

Práctica 3

Interfaz de usuario y acceso a BD desde aplicaciones web

Objetivos

- Aprender a crear una solución en **Visual Studio** que conste de varios proyectos.
- Aprender a crear una aplicación web con interfaz gráfica de usuario haciendo uso de C# en el entorno **ASP.Net**.
- Aprender a crear una aplicación que haga uso de un motor de BD SQL, como SQL Server, desde una aplicación web **ASP.Net**.
- Continuar aprendiendo a usar Git.

Requisitos técnicos

Requisitos que tiene que cumplir este trabajo práctico para ser evaluado (si no se cumple alguno de los requisitos la calificación será **cero**):

- La solución entregada no contiene archivos compilados, por lo que, **antes de comprimir, debes limpiar la solución (Compile > Clean solution)**.
- El archivo entregado se llama `hada-p3.zip` (**todo en minúsculas**).
- Al descomprimir el archivo `hada-p3.zip` se crea un directorio de nombre `hada-p3` (**todo en minúsculas**).
- Dentro del directorio `hada-p3` hay un archivo de nombre `hada-p3.sln`.
- Dentro del directorio `hada-p3` hay tres directorios: `usuWeb`, `library` y `.git`.
- Los directorios `hada-p3/usuWeb` y `hada-p3/library` contienen los archivos con el código de asignación y se llaman como se indica en la sentencia (en todos los casos, distinguiendo entre mayúsculas y minúsculas).
- Los nombres de los namespaces, clases y métodos implementados, así como sus argumentos, se llaman como se indica en la sentencia (en todos los casos, distinguiendo entre mayúsculas y minúsculas).
- Los mensajes producidos siguen el formato especificado en la declaración (en todos los casos, distinguiendo entre mayúsculas y minúsculas).
- Se han realizado **al menos 3 commits** y en cada uno de ellos los cambios o adiciones realizados **demuestran avances** en el desarrollo de la práctica. Cada uno de estos commits deberá contener, tu DNI/NIE en el comentario del commit y un mensaje explicativo.
- El código debe compilar y ejecutarse sin problemas en los ordenadores del

laboratorio de prácticas de Hada.

Guía de evaluación

- La creación de cada una de las clases e interfaces así como de todos y cada uno de sus componentes (constructores, propiedades, métodos, etc...) trabajando de forma correcta supondrá hasta el 90% de la nota.
- Los distintos commits realizados a lo largo de la creación de la práctica supondrán hasta el 10% de la nota.

Entrega

La entrega de esta práctica consiste en el directorio de la solución `hada-p3`, junto con todo su contenido, comprimido en un fichero llamado `hada-p3.zip`

Lugar y fecha de entrega: La entrega se realizará en <http://pracdlsi.dlsi.ua.es>. No se admitirá ningún otro método de entrega.

Fecha límite: 05/04/2020

Descripción

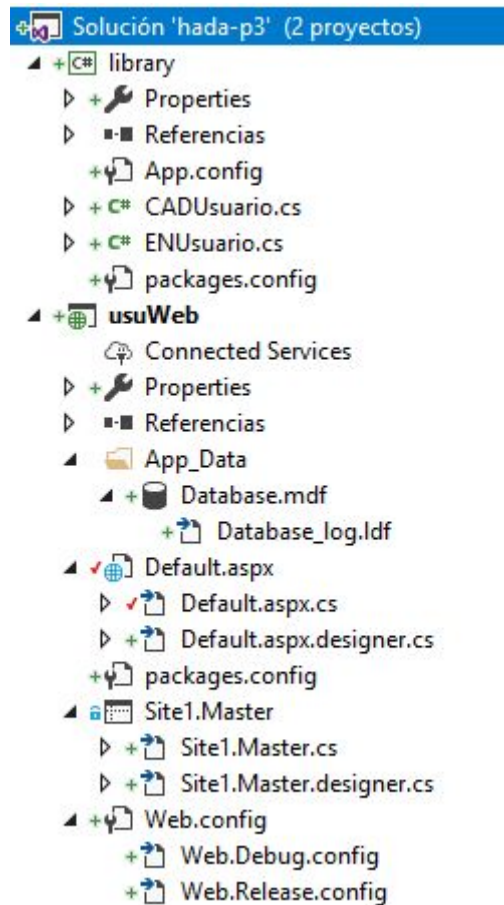
En esta tercera práctica vamos a crear una “solución” (un contenedor de proyectos) con *Visual Studio* para implementar una aplicación con interfaz gráfico de usuario que acceda a una BD en SQL Server.

Sigue los pasos indicados, **respetar el uso de mayúsculas y minúsculas** así como el **nombre** de las **carpetas**, **archivos**, **espacios de nombres**, **clases**, **métodos** y **argumentos** y el **formato** de **mensajes** que se te indique.

Crea una solución de nombre **hada-p3** y un repositorio **git** en la carpeta de la solución y ve haciendo commits en él de todo lo que vayas haciendo. En el **comentario de cada commit** debes incluir tu **DNI/NIE** además de un mensaje explicativo (ej: `git commit -m "12345678A Añadido método M1"`).








Estructura de la solución

En esta práctica la solución (**hada-p3**) contendrá dos proyectos: **usuWeb** y **library**. El proyecto **usuWeb** creará el interfaz gráfico de la aplicación para llevar a cabo la interacción con el usuario, mientras que **library** contendrá el código de las entidades de negocio (EN) y de la capa de acceso a datos (CAD). De este modo, en el IDE de *Visual Studio* deberás crear una estructura como la siguiente:



Proyecto usuWeb

Es un proyecto de tipo **Aplicación web ASP .Net (.NET FRAMEWORK)** con **plantilla vacía** que debes añadir a la solución y que será el encargado de construir la interfaz de usuario. En él debes de crear una página maestra (Web Forms con página maestra) con nombre Site1.Master como se muestra en las siguientes imágenes:

Icono	Nombre	Tipo	Descripción
	Formulario Web Forms	Visual C#	Tipo: Visual C# Página maestra para aplicaciones web
	Formulario web con página maestra	Visual C#	
	Archivo de máscara de Web Forms	Visual C#	
	Control de usuario de Web Forms	Visual C#	
	Control del servidor de formularios Web Forms	Visual C#	
	Página maestra de Web Forms (anidada)	Visual C#	
	Web Forms con página maestra	Visual C#	

Site1.Master








div#header

Nombre alumno - DNI alumno

HADA P3 - CURSO 2019-2020

En la sección superior deberás poner tu nombre y tu NIF/NIE, y debajo el nombre de la asignatura y curso con una etiqueta HTML `<h1>`.

A continuación, deberás **añadir un formulario web** que derive de la página maestra (Formulario Web con página maestra) con nombre Default.aspx.

Icono	Nombre	Tipo	Descripción
	Formulario Web Forms	Visual C#	Tipo: Visual C# Formulario de aplicaciones web creado a partir de una página maestra
	Formulario web con página maestra	Visual C#	
	Archivo de máscara de Web Forms	Visual C#	
	Control de usuario de Web Forms	Visual C#	
	Control del servidor de formularios Web Forms	Visual C#	
	Página maestra de Web Forms (anidada)	Visual C#	
	Web Forms con página maestra	Visual C#	

Este formulario web será la página inicial de la aplicación y su contenido debe ser

el siguiente:

Nombre alumno - DNI alumno

HADA P3 - CURSO 2019-2020

Página de usuarios

NIF:

Nombre:

Edad:

Como puedes apreciar, se trata de una formulario ASPNET que se utilizará para crear nuevos usuarios, actualizar los ya existentes, leer un usuario de la base de datos y mostrarlo en el formulario, leer el siguiente o el anterior usuario en la base de datos y, por último, borrar el usuario. El usuario tendrá como datos el NIF, nombre y edad, que serán los campos en los que se puede escribir (`<asp:TextBox>`), y además los botones (`<asp:Button>`) para leer (lee el usuario dado un NIF), leer primero (el primer usuario en la base de datos), leer anterior (lee el usuario anterior al indicado con NIF), leer siguiente (lee el usuario siguiente al indicado con NIF), crear, actualizar y borrar.

Operaciones con usuarios

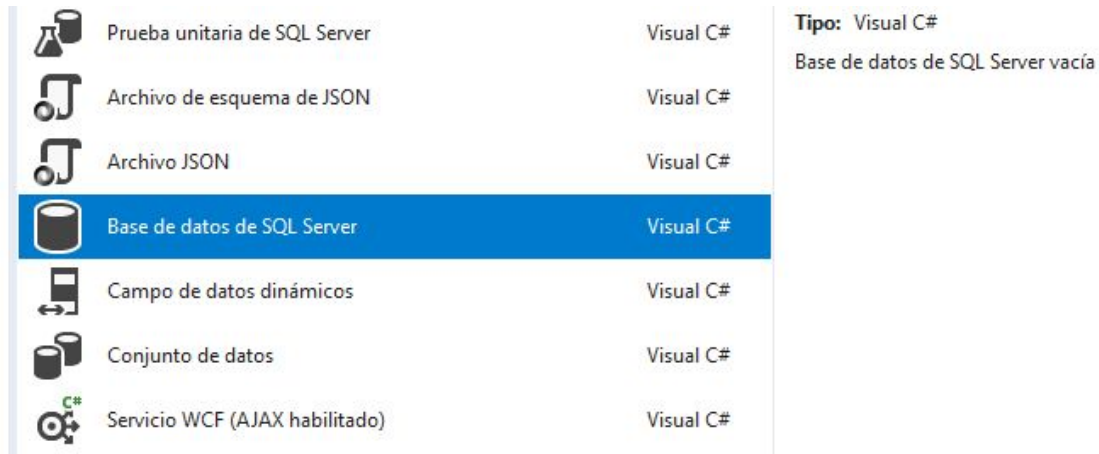
Con el formulario visto anteriormente tendremos la opción de leer, crear un nuevo usuario, actualizar uno ya existente o eliminarlo. Es importante que a la hora de realizar operaciones en la base de datos se hagan las comprobaciones correspondientes. Por ejemplo, antes de crear un usuario se comprueba que no exista un usuario con el mismo NIF en la base de datos. A su vez, a la hora de actualizar un usuario, es importante comprobar que, en este caso, sí que exista previamente en la base de datos uno con el mismo NIF.

Cuando se realice la operación, se debe mostrar un texto indicando el éxito de la operación. En el caso que se haya producido un error también se debe indicar. Para ello, podéis utilizar una etiqueta (`asp:Label`) que sirva para mostrar texto con el mensaje de error en cada una de las operaciones a realizar. También debéis

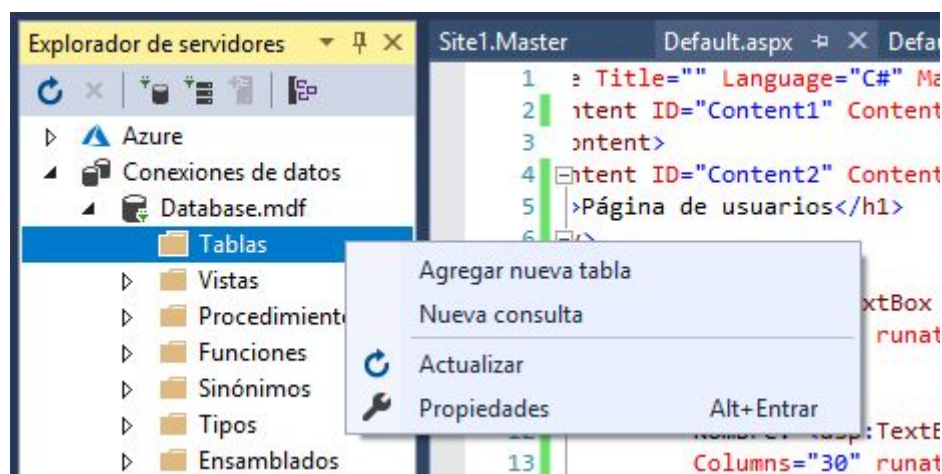
mostrar por consola dicho error: `Console.WriteLine("User operation has failed. Error: {0}", ex.Message);`

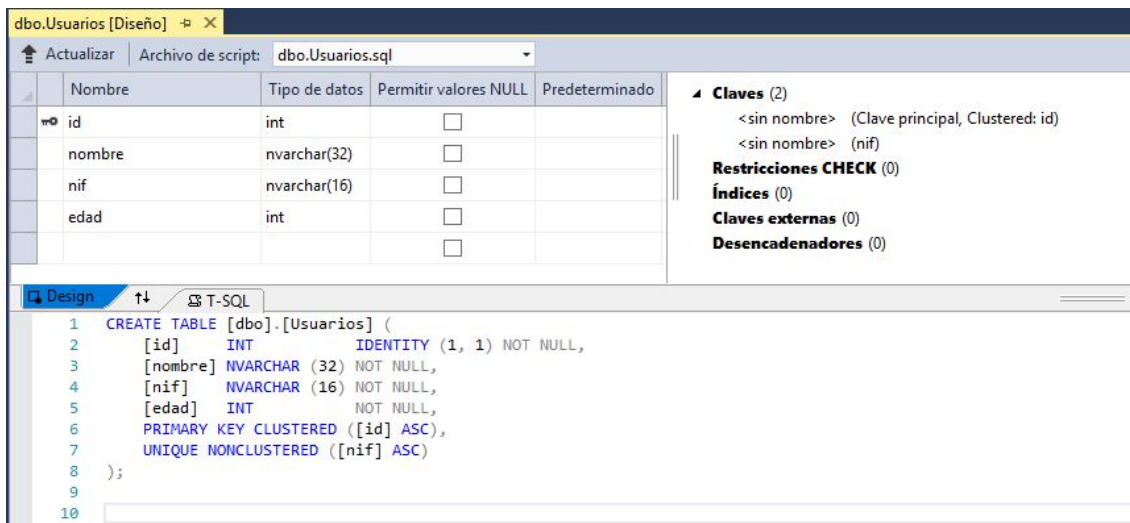
Creación de la BD SQL Server

En el proyecto **usuWeb** deberás crear una base de datos del tipo archivo mdf que llamarás **Database.mdf**.



Ello creará la carpeta **APP_Data** dentro del proyecto donde estará contenido este archivo de base de datos creado. Esta base de datos contendrá toda la estructura y los datos para esta práctica. Haz doble click sobre dicho archivo para abrir el explorador de servidores y crear la tabla **Usuarios** del mismo modo al indicado en las siguientes imágenes:



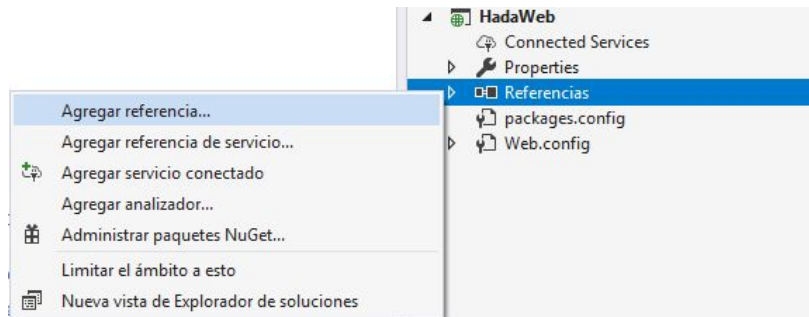


```
CREATE TABLE [dbo].[Usuarios]  
(  
    [id] INT IDENTITY (1, 1) NOT NULL,  
    [nombre] NVARCHAR (32) NOT NULL,  
    [nif] NVARCHAR (16) NOT NULL,  
    [edad] INT NOT NULL,  
    PRIMARY KEY CLUSTERED ([id] ASC),  
    UNIQUE NONCLUSTERED ([nif] ASC)  
)
```

Relación entre los proyectos usuWeb y library

Como habrás podido intuir, el código que debe haber en los archivos del proyecto **usuWeb** pertenece sólo a la capa de presentación. Cada vez que desde el interfaz se tenga que hacer algo con usuarios se deberá llamar a los métodos apropiados de la clase correspondiente de la lógica de negocio (EN). A su vez, cada vez que haya que hacer algo relacionado con la parte de acceso a datos habrá que llamar desde la capa de la lógica de negocio a los métodos apropiados de la clase correspondiente de la capa de acceso a datos (CAD).

Tal y como vimos en la Práctica 0, **se debe añadir una referencia en el proyecto usuWeb al proyecto library** para poder utilizar sus métodos.



Proyecto library

Es un proyecto de tipo **Librería** (Biblioteca de clases C# .NET FRAMEWORK con nombre library) que debes añadirlo a la solución. Su código fuente se encuentra distribuido en las siguientes clases CADUsuario y ENUsuario como se muestra a continuación:

```
▲ + C# CADUsuario.cs
  ▲ CADUsuario
    ├── constring : string
    ├── CADUsuario()
    ├── readUsuario(ENUsuario) : bool
    ├── readNextUsuario(ENUsuario) : bool
    ├── readPrevUsuario(ENUsuario) : bool
    ├── readFirstUsuario(ENUsuario) : bool
    ├── createUsuario(ENUsuario) : bool
    ├── updateUsuario(ENUsuario) : bool
    └── deleteUsuario(ENUsuario) : bool

▲ + C# ENUsuario.cs
  ▲ ENUsuario
    ├── nif : string
    ├── nombre : string
    ├── edad : int
    ├── nifUsuario : string
    ├── nombreUsuario : string
    ├── edadUsuario : int
    ├── ENUsuario()
    ├── ENUsuario(string, string, int)
    ├── readUsuario() : bool
    ├── readFirstUsuario() : bool
    ├── readPrevUsuario() : bool
    ├── readNextUsuario() : bool
    ├── createUsuario() : bool
    ├── updateUsuario() : bool
    └── deleteUsuario() : bool
```

Clase CADUsuario

Esta clase está definida en el archivo CADUsuario.cs y tendrá el atributo privado *constring* que contendrá la cadena de conexión a la base de datos (configurada en Web.config) y los siguientes métodos. Recuerda que en cada método si se produce

un error, se devolverá false y se capturará la excepción de tipo `SQLException` y se mostrará por consola el mensaje de error indicado previamente en esta práctica (Sección *Operaciones con usuarios*):

- **public CADUsuario ()**: Inicializa la cadena de conexión de la BD.
- **public bool createUsuario (ENUsuario en)**: Crea un nuevo usuario en la BD con los datos del usuario representado por el parámetro **en**.
- **public bool readUsuario (ENUsuario en)**: Devuelve solo el usuario indicado leído de la BD.
- **public bool readFirstUsuario (EnUsuario en)**: Devuelve solo el primer usuario de la BD.
- **public bool readNextUsuario (ENUsuario en)**: Devuelve solo el usuario siguiente al indicado.
- **public bool readPrevUsuario (ENUsuario en)**: Devuelve solo el usuario anterior al indicado.
- **public bool updateUsuario (ENUsuario en)**: Actualiza los datos de un usuario en la BD con los datos del usuario representado por el parámetro **en**.
- **public bool deleteUsuario (ENUsuario en)**: Borra el usuario representado en **en** de la BD.

Clase ENUsuario

Esta clase está definida en el archivo `ENUsuario.cs` y tendrá los atributos privados *nif*, *nombre* y *edad* junto a sus correspondientes propiedades públicas con campo de respaldo para poder obtener dichos valores, además de los siguientes métodos:

- **public ENUsuario ()**
- **public ENUsuario (string nif, string nombre, int edad)**
- **public bool createUsuario ()**: Guarda este usuario en la BD. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.
- **public bool readUsuario ()**: Recupera el usuario indicado de la BD. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.
- **public bool readFirstUsuario ()**: Recupera todos los usuarios de la BD y devuelve solo el primer usuario. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.



- **public bool readNextUsuario ()**: Recupera todos los usuarios de la BD y devuelve solo el usuario siguiente al indicado. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.
- **public bool readPrevUsuario ()**: Recupera todos los usuarios de la BD y devuelve solo el usuario anterior al indicado. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.
- **public bool updateUsuario ()**: Actualiza este usuario en la BD. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.
- **public bool deleteUsuario ()**: Borra este usuario de la BD. Para ello hará uso de los métodos apropiados de **CADUsuario**. Devuelve false si no se ha podido realizar la operación.