



La aplicación se creó con el objetivo de agilizar el trabajo de recepcionistas del centro médico "Asalud". A través de ella pueden cargar datos de pacientes y asignarles su turno correspondiente, ya sea que lo soliciten por vía telefónica o por mostrador, e incluso modificar turnos ya gestionados.

Recepcion

Password

Iniciar Sesión

Para poder ingresar al sistema, se debe tipear el password "asalud123"

Haciendo click en el botón "registrar turno" van a poder registrarse los turnos de forma manual. En el botón "turnos registrados" se podrán visualizar y en el caso de que sea necesario, modificar los turnos ya gestionados

lunes, 6 de junio de 2022 01:28

Registrar Turno

Turnos Registrados

Salir

ASALUD CENTRO MÉDICO

Completá los campos con los datos del paciente

Nombre	Apellido	Obra Social
<input type="text"/>	<input type="text"/>	<input type="text"/>
Dni	Teléfono	Fecha
<input type="text"/>	<input type="text"/>	06/06/2022 
Mail		Horario
<input type="text"/>		<input type="text"/>
Tipo de Consulta		
<input type="text"/>		
<input type="button" value="Registro"/>		
<input type="button" value="Volver"/>	<input type="button" value="Borrar"/>	

Este es el formulario que deberá completarse con la información correspondiente al paciente para poder generarle un turno


	Nombre	Apellido	Dni	Mail	Telefono	FechaConsulta	Horario	ObraSocial	TipoConsulta
▶	qwew	qweq	123123	sdfsdf	132123	04/06/2022	11 - 12 hs	Galeno	Ginecologia
	lucas	diaz	12312	lasduasid	1231	06/06/2022	10 - 11 hs	Galeno	Nutricion

ImportarExportarEliminarVolver

- Aquí se visualizarán los turnos ya cargados a mano. En el caso de que no haya ninguno cargado se podrán importar en un archivo xml o json.
- Además, se podrá exportar la información del día de la fecha en un archivo xml, json y un txt.
- En el caso de que se necesite modificar la información se hace un doble click en la posición que se quiere modificar y se abrirá otro formulario con los datos ya cargados que se podrán modificar. Una vez modificados para obtener la información actualizada se deberán exportar los datos nuevamente.

Completá los campos a modificar

Nombre <input type="text" value="Lucas"/>	Apellido <input type="text" value="Diaz"/>	Obra Social <input type="text" value="Osde"/>
Dni <input type="text" value="12345678"/>	Teléfono <input type="text" value="112233665788999"/>	Fecha <input type="text" value="06/06/2022"/>
Mail <input type="text" value="prueba@prueba.com"/>	Horario <input type="text" value="9 - 10 hs"/>	
Tipo de Consulta <input type="text" value="Ginecologia"/>		
<input type="button" value="Modificar"/>		



Este es el formulario que se visualizará en el caso de que se deseen modificar datos. Deberán ir al campo en el que quieran realizar un cambio y hacer click en el botón "modificar" para registrarlo nuevamente.

TEMAS APLICADOS:

1. Excepciones
2. Pruebas unitarias
3. Tipos genéricos
4. Interfaces
5. Archivos y serialización

1- EXCEPCIONES: utilizado para controlar posibles errores

```
9 referencias
public class ExcepcionArchivos : Exception
{
    /// <summary>
    /// Creación de la excepción con un mensaje.
    /// </summary>
    /// <param name="message"></param>
    1 referencia
    public ExcepcionArchivos(string message) : this(message, null)
    {
    }

    /// <summary>
    /// Creación de la excepción con un mensaje y su innerExcepcion.
    /// </summary>
    /// <param name="message"></param>
    /// <param name="innerException"></param>
    5 referencias
    public ExcepcionArchivos(string message, Exception innerException) : base(message, innerException)
    {
    }
}
```

2- PRUEBAS UNITARIAS: pruebas que se testean para verificar que los métodos cumplan su función. En el primer test se verifica que se pudo guardar el archivo.

```
namespace Test_Unitarios
{
    [TestClass]
    public class TestArchivos
    {
        [TestMethod]
        public void Archivos()
        {
            //Arrange
            Paciente paciente1 = new Paciente("pruebaNom", "pruebaApellido", "111111", "prueba@prueba.com", "15166", "01/01/1900", "11 - 12 hs", EObraSocial.Osde, ETipoConsulta.Nutricion);
            Paciente paciente2 = new Paciente("pruebaNom1", "pruebaApellido2", "1111112", "prueba2@prueba2.com", "2222", "01/01/2000", "13 - 14 hs", EObraSocial.SwissMedical, ETipoConsulta.G);

            Xml<List<Paciente>> xml = new Xml<List<Paciente>>();
            List<Paciente> pruebaLista = new List<Paciente>();

            //Act
            pruebaLista.Add(paciente1);
            pruebaLista.Add(paciente2);
            bool pudoGuardar = xml.Guardar("TestArchivo.xml", pruebaLista);

            //Assert
            Assert.IsTrue(pudoGuardar);
        }
    }
}
```

En el segundo test se lanza la excepción en el caso de que el archivo no se pueda leer.

```
[TestClass]
0 referencias
public class TestExcepcion
{
    [TestMethod]
    [ExpectedException(typeof(ExcepcionArchivos))]

    0 referencias
    public void Exception_Test()
    {
        string pruebaExc = null;
        List<Paciente> pruebaListaCliente = null;

        //Arrange
        Xml<List<Paciente>> xml = new Xml<List<Paciente>>();

        //Act

        //Assert
        xml.Leer(pruebaExc, out pruebaListaCliente);
    }
}
```

3- TIPOS GENÉRICOS: se implementa para manejar archivos

```
2 referencias
public interface IManejadorArchivos<T>
{
    /// <summary>
    /// Guarda la información que se le pasa por parametro en un archivo.
    /// </summary>
    /// <param name="archivo"></param>
    /// <param name="info"></param>
    /// <returns></returns>
    6 referencias
    bool Guardar(string archivo, T info);

    /// <summary>
    /// Lee la información del archivo guardado.
    /// </summary>
    /// <param name="archivo"></param>
    /// <param name="info"></param>
    /// <returns></returns>
    4 referencias
    bool Leer(string archivo, out T info);
}
```

```
11 referencias
public class Xml<T> : Extender, IManejadorArchivos<T>
    where T : class
{
    string carpeta;

    /// <summary>
    /// Propiedad de Extension del tipo archivo
    /// </summary>
    3 referencias
    protected override string Extension
    {
        get
        {
            return ".xml";
        }
    }
}
```

4- INTERFACES: es utilizada para la clase de xml con su genérico y para manejar los archivos

```

2 referencias
public interface IManejadorArchivos<T>
{
    /// <summary>
    /// Guarda la información que se le pasa por parametro en un archivo.
    /// </summary>
    /// <param name="archivo"></param>
    /// <param name="info"></param>
    /// <returns></returns>
    6 referencias
    bool Guardar(string archivo, T info);

    /// <summary>
    /// Lee la información del archivo guardado.
    /// </summary>
    /// <param name="archivo"></param>
    /// <param name="info"></param>
    /// <returns></returns>
    4 referencias
    bool Leer(string archivo, out T info);
}

```

5- ARCHIVOS: es utilizado en la clase "text" para guardar la información de los turnos y tener un historial con lo cargado anteriormente. Los archivos se guardan en la carpeta Formularios\bin\Debug\net5.0-windows

```

/// <summary>
/// Evento del boton click que exporta la información de turnos en archivo txt y xml.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void btn_Exportar_Click(object sender, EventArgs e)
{
    List<Paciente> listaPacientes = Consultorio.ObtenerPacientes();

    if (listaPacientes.Count < 1)
    {
        MessageBox.Show("Error. Por favor, cargue turnos para poder exportar.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        xml.Guardar(archivo, listaPacientes);
        txtListaPacientes.Guardar("txtPacientes.txt", Consultorio.InformacionPacientes());
        MessageBox.Show("Exportando... Se exporto el archivo.", "Turnos", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```


5- SERIALIZACIÓN: Utilizado para serializar y deserializar archivos de formato xml y json

```

/// <summary>
/// Evento del boton click que importa la información de los turnos cargados.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void btn_Importar_Click(object sender, EventArgs e)
{
    ImportarDatos();
}

```

```

/// <summary>
/// Método que importa los datos de un archivo xml.
/// </summary>
1 referencia
private void ImportarDatos()
{
    /// <summary>
    t /// Evento del boton click que exporta la información de turnos en archivo txt v xml.
    {
        /// <summary>
        /// Evento del boton click que exporta la información de turnos en archivo txt v xml.
        1 refe
        priv
        {
            1 refe
            priv
            {
                1 refe
                priv
                {
                    1 referencia
                    private void btn_Exportar_Click(object sender, EventArgs e)
                    {
                        List<Paciente> listaPacientes = Consultorio.ObtenerPacientes();

                        if (listaPacientes.Count < 1)
                        {
                            MessageBox.Show("Error. Por favor, cargue turnos para poder exportar.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        else
                        {
                            xml.Guardar(archivo, listaPacientes);
                            json.Guardar(archivoJson, listaPacientes);
                            txtlistaPacientes.Guardar("txtPacientes.txt", Consultorio.InformacionPacientes());
                            MessageBox.Show("Exportando... Se exporto el archivo.", "Turnos", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                    }
                }
            }
        }
    }
    catch (Except
    {
        MessageBox.Show(exe.Message, "Excepcion", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

/// <summary>
/// Evento del boton click que exporta la información de turnos en archivo txt y xml.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void btn_Exportar_Click(object sender, EventArgs e)
{
    List<Paciente> listaPacientes = Consultorio.ObtenerPacientes();

    if (listaPacientes.Count < 1)
    {
        MessageBox.Show("Error. Por favor, cargue turnos para poder exportar.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        xml.Guardar(archivo, listaPacientes);
        json.Guardar(archivoJson, listaPacientes);
        txtlistaPacientes.Guardar("txtPacientes.txt", Consultorio.InformacionPacientes());
        MessageBox.Show("Exportando... Se exporto el archivo.", "Turnos", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

6- BASE DE DATOS: utilizado al ingresar en el botón "turnos registrados" para cargar la información desde la base de datos en SQL. Para levantar la información de la base de datos se adjuntó un "script" en la carpeta "TP4" junto al anexo. Se pueden realizar distintas acciones como insertar, modificar y eliminar turnos.

IdPaciente	Nombre	Apellido	Dni	Mail	Telefono	FechaConsulta	Horario	ObraSocial	TipoConsulta
2	Lucas	Rodriguez	26564823	lucasrodriguez@prueba.com	1165987415	05/06/2022	10 - 11 hs	Osde	Nutricion
1	Lucas	Diaz Giosso	26457859	lucas@prueba.com	1165986589	06/06/2022	11 - 12 hs	Galeno	Nutricion
3	Lautaro	Galarza	34567654	lautaro@prueba.com	1123456745	15/06/2022	9 - 10 hs	Galeno	Ginecologia

```

1 referencia
public static bool EliminarPaciente(int idPaciente)
{
    try
    {
        command.CommandText = "DELETE FROM Pacientes WHERE Id = @Id";
        connection.Open();

        command.Parameters.AddWithValue("@Id", idPaciente);

        command.ExecuteNonQuery();
        command.Parameters.Clear();

        return true;
    }
    catch (Exception ex)
    {
        throw new ExcepcionArchivos($"Error al guardar el archivo(SQL)", ex);
    }
    finally
    {
        if (connection.State == ConnectionState.Open)
        {
            connection.Close();
        }
    }
}

```

SQLQuery6.sql - (local).dbConsultorio (DESKTOP-LG\lucas (54)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

dbConsultorio

Execute

Verificada

Eliminar Volver

SQLQuery6.sql - (L:\TP4-LG\lucas (54))

***** Script for SelectTopRows command from SSMS *****

```

SELECT TOP (1000) [Id]
,[Nombre]
,[Apellido]
,[Dni]
,[Mail]
,[Telefono]
,[Fecha_Consulta]
,[Horario]
,[Obra_Social]
,[Tipo_Consulta]
FROM [dbConsultorio].[dbo].[Pacientes]

```

Results Messages

Id	Nombre	Apellido	Dni	Mail	Telefono	Fecha_Consulta	Horario	Obra_Social	Tipo_Consulta
2	Lucas	Rodriguez	26564823	lucasrodriguez@prueba.com	1165987415	05/06/2022	10 - 11 hs	Osde	Nutricion
1	Lucas	Diaz Giosso	26457859	lucas@prueba.com	1165986589	06/06/2022	11 - 12 hs	Galeno	Nutricion
3	Lautaro	Galarza	34567654	lautaro@prueba.com	1123456745	15/06/2022	9 - 10 hs	Galeno	Ginecologia

Query executed successfully.

```

1 referencia
public static bool InsertarPacientes(Paciente pacientes)
{
    try
    {
        command.CommandText = "INSERT INTO Pacientes VALUES (@Nombre, @Apellido, " +
            "@Dni, @Mail, @Telefono, @Fecha_Consulta, @Horario, @Obra_Social, @Tipo_Consulta)";
        connection.Open();

        QueryPacientes(pacientes);
        command.ExecuteNonQuery();
        command.Parameters.Clear();

        return true;
    }
    catch (Exception ex)
    {
        throw new ExcepcionArchivos($"Error al insertar un valor en la tabla.(SQL)", ex);
    }
    finally
    {
        if (connection.State == ConnectionState.Open)
        {
            connection.Close();
        }
    }
}

```

```

1 referencia
public static bool ModificarPaciente(Paciente paciente)
{
    try
    {
        command.CommandText = "UPDATE Pacientes SET Nombre = @Nombre, Apellido = @Apellido, Dni = @Dni, Mail = @Mail, Telefono = @Telefono, Fecha_Consulta = @Fecha_Consulta, Horario = @Horario, Obra_social = @Obra_Social, Tipo_Consulta = @Tipo_Consulta WHERE Id = @Id";
        connection.Open();

        command.Parameters.AddWithValue("@Id", paciente.IdPaciente);
        command.Parameters.AddWithValue("@Nombre", paciente.Nombre);
        command.Parameters.AddWithValue("@Apellido", paciente.Apellido);
        command.Parameters.AddWithValue("@Dni", paciente.Dni);
        command.Parameters.AddWithValue("@Mail", paciente.Mail);
        command.Parameters.AddWithValue("@Telefono", paciente.Telefono);
        command.Parameters.AddWithValue("@Fecha_Consulta", paciente.FechaConsulta);
        command.Parameters.AddWithValue("@Horario", paciente.Horario);
        command.Parameters.AddWithValue("@Obra_Social", paciente.ObraSocial.ToString());
        command.Parameters.AddWithValue("@Tipo_Consulta", paciente.TipoConsulta.ToString());

        command.ExecuteNonQuery();
        command.Parameters.Clear();

        return true;
    }
    catch (Exception ex)
    {
        throw new ExcepcionArchivos($"Error al modificar el turno(SQL)", ex);
    }
}

```

7- DELEGADOS: utilizado al momento de registrar un turno y se le informa al usuario que el turno con el apellido correspondiente al paciente se ha agregado correctamente.

```

/// <summary>
/// Método que devuelve la informacion con el apellido del paciente.
/// </summary>
/// <param name="dato"></param>
1 referencia
private void PasarApell(string dato)
{
    lbl_Info.Text = $"El turno del paciente {dato} fue agregado.";
}

```

```

/// <summary>
/// Delegado que muestra el apellido del paciente que se agrego el turno.
/// </summary>
/// <param name="mensaje"></param>
public delegate void DelegadoMensaje(string mensaje);

/// <summary>
/// Delegado que va a desactivar el boton de eliminar
/// </summary>
public delegate void DelegadoDesactivarBoton();

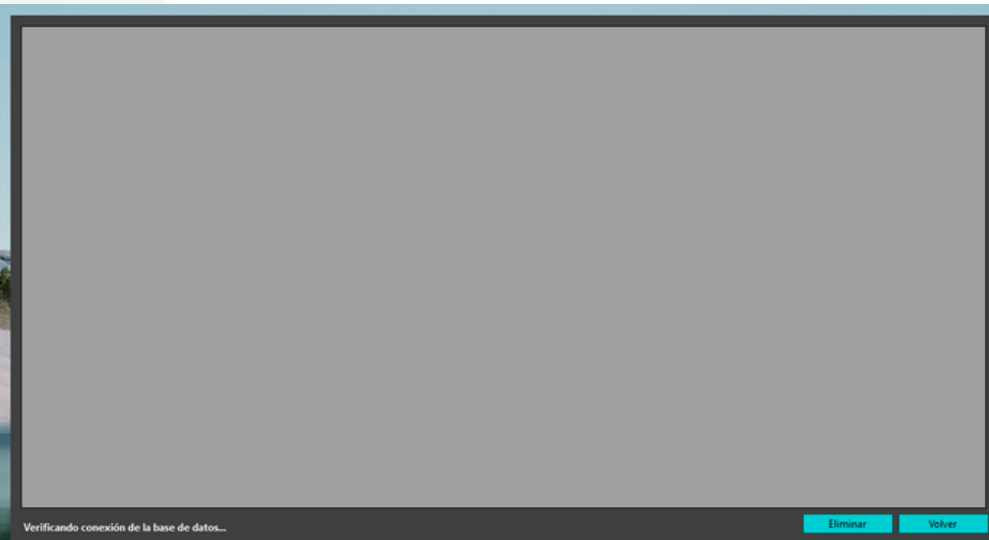
```

```

if (!string.IsNullOrEmpty(nombre) && !string.IsNullOrEmpty(apellido)
    && !string.IsNullOrEmpty(dni) && !string.IsNullOrEmpty(mail)
    && !string.IsNullOrEmpty(telefono) && !string.IsNullOrEmpty(fecha)
    && !string.IsNullOrEmpty(obraSocial) && !string.IsNullOrEmpty(hora) && !string.IsNullOrEmpty(obraSocial))
{
    if (dni.Length == 8)
    {
        if (mail.Contains("@"))
        {
            paciente = new Paciente(idPaciente, nombre, apellido, dni, mail, telefono, fecha, hora, (EObraSocial)ObtenerObraSocial(obraSocial), (
            Consultorio.AgregarPaciente(paciente, PasarApell);
            DAO.InsertarPacientes(paciente);
        }
    }
}

```

9- HILOS Y EXPRESIONES LAMDA: se utilizo para simular que hay un tiempo de espera para que la base de datos se cargue si tiene información o no.



```
public class ManejadorDatos
{
    /// <summary>
    /// Metodo que simula la carga de la base de datos.
    /// </summary>
    /// <returns></returns>
    1 referencia
    public static async Task<string> LevantarDatos()
    {
        string informacion = await Task.Run(() => { Thread.Sleep(3000); return "Verificada"; });

        return informacion;
    }
}
```

```
/// <summary>
/// Método que actualiza los datos del datagrid.
/// </summary>
1 referencia
private async void ActualizarDatos()
{
    List<Paciente> listaClientes = Consultorio.ObtenerPacientes();

    DAO.Leer(out listaClientes);

    if (listaClientes.Count > 0)
    {
        this.dgv_Lista.DataSource = new BindingList<Paciente>(listaClientes);
    }
}
```

```
private async void FormLista_Load(object sender, EventArgs e)
{
    btn_Importar.Visible = false;
    btn_Exportar.Visible = false;
    this.lbl_InfoCarga.Text = await ManejadorDatos.LevantarDatos();
    ActualizarDatos();
}
```

10- EVENTOS: utilizado para desactivar el botón de "Eliminar" si la base de datos no tiene información cargada. En el caso de que si contenga información se activa nuevamente.

```
public event DelegadoDesactivarBoton desactivar;
```

```
1 referencia
public FormLista()
{
    InitializeComponent();
    xml = new Xml<List<Paciente>>();
    json = new Json<List<Paciente>>();
    txtListaPacientes = new Text();
    listaPacientesInicial = new List<Paciente>();
    desactivar += DesactivarBoton;
}
```

```
    }
    else
    {
        desactivar.Invoke();
        MessageBox.Show("No hay datos para mostrar.", "Información", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

/// <summary>
/// Método que deshabilita el boton de eliminar.
/// </summary>
1 referencia
private void DesactivarBoton()
{
    btn_Eliminar.Enabled = false;
}
```