



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 <input type="text"/>
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"/>		

 **ASALUD**
CENTRO MÉDICO

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 innerException.
    /// </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", "pruebaApell", "111111", "prueba@prueba.com", "15166", "01/01/1900", "11 - 12 hs", EObraSocial.Osde, ETipoConsulta.Nutricion);
            Paciente paciente2 = new Paciente("pruebaNom1", "pruebaApell2", "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 y xml.
    {
        /// <summary>
        /// Evento del boton click que exporta la información de turnos en archivo txt y xml.
        /// </summary>
        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);
    }
}
```