**Breve descripción de sistema de solicitudes de servicios (prueba técnica Grupo Visión)**

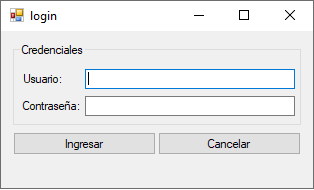


Figura 1.

La Figura 1 muestra la pantalla de “Login” donde el usuario del sistema, detalla sus credenciales en los campos específicos del formulario para poder ingresar al sistema. El formulario tiene condiciones como:

* No ha ingresado ningún usuario.
* No ha ingresado ninguna contraseña.
* Contraseña inválida.

A continuación, el código fuente de la pantalla “login”:

public partial class login : Form

{

public login()

{

InitializeComponent();

}

private SqlConnection con = new SqlConnection("Server= localhost\\localsqlbd; Database= solicitudes\_bd;Integrated Security=SSPI");

private SqlCommand command;

private Int32 levelID = -1;

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

Verificar();

}

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

{

this.Close();

}

private void Verificar()

{

con.Close();

if (txtUsuario.Text.Length > 0)

if (txtContrasena.Text.Length > 0)

{

try

{

con.Open();

string sql = null;

SqlDataReader dataReader;

sql = "select password, levelid from USUARIOS where username = '" + txtUsuario.Text.Trim() + "'";

command = new SqlCommand(sql, con);

dataReader = command.ExecuteReader();

while (dataReader.Read())

{

if (txtContrasena.Text == dataReader.GetValue(0).ToString())

{

levelID = Convert.ToInt32(dataReader.GetValue(1));

}

else

{

MessageBox.Show("Contraseña incorrecta");

txtContrasena.Text = "";

txtUsuario.Text = "";

}

}

dataReader.Close();

dataReader.Dispose();

con.Close();

if (levelID != -1)

{

Principal frm = new Principal(txtUsuario.Text.Trim(), levelID, con);

frm.ShowDialog(this);

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

string message = "Debe ingresar una contraseña. Desea continuar?";

string caption = "Contraseña inválida";

MessageBoxButtons buttons = MessageBoxButtons.YesNo;

DialogResult result;

result = MessageBox.Show(message, caption, buttons);

if (result == System.Windows.Forms.DialogResult.No)

{

// Closes the parent form.

this.Close();

}

}

else

{

string message = "Debe ingresar un usuario. Desea continuar?";

string caption = "Usuario inválido";

MessageBoxButtons buttons = MessageBoxButtons.YesNo;

DialogResult result;

result = MessageBox.Show(message, caption, buttons);

if (result == System.Windows.Forms.DialogResult.No)

{

// Closes the parent form.

this.Close();

}

}

}

}

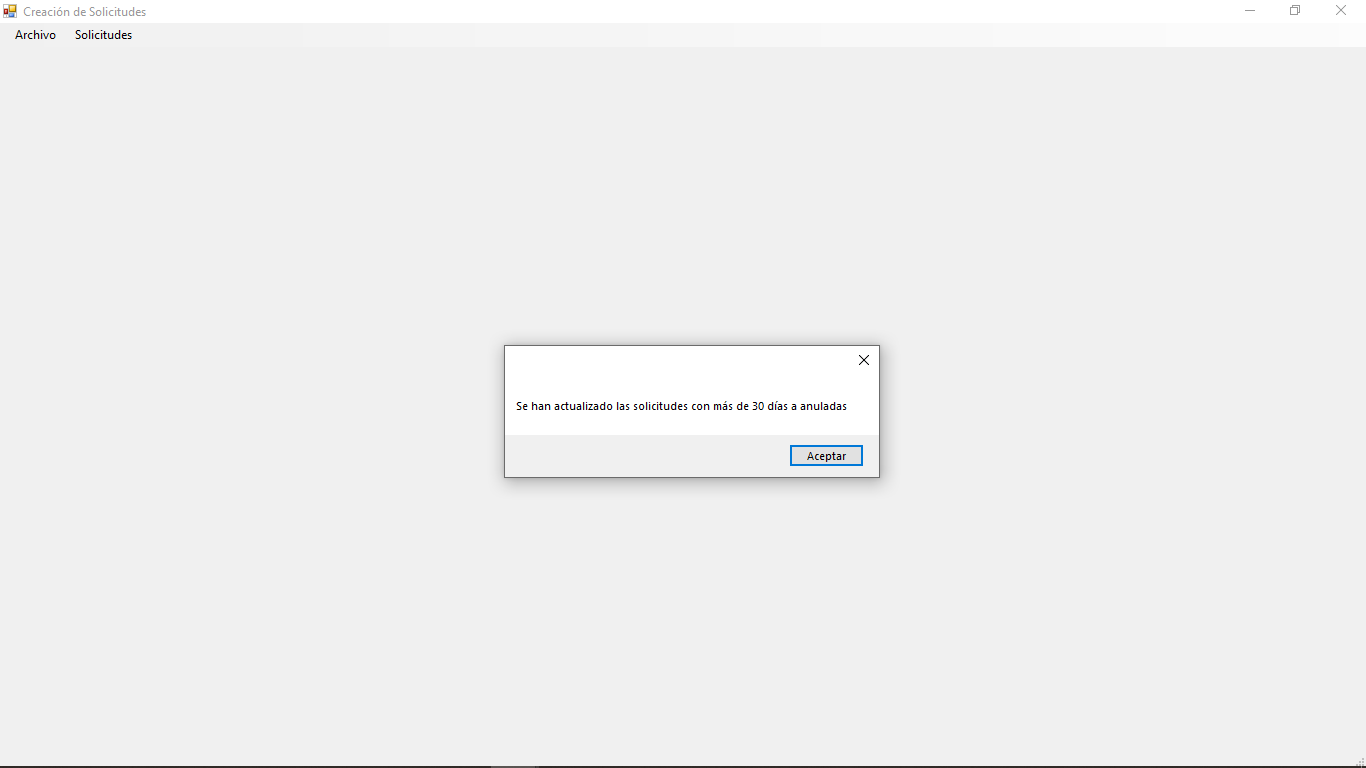


Figura 2.

Una vez que las credenciales son verificadas y aceptadas, el sistema actualiza las solicitudes que tienen más de 30 días y no han sido aprobadas. La Figura 2 muestra la pantalla principal y el mensaje de confirmación de la actualización de las solicitudes a un estado de “Anuladas”.

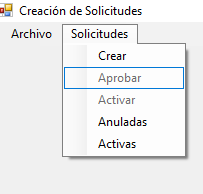


Figura 3.

Al ingresar con un perfil de “Digitador” o “Creador de solicitudes” el sistema inactiva las opciones de aprobar y activar las solicitudes. En la Figura 3 podemos observar que ambas opciones estás deshabilitadas.

A continuación, el código fuente del formulario principal:

public partial class Principal : Form

{

private SqlConnection con;

private string userFrm;

public Principal(string user, int level, SqlConnection conParametro)

{

InitializeComponent();

if (level == 0)

{

aprobarToolStripMenuItem.Enabled = false;

activarToolStripMenuItem.Enabled = false;

}

con = conParametro;

userFrm = user;

}

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

{

this.Close();

}

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

{

Aprobar frm = new Aprobar(userFrm, con);

frm.ShowDialog(this);

}

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

{

Activar frm = new Activar(userFrm, con);

frm.ShowDialog(this);

}

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

{

CrearSolicitudes frm = new CrearSolicitudes(userFrm, con);

frm.ShowDialog(this);

}

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

{

ActualizarAnuladas();

}

private void ActualizarAnuladas()

{

con.Open();

string sql = null;

sql = "update SOLICITUDES\_CLIENTES set ESTADOID = 4 where DATEDIFF (DAY, fecha\_creada , '" + DateTime.Today.Date + "' ) > 30";

SqlCommand command = new SqlCommand(sql, con);

command.ExecuteNonQuery();

con.Close();

MessageBox.Show("Se han actualizado las solicitudes con más de 30 días a anuladas");

}

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

{

Anuladas frm = new Anuladas(userFrm, con);

frm.ShowDialog(this);

}

}

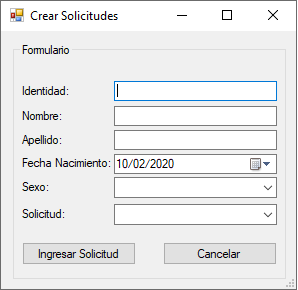


Figura 4.

El formulario para crear solicitudes es, por decirlo así, el más complejo de la solución. El primer paso es ingresar el número de identidad (con guión o sin guión el sistema lo reconoce) y verifica si el cliente ya existe en el sistema, si ya existe, verifica que no tenga una solicitud pendiente de aprobación y si no existe, lo agrega automáticamente.

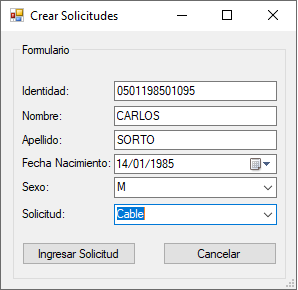


Figura 5.

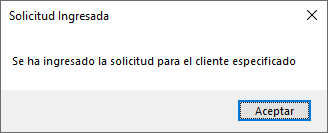


Figura 6.

La figura 5 y 6 son una secuencia del procedimiento de ingreso de una nueva solicitud, la figura 6 es la confirmación que la solicitud se ingresó correctamente.

Código fuente de Crear una Solicitud:

public partial class CrearSolicitudes : Form

{

private SqlConnection con;

private string userFrm;

private SqlCommand command;

private bool existe = false;

public CrearSolicitudes(string user, SqlConnection conParametro)

{

InitializeComponent();

con = conParametro;

userFrm = user;

}

private void textBox1\_Leave(object sender, EventArgs e)

{

con.Open();

string sql = null;

SqlDataReader dataReader;

sql = "SELECT \* FROM CLIENTES where id = '" + txtID.Text.Replace("-", "") + "'";

command = new SqlCommand(sql, con);

dataReader = command.ExecuteReader();

while (dataReader.Read())//si lee, el cliente existe y llena los campos

{

txtNombre.Text = dataReader.GetValue(1).ToString();

txtApellido.Text = dataReader.GetValue(2).ToString();

dtFechaNacimiento.Value = Convert.ToDateTime(dataReader.GetValue(3));

cbSexo.Text = dataReader.GetValue(4).ToString();

existe = true;

}

dataReader.Close();

dataReader.Dispose();

con.Close();

}

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

{

//

con.Open();

string sql = null;

SqlDataReader dataReader;

int solicitudId = comboBox1.SelectedIndex;

if (!existe)

{

DialogResult dialogResult = MessageBox.Show("El cliente no existe en la base de datos, ¿Desea agregarlo?", "Cliente no existe", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)//y lo inserta automaticamente si responde que si.

{

sql = "insert into CLIENTES (ID, NOMBRE, APELLIDO, FECHA\_NACIMIENTO, SEXO) VALUES ('" + txtID.Text.Replace("-", "") + "', '" + txtNombre.Text + "', '" + txtApellido.Text + "', '" + dtFechaNacimiento.Value.Date + "', '" + cbSexo.Text + "')";

command = new SqlCommand(sql, con);

command.ExecuteNonQuery();

sql = "insert into SOLICITUDES\_CLIENTES (SOLICITUDID, ID, ESTADOID, USERNAME) VALUES (" + solicitudId + ", '" + txtID.Text.Replace("-", "") + "', 0, '" + userFrm + "')";

command = new SqlCommand(sql, con);

command.ExecuteNonQuery();

}

else if (dialogResult == DialogResult.No)

{

txtID.Text = "";

txtApellido.Text = "";

txtNombre.Text = "";

}

}

else//verificar que no tenga solicitudes pendientes

{

bool tienePendiente = false;

sql = "select estadoid from SOLICITUDES\_CLIENTES where id = '"+ txtID.Text.Replace("-", "") + "'";

command = new SqlCommand(sql, con);

dataReader = command.ExecuteReader();

while (dataReader.Read())//Leerá hasta encontrar una pendiente

{

if(Convert.ToInt32(dataReader.GetValue(0)) == 1)

{

tienePendiente = true;

break;

}

}

dataReader.Close();

dataReader.Dispose();

if (tienePendiente)

{

txtID.Text = "";

txtApellido.Text = "";

txtNombre.Text = "";

dtFechaNacimiento.Value = DateTime.Today;

cbSexo.Text = "";

comboBox1.Text = "";

MessageBox.Show("No se puede ingresar otra solicitud para el cliente especificado", "Cliente pendiente de aprobar solicitud", MessageBoxButtons.OK);

}

else

{

sql = "insert into SOLICITUDES\_CLIENTES (SOLICITUDID, ID, ESTADOID, USERNAME) VALUES (" + solicitudId + ", '" + txtID.Text.Replace("-", "") + "', 1, '" + userFrm + "')";

command = new SqlCommand(sql, con);

command.ExecuteNonQuery();

txtID.Text = "";

txtApellido.Text = "";

txtNombre.Text = "";

dtFechaNacimiento.Value = DateTime.Today;

cbSexo.Text = "";

comboBox1.Text = "";

MessageBox.Show("Se ha ingresado la solicitud para el cliente especificado", "Solicitud Ingresada", MessageBoxButtons.OK);

}

}

}

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

{

this.Close();

}

}

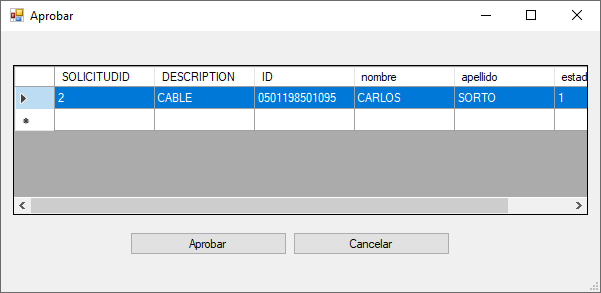


Figura 7

Una vez habiendo ingresado la solicitud, en la pantalla de aprobación aparecen todas las solicitudes recién ingresadas y que esperan su aprobación. Para aprobar una solicitud, solo se selecciona el registro y se da click en el botón de “Aprobar”, automáticamente se actualiza el estado y el usuario que aprobó dicha solicitud.

Código fuente de formulario “Aprobar”:

public partial class Aprobar : Form

{

private SqlConnection con;

private string userFrm;

private SqlCommand command;

public Aprobar(string user, SqlConnection conParametro)

{

InitializeComponent();

con = conParametro;

userFrm = user;

}

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

{

LlenarGrid();

con.Close();

}

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

{

this.Close();

}

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

{

string sql;

con.Open();

if (gvListadoCreados.SelectedRows.Count > 0)

{

sql = "UPDATE SOLICITUDES\_CLIENTES set estadoid = 2, username = '" + userFrm + "' where ID = '" + gvListadoCreados.SelectedRows[0].Cells[2].Value.ToString() + "' and solicitudid = " + gvListadoCreados.SelectedRows[0].Cells[0].Value.ToString() + " and estadoid = " + gvListadoCreados.SelectedRows[0].Cells[5].Value.ToString();

command = new SqlCommand(sql, con);

command.ExecuteNonQuery();

con.Close();

LlenarGrid();

}

else

MessageBox.Show("No ha seleccionado ningun registro, debe seleccionar uno para aprobarlo", "Seleccione un registro", MessageBoxButtons.YesNo);

}

private void LlenarGrid()

{

con.Open();

string sql = null;

SqlDataReader dataReader;

sql = "select A.SOLICITUDID, c.DESCRIPTION, A.ID, b.nombre, b.apellido, a.estadoid from SOLICITUDES\_CLIENTES A, clientes b, tipos\_solicitudes c where a.ID = b.ID and a.SOLICITUDID = c.SOLICITUDID and a.estadoid = 1";

command = new SqlCommand(sql, con);

SqlDataAdapter DaRec2 = new SqlDataAdapter(command);

DataTable dt = new DataTable();

DaRec2.Fill(dt);

gvListadoCreados.DataSource = dt;

}

}

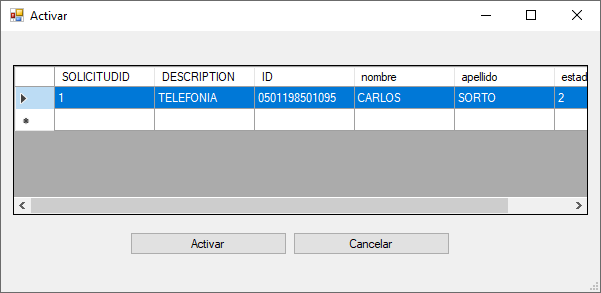


Figura 8.

Una similar pantalla a la anterior podemos ver en “Activar” una solicitud, la única diferencia es que el Grid View carga únicamente las solicitudes “Aprobadas”

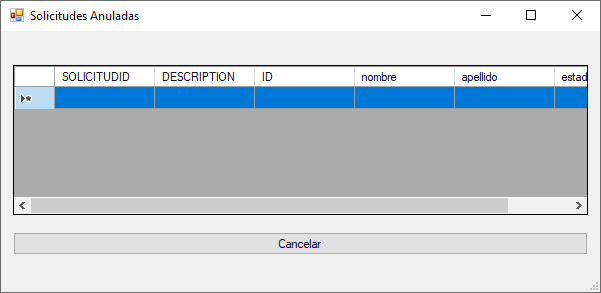


Figura 9.

En la Figura 9 se cargan las solicitudes que ya han sido anuladas por el sistema por tener más de 30 días sin ser aprobadas.

Codigo fuente de “Solicitudes Anuladas”:

public partial class Anuladas : Form

{

private SqlConnection con;

private string userFrm;

private SqlCommand command;

public Anuladas(string user, SqlConnection conParametro)

{

InitializeComponent();

con = conParametro;

userFrm = user;

}

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

{

con.Close();

LlenarGrid();

con.Close();

}

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

{

this.Close();

}

private void LlenarGrid()

{

con.Open();

string sql = null;

sql = "select A.SOLICITUDID, c.DESCRIPTION, A.ID, b.nombre, b.apellido, a.estadoid from SOLICITUDES\_CLIENTES A, clientes b, tipos\_solicitudes c where a.ID = b.ID and a.SOLICITUDID = c.SOLICITUDID and a.estadoid = 4";

command = new SqlCommand(sql, con);

SqlDataAdapter DaRec2 = new SqlDataAdapter(command);

DataTable dt = new DataTable();

DaRec2.Fill(dt);

gvListadoCreados.DataSource = dt;

}

}

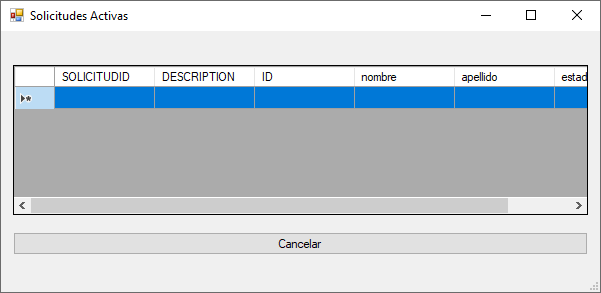


Figura 10.

En la figura 10 se cargan las solicitudes que después de haber sido “Aprobadas” son activas.

Código fuente de solicitudes “Activas”:

public partial class SoliActivas : Form

{

private SqlConnection con;

private string userFrm;

private SqlCommand command;

public SoliActivas(string user, SqlConnection conParametro)

{

InitializeComponent();

con = conParametro;

userFrm = user;

}

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

{

LlenarGrid();

con.Close();

}

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

{

this.Close();

}

private void LlenarGrid()

{

con.Open();

string sql = null;

sql = "select A.SOLICITUDID, c.DESCRIPTION, A.ID, b.nombre, b.apellido, a.estadoid from SOLICITUDES\_CLIENTES A, clientes b, tipos\_solicitudes c where a.ID = b.ID and a.SOLICITUDID = c.SOLICITUDID and a.estadoid = 3";

command = new SqlCommand(sql, con);

SqlDataAdapter DaRec2 = new SqlDataAdapter(command);

DataTable dt = new DataTable();

DaRec2.Fill(dt);

gvListadoCreados.DataSource = dt;

}

}

Accesos:

* Digitador:
  + Usuario: DIGI
  + Clave: 123HOLA
* Aprobador:
  + Usuario: APRO
  + Clave: 123HI
* Activador:
  + Usuario: ACTI
  + Clave: 123BYE