

Modificación Patrón Singleton

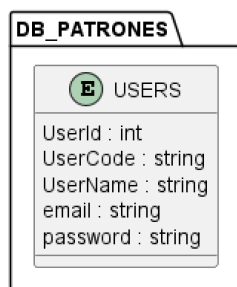
Objetivo

Crear aplicación que cargue la información de una base de datos. Si alguna propiedad es actualizada, debe reflejarse en la base de datos y al reiniciar el programa deberá leer el dato actualizado

Resolución

A continuación, se detallarán los pasos realizados para completar el objetivo de la tarea

- i. Creación de la base de datos y tabla a ser utilizada



El proyecto se realizó en Punto Net (C#)

- ii. Creación de la Entidad

```
class E_Users
{
    public int UserId { get; set; }
    public string UserCode { get; set; }
    public string UserName { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
}
```

- iii. Creación del acceso a base de datos

Aquí es donde configuramos el patrón Singleton

```
public class HelperSQLServer
{
    private static HelperSQLServer HSQLError { get; set; }
    private static SqlConnection oConnection { get; set; }

    private HelperSQLServer() { }

    public static HelperSQLServer getInstance() {
        if (HSQLError == null) {
            HSQLError = new HelperSQLServer();
        }
        return HSQLError;
    }

    public SqlConnection getConnection() {
        string connectionString = "Data Source=192.168.1.37;Initial
Catalog=DB_PATRONES;User ID=sa; Password=*****";

        if (oConnection == null) {
            oConnection = new SqlConnection(connectionString);
        }
    }
}
```

```

        oConnection.Open();
    }

    if(oConnection.State != System.Data.ConnectionState.Open) {
        while(true) {
            if(oConnection.State == System.Data.ConnectionState.Open) {
                break; }
            if(oConnection.State == System.Data.ConnectionState.Broken) {
                oConnection.Close(); oConnection.Open(); }
            if(oConnection.State == System.Data.ConnectionState.Closed) {
                oConnection.Open(); }
        }
    }
    return oConnection;
}
}

```

- iv. Configuración de los DAO para el acceso (consulta/modificación) de la base de datos

Aquí se utiliza la instancia del Singleton creado

```

class UsersDAO
{
    private SqlConnection oConnection;

    public UsersDAO() {
        oConnection = Singleton.HelperSQLServer.GetInstance().getConnection();
    }

    public DataSet getListUsers() {
        DataSet oDataSet = new DataSet();
        var oAdapter = new SqlDataAdapter("SELECT * FROM USERS", oConnection);
        oAdapter.Fill(oDataSet, "USERS");
        oAdapter.Dispose();
        return oDataSet;
    }

    public int setUsers(Entities.E_Users user) {
        int iResp = 0;
        string SQL = $" UPDATE USERS " +
            $"      SET UserName = '{user.UserName}', email = '{user.Email}', [password] = '{user.Password}' " +
            $"      WHERE UserId = '{user.UserId}' ";
        var oCommand = new SqlCommand(SQL, oConnection);
        iResp = oCommand.ExecuteNonQuery();

        return iResp;
    }
}

```

- v. Creación de la capa lógica de negocio

```

class Users {
    DataAccess.UsersDAO UserDAO;
    public Users() {
        UserDAO = new DataAccess.UsersDAO();
    }
    public DataSet getListUsers()
    {
        return UserDAO.getListUsers();
    }
    public int setUsers(Entities.E_Users user) {
        return UserDAO.setUsers(user);
    }
}

```

Patrón Singleton

La implementación del patrón singleton, lo realizamos dentro de la clase [HelperSQLServer](#) en la cual se tienen las variables de conexión y de instancia de la clase de forma estática

Para poder utilizar la clase lo realizamos por medio del método [getInstance\(\)](#), el cual es un método estático, que me creara la instancia a mi clase una única vez, y cada vez que se la invoque devolverá la instancia ya creada

La utilización de la instancia se deberá realizar dentro de las clases de tipo DAO; las cuales estarán las encargadas de realizar el CRUD contra la base de datos

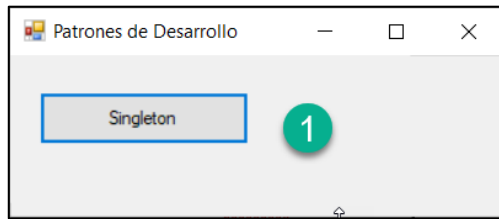
Una vez obtenida la instancia del manejador de base de datos, se deberá acceder al objeto de conexión a datos (el cual también fue definido para que sea único), es decir, también será de tipo singleton; en donde al llamar al método [getConnection\(\)](#) me devolverá la misma instancia para las conexiones hacia la base de datos

Una vez que se tienen la conexión a los datos, podremos ejecutar los CRUD por medio de los componentes correspondientes, ya sea mediante el [SqlCommand](#) o por el [SqlDataAdapter](#)

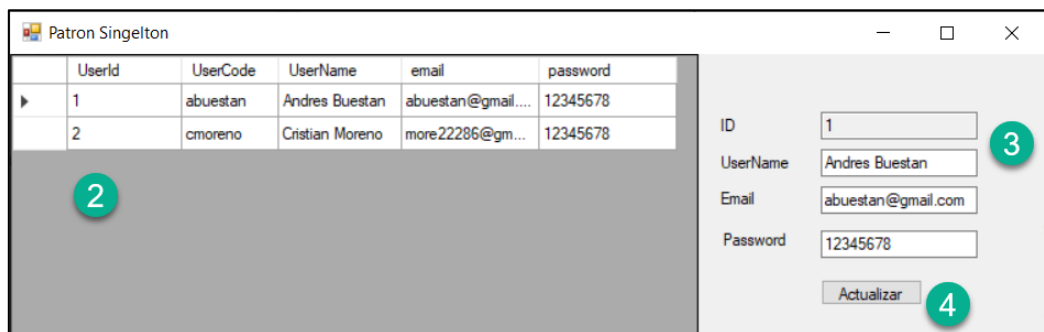
Los archivos de tipo DAO, deberán ser accedidos por los de tipo BL o Lógica de negocio; estos BL serán los que se invoquen en el Front para poder realizar la interacción entre el usuario y la base de datos

Resultado

Creación del Front, para que el usuario interactúe



1. Abrir el formulario para el mostrar los datos por medio del patrón Singleton



2. En el Grid se recuperan los datos de la base de datos; y al dar doble clic sobre un registro se habilitará la opción de modificación (derecha del formulario)
3. Modificación de los datos
4. Enviar los datos a la base de datos

