Primer Trabajo para entregar

Seminario de Lenguajes opción .NET

2do. Semestre - 2024

Importante

Queremos recordarles la importancia de este primer trabajo del curso. Aunque no recibirá una calificación numérica, será objeto de corrección y dará lugar a una devolución que servirá como guía para continuar hacia el próximo trabajo.

La entrega de este primer trabajo es de carácter obligatorio y requisito excluyente para la aprobación del curso. Se considerará que los alumnos que no realicen esta entrega han decidido abandonar la materia.

La entrega debe ser significativa. No se admitirán entregas en blanco ni aquellas que muestren un esfuerzo mínimo, en lugar de un sincero compromiso y dedicación en la realización del trabajo.

NOTA: Desarrollar con la versión **.NET 8.0**. 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: 25/9/2024

Fecha límite para la entrega: 11/10/2024 hasta las 23:00 hs.

El trabajo puede realizarse en grupo de hasta 4 integrantes.

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
- 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.
- Subir la solución completa y el documento explicativo, todo comprimido en un único archivo de extensión zip. El nombre del archivo zip debe contener los apellidos de los autores del trabajo

Sistema de Gestión de Inventario (SGI)

Descripción general

El objetivo es desarrollar un sistema de gestión de inventario que permita la organización y administración efectiva de productos, categorías, y transacciones de entrada y salida de stock. Este sistema deberá gestionar la información de productos, asignarles una categoría, y registrar transacciones que afecten el stock.

Entidades

Se trabajará con cuatro entidades fundamentales: **Producto**, **Categoría**, **Transacción y Usuario**. Los elementos deben ser gestionados y persistidos de forma independiente, siguiendo el patrón de repositorios e inyección de dependencias visto en la teoría 7 de este curso.
Cada entidad estará caracterizada por una propiedad Id única, lo que facilitará su identificación entre otras instancias del mismo tipo.
A continuación se detallan los atributos de cada entidad

Producto:

- **Id**: Identificador único numérico.
- **Nombre**: Nombre del producto.
- **Descripción**: Texto que describe el producto.
- **PrecioUnitario**: Precio por unidad.
- **StockDisponible**: Número de unidades disponibles.
- **FechaCreacion**: Fecha y hora en que el producto fue creado.
- **FechaUltimaModificacion**: Fecha y hora de la última modificación.
- **CategoriaId**: Relación con la categoría a la que pertenece.

Categoría:

- **Id**: Identificador único numérico.
- **Nombre**: Nombre de la categoría.
- **Descripción**: Descripción de la categoría.
- **FechaCreacion**: Fecha y hora de creación.
- **FechaUltimaModificacion**: Fecha y hora de la última modificación.

Transacción:

- **Id**: Identificador único numérico.
- **ProductoId**: Relación con el producto involucrado.
- **Cantidad**: Número de unidades de la transacción.
- **Tipo**: Tipo de transacción (Entrada o Salida).
- **FechaTransaccion**: Fecha y hora en que se realizó la transacción.

Reglas de negocio

1. El sistema debe validar que el stock disponible sea suficiente al intentar dar de alta una transacción de salida. En caso contrario, se debe lanzar una excepción.
2. Al dar de alta una transacción de **entrada**, el stock del producto debe incrementarse. Al dar de alta una transacción de **salida**, el stock del producto debe decrementarse.
3. Al dar de baja una transacción de **entrada**, el stock del producto debe decrementarse. Al dar de baja una transacción de **salida**, el stock del producto debe incrementarse.
4. Las transacciones no se modifican, solo se dan de alta o de baja.
5. **Categorías** pueden ser asignadas a productos, y cada producto debe tener una categoría asignada. Las categorías se pueden modificar o dar de baja. Para dar de baja una categoría de debe verificar que no haya productos asignados.
6. Al dar de baja un producto, si tiene transacciones asociadas también se deben dar de baja.

Permisos de usuario

Cada usuario podrá poseer varios de los permisos que se detallan a continuación:

- ProductoAlta: Puede realizar altas de Productos
- ProductoBaja: Puede realizar bajas de Productos
- ProductoModificacion: Puede realizar modificaciones de Productos
- CategoriaAlta: Puede realizar altas de Categorías
- CategoriaBaja: Puede realizar bajas de Categorías
- CategoriaModificacion: Puede realizar modificaciones de Categorías
- TransaccionAlta: Puede realizar altas de Transacciones
- TransaccionBaja: Puede realizar bajas de Transacciones
- UsuarioAlta: Puede realizar altas de Usuarios
- UsuarioBaja: Puede realizar bajas de Usuarios
- UsuarioModificacion: Puede realizar modificaciones de Usuarios

Validaciones

1. **Productos**:
    - El **nombre** no puede estar vacío.
    - El **precio unitario** debe ser mayor a 0.
    - El **stock disponible** no puede ser negativo.
2. **Categorías**:
    - El **nombre** de la categoría no puede estar vacío.
3. **Transacciones**:
    - La **cantidad** debe ser mayor a 0.
    - En transacciones de salida, el **stock disponible** debe ser suficiente.

Manejo de excepciones

Se deberán manejar las siguientes excepciones:

- **ValidacionException**: Lanzada cuando los datos de producto, categoría o transacción no cumplen con las reglas de validación.

- **StockInsuficienteException**: Lanza cuando no hay suficiente stock disponible en una transacción de salida.
- **PermisosException**: Se lanza cuando un usuario intenta realizar una operación sin los permisos adecuados.

Detalles de implementación

El sistema se llamará **SGI** (Sistema de Gestión de Inventario) y tendrá tres proyectos:

1. **SGI.Aplicacion**: Gestionará la lógica de negocio, entidades y casos de uso.
    a. Para obtener un código desacoplado, implementar las validaciones de las entidades por medio de clases especializadas. Ejemplo, para validar la entidad **Producto** definir una clase **ProductoValidador**.

    b. Codificar también en este proyecto el enumerativo **TipoTransaccion**. Contendrá el valor para asignar a la propiedad Tipo (de la entidad **Transacción**). Además, implementar el enumerativo **Permiso**, el cual incluirá los valores correspondientes a los permisos que el sistema debe manejar.

    c. Codificar también en este proyecto un servicio de autorización para validar las credenciales del usuario al momento de realizar operaciones. El servicio debe implementar la siguiente interfaz:

```
public interface IServicioAutorizacion
{
   bool PoseeElPermiso(int IdUsuario, Permiso permiso);
}
```

    Para esta primera entrega, desarrollar un servicio de autorización provisional llamado **ServicioAutorizacionProvisorio**, que implemente la interfaz **IServicioAutorizacion**. Este servicio siempre debe responder **true** cuando el **IdUsuario** sea igual a **1** y **false** en cualquier otro caso (no hace falta realizar ninguna verificación). Utilizaremos este servicio provisional temporalmente hasta que la gestión completa de usuarios esté implementada en la próxima entrega. En ese momento, se reemplazará **ServicioAutorizacionProvisorio** por un servicio de autorización definitivo que verificará adecuadamente si el usuario tiene el permiso por el cual se está consultando.

    d. Para acceder a los repositorios usaremos inversión de dependencia, por lo tanto, se deben definir en este proyecto las interfaces necesarias para trabajar con ellos.

    e. Definir también en este proyecto las excepciones mencionadas anteriormente.

    f. Codificar los casos de uso para realizar las operaciones que se detallan para cada entidad.

    Producto:
        - Alta

- Baja (Debe eliminar las Transacciones asociadas al mismo)
- Modificación

Categoría:

- Alta
- Baja (Solo se permite si no está asignada a un producto)
- Modificación

Transacción:

- Alta
- Baja

g. Las operaciones de eliminación, se llevarán a cabo utilizando el **Id** de la entidad que se desea eliminar y que se pasa como parámetro.

h. Se debe garantizar que para las operaciones de alta, baja y modificación se verifique la autorización del usuario antes de proceder. Por lo tanto, el método Ejecutar de estos casos de uso deberá recibir también el **Id** del usuario como parámetro.

i. Se debe implementar un caso de uso que permita la consulta de una transacción (por medio del id) que incluya el detalle completo del producto.

j. Se debe desarrollar un caso de uso para listar todas las transacciones.

k. Se debe desarrollar un caso de uso para listar todas las transacciones dentro de un rango de fechas.

2. **SGI.Repositorios**: En el proyecto **SGI.Repositorios** definir los repositorios provisorios para persistir las entidades (Producto, Categoría y Transacción). La persistencia en estos repositorios debe implementarse utilizando archivos de texto plano. En la próxima entrega reemplazaremos estos repositorios provisorios por otros definitivos que accedan a una base de datos **SQLite**. Respecto a las altas, es responsabilidad de los repositorios determinar el **Id** con el que se persistirán las entidades, asegurándose de que este sea incremental y garantizando su unicidad.

   Importante: Se debe desarrollar un repositorio por cada entidad, esto aporta alta cohesión, menor acoplamiento, mayor testabilidad y mayor escalabilidad.

   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.

3. **SGI.Consola**: El proyecto **SGI.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.