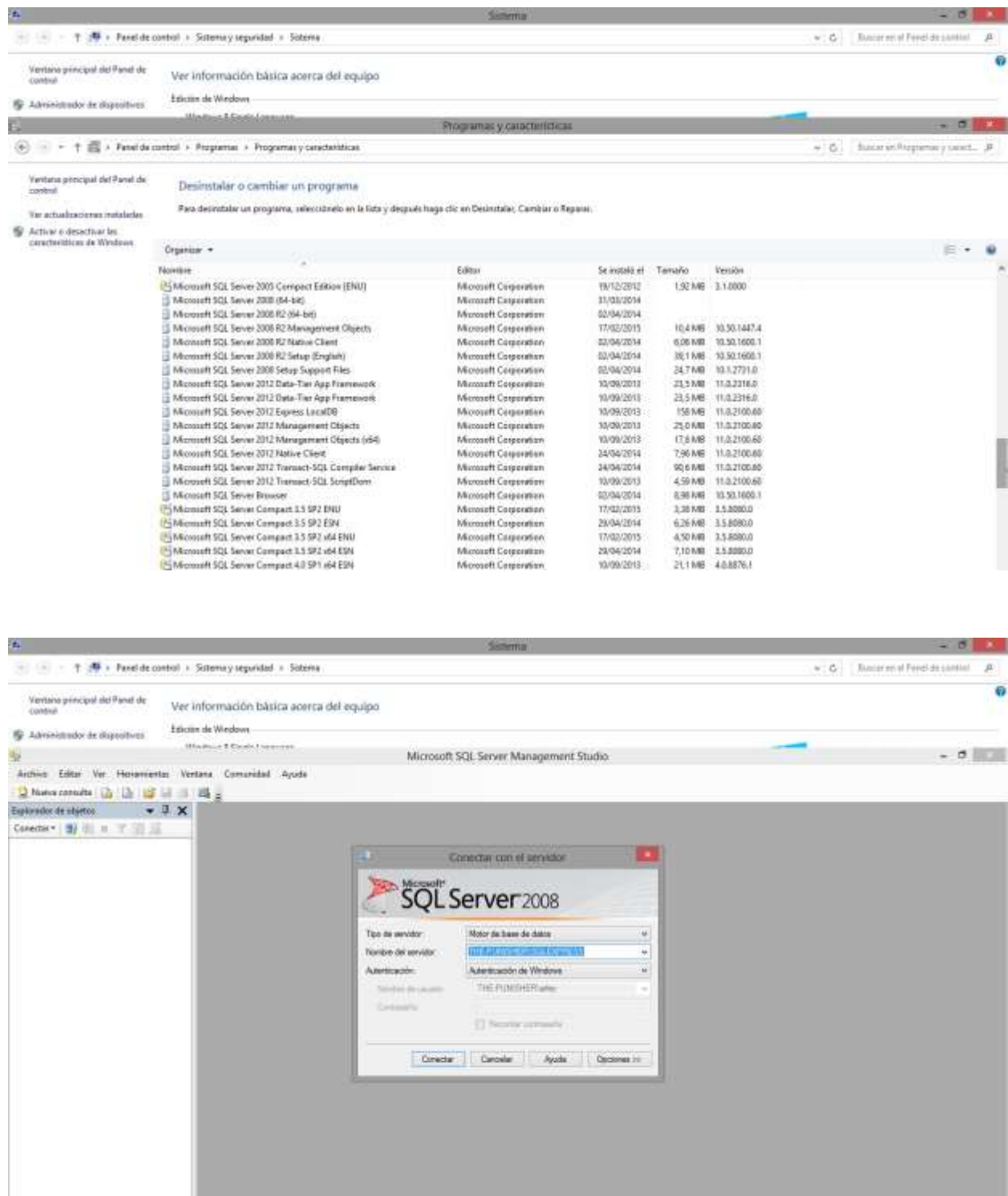
Sistema Operativo	Familia de sistemas Operativos Windows, Windows 7, Windows 8
Tipo de Sistema	Sistemas Operativo de 64 y 32 bits
Gestores de Base de Datos	Microsoft SQL Server 2008 r2, MYSQL
Servidor local	Wamp Server
Microsoft .Net Framework	Version 4 Multi-Targeting Pack
IDES de Desarrollo	Netbeans 7.4, Microsoft Visual Studio 2012

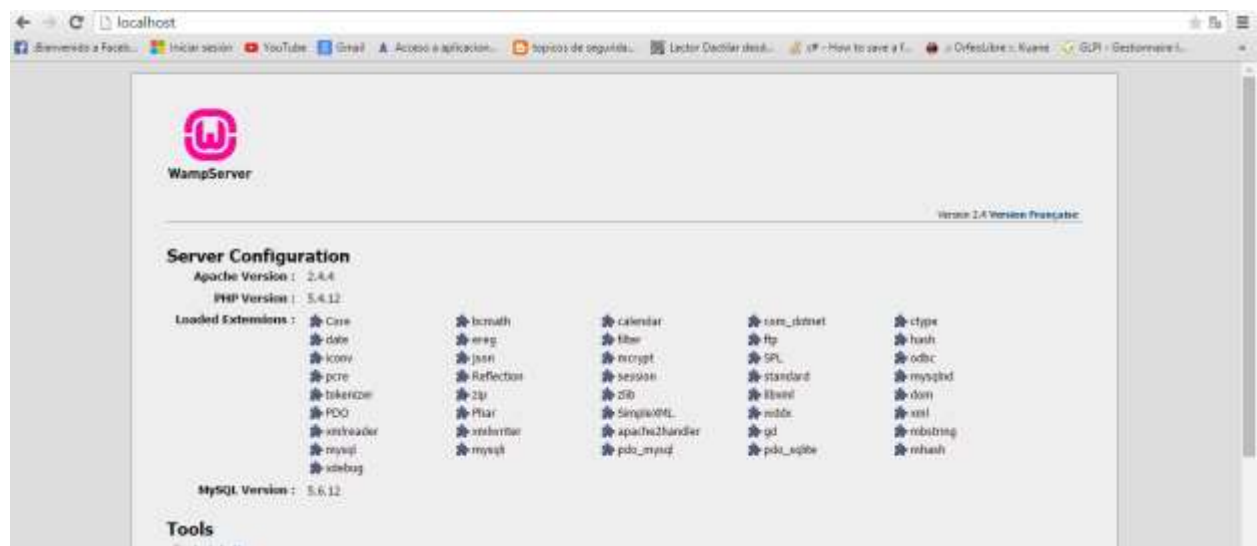
A continuación se detalla en imagenes la configuración recomendada

Sistema Operativo:

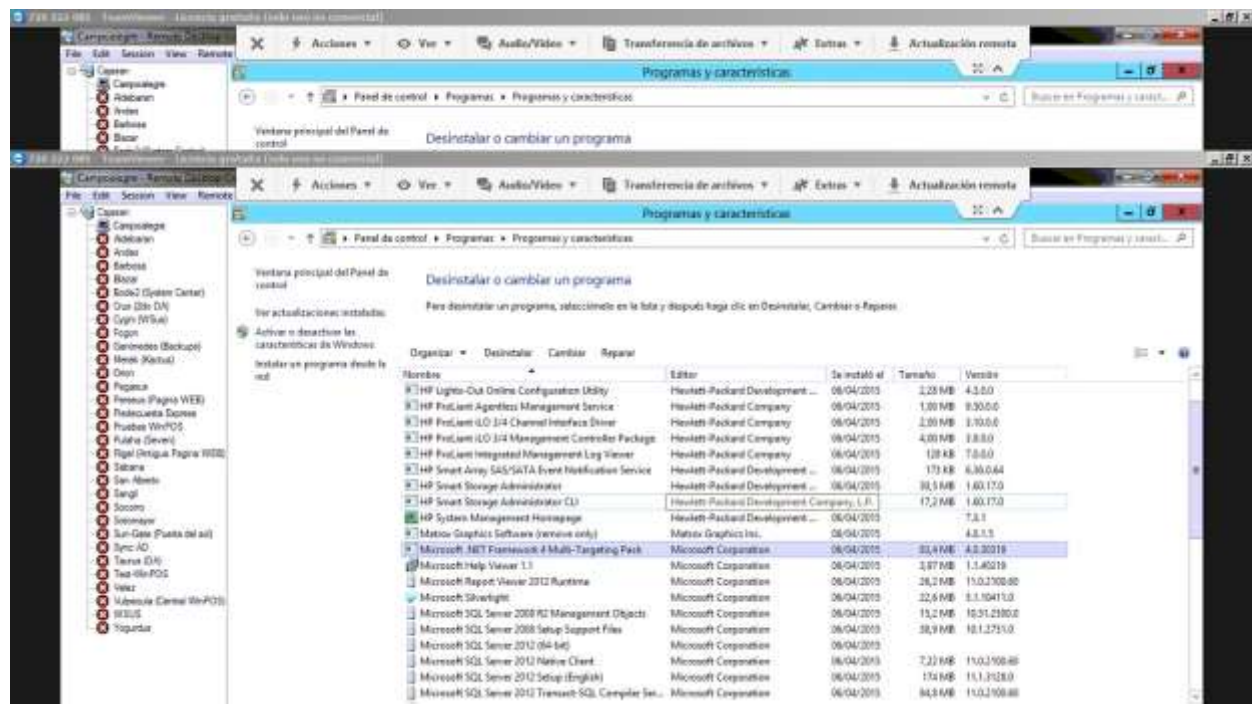


Gestores de Base de Datos

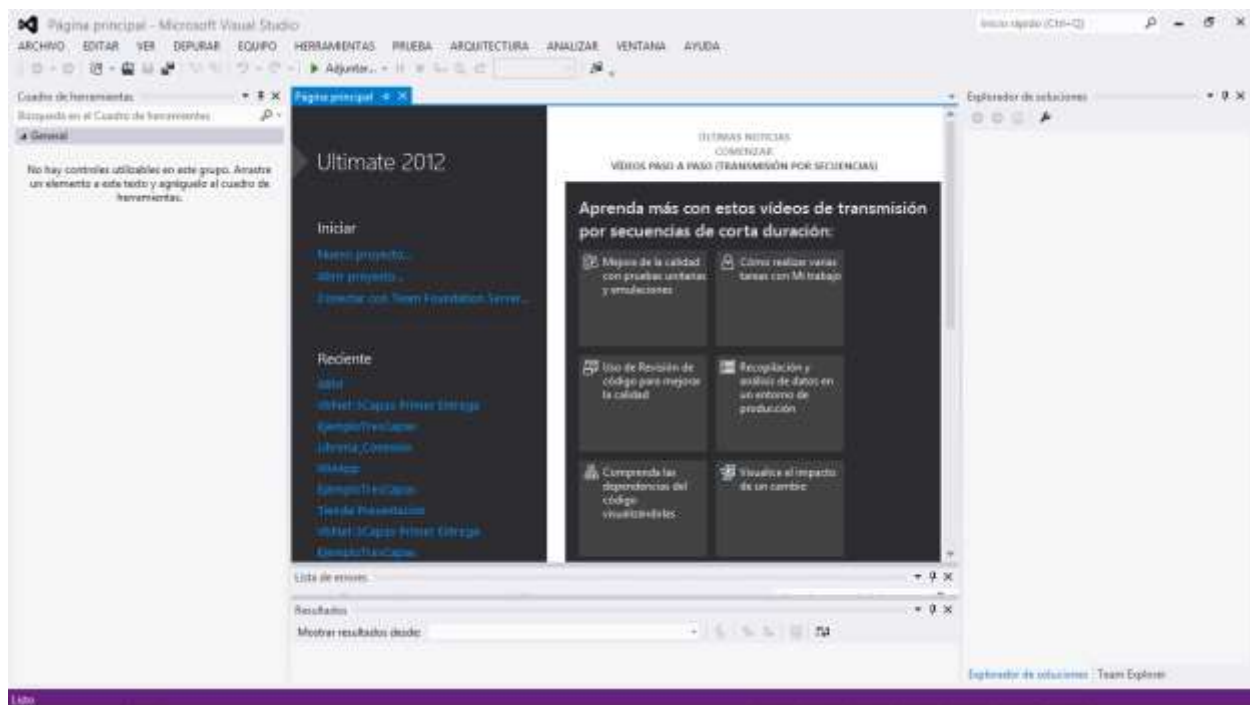
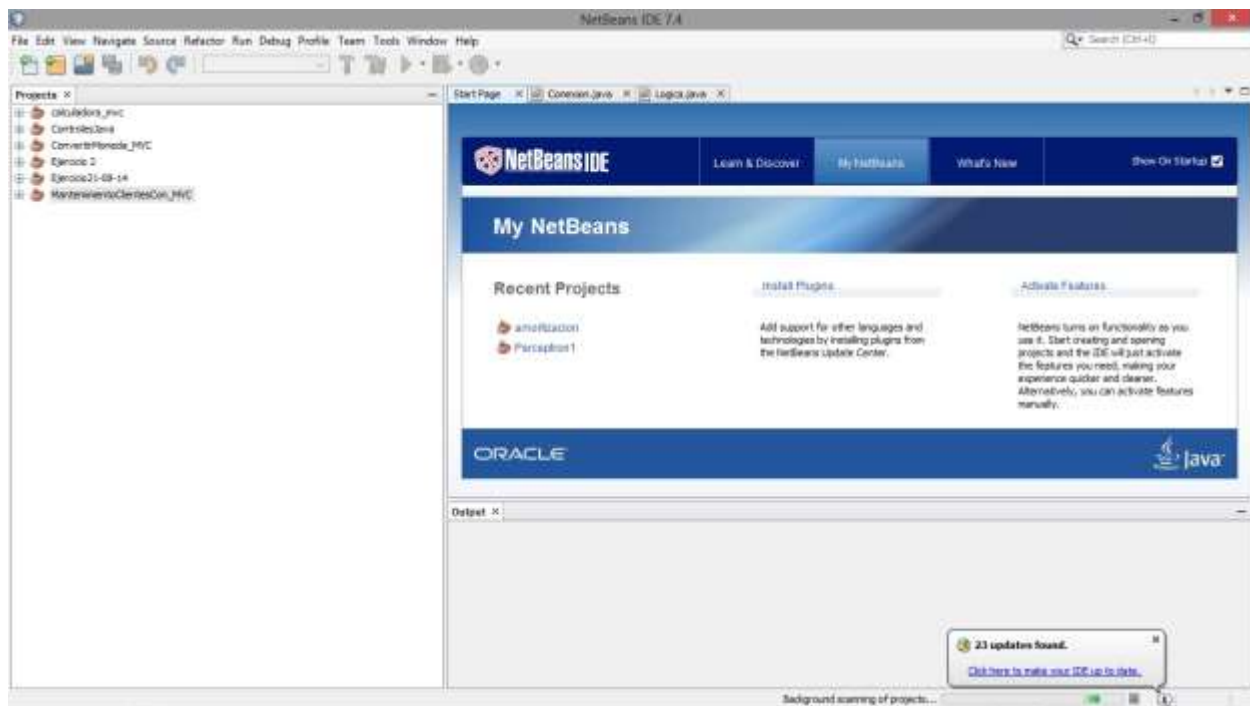




Microsoft .Net Framework



IDES de Desarrollo



Observaciones

Grupo 1

Las siguientes aplicaciones corren bajo el IDE de desarrollo Netbeans 7.4, y la conexión al gestor de Base de datos se realiza con el Gestor MYSQL.

- Calculadora Bajo POO y Arquitectura de Capas
- Convertidor Moneda Bajo POO y Arquitectura de Capas
- Operaciones básicas bajo POO y Arquitectura de Capas
- Operaciones Matemáticas bajo POO y Arquitectura de Capas
- Mantenimiento Clientes con conexión a Base de Datos bajo POO y Arquitectura de Capas

Grupo 2

Las siguientes aplicaciones corren bajo el IDE de desarrollo Microsoft Visual Studio 2012, y la conexión al gestor de Base de datos se realiza con el Gestor MYSQL.

- Aplicación para listar Bases de Datos de un Servidor, junto con sus Tablas y Registros bajo POO y Arquitectura de capas
- Aplicación con acceso a Datos, información países, ciudades y Provincias bajo POO y Arquitectura de Capas

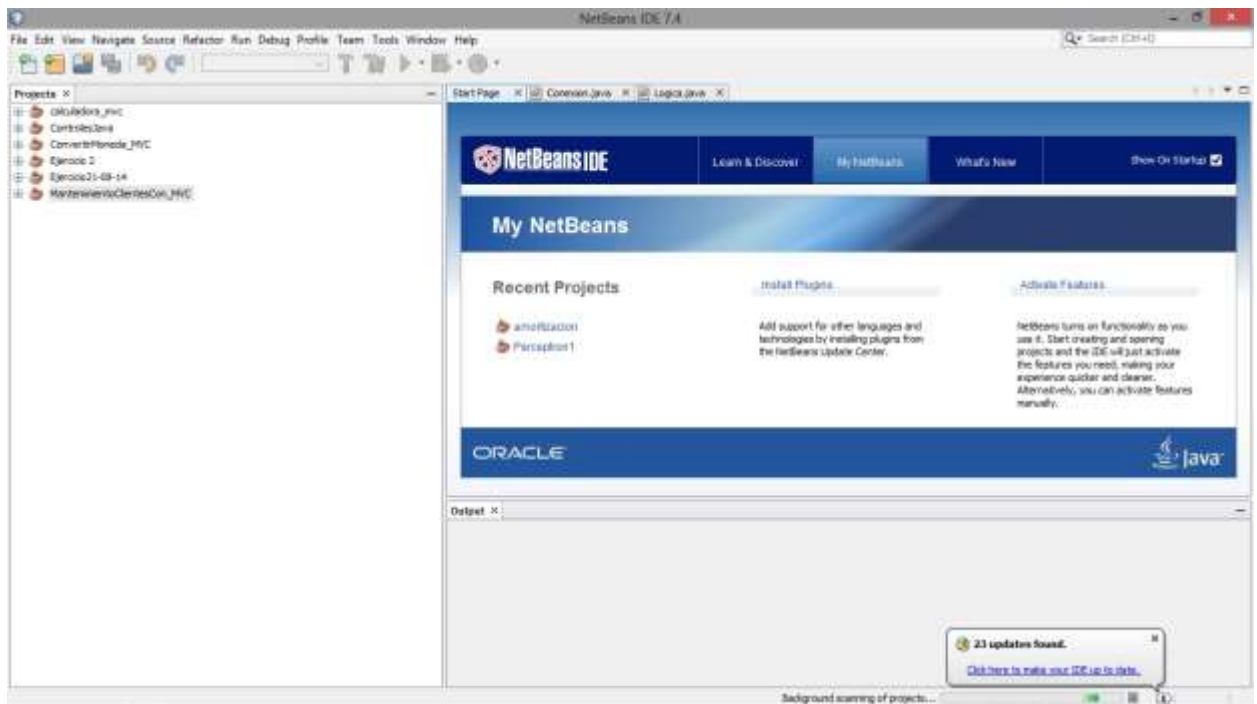
Procedimiento para ejecutar aplicaciones del Grupo 1

- Descargar las aplicaciones a utilizar del repositorio designado para su almacenamiento. Cuyo servicio es Github con la cuenta:

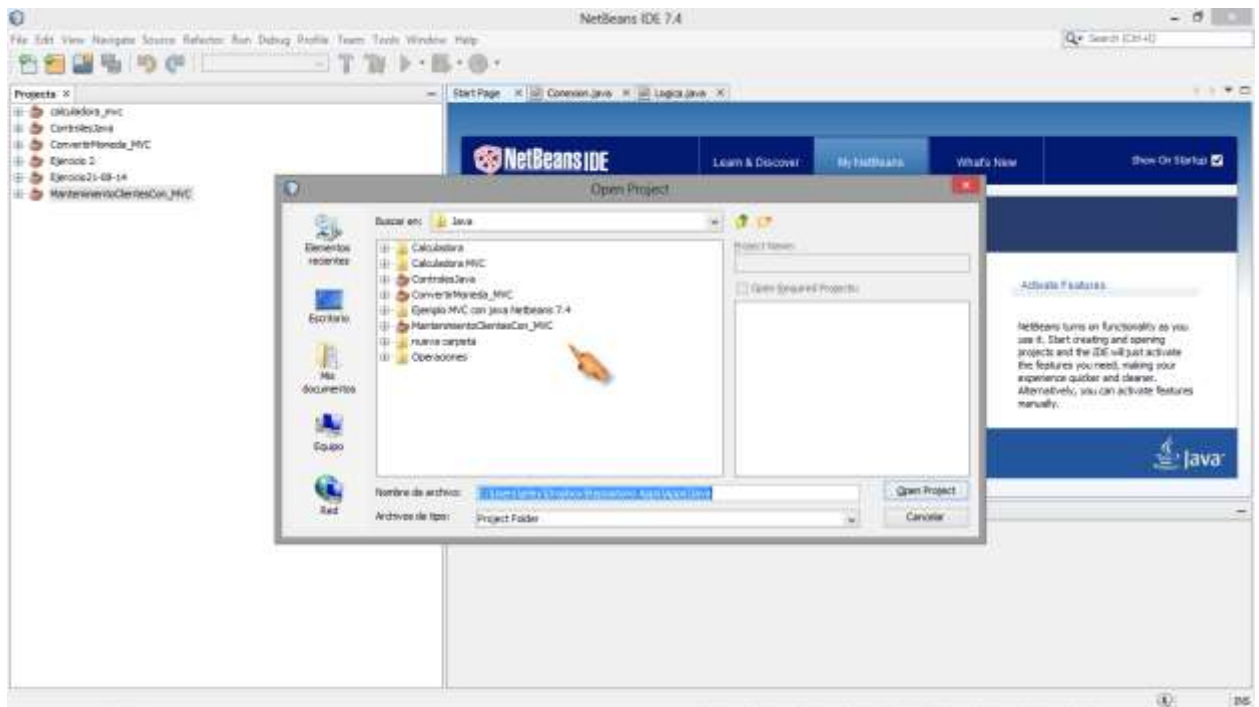
Usuario: aplicacionesudi@gmail.com

Password: appsudi2015

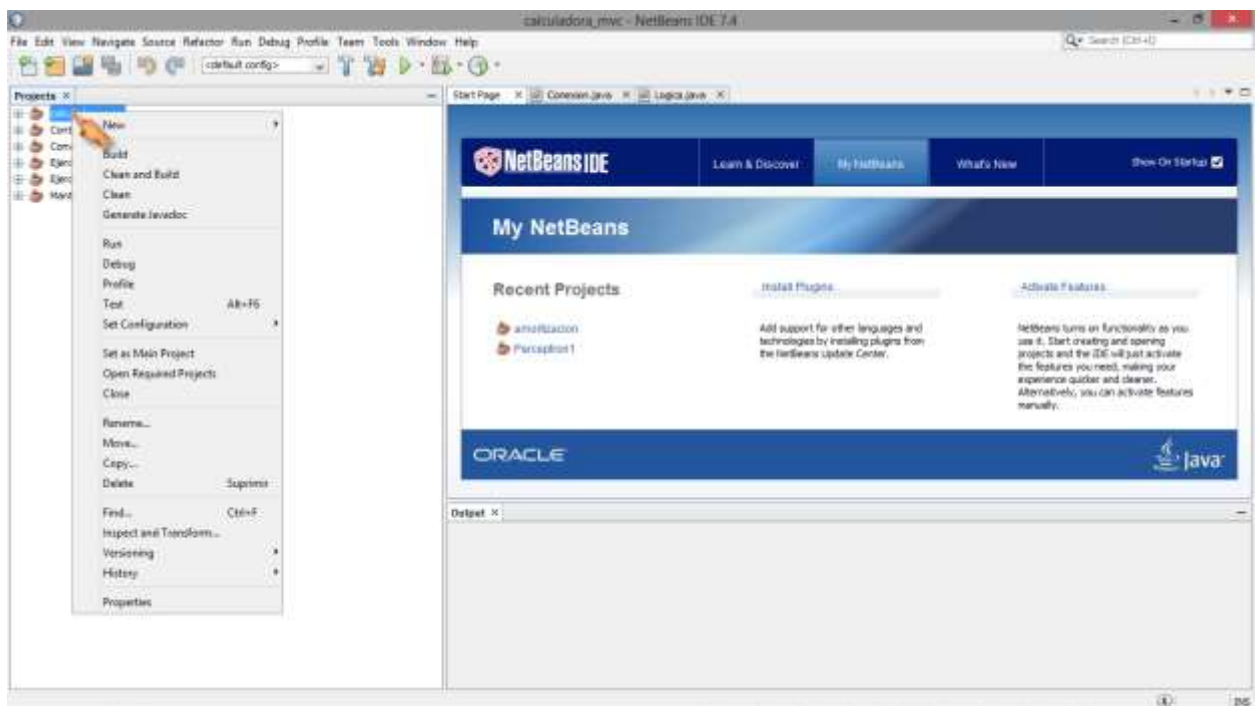
Para ejecutar aplicaciones del grupo 1, ejecutar el IDE de desarrollo Netbeans 7.4



Dirigirse al menú Archivo-Abrir proyecto y ubicar la carpeta de la aplicación que se desea abrir, ubicando la carpeta src para abrir el proyecto



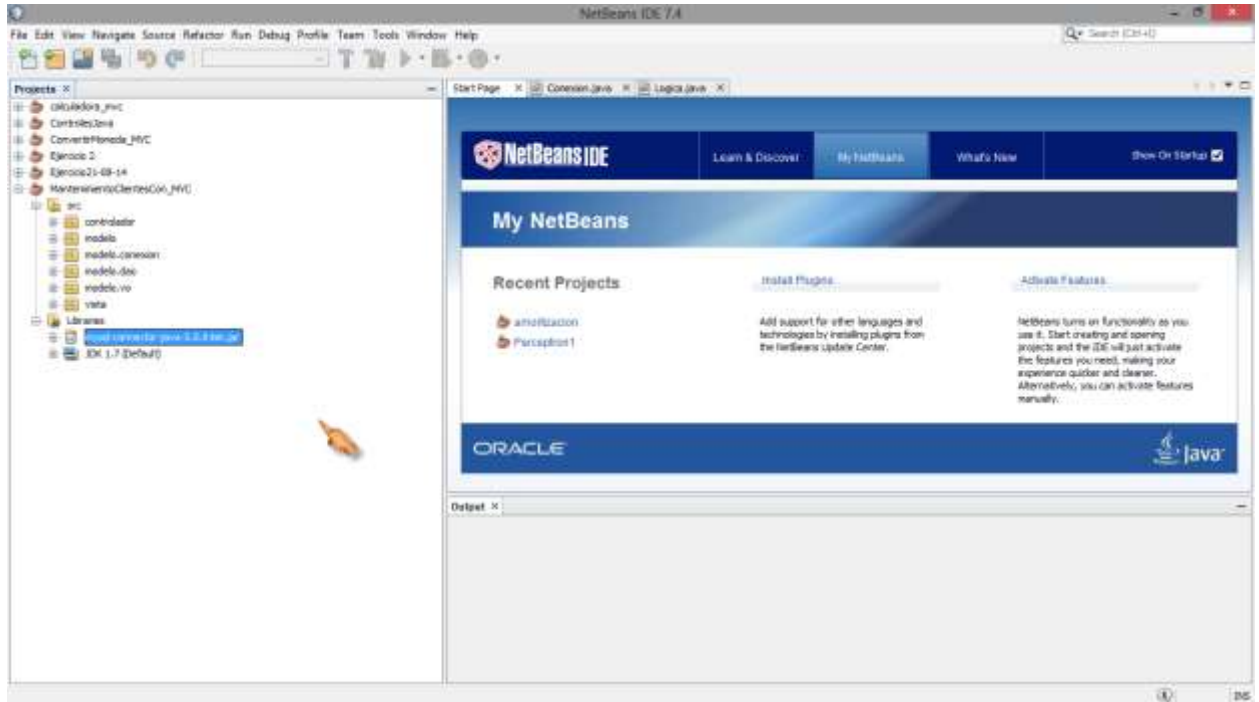
Una vez abierto, para su ejecución se debe estar ubicado en la parte izquierda del IDE, sección projects, pulsar click derecho sobre el proyecto y seleccionar la opción ejecutar (run)



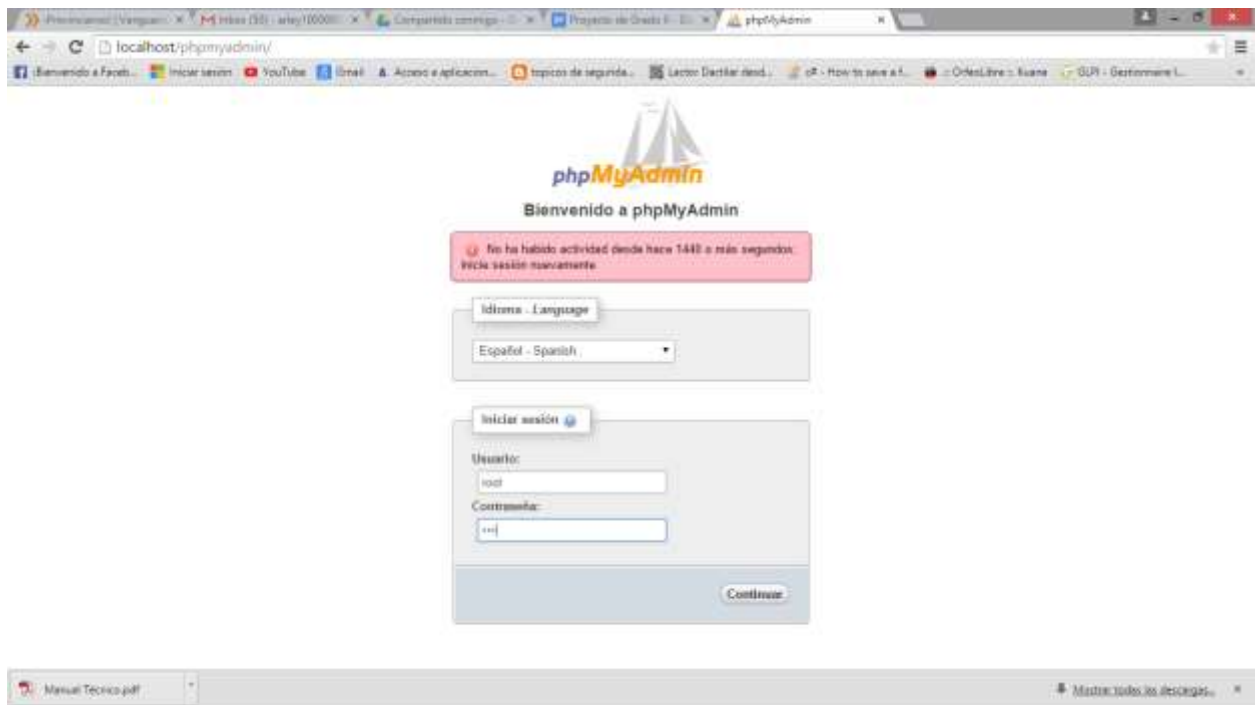
Este procedimiento se repite para todas las aplicaciones de este grupo, exceptuando las aplicaciones que manejen base de datos, para las cuales el IDE

solicita ubicar en el proyecto el archivo que sirve de conector entre el IDE- lenguaje de programación y el gestor de base de datos.

Dicho archivo se puede ubicar en la carpeta del proyecto o se puede descargar desde su pagina oficial, su nombre es mysql-connector-java-5.0.8-bin. el cual debe estar agregado en las librerias del proyecto.

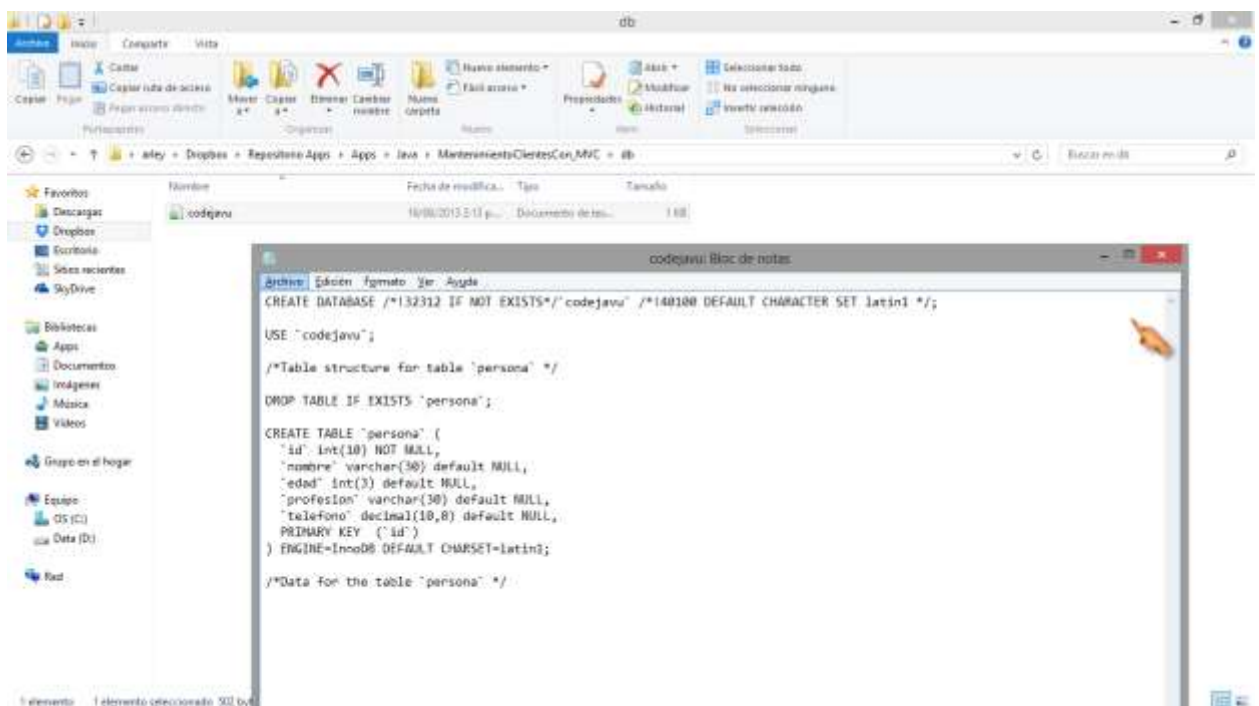


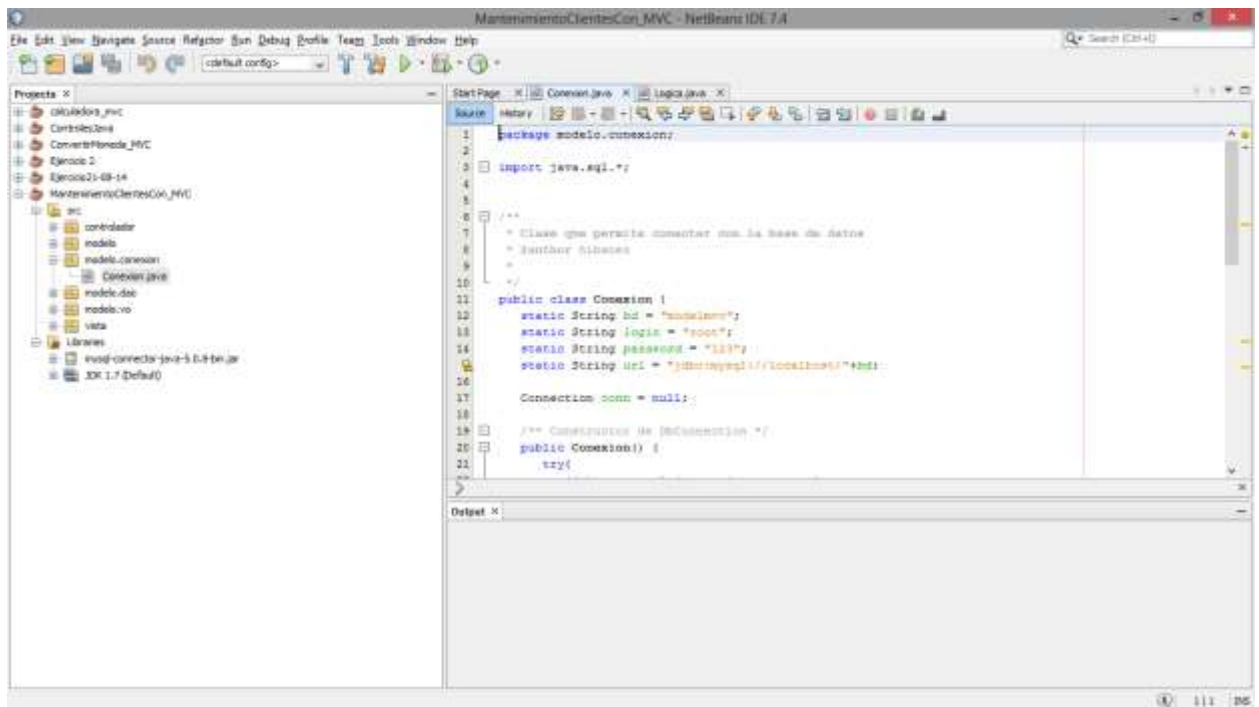
La base de datos debe estar ubicada en MYSQL, para esto se ingresa por phpmyadmin con el usuario respectivo



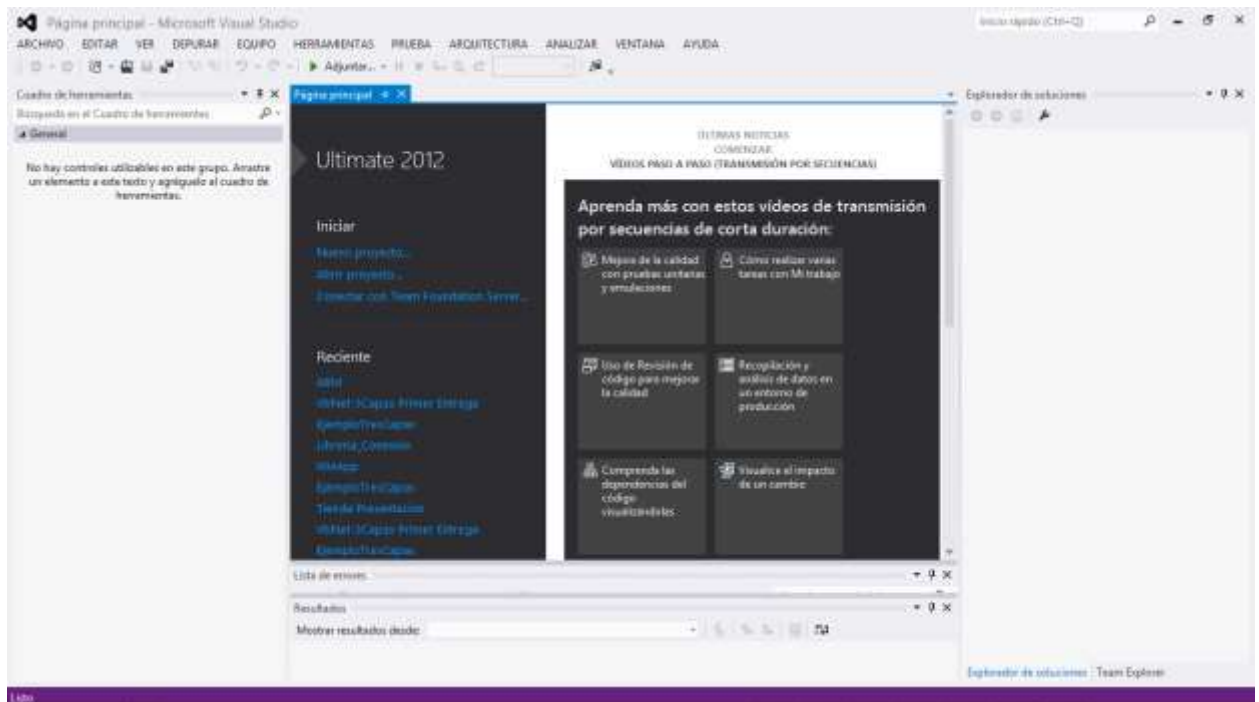
El script de creación de la base de datos se encuentra en la carpeta db que hace parte del proyecto, dicho script se debe crear en MYSQL con el nombre de la base de datos correspondiente y ajustar los parametros de conexion en la clase conexxion del proyecto, donde deben coincidir los datos con la base de datos del servidor.

Ejemplo Script:

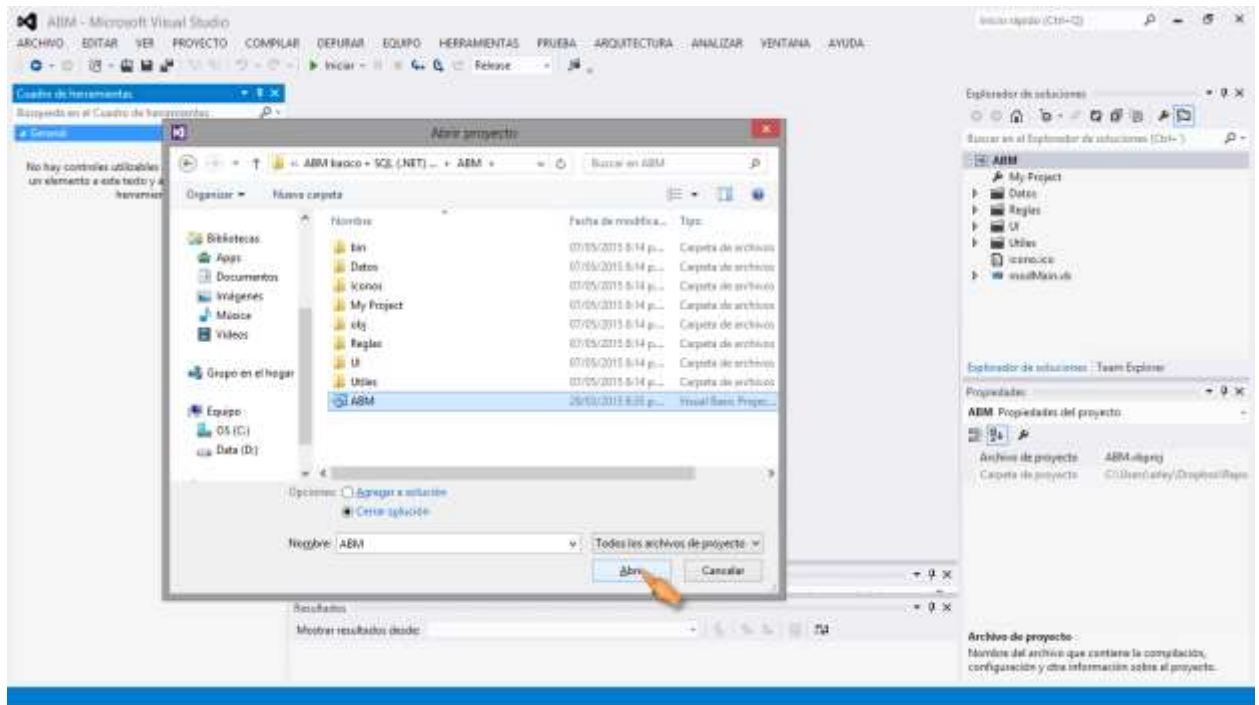




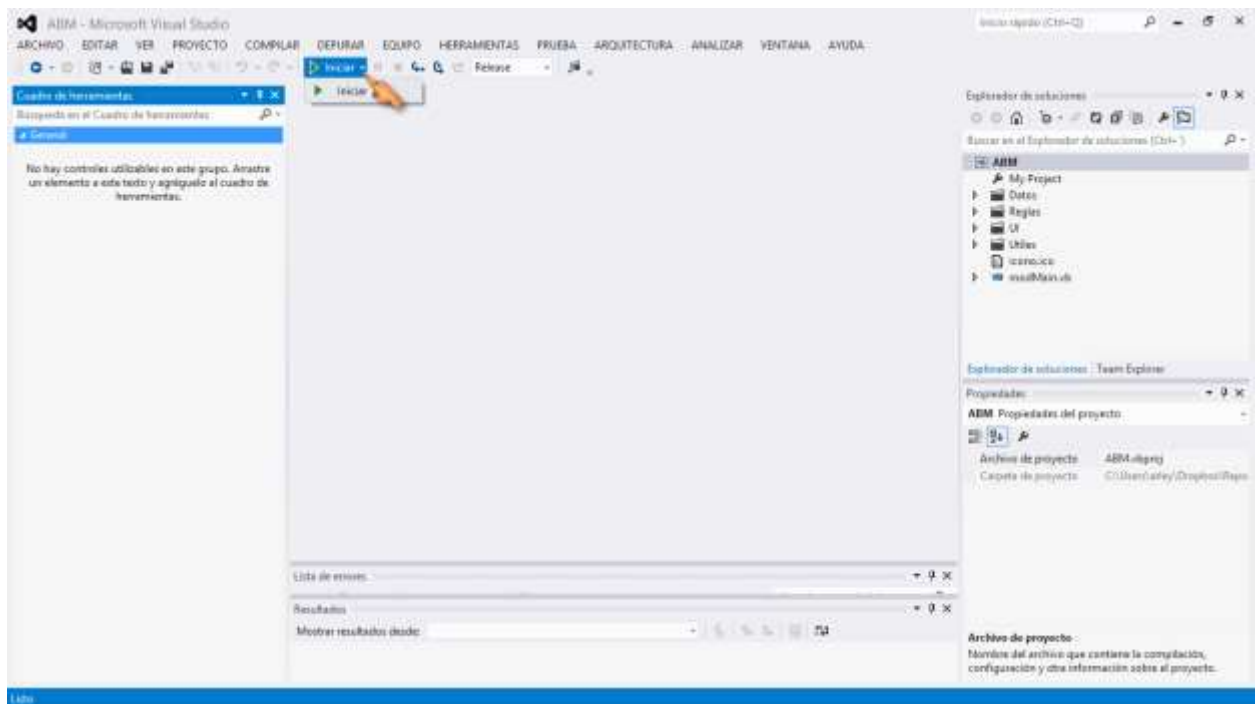
Para ejecutar aplicaciones del grupo 2, ejecutar el IDE de desarrollo Microsoft Visual Studio 2012



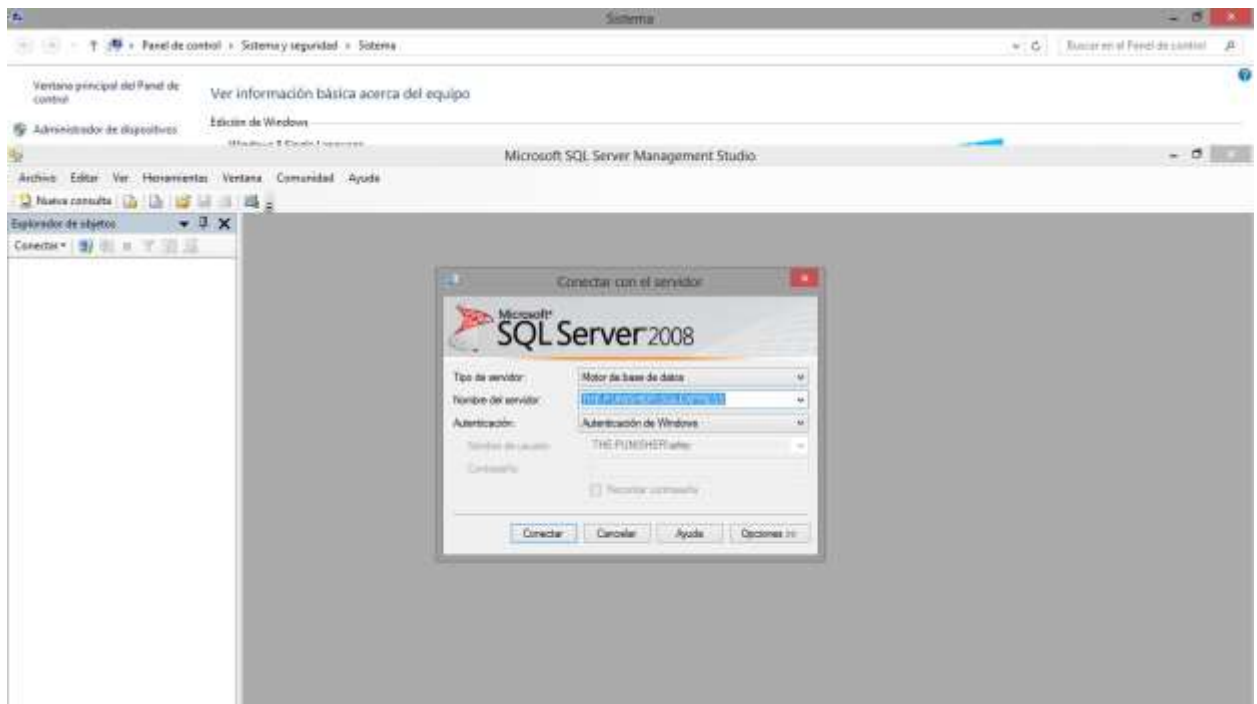
Dirigirse al menu Archivo-Abrir proyecto-Ubicar el archivo correspondiente al archivo que se desea abrir



En la parte derecha del IDE se visualiza la separación del proyecto en las diferentes capas. Para ejecutar el proyecto pulsar la tecla f5 o haciendo click en la opcion iniciar

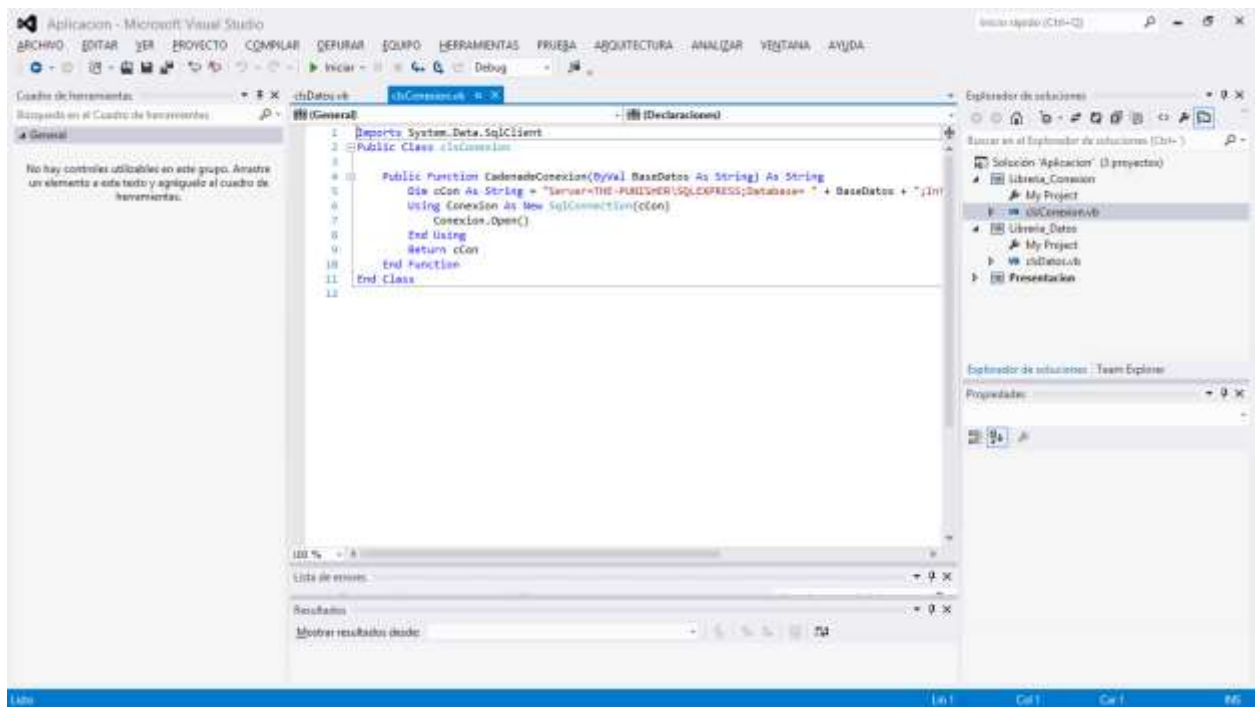


Las bases de datos de las que solicitan acceso estas aplicaciones para mostrar su información se encuentran ubicadas en el gestor de base de datos SQL Server 2008 r2



Para la aplicación que lista las Bases de Datos de un Servidor, junto con sus Tablas y Registros bajo POO y Arquitectura de capas sus parámetros de conexión se encuentran ubicados en la clase clsConexion, los cuales deben ser

ajustados acorde a la instalación de SQL Server existente en el equipo a ejecutar



APLICACIONES BASICAS EN CAPAS PARA EL APOYO DEL ESTUDIO DE LA PROGRAMACIÓN ORIENTADA A OBJETOS.

Temario de manual de aplicaciones

Algoritmos

Se puede determinar como un algoritmo a una secuencia de pasos estructurados, ordenados y finitos.

Un software de computación es una secuencia de órdenes que describen un algoritmo, estas deben ser escritas de forma adecuada para que puedan ser interpretadas por el ordenador. Dicho lo anterior se entiende que debe llevarse un proceso o secuencia de acciones que podrían describirse como:

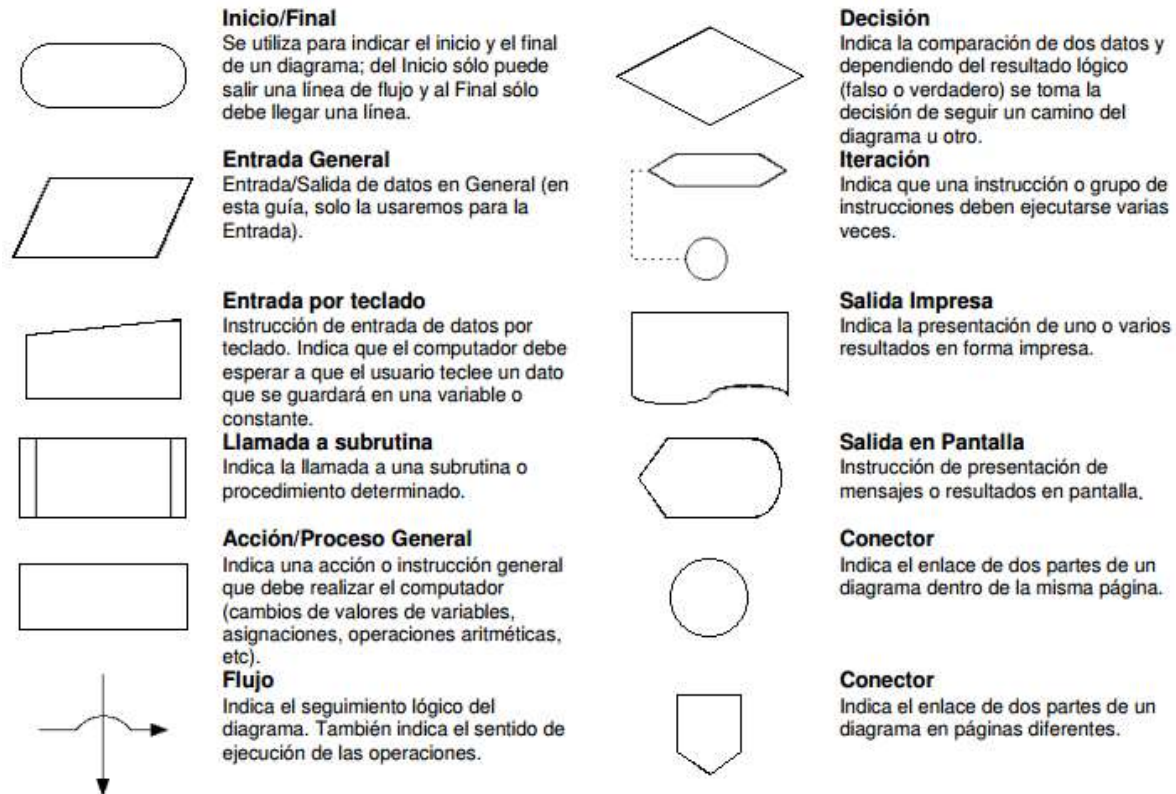
- Entrada de datos o información: Son los caracteres de referencia que necesita el algoritmo para dar inicio.
- Proceso de datos: Son las consultas u operaciones que realiza el algoritmo.
- Salida de datos: Resultados obtenidos.

Para representar estas acciones las herramientas comúnmente más utilizadas en los algoritmos son: Los diagramas de flujo y los Seudocódigos.

Diagramas de Flujo

Estos diagramas los podemos describir como graficas de secuencias de acciones a elaborar. Cada proceso que se realiza dentro del diagrama se representa mediante un símbolo normalizado por el ANSI (American National Standards Institute).

Figura. Símbolos de representación utilizados en los diagramas de flujo



Fuente: ingenieriasdesistemas.net obtenido de <http://www.ingenieriasistemas.net/2012/12/simbologia-de-los-diagramas-de-flujo.html>

Pseudocódigos

Son estructuras de algoritmos similares al lenguaje de programación, cuenta con un nivel de complejidad muy bajo para interpretar e implementar a un lenguaje de programación, son más compactos en comparación a los diagramas de flujo.

Ejemplo

Calcular la altura en pulgadas (1 pulgada=2.54cm) y pies (1 pie=12 pulgadas), a partir de la altura en centímetros introducida por el teclado.

❖ Pseudocódigo

Inicio

1-Imprimir 'Introduce la altura en centímetros: '

2-Leer: altura

3-Calcular $\text{pulgada} = \text{altura} / 2.54$

4-Calcular $\text{pies} = \text{pulgada} / 12$

5-'Imprimir la altura en pulgadas es:', pulgada

6-'Imprimir la altura en pies es:', pies

Fin

❖ Diagrama de Flujo

Figura14: Representación gráfica del diagrama de flujo del ejemplo anterior



Fuente: Autores del proyecto

Condicional multiple IF – ELSEIF - ELSE.

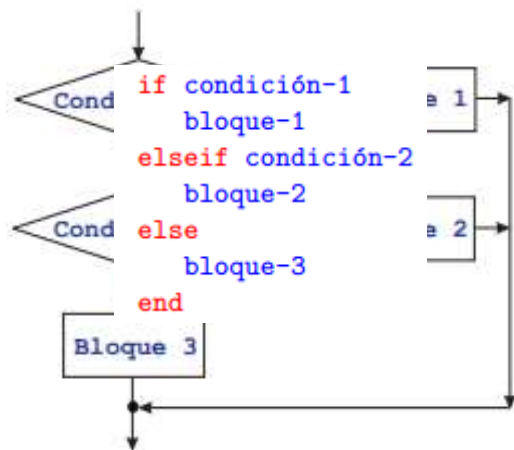
Esta estructura en su forma más general, nos permite implementar condicionales un poco más complejas en las cuales encadenamos o concatenamos una condición con otra.

Ejemplo

- Si se cumple la condición 1, ejecutar la instancia 1
- Si no se cumple la condición 1, pero si cumple la condición 2, ejecutar la instancia 2
- Y si no cumple con ninguna de las condiciones anteriores, ejecutar la instancia 3

❖ Diagrama de Flujo

Figura15: Representación gráfica del diagrama de flujo del ejemplo anterior



Fuente: Autores del proyecto

❖ Pseudocódigo

Inicio

- 1- Leer X
- 2- SI $X > 0$
- 3- Imprimir 'El número que digitó es positivo'
- 4- SI NO, SI $X < 0$
- 5- Imprimir 'El número que digitó es negativo'
- 6- SI NO,
- 7- Imprimir 'El número que digitó es nulo'

Fin

Figura16: Operadores matemáticos y lógicos utilizados en los algoritmos

Operadores Aritméticos:

SÍMBOLO	DESCRIPCION	EJEMPLO	ORDEN DE EVALUACION
+	SUMA	$a + b$	3
-	RESTA	$a - b$	3
*	MULTIPLICACION	$a * b$	2
/	DIVISION	a / b	2
%	MODULO	$a \% b$	2
-	SIGNO	$-a$	2

Operadores lógicos:

SÍMBOLO	DESCRIPCION	EJEMPLO	ORDEN DE EVALUACION
&&	Y (AND)	$(a > b) \&\& (c < d)$	10
	O (OR)	$(a > b) (c < d)$	11
!	NEGACION (NOT)	$!(a > b)$	1

Fuente: ingenieriasdesistemas.net obtenido de <http://www.ingenieriasistemas.net/2012/12/simbologia-de-los-diagramas-de-flujo.html>

Figura17: Operadores matemáticos y lógicos utilizados en los algoritmos

Operadores Relacionales:

=	Igual
<>	Distinto
<	Menor
>	Mayor
<= ó =<	Menor o igual
>= ó =>	Mayor o igual

Operador de incremento o decremento

SÍMBOLO	DESCRIPCION	EJEMPLO	ORDEN DE EVALUACION
++	incremento	++i ó i++	1
--	decremento	--i ó i--	1

Fuente: ingenieriasdesistemas.net obtenido de <http://www.ingenieriasistemas.net/2012/12/simbologia-de-los-diagramas-de-flujo.html>

Estructura de repetición FOR

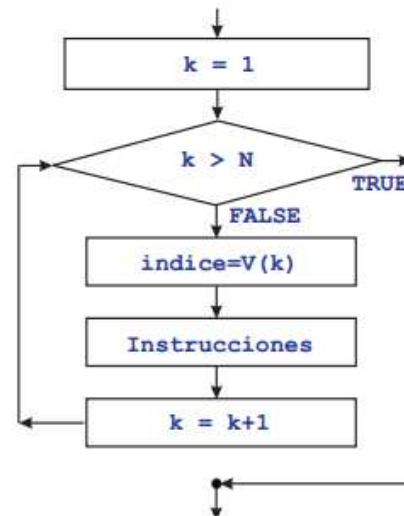
Esta estructura permite implementar la repetición de un conjunto de órdenes, un número predeterminado de veces.

Para esto se utiliza una variable de control, que también es llamada índice la cual se encarga de tener el control de la cantidad de veces que se repetirá la acción.

Diagrama de flujo

Figura18: Representación gráfica del diagrama de

Del ejemplo anterior



Fuente: Autores del proyecto
Pseudocódigo

Inicio

- 1-Leer n
- 2-Hacer Suma = 0
- 3-PARA i = 1, 3, 5....., 2*n-1
- 4-Hacer Suma = Suma + i
- 5-Fin PARA
- 6-Imprimir 'La suma vale', Suma

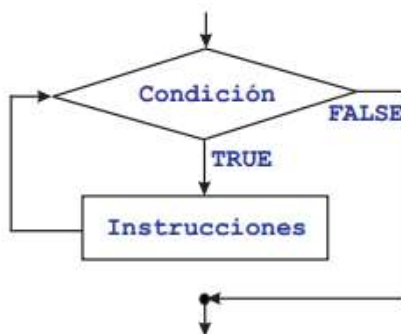
Fin

Estructura repetitiva WHILE

Esta estructura permite realizar la repetición de un mismo conjunto de órdenes mientras se verifique una determinada condición, no se define el número de veces que tendrá que repetirse el ciclo.

Diagrama de flujo

Figura19: Representación gráfica del diagrama de flujo del ejemplo anterior



Fuente: Autores del proyecto

Pseudocódigo

Inicio

- 1- $i = 1$
- 2- Mientras que $i \leq 100$
- 3- Imprimir i
- 4- Hacer $i = i + 1$
- 5- Fin Mientras

Fin

CLASES, OBJETOS, METODOS Y EVENTOS

Todos estos son características básicas de la programación orientada a objetos. A un objeto se puede denominar como un elemento de una aplicación o como una instancia particular de las clases. Las clases son consideradas como tipos de datos agrupados para un fin. En las clases podemos definir cualquier cantidad de objetos, funciones, métodos y eventos los cuales son denominados como unidades de creación básica de los objetos que hacen parte de la construcción de sus miembros.

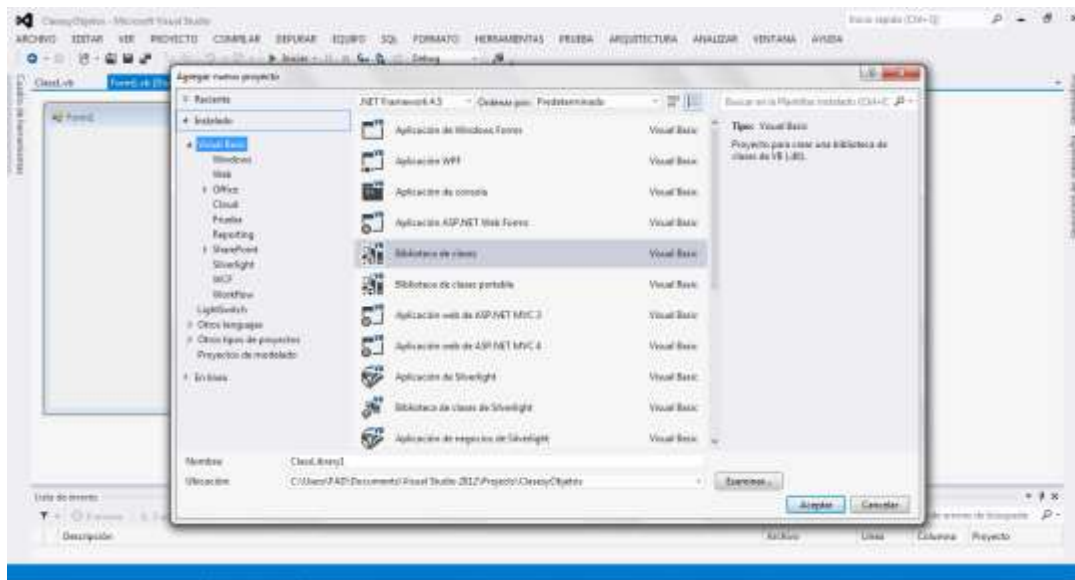
En este ejercicio encontraran como crear una biblioteca de clases como proyecto independiente, pero funcional con el proyecto principal y denominarlo como capas, como crear un objeto, una función, como llamar la función, etc.

Ejemplo

Se deberá capturar un valor en una caja de texto y en otra caja de texto visualizar la suma de todos los números que componen el valor digitado

- Primero se elige un nuevo proyecto y crea, luego ubican el explorador de soluciones en el cual se encuentra su nuevo proyecto, lo señalan y lo dejan indicado, luego se desplazan a archivo agregar nuevo proyecto, se despliega una serie de opciones para crear un nuevo proyecto dentro del que ya se tiene creado, se selección biblioteca de clases para crear la clase y se asigna un nombre. En esta ocasión se llamarán capa Datos, Capa Negocio, Capa Presentación.

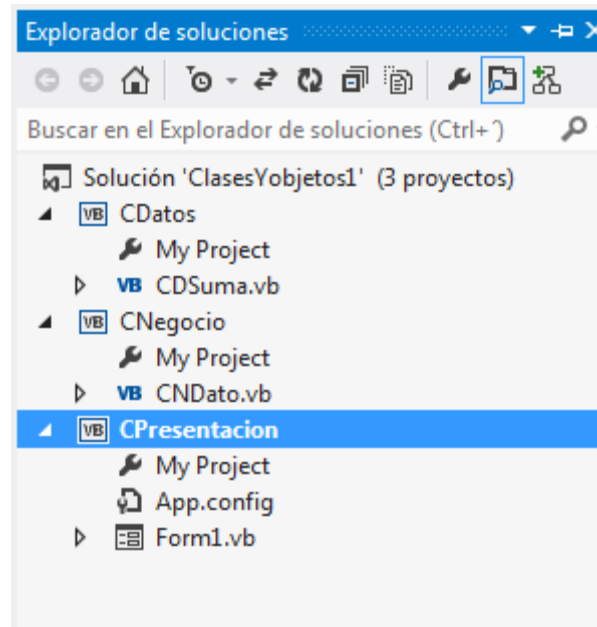
Figura. Creación de biblioteca de clases Capas



Fuente: Autores del proyecto

- Quedaría creado la nueva librería de clases dentro del proyecto algo como lo ilustra la imagen.

Figura. Presentación del explorador de soluciones del proyecto



Fuente: Autores del proyecto

- Cada capa creada contiene unas clases, se asignan nombres a esas clases para poder identificarlas más adelante. Luego se ubica cada capa y se oprime click derecho, referencias donde se asignará la comunicación entre las capas, a la capa presentación se deberá asignar como referencia la capa de negocio y a la capa de negocio se deberá asignar la capa datos, quedando esta capa como puente o intermediaria de comunicación entre estas dos capas

Figura. Asignación de referencias



Fuente: Autores del proyecto

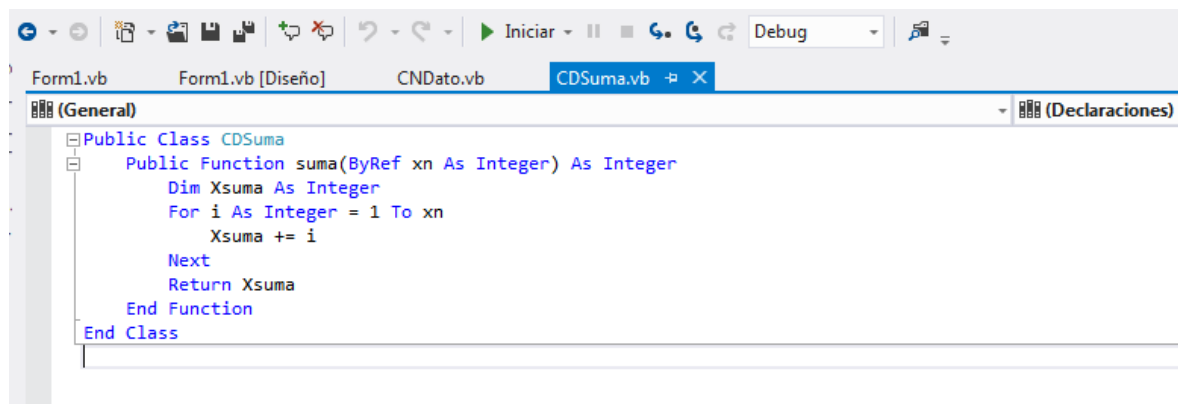
Figura23: Asignación de referencias



Fuente: Autores del proyecto

- Luego de crear las referencias se ubica la capa datos y se abre la clase perteneciente a ella, que en esta ocasión fue nombrada como CDSuma. Se crea la función suma que será la encargada de traer los datos, hacer la operación y retornar un valor que luego será llamado para visualizar un resultado.

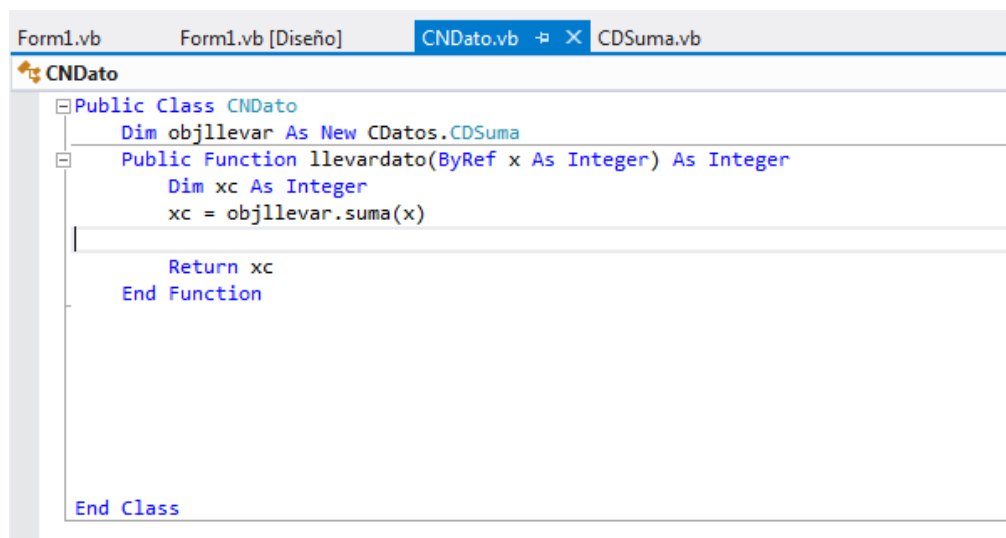
Figura. Creación de funciones y objetos en capa de datos



Fuente: Autores del proyecto

- A continuación se ubica la capa de negocio y se da click en la clase la cual fue nombrada en esta posibilidad como CNDato. En ella se realizara la transacción de datos entre la capa presentación y la capa datos donde se creará una función y se nombraran varios objetos que llevarán y traerán valores entre las capas.

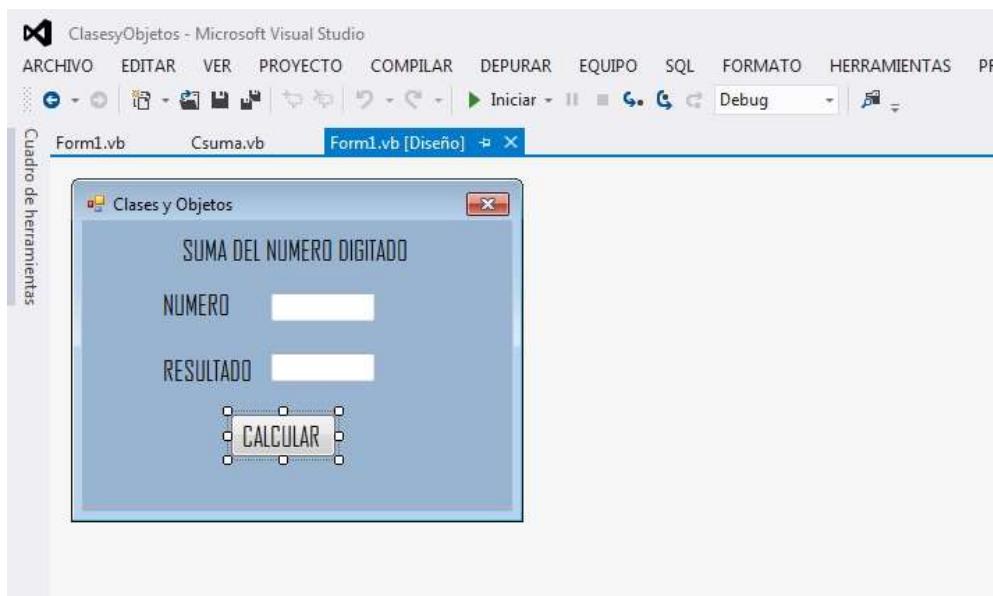
Figura. Creación de funciones y objetos en capa de negocios



Fuente: Autores del proyecto

- Se ubicara el form1 que se encuentra en la capa de presentación al cual se le hará un diseño como lo muestra la imagen que será lo necesario para desarrollar el ejemplo.

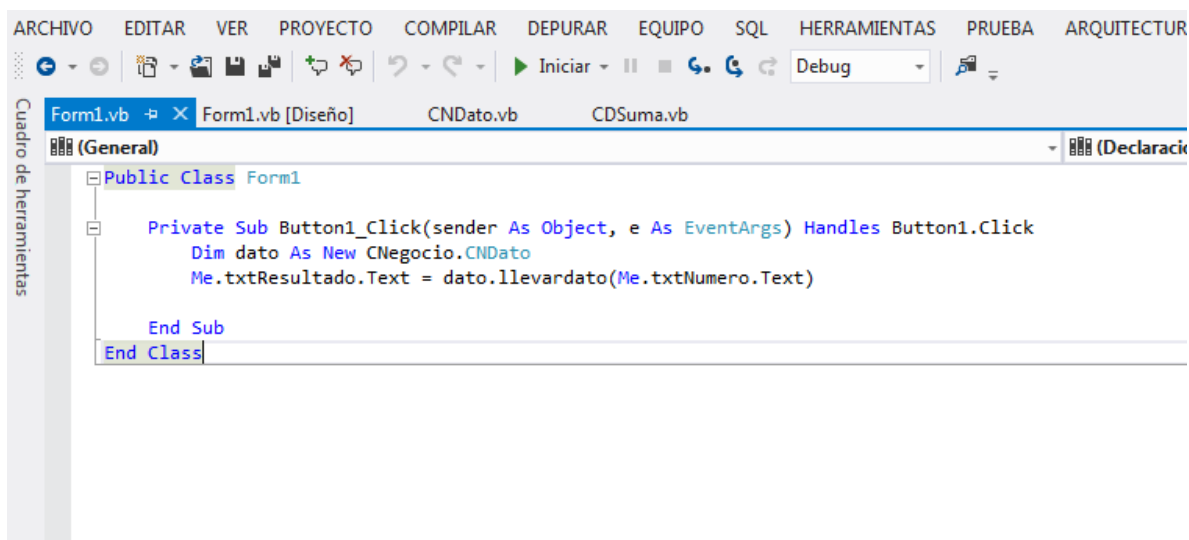
Figura. Creación del diseño del formulario en capa de presentación



Fuente: Autores del proyecto

- En el evento click del formulario se nombra el objeto que llevará los valores a la capa de negocio y visualizara dentro de la caja de texto resultado, el valor de retorno por la operación hecha en la capa de datos pero la transacción se realiza con la capa de negocio.

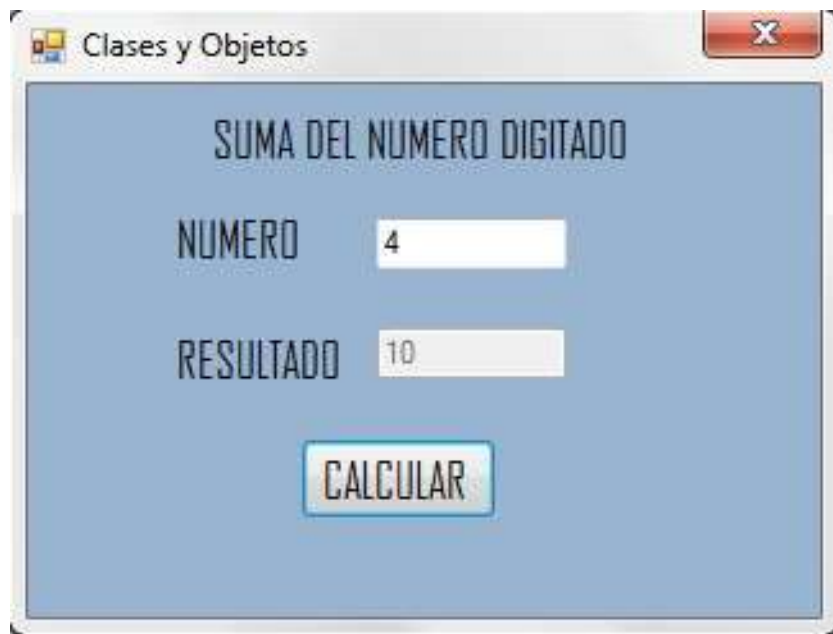
Figura. Creación del evento click del formulario



Fuente: Autores del proyecto

- Luego de completar todas las transacciones de dará click en ejecutar.
- El resultado obtenido será algo así:

Figura. Visualización del formulario ejecutado



The image shows a Java Swing window titled "Clases y Objetos". Inside the window, there is a form with a light blue background. At the top of the form, the text "SUMA DEL NUMERO DIGITADO" is displayed. Below this, there are two input fields. The first is labeled "NUMERO" and contains the value "4". The second is labeled "RESULTADO" and contains the value "10". At the bottom of the form, there is a button labeled "CALCULAR".

Fuente: Autores del proyecto

Herencia y polimorfismo

Se puede definir herencia como una función que permite obtener características tales como métodos y atributos, ya implementados en una clase madre o clase base de la cual se pueden crear clases derivadas, esta permite compartir automáticamente métodos y datos entre clases, subclases y objetos.

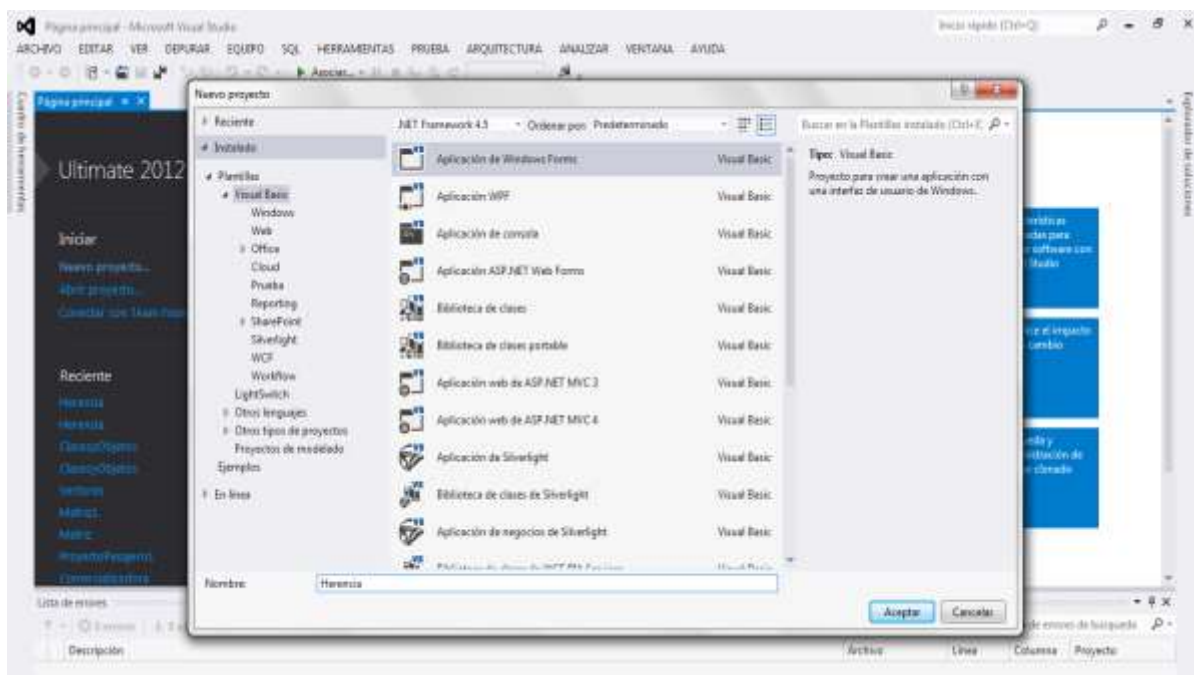
Del polimorfismo se podría decir que es una característica de la programación orientada a objetos (POO) la cual haría referencia a la posibilidad de definir múltiples clases, funciones, variables y métodos con funcionalidad diferente, pero con propiedades denominados de forma idéntica, que pueden utilizarse de manera intercambiable mediante código cliente en tiempo de ejecución.

Ejemplo

Realizar una aplicación mediante herencia y polimorfismo donde me calcule el área de algunas figuras geométricas.

1. Se crea un proyecto nuevo de Windows forms llamado herencia

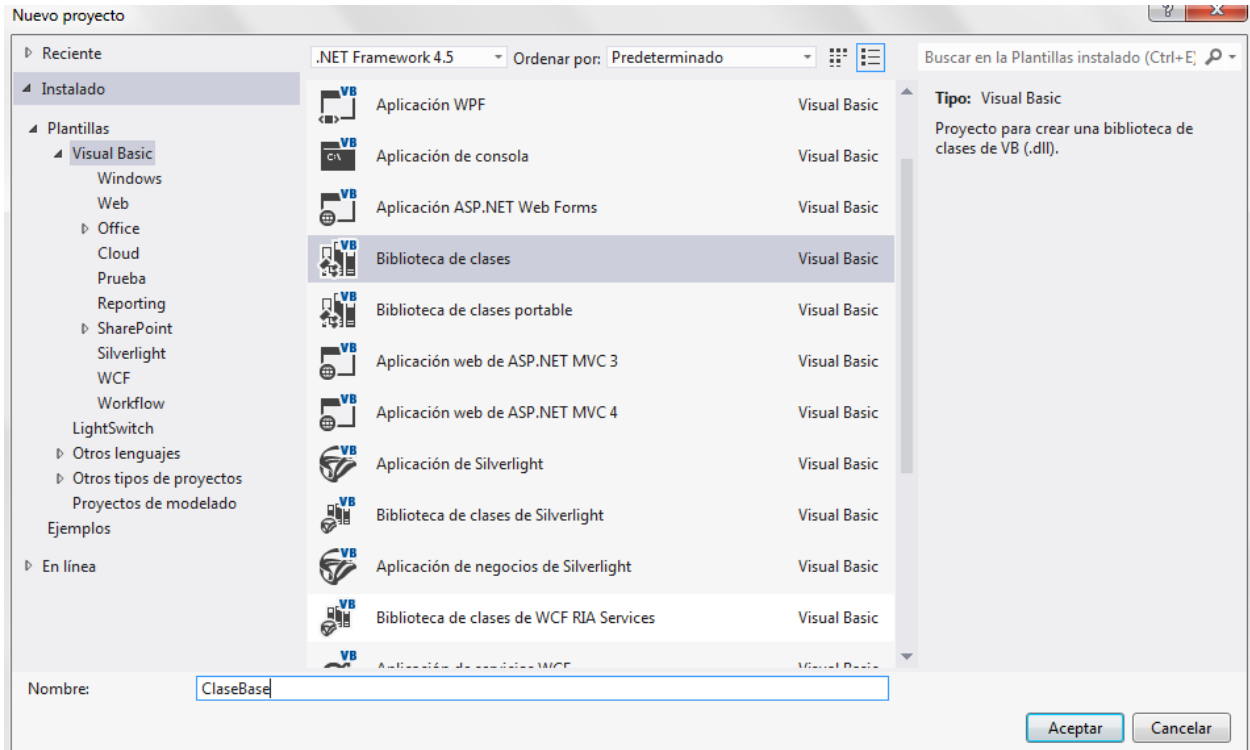
Figura. Creación de un nuevo proyecto



Fuente: Autores del proyecto

2. Teniendo ya creado el proyecto, se agrega un nuevo proyecto sobre el ya creado, pero no se selecciona proyecto de Windows forms sino se toma la opción de biblioteca de clases.

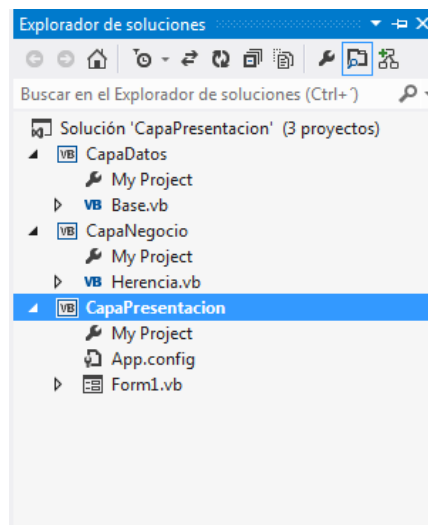
Figura. Creación de la biblioteca de clase para crear las capas



Fuente: Autores del proyecto

3. Luego a esto en el explorador de soluciones se ubica el proyecto herencia y se genera un nuevo proyecto para crear una nueva capa que se le asignara el nombre de capa de datos.

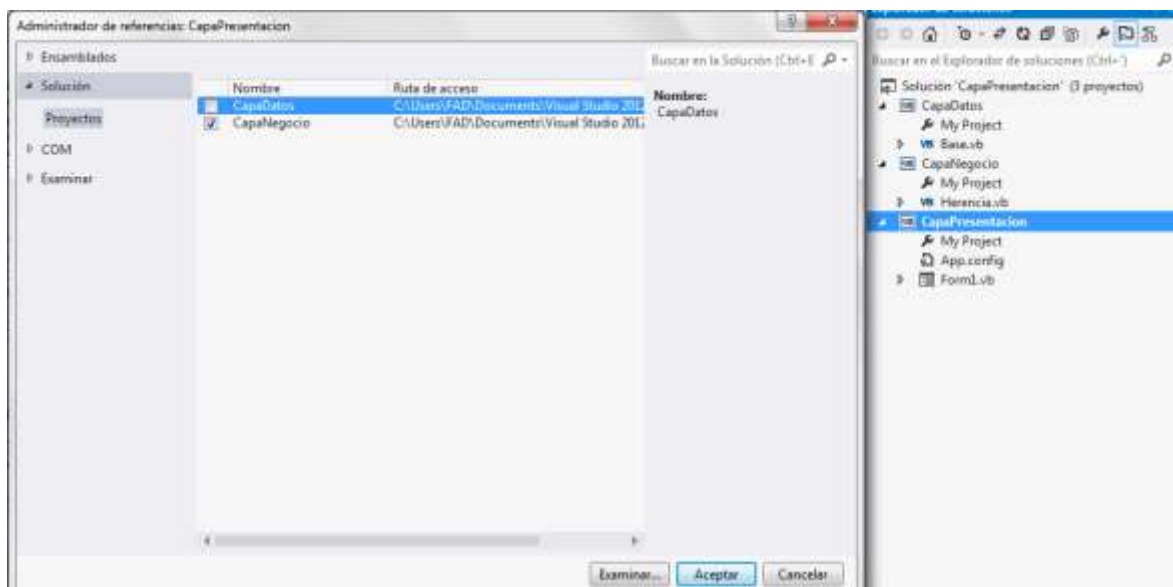
Figura31: Visualización del explorador de soluciones del proyecto



Fuente: Autores del proyecto

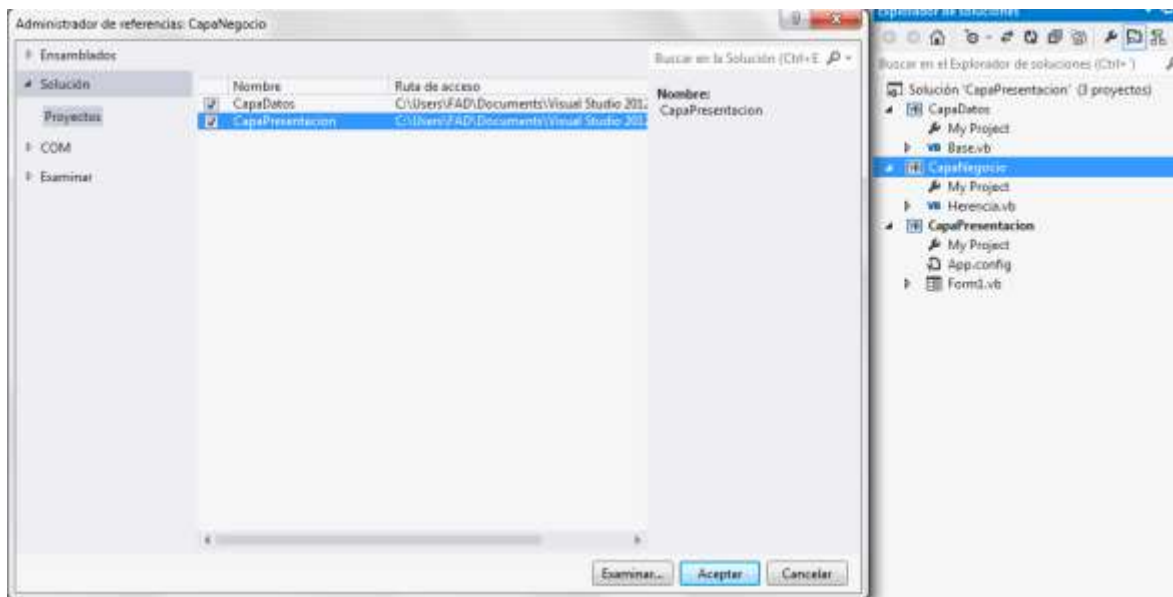
4. Ya teniendo nuestro explorador de soluciones como la imagen anterior, se pasa a crear las referencias entre las capas para que puedan tener comunicación entre ellas

Figura. Creación de las referencias entre capas



Fuente: Autores del proyecto

Figura33: Creación de las referencias entre capas



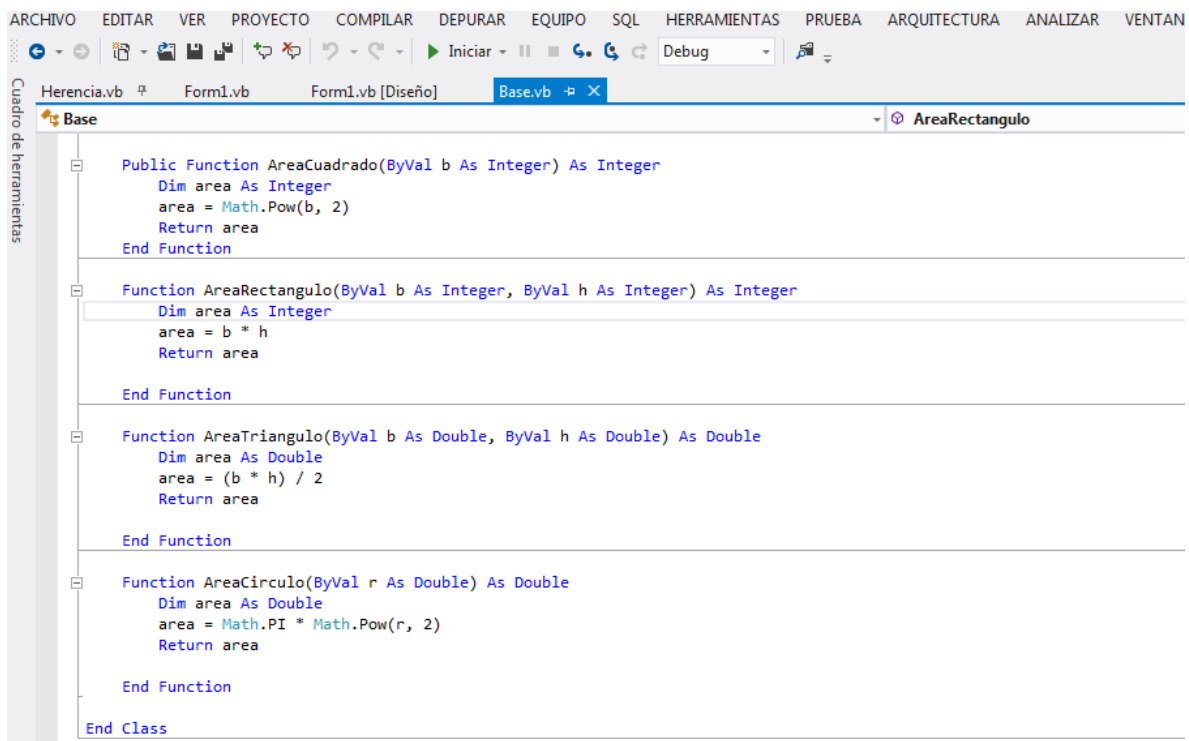
Fuente: Autores del proyecto

5. Luego de tener creadas las referencias, se ubicara la clase Base que pertenece a la capa datos la cual se le asignara la función de heredado con el comando MUSTINHERIT la cual es definida en la cabecera de la clase.

```
Public MustInherit Class Base
```

6. Luego se crean las funciones que se encargaran de hacer los cálculos respectivos para el área de cada figura

Figura. Creación de funciones y cálculos matemáticos en la capa de datos



Fuente: Autores del proyecto

- Se ubica la clase herencia que pertenece a la capa de negocio que será la encargada de heredar todas las funciones creadas en la capa base anteriormente mencionada. con el comando INHERITS se podrá heredar todas las funciones, objetos y métodos creados en la clase base.

Figura. Creación de la propiedad herencia en la capa de negocios

```

Public Class Herencia

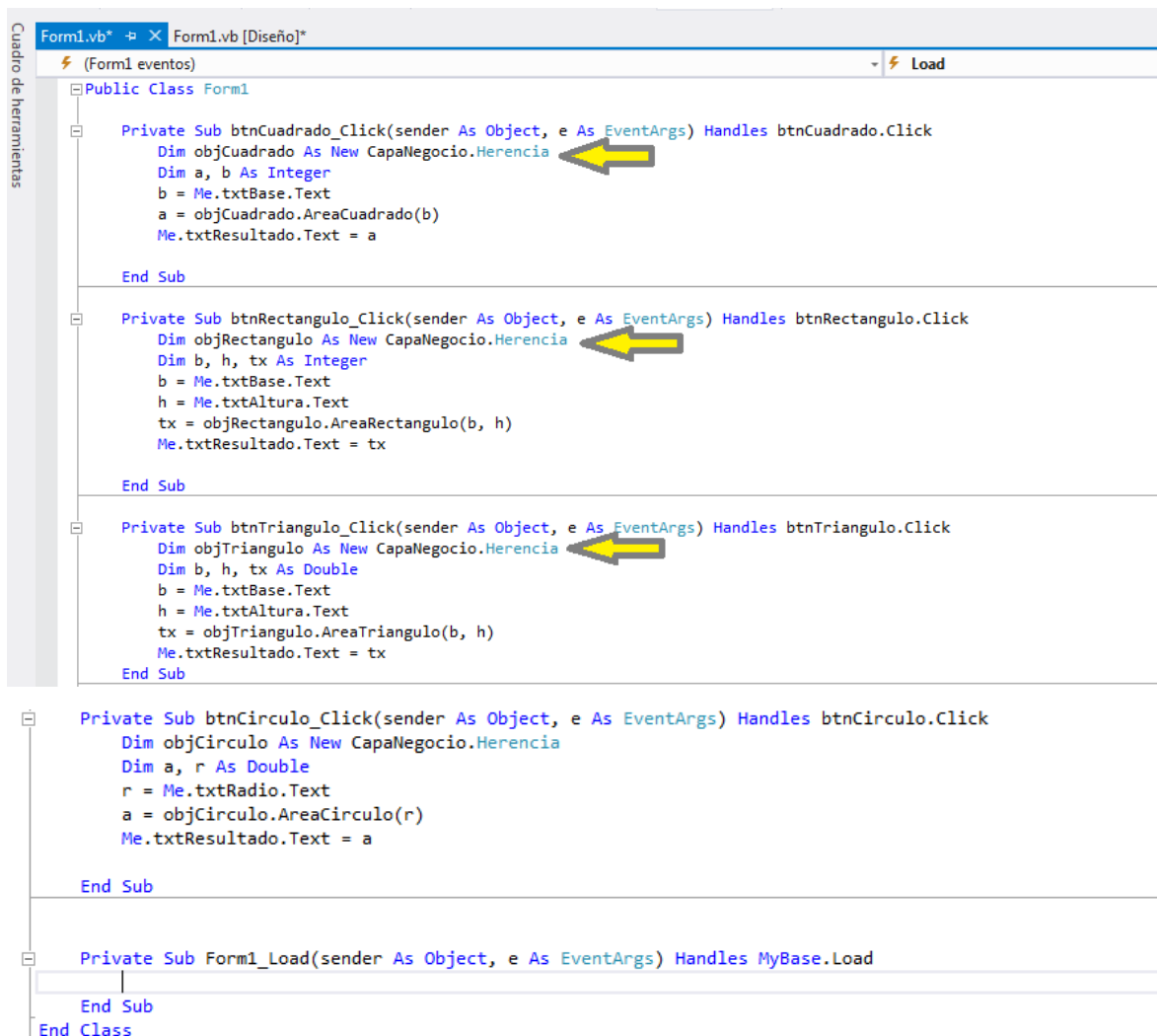
    Inherits CapaDatos.Base

End Class
    
```

Fuente: Autores del proyecto

8. Luego en el evento click de cada command buton se llamarán las funciones heredadas en la clase herencia, que son las encargadas de retornar los valores de las operaciones que fueron realizadas en la capa de datos para obtener un resultado. Se visualizara en la caja de texto resultado.

Figura. Creación del evento click dentro del formulario



```
Form1.vb*  Form1.vb [Diseño]*
(Form1 eventos) Load

Public Class Form1

    Private Sub btnCuadrado_Click(sender As Object, e As EventArgs) Handles btnCuadrado.Click
        Dim objCuadrado As New CapaNegocio.Herencia
        Dim a, b As Integer
        b = Me.txtBase.Text
        a = objCuadrado.AreaCuadrado(b)
        Me.txtResultado.Text = a
    End Sub

    Private Sub btnRectangulo_Click(sender As Object, e As EventArgs) Handles btnRectangulo.Click
        Dim objRectangulo As New CapaNegocio.Herencia
        Dim b, h, tx As Integer
        b = Me.txtBase.Text
        h = Me.txtAltura.Text
        tx = objRectangulo.AreaRectangulo(b, h)
        Me.txtResultado.Text = tx
    End Sub

    Private Sub btnTriangulo_Click(sender As Object, e As EventArgs) Handles btnTriangulo.Click
        Dim objTriangulo As New CapaNegocio.Herencia
        Dim b, h, tx As Double
        b = Me.txtBase.Text
        h = Me.txtAltura.Text
        tx = objTriangulo.AreaTriangulo(b, h)
        Me.txtResultado.Text = tx
    End Sub

    Private Sub btnCirculo_Click(sender As Object, e As EventArgs) Handles btnCirculo.Click
        Dim objCirculo As New CapaNegocio.Herencia
        Dim a, r As Double
        r = Me.txtRadio.Text
        a = objCirculo.AreaCirculo(r)
        Me.txtResultado.Text = a
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    End Sub
End Class
```

Fuente: Autores del proyecto

9. Se obtendrá un resultado como lo muestra la gráfica.

Figura. Visualización del formulario ejecutado

The screenshot shows a Windows application window titled "Form1" with a close button in the top right corner. The main content area has a light blue background and is titled "AREA DE FIGURAS" in bold black text. Below the title, there are four rows of controls:

- Row 1: The label "Base" is followed by a text box containing the number "3".
- Row 2: The label "Altura" is followed by a text box containing the number "3".
- Row 3: The label "Resultado" is followed by a text box containing the number "4.5".
- Row 4: The label "Radio Circunferencia" is followed by an empty text box.

To the right of these input fields are four buttons stacked vertically: "Cuadrado", "Triangulo", "Rectangulo", and "Circulo". The "Triangulo" button is highlighted with a blue border and a blue arrow points to it from the right. The other buttons have a grey border.

Fuente: Autores del proyecto

Vectores

Un vector es llamado o denominado ARRAY UNIDIMENSIONAL. Un array se puede determinar como una colección de posiciones de almacenamiento de datos, todos ellos con el mismo tipo de dato en una sola dimensión.

Las posiciones o longitud del vector, es definida por el usuario, el cual siempre iniciará en la posición (0).

Ejemplo:

Si se necesita un vector de 5 posiciones para almacenar números enteros se determinará o se nombrará de la siguiente manera:

Dim Vector(4) As Integer

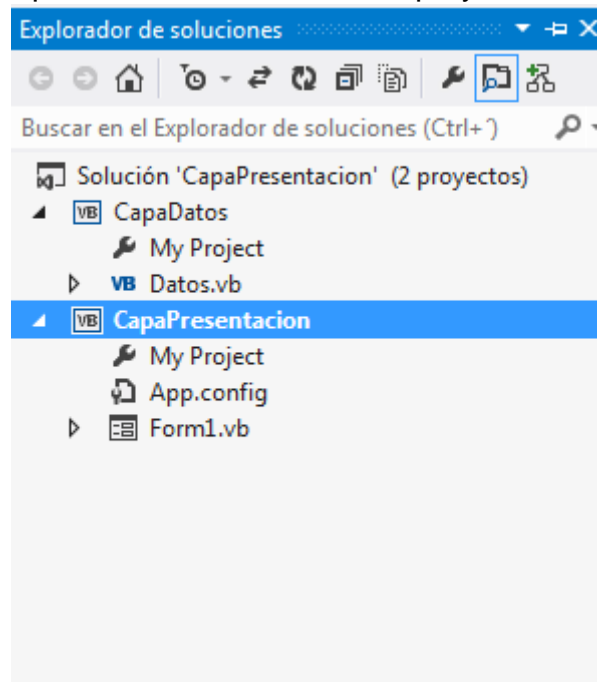
0	1	2	3	4
---	---	---	---	---

Ejemplo grafico sobre la plataforma de Visual Basic .Net

Un vector que capture un mes del año representado en número y guarde junto un valor pagado en ese mes.

1. Se crea un nuevo proyecto de Windows form se asigna un nombre el cual se llamara arreglos. Luego dentro de este se crea un nuevo proyecto y se escoge la opción librería de clases para crear las capas

Figura. Visualización del explorador de soluciones del proyecto creado

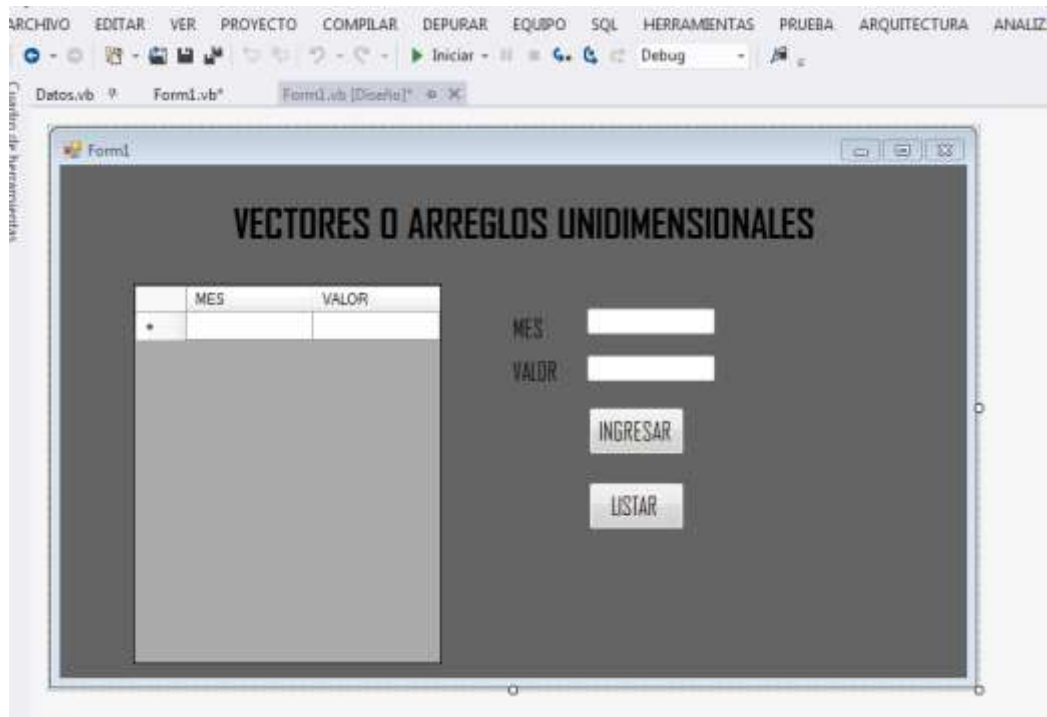


Fuente: Autores del proyecto

2. Se define la interfaz

Para esta se utilizara dos cajas de texto para capturar los valores, 2 botones de comando y 1 grilla de datos para visualizar los valores capturados en lista.

Figura. Creación del formulario para el proyecto

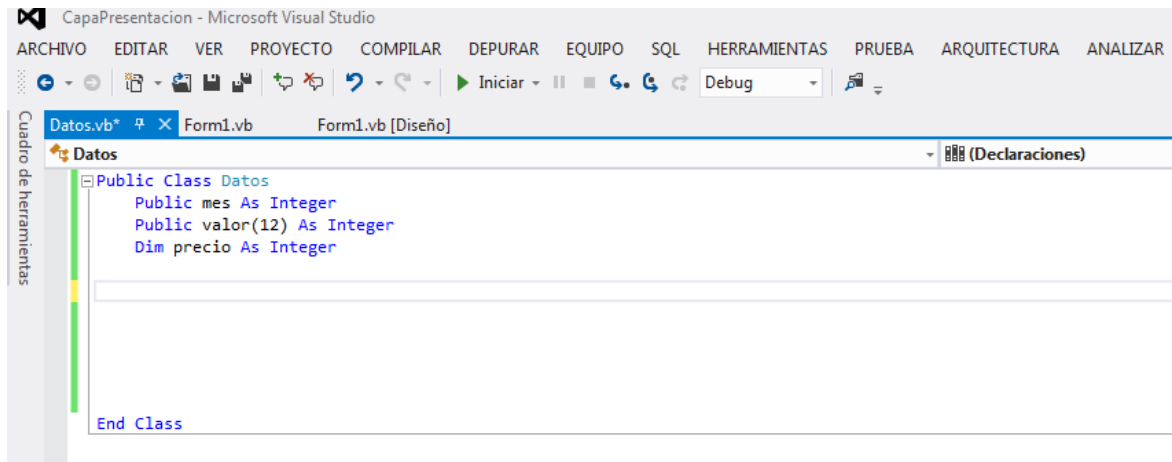


Fuente: Autores del proyecto

3. Definición de las variables y el vector

En esta ocasión se necesitan 12 posiciones del vector y se le asignaran al dicho vector esta misma cantidad de posiciones porque como ya se sabe el vector cuenta con una posición más, que es la 0, pero en este caso no se tendrá en cuenta dicha posición porque se hará referencia de los meses del año en número y el mes 0 no existe.

Figura. Definición de las variable y el arreglo utilizados en la capa datos

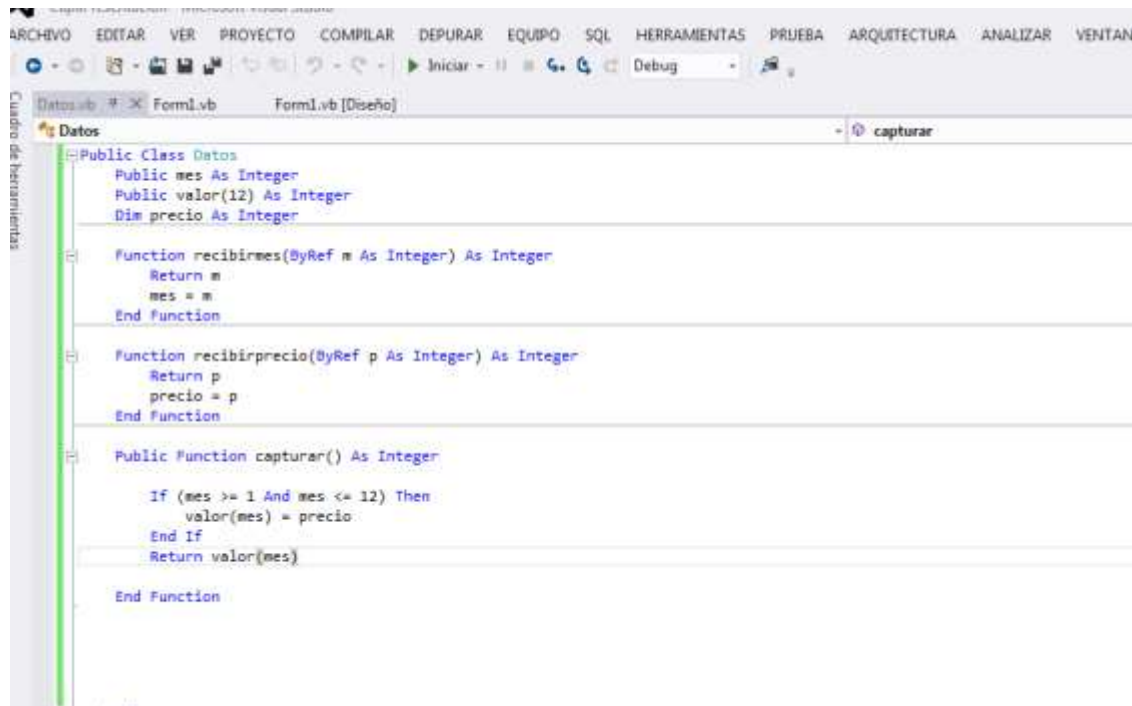


Fuente: Autores del proyecto

4. Capturar los valores

Luego de crear el vector y las variables que se necesitaran para guardar los datos se prosigue a crear las funciones que recibirán los datos traídos desde las cajas de textos para poder asignarle valor a las variables. Todo lo anterior es referenciado en la capas de datos en la clase datos

Figura. Creación de las funciones en la capa de datos



```
Public Class Datos
    Public mes As Integer
    Public valor(12) As Integer
    Dim precio As Integer

    Function recibirmes(ByRef m As Integer) As Integer
        Return m
        mes = m
    End Function

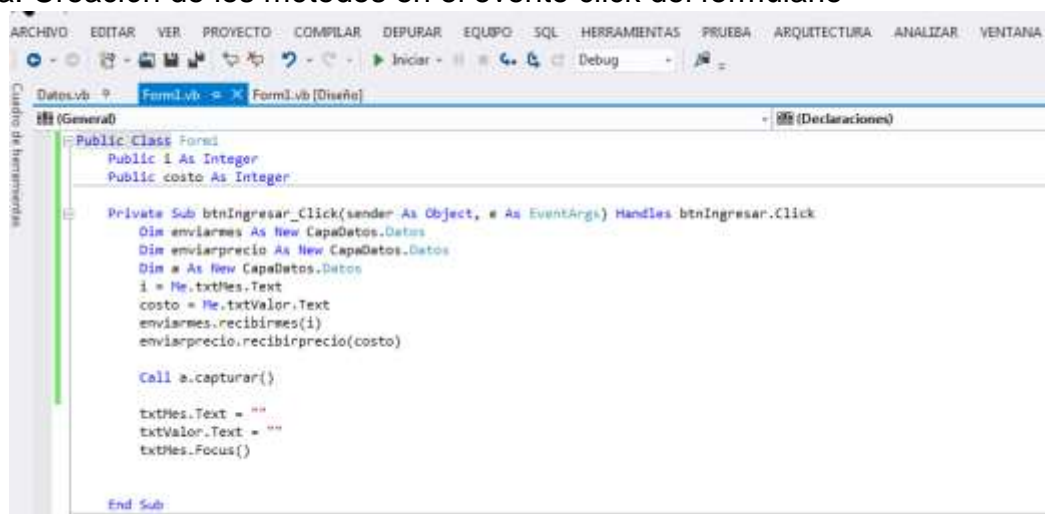
    Function recibirprecio(ByRef p As Integer) As Integer
        Return p
        precio = p
    End Function

    Public Function capturar() As Integer
        If (mes >= 1 And mes <= 12) Then
            valor(mes) = precio
        End If
        Return valor(mes)
    End Function
End Class
```

Fuente: Autores del proyecto

5. A continuación se ingresa al evento click del botón capturar para definir las variables que llevarán los valores y realizar limpieza en las cajas de texto, adicional a esto se realiza el llamado a la función capturar que se encuentra ubicada en la capa datos.

Figura. Creación de los métodos en el evento click del formulario



```
Public Class Form1
    Public i As Integer
    Public costo As Integer

    Private Sub btnIngresar_Click(sender As Object, e As EventArgs) Handles btnIngresar.Click
        Dim enviarmes As New CapaDatos.Datos
        Dim enviarprecio As New CapaDatos.Datos
        Dim a As New CapaDatos.Datos
        i = Me.txtMes.Text
        costo = Me.txtValor.Text
        enviarmes.recibirmes(i)
        enviarprecio.recibirprecio(costo)

        Call a.capturar()

        txtMes.Text = ""
        txtValor.Text = ""
        txtMes.Focus()
    End Sub
End Class
```

Fuente: Autores del proyecto

6. Listar la captura en la grilla de datos

Dentro del botón listar se asigna el valor a la variable i para que inicie desde ese valor (1), luego se limpia la grilla de datos. Contando con la función de repetición DO WHILE se asigna la condición, se cargan los datos en la grilla y se incrementa en 1 la posición de la variable i.

Figura. Creación de los métodos en el evento click del formulario

```
Private Sub btnListar_Click(sender As Object, e As EventArgs) Handles btnListar.Click
    Dim a, b As New CapaDatos.Datos
    i = 1
    dgvDatos.Rows.Clear()
    Do While (i <= 12)
        dgvDatos.Rows.Add(i, a.valor(b.mes))
        i = i + 1
    Loop
End Sub
End Class
```

Fuente: Autores del proyecto

Figura. Visualización del formulario ejecutado

The screenshot shows a Windows application window titled "Form1". The window has a dark gray background. At the top, the text "VECTORES O ARREGLOS UNIDIMENSIONALES" is displayed in a bold, black, sans-serif font. Below this text, there is a data grid with two columns: "MES" and "VALOR". The grid contains 12 rows of data. To the right of the grid, there are two input fields labeled "MES" and "VALOR", each followed by a button labeled "INGRESAR". Below these input fields, there are two buttons labeled "LISTAR" and "LIMPIAR".

MES	VALOR
1	5000
2	6000
3	24000
4	25000
5	32000
6	12000
7	90000
8	0
9	54000
10	67000
11	92000
12	102000

Fuente: Autores del proyecto

Programación en Capas

La programación en capas es una técnica de ingeniería de software que es propia de la programación orientada a objetos (POO).

La característica que posee esta técnica es que divide la aplicación general en varios módulos y capas que se puede manejar de forma independiente o de forma paralela.

En esta funcionalidad se destacan tres capas principales: La capa de presentación que es la encargada de interactuar con los usuarios, en la cual lleva los formularios e interfaces de aplicación; También se encuentra la capa lógica o de Negocio que es la encargada de la funcionalidad de servir de enlace entre las otras capas y tiene los objetos que realizan la mayor parte del trabajo; Por último se encuentra la capa de datos que es la encargada de comunicarse con la base de datos y otros sistemas de información.

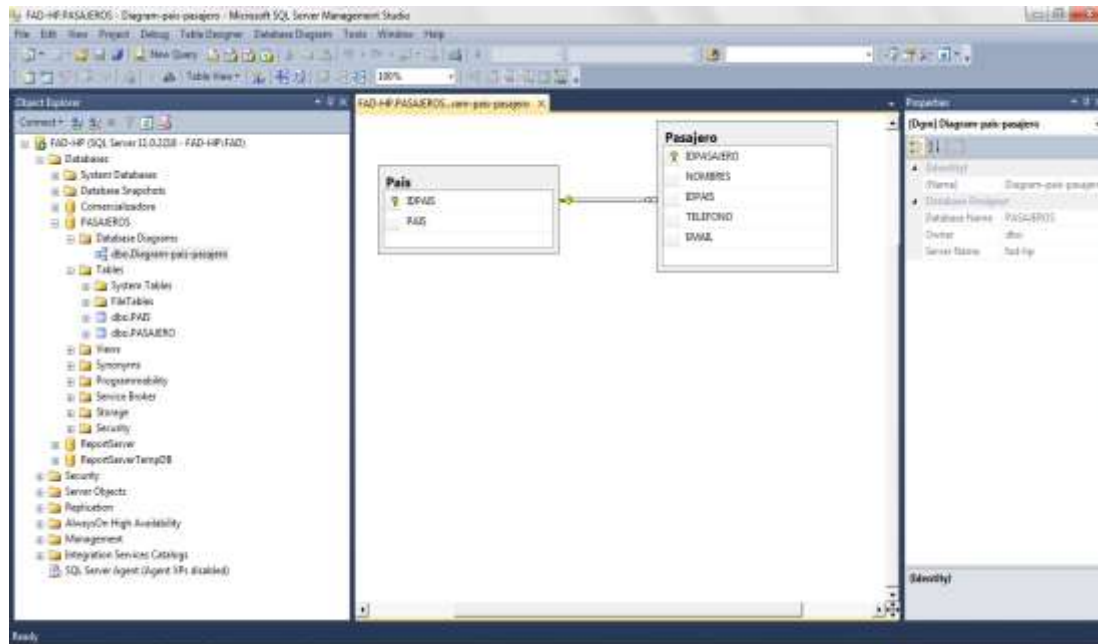
Cabe resaltar que esta técnica de programación se puede implementar de muchas formas, todo esto dependiendo de las tendencias y tecnologías que estén implementando el equipo de desarrollo del proyecto.

Ejemplo

Desarrollar una aplicación utilizando la técnica de programación en capas, que permita hacer mantenimiento a una base de datos de usuarios de una aerolínea.

1. Se crea la base de datos, en este caso se utilizara la plataforma de SQL server para administrar los datos.

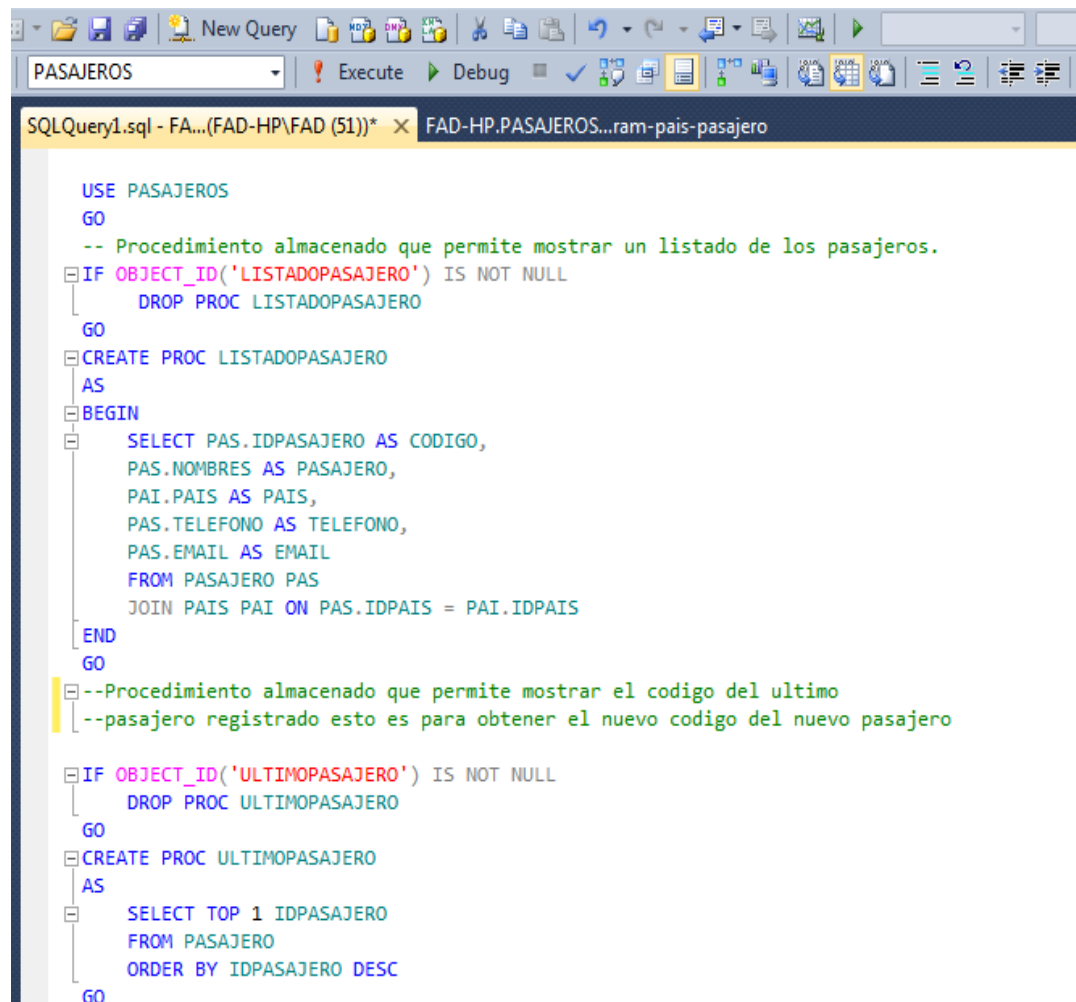
Figura. Creación de la base de datos para el proyecto



Fuente: Autores del proyecto

2. Luego de crear la base de datos y relacionar las tablas, se realizan los procedimientos almacenados requeridos por SQL server para hacer inclusiones y modificaciones a los datos almacenados en ella.

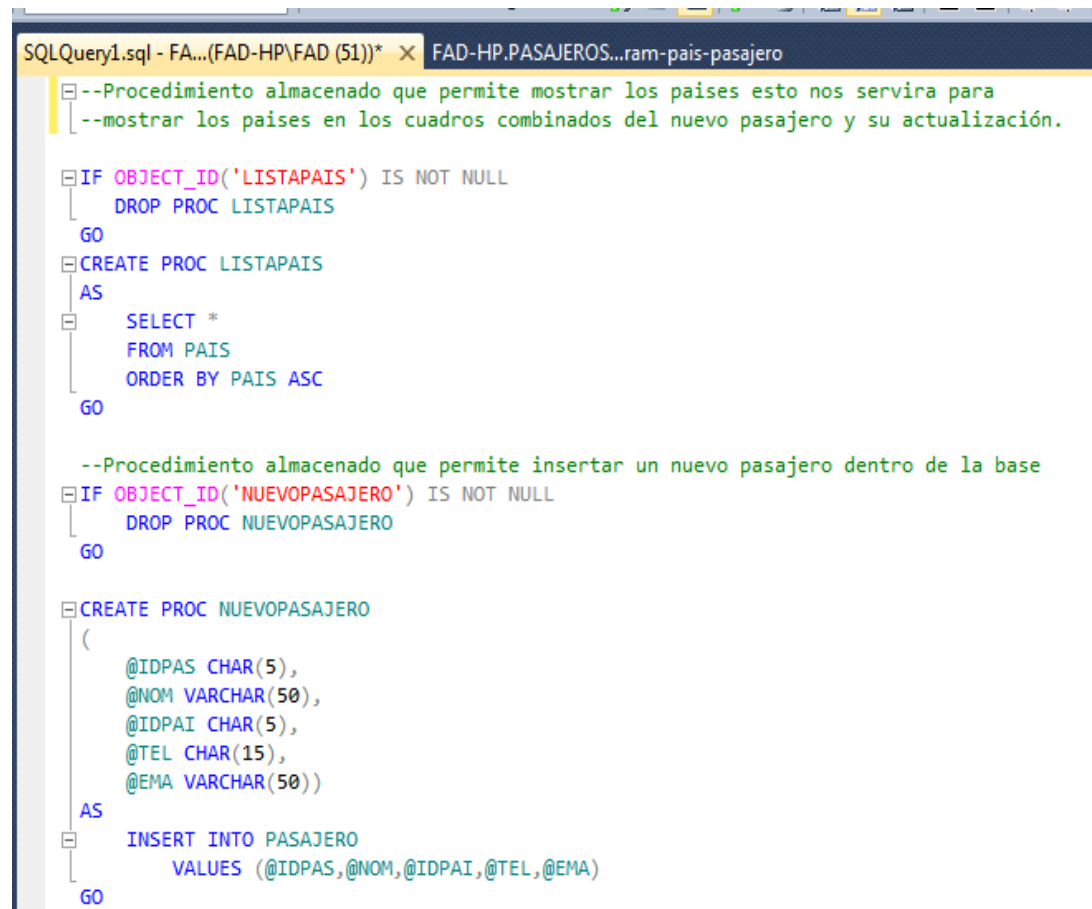
Figura. Creación de los procedimientos almacenados en la base de datos



```
USE PASAJEROS
GO
-- Procedimiento almacenado que permite mostrar un listado de los pasajeros.
IF OBJECT_ID('LISTADOPASAJERO') IS NOT NULL
    DROP PROC LISTADOPASAJERO
GO
CREATE PROC LISTADOPASAJERO
AS
BEGIN
    SELECT PAS.IDPASAJERO AS CODIGO,
           PAS.NOMBRES AS PASAJERO,
           PAI.PAIS AS PAIS,
           PAS.TELEFONO AS TELEFONO,
           PAS.EMAIL AS EMAIL
    FROM PASAJERO PAS
    JOIN PAIS PAI ON PAS.IDPAIS = PAI.IDPAIS
END
GO
--Procedimiento almacenado que permite mostrar el codigo del ultimo
--pasajero registrado esto es para obtener el nuevo codigo del nuevo pasajero
IF OBJECT_ID('ULTIMOPASAJERO') IS NOT NULL
    DROP PROC ULTIMOPASAJERO
GO
CREATE PROC ULTIMOPASAJERO
AS
    SELECT TOP 1 IDPASAJERO
    FROM PASAJERO
    ORDER BY IDPASAJERO DESC
GO
```

Fuente: Autores del proyecto

Figura. Creación de los procedimientos almacenados en la base de datos



```
SQLQuery1.sql - FA...(FAD-HP\FAD (51))* x FAD-HP.PASAJEROS...ram-pais-pasajero

--Procedimiento almacenado que permite mostrar los paises esto nos servira para
--mostrar los paises en los cuadros combinados del nuevo pasajero y su actualización.

IF OBJECT_ID('LISTAPAIS') IS NOT NULL
    DROP PROC LISTAPAIS
GO

CREATE PROC LISTAPAIS
AS
    SELECT *
    FROM PAIS
    ORDER BY PAIS ASC
GO

--Procedimiento almacenado que permite insertar un nuevo pasajero dentro de la base
IF OBJECT_ID('NUEVOPASAJERO') IS NOT NULL
    DROP PROC NUEVOPASAJERO
GO

CREATE PROC NUEVOPASAJERO
(
    @IDPAS CHAR(5),
    @NOM VARCHAR(50),
    @IDPAI CHAR(5),
    @TEL CHAR(15),
    @EMA VARCHAR(50))
AS
    INSERT INTO PASAJERO
    VALUES (@IDPAS,@NOM,@IDPAI,@TEL,@EMA)
GO
```

Fuente: Autores del proyecto

Figura. Creación de los procedimientos almacenados en la base de datos.

```
SQLQuery1.sql - FA...(FAD-HP\FAD (51))* × FAD-HP.PASAJEROS...ram-pais-pasajero

--Procedimiento almacenado que permite actualizar los datos del pasajero

IF OBJECT_ID('ACTUALIZAPASAJERO') IS NOT NULL
    DROP PROC ACTUALIZAPASAJERO
GO

CREATE PROC ACTUALIZAPASAJERO
(
    @IDPAS CHAR(5),
    @NOM VARCHAR(50),
    @IDPAI CHAR(5),
    @TEL CHAR(15),
    @EMA VARCHAR(50))
AS
    UPDATE PASAJERO
    SET NOMBRES=@NOM,
        IDPAIS=@IDPAI,
        TELEFONO=@TEL,
        EMAIL=@EMA
    WHERE IDPASAJERO=@IDPAS
GO

--Procedimiento almacenado que permite eliminar un pasajero según su código.

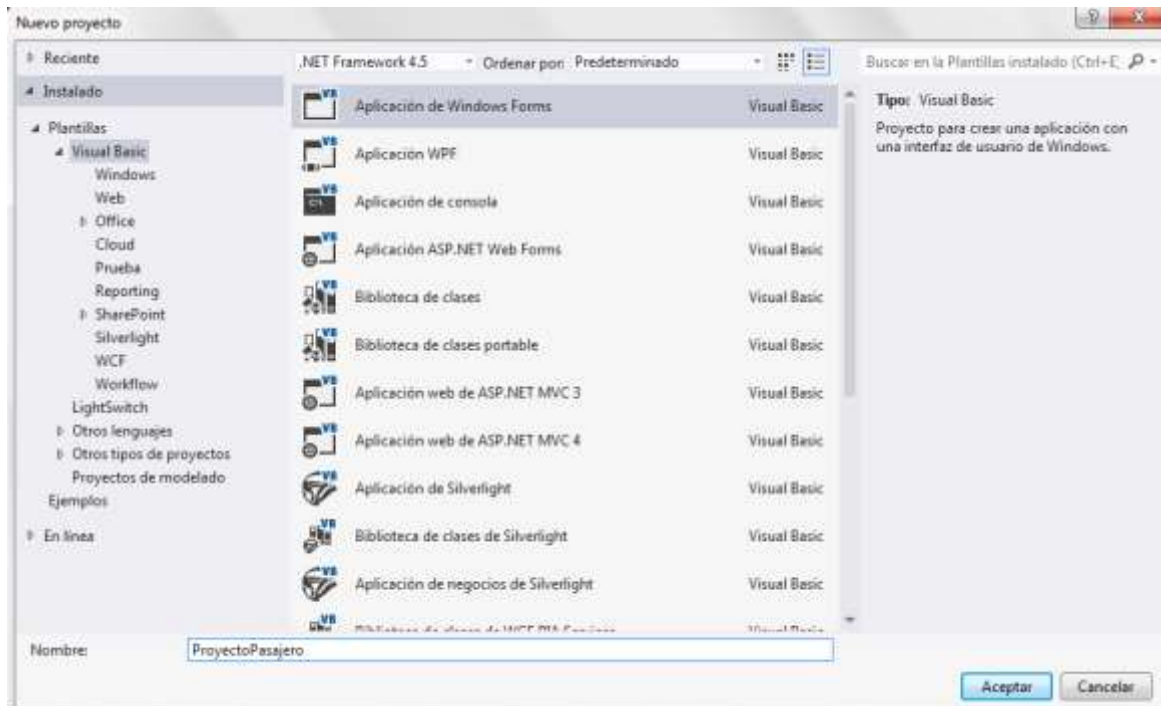
IF OBJECT_ID('ELIMINAPASAJERO') IS NOT NULL
    DROP PROC ELIMINAPASAJERO
GO

CREATE PROC ELIMINAPASAJERO (@IDPAS CHAR(5))
AS
    DELETE PASAJERO
    WHERE IDPASAJERO=@IDPAS
GO
```

Fuente: Autores del proyecto

3. Luego de haber creado la base de datos y los procedimientos almacenados, se da inicio al desarrollo de la aplicación creando un nuevo proyecto.

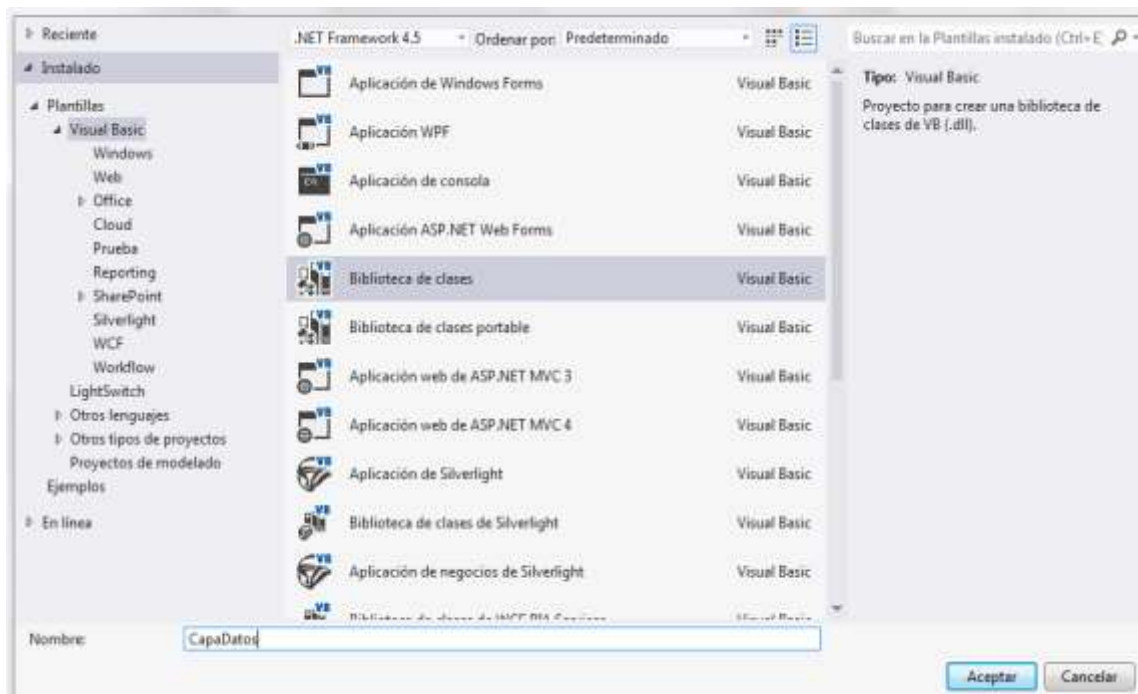
Figura. Creación de un nuevo proyecto de Windows forms.



Fuente: Autores del proyecto

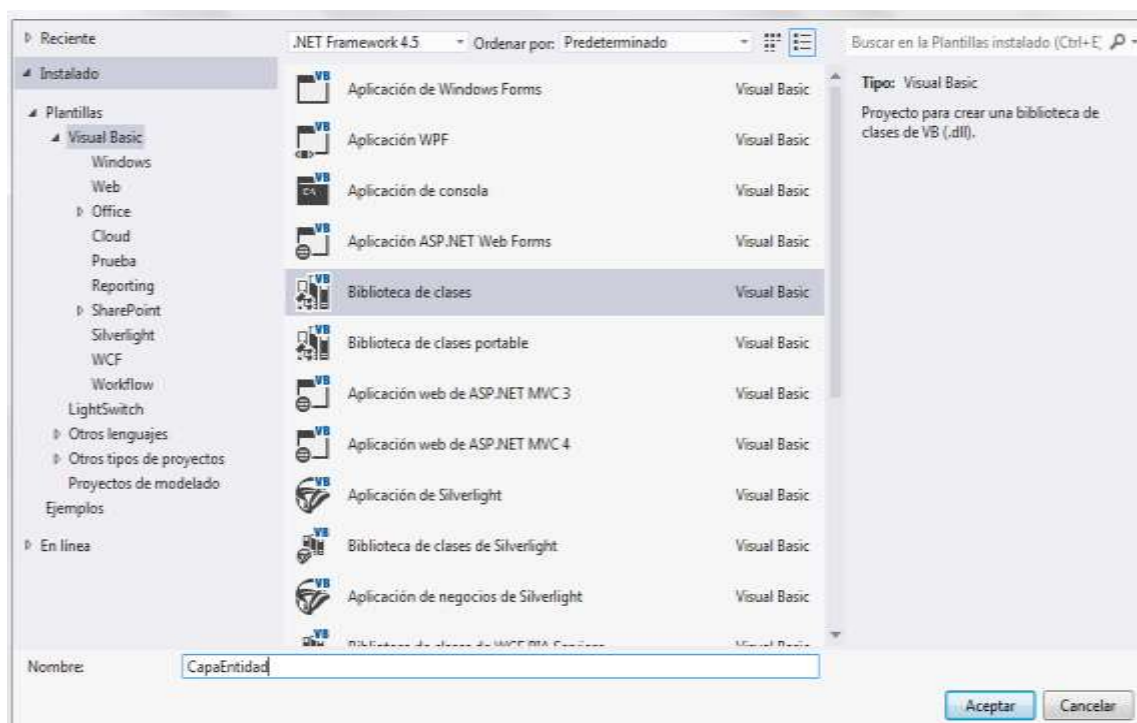
4. Luego de crear el proyecto se debe crear un nuevo proyecto dentro de este, pero ya no se seleccionara proyecto de Windows forms, sino biblioteca de clases y se realiza tres veces este mismo procedimiento, en el cual se están creando las capas que forman parte del proyecto, pero son creadas como nuevos proyectos para que queden como independientes del proyecto general.

Figura. Creación de la Biblioteca de clases para las capas.



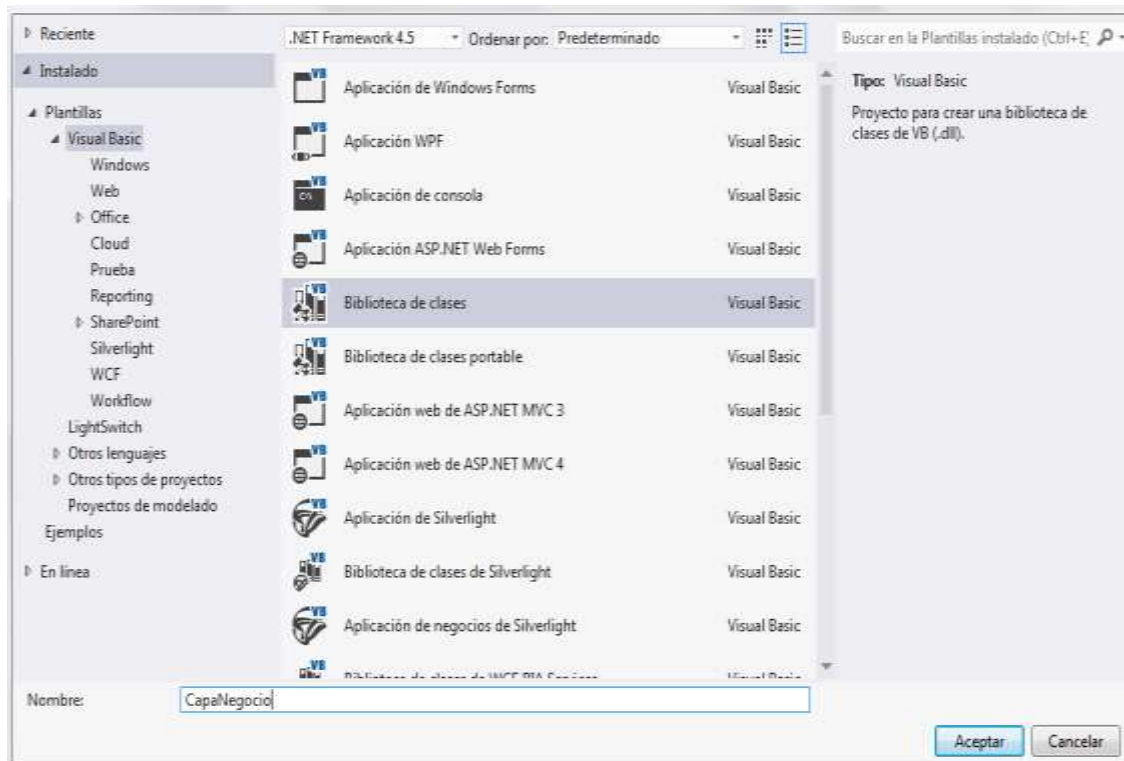
Fuente: Autores del proyecto

Figura. Creación de la Biblioteca de clases para las capas.



Fuente: Autores del proyecto

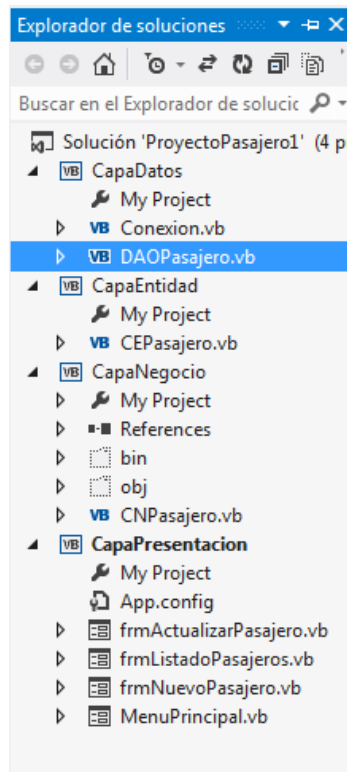
Figura. Creación de la Biblioteca de clases para las capas.



Fuente: Autores del proyecto

5. Luego de crear las bibliotecas de clases con sus respectivos nombres, se toma la clase donde se encuentra el forms y se cambia el nombre por capa de presentación, en la capa de datos se crea una clase adicional que se llamara conexión la cual se utilizara para realizar la conexión con la base de datos quedando el explorador de soluciones de la forma como lo muestra la imagen

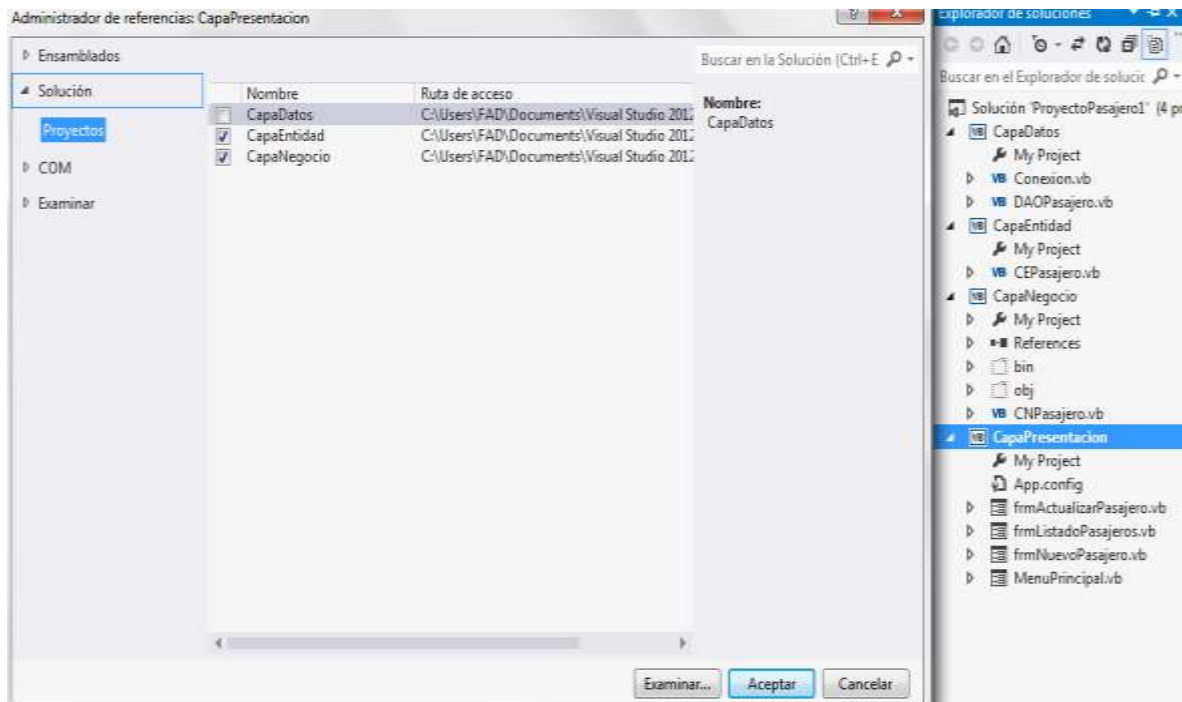
Figura. Visualización del explorador de soluciones del proyecto



Fuente: Autores del proyecto

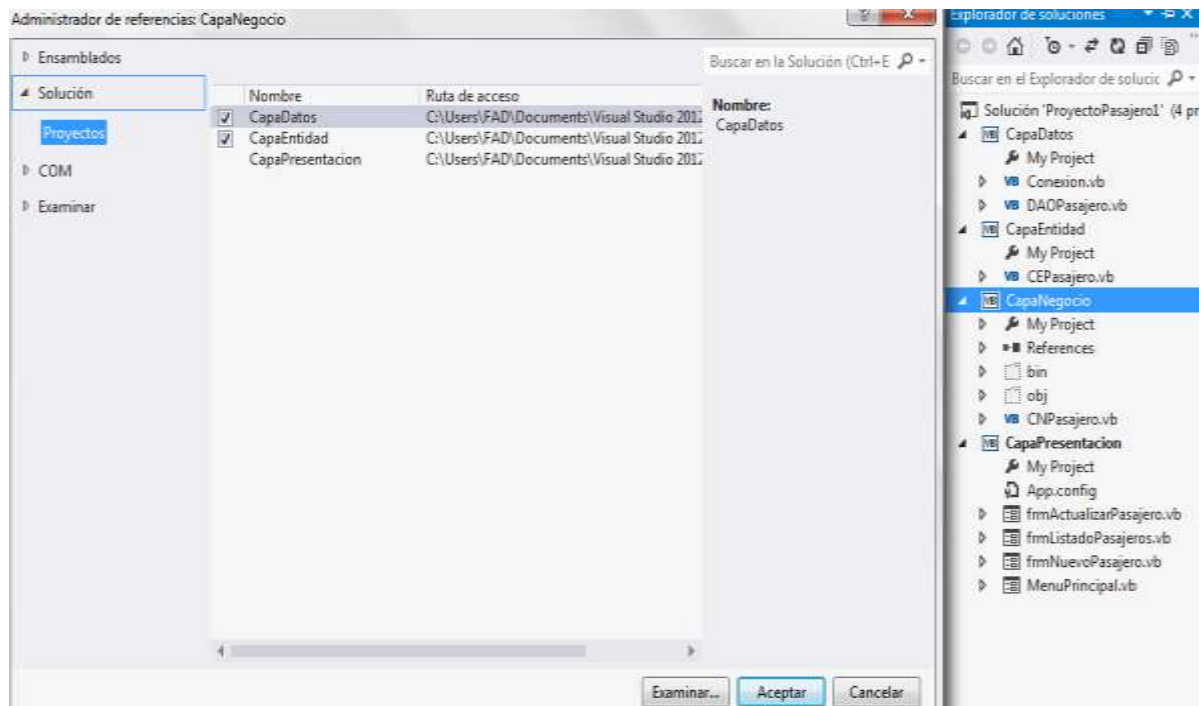
6. Luego a esto se crean las referencias de las capas para que sea posible la comunicación entre ellas. De la capa de presentación se hace referencia con la capa de negocio y la capa entidad. Dela capa de negocio se referencia la capa de datos y la capa entidad. Dela capa datos se referencia solo la capa entidad como lo muestra las imágenes.

Figura. Creación de las referencias entre capas



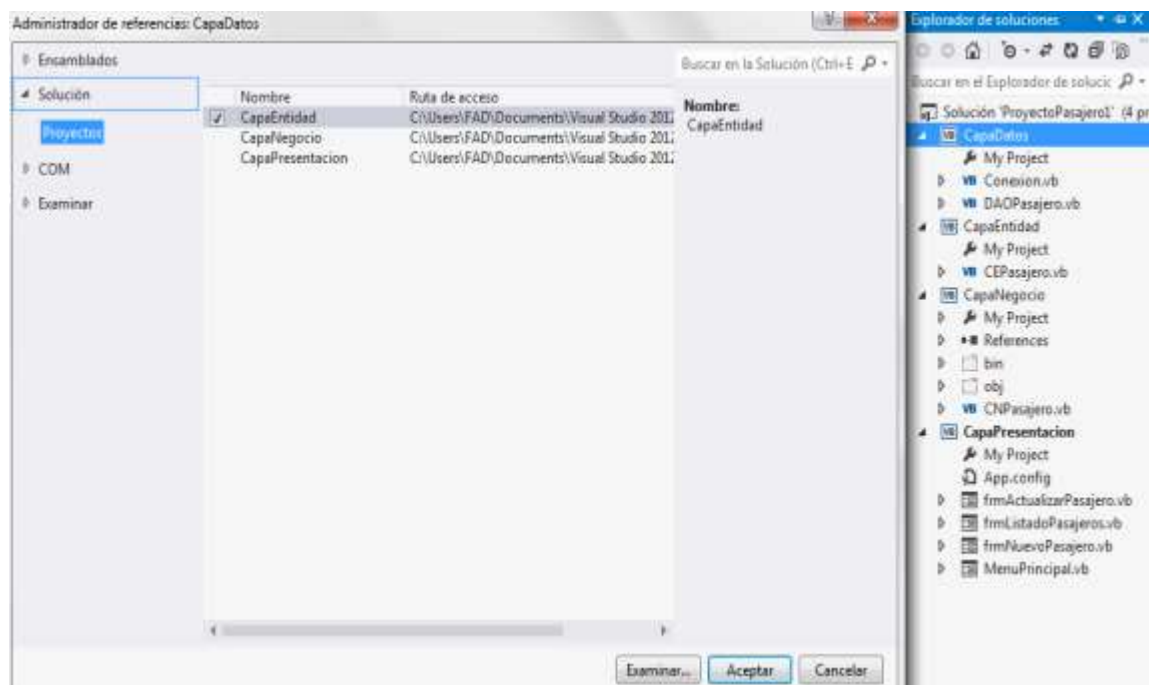
Fuente: Autores del proyecto

Figura. Creación de las referencias entre capas



Fuente: Autores del proyecto

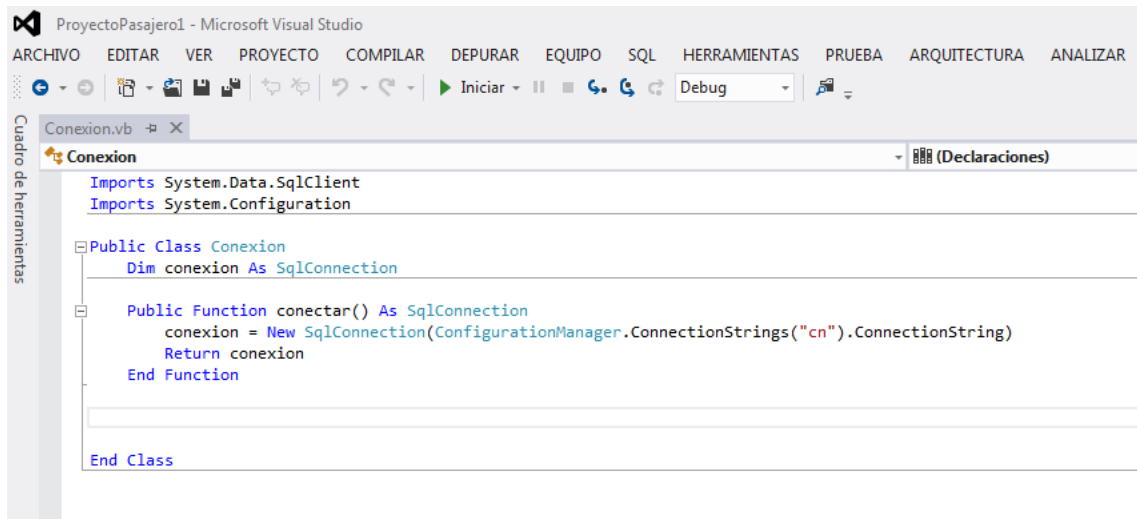
Figura. Creación de las referencias entre capas.



Fuente: Autores del proyecto

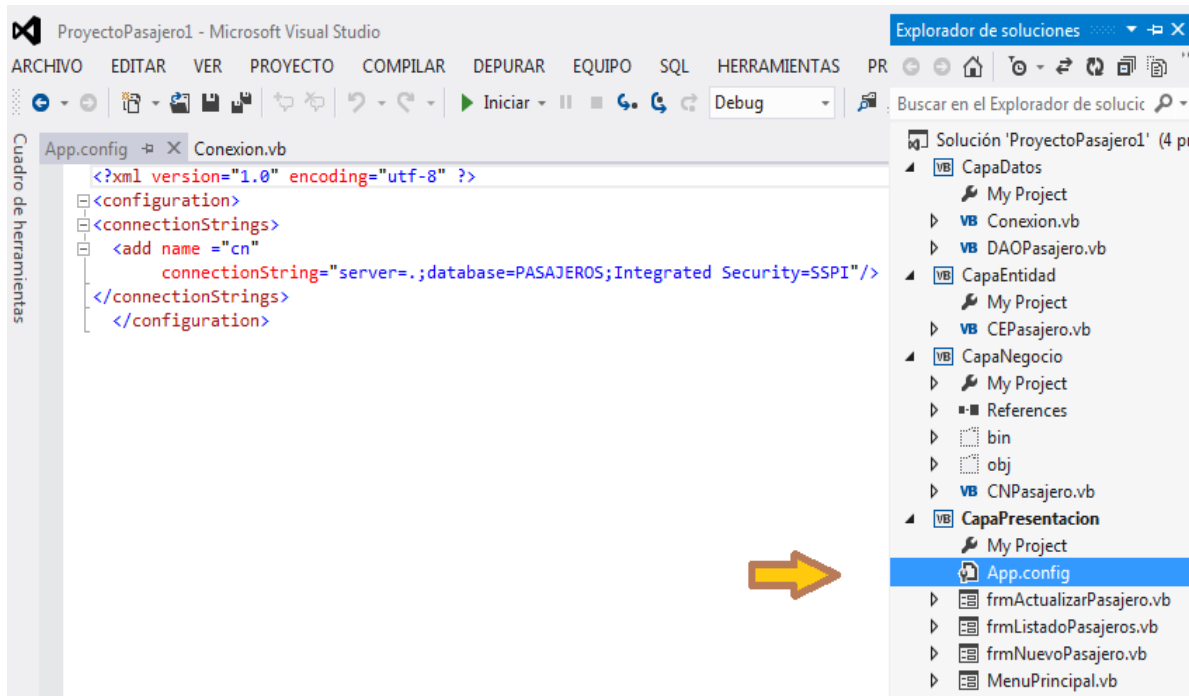
7. Se prosigue creando la conexión con la base de datos en la capa datos, dentro de la clase conexión, además a esto creamos la sesión de inicio con la base de datos en el app.config de la capa de presentación.

Figura. Creación de la variable de conexión con la base de datos.



Fuente: Autores del proyecto

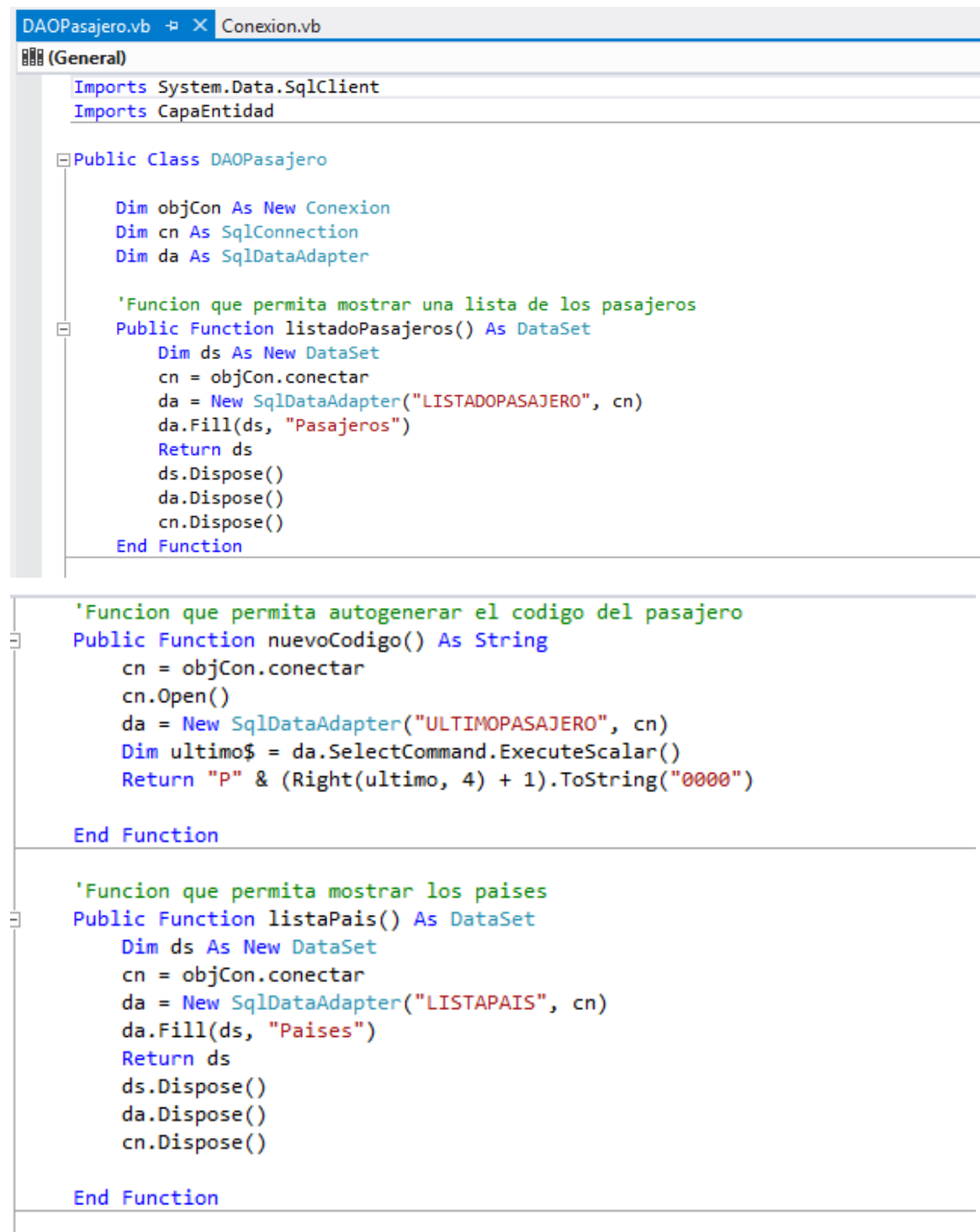
Figura. Creación de acceso e inicio de sesión a la base de datos SQL Server



Fuente: Autores del proyecto

8. Luego de tener la conexión con la base de datos, se prosigue dentro de la capa datos y se crea la clase DAOPasajero que es en la que se hará las funciones y los métodos con los procedimientos almacenados para poder traer y llevar información a la base de datos.

Figura. Creación de las funciones de interactuar con los procedimientos almacenados en la base de datos.



```
Imports System.Data.SqlClient
Imports CapaEntidad

Public Class DAOPasajero

    Dim objCon As New Conexion
    Dim cn As SqlConnection
    Dim da As SqlDataAdapter

    'Funcion que permita mostrar una lista de los pasajeros
    Public Function listadoPasajeros() As DataSet
        Dim ds As New DataSet
        cn = objCon.conectar
        da = New SqlDataAdapter("LISTADOPASAJERO", cn)
        da.Fill(ds, "Pasajeros")
        Return ds
        ds.Dispose()
        da.Dispose()
        cn.Dispose()
    End Function

    'Funcion que permita autogenerar el codigo del pasajero
    Public Function nuevoCodigo() As String
        cn = objCon.conectar
        cn.Open()
        da = New SqlDataAdapter("ULTIMOPASAJERO", cn)
        Dim ultimo$ = da.SelectCommand.ExecuteScalar()
        Return "P" & (Right(ultimo, 4) + 1).ToString("0000")
    End Function

    'Funcion que permita mostrar los paises
    Public Function listaPais() As DataSet
        Dim ds As New DataSet
        cn = objCon.conectar
        da = New SqlDataAdapter("LISTAPAIS", cn)
        da.Fill(ds, "Países")
        Return ds
        ds.Dispose()
        da.Dispose()
        cn.Dispose()
    End Function

End Class
```

Fuente: Autores del proyecto

Figura. Creación de las funciones de interactuar con los procedimientos almacenados en la base de datos.

```
'procedimiento que inserta un nuevo pasajero
Public Sub NuevoPasajero(ByVal objp As CEPasajero)
    cn = objCon.conectar
    Try
        cn.Open()
        da = New SqlDataAdapter("NUEVOPASAJERO", cn)
        da.SelectCommand.CommandType = CommandType.StoredProcedure
        With da.SelectCommand.Parameters
            .Add("@idpas", SqlDbType.Char).Value = objp.idPasajero
            .Add("@nom", SqlDbType.VarChar).Value = objp.nombres
            .Add("@idpai", SqlDbType.Char).Value = objp.idPais
            .Add("@tel", SqlDbType.VarChar).Value = objp.telefono
            .Add("@ema", SqlDbType.VarChar).Value = objp.email
        End With
        da.SelectCommand.ExecuteNonQuery()
        MsgBox("Pasajero registrado correctamente", MsgBoxStyle.Information)
    Catch ex As Exception
        MsgBox("Error al registrar el pasajero", MsgBoxStyle.Critical)
    Finally
        da.Dispose()
        cn.Dispose()
    End Try
End Sub

'Procedimiento que permita actualizar o modificar a un pasajero
Public Sub ActualizaPasajero(ByVal objp As CEPasajero)
    cn = objCon.conectar
    Try
        cn.Open()
        da = New SqlDataAdapter("ACTUALIZAPASAJERO", cn)
        da.SelectCommand.CommandType = CommandType.StoredProcedure
        With da.SelectCommand.Parameters
            .Add("@idpas", SqlDbType.Char).Value = objp.idPasajero
            .Add("@nom", SqlDbType.VarChar).Value = objp.nombres
            .Add("@idpai", SqlDbType.Char).Value = objp.idPais
            .Add("@tel", SqlDbType.VarChar).Value = objp.telefono
            .Add("@ema", SqlDbType.VarChar).Value = objp.email
        End With
        da.SelectCommand.ExecuteNonQuery()
        MsgBox("Pasajero se Modifico correctamente", MsgBoxStyle.Information)
    Catch ex As Exception
        MsgBox("Error al modificar el pasajero", MsgBoxStyle.Critical)
    Finally
        da.Dispose()
        cn.Dispose()
    End Try
End Sub
```

Fuente: Autores del proyecto

Figura. Creación de las funciones de interactuar con los procedimientos almacenados en la base de datos.

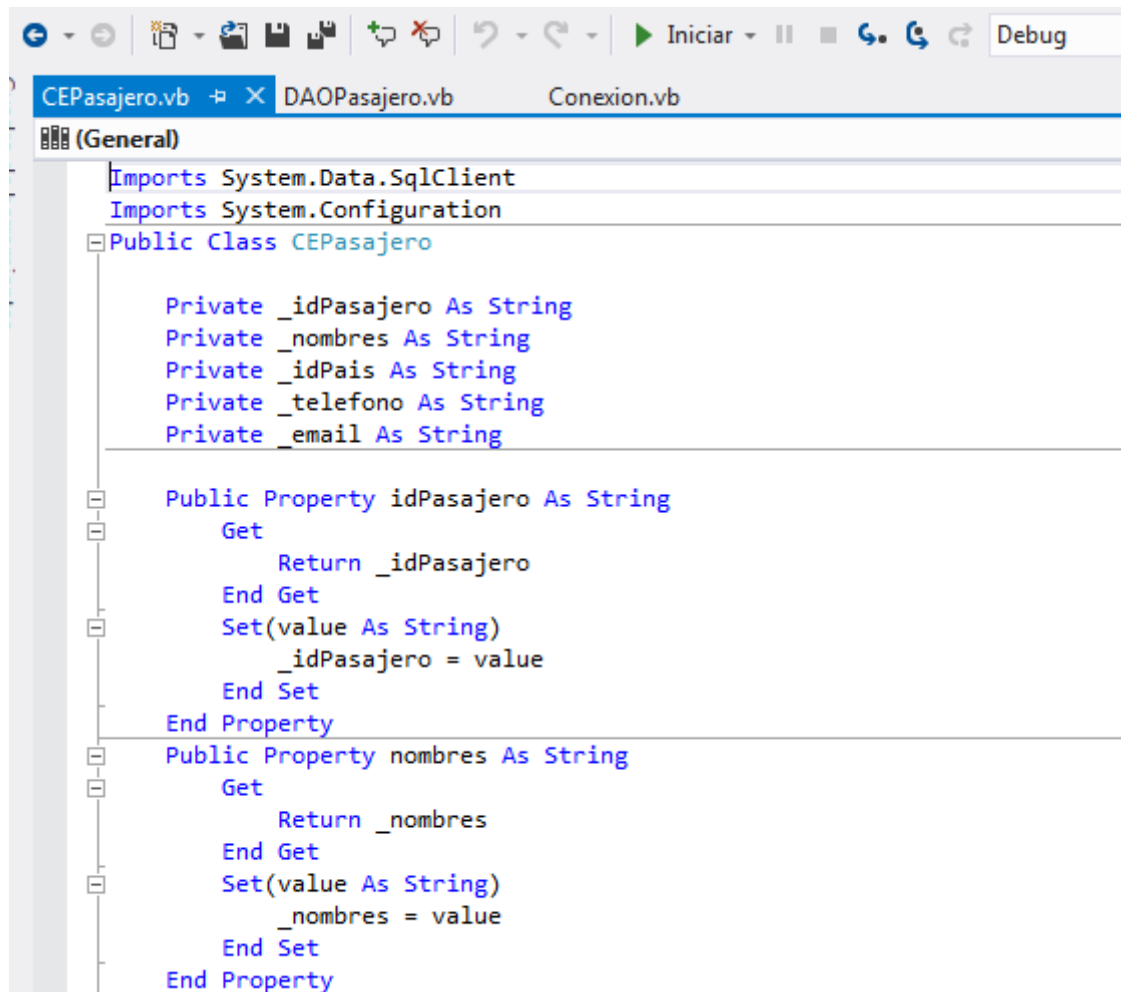
```
'Procedimiento para eliminar a un pasajero
Public Sub EliminaPasajero(ByVal objp As CEPasajero)
    cn = objCon.conectar
    Try
        da = New SqlDataAdapter("ELIMINAPASAJERO", cn)
        da.SelectCommand.CommandType = CommandType.StoredProcedure
        da.SelectCommand.Parameters.Add("@idpas", SqlDbType.Char).Value = objp.idPasajero
        cn.Open()
        da.SelectCommand.ExecuteNonQuery()
        MsgBox("Pasajero se Eliminó correctamente", MsgBoxStyle.Information)
    Catch ex As Exception
        MsgBox("Error al Eliminar el pasajero", MsgBoxStyle.Critical)
    Finally
        da.Dispose()
        cn.Dispose()
    End Try
End Sub

End Class
```

Fuente: Autores del proyecto

9. Luego se procede a crear la clase CEPasajero que es la encargada del procedimiento de retornar los valores mediante las variables definidas en los Get y los Set de la aplicación que capturan y muestran los valores dentro de las variables.

Figura. Creación de los métodos Get y Set



The image shows a screenshot of a Visual Studio code editor window. The title bar at the top displays three open files: 'CEPasajero.vb', 'DAOPasajero.vb', and 'Conexion.vb'. The 'CEPasajero.vb' file is the active document. The editor's toolbar at the top includes icons for navigation, editing, and a 'Debug' button. The code is written in Visual Basic and is displayed in the 'General' tab. It defines a class named 'CEPasajero' with five private fields: '_idPasajero', '_nombres', '_idPais', '_telefono', and '_email', all of type 'String'. Two public properties are implemented: 'idPasajero' and 'nombres'. Each property has a 'Get' method that returns the value of the corresponding private field and a 'Set' method that assigns the value of the parameter to the private field. The code is as follows:

```
Imports System.Data.SqlClient
Imports System.Configuration

Public Class CEPasajero

    Private _idPasajero As String
    Private _nombres As String
    Private _idPais As String
    Private _telefono As String
    Private _email As String

    Public Property idPasajero As String
        Get
            Return _idPasajero
        End Get
        Set(value As String)
            _idPasajero = value
        End Set
    End Property

    Public Property nombres As String
        Get
            Return _nombres
        End Get
        Set(value As String)
            _nombres = value
        End Set
    End Property

End Class
```

Fuente: Autores del proyecto

Figura. Creación de los métodos Get y Set

```
Public Property idPais As String
    Get
        Return _idPais
    End Get
    Set(value As String)
        _idPais = value
    End Set
End Property

Public Property telefono As String
    Get
        Return _telefono
    End Get
    Set(value As String)
        _telefono = value
    End Set
End Property

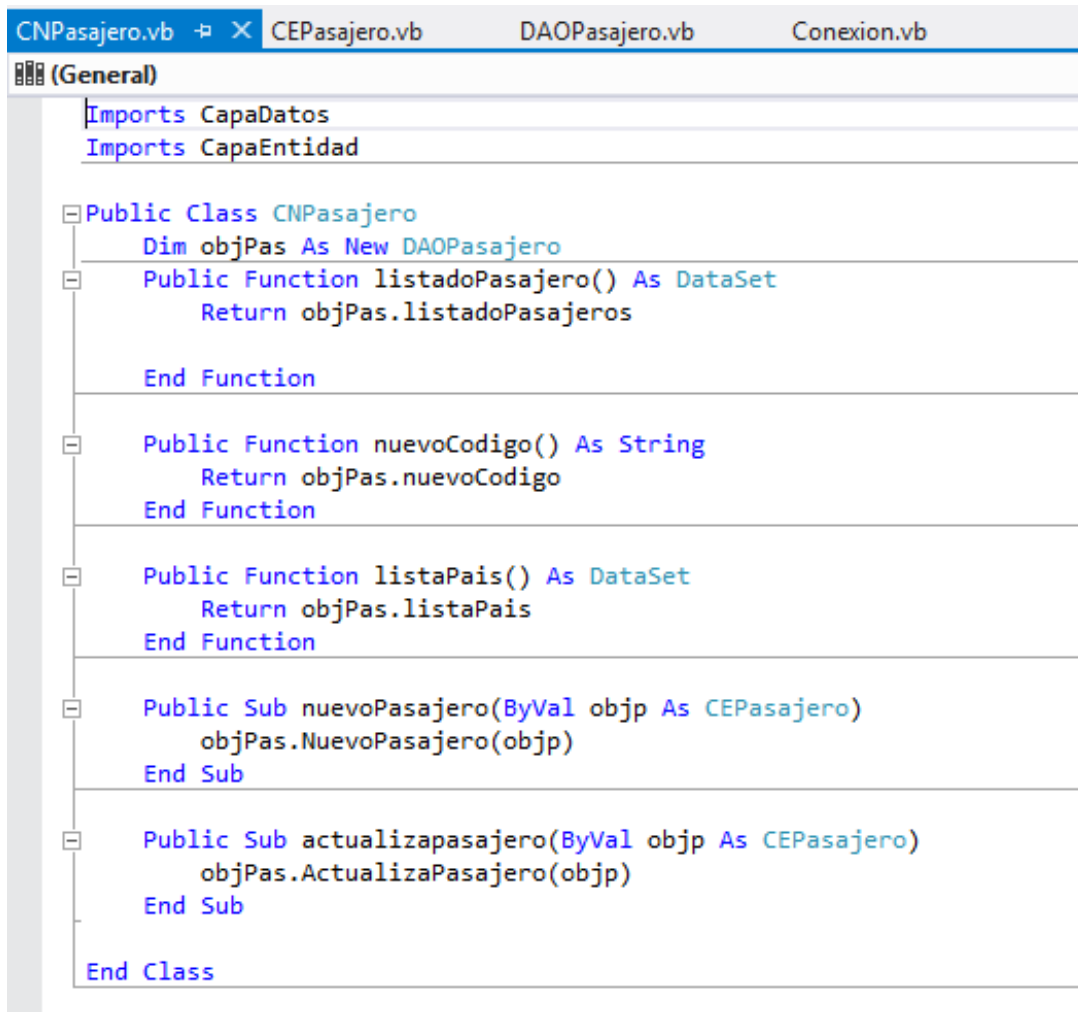
Public Property email As String
    Get
        Return _email
    End Get
    Set(value As String)
        _email = value
    End Set
End Property

End Class
```

Fuente: Autores del proyecto

10. Luego de tener esta capa lista se prosigue con la capa de negocio que es la encargada de hacer el puente de los datos entre la capa de presentación y las capas de datos y entidad.

Figura. Creación de las funciones de transporte de datos mediante objetos



The screenshot displays a Visual Basic code editor with four tabs at the top: CNPasajero.vb, CEPasajero.vb, DAOPasajero.vb, and Conexion.vb. The CNPasajero.vb tab is active, showing the 'General' view. The code defines a class CNPasajero with a private attribute objPas of type DAOPasajero. It includes several public methods: listadoPasajero() returning a DataSet, nuevoCodigo() returning a String, listaPais() returning a DataSet, nuevoPasajero() and actualizapasajero() as public subs that delegate calls to objPas. The code is as follows:

```
Imports CapaDatos
Imports CapaEntidad

Public Class CNPasajero
    Dim objPas As New DAOPasajero

    Public Function listadoPasajero() As DataSet
        Return objPas.listadoPasajeros
    End Function

    Public Function nuevoCodigo() As String
        Return objPas.nuevoCodigo
    End Function

    Public Function listaPais() As DataSet
        Return objPas.listaPais
    End Function

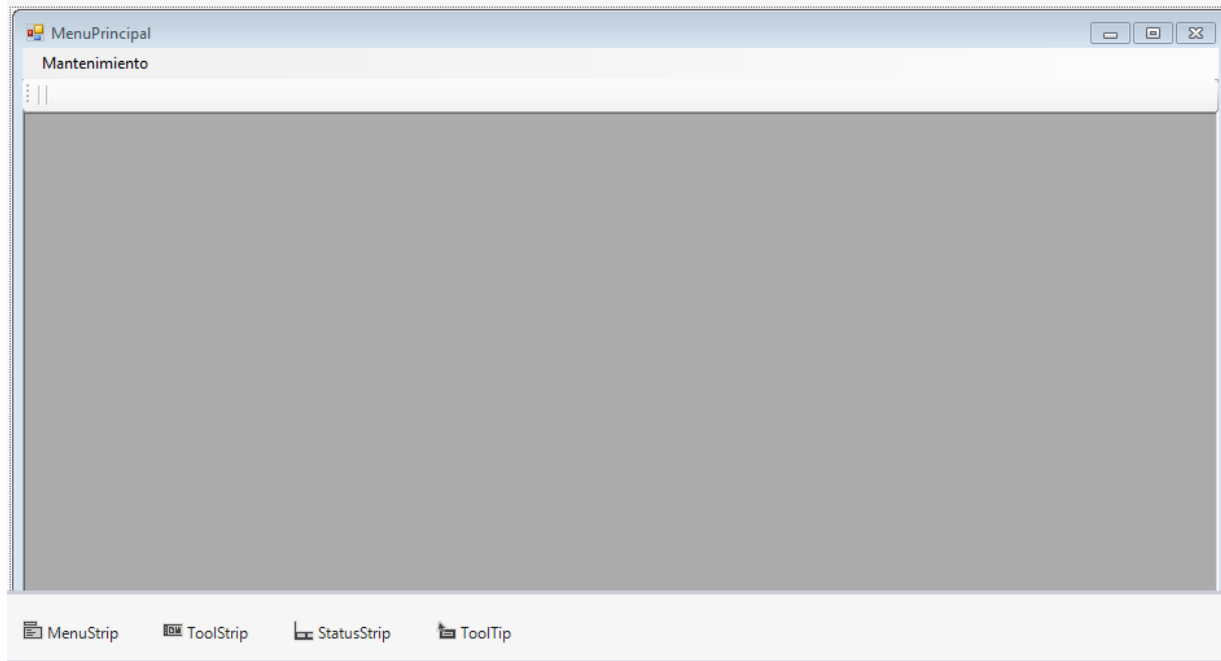
    Public Sub nuevoPasajero(ByVal objp As CEPasajero)
        objPas.NuevoPasajero(objp)
    End Sub

    Public Sub actualizapasajero(ByVal objp As CEPasajero)
        objPas.ActualizaPasajero(objp)
    End Sub
End Class
```

Fuente: Autores del proyecto

11. Por último se maneja la capa de presentación que es la encargada de interactuar con los usuarios finales, ya teniendo los procedimientos definidos, se definen las formas que se manejarán para la recepción y visualización de los datos.

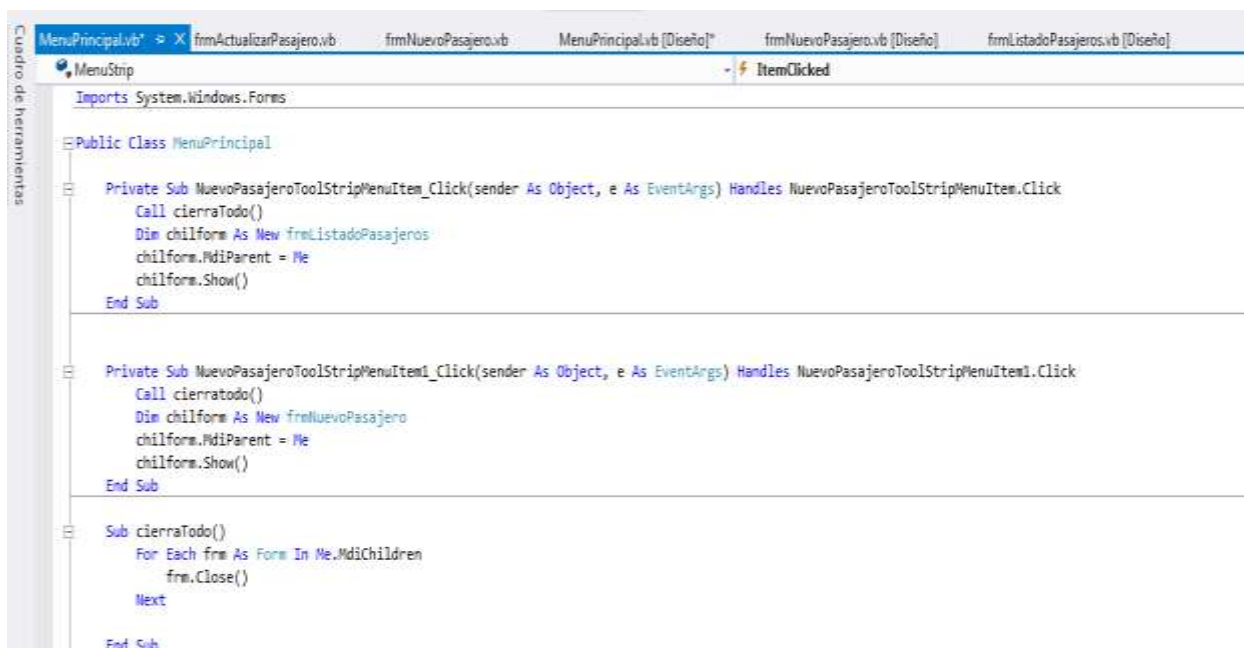
Figura. Creación del formulario de inicio o menús de selección



Fuente: Autores del proyecto

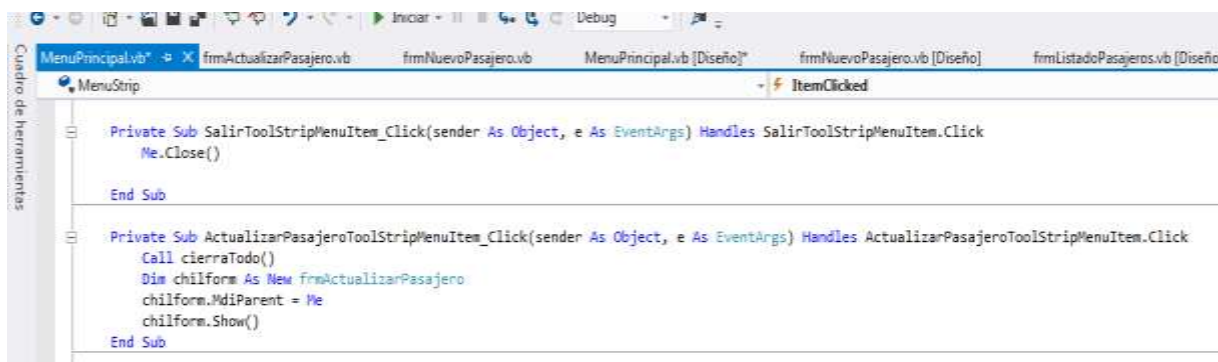
12. En el formulario menú principal se definirán los eventos click siguientes en el toolStrip

Figura. Creación de los eventos click del menú ToolStripMenuItem



Fuente: Autores del proyecto

Figura. Creación de los eventos click del menú ToolStripMenuItem



Fuente: Autores del proyecto

13. En los eventos click del form actualización de los pasajeros se llamarán las funciones de modificar o editar los registros de la grilla.

Figura. Creación del formulario de modificación de información de los pasajeros.

ACTUALIZACIÓN DE DATOS PASAJEROS

CODIGO

NOMBRES

PAIS

TELEFONO

EMAIL

ACTUALIZAR

CANCELAR

Fuente: Autores del proyecto

Figura. Creación de las funciones para cargar datos y modificar el pasajero.

MenuPrincipal.vb* frmActualizarPasajero.vb X MenuPrincipal.vb [Diseño]* frmNuevoPasajero.vb [Diseño] frmLi

btnActualizar Click

```
Imports CapaEntidad
Imports CapaNegocio

Public Class frmActualizarPasajero

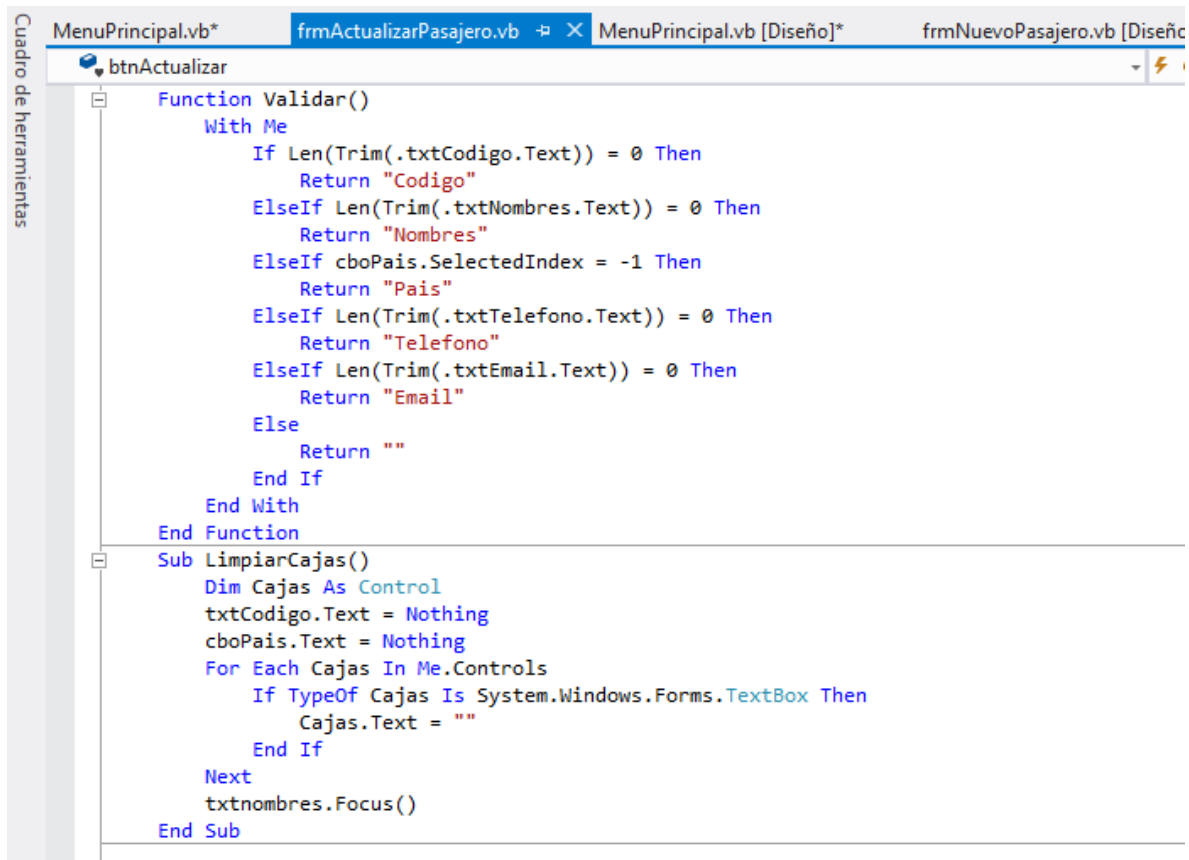
    Dim capanegocios As New CNPasajero

    Private Sub frmActualizarPasajero_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Call CargarPasajeros()
        Call CargarPais()
    End Sub

    Sub CargarPasajeros()
        Dim capanegocios1 As New CNPasajero
        dgPasajeros.DataSource = capanegocios1.listadoPasajero.Tables("Pasajeros")
    End Sub

    Sub CargarPais()
        cboPais.DataSource = capanegocios.listaPais.Tables("países")
        cboPais.DisplayMember = "PAIS"
        cboPais.ValueMember = "IDPAIS"
    End Sub

End Class
```



Fuente: Autores del proyecto

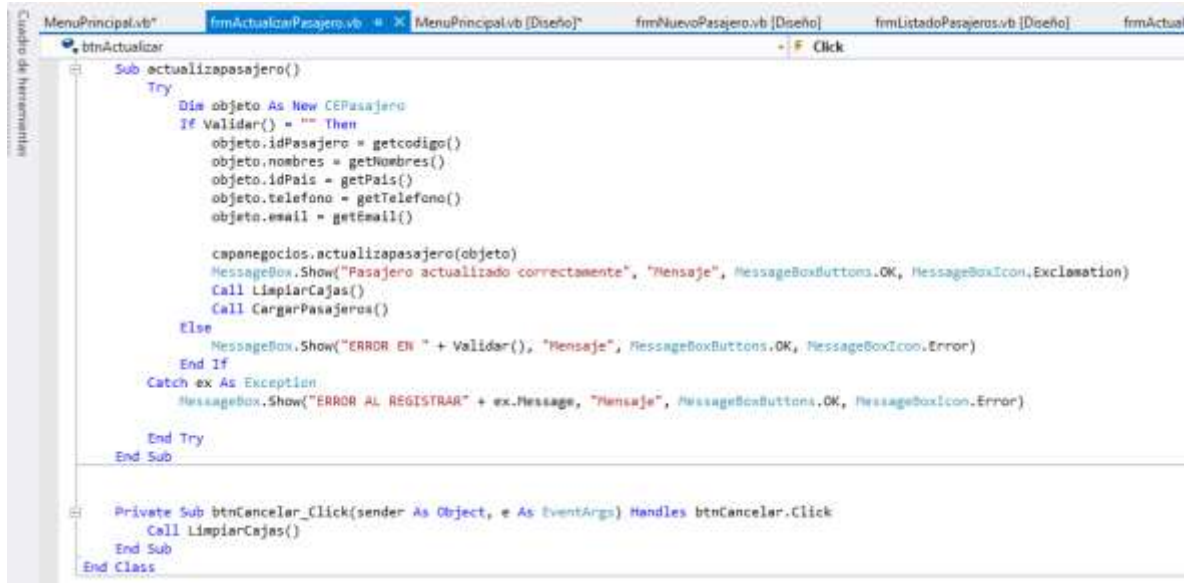
Figura. Creación de las funciones para cargar datos y modificar el pasajero.



```
Function getEmail() As String
    Return txtEmail.Text
End Function
```

```
Private Sub btnActualizar_Click(sender As Object, e As EventArgs) Handles btnActualizar.Click
    Call actualizapasajero()
```

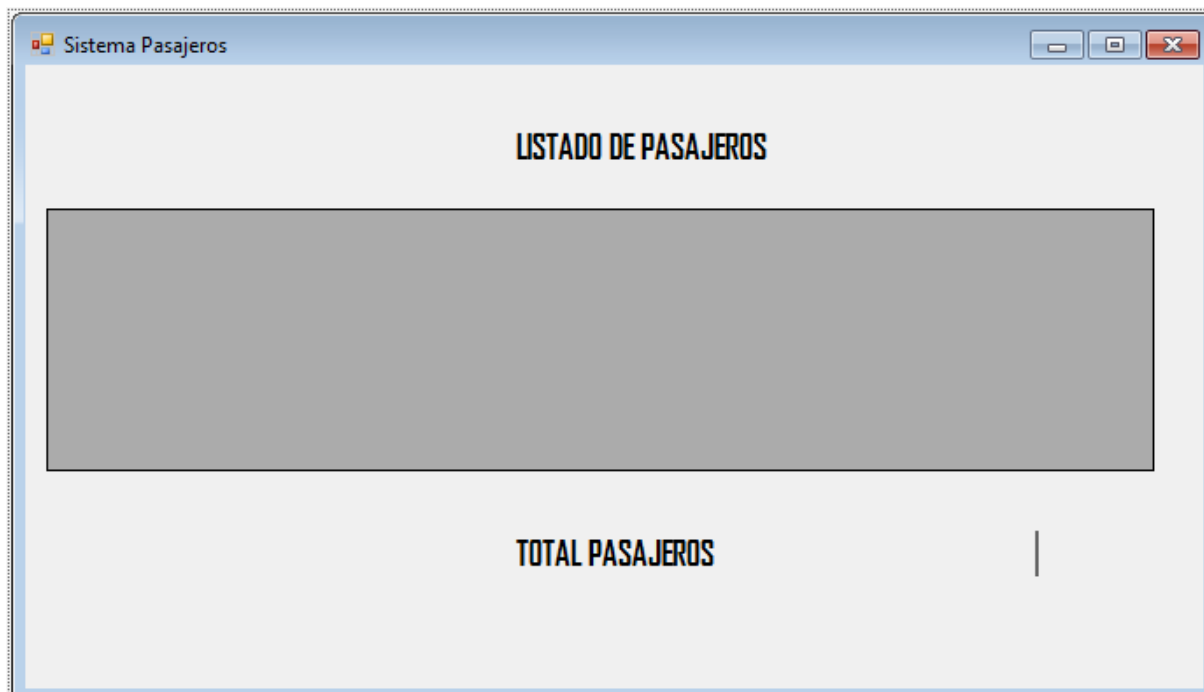
```
End Sub
```



Fuente: Autores del proyecto

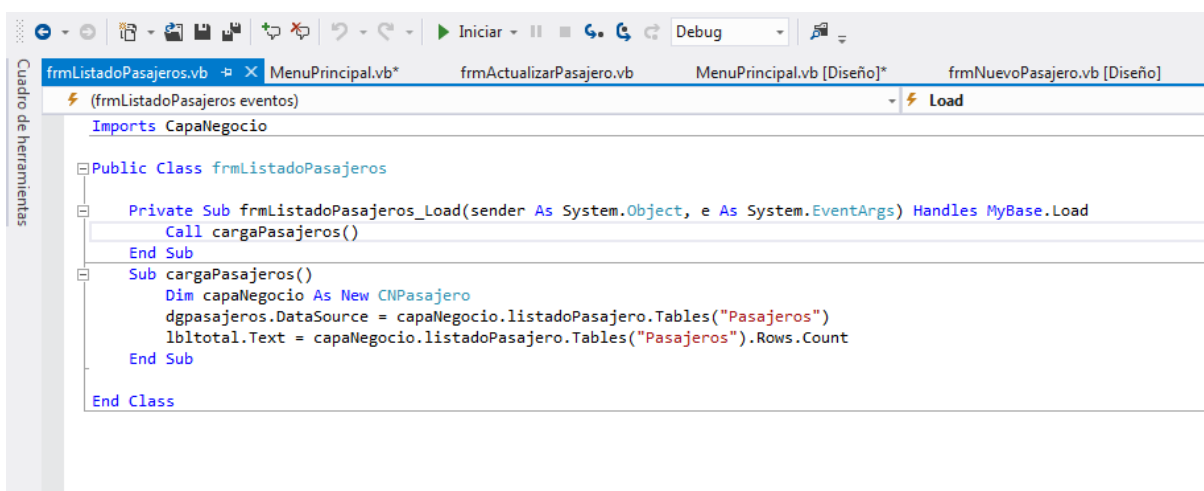
14. En el form del listado de pasajeros en el evento load del form se cargará el listado de los pasajeros ya registrados.

Figura. Visualización de la lista de los pasajeros registrados actualmente.



Fuente: Autores del proyecto

Figura. Llamado de la función cargar pasajeros al evento load del formulario



Fuente: Autores del proyecto

15. En el form ingresar nuevo registro de pasajero en el evento click del botón registrar se ingresarán a la base de datos la información digitada en las cajas de texto el cual generará un código secuencial.

Figura. Creación del formulario nuevo pasajero

The image shows a screenshot of a Windows application window titled "frmNuevoPasajero". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area has a light gray background and is titled "REGISTRAR NUEVO PASAJERO" in bold black text. Below the title, there are five input fields arranged vertically on the left side, each with a label in bold black text: "CODIGO", "NOMBRES", "PAIS", "TELEFONO", and "EMAIL". The "PAIS" field is a dropdown menu. To the right of these fields, there are two buttons: "REGISTRAR" and "LIMPIAR", both in bold black text. The "REGISTRAR" button is positioned above the "LIMPIAR" button.

Fuente: Autores del proyecto

Figura. Creación de las funciones y métodos para envío de información.

```

ARCHIVO  EDITAR  VER  PROYECTO  COMPILAR  DEPURAR  EQUIPO  SQL  HERRAMIENTAS  PRUEBA  ARQUITECTURA  ANALIZAR
[Icons]  Iniciar  [Icons]  Debug
frmListadoPasajeros.vb  MenuPrincipal.vb*  frmActualizarPasajero.vb  MenuPrincipal.vb [Diseño]*  frmNuevoPasajero.vb [Diseño]*

Cuadro de herramientas  btnregistrar Click

Imports CapaNegocio
Imports CapaEntidad

Public Class frmNuevoPasajero
    Dim capaNegocios As New CNPasajero

    Private Sub frmNuevoPasajero_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Call generaCodigo()
        Call cargarPais()
    End Sub

    Sub generaCodigo()
        txtcodigo.Text = capaNegocios.nuevoCodigo
    End Sub

    Sub cargarPais()
        cboPais.DataSource = capaNegocios.listaPais.Tables("países")
        cboPais.DisplayMember = "PAIS"
        cboPais.ValueMember = "IDPAIS"
    End Sub

    'Funciones de capturar valor
    Function getcodigo() As String
        Return txtcodigo.Text
    End Function

    Function getNombres() As String
        Return txtnombres.Text
    End Function

    Function getPais() As String
        Return cboPais.SelectedValue
    End Function

    Function getTelefono() As String
        Return txttelefono.Text
    End Function

    Function getEmail() As String
        Return txtemail.Text
    End Function

    Sub LimpiarCajas()
        Dim Cajas As Control

        For Each Cajas In Me.Controls
            If TypeOf Cajas Is System.Windows.Forms.TextBox Then
                Cajas.Text = ""
            End If
        Next
        txtnombres.Focus()
    End Sub
End Class

```

Fuente: Autores del proyecto

Figura. Creación de las funciones y métodos para envío de información.

```

btnregistrar
Function Validar()
    With Me
        If Len(Trim(.txtcodigo.Text)) = 0 Then
            Return "Codigo"
        ElseIf Len(Trim(.txtnombres.Text)) = 0 Then
            Return "Nombres"
        ElseIf cboPais.SelectedIndex = -1 Then
            Return "Pais"
        ElseIf Len(Trim(.txttelefono.Text)) = 0 Then
            Return "Telefono"
        ElseIf Len(Trim(.txtemail.Text)) = 0 Then
            Return "Email"
        Else
            Return ""
        End If
    End With
End Function

Sub RegistraPasajero()
    Try
        Dim objeto As New CEPasajero
        If Validar() = "" Then
            objeto.idPasajero = getcodigo()
            objeto.nombres = getNombres()
            objeto.idPais = getPais()
            objeto.telefono = getTelefono()
            objeto.email = getEmail()

            capaNegocios.nuevoPasajero(objeto)
            Call LimpiarCajas()
            Call generaCodigo()
        Else
            MessageBox.Show("ERROR EN " + Validar(), "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End If
    Catch ex As Exception
        MessageBox.Show("ERROR AL REGISTRAR" + ex.Message, "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub

Private Sub btnRegistrar_Click(sender As Object, e As EventArgs) Handles btnregistrar.Click
    Call RegistraPasajero()
End Sub

Private Sub btnlimpiar_Click(sender As Object, e As EventArgs) Handles btnlimpiar.Click
    Call LimpiarCajas()
    Call generaCodigo()
End Sub
End Class

```

Fuente: Autores del proyecto

16. Ejecución de la solución.

Figura. Visualización de la aplicación en ejecución.

MenuPrincipal

Mantenimiento

frmActualizarPasajero

ACTUALIZACIÓN DE DATOS PASAJEROS

	CODIGO	PASAJERO	PAIS	TELEFONO	FONO
	0001	MIGUEL MORAL...	COLOMBIA	6303077	MIGUEL@GMAI...
	0002	CARLOS VALDE...	COLOMBIA	6453077	ELPIBE@GMAIL...
▶	P0003	RICARDO CAST...	BRASIL	9800205240	RICARDO@GM...
	P0004	MELISSA SUAR...	EEUU	6523789	MELISSA@GMA...
	P0005	CAMILO ESPAR...	MEXICO	3184356790	CAMILOE@GMA...
	P0006	JUAN PABLO A...	PANAMA	319872345	JUANPABLOARI...
	P0007	CARLOS	PANAMA	310234566	SADFSDFDS@G...
	P0008	PEDRO CRISTA...	CANADA	321207897	PEDROCRIS@G...

CODIGO P0003

NOMBRES RICARDO CASTRO

PAIS BRASIL

TELEFONO 9800205240

EMAIL RICARDO@GMAIL.COM

ACTUALIZAR

CANCELAR

Estado

Fuente: Autores del proyecto