**Planeación:**

**Objetivo**

Realizar una aplicación que registre diversos clientes y sus respectivas categorías para logra el control específico de cada uno, ingresando datos personales como correo, teléfono y nombre.

**Alcance**

Se manejar por medio de 2 etapas en la que la primera etapa consiste en realizar un prototipo que realice el registro del cliente y almacenarlo a la base de datos, para ello se diseñara la interfaz gráfica como la codificación del proceso a lograr en esta primera etapa.

La segunda etapa consta en realizar el registro del usuario y las categorías de manera correcta, dónde se diseñara la segunda ventana la cual visualizara el registro de los clientes por categoría.

**Requerimientos**

**-**-no poder eliminar una categoría, si esta relaciona con un cliente

-nombre y apellidos son obligatorios

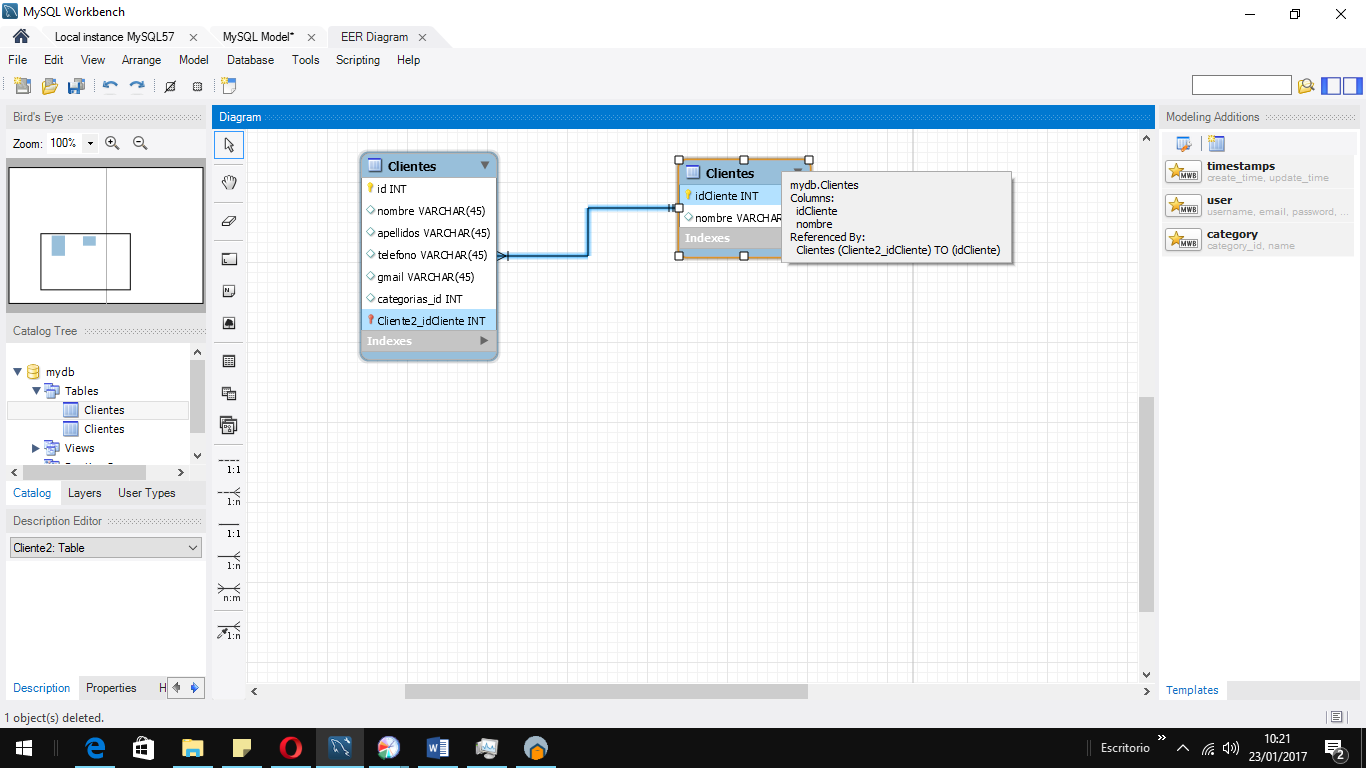
-correo electrónico valido

-los correos electrónicos no podrán repetirse.

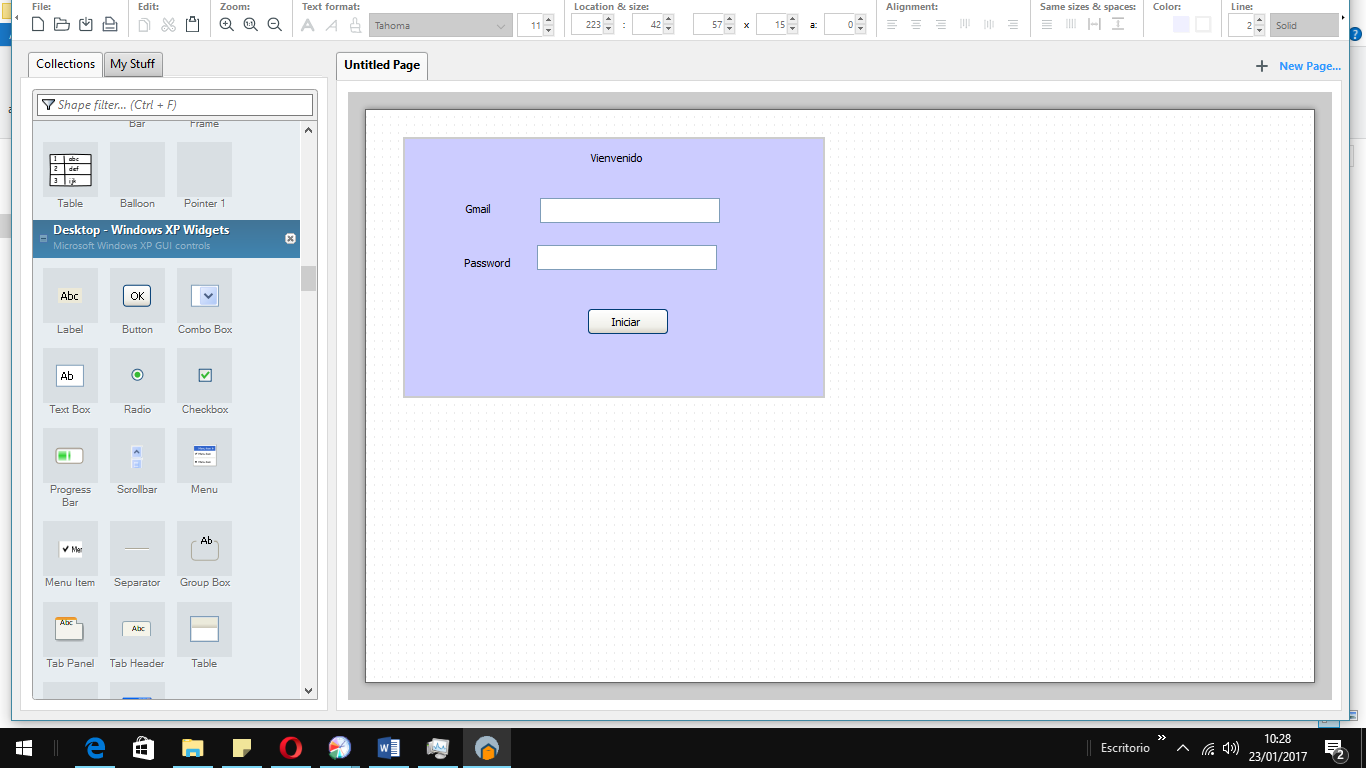
-el nombre de la categoría no podrán repetirse mucho menos quedar vacío.

Diseño:

uml



Inicio de sección:



**Codificación**

**Convención de nombres**

**-Nombre de archivos**

El archivo se nombrara teniendo en cuenta el tipo de proyecto que se desea realizar, asignándole una relación relativa.

**-Nombre de clases**

Las clases se definirá con la primera letra en mayúscula y posteriormente su complementación en minúscula

**-Nombre de funciones/métodos**

El nombre del método se escribirá en minúscula, en caso de que el método este integrado por 2 palabras se escribe la primera letra de la segunda palabra en mayuscala.

1.- Ejemplo de una metodo

clientes()

{

}

2.- Ejemplo de 2 metodo:

*categoriaCliente()*

*{*

*}*

*Los corchetes se escribirá de baja del método para que el programador verifique con facilidad donde termina el método.*

**-Nombre de variables/propiedades**

Las variables se declaran haciendo referencia a cada proceso y función para entender de manera sencillas la explicación que realiza cada variable.

**-Nombre de constante**

Para nombrar un constante se debe tener en cuenta la funcionalidad que se le desea dar al proceso o función, ya que se reflejara de manera fija.

Se declara un constante con mayúsculas para diferenciarla de las variables.

**-Nombre de tablas**

El nombre de las tablas se escribirá timando en cuenta el proceso que cumplirá el proyecto para colocar cada elemento en su campo correspondiente.

Es preferible escribir el nombre de la tabla con la primera letra en mayúscula y su complemento en minúscula.

Es necesario integrarle los guiones bajos”\_\_\_”.

**-Archivos de control:**

**If**

if (condition)

{

Console.WriteLine("The variable is set to true.");

}

else

{

Console.WriteLine("The variable is set to false.");

}

Else

if (condition)

{

then-statement;

}

else

{

else-statement;

}

// Next statement in the program.

// if statement without an else

if (condition)

{

then-statement;

}

Swirch

int caseSwitch = 1;

switch (caseSwitch)

{

case 1:

Console.WriteLine("Case 1");

break;

case 2:

Console.WriteLine("Case 2");

break;

default:

Console.WriteLine("Default case");

break;

}

Goto

for (int i = 0; i < x; i++)

{

for (int j = 0; j < y; j++)

{

if (array[i, j].Equals(myNumber))

{

goto Found;

}

}

}

Console.WriteLine("The number {0} was not found.", myNumber);

goto Finish;

Found:

Console.WriteLine("The number {0} is found.", myNumber);

Finish:

Console.WriteLine("End of search.");

// Keep the console open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

-Estructuras loops/herativas

Do

public class TestDoWhile

{

public static void Main ()

{

int x = 0;

do

{

Console.WriteLine(x);

x++;

} while (x < 5);

}

}

Do while

var s = "";

var i = 0;

do

{

s += i + " ";

i++;

} while (i < 10);

print (s);

// Output: 0 1 2 3 4 5 6 7 8 9

While

class WhileTest

{

static void Main()

{

int n = 1;

while (n < 6)

{

Console.WriteLine("Current value of n is {0}", n);

n++;

}

}

}

/\*

Output:

Current value of n is 1

Current value of n is 2

Current value of n is 3

Current value of n is 4

Current value of n is 5

\*/

For

class ForLoopTest

{

static void Main()

{

for (int i = 1; i <= 5; i++)

{

Console.WriteLine(i);

}

}

}

/\*

Output:

1

2

3

4

5

\*/

For each

class ForEachTest

{

static void Main(string[] args)

{

int[] fibarray = new int[] { 0, 1, 1, 2, 3, 5, 8, 13 };

foreach (int element in fibarray)

{

System.Console.WriteLine(element);

}

System.Console.WriteLine();

// Compare the previous loop to a similar for loop.

for (int i = 0; i < fibarray.Length; i++)

{

System.Console.WriteLine(fibarray[i]);

}

System.Console.WriteLine();

-clases

Clas ecliente:

public class Cliente

{

public int codcliente { get; set; }

// Esto indica que la entidad cliente esta relacionada al entidad Distrito

public Categoria ecategoria { get; set; }

public string nombre { get; set; }

public string apellidos { get; set; }

public string direccion { get; set; }

public string telefono { get; set; }

// Prop. de solo lectura para obtener el nombre de la categoria

public string nombrecategoria

{

get { return ecategoria.nomcategoria; }

}

}

Clase categoría:

public class Categoria

{

// Propiedades de lectura y escritura

public int codcategoria { get; set; }

public string nomcategoria { get; set; }

}

-CLase conexión:

public sealed class conexion

{

public static string LeerCC

{

get {

return "Server=localhost; Port=3306; User Id=root; password=root; Persist Security Info=True; database=app";

}

}

}

cn.Open();

// Ejecutamos el comando y verificamos si el resultado es mayor a cero actualizar, caso contrario insertar

if (Convert.ToInt32(cmd.ExecuteScalar()) > 0)

{

// Si es mayor a cero, quiere decir que existe al menos un registro con los datos ingresados

// Entonces antes de actualizar, hacer las siguientes comprobaciones

if (pEntidad.codcliente == 0)

throw new Exception("El cliente(a) ya esta registrado en el sistema, verifique los datos por favor!...");

// Verifica si ya existe un registro con el mismo nombre de la categoria

cmd.CommandText = @"select ifnull(count(codcliente),0) from clientes where codcliente<>@cod and telefono=@telef;";

if (Convert.ToInt32(cmd.ExecuteScalar()) > 0)

throw new Exception("No se puede grabar un valor duplicado, verifique los datos por favor!...");

// Si las comprobaciones anteriores resultaron ser falsa, entonces actualizar

cmd.CommandText = @"update clientes set codcategoria=@cdis, nombre=@nom,apellidos=@apellido, direccion=@dir, telefono=@telef where codcliente=@cod;";

}

else

cmd.CommandText = @"insert into clientes (codcategoria, nombre, apellidos, direccion, telefono) values (@cdis, @nom, @apellido, @dir, @telef);";

// Ejecutamos el comando que puede ser para update o insert

return Convert.ToBoolean(cmd.ExecuteNonQuery());

}

}