

TP N° 9

Application web MVC avec Entity Framework Core

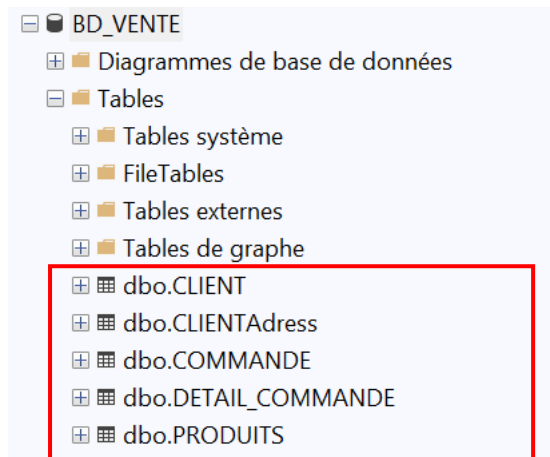
Approche Database First

Partie 1 : Utiliser la Base de Données BD_VENTE (TP N°5):

1.1. Utilisez la base de données BD_VENTE déjà créée :

```
USE BD_VENTE;  
GO
```

1.2. Utilisez les tables PRODUITS, CLIENT, COMMANDE, DETAIL_COMMANDE, CLIENTADDRESS déjà réalisées :



Partie 2 : Créer le projet ASP.NET Core MVC

- Ouvrir Visual Studio
- Aller dans Fichier > Nouveau > Projet
- Choisir : **ASP.NET Core Web App (Model-View-Controller)**
- Cliquer sur Suivant
- Nom du projet : **WebApplication_achat**
- Cliquer sur Créer
- Choisir le Framework .NET 9
- Laisser les options par défaut, puis cliquer sur Créer

Partie 3 : Installer les packages NuGet pour EF Core

Méthode 1 – Interface graphique : Outils > Gestionnaire de packages NuGet > Gérer les packages NuGet pour la solution
Chercher et installer ces packages :

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.VisualStudio.Web.CodeGeneration.Design

- Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore

Méthode 2 – Console : Outils > Gestionnaire de packages NuGet > Console

- Install-Package Microsoft.EntityFrameworkCore
- Install-Package Microsoft.EntityFrameworkCore.SqlServer
- Install-Package Microsoft.EntityFrameworkCore.Tools
- Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design
- Install-Package Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore

NB:

En cas de problème, les versions compatibles avec .NET 9.0 testées:

- Microsoft.EntityFrameworkCore.Design Version=9.0.0
- Microsoft.EntityFrameworkCore.SqlServer Version=9.0.0
- Microsoft.EntityFrameworkCore.Tools Version=10.0.0
- Microsoft.VisualStudio.Web.CodeGeneration.Design Version=9.0.0
- Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore Version=9.0.0

Pour verification:

- Projet>Modifier le fichier projet (.csproj)

```
<Project Sdk="Microsoft.NET.Sdk.Web">
<PropertyGroup>
  <TargetFramework>net9.0</TargetFramework>
  <Nullable>enable</Nullable>
  <ImplicitUsings>enable</ImplicitUsings>
</PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="9.0.0">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="9.0.0"
/>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="10.0.0">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design"
Version="9.0.0" />
    <PackageReference
Include="Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore" Version="9.0.0"
/>
  </ItemGroup>

</Project>
```

A éviter :

- ⇒ Ne pas oublier d'installer EFCore.Tools, sinon Scaffold-DbContext ne fonctionnera pas.
- ⇒ Ne jamais modifier directement DbContext.cs généré (on utilisera partial ou configuration séparée si besoin).

Partie 4 : Générer les modèles depuis la base de données

Commande Scaffold pour générer automatiquement :

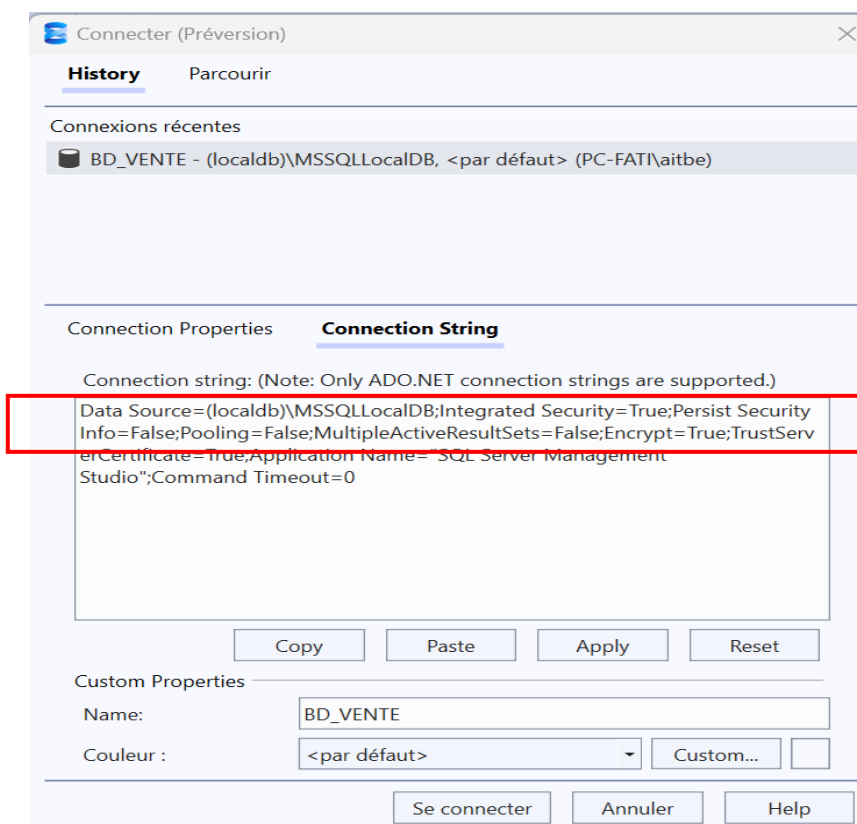
- Le **DbContext** (VenteContext)
- Les **classes de modèles** (par exemple Produit.cs) à partir d'une base de données existante.

⇒ C'est l'approche **Database First** d'Entity Framework Core.

1. Commande Database First, dans la Console du Gestionnaire de Package :

```
Scaffold-DbContext "Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=BD_VENTE;Integrated
Security=True;TrustServerCertificate=True"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -
Context VenteContext
```

- Vérifiez Data Source depuis la connexion à votre serveur sur SSMS



⇒ Résultat :

- Un fichier VenteContext.cs (DbContext)
- Les modèles qui représentent chaque table, fichier Produit.cs,...

| Élément | Description |
|---|--|
| Scaffold-DbContext | Commande de scaffolding Database First |
| "Data Source=...;Initial Catalog=...;" | Chaîne de connexion vers la base SQL Server |
| Microsoft.EntityFrameworkCore.SqlServer | Provider EF Core pour SQL Server |
| -OutputDir Models | Dossier de sortie où générer les classes de modèle |
| -Context VenteContext | Nom à donner à la classe DbContext générée |

Partie 5 : Configurer la connexion dans appsettings.json

- Ajouter la partie suivante dans le fichier json existant :

```
"ConnectionStrings": {  
  "VenteDb": "Data Source=(localdb)\\MSSQLLocalDB;Initial  
Catalog=BD_VENTE;Integrated Security=True;TrustServerCertificate=True"  
}
```

Partie 6 : Ajouter le DbContext dans Program.cs

Dans Program.cs, **avant** `var app = builder.Build();`

```
builder.Services.AddDbContext<VenteContext>(options =>  
options.UseSqlServer(builder.Configuration.GetConnectionString("VenteDb")));
```

Attention :

- ⇒ Cette ligne doit être avant **builder.Build();**
- ⇒ S'assurer que **using Microsoft.EntityFrameworkCore;** est présent

Partie 7 : Créer les contrôleurs avec vues (scaffolding)

Les Étapes :

1. Clic droit sur Controllers → Ajouter → Contrôleur
2. Choisir : **MVC Controller with views, using Entity Framework**
3. Sélectionner :
 - **Classe modèle** : Produit
 - **DbContext** : VenteContext
 - Nom du contrôleur : ProduitsController
4. Valider → Visual Studio génère :
 - ProduitsController.cs
 - Les vues Razor (Index, Create, Edit, Details, Delete)
5. Répéter les étapes pour les autres contrôleurs

Partie 8 : Exécuter l'application

- Lancer l'application
- Naviguer vers : `https://localhost:xxxx/Produits`
- Visualiser la liste des produits (avec CRUD complet)

Exemple de résultat avec CRUD :

Index - WebApplication_achat

localhost:7112/Produits

WebApplication_achat

Home

Privacy

Index

Create New

| Nom | Description | Prix | Quantite | DateAjout | Disponible |
|---------------------|---------------------------------|---------|----------|------------|---|
| Ordinateur Portable | PC 15 pouces, i5, 8Go RAM | 7990,00 | 12 | 02/11/2025 | <div><input checked="" type="checkbox"/></div> <div>Edit Details Delete</div> |
| Souris Optique | Souris USB avec capteur optique | 19,99 | 50 | 02/11/2025 | <div><input checked="" type="checkbox"/></div> <div>Edit Details Delete</div> |
| Imprimante HP | Imprimante couleur jet d'encre | 1200,00 | 25 | 08/11/2025 | <div><input checked="" type="checkbox"/></div> <div>Edit Details Delete</div> |
| Écran Dell 24" | Écran Full HD 24 pouces | 1850,50 | 40 | 08/11/2025 | <div><input checked="" type="checkbox"/></div> <div>Edit Details Delete</div> |
| Clavier Logitech | Clavier sans fil | 350,00 | 75 | 08/11/2025 | <div><input checked="" type="checkbox"/></div> <div>Edit Details Delete</div> |

Partie 9 : Ajouter barre de recherche dans la page Produits

1. Modifier la vue **Razor** (Index.cshtml)

- Avant le `<table>`, ajouter ce formulaire :

```
<form asp-action="Index" method="get" class="form-inline mb-3">
  <div class="form-group">
    <input type="text" name="searchString" value="@ViewData["CurrentFilter"]"
    class="form-control" placeholder="Rechercher un produit" />
    <button type="submit" class="btn btn-primary ml-2">Rechercher</button>
  </div>
</form>
```

⇒ Cela envoie la recherche à `ProduitsController.Index(string searchString)` via GET

2. Modifier l'action Index dans **ProduitsController.cs** :

Dans le contrôleur, modifier l'action **Index** comme ceci :

```
public async Task<IActionResult> Index(string? searchString)
{
    var produits = from p in _context.Produits
                   select p;

    if (!string.IsNullOrEmpty(searchString))
    {
        produits = produits.Where(p => p.Nom.Contains(searchString));
    }

    ViewData["CurrentFilter"] = searchString;

    return View(await produits.ToListAsync());
}
```

- ⇒ **_context** est votre **VenteContext** injecté dans le contrôleur
- ⇒ Le champ de recherche filtre les produits par Nom

3. Résultat visuel

Index

[Create New](#)

| Nom | Description | Prix | Quantite | DateAjout | Disponible | |
|---------------------|---------------------------------|---------|----------|------------|-------------------------------------|---|
| Ordinateur Portable | PC 15 pouces, i5, 8Go RAM | 7990,00 | 12 | 02/11/2025 | <input checked="" type="checkbox"/> | Edit Details Delete |
| Souris Optique | Souris USB avec capteur optique | 19,99 | 50 | 02/11/2025 | <input checked="" type="checkbox"/> | Edit Details Delete |
| Imprimante HP | Imprimante couleur jet d'encre | 1200,00 | 25 | 08/11/2025 | <input checked="" type="checkbox"/> | Edit Details Delete |
| Écran Dell 24" | Écran Full HD 24 pouces | 1850,50 | 40 | 08/11/2025 | <input checked="" type="checkbox"/> | Edit Details Delete |
| Clavier Logitech | Clavier sans fil | 350,00 | 75 | 08/11/2025 | <input checked="" type="checkbox"/> | Edit Details Delete |