Laboratoire: héritage

# Objectif(s)

* Héritage :
  + Patterns d’implantation
  + Syntaxe Entity Framework

# Choisir l’implémentation appropriée

Chaque équipe doit compléter le diagramme de classes UML de l’étude de cas, choisir l’implantation appropriée et justifier son choix.

1. Lire l’étude de cas
2. Créer un diagramme de classes (Visio)
3. Compléter le diagramme
4. Choisir l’implantation
   * Implémentation A : Une table par enfant (contenant les propriétés du parent)
   * Implémentation B : Une table
   * Implémentation C : Une table pour chaque classes, parent et enfants
5. Justifier votre choix

## Étude de cas A

L’application gère les informations sur les employés et les clients, dont le prénom, nom, adresse complète (avec liste de domaine pour la ville et le code postal) et les numéros de téléphone (1 à plusieurs). Pour les employés plusieurs informations sont sensibles : Numéro d’assurance social NAS, le salaire et le statut familiale. D’autres informations sont accessibles par les autres employés tel que le poste téléphonique, le courriel, le département auquel il est rattaché. Les informations des clients ne sont accessibles que par les départements des ventes et de la facturation, soit le nom de l’entreprise, le statut (VIP, Construction, etc.), le volume d’achats, la spécialité de l’entreprise et le nombre d’employés.

|  |  |  |
| --- | --- | --- |
| **Implémentation choisie:** |  | |
| **Quel(s) DbSet sont requis :**  public DbSet<Classe> NomTable { get; set; } | |  |
| **Fluent API requis (si nécessaire) :**  modelBuilder.Entity<Parent>()  .HasDiscriminator<string>("NomDiscriminator")  .HasValue<Parent>("Nom\_base")  .HasValue<Enfant1>("Enfant1\_spec")  .HasValue<Enfant2>("Enfant2\_spec");  OU  modelBuilder.Entity<CParent>().ToTable("TParent");  modelBuilder.Entity<CEnfant1>().ToTable("Tenfant1");  modelBuilder.Entity<CEnfant2>().ToTable("TableEnfant2"); | |  |

**Justification du choix :**

|  |
| --- |
|  |

## Étude de cas B

L’application gère les informations produits tel que le code du fournisseur, la description, le prix coutant et le prix de vente. Il y a trois sortes de produits : les ordinateurs pour lesquels les informations suivantes sont requises : la mémoire ram, la capacité du disque dur, la marque, le nombre de cœurs, la vitesse du processeur, la hauteur, le type de carte vidéo, le type de carte de son; les télévisions pour lesquelles nous avons besoin de la grandeur, du modèle, du nombre et du type de connecteur(s), wifi et/ou bluetooth, la résolution maximum; et les imprimantes pour lesquelles nous avons besoin de la grandeur, du nombre de plateau(x), des formats de papiers possibles, wifi et/ou bluetooth, du modèle.

|  |  |  |
| --- | --- | --- |
| **Implémentation choisie:** |  | |
| **Quel(s) DbSet sont requis :**  public DbSet<Classe> NomTable { get; set; } | |  |
| **Fluent API requis (si nécessaire) :**  modelBuilder.Entity<Parent>()  .HasDiscriminator<string>("NomDiscriminator")  .HasValue<Parent>("Nom\_base")  .HasValue<Enfant1>("Enfant1\_spec")  .HasValue<Enfant2>("Enfant2\_spec");  OU  modelBuilder.Entity<CParent>().ToTable("TParent");  modelBuilder.Entity<CEnfant1>().ToTable("Tenfant1");  modelBuilder.Entity<CEnfant2>().ToTable("TableEnfant2"); | |  |

**Justification du choix :**

|  |
| --- |
|  |

## Étude de cas C

L’application gère les informations sur les items loués par la bibliothèque : les livres pour lesquels nous devons gérer le titre, le ISBN, l’auteur(s), la catégorie, l’année d’édition; les revues pour lesquelles nous devons gérer le titre, le volume, la fréquence, la date de publication; les films pour lesquels nous devons gérer le titre, la durée, l’année de parution et le format.

|  |  |  |
| --- | --- | --- |
| **Implémentation choisie:** |  | |
| **Quel(s) DbSet sont requis :**  public DbSet<Classe> NomTable { get; set; } | |  |
| **Fluent API requis (si nécessaire) :**  modelBuilder.Entity<Parent>()  .HasDiscriminator<string>("NomDiscriminator")  .HasValue<Parent>("Nom\_base")  .HasValue<Enfant1>("Enfant1\_spec")  .HasValue<Enfant2>("Enfant2\_spec");  OU  modelBuilder.Entity<CParent>().ToTable("TParent");  modelBuilder.Entity<CEnfant1>().ToTable("Tenfant1");  modelBuilder.Entity<CEnfant2>().ToTable("TableEnfant2"); | |  |

**Justification du choix :**

|  |
| --- |
|  |