Frameworks de développement

TP4 suite Generic Repositories

Le **Repository pattern** se concentre sur l'accès aux données, en laissant la logique métier à la couche de service (souvent appelée couche Business) dans le but de centraliser la logique d'accès aux données et cacher les données d'implémentation.

Structure du Repository Pattern

- **IGenericRepository :** Interface qui définit les méthodes de base pour accéder aux données.
- **Repository :** Classe qui implémente l'interface IRepository et qui contient la logique d'accès aux données (en utilisant un ORM comme Entity Framework Core par exemple).
- **Entity :** Classe représentant l'entité métier.
- **DbContext :** Classe qui gère l'accès à la base de données.

C'est à vous

L'objectif de cette partie est de réaliser un repository générique (suite du TP4) pour éliminer la redondance du code (réutilisation du code par plusieurs entités de l'application).

1. Créer une interface pour définir les méthodes de base qui gèrent les différentes entités de type T
   public interface IGenericRepository<T> where T : class {}
2. Implémenter le *Repository* générique qui contient la logique d'accès aux données en utilisant EF Core
   public class GenericRepository<T> : IGenericRepository<T> where T : class { private readonly DbContext _context; private readonly DbSet<T> _dbSet;……..
3. Créer les services pour accéder aux données via *repository.*
4. Ajouter des *Repositories* spécifiques pour gérer les relations entre les entités.
5. Modifier les contrôleurs pour accéder aux services.
6. N'oubliez pas de configurer les services dans program.cs

```
//Register Generic Repository
builder.Services.AddScoped(typeof(IGenericRepository<>), typeof(GenericRepository<>));
```