david bacelis ordoñez

cALIDAD EN EL DESARROLLO DEL SOFTWARE  15/02/17

psp0

Contenido

[Planeación. 2](#_Toc474945235)

[Objetivo 2](#_Toc474945236)

[Requisitos 2](#_Toc474945237)

[Diseño 2](#_Toc474945238)

[Codificación. 3](#_Toc474945239)

[Clase: ConexionBD. 3](#_Toc474945240)

[Clase: Cliente. 3](#_Toc474945241)

[Clase: ClientesConexion. 4](#_Toc474945242)

[Clase: Clientes. 6](#_Toc474945243)

[Clase: CategoriaMetodos. 10](#_Toc474945244)

[Clase: CategoriaCone. 10](#_Toc474945245)

[Clase: Categorias. 11](#_Toc474945246)

[Compilación. 13](#_Toc474945247)

[Pruebas. 13](#_Toc474945248)

[PostMorten. 13](#_Toc474945249)

# Planeación.

Objetivo**:** Brindarle al cliente la oportunidad de consultar, actualizar, agregar y/o eliminar la información de los clientes almacenados en la base de datos de su empresa, esto para llevar un control de los clientes que administra dicha empresa.

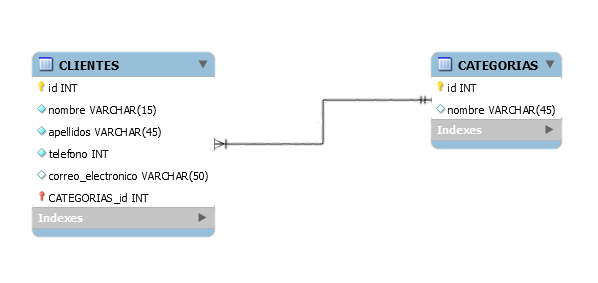
**Alcance:** El sistema abarcara a todo cliente registrado en la aplicación, aunque también incluye a los clientes que adquieran y utilicen el sistema de registro de clientes (alta, baja, consulta, etc…).

EL cliente que adquiere la aplicación, podrá registrar a otros clientes de su empresa, organización o comercio, incrementando el impacto del sistema de registro de clientes.

Requisitos**:**

* No poder eliminar una categoría si está relacionada con un cliente.
* Nombre y apellidos son obligatorios.
* Correo electrónico valido.
* El espacio de categoría no puede quedar vacío.
* No se pueden repetir dos correos iguales.

Diseño.

Modelo relacional Base de Datos Clientes. 

# Codificación.

## Clase: ConexionBD.

Conecta a Visual Studio con el sistema Gestor de Base de Datos MySQL.

namespace MiAppClientes

{

public class ConexionBD

{

public static MySqlConnection ObtenerConexion()

{

MySqlConnection conectar = new MySqlConnection("server=127.0.0.1; database=clientes; Uid=root; pwd=GILGAMESH117;");

conectar.Open();

return conectar;

}

}

}

## Clase: Cliente.

En esta clase se encuentran todos los atributos que componen al cliente en si, relacionado a la información personal del mismo.

class Cliente

{

public int Id { get; set; }

public string Nombre { get; set; }

public string Apellido { get; set; }

public int Telefono { get; set; }

public string Correo { get; set; }

public Cliente() { }

public Cliente(int pId, string pNombre, string pApellido, int pTel, string pCorreo)

{

this.Id = pId;

this.Nombre = pNombre;

this.Apellido = pApellido;

this.Telefono = pTel;

this.Correo =pCorreo;

}

}

# Clase: ClientesConexion.

Esta clase almacena todo lo relacionado a la conexión, consultas, relaciones y acciones a la tabla cliente de la base de datos Cliente.

class ClientesConexion

{

//Info. de la base de datos.

private const String SERVER = "127.0.0.1";

private const String DATABASE = "clientes";

private const String UID = "root";

private const String PASSWORD = "GILGAMESH117";

private static MySqlConnection dbConn;

// Info. del cliente.

public int Id { get; private set; }

public String Nombre { get; private set; }

public String Apellidos { get; private set; }

public int Telefono { get; private set; }

public String Correo { get; private set; }

private ClientesConexion(int id, String N, String A, int T, String C)

{

Id = id;

Nombre = N;

Apellidos = A;

Telefono = T;

Correo = C;

}

public static void InitializeDB()

{

MySqlConnectionStringBuilder builder = new MySqlConnectionStringBuilder();

builder.Server = SERVER;

builder.UserID = UID;

builder.Password = PASSWORD;

builder.Database = DATABASE;

String connString = builder.ToString();

builder = null;

Console.WriteLine(connString);

dbConn = new MySqlConnection(connString);

Application.ApplicationExit += (sender, args) => {

if (dbConn != null)

{

dbConn.Dispose();

dbConn = null;

}

};

}

public static List<ClientesConexion> GetUsers()

{

List<ClientesConexion> client = new List<ClientesConexion>();

String query = "SELECT \* FROM cliente";

MySqlCommand cmd = new MySqlCommand(query, dbConn);

dbConn.Open();

MySqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = (int)reader["id"];

String Nombre = reader["nombre"].ToString();

String Apellidos = reader["apellidos"].ToString();

int Telefono = (int)reader["telefono"];

String Correo = reader["correo\_electronico"].ToString();

ClientesConexion Clie = new ClientesConexion(id, Nombre, Apellidos, Telefono, Correo);

client.Add(Clie);

}

reader.Close();

dbConn.Close();

return client;

}

public static ClientesConexion Insert(String N, String A, int T, String C)

{

String query = string.Format("INSERT INTO cliente(nombre, apellidos, telefono, correo\_electronico, id\_categoria) VALUES ('{0}', '{1}', '{2}', '{3}')", N, A, T, C);

MySqlCommand cmd = new MySqlCommand(query, dbConn);

dbConn.Open();

cmd.ExecuteNonQuery();

int id = (int)cmd.LastInsertedId;

ClientesConexion user = new ClientesConexion(id, N, A, T, C);

dbConn.Close();

return user;

}

public void Update(string N, string A, int T, string C)

{

String query = string.Format("UPDATE cliente SET nombre='{0}', apellidos='{1}', telefono='{2}', correo\_electronico='{3}' WHERE Id={4}", N, A, T, C, Id);

MySqlCommand cmd = new MySqlCommand(query, dbConn);

dbConn.Open();

cmd.ExecuteNonQuery();

dbConn.Close();

}

public void Delete()

{

String query = string.Format("DELETE FROM cliente WHERE ID={0}", Id);

MySqlCommand cmd = new MySqlCommand(query, dbConn);

dbConn.Open();

cmd.ExecuteNonQuery();

dbConn.Close();

}

}

# Clase: Clientes.

Esta clase alberga absolutamente toda la información relacionada a las acciones a realizar en la aplicación para la ventana clientes, toda acción que se pueda realizar con la tabla clientes y sus datos incluidos, como almacenar un nuevo cliente, modificarlo, eliminarlo, etc.

public partial class Clientes : Form

{

CategoriaCone cate = new CategoriaCone();

CategoriaNuevo catenue = new CategoriaNuevo();

private ClientesConexion ClienteActual;

public Clientes()

{

InitializeComponent();

ClientesConexion.InitializeDB();

}

private void LoadAll()

{

List<ClientesConexion> users = ClientesConexion.GetUsers();

lvUsers.Items.Clear();

foreach (ClientesConexion u in users)

{

ListViewItem item = new ListViewItem(new String[] { u.Id.ToString(), u.Nombre, u.Apellidos, u.Telefono.ToString(), u.Correo });

item.Tag = u;

lvUsers.Items.Add(item);

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void txtTelefono\_TextChanged(object sender, EventArgs e)

{

}

private void txtTelefono\_KeyPress(object sender, KeyPressEventArgs e)

{

validar.SoloNumeros(e);

}

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

{

try

{

int catenue = (int)cbxCategoria.SelectedValue;

String N = txtNombre.Text;

String A = txtApellido.Text;

int T = int.Parse(txtTelefono.Text);

String C = txtCorreo.Text;

if (String.IsNullOrEmpty(N) || String.IsNullOrEmpty(A) || String.IsNullOrEmpty(C))

{

MessageBox.Show("Por favor, introduzca valores, los campos no pueden estar vacios.");

return;

}

else

{

MessageBox.Show("Cliente Guardado Con Exito!!", "Guardado", MessageBoxButtons.OK, MessageBoxIcon.Information);

txtNombre.Clear();

txtApellido.Clear();

txtCorreo.Clear();

txtTelefono.Clear();

}

ClienteActual = ClientesConexion.Insert(N, A, T, C);

LoadAll();

}

catch (System.OverflowException)

{

}

}

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

{

{

LoadAll();

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

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

{

String N = txtNombre.Text;

String A = txtApellido.Text;

int T = int.Parse(txtTelefono.Text);

String C = txtCorreo.Text;

if (String.IsNullOrEmpty(N) || String.IsNullOrEmpty(A) || String.IsNullOrEmpty(C))

{

MessageBox.Show("Seleccione un usuario para actualizar sus datos.");

return;

}

else

{

MessageBox.Show("Cliente Modificado Con Exito!!", "Guardado", MessageBoxButtons.OK, MessageBoxIcon.Information);

txtNombre.Clear();

txtApellido.Clear();

txtCorreo.Clear();

txtTelefono.Clear();

}

ClienteActual.Update(N, A, T, C);

LoadAll();

}

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

{

if (ClienteActual == null)

{

MessageBox.Show("Por favor, seleccione un usuario para borrar.");

return;

}

else

{

MessageBox.Show("Cliente Modificado Con Exito!!", "Guardado", MessageBoxButtons.OK, MessageBoxIcon.Information);

txtNombre.Clear();

txtApellido.Clear();

txtCorreo.Clear();

txtTelefono.Clear();

}

ClienteActual.Delete();

LoadAll();

}

private void listView1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvUsers.SelectedItems.Count > 0)

{

ListViewItem item = lvUsers.SelectedItems[0];

ClienteActual = (ClientesConexion)item.Tag;

int id = ClienteActual.Id;

String N = ClienteActual.Nombre;

String A = ClienteActual.Apellidos;

int T = ClienteActual.Telefono;

String C = ClienteActual.Correo;

txtNombre.Text = N;

txtApellido.Text = A;

txtTelefono.Text = T.ToString();

txtCorreo.Text = C;

}

}

private void lvUsers\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvUsers.SelectedItems.Count > 0)

{

ListViewItem item = lvUsers.SelectedItems[0];

ClienteActual = (ClientesConexion)item.Tag;

int id = ClienteActual.Id;

String N = ClienteActual.Nombre;

String A = ClienteActual.Apellidos;

int T = ClienteActual.Telefono;

String C = ClienteActual.Correo;

txtNombre.Text = N;

txtApellido.Text = A;

txtTelefono.Text = T.ToString();

txtCorreo.Text = C;

}

}

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

{

txtNombre.Clear();

txtApellido.Clear();

txtCorreo.Clear();

txtTelefono.Clear();

}

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

{

}

private void cbxCategoria\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

# Clase: CategoriaMetodos.

Todos los atributos y métodos utilizados para elaborar la ventana de categoría.

class CategoriaMetodos

{

public int Id { get; set; }

public string Nombre { get; set; }

public CategoriaMetodos() { }

public CategoriaMetodos(int pId, string pNombre)

{

this.Id = pId;

this.Nombre = pNombre;

}

}

# Clase: CategoriaCone.

Todo relacionado a la conexión a base de datos para realizar las consultas indicadas.

class CategoriaCone

{

public static int Agregar(CategoriaMetodos pCategoria)

{

int retorno = 0;

MySqlCommand comando = new MySqlCommand(string.Format("Insert into categorias (nombre) values ('{0}')",

pCategoria.Nombre), ConexionBD.ObtenerConexion());

retorno = comando.ExecuteNonQuery();

return retorno;

}

public ArrayList getColumnas(String categorias)

{

MySqlConnection conexion = new MySqlConnection("server=127.0.0.1; database=clientes; Uid=root; pwd=GILGAMESH117;");

System.Data.DataSet ds = new System.Data.DataSet();

MySqlDataAdapter adaptador = new MySqlDataAdapter("select \* from categoria", conexion);

adaptador.Fill(ds, "categoria");

ArrayList columnas = new ArrayList();

for (int i = 0; i < ds.Tables[0].Columns.Count; i++)

{

columnas.Add(ds.Tables[0].Columns[i].ColumnName);

}

return columnas;

}

}

# Clase: Categorias.

public partial class Categoria : Form

{

public Categoria()

{

InitializeComponent();

}

private void button2\_KeyPress(object sender, KeyPressEventArgs e)

{

}

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

{

Form1 FormP = new Form1();

FormP.Show();

Categoria categoria = new Categoria();

categoria.Close();

}

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

{

CategoriaMetodos pCategoria = new CategoriaMetodos();

pCategoria.Nombre = txtNombreCategoria.Text.Trim();

int resultado = CategoriaCone.Agregar(pCategoria);

if (resultado > 0)

{

MessageBox.Show("Categoria Guardada Con Exito!!", "Guardado", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("No se pudo guardar el cliente", "Fallo!!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void txtNombreCategoria\_TextChanged(object sender, EventArgs e)

{

}

}

# Compilación.

Se registraron distintos errores al momento de compilar.

* La conexión a la base de datos no fue posible por un momento debido a errores de sintaxis (corregido).
* Algunos métodos no hacían nada más que ocupar espacio y provocar errores en el código (eliminado).
* Los datos de cliente y categoría se registraban con éxito en la base de datos, sin embargo, al momento de listarlos en el listView, estos no se podían visualizar correctamente.

Todos los errores anteriores fueron corregidos con éxito, la compilación se concluyó de forma exitosa a los 24 minutos transcurridos después de finalizar la fase de codificación.

# Pruebas.

Debido a que la creación del instalador no fue posible a causa de distintos errores (tanto del código como del software de la pc utilizada para la prueba) no a sido posible concluir esta fase.

# PostMorten.

Debido al error en la fase de pruebas, no se obtuvieron los resultados de forma completa, además de que la herramienta de resultados no puede ser ejecutada correctamente por los mismos motivos de la fase anterior.