1) Crear GestorPresupuesto.cs en carpeta servicios (tiene una instancia dao de IPresupuestoDao.cs, se ocupa de la logica como que me diga el proximo presupuesto, creo un private IPresupuestoDAO dao siendo dao un objeto y le doy directamente la interfaz que contiene los metodos, uso la interfaz pq me obliga a tener los metodos hechos, dao es la interfaz que llama a PresupuestoDao.cs junto con sus metodos)

Captura de pantalla de computadora

Descripción generada automáticamente

2) Crear IPresupuestoDAO.cs en carpeta accesoDatos (interfaz que contiene los metodos que debe implementar PResupuestoDAO al acceder a datos, me obliga a implementar los metodos en PresupuestoDAO.cs)

Texto

Descripción generada automáticamente

3) Crear PresupuestoDAO.cs en carpeta accesoDatos (va a contener los metodos encargados del presupuesto que utilizan la BD y contienen codigo SQL)

Texto

Descripción generada automáticamente

APLICACIÓN DE ESTO SACANDO EL PROXIMO\_PRESUOPUESTO DESDE FORM

1)

Texto

Descripción generada automáticamente

2)

Texto

Descripción generada automáticamente

3)

Texto

Descripción generada automáticamente

4) Patron Factory para quitar new GestorPresupuesto()

Texto

Descripción generada automáticamente

5) Creamos patron factory dentro de carpeta accesoDatos

Primero)

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Segundo)

Texto

Descripción generada automáticamente

Tercero) Tengo un gestor de presupuesto que tiene un constructor, ese constructor tiene un factory y ese factory lo que hace es crear el presupuesto.

Texto

Descripción generada automáticamente

Cuarto)

Texto

Descripción generada automáticamente

6) Ahora lo mismo con el método CargarProducto() del Form

Captura de pantalla de un celular

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

7) Ahora con el botón de Aceptar que el confirmar se encuentra en la clase Presupuesto

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

8) Aplicar patron Singleton

En el Form solo quedan los métodos referentes a la Interfaz

El Gestor se va a encargar de mandar a llamar a las funciones

El DAO repite lo de creación de conexión por lo tanto deberíamos crear una clase manipuladora que me permite tener siempre una instancia

Primero creamos una clase en accesoDatos a la cual la pueda llamar desde el DAO y no tener que siempre repetir lo de la conexión.

Primero) Crear helperDAO que seria un gestor de DAO o ayudante de DAO en carpeta accesoDatos

En HelperDAO.cs vamos a hacer toda la funcionalidad genérica que tenga que ver con los datos, quitar todo lo que se repite en PresupuestoDAO referido a la conexión a la BD.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Segundo) Creamos el Patron SIngleton para la instancia pero ahora tengo que generar a partir de ahí todos los métodos necesarios que me den la funcionalidad.

Texto

Descripción generada automáticamente

Tercero) Crear los métodos de las consultas repetitivas

Captura de pantalla de computadora

Descripción generada automáticamente

Cuarto) Ejecutar este método desde PresupuestoDAO

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Quinto) Optimizar método de ConsultaSQL en HelperDAO

Captura de pantalla de computadora

Descripción generada automáticamente

Sexto) Crear método en HelperDAO para obtener el próximo ID

Texto

Descripción generada automáticamente

Séptimo) Refactorizar método de obtención de próximo ID en PresupuestoDAO

Captura de pantalla de un celular

Descripción generada automáticamente

Octavo) Crear método en HelperDAO para Crear nuevo Presupuesto

Noveno)

Refactorizar método de obtención de próximo ID en PresupuestoDAO