

**Ingeniería en Tecnologías de la Información y Comunicación**

**Nombre de los alumnos:**

María Guadalupe Aguilar Lara

Miguel de Jesús Cisneros Pallares

Anahis Rangel Maciel

**Nombre del Maestro:**

Ariana Gómez Contreras

**Materia**:

Programación de aplicaciones

**Fecha de entrega:**

23/08/2019

**El proyecto creado es una aplicación para calcular el área y perímetro de 3 figuras que son: Rectangulo, Cuadrado y Rombo.**

Esta realizado con el patrón de diseño Modelo, Vista, Controlador; un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos (Modelo, Vista y Controlador).

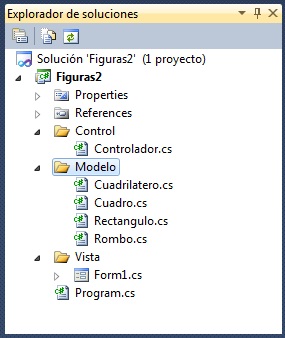
* El Modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio.
* La Vista es la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí.
* El controlador es el responsable de recibir los eventos de entrada desde la Vista.

Este patrón de diseño tiene como principales ventajas:

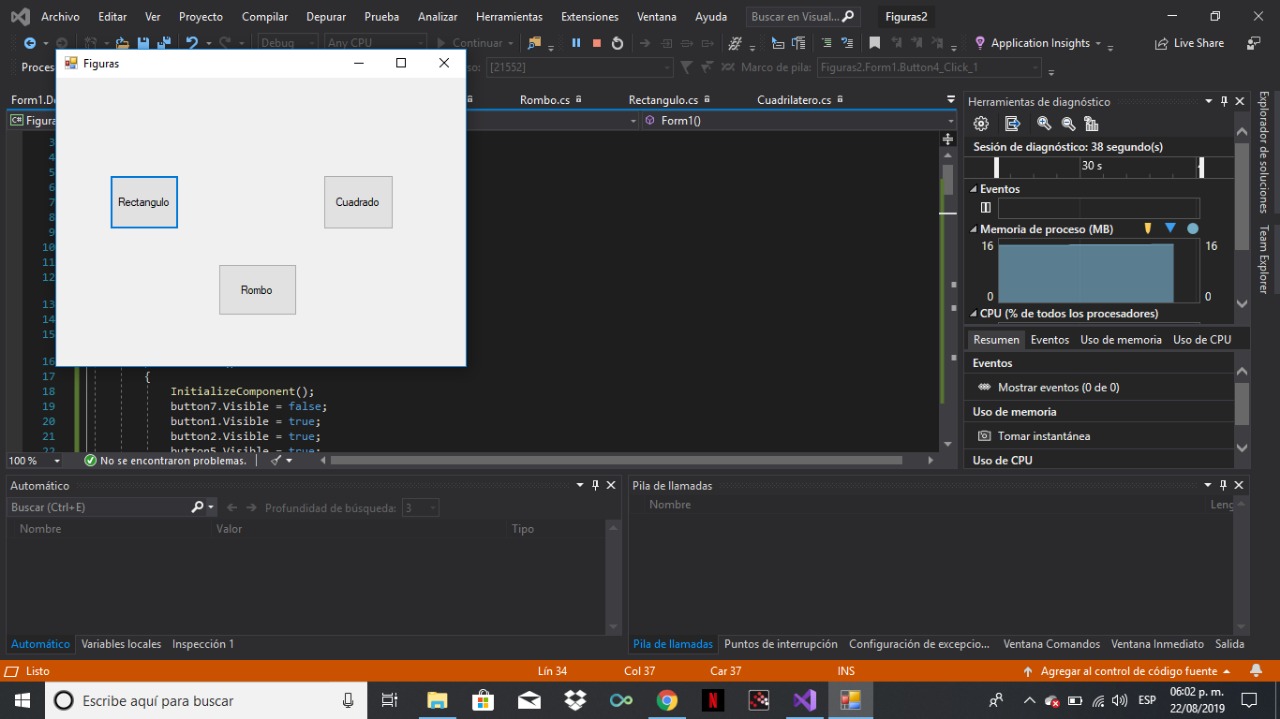
* La implementación se realiza de forma modular.
* Sus vistas muestran información actualizada siempre.
* El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
* Cualquier modificación que afecte al dominio, como aumentar métodos o datos contenidos, implica una modificación sólo en el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.
* Las modificaciones a las vistas no afectan al modelo de dominio, simplemente se modifica la representación de la información, no su tratamiento.

La distribución del Modelo, Vista y Controlador se muestra a continuación:

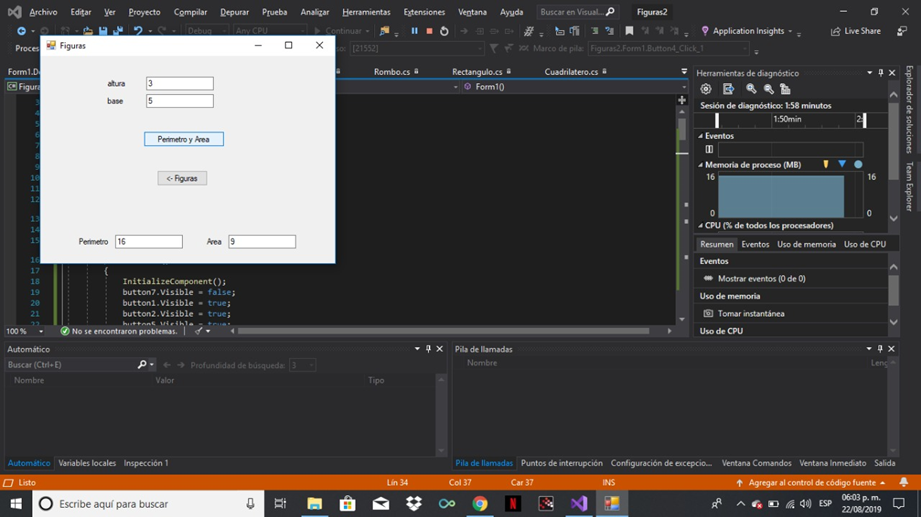
Como podemos observar tenemos separadas nuestras clases en carpetas: En el control tenemos la clase controlador, es que nos dará los permisos necesarios para llamar a las clases de modelos, en el modelo tenemos un ejemplo de herencia con cuatro clases, la clase se llama Cuadrilátero, donde contiene subclases que son Rectángulo y Rombo, la clase Cuadro es una subclase de Rectángulo y en la vista tenemos los 7 botones de cada interfaz, 4 label y 4 textbox con sus respectivos nombres.



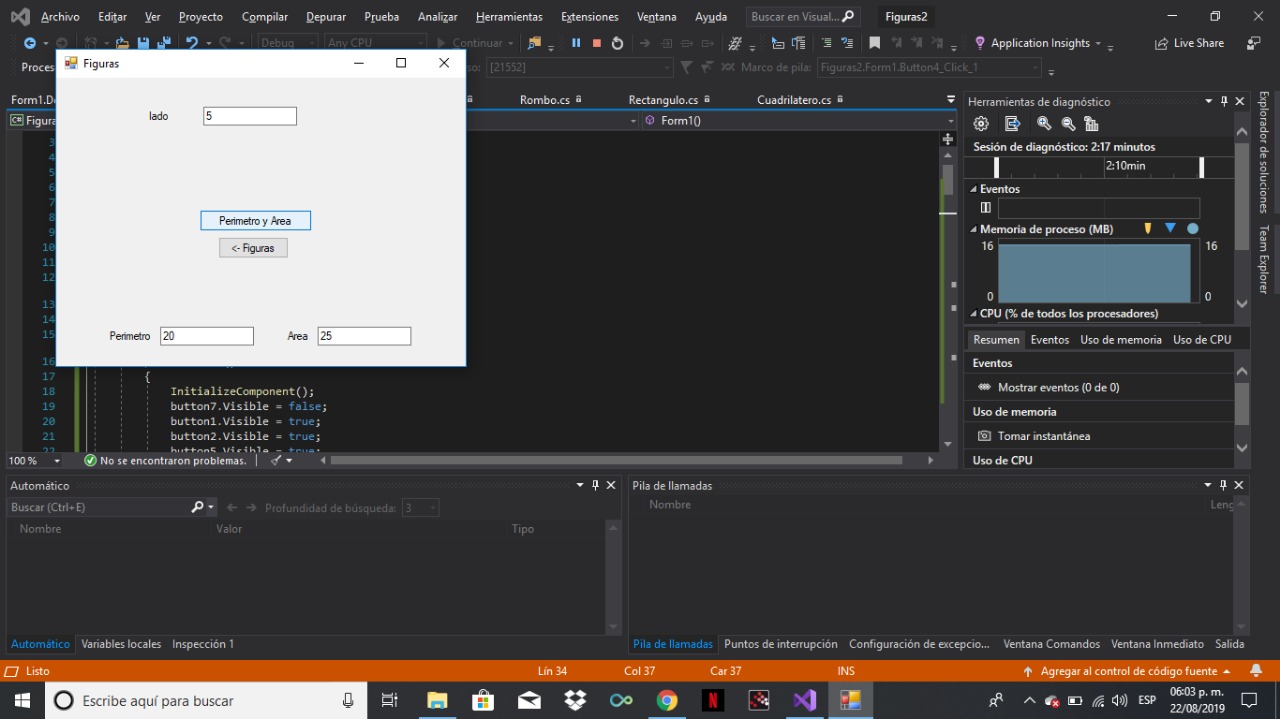
La estructura de la aplicación es la siguiente, la vista principal que contiene 3 recuadros cada uno de ellos con su respectivo nombre de cada figura:



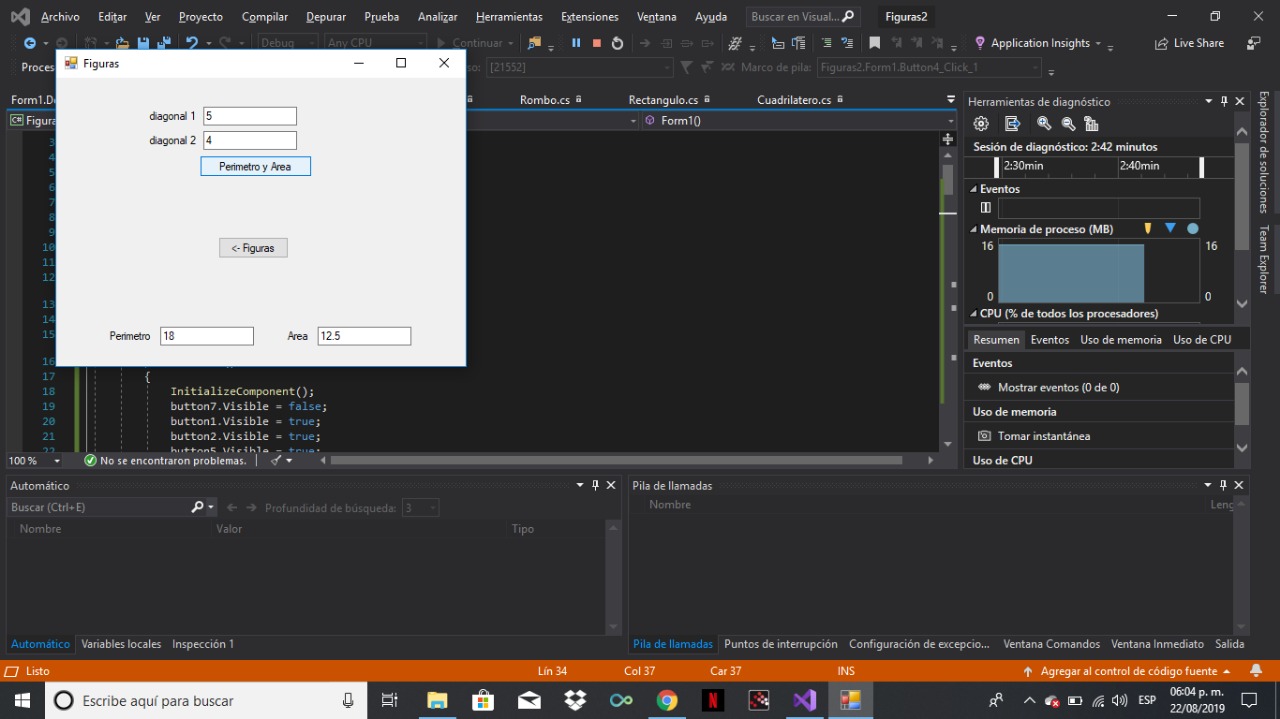
La siguiente figura muestra el cálculo del área y perímetro del rectángulo.



La siguiente figura muestra el cálculo del área y perímetro del cuadrado.



La siguiente figura muestra el cálculo del área y perímetro del rombo.



**Código utilizado:**

Código Controlador

class Controlador

{

        static Controlador control;

        Cuadrilatero cua;

        private Controlador()

        {

        }

        public bool CrearRectangulo(double a,double b) {

            cua = new Rectangulo(a, b); // instanciamos la clase Rectangulo con 2 parámetros

            return true;

        }

         public bool CrearCuadra(double a)

        {

            cua = new Cuadro(a); // instanciamos la clase Cuadro con 1 parámetros

            return true;

        }

         public bool crearRombo(double a, double b)

         {

             cua = new Rombo(a, b); // instanciamos la clase Rombo con 2 parámetros

             return true;

         }

        public double ConsultarPerimetro()

        {

            return cua.calculoPerimetro();

        }

        public double ConsultarArea()

        {

            return cua.calculoArea();

        }

        public static Controlador obtenerControl()

        {

            if (control == null) // aqui verificamos si existe un controlador iniciado

                control = new Controlador();// si no existe, crearemos uno nuevo

            return control;

        }

    }

Código Modelo

Código de clase cuadrilátero

namespace Figuras2.Modelo

{

    public abstract class Cuadrilatero

    {

        protected double lado1, lado2, lado3, lado4;

        public Cuadrilatero(double lado1, double lado2, double lado3, double lado4)

        {

            this.lado1 = lado1;

            this.lado2 = lado2;

            this.lado3 = lado3;

            this.lado4 = lado4;

        }

        public virtual double calculoPerimetro()

        {

            return lado1 + lado2 + lado3 + lado4;

        }

        public abstract double calculoArea();

    }

}

Código de clase rectángulo que hereda la clase cuadrilátero

namespace Figuras2.Modelo

{

    public class Rectangulo : Cuadrilatero

    {

        public Rectangulo(double ancho, double altura)

            : base(ancho, ancho, altura, altura)

        {

        }

        public override double calculoArea()

        {

            return lado1 \* lado2;

        }

    }

}

Código de clase rombo que hereda la clase cuadrilátero

namespace Figuras2.Modelo

{

    class Rombo : Cuadrilatero

    {

        public Rombo(double diag1, double diag2)

            : base (diag1,diag1,diag2,diag2)

        {

        }

        public override double calculoArea()

        {

            return (lado1 \* lado2) / 2;

        }

      }

}

Código de la clase Cuadro que hereda de la clase Rectángulo

namespace Figuras2.Modelo

{

    class Cuadro : Rectangulo

    {

        public Cuadro(double lado)

            : base(lado, lado)

        {

        }

    }

}

Código Vista

namespace Figuras2

{

    public partial class Form1 : Form

    {

        public Form1()

        {

            InitializeComponent();

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            button1.Visible = false;

            button2.Visible = false;

            button5.Visible = false;

            label1.Visible = true;

            label1.Text = "altura";

            label2.Visible = true;

            label2.Text = "base";

            textBox1.Visible = true;

            textBox2.Visible = true;

            button3.Visible = true;

            button7.Visible = true;

            textBox1.Text = "";

            textBox2.Text = "";

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            label3.Visible = true;

            textBox3.Visible = true;

            label4.Visible = true;

            textBox4.Visible = true;

            int a=0,b=0;

            a = Convert.ToInt32(textBox1.Text);

            b = Convert.ToInt32(textBox2.Text);

            Control.Controlador.obtenerControl().CrearRectangulo(a, b);// enviamos los parametros

            textBox3.Text =Convert.ToString(Control.Controlador.obtenerControl().ConsultarPerimetro());

            textBox4.Text =Convert.ToString(Control.Controlador.obtenerControl().ConsultarArea());

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            button1.Visible = false;

            button2.Visible = false;

            button5.Visible = false;

            label1.Visible = true;

            label1.Text = "lado";

            textBox1.Visible = true;

            button4.Visible = true;

            button7.Visible = true;

            textBox1.Text = "";

            textBox2.Text = "";

        }

        private void button4\_Click(object sender, EventArgs e)

        {

            label3.Visible = true;

            textBox3.Visible = true;

            label4.Visible = true;

            textBox4.Visible = true;

            int a = 0;

            a = Convert.ToInt32(textBox1.Text);

            Control.Controlador.obtenerControl().CrearCuadra(a);

            textBox3.Text =Convert.ToString(Control.Controlador.obtenerControl().ConsultarPerimetro());

            textBox4.Text =Convert.ToString(Control.Controlador.obtenerControl().ConsultarArea());

        }

        private void button5\_Click(object sender, EventArgs e)

        {

            button1.Visible = false;

            button2.Visible = false;

            button5.Visible = false;

            label1.Visible = true;

            label1.Text = "diagonal 1";

            label2.Visible = true;

            label2.Text = "diagonal 2";

            textBox1.Visible = true;

            textBox2.Visible = true;

            button6.Visible = true;

            button7.Visible = true;

            textBox1.Text = "";

            textBox2.Text = "";

        }

        private void button6\_Click(object sender, EventArgs e)

        {

            label3.Visible = true;

            textBox3.Visible = true;

            label4.Visible = true;

            textBox4.Visible = true;

            int a1 = 0, b1 = 0;

            a1 = Convert.ToInt32(textBox1.Text);

            b1 = Convert.ToInt32(textBox2.Text);

            Control.Controlador.obtenerControl().crearRombo(a1, b1);

            textBox3.Text =Convert.ToString(Control.Controlador.obtenerControl().ConsultarPerimetro());

            textBox4.Text =Convert.ToString(Control.Controlador.obtenerControl().ConsultarArea());

        }

        private void Form1\_Load(object sender, EventArgs e)

        {

            button7.Visible = false;

        }

        private void button7\_Click(object sender, EventArgs e)

        {

            button7.Visible = false;

            button1.Visible = true;

            button2.Visible = true;

            button5.Visible = true;

            label1.Visible = false;

            button6.Visible = false;

            button4.Visible = false;

            button3.Visible = false;

            label2.Visible = false;

            label3.Visible = false;

            label4.Visible = false;

            textBox3.Visible = false;

            textBox4.Visible = false;

            textBox1.Visible = false;

            textBox2.Visible = false;

            button3.Visible = false;

        }

    }

}

**Buenas prácticas y seguridad implementada en la aplicación**

* Creación de clases abstractas, esto nos ayudó a no escribir demasiado código.
* Hacer un buen uso de la nomenclatura, por ejemplo al crear un método si son más de dos palabras, colocar la primera palabra en minúscula y comenzar con la segunda palabra mayúscula.
* Usar un único return por función, que se colocará como última sentencia de la función
* Evitar escribir funciones y procedimientos demasiado largos.
* Colocar la función main en una clase o módulo separado e independiente.
* **No usar la herencia para heredar atributos de una clase.**