Primer Trabajo para entregar

Seminario de Lenguajes opción .NET

1er. Semestre - 2023

Importante: Desarrollar con la versión **.NET 7.0**. No deshabilitar **<ImplicitUsings>** ni **<Nullable>** en los archivos de proyecto (**.csproj**). Resolver todos los *warnings* del compilador respecto de las referencias **null**.

Fecha de publicación: 21/4/2023
Fecha límite para la entrega: 8/5/2023

La entrega se realizará por medio de un formulario de Google que se publicará más adelante. Se deberá entregar:

- El código de la solución completa en un único archivo .zip
- Un documento explicativo donde se detalle cómo probar la funcionalidad desarrollada desde Program.cs. Se debe explicitar código de ejemplo junto con la salida por consola producida.

SISTEMA PARA LA GESTIÓN DE SEGUROS

Una empresa de seguros tiene el propósito de desarrollar un sistema que permita el seguimiento de los siniestros de los vehículos asegurados. Este sistema deberá gestionar información sobre vehículos, titulares de vehículos (clientes de la empresa), pólizas, siniestros y terceros involucrados en los siniestros. Con el fin de garantizar una gestión eficiente de esta información, se ha decidido asignar un identificador numérico único (Id) a cada entidad. De esta manera, no se permitirá la existencia de dos instancias diferentes de la misma clase con el mismo Id. El Id será establecido en el momento de persistir la entidad, asegurándose de obtener un valor de Id único.

El valor de la propiedad Id se utilizará para establecer vínculos entre distintas entidades. Por ejemplo, si un titular es dueño de uno o varios vehículos, se establecerá un vínculo en la entidad "vehículo" a través de una propiedad llamada "TitularId", donde se registrará el Id del titular previamente ingresado en la base de datos de la empresa. Del mismo modo, en la propiedad "VehiculoId" de una póliza se incluirá el Id del vehículo asegurado. Este mismo tipo de vinculación se aplicará entre las entidades "Siniestro" y "Póliza", así como entre "Tercero" y "Siniestro".

A continuación se detalla la información que se maneja de cada entidad mencionada:

Titular: Id, DNI, apellido, nombre, dirección, teléfono y correo electrónico.

Vehículo: Id, dominio, marca, año de fabricación y titular (identificado por el id del titular en la base de datos de la empresa)

Póliza: Id, vehículo asegurado (identificado mediante el Id del vehículo en la base de datos de la empresa), valor asegurado, franquicia, tipo de cobertura (Responsabilidad Civil o Todo Riesgo), fecha de inicio de vigencia y fecha de fin de vigencia.

Siniestro: Id, póliza (identificada mediante el Id de la póliza en la base de datos de la empresa), la fecha de ingreso (tomada automáticamente por el sistema al momento de carga), fecha de ocurrencia, dirección del hecho y descripción del accidente.

Tercero: Id, DNI, apellido, nombre, teléfono, nombre de su aseguradora y siniestro en el que participó (identificado por medio del Id del siniestro en la base de datos de la empresa).

Codificar una solución Aseguradora con tres proyectos: Aseguradora.Aplicacion, Aseguradora.Repositorios y Aseguradora.Consola, utilizando como modelo de arquitectura limpia el visto en la teoría 7 de este curso.

Proyecto Aseguradora.Aplicacion

En el proyecto Aseguradora.Aplicacion codificar las 5 entidades mencionadas. Observar que Titular y Tercero pueden derivar de una clase distinta de object, que además puede ser abstracta.

Invalidar el método ToString() en las clases donde sea conveniente.

Usaremos inversión de dependencia, por lo tanto, se deben definir en este proyecto las interfaces necesarias para trabajar con los repositorios.

Codificar los casos de uso para realizar las altas, bajas, modificaciones y listado completo de titulares, vehículos y pólizas (dejaremos el manejo de siniestros y terceros para la próxima entrega). En todos los casos utilizar inyección de dependencias por constructor.

A modo de ejemplo se detallan las clases de los casos de uso relacionados con la gestión de pólizas (Definir los casos de uso relativos a los titulares y vehículos de manera análoga):

- Clase **AgregarPolizaUseCase** con el método **void Ejecutar(Poliza p)** que agrega la póliza **p** al repositorio de pólizas. El Id de **p** debe asignarse de manera incremental al momento de persistirse la información.
- Clase **ModificarPolizaUseCase** con el método **void Ejecutar(Poliza p)** que actualiza los datos de **p** en el repositorio de pólizas. La póliza se identifica por su Id. En caso de no existir ningúna póliza con ese Id se debe lanzar una excepción.
- Clase **EliminarPolizaUseCase** con el método **void Ejecutar(int Id)** que elimina del repositorio de pólizas a la póliza cuyo Id se recibe como parámetro. En caso de no existir ningúna póliza con ese Id se debe lanzar una excepción.
- Clase **ListarPolizasUseCase** con el método **List<Poliza> Ejecutar()** que devuelve la lista de todas las pólizas.

Nota: En el caso **ModificarTitularUseCase**, se prefiere que el método **void Ejecutar(Titular t)** utilice el **DNI** en lugar del **Id** para identificar al titular cuyos datos serán actualizados. Esta práctica difiere de la utilizada para las entidades póliza y vehículo.

Además, es necesario añadir un caso de uso llamado **ListarTitularesConSusVehiculosUseCase** que permita listar todos los titulares junto con la información detallada de sus vehículos asegurados. Es importante tener en cuenta que cada titular puede ser propietario de uno o varios vehículos. Una posible forma de resolverlo es contar con una propiedad de tipo List<Vehiculo> en la entidad Titular que puede usarse, cuando sea necesario, para agregar allí la lista de vehículos que posee el titular.

Proyecto Aseguradora.Repositorios

En el proyecto Aseguradora.Repositorios definir los repositorios concretos para persistir a los titulares, vehículos y pólizas (dejaremos los repositorios de siniestros y terceros para la próxima entrega). La persistencia se debe lograr utilizando archivos de texto plano.

Nota: Para actualizar los datos en los archivos de texto no se deben tener en cuenta aspectos de rendimiento. Por ejemplo, se podría sobrescribir el archivo completo sólo para modificar un registro en el mismo. En la próxima entrega vamos a utilizar un manejador de bases de datos para gestionar la persistencia de manera más eficiente.

Proyecto Aseguradora.Consola

El proyecto Aseguradora.Consola, se utilizará para interactuar con las otras partes de la solución y verificar el correcto comportamiento de las partes desarrolladas. En la próxima entrega vamos a crear una verdadera interfaz de usuario web utilizando Blazor.

A modo orientativo, observar que si el repositorio de titulares está vacío (aún no existe el archivo de texto correspondiente), el siguiente código en Program.cs producirá por la consola una salida como la indicada posteriormente.

```
//creamos los casos de uso inyectando dependencias
RepositorioTitular repoTitular = new RepositorioTitular();
AgregarTitularUseCase agregarTitular = new AgregarTitularUseCase(repoTitular);
ListarTitularesUseCase listarTitulares = new ListarTitularesUseCase(repoTitular);
ModificarTitularUseCase modificarTitular = new ModificarTitularUseCase(repoTitular);
EliminarTitularUseCase eliminarTitular = new EliminarTitularUseCase(repoTitular);

//Instanciamos un titular
Titular titular = new Titular(33123456, "García", "Juan")
{
   Direccion = "13 nro. 546",
   Telefono = "221-456456",
   Email = "joseGarcia@gmail.com"
};

Console.WriteLine($"Id del titular recién instanciado: {titular.Id}");

//agregamos el titular utilizando un método local
PersistirTitular(titular);

//el id que corresponde al titular es establecido por el repositorio
Console.WriteLine($"Id del titular una vez persistido: {titular.Id}");

//agregamos unos titulares más
PersistirTitular(new Titular(20654987, "Rodriguez", "Ana"));
PersistirTitular(new Titular(31456444, "Alconada", "Fermín"));
PersistirTitular(new Titular(12345654, "Perez", "Cecilia"));

//listamos los titulares utilizando un método local
ListarTitulares();

//no debe ser posible agregar un titular con igual DNI que uno existente
Console.WriteLine("Intentando agregar un titular con DNI 20654987");
titular = new Titular(20654987, "Alvarez", "Alvaro");
```

```
PersistirTitular(titular); //este titular no pudo persistirse

//Entonces vamos a modificar el titular existente
Console.WriteLine("Modificando el titular con DNI 20654987");
modificarTitular.Ejecutar(titular);

ListarTitulares();

//Eliminando un titular
Console.WriteLine("Eliminando al titular con id 1");
eliminarTitular.Ejecutar(1);

ListarTitulares();


//métodos locales
void PersistirTitular(Titular t)
{
   try
   {
       agregarTitular.Ejecutar(t);
   }
   catch (Exception e)
   {
       Console.WriteLine(e.Message);
   }
}

void ListarTitulares()
{
   Console.WriteLine("Listando todos los titulares de vehículos");
   List<Titular> lista = listarTitulares.Ejecutar();
   foreach (Titular t in lista)
   {
       Console.WriteLine(t);
   }
}
```

Salida por consola:

```
Id del titular recién instanciado: -1
Id del titular una vez persistido: 1
Listando todos los titulares de vehículos
1: 33123456 García, Juan 13 nro. 546 221-456456 joseGarcia@gmail.com
2: 20654987 Rodriguez, Ana
3: 31456444 Alconada, Fermín
4: 12345654 Perez, Cecilia
Intentando agregar un titular con DNI 20654987
Ya existe un titular con DNI = 20654987
Modificando el titular con DNI 20654987
Listando todos los titulares de vehículos
1: 33123456 García, Juan 13 nro. 546 221-456456 joseGarcia@gmail.com
2: 20654987 Alvarez, Alvaro
3: 31456444 Alconada, Fermín
4: 12345654 Perez, Cecilia
```

```
Eliminando al titular con id 1
Listando todos los titulares de vehículos
2: 20654987 Alvarez, Alvaro
3: 31456444 Alconada, Fermín
4: 12345654 Perez, Cecilia
```