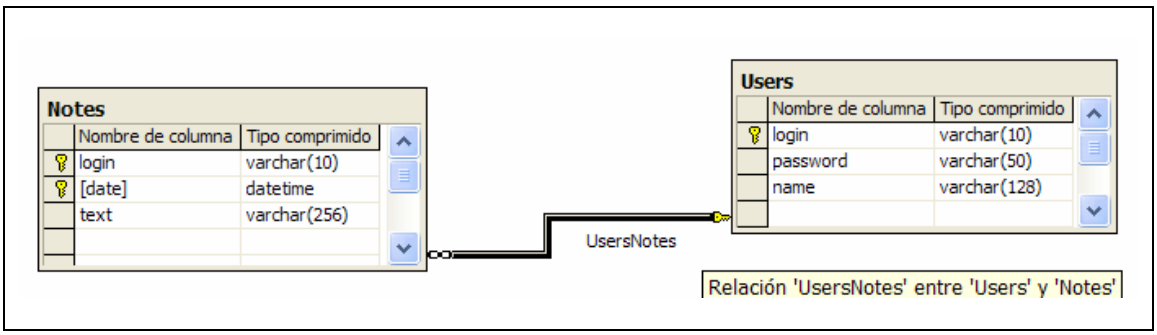
Boletín de ADO.Net

En este boletín vamos a realizar una aplicación completa con n-capas de acceso a datos. La aplicación será la gestión simple de una agenda personal. Los pasos que vamos a dar son los siguientes:

1. Crear la BD con SQL Server.
2. Crear una DLL que añada/borre/actualice usuarios y notas en la agenda.
3. Crear una aplicación de consola que añada/borre/actualice usuarios y notas. Los resultados de las pruebas de ejecución los iremos viendo en el mismo SQL Server.
4. Haremos una pequeña aplicación Windows que use esa DLL (aplicación 3-capas)
5. Haremos un servicio Web que publique la funcionalidad más importante de la DLL anterior.
6. Modificaremos la aplicación Windows realizada para utilizar el servicio Web (aplicación 4-capas)

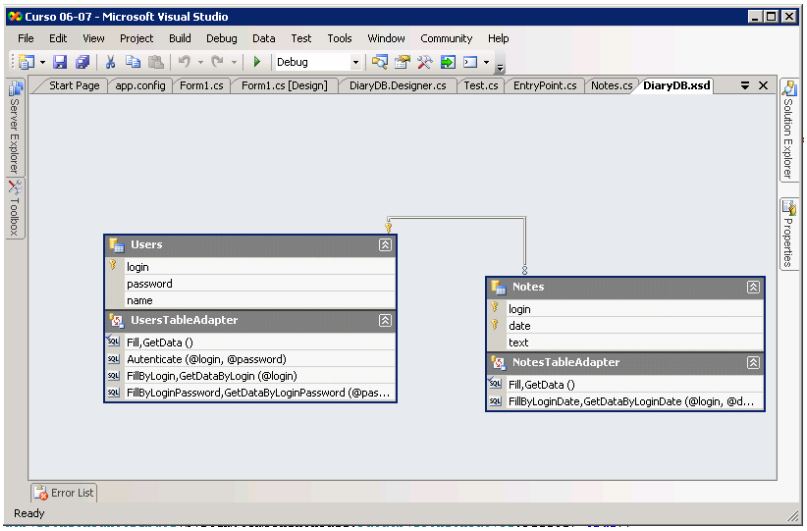
Creando la base de datos

Con el administrador corporativo del SQL Server crearemos la base de datos DiaryDB con el siguiente DER:



Creando la DLL

Crearemos una biblioteca de clases DiaryDAL (Data Access Layer). A dicha biblioteca le agregaremos un conjunto de datos, y a ese conjunto de datos la parte de la bases de datos con la que queremos trabajar. En esta caso las dos tablas:



Para cada tabla X, VS nos crea un objeto de tipo XDataTable para poder manejar dicha tabla, además, también crea un objeto de tipo XTableAdapter

para poder agregar consultas a dicha tabla. Para este ejemplo las consultas que hemos añadido son:

Users:

- Authenticate:

```
SELECT     COUNT(*) AS Expr1, login, password
FROM         Users
GROUP BY login, password
HAVING      (login = @login) AND (password = @password)
```

- FillByLogin:

```
SELECT login, password, name
FROM dbo.Users
WHERE login = @login
```

- FillByLoginPassword:

```
SELECT login, name, password
FROM Users
WHERE (password = @password) AND (login = @login)
```

Notes:

- FillByLoginDate:

```
SELECT date, login, text
FROM Notes
WHERE (login = @login) AND (date BETWEEN @date1 AND @date2)
```

Una vez creado el conjunto de datos, el siguiente paso es crear las clases de acceso a datos, en esta caso dos clases (Users.cs y Notes.cs). Dichas clases harán uso del patrón singlenton para manejar una única instancia del objeto TableAdapter que corresponda, haciendo de envoltorio del conjunto de datos. Por ejemplo para añadir un nuevo usuario bastaría con:

```
private UsersTableAdapter _usersAdapter = null;

protected UsersTableAdapter Adapter
{
    get
    {
        if (_usersAdapter == null)
            _usersAdapter = new UsersTableAdapter();
        return _usersAdapter;
    }
}
```

```
public bool Add(string login, string password, string name)
{
    DiaryDB.UsersDataTable users = new DiaryDB.UsersDataTable();
    users.AddUsersRow(login, password, name);
    int rowsAffected = Adapter.Update(users);
    return rowsAffected == 1;

}
```

Para obtener los usuarios con un determinado login y password sería:

```
public DiaryDB.UsersDataTable getUsersByLoginPassword(string login, string password)
{
    return Adapter.GetDataByLoginPassword(login, password);
}
```

Un aspecto importante a tener en cuenta más adelante es utilizar el modificador de método adecuado para indicarle a VS que estas clases son de acceso a datos para consulta/añadir/borrar/actualizar. Teniendo en cuenta esto, para eliminar un usuario sería:

```
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodTy
pe.Delete, true)]
public bool Delete(string login)
{
    int rowsAffected = Adapter.Delete(login);
    return rowsAffected == 1;

}
```

Creando una aplicación consola para probar la DLL (DiaryCon)

Este proyecto carece de interés a estas alturas del curso, el objetivo del mismo del mismo es crear una aplicación de consola para poder comprobar que la DLL funciona correctamente, para ello iremos haciendo pruebas y viendo los resultados en la consola de administración de SQLServer.

```
public class Test
{
    private Users u;
    private Notes n;

    public Test()
    {
        u = new Users();
        n = new Notes();
    }
    public void run()
    {
        int op = 0;

        do
        {
```

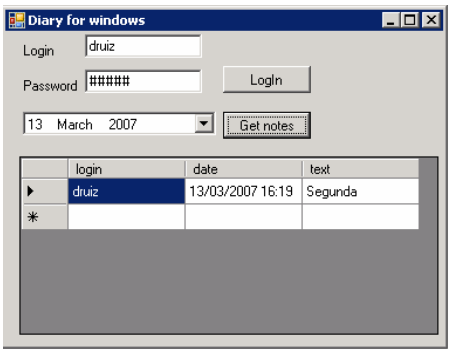
```
            op = showMenu();
            switch (op)
            {
                case 1: //add
                    testAddUser();
                    break;
                case 2: //delete
                    testDeleteUser();
                    break;
                case 3: //update
                    testUpdateUser();
                    break;
                case 4: //add note
                    testAddNote();
                    break;
                case 5: //delete note
                    testDeleteNote();
                    break;
                case 6: //update note
                    testUpdateNote();
                    break;
                case 7: //autenticate
                    testAuthenticate();
                    break;
                case 8: //get name
                    testGetName();
                    break;
            }
        } while (op != 0);
    }

    private int showMenu()
    {
        int result = 0;

        Console.Out.WriteLine("1.- Add User");
        Console.Out.WriteLine("2.- Delete User");
        Console.Out.WriteLine("3.- Update User");
        Console.Out.WriteLine("4.- Add Note");
        Console.Out.WriteLine("5.- Delete Note");
        Console.Out.WriteLine("6.- Update Note");
        Console.Out.WriteLine("7.- Authenticate");
        Console.Out.WriteLine("8.- Get name");
        result = Convert.ToInt32(askFor("Choose an option (0 to finish): "));
        return result;

    }

    private void testAddUser()
    {
        Console.Out.WriteLine(
        u.Add(
            askFor("Login: "),
            askFor("Password: "),
            askFor("Name: ")));
    }

    private void testUpdateUser()
    {
        Console.Out.WriteLine(
        u.Update(
            askFor("newLogin: "),
```

```
            askFor("newPassword: "),
            askFor("newName: "),
            askFor("Login: ")));
    }

    private void testDeleteUser()
    {
        Console.Out.WriteLine(
        u.Delete(
            askFor("Login: ")));
    }

    private void testAddNote()
    {
        Console.Out.WriteLine(
        n.Add(
            askFor("Login: "),
            System.DateTime.Now,
            askFor("Text: ")));
    }

    private void testDeleteNote()
    {
        Console.Out.WriteLine(n.Delete(
            askFor("Login: "),
            Convert.ToDateTime(askFor("Date (dd/mm/yyyy hh:mm:ss): "))));
    }

    private void testUpdateNote()
    {
        Console.Out.WriteLine(
                n.Update(
                    askFor("newLogin: "),
                    Convert.ToDateTime(DateTime.Now.ToShortDateString() +
DateTime.Now.ToShortTimeString()),
                    askFor("newText: "),
                    askFor("Login: "),
                    Convert.ToDateTime(askFor("Date (dd/mm/yyyy hh:mm:ss): "))));
    }

    private void testAuthenticate()
    {
        Console.Out.WriteLine(
            u.Autenticate(
            askFor("Login: "),
            askFor("Password: ")));
    }

    private void testGetName()
    {
        Console.Out.WriteLine(
            u.GetName(
            askFor("Login: ")));
    }


    private string askFor(string message)
    {
        string result = "";

        Console.Write(message);
        try
```

```
        {
            result = Console.In.ReadLine();
        }
        catch (Exception oops)
        {
            Console.Out.WriteLine(oops);
        }
        return result;
    }
}
```

Creando una aplicación Windows para acceder a la BD haciendo uso de la DLL (DiaryWin)

El interfaz de la aplicación a desarrollar es el siguiente:



Una vez que agreguemos la referencia a la DLL y hayamos creado un formulario con este aspecto, nos dispondremos a escribir código.

```
private Users users;
private Notes notes;

private void Form1_Load(object sender, EventArgs e)
{
    users = new Users();
    notes = new Notes();
}

private void loginBtn_Click(object sender, EventArgs e)
{
    if (users.Authenticate(loginTbx.Text, passwordTbx.Text))
        MessageBox.Show("Login successfully!", "Diary for Windows",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    else
        MessageBox.Show("Login error!", "Diary for Windows",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}

private void getNotes_Click(object sender, EventArgs e)
```

```
{
    notesDtv.DataSource = notes.getNotesByLoginDate(loginTbx.Text,
dateTimePicker1.Value);
}
```

Fíjese que tal y como lo hemos diseñado la DLL, podemos consultar las citas de un usuario aunque no se haya autenticado. Esta lógica de autenticación la vamos a meter en el servicio Web que haremos a continuación.

Creando un servicio Web de acceso a datos (DiaryWS)

El objetivo de este proyecto es hacer un servicio Web para acceder a los datos, como es lógico este servicio no dará acceso a toda la capa de datos, sino sólo a los datos (y funcionalidad) que queremos que se tenga acceso desde el Web. En este caso el servicio Web servirá para autenticar a usuarios, devolver las citas de un usuario sólo si este se ha autenticado previamente y permitir modificar las citas (también requiere autenticación).

Para hacer esto es necesario que el servicio Web guarde (en una sesión o en una cookie) el login del usuario autenticado:

```
[WebMethod(true)]
public bool Authenticate(string login, string password)
{
    bool result = Users.Authenticate(login, password);
    if (result)
        Login = login;
    return result;
}

[WebMethod(true)]
public DiaryDB.NotesDataTable FindNotesByLoginDate(string login, DateTime date)
{
    checkAuthentification();
    return Notes.getNotesByLoginDate(login, date);
}

protected void checkAuthentification()
{
    if (Login == null)
        //throw new Exception("Ha fallado la autenticación");
        throw new System.Security.SecurityException();
}
```

Fíjese, que al igual que la aplicación Windows, el servicio Web no sabe dónde están los datos, simplemente hace uso de una DLL que se encarga de acceder a los datos (estén estos en SQLServer, Oracle, MiniSQL, …).

Creando una aplicación Windows que consuma el servicio Web

El objetivo de este proyecto es modificar DiaryWin para que en lugar de utilizar la DLL haga uso del servicio Web.

Las modificaciones son pocas, basta con agregar la referencia Web del servicio que acabamos de crear, crear un objeto que apunte a dicho servicio e invocar sus operaciones, pero ahora el control de la autenticación la gestiona el servicio Web:

```
private localhost.DiaryWS ws;

private void Form1_Load(object sender, EventArgs e)
{
    ws = new localhost.DiaryWS();
    ws.CookieContainer = new System.Net.CookieContainer();
}

private void loginBtn_Click(object sender, EventArgs e)
{
    try
    {
        if (ws.Authenticate(loginTbx.Text, passwordTbx.Text))
        MessageBox.Show("Login successfully!", "Diary for Windows",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        else
        MessageBox.Show("Login error!", "Diary for Windows",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
    catch (Exception oops)
    {
        MessageBox.Show("Autentication needed!", "Diary for Windows",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}

private void getNotes_Click(object sender, EventArgs e)
{
    try
    {
        notesDtv.DataSource = ws.FindNotesByLoginDate(loginTbx.Text,
dateTimePicker1.Value);
    }
    catch (Exception oops)
    {
        MessageBox.Show("Autentication needed!", "Diary for Windows",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}
```