

JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 1 / 59

Auteur du document

JC Cherid Analyste Solutions	 
------------------------------------	--

Présents

Prénom Nom	Fonction	Société

Destinataires

Prénom Nom	Fonction	Société
Tous		

Historique des versions

Version	Date	Auteur	Objet
1	02/02/2023	JCC	Création/Modification du document



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 2 / 59

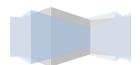
Sommaire

I – Formation C# – lien.....	4
II – Introduction.....	4
III – Rappels sur DotNet	5
IV – Rappel : Présentation de l'architecture hexagonale	6
V – Création de la solution « API.Demo.Param.CleanCode »	7
VI – Cr éation d'un dossier « Libs ».....	9
VII – Cr éation du projet « API.Demo.Param.CleanCode.Domain »	10
VII – A Cr éer la description de la table « CarBrand »	12
VII – B Cr éer la description de la table « CarModels»	13
VII – C Cr éer la description de la table « Car».....	14
VIII – Cr éation du projet « API.Demo.Param.CleanCode.Framework »	15
VIII –A Ajout de l'interface « IUnitOfWork »	16
VIII –B Ajout de l'interface « IRepository»	18
IX – Cr éation du projet « API.Demo.Param.CleanCode.Infrastructure »	20
IX –A Ajout des diff érents NuGets.....	21
IX –A -1 Ajout du NuGet « EntityFrameworkCore »	21
IX –A -2 Ajout du NuGet « EntityFrameworkCore.Design ».....	22
IX –A -3 Ajout du NuGet « EntityFrameworkCore.Relational ».....	22
IX –A-4 Ajout du NuGet « EntityFrameworkCore.Sqlite»	22
IX –A-5 Ajout du NuGet « EntityFrameworkCore.SqlServer».....	22
IX –B Cr éation du dossier « Data».....	23
IX –C Cr éation du dossier « TypeConfiguration» dans le dossier « Data ».....	24
IX –D Ajout de la configuration « CFG_CarBrand » pour la table « CarBrand » dans le dossier « TypeConfiguration »	25
IX –E Ajout de la configuration « CFG_CarModels » pour la table « CarModels » dans le dossier « TypeConfiguration »	27
IX –F Ajout de la configuration « CFG_Car » pour la table « Car » dans le dossier « TypeConfiguration ».....	28
IX –G Mise en place du « context » dans la classe « DataContext »	29
X –L'interface « IRepository » dans le projet « .Domaine ».....	30
XI – La classe « DefaultParamRepository » dans le projet « .Infrastructure »	32
XI –A Cr éer le dossier « Repositories » dans le projet « API.Demo.Param.CleanCode.Infrastructure »	32
XI –B La classe « DefaultParamRepository » (dans le dossier « Repositories »)	33
XI – B - 1 Les marques de voitures	33
XI – B - 2 Les modèles de voitures	35



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 3 / 59

XII – Configurer les Connection Strings (« appsettings.json »).....	37
XIII – Classe « Program.cs » (.Net6)	38
XIII –A Mettre en place l'injection de dépendance.....	38
XIII –B Connecter le « context » ef Core avec la chaîne de connexion décrite dans « appsettings.json »	39
XIII –C Créer la base de données (si elle n'existe pas déjà) au lancement de l'appli	40
XIV – Les DTO (Data Transfert Object).....	41
XIV –A Créer le dossier « Applications/DTOs ».....	41
XIV –B Créer le DTO « CarBrandDTO ».....	42
XIV –C Créer le DTO « CarBrandResumeDTO »	44
XIV –D Créer le DTO « CarModelDTO »	45
XIV –E Créer le DTO « CarDTO ».....	46
XV Crédit des méthodes dans le Contrôleur API	47
XV-A - Crédit du contrôleur.....	47
XV-B – Ajout du constructeur	49
XV-C – La méthode « AddCarBrand » (ajout d'une marque de voiture).....	50
XV-D – La méthode « UpdCarBrand » (Mettre à jour une marque de voiture)	51
XV-E – La méthode « GetCarBrand » (Récupérer marque(s) de voiture).....	52
XVI Erreurs Exécution (au premier lancement).....	53
XVI – A Erreur à la création de la base de données	53
XVI – A-1 Mauvaise syntaxe de « HasDefaultValueSql » dans la classe de configuration	54
XVI – A-2 Bonne syntaxe de « HasDefaultValueSql » dans la classe de configuration	55
XVI – B Swagger : Impossible de charger la définition	56
XVII Liens Divers.....	58
XVII – A dotnet efcore (liens documentation)	58
XVII – B Github (Dans Visual studio).....	59



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 4 / 59

I – Formation C# – lien

<https://www.udemy.com/course/maitriser-web-api-rest-avec-aspnet-core-dotnet-50-full/learn/lecture/25124540#overview>

Profil de connexion : lianazel@me.com
mot de passe : B@listE599

II – Introduction

- Synthèse qui récapitule la mise en œuvre de la méthode de développement **Clean-Code** (ou **architecture hexagonale**) à travers une petite solution.
- La solution « **API.Demo.Param.CleanCode** » est une petite solution dans laquelle une **API Rest** met à jour deux fichiers paramètre « **CarBrand** » (Marque de voiture) et « **CarModels** » (Modèles de voiture)
- Cette petite solution est développée avec **Visual Studio 2022 (.Net 6)**

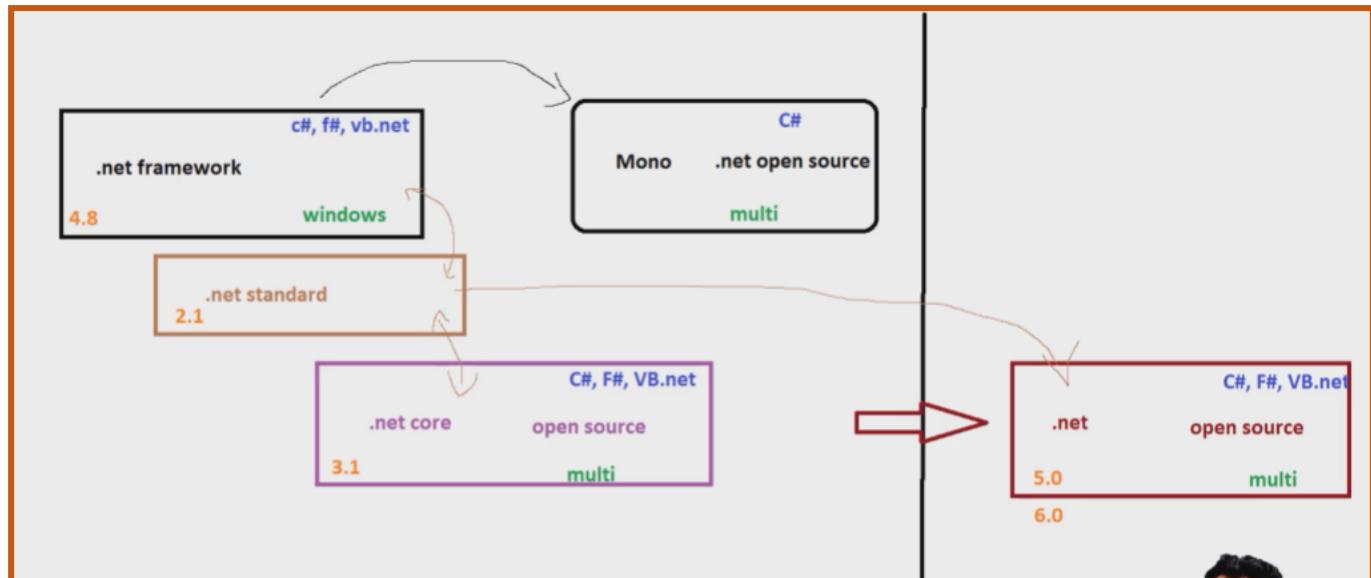


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 5 / 59

III – Rappels sur DotNet

Multi → Multi plateforme

Microsoft a décidé en 2020 de tout simplifier avec la création de .Net 5.0, puis 6.0 en 2022.



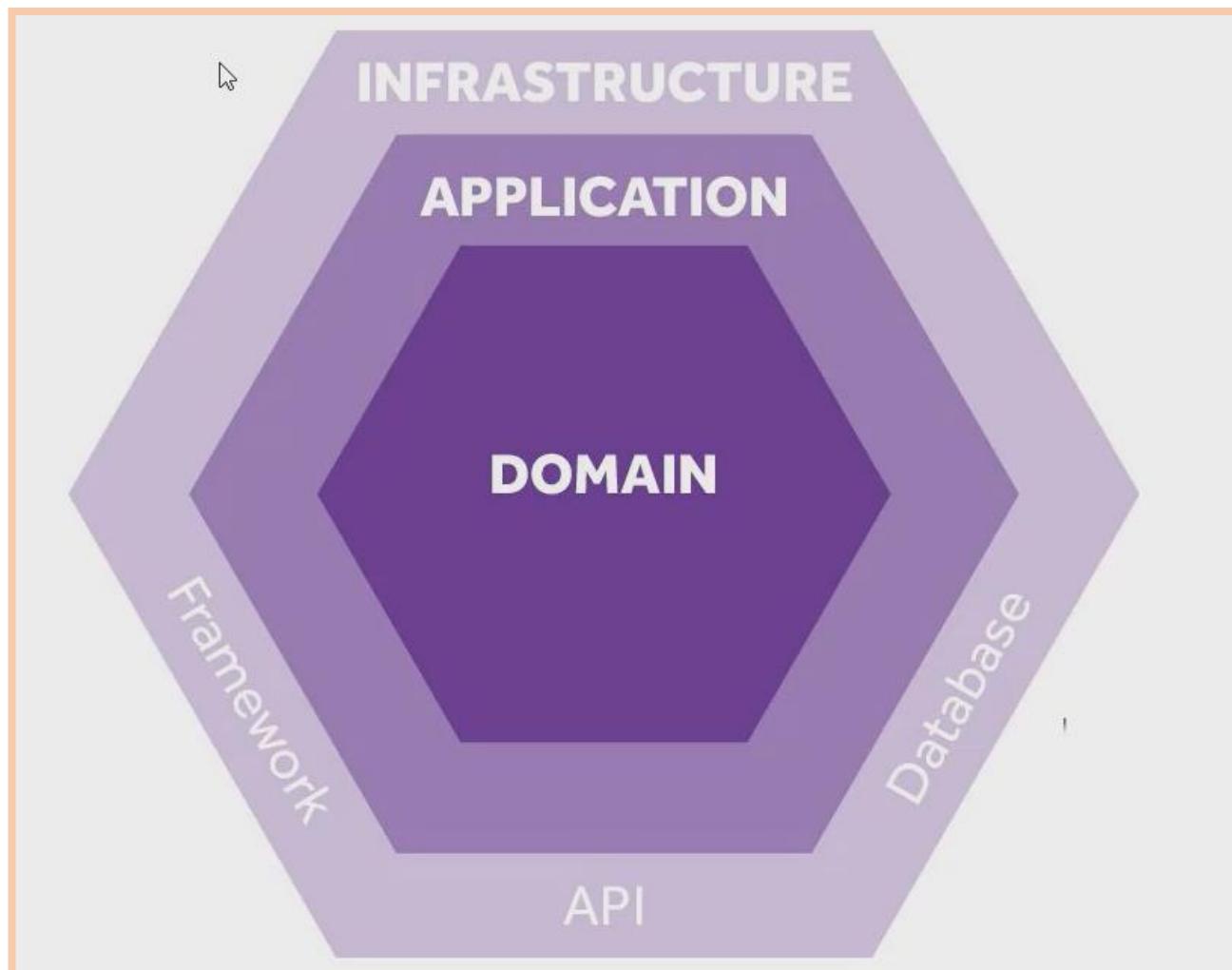
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 6 / 59

IV – Rappel : Présentation de l'architecture hexagonale

L'infrastructure d'une application, les bases de données, les appels API, de même que les frameworks évoluent constamment. Mais ce n'est pas forcément le cas du besoin métier.

En partant de ce principe, quelle solution est envisageable afin de rendre une application maintenable dans le temps, d'avoir un besoin métier complètement agnostique des choix techniques de son implémentation ?

Pour visualiser l'architecture hexagonale on la représente souvent par un hexagone avec au coeur de cet hexagone le Domain. La couche du dessus correspond à l'Application et le tout est entouré de l'Infrastructure.



Rappel : Le **DOMAINE** fait référence aux modèles de données (qui « parle » d'un métier).

Exemple : Domaine BANCAIRE ;

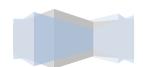
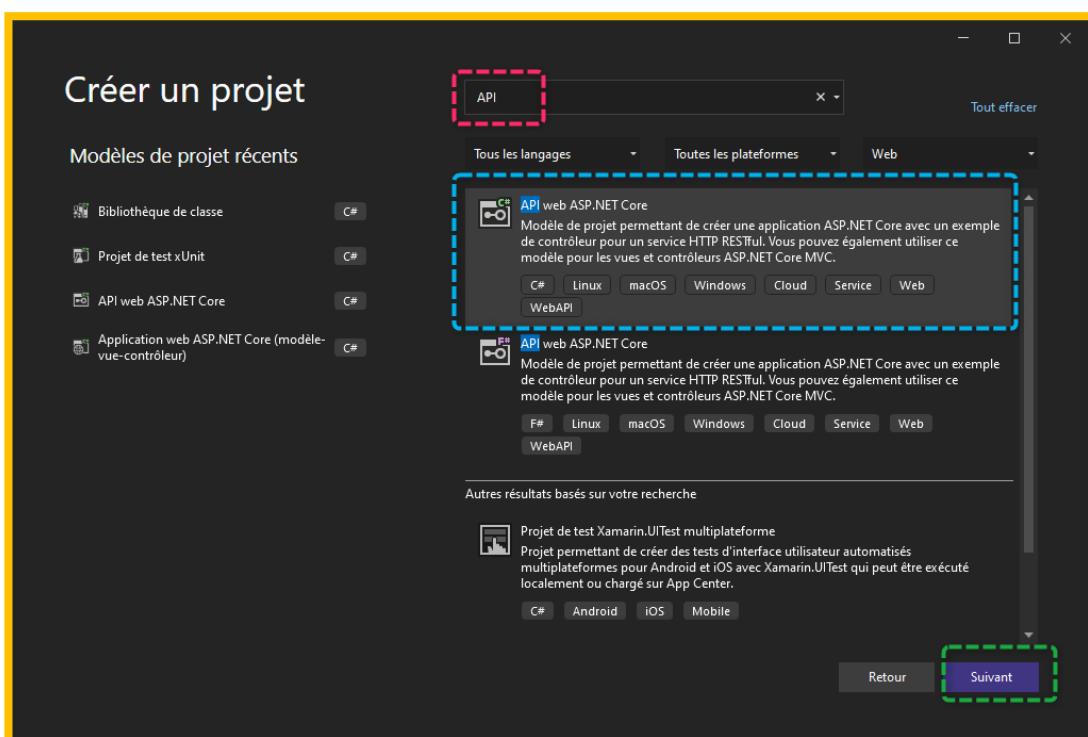
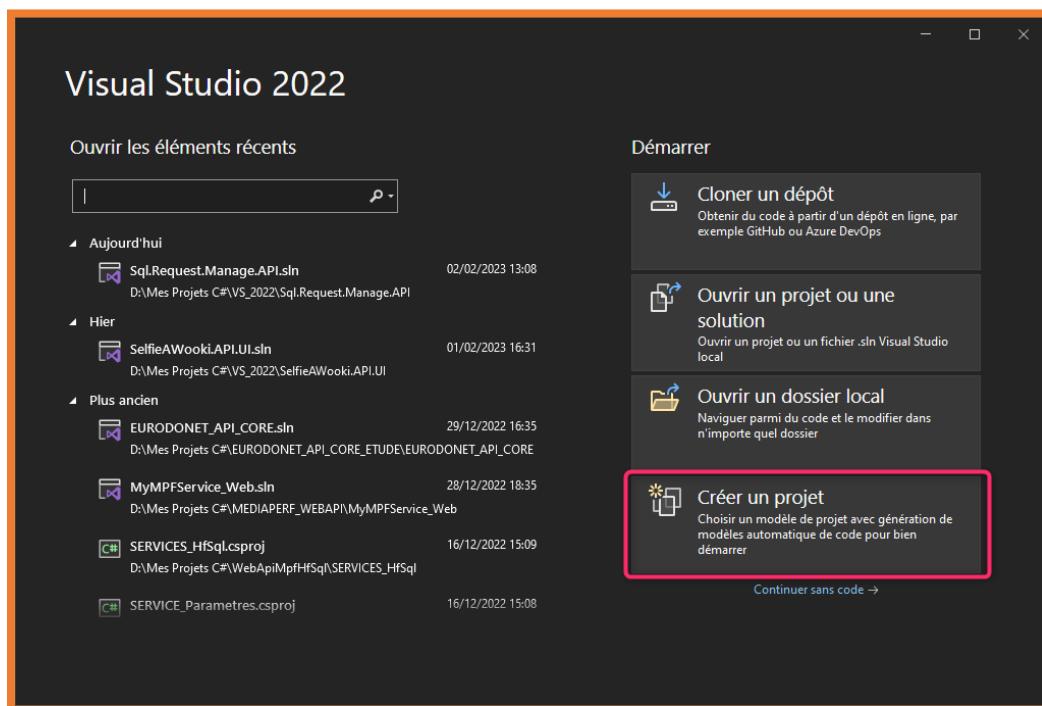
Domaine LOGISTIQUE ;

Etc...

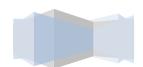
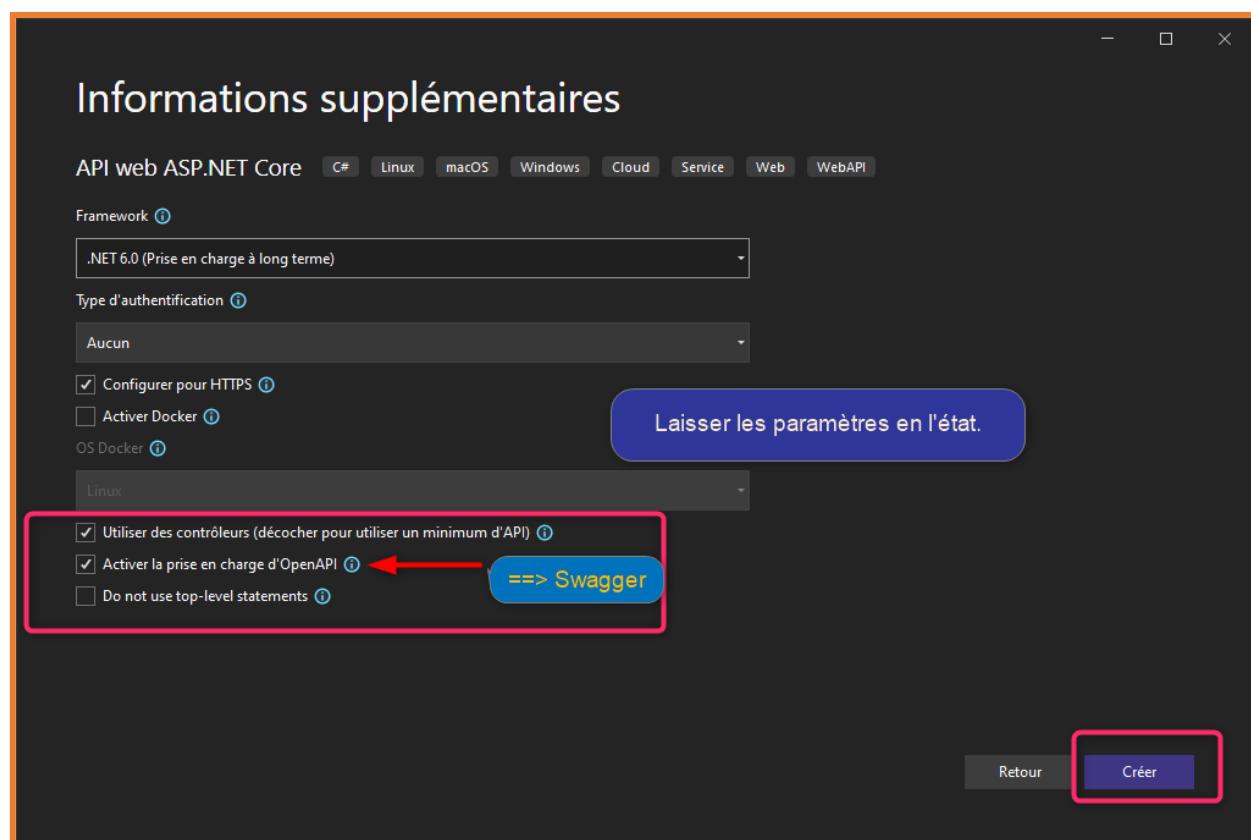
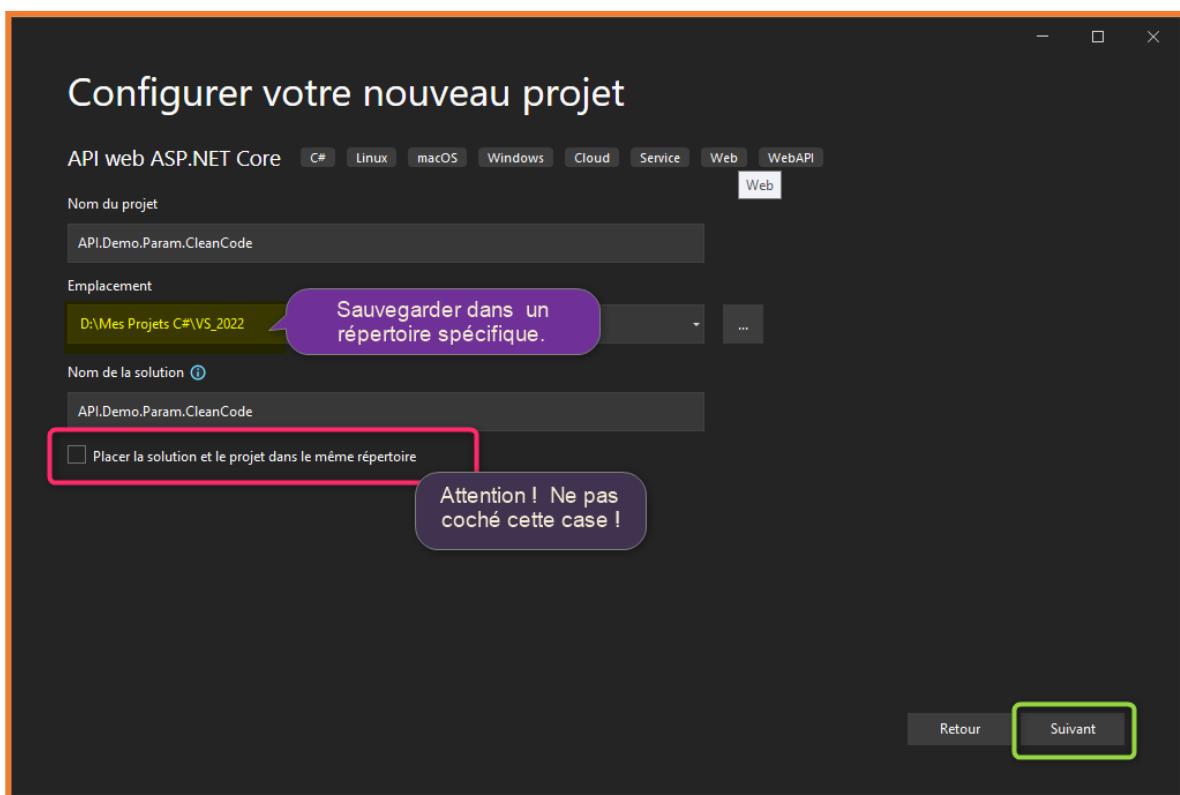


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 7 / 59

V - Création de la solution « API.Demo.Param.CleanCode »

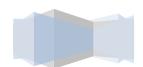
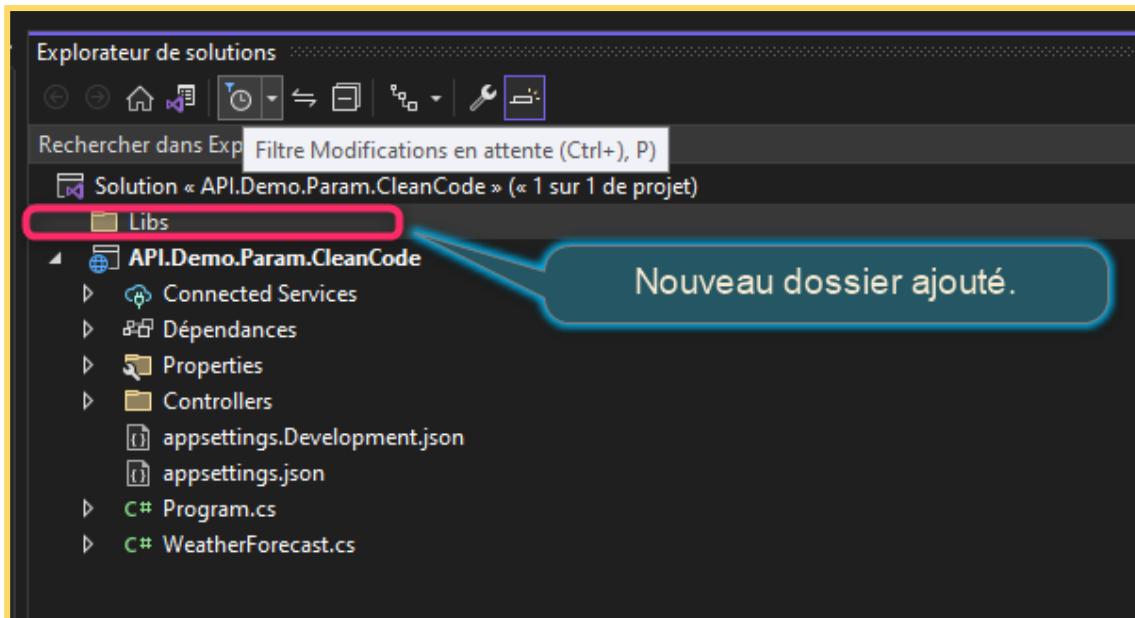
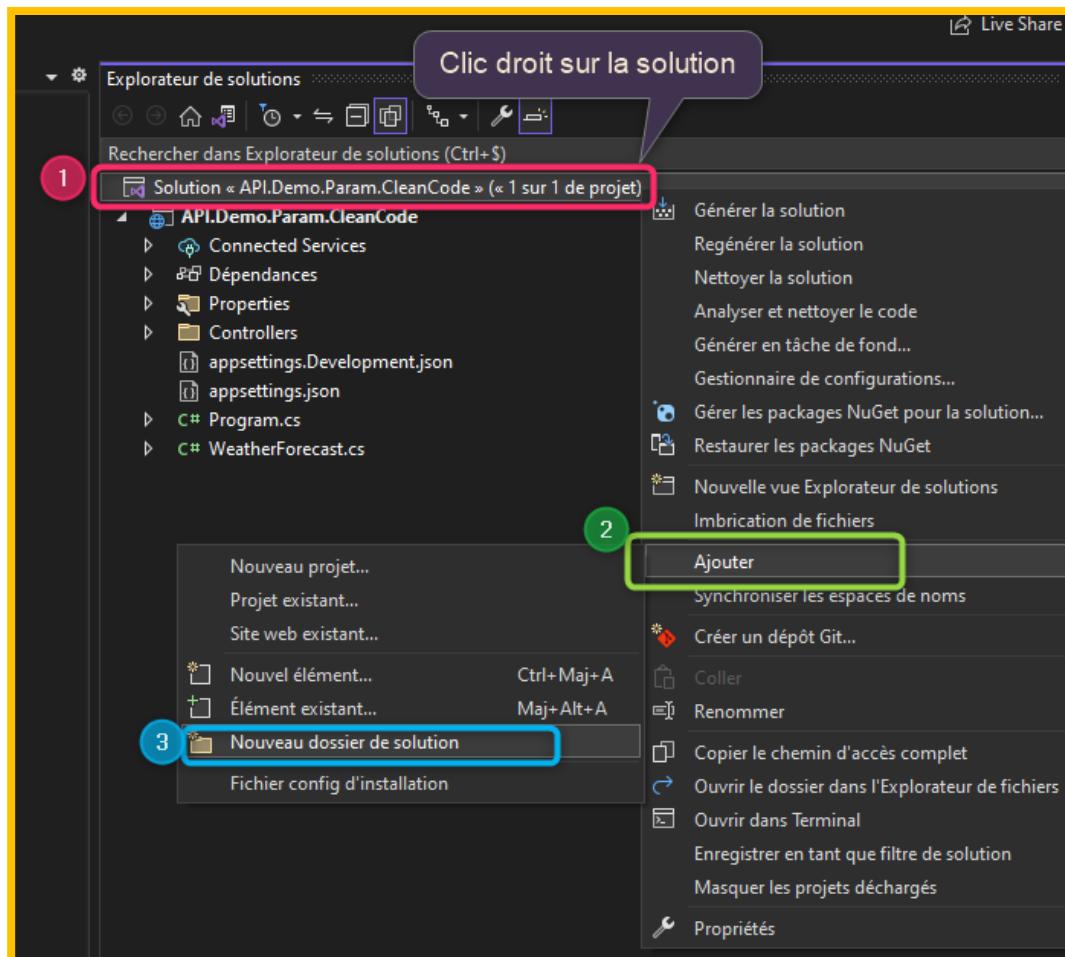


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 8 / 59



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 9 / 59

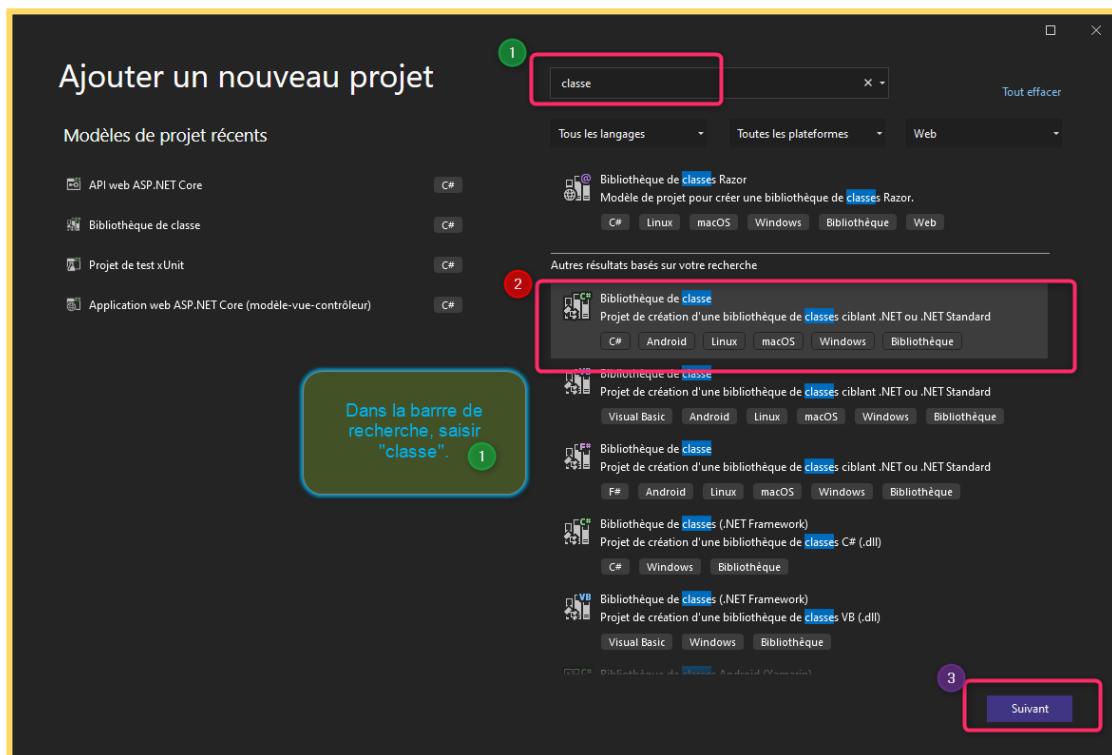
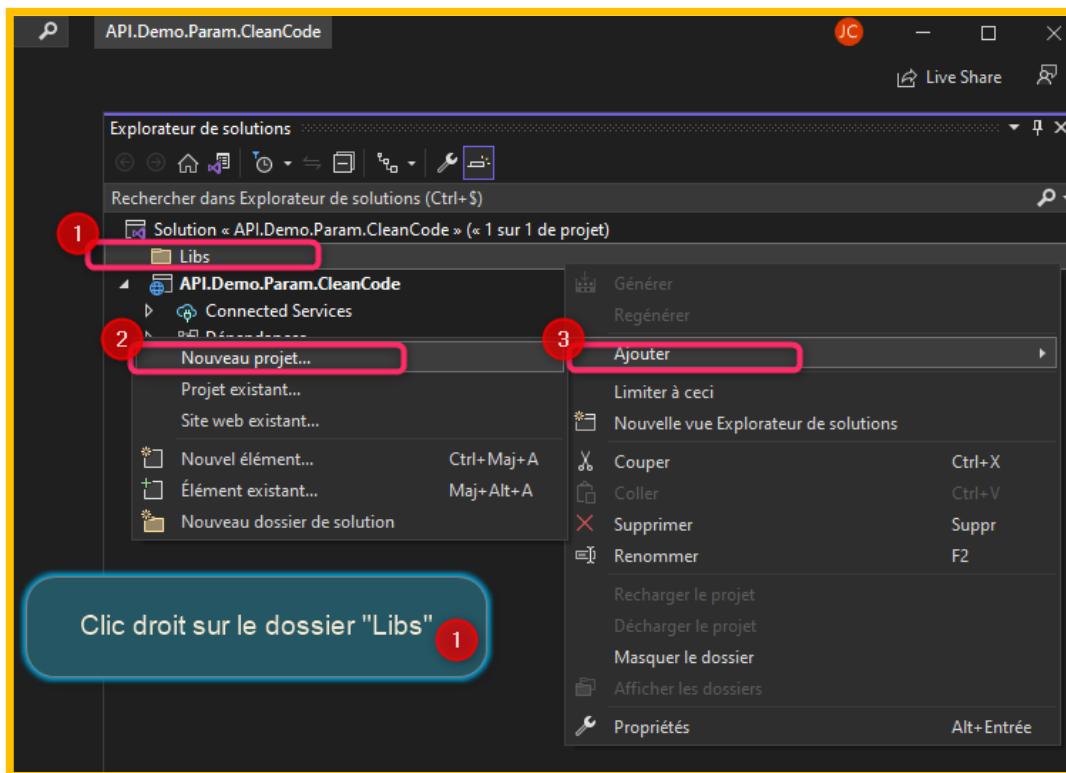
VI – Création d'un dossier « Libs »



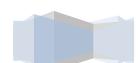
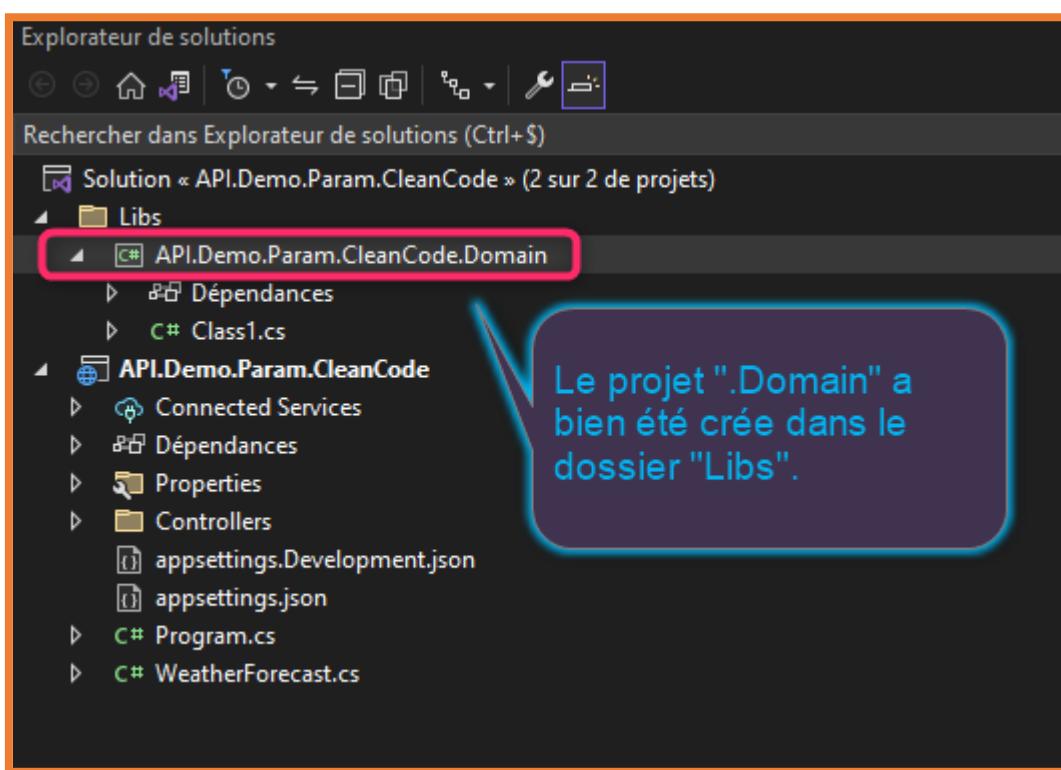
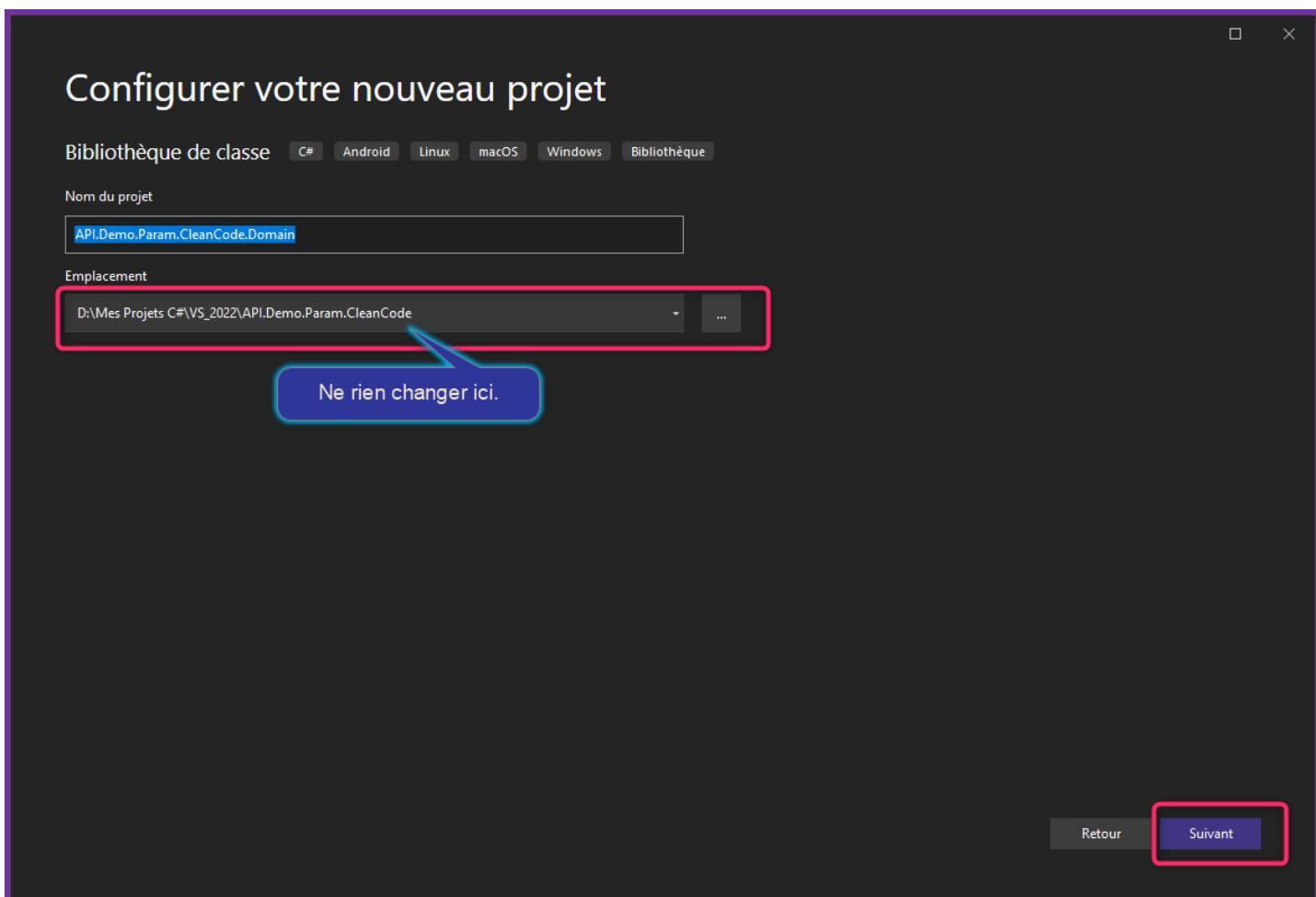
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 10 / 59

VII – Création du projet « API.Demo.Param.CleanCode.Domain »

- Le projet « API.Demo.Param.CleanCode.Domain » va contenir la ou les descriptions de table(s).
- Ici, pour l'exemple, on crée la structure de la table « **Car.Brand** ».



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 11 / 59



JC CERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 12 / 59

VII – A Créer la description de la table « CarBrand »

- Il s'agit de la table des marques de voitures

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `CarModels.cs` file. The code defines a class `CarModels` with properties `ID_CarModel`, `Libelle`, and `Brand`. The `Brand` property is annotated with `[NotMapped]`. The code editor has several annotations in green and red, such as `// > ID générée par code <`, `// > ID Générée par la base <`, and `// > Libellé modèle de la voiture <`. The right side of the interface shows the **Explorateur de solutions** (Solution Explorer) pane, which lists the projects and files in the solution. The file `CarModels.cs` is highlighted with a red box.

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Demo.Param.CleanCode.Domain
{
    public class CarModels
    {
        #region properties
        // > ID générée par code <
        public string ID_CarModel { get; set; }

        // > ID Générée par la base <
        public int ID_auto { get; set; }

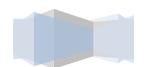
        // > Libellé modèle de la voiture <
        public string Libelle { get; set; }

        // > Clé étrangère : Marque de la voiture <
        public string FK_ID_CarBrand { get; set; }

        // ---- Une modélisation de voiture ne PEUT qu'une marque ---
        // > pour la relation dans "CFG_CarModels" ) <
        // > Remarque => pas insérer en base <
        [NotMapped]
        public CarBrand Brand { get; set; }

        #endregion
    }
}

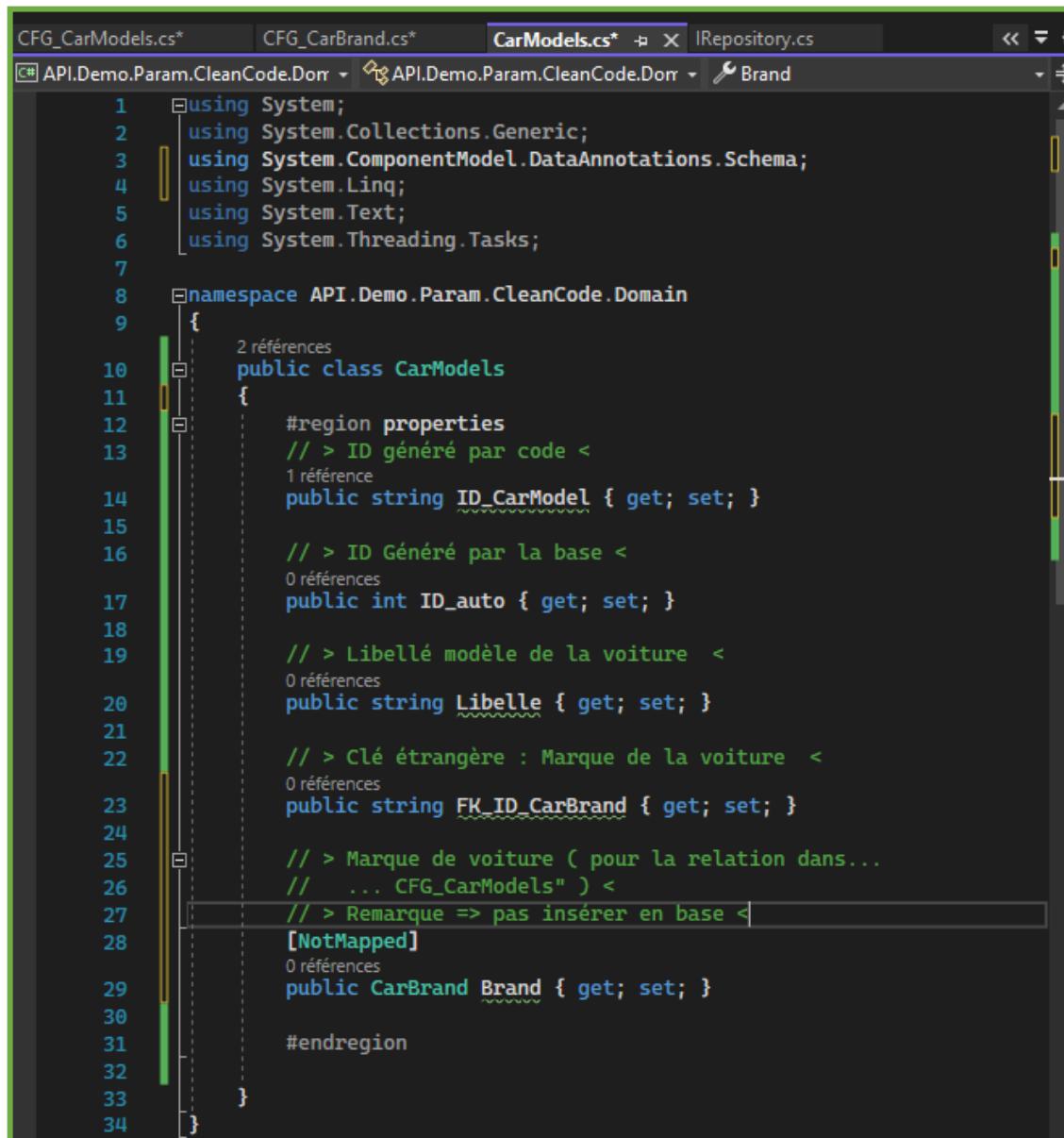
```



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 13 / 59

VII – B Créer la description de la table « CarModels »

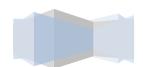
- Il s'agit de la table des modèles de voitures



```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations.Schema;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace API.Demo.Param.CleanCode.Domain
9  {
10    public class CarModels
11    {
12      #region properties
13      // > ID généré par code <
14      public string ID_CarModel { get; set; }
15
16      // > ID Généré par la base <
17      public int ID_auto { get; set; }
18
19      // > Libellé modèle de la voiture <
20      public string Libelle { get; set; }
21
22      // > Clé étrangère : Marque de la voiture <
23      public string FK_ID_CarBrand { get; set; }
24
25      // > Marque de voiture ( pour la relation dans...
26      // ... CFG_CarModels" ) <
27      // > Remarque => pas insérer en base <
28      [NotMapped]
29      public CarBrand Brand { get; set; }
30
31      #endregion
32    }
33  }
34

```



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 14 / 59

VII – C Créer la description de la table « Car »

- Il s'agit de la table des voitures

The screenshot shows the Visual Studio interface with the following details:

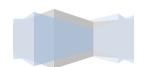
- Solution Explorer:** Shows the project structure under "API.Demo.Param.CleanCode". A file named "Car.cs" is selected and highlighted with a red rectangle.
- Code Editor:** Displays the "Car.cs" file content. The code defines a class "Car" with properties: "ID_Car", "ID_auto", "FK_ID_CarBrand", and "FK_ID_CarModel". The "FK_ID_CarBrand" property is currently selected.
- Toolbars and Status Bar:** Standard Visual Studio toolbars and status bar are visible at the top and bottom of the interface.

```

Car.cs*  X  CFG_CarModels.cs*  CFG_CarBrand.cs*  CarModels.cs
API.Demo.Param.CleanCode.I  API.Demo.Param.CleanCode.I  FK_ID_CarBrand

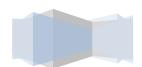
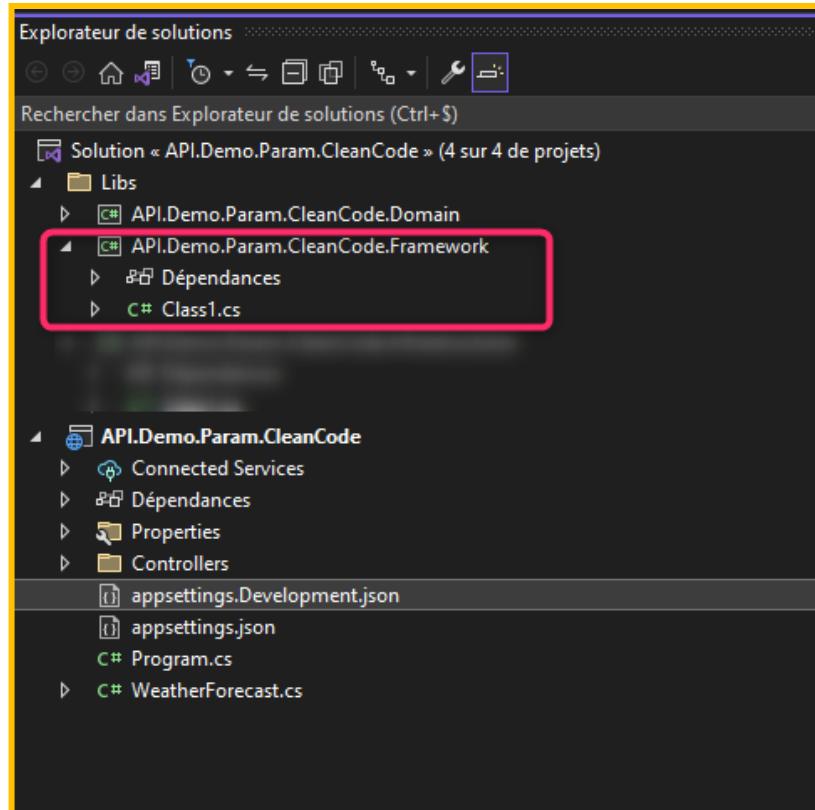
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace API.Demo.Param.CleanCode.Domain
8  {
9      /// <summary>
10     /// Table des voitures
11     /// </summary>
12     public class Car
13     {
14         #region properties
15         // > ID généré par code <
16         public string ID_Car { get; set; }
17
18         // > ID Généré par la base <
19         public int ID_auto { get; set; }
20
21         // > Clé étrangère : Marque de la voiture <
22         public string FK_ID_CarBrand { get; set; }
23
24         // > Clé étrangère : Model de la voiture <
25         public string FK_ID_CarModel { get; set; }
26         #endregion
27
28     }
29
30

```



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 15 / 59

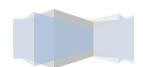
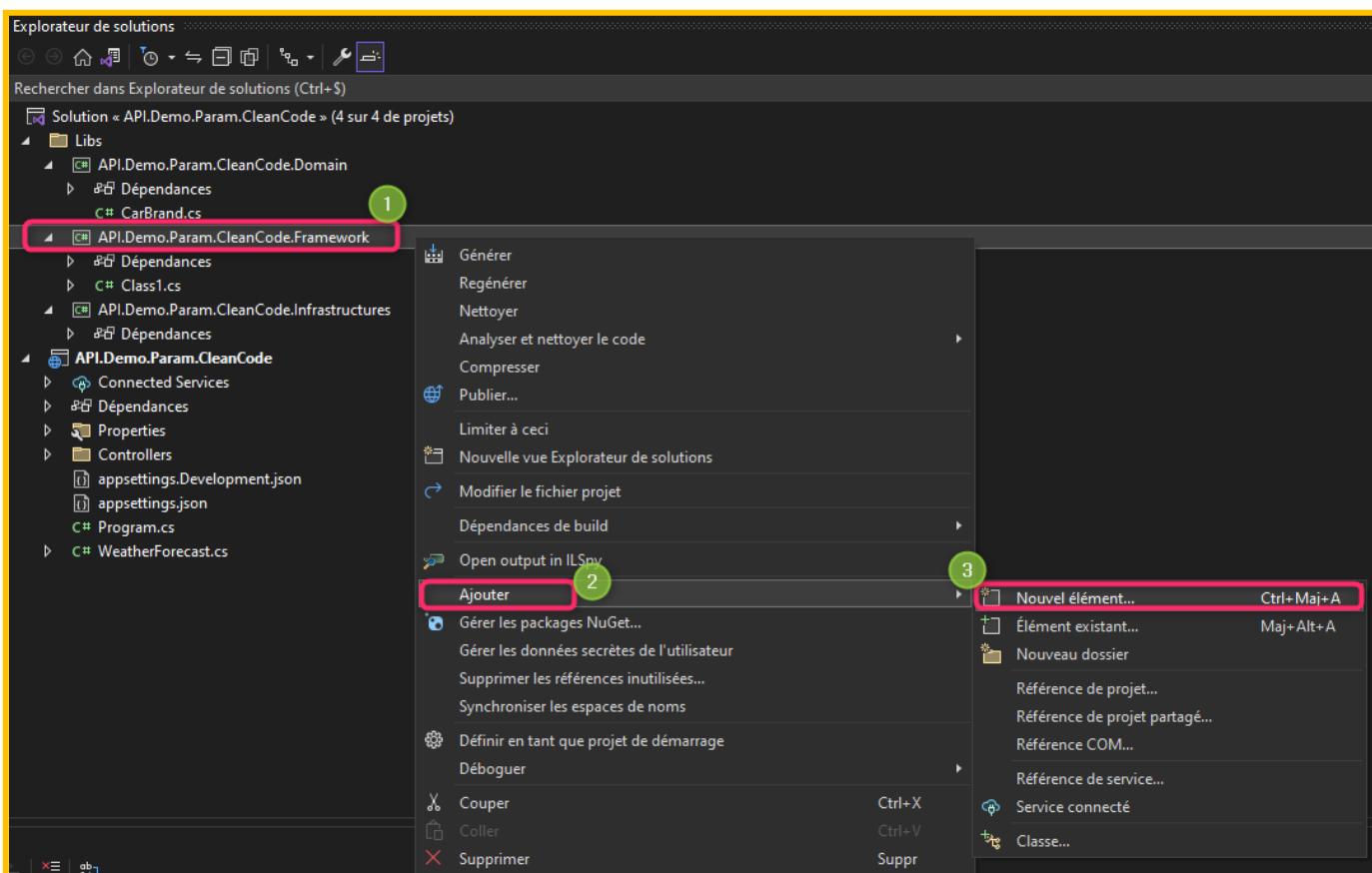
VIII – Création du projet « API.Demo.Param.CleanCode.Framework »



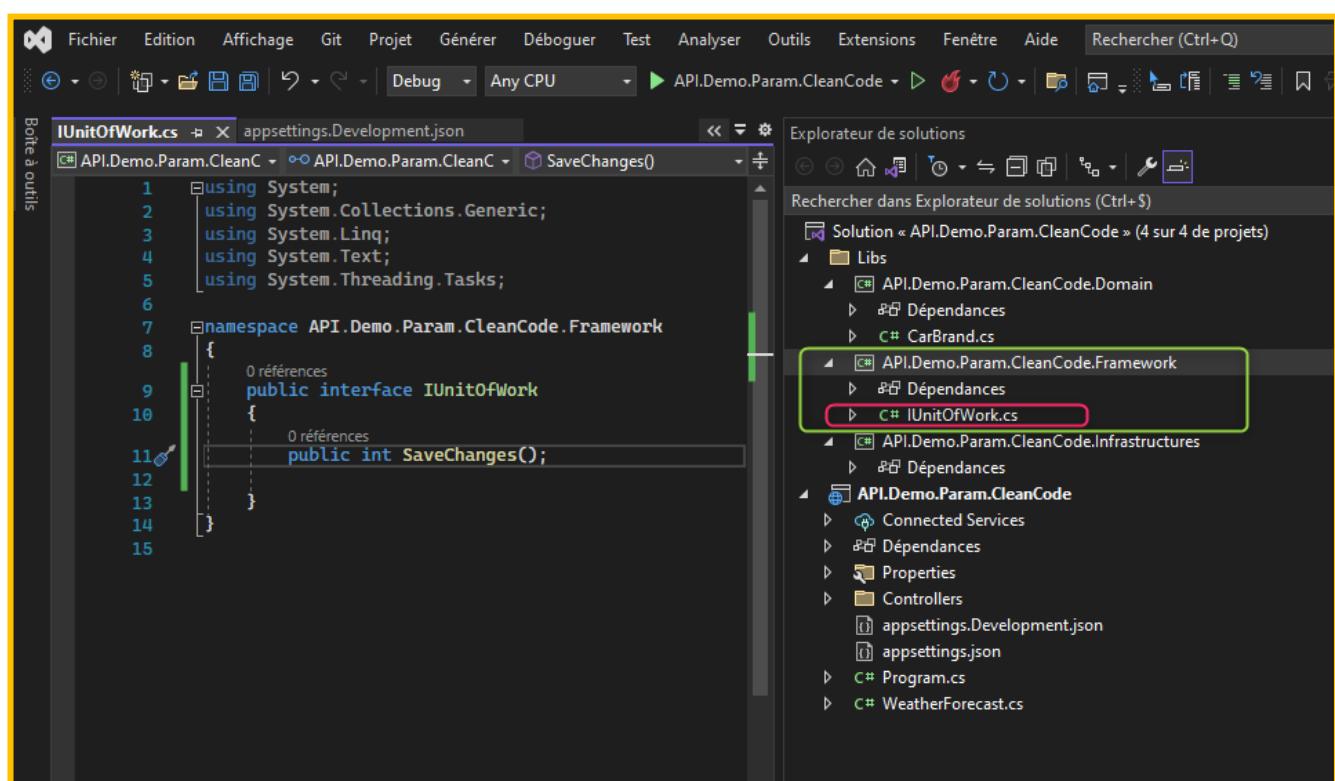
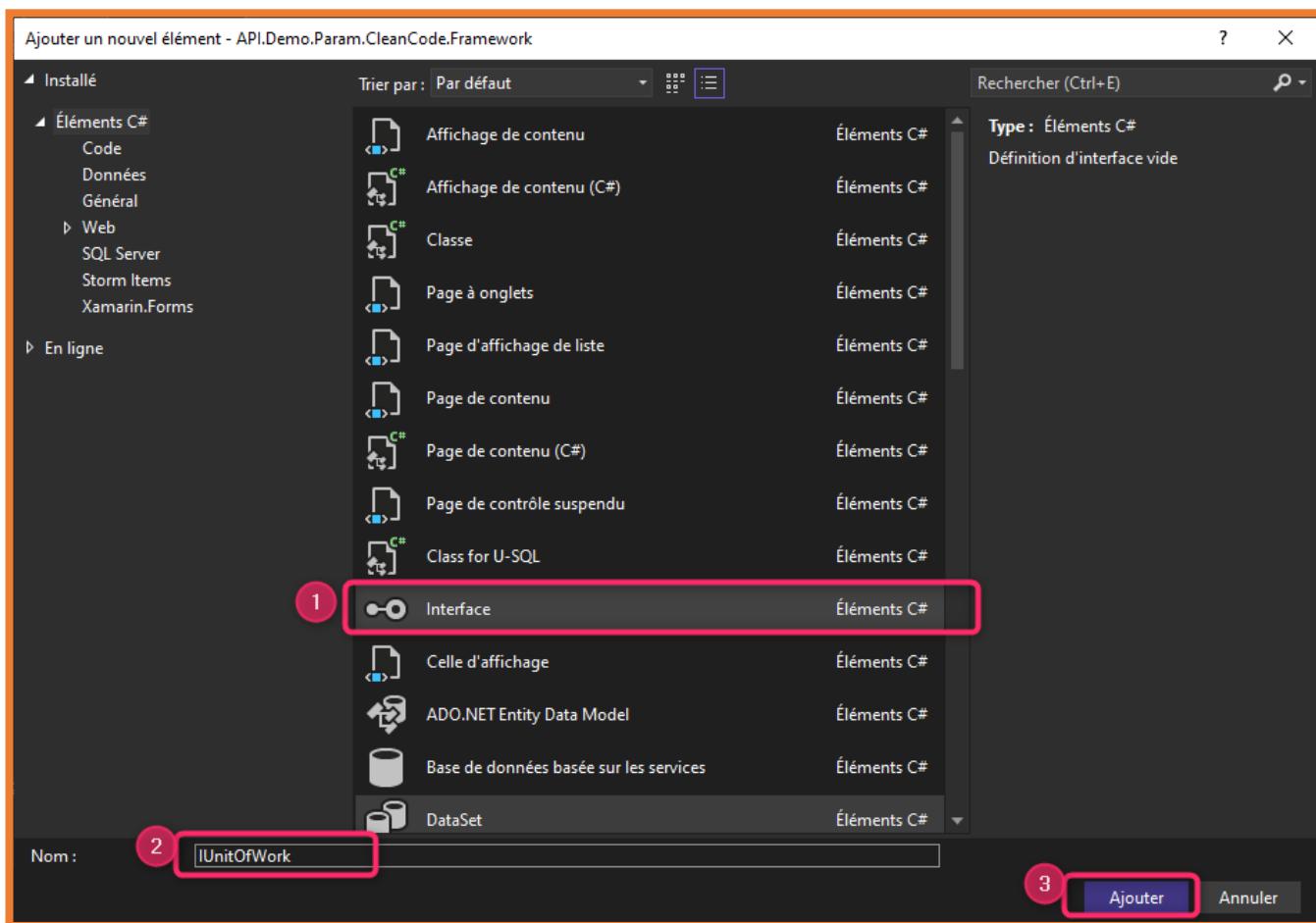
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 16 / 59

VIII – A Ajout de l'interface « IUnitOfWork »

- « **Unit of Work** » est un pattern de développement.
- L'idée, c'est de découpler la partie accès enregistrement de la partie Base de données.
- Grâce à cela, on va pouvoir respecter la **séparation** entre la partie **contrôleur** et la partie **DbContext**.
- L'essentiel étant que de notre côté **repository**, on va ignorer complètement pour l'enregistrement toute la partie du fait qu'elle est liée à notre base de données.
- L'essentiel, c'est juste de savoir qu'on a une unité d'enregistrement qui représente une **transaction**, quelque chose que l'on veut enregistrer de manière bloc : c'est ce qu'on appellera un **agrégat**.
- On va enregistrer cet agrégat, ce bloc de données à stocker en base de données.

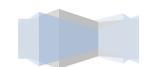
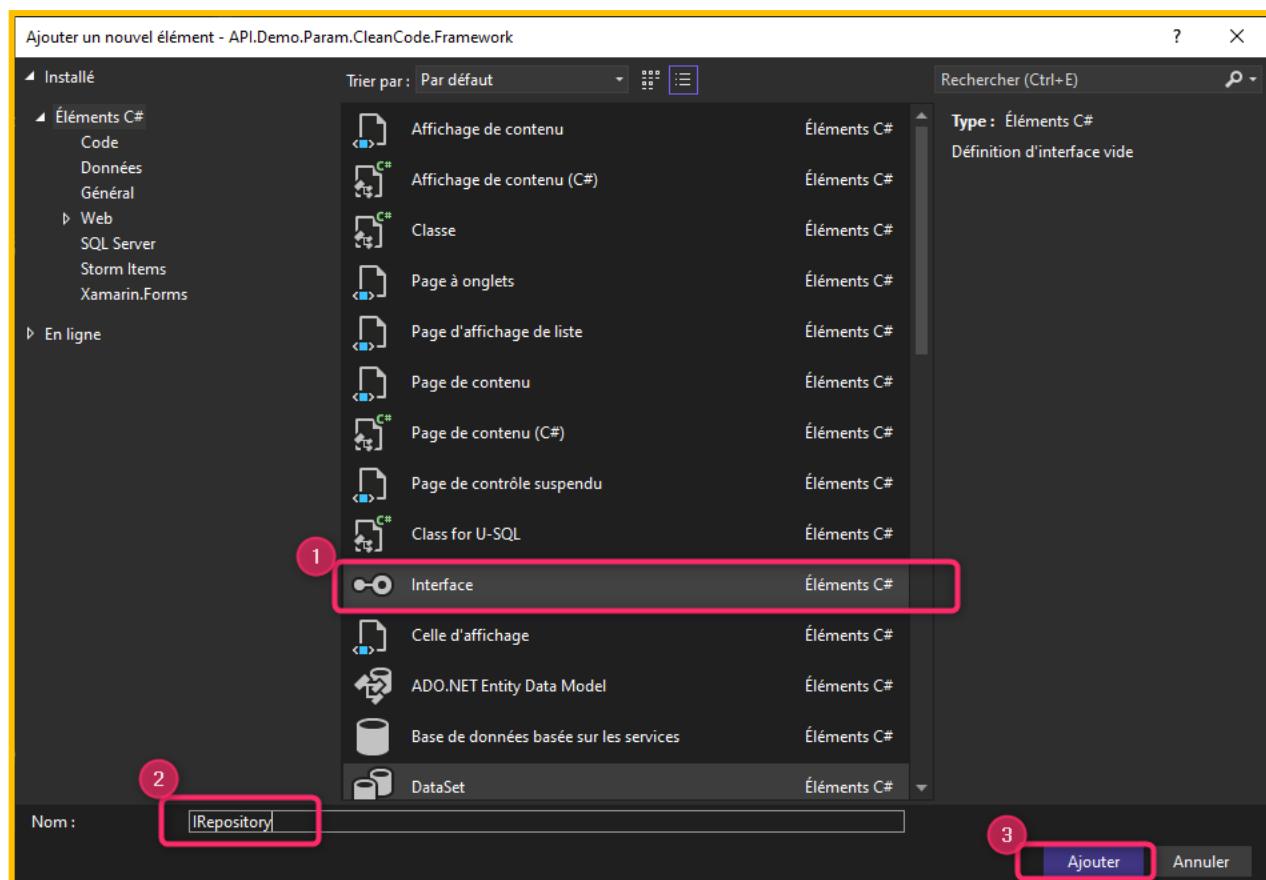
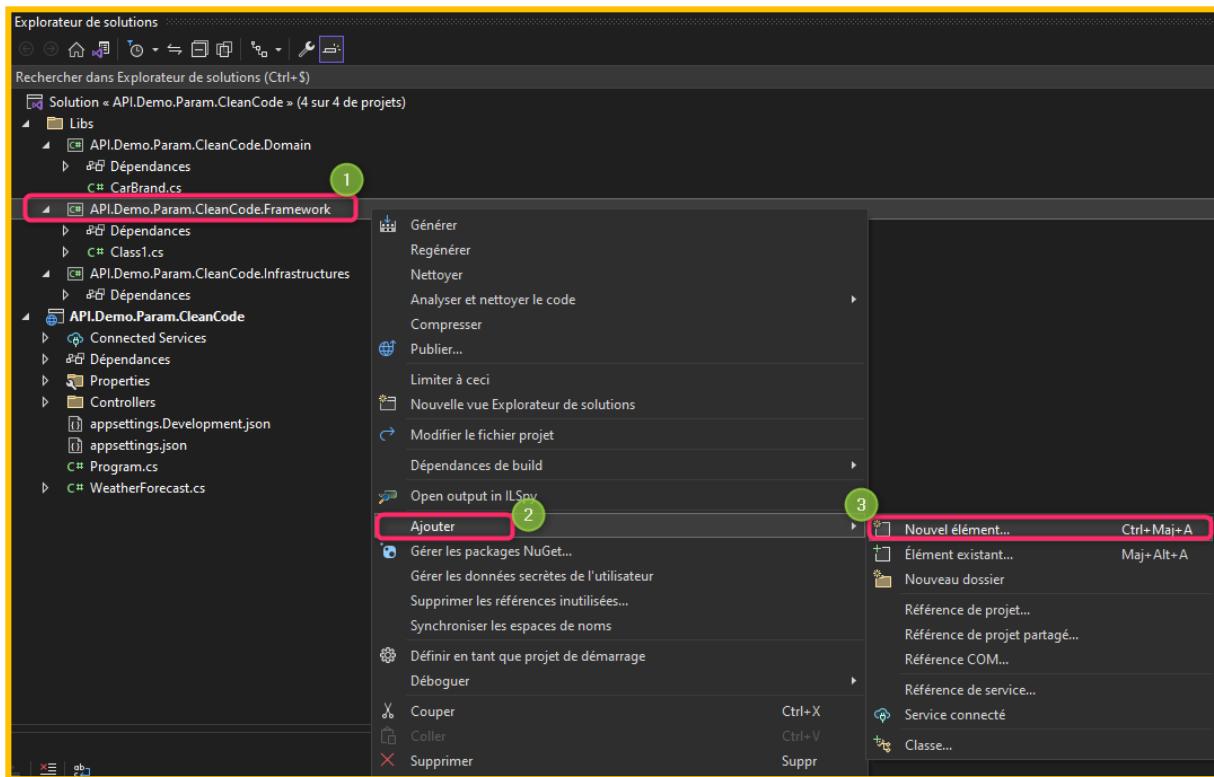


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 17 / 59



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 18 / 59

VIII –B Ajout de l'interface « IRepository »



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 19 / 59

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace API.Demo.Param.CleanCode.Framework
8  {
9      /// <summary>
10     /// Use it to define class as repository
11     /// </summary>
12
13     0 références
14     public interface IRepository
15     {
16         // > !! TOUT CE QUI SERA REPOSITORY IMPLEMENTERA CELA !! <
17         // > On renvoie l'interface "IUnitOfWork" <
18         // ( Pas de "set" car on ne pas l'affecter, ...
19         // ...car ici one fait que renvoyer )
20         0 références
21         IUnitOfWork UnitOfWork { get; }
22     }
23

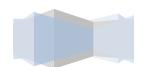
```

Explorateur de solutions

Rechercher dans Explorateur de solutions (Ctrl + \$)

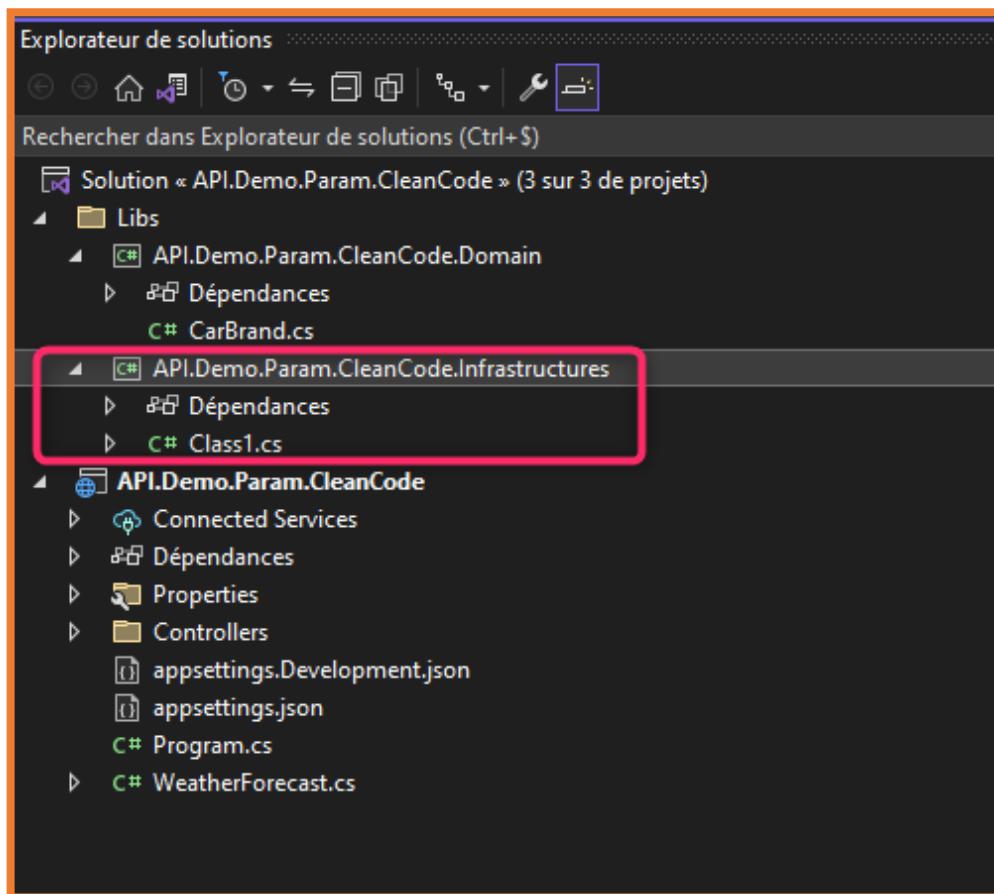
Solution « API.Demo.Param.CleanCode » (4 sur 4 de projets)

- Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - CarBrand.cs
- API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
- API.Demo.Param.CleanCode.Infrastructures
 - Dépendances
- API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Controllers
 - appsettings.Development.json
 - appsettings.json
 - Program.cs
 - WeatherForecast.cs



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 20 / 59

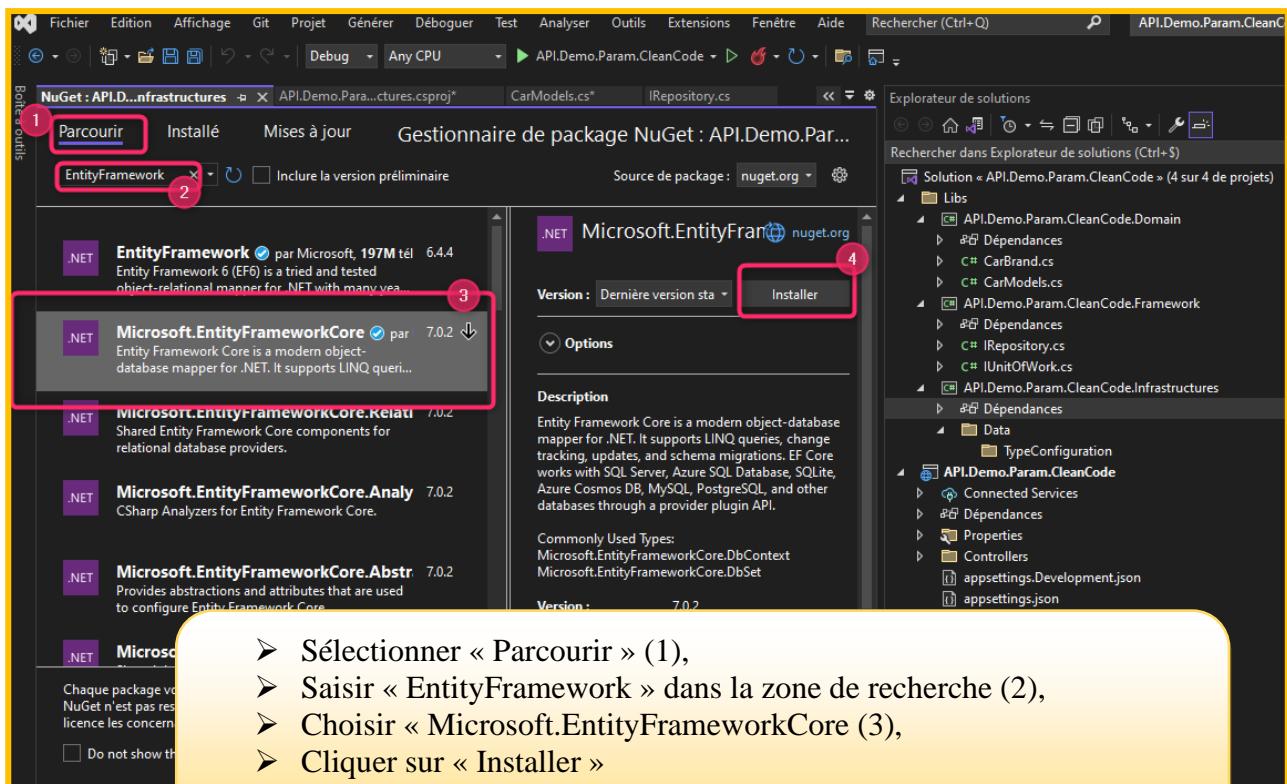
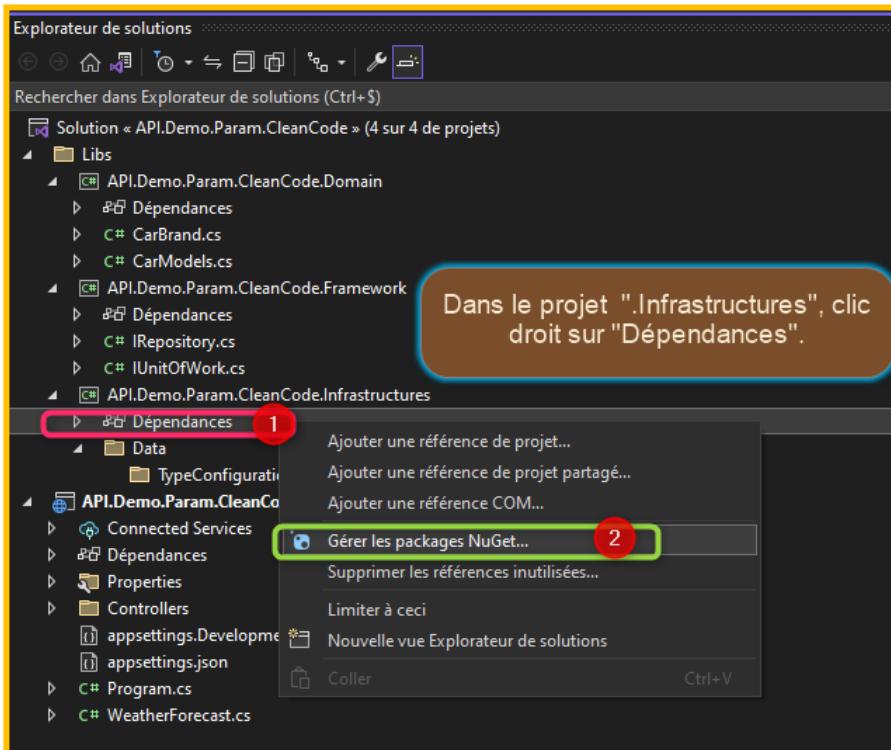
IX – Création du projet « API.Demo.Param.CleanCode.Infrastructure »



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 21 / 59

IX –A Ajout des différents NuGets

IX –A -1 Ajout du NuGet « EntityFrameworkCore »



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 22 / 59

IX –A -2 Ajout du NuGet « EntityFrameworkCore.Design »

- Répéter la même opération pour le NuGet « **EntityFramweworkCore.Design** »

IX –A -3 Ajout du NuGet « EntityFrameworkCore.Relational »

- Répéter la même opération pour le NuGet « **EntityFramweworkCore.Relational** »

IX –A-4 Ajout du NuGet « EntityFrameworkCore.Sqlite»

- Pour la phase de développement, on travaille sur une base **Sqlite**
- Répéter la même opération pour le NuGet « **EntityFramweworkCore.Sqlite**»

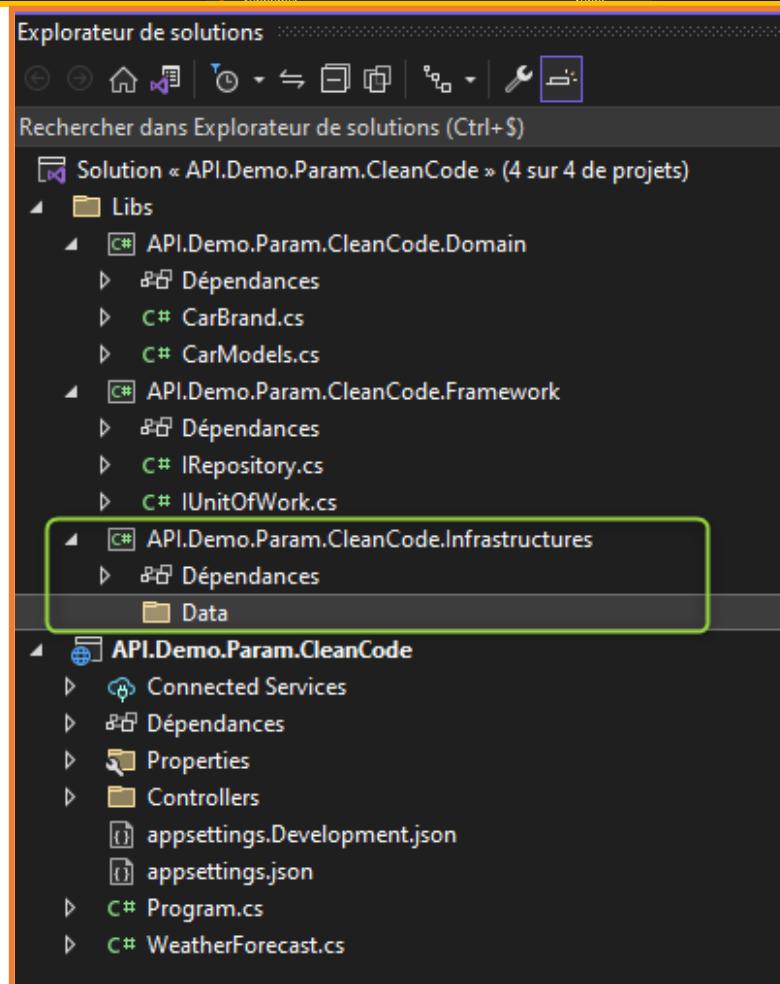
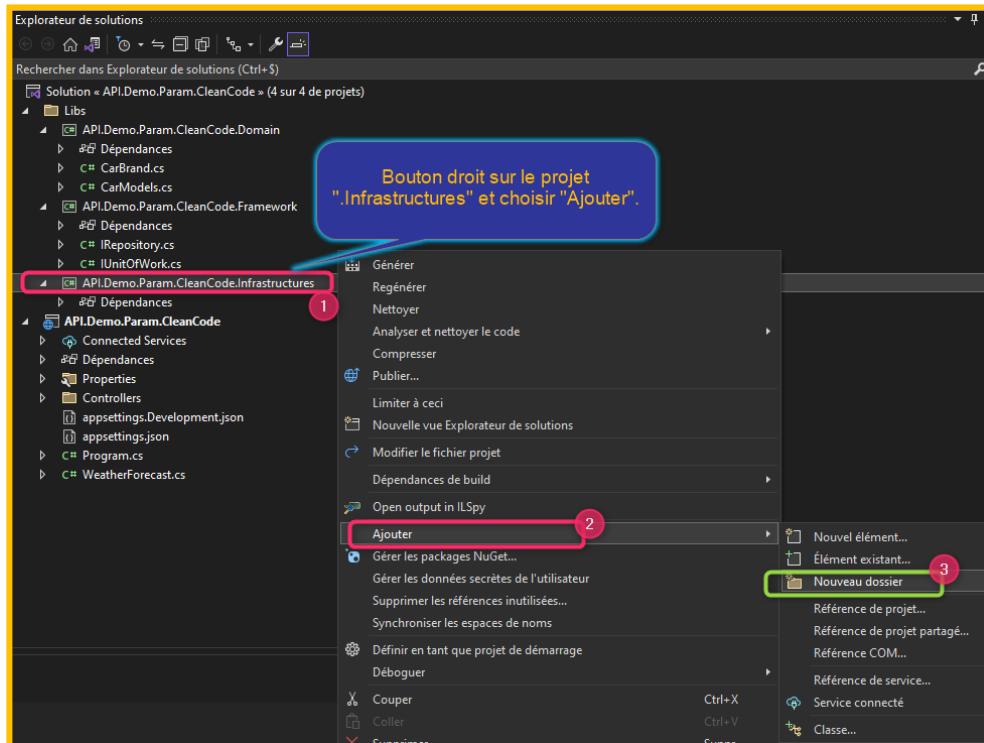
IX –A-5 Ajout du NuGet « EntityFrameworkCore.SqlServer»

- Pour la phase de production (ici c'est un exemple), on travaille sur une base **SqlServer**
- Répéter la même opération pour le NuGet « **EntityFramweworkCore.SqlServer**»



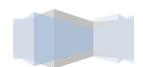
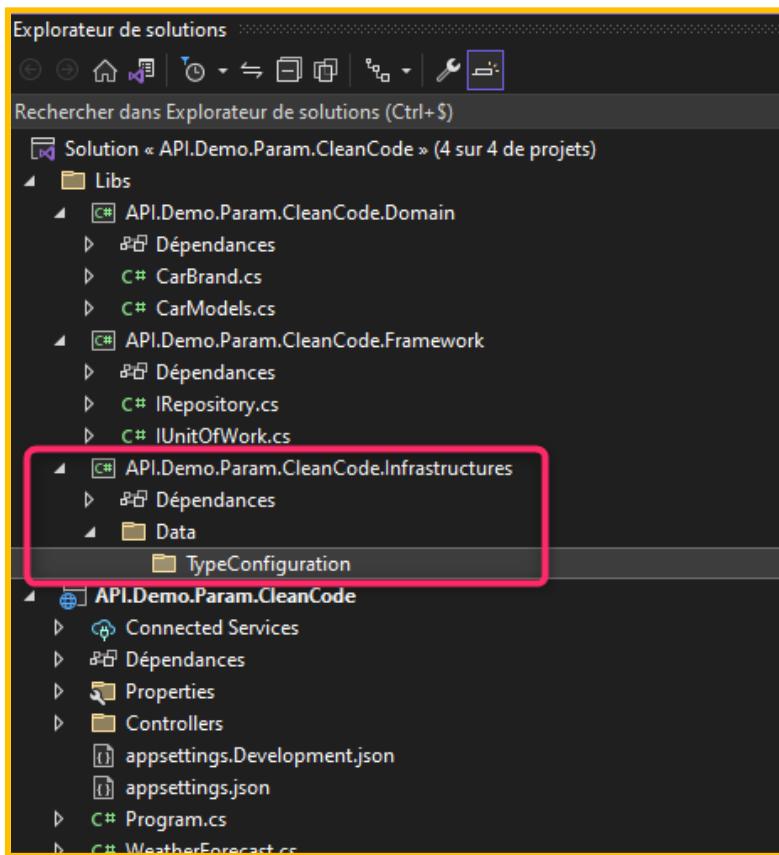
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 23 / 59

IX –B Création du dossier « Data »



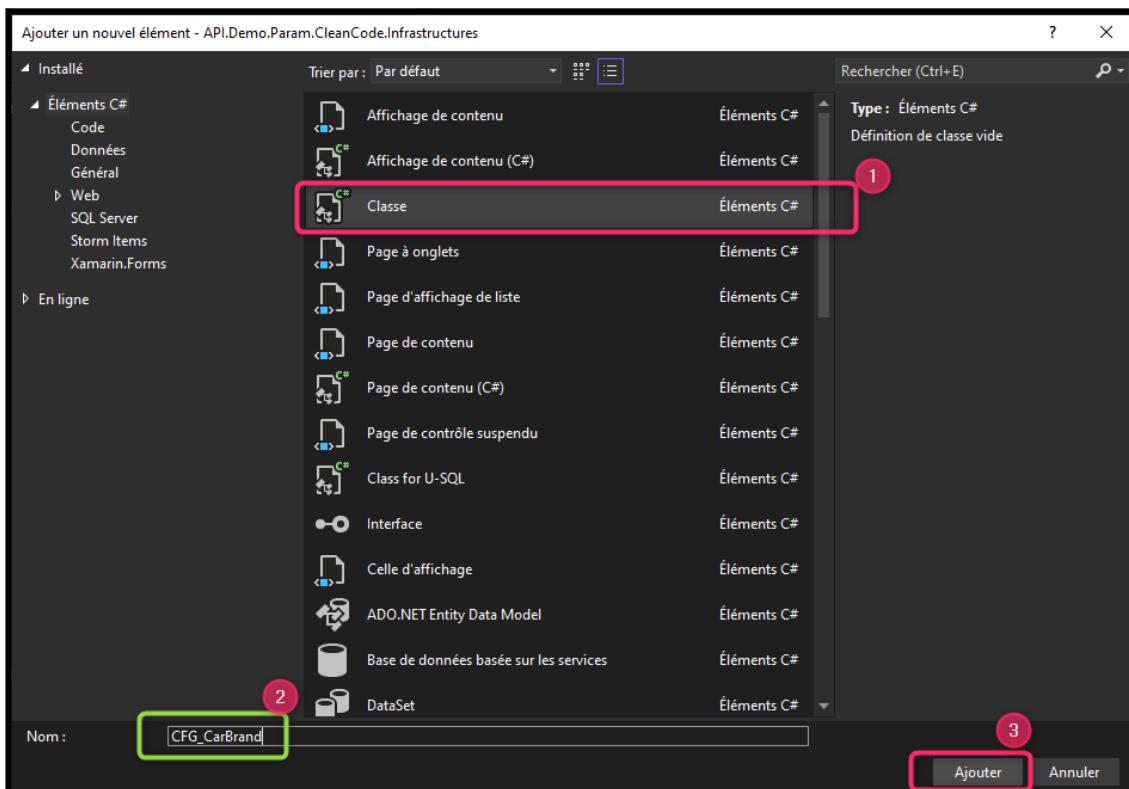
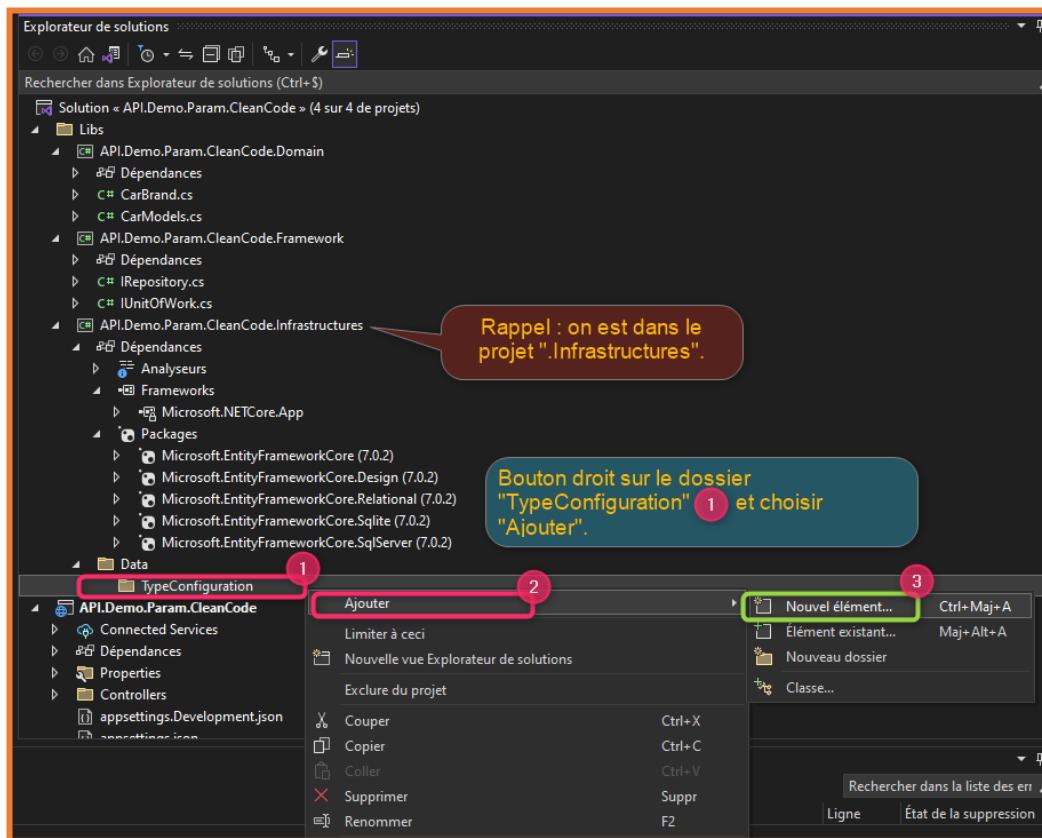
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 24 / 59

IX –C Création du dossier « TypeConfiguration » dans le dossier « Data »



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 25 / 59

IX – D Ajout de la configuration « CFG_CarBrand » pour la table « CarBrand » dans le dossier « TypeConfiguration »



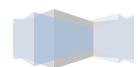
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 26 / 59

Remarque :

- Ces informations concernant les tables seront utilisées par le l'**ORM (Object Relational Mapping)**
 - Elles seront utilisées aussi si l'on crée un projet de Migrations en utilisant « dotnet ef core » (voir document de synthèse « 99_A_TEC_C#_CORE_MIGRATION_SYNTHÈSE »

The screenshot shows the Visual Studio IDE interface. The code editor on the left displays the file `CFG_CarBrand.cs` with C# code for configuring a database table. The Solution Explorer on the right shows the project structure for `API.Demo.Param.CleanCode`, including files like `Car.cs`, `CarBrand.cs`, `CarModels.cs`, and `EntityTypeBuilder.cs`. A pink box highlights the `CFG_CarBrand.cs` file in the Solution Explorer.

```
Car.cs* CFG_CarModels.cs* CFG_CarBrand.cs* CarBrand.cs CarModels.cs
API.Demo.Param.CleanCode.Infrastructure API.Demo.Param.CleanCode.Infrastructure Configure(EntityTypeBuilder<CarBrand>
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  // > pour "CarBrand" <
8  using API.Demo.Param.CleanCode.Domain;
9
10 // > pour "IEntityTypeConfiguration" <
11 using Microsoft.EntityFrameworkCore;
12
13 // > Pour "EntityTypeBuilder"
14 using Microsoft.EntityFrameworkCore.Metadata.Builders;
15
16 namespace API.Demo.Param.CleanCode.Infrastructure.Data.TypeConfiguration
17 {
18     // Remarque : Ces informations seront utilisées si l'on ...
19     //.... met en place un projet "Migrations" avec
20     //.... dotnet ef core ( cli => Comand Line Interface ).  
0 références
21     public class CFG_CarBrand : IEntityTypeConfiguration<CarBrand>
22     {
23         0 références
24         public void Configure(EntityTypeBuilder<CarBrand> builder)
25         {
26             // > Ici, on indique ici que le nom de la table sera bien
27             // ... "CarBrand"
28             builder.ToTable("CarBrand");
29
30             // > On définit une clé à la table "CarBrand" :
31             // => La clé sera l'ID du modèle calculée par le code
32             builder.HasKey(item => item.ID_CarBrand);
33
34             // > On définit une valeur par défaut pour le libellé <
35             builder.Property(item => item.Libelle)
36                 .HasDefaultValueSql("## Marque voiture (à renseigner) ##");
37         }
38     }
39 }
```

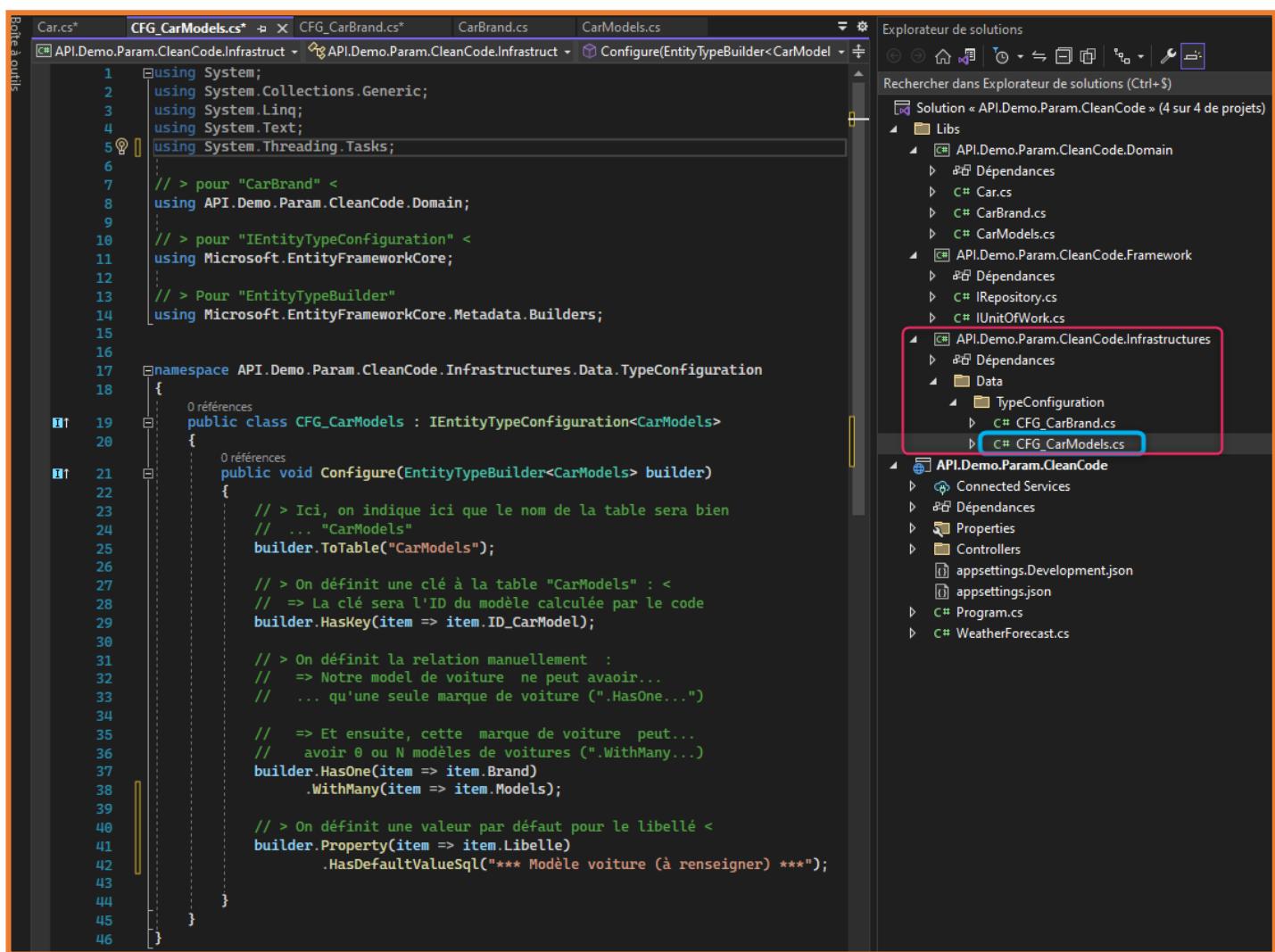


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 27 / 59

IX-E Ajout de la configuration « CFG_CarModels » pour la table « CarModels » dans le dossier « TypeConfiguration »

Remarque :

- Ces informations concernant les tables seront utilisées par le l'**ORM (Object Relational Mapping)**
- Elles seront utilisées aussi si l'on crée un projet de Migrations en utilisant « dotnet ef core » (voir document de synthèse « 99_A_TEC_C#_CORE_MIGRATION_SYNTHÈSE »)



```

Car.cs* CFG_CarModels.cs* X CFG_CarBrand.cs* CarBrand.cs CarModels.cs
API.Demo.Param.CleanCode.Infrastructure API.Demo.Param.CleanCode.Domain Configure(EntityTypeBuilder<CarModel>
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  // > pour "CarBrand" <
8  using API.Demo.Param.CleanCode.Domain;
9
10 // > pour "IEntityTypeConfiguration" <
11 using Microsoft.EntityFrameworkCore;
12
13 // > Pour "EntityTypeBuilder"
14 using Microsoft.EntityFrameworkCore.Metadata.Builders;
15
16
17 namespace API.Demo.Param.CleanCode.Infrastructure.Data.TypeConfiguration
18 {
19     public class CFG_CarModels : IEntityTypeConfiguration<CarModels>
20     {
21         public void Configure(EntityTypeBuilder<CarModels> builder)
22         {
23             // > Ici, on indique ici que le nom de la table sera bien
24             // ... "CarModels"
25             builder.ToTable("CarModels");
26
27             // > On définit une clé à la table "CarModels" : <
28             // => La clé sera l'ID du modèle calculée par le code
29             builder.HasKey(item => item.ID_CarModel);
30
31             // > On définit la relation manuellement :
32             // => Notre model de voiture ne peut avoir...
33             // ... qu'une seule marque de voiture ("HasOne...")
34
35             // => Et ensuite, cette marque de voiture peut...
36             // avoir 0 ou N modèles de voitures ("WithMany...")
37             builder.HasOne(item => item.Brand)
38                 .WithMany(item => item.Models);
39
40             // > On définit une valeur par défaut pour le libellé <
41             builder.Property(item => item.Libelle)
42                 .HasDefaultValueSql("*** Modèle voiture (à renseigner) ***");
43
44         }
45     }
46 }

```

The screenshot shows the Visual Studio IDE interface. The left pane displays the code editor with 'CFG_CarModels.cs' open, containing configuration code for EntityFrameworkCore. The right pane shows the 'Explorateur de solutions' (Solution Explorer) with the project structure. A red box highlights the 'TypeConfiguration' folder under the 'Data' folder, which contains 'CFG_CarBrand.cs' and 'CFG_CarModels.cs'.



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 28 / 59

IX – F Ajout de la configuration « CFG_Car » pour la table « Car » dans le dossier « TypeConfiguration »

Remarque :

- Ces informations concernant les tables seront utilisées par le l'**ORM (Object Relational Mapping)**
- Elles seront utilisées aussi si l'on crée un projet de Migrations en utilisant « dotnet ef core » (voir document de synthèse « 99_A_TEC_C#_CORE_MIGRATION_SYNTHÈSE »)

The screenshot shows the Visual Studio interface. The code editor on the left displays the file `CFG_Car.cs` with the following content:

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  // > pour "CarBrand" <
8  using API.Demo.Param.CleanCode.Domain;
9
10 // > pour "IEntityTypeConfiguration" <
11 using Microsoft.EntityFrameworkCore;
12
13 // Pour "EntityTypeBuilder"
14 using Microsoft.EntityFrameworkCore.Metadata.Builders;
15
16 namespace API.Demo.Param.CleanCode.Infrastructures.Data.TypeConfiguration
17 {
18     /// <summary>
19     /// Configuration Table "Car"
20     /// </summary>
21     public class CFG_Car : IEntityTypeConfiguration<Car>
22     {
23         public void Configure(EntityTypeBuilder<Car> builder)
24         {
25             // > Ici, on indique ici que le nom de la table sera bien
26             // ... "Car"
27             builder.ToTable("Car");
28
29             // > On définit une clé à la table "Car" : <
30             // => La clé sera l'ID du modèle calculée par le code
31             builder.HasKey(item => item.ID_Car);
32
33         }
34     }
35 }
36
37

```

The solution explorer on the right shows the project structure:

- Solution « API.Demo.Param.CleanCode » (4 sur 4 de projets)
 - Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - C# Car.cs
 - C# CarBrand.cs
 - C# CarModels.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - C# IRepository.cs
 - C# IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructures
 - Dépendances
 - Data
 - TypeConfiguration
 - C# CFG_Car.cs (highlighted with a red box)
 - C# CFG_CarBrand.cs
 - C# CFG_CarModels.cs
 - API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Controllers
 - appsettings.Development.json
 - appsettings.json
 - Program.cs
 - WeatherForecast.cs



JC CERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 29 / 59

IX-G Mise en place du « context » dans la classe « DataContext »

Classe DataContext (Partie 1 du code)

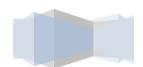
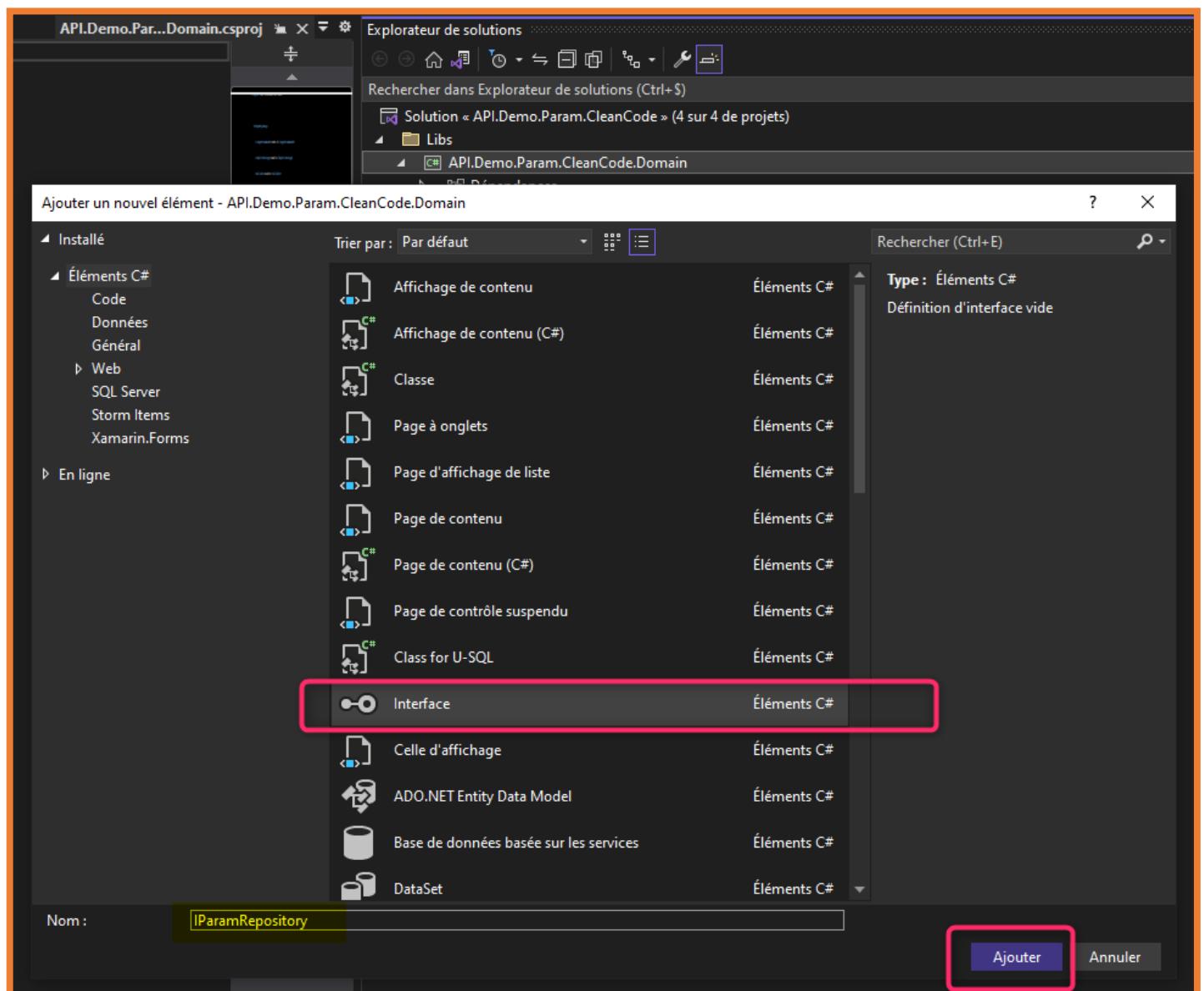
```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  // > Pour "IUnitOfWork" <
8  using API.Demo.Param.CleanCode.Framework;
9
10 // > Pour "DbContext" <
11 using Microsoft.EntityFrameworkCore;
12
13 // > Pour "NotNullAttribute" <
14 using System.Diagnostics.CodeAnalysis;
15
16 // > Pour les configurations de table ("CFG_xxxx") <
17 using API.Demo.Param.CleanCode.Infrastructures.Data.TypeConfiguration;
18
19 // > Pour les définitions de tables <
20 using API.Demo.Param.CleanCode.Domain;
21
22 namespace API.Demo.Param.CleanCode.Infrastructures
23 {
24     2 références
25     public class DataContext : DbContext, IUnitOfWork
26     {
27         // > Constructeur <
28         //   ( Hérite de "DbContext" => F12 sur ":base()" )
29         //   ( Execute le constructeur de "DbContext" )
30         0 références
31         public DataContext() : base()
32         {
33         }
34
35         // > Constructeur <
36         0 références
37         public DataContext([NotNullAttribute] DbContextOptions options) : base(options)
38         {
39         }
40
41         // #####=-----=
42         // ##### -- Installe les configurations pour les tables
43         // #####=-----=
44         0 références
45         protected override void OnModelCreating(ModelBuilder modelBuilder)
46         {
47             base.OnModelCreating(modelBuilder);
48
49             // > On déclare la classe de configuration pour la table CarBrand (Marque de voiture) <
50             modelBuilder.ApplyConfiguration(new CFG_CarBrand());
51
52             // > On déclare la classe de configuration pour la table CarModels (Models de voiture) <
53             modelBuilder.ApplyConfiguration(new CFG_CarModels());
54
55             // > On déclare la classe de configuration pour la table Car (voiture) <
56             modelBuilder.ApplyConfiguration(new CFG_Car());
57
58             // > Table CarBrand (Marque de voiture) <
59             0 références
60             public DbSet<CarBrand> Brand { get; set; }
61
62             // > Table CarModels (Models de voiture) <
63             0 références
64             public DbSet<CarModels> Models { get; set; }
65
66             // > Table Car (voiture) <
67             0 références
68             public DbSet<Car> Car { get; set; }
69         }
70
71     }
72
73     2 références
74     public void Dispose()
75     {
76         Dispose(true);
77     }
78
79     protected void Dispose(bool disposing)
80     {
81         if (disposing)
82         {
83             ((IDisposable)Brand).Dispose();
84             ((IDisposable)Models).Dispose();
85             ((IDisposable)Car).Dispose();
86         }
87     }
88
89     // > Propriétés <
90     public IUnitOfWork UnitOfWork { get; private set; }
91
92     // > Méthodes <
93     public void Commit()
94     {
95         UnitOfWork.Commit();
96     }
97
98     public void Rollback()
99     {
100        UnitOfWork.Rollback();
101    }
102
103    // > Propriétés <
104    public void Add(T entity)
105    {
106        UnitOfWork.Add(entity);
107    }
108
109    public void Remove(T entity)
110    {
111        UnitOfWork.Remove(entity);
112    }
113
114    public void Update(T entity)
115    {
116        UnitOfWork.Update(entity);
117    }
118
119    public void Save()
120    {
121        UnitOfWork.Save();
122    }
123
124    public void Commit()
125    {
126        UnitOfWork.Commit();
127    }
128
129    public void Rollback()
130    {
131        UnitOfWork.Rollback();
132    }
133
134    public void Add(T entity)
135    {
136        UnitOfWork.Add(entity);
137    }
138
139    public void Remove(T entity)
140    {
141        UnitOfWork.Remove(entity);
142    }
143
144    public void Update(T entity)
145    {
146        UnitOfWork.Update(entity);
147    }
148
149    public void Save()
150    {
151        UnitOfWork.Save();
152    }
153
154    public void Commit()
155    {
156        UnitOfWork.Commit();
157    }
158
159    public void Rollback()
160    {
161        UnitOfWork.Rollback();
162    }
163
164    public void Add(T entity)
165    {
166        UnitOfWork.Add(entity);
167    }
168
169    public void Remove(T entity)
170    {
171        UnitOfWork.Remove(entity);
172    }
173
174    public void Update(T entity)
175    {
176        UnitOfWork.Update(entity);
177    }
178
179    public void Save()
180    {
181        UnitOfWork.Save();
182    }
183
184    public void Commit()
185    {
186        UnitOfWork.Commit();
187    }
188
189    public void Rollback()
190    {
191        UnitOfWork.Rollback();
192    }
193
194    public void Add(T entity)
195    {
196        UnitOfWork.Add(entity);
197    }
198
199    public void Remove(T entity)
200    {
201        UnitOfWork.Remove(entity);
202    }
203
204    public void Update(T entity)
205    {
206        UnitOfWork.Update(entity);
207    }
208
209    public void Save()
210    {
211        UnitOfWork.Save();
212    }
213
214    public void Commit()
215    {
216        UnitOfWork.Commit();
217    }
218
219    public void Rollback()
220    {
221        UnitOfWork.Rollback();
222    }
223
224    public void Add(T entity)
225    {
226        UnitOfWork.Add(entity);
227    }
228
229    public void Remove(T entity)
230    {
231        UnitOfWork.Remove(entity);
232    }
233
234    public void Update(T entity)
235    {
236        UnitOfWork.Update(entity);
237    }
238
239    public void Save()
240    {
241        UnitOfWork.Save();
242    }
243
244    public void Commit()
245    {
246        UnitOfWork.Commit();
247    }
248
249    public void Rollback()
250    {
251        UnitOfWork.Rollback();
252    }
253
254    public void Add(T entity)
255    {
256        UnitOfWork.Add(entity);
257    }
258
259    public void Remove(T entity)
260    {
261        UnitOfWork.Remove(entity);
262    }
263
264    public void Update(T entity)
265    {
266        UnitOfWork.Update(entity);
267    }
268
269    public void Save()
270    {
271        UnitOfWork.Save();
272    }
273
274    public void Commit()
275    {
276        UnitOfWork.Commit();
277    }
278
279    public void Rollback()
280    {
281        UnitOfWork.Rollback();
282    }
283
284    public void Add(T entity)
285    {
286        UnitOfWork.Add(entity);
287    }
288
289    public void Remove(T entity)
290    {
291        UnitOfWork.Remove(entity);
292    }
293
294    public void Update(T entity)
295    {
296        UnitOfWork.Update(entity);
297    }
298
299    public void Save()
300    {
301        UnitOfWork.Save();
302    }
303
304    public void Commit()
305    {
306        UnitOfWork.Commit();
307    }
308
309    public void Rollback()
310    {
311        UnitOfWork.Rollback();
312    }
313
314    public void Add(T entity)
315    {
316        UnitOfWork.Add(entity);
317    }
318
319    public void Remove(T entity)
320    {
321        UnitOfWork.Remove(entity);
322    }
323
324    public void Update(T entity)
325    {
326        UnitOfWork.Update(entity);
327    }
328
329    public void Save()
330    {
331        UnitOfWork.Save();
332    }
333
334    public void Commit()
335    {
336        UnitOfWork.Commit();
337    }
338
339    public void Rollback()
340    {
341        UnitOfWork.Rollback();
342    }
343
344    public void Add(T entity)
345    {
346        UnitOfWork.Add(entity);
347    }
348
349    public void Remove(T entity)
350    {
351        UnitOfWork.Remove(entity);
352    }
353
354    public void Update(T entity)
355    {
356        UnitOfWork.Update(entity);
357    }
358
359    public void Save()
360    {
361        UnitOfWork.Save();
362    }
363
364    public void Commit()
365    {
366        UnitOfWork.Commit();
367    }
368
369    public void Rollback()
370    {
371        UnitOfWork.Rollback();
372    }
373
374    public void Add(T entity)
375    {
376        UnitOfWork.Add(entity);
377    }
378
379    public void Remove(T entity)
380    {
381        UnitOfWork.Remove(entity);
382    }
383
384    public void Update(T entity)
385    {
386        UnitOfWork.Update(entity);
387    }
388
389    public void Save()
390    {
391        UnitOfWork.Save();
392    }
393
394    public void Commit()
395    {
396        UnitOfWork.Commit();
397    }
398
399    public void Rollback()
400    {
401        UnitOfWork.Rollback();
402    }
403
404    public void Add(T entity)
405    {
406        UnitOfWork.Add(entity);
407    }
408
409    public void Remove(T entity)
410    {
411        UnitOfWork.Remove(entity);
412    }
413
414    public void Update(T entity)
415    {
416        UnitOfWork.Update(entity);
417    }
418
419    public void Save()
420    {
421        UnitOfWork.Save();
422    }
423
424    public void Commit()
425    {
426        UnitOfWork.Commit();
427    }
428
429    public void Rollback()
430    {
431        UnitOfWork.Rollback();
432    }
433
434    public void Add(T entity)
435    {
436        UnitOfWork.Add(entity);
437    }
438
439    public void Remove(T entity)
440    {
441        UnitOfWork.Remove(entity);
442    }
443
444    public void Update(T entity)
445    {
446        UnitOfWork.Update(entity);
447    }
448
449    public void Save()
450    {
451        UnitOfWork.Save();
452    }
453
454    public void Commit()
455    {
456        UnitOfWork.Commit();
457    }
458
459    public void Rollback()
460    {
461        UnitOfWork.Rollback();
462    }
463
464    public void Add(T entity)
465    {
466        UnitOfWork.Add(entity);
467    }
468
469    public void Remove(T entity)
470    {
471        UnitOfWork.Remove(entity);
472    }
473
474    public void Update(T entity)
475    {
476        UnitOfWork.Update(entity);
477    }
478
479    public void Save()
480    {
481        UnitOfWork.Save();
482    }
483
484    public void Commit()
485    {
486        UnitOfWork.Commit();
487    }
488
489    public void Rollback()
490    {
491        UnitOfWork.Rollback();
492    }
493
494    public void Add(T entity)
495    {
496        UnitOfWork.Add(entity);
497    }
498
499    public void Remove(T entity)
500    {
501        UnitOfWork.Remove(entity);
502    }
503
504    public void Update(T entity)
505    {
506        UnitOfWork.Update(entity);
507    }
508
509    public void Save()
510    {
511        UnitOfWork.Save();
512    }
513
514    public void Commit()
515    {
516        UnitOfWork.Commit();
517    }
518
519    public void Rollback()
520    {
521        UnitOfWork.Rollback();
522    }
523
524    public void Add(T entity)
525    {
526        UnitOfWork.Add(entity);
527    }
528
529    public void Remove(T entity)
530    {
531        UnitOfWork.Remove(entity);
532    }
533
534    public void Update(T entity)
535    {
536        UnitOfWork.Update(entity);
537    }
538
539    public void Save()
540    {
541        UnitOfWork.Save();
542    }
543
544    public void Commit()
545    {
546        UnitOfWork.Commit();
547    }
548
549    public void Rollback()
550    {
551        UnitOfWork.Rollback();
552    }
553
554    public void Add(T entity)
555    {
556        UnitOfWork.Add(entity);
557    }
558
559    public void Remove(T entity)
560    {
561        UnitOfWork.Remove(entity);
562    }
563
564    public void Update(T entity)
565    {
566        UnitOfWork.Update(entity);
567    }
568
569    public void Save()
570    {
571        UnitOfWork.Save();
572    }
573
574    public void Commit()
575    {
576        UnitOfWork.Commit();
577    }
578
579    public void Rollback()
580    {
581        UnitOfWork.Rollback();
582    }
583
584    public void Add(T entity)
585    {
586        UnitOfWork.Add(entity);
587    }
588
589    public void Remove(T entity)
590    {
591        UnitOfWork.Remove(entity);
592    }
593
594    public void Update(T entity)
595    {
596        UnitOfWork.Update(entity);
597    }
598
599    public void Save()
600    {
601        UnitOfWork.Save();
602    }
603
604    public void Commit()
605    {
606        UnitOfWork.Commit();
607    }
608
609    public void Rollback()
610    {
611        UnitOfWork.Rollback();
612    }
613
614    public void Add(T entity)
615    {
616        UnitOfWork.Add(entity);
617    }
618
619    public void Remove(T entity)
620    {
621        UnitOfWork.Remove(entity);
622    }
623
624    public void Update(T entity)
625    {
626        UnitOfWork.Update(entity);
627    }
628
629    public void Save()
630    {
631        UnitOfWork.Save();
632    }
633
634    public void Commit()
635    {
636        UnitOfWork.Commit();
637    }
638
639    public void Rollback()
640    {
641        UnitOfWork.Rollback();
642    }
643
644    public void Add(T entity)
645    {
646        UnitOfWork.Add(entity);
647    }
648
649    public void Remove(T entity)
650    {
651        UnitOfWork.Remove(entity);
652    }
653
654    public void Update(T entity)
655    {
656        UnitOfWork.Update(entity);
657    }
658
659    public void Save()
660    {
661        UnitOfWork.Save();
662    }
663
664    public void Commit()
665    {
666        UnitOfWork.Commit();
667    }
668
669    public void Rollback()
670    {
671        UnitOfWork.Rollback();
672    }
673
674    public void Add(T entity)
675    {
676        UnitOfWork.Add(entity);
677    }
678
679    public void Remove(T entity)
680    {
681        UnitOfWork.Remove(entity);
682    }
683
684    public void Update(T entity)
685    {
686        UnitOfWork.Update(entity);
687    }
688
689    public void Save()
690    {
691        UnitOfWork.Save();
692    }
693
694    public void Commit()
695    {
696        UnitOfWork.Commit();
697    }
698
699    public void Rollback()
700    {
701        UnitOfWork.Rollback();
702    }
703
704    public void Add(T entity)
705    {
706        UnitOfWork.Add(entity);
707    }
708
709    public void Remove(T entity)
710    {
711        UnitOfWork.Remove(entity);
712    }
713
714    public void Update(T entity)
715    {
716        UnitOfWork.Update(entity);
717    }
718
719    public void Save()
720    {
721        UnitOfWork.Save();
722    }
723
724    public void Commit()
725    {
726        UnitOfWork.Commit();
727    }
728
729    public void Rollback()
730    {
731        UnitOfWork.Rollback();
732    }
733
734    public void Add(T entity)
735    {
736        UnitOfWork.Add(entity);
737    }
738
739    public void Remove(T entity)
740    {
741        UnitOfWork.Remove(entity);
742    }
743
744    public void Update(T entity)
745    {
746        UnitOfWork.Update(entity);
747    }
748
749    public void Save()
750    {
751        UnitOfWork.Save();
752    }
753
754    public void Commit()
755    {
756        UnitOfWork.Commit();
757    }
758
759    public void Rollback()
760    {
761        UnitOfWork.Rollback();
762    }
763
764    public void Add(T entity)
765    {
766        UnitOfWork.Add(entity);
767    }
768
769    public void Remove(T entity)
770    {
771        UnitOfWork.Remove(entity);
772    }
773
774    public void Update(T entity)
775    {
776        UnitOfWork.Update(entity);
777    }
778
779    public void Save()
780    {
781        UnitOfWork.Save();
782    }
783
784    public void Commit()
785    {
786        UnitOfWork.Commit();
787    }
788
789    public void Rollback()
790    {
791        UnitOfWork.Rollback();
792    }
793
794    public void Add(T entity)
795    {
796        UnitOfWork.Add(entity);
797    }
798
799    public void Remove(T entity)
800    {
801        UnitOfWork.Remove(entity);
802    }
803
804    public void Update(T entity)
805    {
806        UnitOfWork.Update(entity);
807    }
808
809    public void Save()
810    {
811        UnitOfWork.Save();
812    }
813
814    public void Commit()
815    {
816        UnitOfWork.Commit();
817    }
818
819    public void Rollback()
820    {
821        UnitOfWork.Rollback();
822    }
823
824    public void Add(T entity)
825    {
826        UnitOfWork.Add(entity);
827    }
828
829    public void Remove(T entity)
830    {
831        UnitOfWork.Remove(entity);
832    }
833
834    public void Update(T entity)
835    {
836        UnitOfWork.Update(entity);
837    }
838
839    public void Save()
840    {
841        UnitOfWork.Save();
842    }
843
844    public void Commit()
845    {
846        UnitOfWork.Commit();
847    }
848
849    public void Rollback()
850    {
851        UnitOfWork.Rollback();
852    }
853
854    public void Add(T entity)
855    {
856        UnitOfWork.Add(entity);
857    }
858
859    public void Remove(T entity)
860    {
861        UnitOfWork.Remove(entity);
862    }
863
864    public void Update(T entity)
865    {
866        UnitOfWork.Update(entity);
867    }
868
869    public void Save()
870    {
871        UnitOfWork.Save();
872    }
873
874    public void Commit()
875    {
876        UnitOfWork.Commit();
877    }
878
879    public void Rollback()
880    {
881        UnitOfWork.Rollback();
882    }
883
884    public void Add(T entity)
885    {
886        UnitOfWork.Add(entity);
887    }
888
889    public void Remove(T entity)
890    {
891        UnitOfWork.Remove(entity);
892    }
893
894    public void Update(T entity)
895    {
896        UnitOfWork.Update(entity);
897    }
898
899    public void Save()
900    {
901        UnitOfWork.Save();
902    }
903
904    public void Commit()
905    {
906        UnitOfWork.Commit();
907    }
908
909    public void Rollback()
910    {
911        UnitOfWork.Rollback();
912    }
913
914    public void Add(T entity)
915    {
916        UnitOfWork.Add(entity);
917    }
918
919    public void Remove(T entity)
920    {
921        UnitOfWork.Remove(entity);
922    }
923
924    public void Update(T entity)
925    {
926        UnitOfWork.Update(entity);
927    }
928
929    public void Save()
930    {
931        UnitOfWork.Save();
932    }
933
934    public void Commit()
935    {
936        UnitOfWork.Commit();
937    }
938
939    public void Rollback()
940    {
941        UnitOfWork.Rollback();
942    }
943
944    public void Add(T entity)
945    {
946        UnitOfWork.Add(entity);
947    }
948
949    public void Remove(T entity)
950    {
951        UnitOfWork.Remove(entity);
952    }
953
954    public void Update(T entity)
955    {
956        UnitOfWork.Update(entity);
957    }
958
959    public void Save()
960    {
961        UnitOfWork.Save();
962    }
963
964    public void Commit()
965    {
966        UnitOfWork.Commit();
967    }
968
969    public void Rollback()
970    {
971        UnitOfWork.Rollback();
972    }
973
974    public void Add(T entity)
975    {
976        UnitOfWork.Add(entity);
977    }
978
979    public void Remove(T entity)
980    {
981        UnitOfWork.Remove(entity);
982    }
983
984    public void Update(T entity)
985    {
986        UnitOfWork.Update(entity);
987    }
988
989    public void Save()
990    {
991        UnitOfWork.Save();
992    }
993
994    public void Commit()
995    {
996        UnitOfWork.Commit();
997    }
998
999    public void Rollback()
1000    {
1001        UnitOfWork.Rollback();
1002    }
1003
1004    public void Add(T entity)
1005    {
1006        UnitOfWork.Add(entity);
1007    }
1008
1009    public void Remove(T entity)
1010    {
1011        UnitOfWork.Remove(entity);
1012    }
1013
1014    public void Update(T entity)
1015    {
1016        UnitOfWork.Update(entity);
1017    }
1018
1019    public void Save()
1020    {
1021        UnitOfWork.Save();
1022    }
1023
1024    public void Commit()
1025    {
1026        UnitOfWork.Commit();
1027    }
1028
1029    public void Rollback()
1030    {
1031        UnitOfWork.Rollback();
1032    }
1033
1034    public void Add(T entity)
1035    {
1036        UnitOfWork.Add(entity);
1037    }
1038
1039    public void Remove(T entity)
1040    {
1041        UnitOfWork.Remove(entity);
1042    }
1043
1044    public void Update(T entity)
1045    {
1046        UnitOfWork.Update(entity);
1047    }
1048
1049    public void Save()
1050    {
1051        UnitOfWork.Save();
1052    }
1053
1054    public void Commit()
1055    {
1056        UnitOfWork.Commit();
1057    }
1058
1059    public void Rollback()
1060    {
1061        UnitOfWork.Rollback();
1062    }
1063
1064    public void Add(T entity)
1065    {
1066        UnitOfWork.Add(entity);
1067    }
1068
1069    public void Remove(T entity)
1070    {
1071        UnitOfWork.Remove(entity);
1072    }
1073
1074    public void Update(T entity)
1075    {
1076        UnitOfWork.Update(entity);
1077    }
1078
1079    public void Save()
1080    {
1081        UnitOfWork.Save();
1082    }
1083
1084    public void Commit()
1085    {
1086        UnitOfWork.Commit();
1087    }
1088
1089    public void Rollback()
1090    {
1091        UnitOfWork.Rollback();
1092    }
1093
1094    public void Add(T entity)
1095    {
1096        UnitOfWork.Add(entity);
1097    }
1098
1099    public void Remove(T entity)
1100    {
1101        UnitOfWork.Remove(entity);
1102    }
1103
1104    public void Update(T entity)
1105    {
1106        UnitOfWork.Update(entity);
1107    }
1108
1109    public void Save()
1110    {
1111        UnitOfWork.Save();
1112    }
1113
1114    public void Commit()
1115    {
1116        UnitOfWork.Commit();
1117    }
1118
1119    public void Rollback()
1120    {
1121        UnitOfWork.Rollback();
1122    }
1123
1124    public void Add(T entity)
1125    {
1126        UnitOfWork.Add(entity);
1127    }
1128
1129    public void Remove(T entity)
1130    {
1131        UnitOfWork.Remove(entity);
1132    }
1133
1134    public void Update(T entity)
1135    {
1136        UnitOfWork.Update(entity);
1137    }
1138
1139    public void Save()
1140    {
1141        UnitOfWork.Save();
1142    }
1143
1144    public void Commit()
1145    {
1146        UnitOfWork.Commit();
1147    }
1148
1149    public void Rollback()
1150    {
1151        UnitOfWork.Rollback();
1152    }
1153
1154    public void Add(T entity)
1155    {
1156        UnitOfWork.Add(entity);
1157    }
1158
1159    public void Remove(T entity)
1160    {
1161        UnitOfWork.Remove(entity);
1162    }
1163
1164    public void Update(T entity)
1165    {
1166        UnitOfWork.Update(entity);
1167    }
1168
1169    public void Save()
1170    {
1171        UnitOfWork.Save();
1172    }
1173
1174    public void Commit()
1175    {
1176        UnitOfWork.Commit();
1177    }
1178
1179    public void Rollback()
1180    {
1181        UnitOfWork.Rollback();
1182    }
1183
1184    public void Add(T entity)
1185    {
1186        UnitOfWork.Add(entity);
1187    }
1188
1189    public void Remove(T entity)
1190    {
1191        UnitOfWork.Remove(entity);
1192    }
1193
1194    public void Update(T entity)
1195    {
1196        UnitOfWork.Update(entity);
1197    }
1198
1199    public void Save()
1200    {
1201        UnitOfWork.Save();
1202    }
1203
1204    public void Commit()
1205    {
1206        UnitOfWork.Commit();
1207    }
1208
1209    public void Rollback()
1210    {
1211        UnitOfWork.Rollback();
1212    }
1213
1214    public void Add(T entity)
1215    {
1216        UnitOfWork.Add(entity);
1217    }
1218
1219    public void Remove(T entity)
1220    {
1221        UnitOfWork.Remove(entity);
1222    }
1223
1224    public void Update(T entity)
1225    {
1226        UnitOfWork.Update(entity);
1227    }
1228
1229    public void Save()
1230    {
1231        UnitOfWork.Save();
1232    }
1233
1234    public void Commit()
1235    {
1236        UnitOfWork.Commit();
1237    }
1238
1239    public void Rollback()
1240    {
1241        UnitOfWork.Rollback();
1242    }
1243
1244    public void Add(T entity)
1245    {
1246        UnitOfWork.Add(entity);
1247    }
1248
1249    public void Remove(T entity)
1250    {
1251        UnitOfWork.Remove(entity);
1252    }
1253
1254    public void Update(T entity)
1255    {
1256        UnitOfWork.Update(entity);
1257    }
1258
1259    public void Save()
1260    {
1261        UnitOfWork.Save();
1262    }
1263
1264    public void Commit()
1265    {
1266        UnitOfWork.Commit();
1267    }
1268
1269    public void Rollback()
1270    {
1271        UnitOfWork.Rollback();
1272    }
1273
1274    public void Add(T entity)
1275    {
1276        UnitOfWork.Add(entity);
1277    }
1278
1279    public void Remove(T entity)
1280    {
1281        UnitOfWork.Remove(entity);
1282    }
1283
1284    public void Update(T entity)
1285    {
1286        UnitOfWork.Update(entity);
1287    }
1288
1289    public void Save()
1290    {
1291        UnitOfWork.Save();
1292    }
1293
1294    public void Commit()
1295    {
1296        UnitOfWork.Commit();
1297    }
1298
1299    public void Rollback()
1300    {
1301        UnitOfWork.Rollback();
1302    }
1303
1304    public void Add(T entity)
1305    {
1306        UnitOfWork.Add(entity);
1307    }
1308
1309    public void Remove(T entity)
1310    {
1311        UnitOfWork.Remove(entity);
1312    }
1313
1314    public void Update(T entity)
1315    {
1316        UnitOfWork.Update(entity);
1317    }
1318
1319    public void Save()
1320    {
1321        UnitOfWork.Save();
1322    }
1323
1324    public void Commit()
1325    {
1326        UnitOfWork.Commit();
1327    }
1328
1329    public void Rollback()
1330    {
1331        UnitOfWork.Rollback();
1332    }
1333
1334    public void Add(T entity)
1335    {
1336        UnitOfWork.Add(entity);
1337    }
1338
1339    public void Remove(T entity)
1340    {
1341        UnitOfWork.Remove(entity);
1342    }
1343
1344    public void Update(T entity)
1345    {
1346        UnitOfWork.Update(entity);
1347    }
1348
1349    public void Save()
1350    {
1351        UnitOfWork.Save();
1352    }
1353
1354    public void Commit()
1355    {
1356        UnitOfWork.Commit();
1357    }
1358
1359    public void Rollback()
1360    {
1361        UnitOfWork.Rollback();
1362    }
1363
1364    public void Add(T entity)
1365    {
1366        UnitOfWork.Add(entity);
1367    }
1368
1369    public void Remove(T entity)
1370    {
1371        UnitOfWork.Remove(entity);
1372    }
1373
1374    public void Update(T entity)
1375    {
1376        UnitOfWork.Update(entity);
1377    }
1378
1379    public void Save()
1380    {
1381        UnitOfWork.Save();
1382    }
1383
1384    public void Commit()
1385    {
1386        UnitOfWork.Commit();
1387    }
1388
1389    public void Rollback()
1390    {
1391        UnitOfWork.Rollback();
1392    }
1393
1394    public void Add(T entity)
1395    {
1396        UnitOfWork.Add(entity);
1397    }
1398
1399    public void Remove(T entity)
1400    {
1401        UnitOfWork.Remove(entity);
1402    }
1403
1404    public void Update(T entity)
1405    {
1406        UnitOfWork.Update(entity);
1407    }
1408
1409    public void Save()
1410    {
1411        UnitOfWork.Save();
1412    }
1413
1414    public void Commit()
1415    {
1416        UnitOfWork.Commit();
1417    }
1418
1419    public void Rollback()
1420    {
1421        UnitOfWork.Rollback();
1422    }
1423
1424    public void Add(T entity)
1425    {
1426        UnitOfWork.Add(entity);
1427    }
1428
1429    public void Remove(T entity)
1430    {
1431        UnitOfWork.Remove(entity);
1432    }
1433
1434    public void Update(T entity)
1435    {
1436        UnitOfWork.Update(entity);
1437    }
1438
1439    public void Save()
1440    {
1441        UnitOfWork.Save();
1442    }
1443
1444    public void Commit()
1445    {
1446        UnitOfWork.Commit();
1447    }
1448
1449    public void Rollback()
1450    {
1451        UnitOfWork.Rollback();
1452    }
1453
1454    public void Add(T entity)
1455    {
1456        UnitOfWork.Add(entity);
1457    }
1458
1459    public void Remove(T entity)
1460    {
1461        UnitOfWork.Remove(entity);
1462    }
1463
1464    public void Update(T entity)
1465    {
1466        UnitOfWork.Update(entity);
1467    }
1468
1469    public void Save()
1470    {
1471        UnitOfWork.Save();
1472    }
1473
1474    public void Commit()
1475    {
1476        UnitOfWork.Commit();
1477    }
1478
1479    public void Rollback()
1480    {
1481        UnitOfWork.Rollback();
1482    }
1483
1484    public void Add(T entity)
1485    {
1486        UnitOfWork.Add(entity);
1487    }
1488
1489    public void Remove(T entity)
1490    {
1491        UnitOfWork.Remove(entity);
1492    }
1493
1494    public void Update(T entity)
1495    {
1496        UnitOfWork.Update(entity);
1497    }
1498
1499    public void Save()
1500    {
1501        UnitOfWork.Save();
1502    }
1503
1504    public void Commit()
1505    {
1506        UnitOfWork.Commit();
1507    }
1508
1509    public void Rollback()
1510    {
1511        UnitOfWork.Rollback();
1512    }
1513
1514    public void Add(T entity)
1515    {
1516        UnitOfWork.Add(entity);
1517    }
1518
1519    public void Remove(T entity)
1520    {
1521        UnitOfWork.Remove(entity);
1522    }
1523
1524    public void Update(T entity)
1525    {
1526        UnitOfWork.Update(entity);
1527    }
1528
1529    public void Save()
1530    {
1531        UnitOfWork.Save();
1532    }
1533
1534    public void Commit()
1535    {
1536        UnitOfWork.Commit();
1537    }
1538
1539    public void Rollback()
1540    {
1541        UnitOfWork.Rollback();
1542    }
1543
1544    public void Add(T entity)
1545    {
1546        UnitOfWork.Add(entity);
1547    }
1548
1549    public void Remove(T entity)
1550    {
1551        UnitOfWork.Remove(entity);
1552    }
1553
1554    public void Update(T entity)
1555    {
1556        UnitOfWork.Update(entity);
1557    }
1558
1559    public void Save()
156
```

JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 30 / 59

X -L'interface « IParamRepository » dans le projet « .Domaine »

- L'interface « **IParamRepository** » va définir la liste des méthodes à utiliser dans la classe « **DefaultParamRepository** » (projet « **API.Demo.Param.CleanCode.Infrastructures** »)
- L'interface « **IParamRepository** » est créée dans le projet « **API.Demo.Param.CleanCode.Domain** »



JC CERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 31 / 59

DefaultParamRepository.cs ApiController.cs IParamRepository.cs CarModels.cs

API.Demo.Param.CleanCode.Domain API.Demo.Param.CleanCode.Domain.IPa UpdateCarModel(CarModels Model)

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  //> Pour " IRepository "
8  using API.Demo.Param.CleanCode.Framework;
9
10 namespace API.Demo.Param.CleanCode.Domain
11 {
12     /// <summary>
13     /// Interface Manage Repository
14     /// </summary>
15     public interface IParamRepository : IRepository
16     {
17         /// <summary>
18         /// Ajout d'une marque de voiture
19         /// </summary>
20         /// <param name="brand"></param>
21         /// <returns></returns>
22         public CarBrand AddOneCarBrand (CarBrand brand);
23
24         /// <summary>
25         /// Mettre à jour une marque
26         /// </summary>
27         /// <param name="brand"></param>
28         /// <returns></returns>
29         public CarBrand UpdateCarBrand(CarBrand brand);
30
31         /// <summary>
32         /// Renvo liste de marque ( ou d'une marque )
33         /// </summary>
34         /// <param name="ID_CarBrand"></param>
35         /// <returns></returns>
36         public ICollection<CarBrand> GetAllCarBrand (string ID_CarBrand);
37
38     }

```

Interface "IParamRepository"

Implémentation des méthodes pour gérer les marques & modèles de voitures.

Méthodes pour gérer les marques de voitures.

1

Explorateur de solutions

Solution « API.Demo.Param.CleanCode » (4 sur 4 de projets)

- Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - Car.cs
 - CarBrand.cs
 - CarModels.cs
 - IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructures
 - Dépendances
 - Data
 - TypeConfiguration
 - CFG_Car.cs
 - CFG_CarBrand.cs
 - CFG_CarModels.cs
 - Repositories
 - DefaultParamRepository.cs
 - DataContext.cs
 - API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - DTOs
 - Controllers
 - appsettings.Development.json
 - appsettings.json
 - Program.cs

DefaultParamRepository.cs ApiController.cs IParamRepository.cs CarModels.cs

API.Demo.Param.CleanCode.Domain API.Demo.Param.CleanCode.Domain.IPa UpdateCarModel(CarModels Model)

```

39
40     /// <summary>
41     /// Ajout d'un modèle de voiture
42     /// </summary>
43     /// <param name="Model"></param>
44     /// <returns></returns>
45     public CarModels AddOneCarModel(CarModels Model);
46
47
48     /// <summary>
49     /// Mettre à jour un modèle de voiture
50     /// </summary>
51     /// <param name="Model"></param>
52     /// <returns></returns>
53     public CarModels UpdateCarModel(CarModels Model);
54
55
56     /// <summary>
57     /// Liste des modèles pour une marque
58     /// </summary>
59     /// <param name="ID_CarBrand"></param>
60     /// <returns></returns>
61     public ICollection<CarModels> GetAllCarModel(string ID_CarBrand);
62
63
64 }

```

Interface "IParamRepository"

Méthodes pour gérer les modèles de voitures.

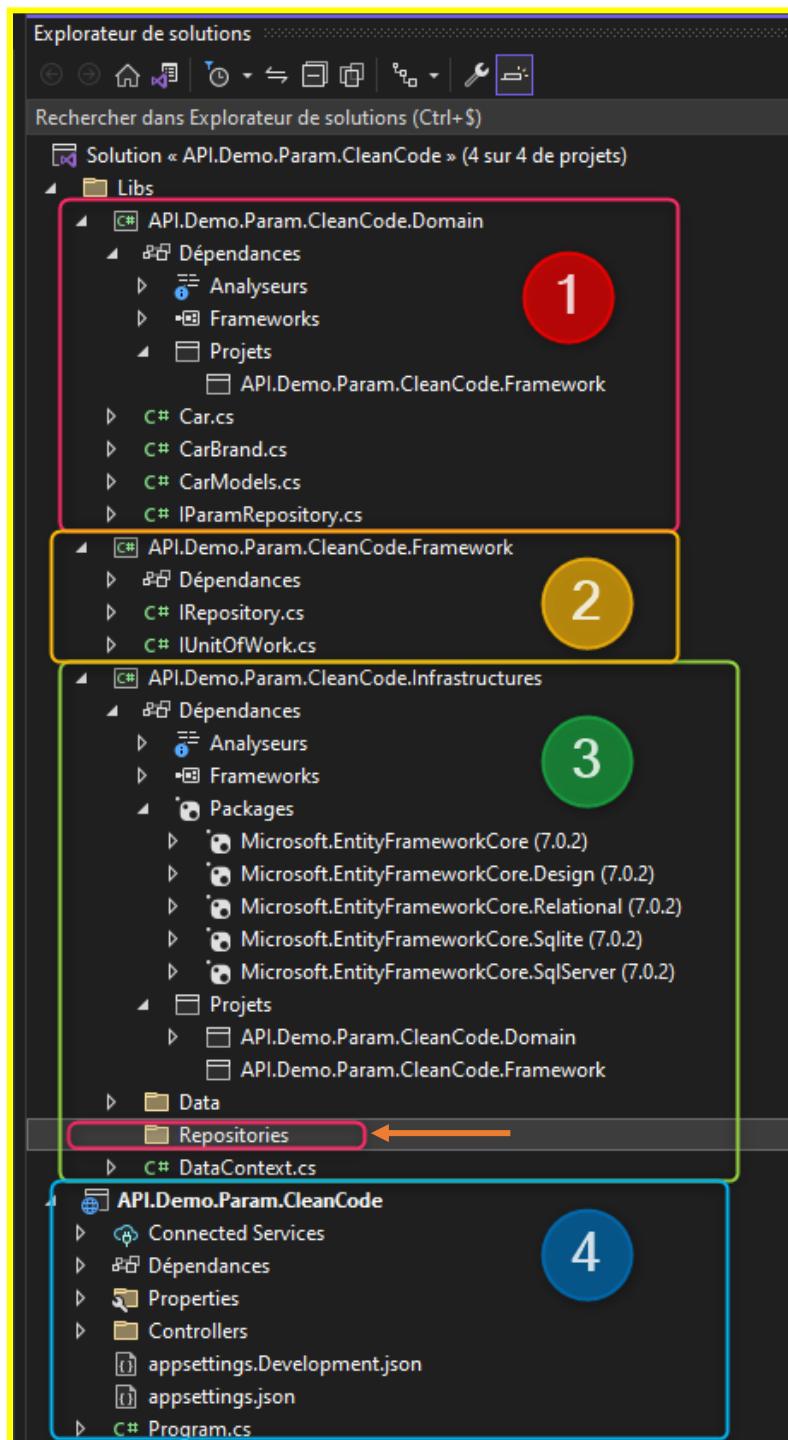
2



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 32 / 59

XI – La classe « DefaultParamRepository » dans le projet « .Infrastructures »

XI-A Créer le dossier « Repositories » dans le projet « API.Demo.Param.CleanCode.Infrastructures »



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 33 / 59

XI – B La classe « DefaultParamRepository » (dans le dossier « Repositories »)

XI – B - 1 Les marques de voitures

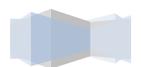
Classe "DefaultParamRepository"
Code partie 1

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  // > Pour "IParamRepository" <
8  using API.Demo.Param.CleanCode.Domain;
9
10 // > Pour "IUnitOfWork" <
11 using API.Demo.Param.CleanCode.Framework;
12
13 // > Pour "Include" <
14 using Microsoft.EntityFrameworkCore;
15
16 namespace API.Demo.Param.CleanCode.Infrastructure.Repositories
17 {
18     public class DefaultParamRepository : IParamRepository
19     {
20         #region fields
21         // > Déclare le context qui sera alimenté par injection de dépendance <
22         public readonly DataContext _Context = null;
23         #endregion
24
25         #region property
26         // > Installe le "SaveChange".
27         public IUnitOfWork UnitOfWork => this._Context;
28         #endregion
29
30         #region methods
31         /// <summary>
32         /// Constructeur de la classe
33         /// </summary>
34         /// <param name="Context"></param>
35         public DefaultParamRepository(DataContext Context)
36         {
37             // > Récupération du context <
38             this._Context = Context;
39         }
        
```

Explorateur de solutions

- Solution « API.Demo.Param.CleanCode » (4 sur 4 de projets)
 - Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - Car.cs
 - CarBrand.cs
 - CarModels.cs
 - IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructure
 - Dépendances
 - Data
 - TypeConfiguration
 - CFG_Car.cs
 - CFG_CarBrand.cs
 - CFG_CarModels.cs
 - Repositories
 - DefaultParamRepository.cs
 - DataContext
 - DataContext()
 - DataContext(DbContextOptions)
 - OnModelCreating(ModelBuilder) : void
 - Brand : DbSet<CarBrand>
 - Models : DbSet<CarModels>
 - Car : DbSet<Car>
 - API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - Controllers



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 34 / 59

DefaultParamRepository.cs* → ApiController.cs IParamRepository.cs CarModels.cs CarBrandResumeDTO.cs

API.Demo.Param.CleanCode.Infrastructures → API.Demo.Param.CleanCode.Infrastructures.Repo DefaultParamRepository(DataContext Context)

```

72  /// <summary>
73  /// Mettre à jour une voiture
74  /// </summary>
75  /// <param name="Item"></param>
76  /// <returns></returns>
77  2 références
78  public CarBrand UpdateCarBrand(CarBrand Item)
79  {
80      // > Lecture d'un enregistrement et mise à jour de celui ci <
81      var Brand = this._Context.Brand.FirstOrDefault(f => f.ID_CarBrand == Item.ID_CarBrand);
82
83      // > Si l'enregistrement LU n'est PAS null <
84      if (Brand != null)
85      {
86          // > Libellé <
87          Brand.Libelle = Item.Libelle;
88      }
89      else
90      {
91          Brand = null;
92      }
93      // > On retourne l'entité tracé par l'ORM <
94      return this._Context.Brand.Update(Brand).Entity;
95
96  }
97

```

Classe "DefaultParamRepository"
Code partie 1

2

DefaultParamRepository.cs* → ApiController.cs IParamRepository.cs CarModels.cs CarBrandResumeDTO.cs

API.Demo.Param.CleanCode.Infrastructures → API.Demo.Param.CleanCode.Infrastructures.Repo DefaultParamRepository(DataContext Context)

```

52  /// <summary>
53  /// Renvoie liste de marque ( ou une marque si ID transmis ).]
54  /// </summary>
55  /// <param name="ID_CarBrand"></param>
56  /// <returns></returns>
57  2 références
58  public ICollection<CarBrand> GetAllCarBrand (string ID_CarBrand)
59  {
60      // > Extraction collection des CarBrand (MARQUE DE VOITURES) <
61      //   ( "AsQueryable()" pour autoriser derrière une clause "where" )
62      var Elements = this._Context.Brand.AsQueryable();
63
64      // > Si un ID est transmis, on ajoute une clause "where"
65      if (ID_CarBrand != "")
66      {
67          Elements = Elements.Where(item => item.ID_CarBrand == ID_CarBrand);
68      }
69      return Elements.ToList();
70
71
72

```

Classe "DefaultParamRepository"
Code partie 3

3



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 35 / 59

XI – B - 2 Les modèles de voitures

Classe "DefaultParamRepository"
Code partie 4

4

```
94
95     /// <summary>
96     /// Ajout d'un modèle de voiture
97     /// </summary>
98     /// <param name="Item"></param>
99     /// <returns></returns>
100    2 références
101    public CarModels AddOneCarModel(CarModels Item)
102    {
103        // > On retourne l'entité tracé par l'ORM <
104        return this._Context.Models.Add(Item).Entity;
105    }
106
107
108    /// <summary>
109    /// Maj d'un modèle de voiture
110    /// </summary>
111    /// <param name="Item"></param>
112    /// <returns></returns>
113    2 références
114    public CarModels UpdateCarModel(CarModels Item)
115    {
116        // > Lecture d'un enregistrement et mise à jour de celui ci <
117        var ModelCar = this._Context.Models.FirstOrDefault(f => f.ID_CarModel== Item.ID_CarModel);
118
119        // > Si l'enregistrement LU n'est PAS null <
120        if (ModelCar != null)
121        {
122            // > clé étrangère Marque de voiture <
123            ModelCar.FK_ID_CarBrand = Item.FK_ID_CarBrand;
124
125            // > Libellé <
126            ModelCar.Libelle = Item.Libelle;
127        }
128        else
129        {
130            ModelCar = null;
131        }
132        // > On retourne l'entité tracé par l'ORM <
133        return this._Context.Models.Update(ModelCar).Entity;
134    }
135}
```



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 36 / 59

- **Remarque :** La méthode « **GetAllCarModels** » utilise les tables de configuration « **CFG_CarBrand.cs** » et « **CFG_CarModels.cs** » pour l'utilisation du « **Include** ».

DefaultParamRepository.cs* → ApiController.cs IParamRepository.cs CarModels.cs CarBrandResumeDTO.cs

API.Demo.Param.CleanCode.Infrastructure → API.Demo.Param.CleanCode.Infrastructure.Reposit GetAllCarModel(string ID_CarBrand)

```

135
136
137
138     /// <summary>
139     /// Renvoyer liste modèles pour une marque
140     /// </summary>
141     /// <param name="ID_CarBrand"></param>
142     /// <returns></returns>
143
144     public ICollection<CarModels> GetAllCarModel(string ID_CarBrand)
145     {
146         // > "Include" se sert des fichiers de configuration "CFG_xxx" pour fonctionner <
147         //   ( La relation est en effet définie dans ces tables de configuration )
148         var ListModelCar = this._Context.Models.Include(Item => Item.Brand).AsQueryable();
149
150         // > Pour la sélection, on s'appuie ici (pour l'exemple) sur le membre...
151         //   ... du modèle "Carbrand" inclue dans la requête <
152
153         // > Mais On aurait pu aussi travailler avec la cré étrangère "FK_ID_CarBrand" <
154         if (ID_CarBrand != null )
155         {
156             ListModelCar = ListModelCar.Where(f => f.Brand.ID_CarBrand == ID_CarBrand);
157         }
158
159         return ListModelCar.ToList();
160
161     }
162
163     #endregion

```

Classe "DefaultParamRepository"
Code partie 5

5



JC CERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 37 / 59

XII – Configurer les Connection Strings (« appsettings.json »)

- Il faut indiquer maintenant les connexions aux bases de données. On utilise pour cela le fichier Json « **appsettings** ».
- Il y a une base de données pour les tests (j'utilise **SQLite**).
- Et il y a une base de données de production : on peut utiliser **SQLServer**, ou bien **Azure**.
- La connexion « **DEV_BDD** » pointe sur une base de développement (SQLite)
- La connexion « **PROD_BDD** » pointe sur une base de développement SQLServer (ou bien Azure).

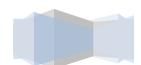
```

Program.cs    CarModels.cs    appsettings.json    X
Schéma : https://json.schemastore.org/appsettings.json
1  {
2      "Logging": {
3          "LogLevel": {
4              "Default": "Information",
5              "Microsoft.AspNetCore": "Warning"
6          }
7      },
8      "AllowedHosts": "*",
9
10     "ConnectionStrings": {
11         "DEV_BDD": "Data Source=DemoParam.db",
12         "PROD_BDD": "Server=xxxxxxxxx"
13     }
14
15
16
17

```

Explorateur de solutions

- Solution « API.Demo.Param.CleanCode » (4 sur 4 de projets)
 - Libs
 - API.Demo.Param.CleanCode.Domain
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructure
 - API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Controllers
 - appsettings.Development.json
 - appsettings.json
- Program.cs



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 38 / 59

XIII – Classe « Program.cs » (.Net6)

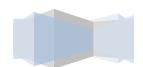
XIII – A Mettre en place l'injection de dépendance

```

1 // > Pour "IParamRepository" & "DefaultParamRepository" <
2 using API.Demo.Param.CleanCode.Infrastructures.Repositories;
3 using API.Demo.Param.CleanCode.Domain;
4 // -----
5 // #####      --- Program.cs --- #####
6 // #####      **** Programme de Démarrage ****      #####
7 // #####      (.Net 6)      #####
8 // -----
9 var builder = WebApplication.CreateBuilder(args);
10
11 // -----
12 // #####      *** Entity Framework Core .Net6 ***      #####
13 // ##### On met en place le lien entre l'interface et la classe qui respecte ce contrat ici. #####
14 // ##### "builder.Services.AddScoped<IParamRepository, DefaultParamRepository>();"
15 // #####      #####
16 // #####      #####
17 // ##### Le Framework pourra alors effectuer l'injection d'une instance de la classe...
18 // ##### "DefaultParamRepository" qui respecte le contrat de l'interface...
19 // ##### ... "IParamRepository".
20 // -----
21 builder.Services.AddScoped<IParamRepository, DefaultParamRepository>();
22
23 // Add services to the container.
24
25 builder.Services.AddControllers();
26 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
27 builder.Services.AddEndpointsApiExplorer();
28 builder.Services.AddSwaggerGen();
29
30 var app = builder.Build();
31
32 // Configure the HTTP request pipeline.
33 if (app.Environment.IsDevelopment())
34 {
35     app.UseSwagger();
36     app.UseSwaggerUI();
37 }
38
39 app.UseHttpsRedirection();
40
41 app.UseAuthorization();
42
43 app.MapControllers();
44
45 app.Run();
46

```

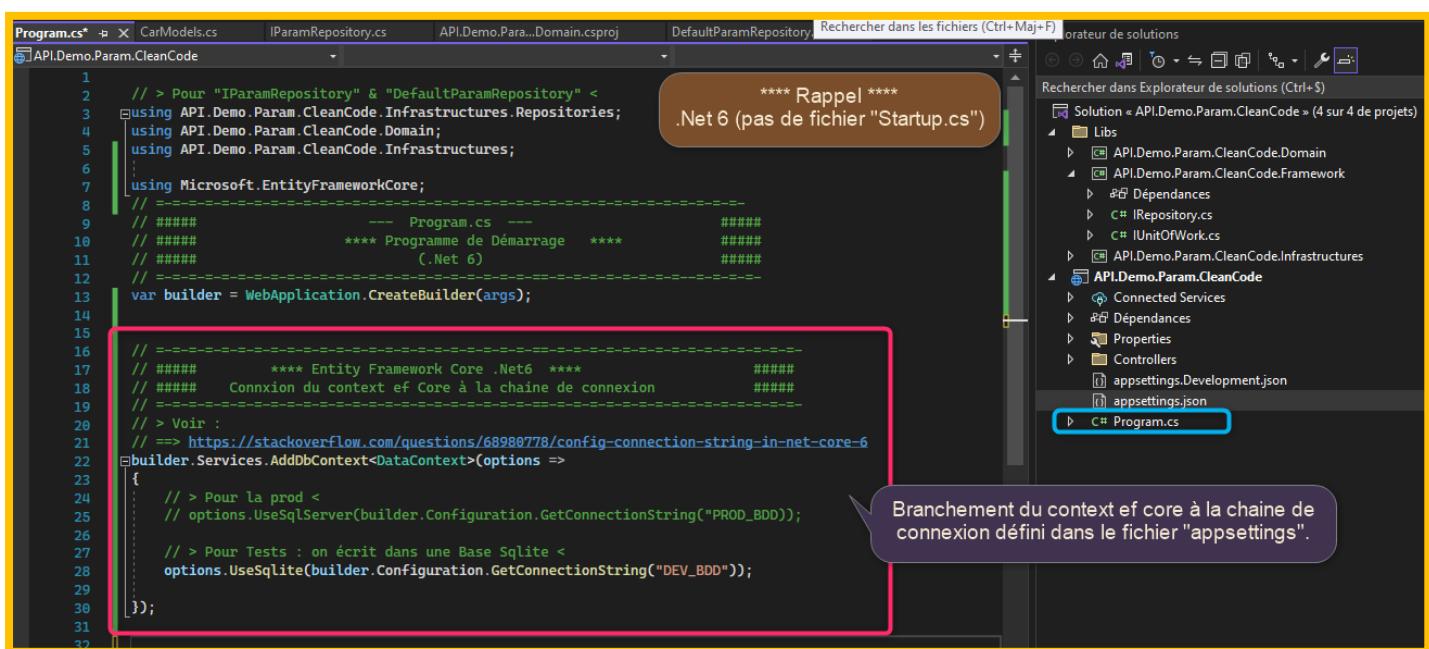
Mise en place injection de dépendance.



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 39 / 59

XIII –B Connecter le « context » ef Core avec la chaîne de connexion décrite dans « appsettings.json »

- Dans le fichier « **Program.cs** », on connecte le context (ici « **DataContext** ») à la chaîne de connexion définie dans le fichier « **appsettings.json** ».
- Dans le code ci-dessous, on connecte le context « **DataContext** » à la base de développement (DEV_BDD), c'est-à-dire une base SQLite



**** Rappel ****
.Net 6 (pas de fichier "Startup.cs")

```

1 // > Pour "IParamRepository" & "DefaultParamRepository" <
2 using API.Demo.Param.CleanCode.Infrastructures.Repositories;
3 using API.Demo.Param.CleanCode.Domain;
4 using API.Demo.Param.CleanCode.Infrastructures;
5 ...
6 using Microsoft.EntityFrameworkCore;
7 // -----
8 // #####           --- Program.cs ---           #####
9 // #####           *** Programme de Démarrage   ***
10 // #####           (.Net 6)                      #####
11 // -----
12 var builder = WebApplication.CreateBuilder(args);
13
14
15 // -----
16 // #####           *** Entity Framework Core .Net6 ***           #####
17 // #####           Connexion du context ef Core à la chaîne de connexion   #####
18 // -----
19 // > Voir :
20 // ==> https://stackoverflow.com/questions/68980778/config-connection-string-in-net-core-6
21 builder.Services.AddDbContext<DataContext>(options =>
22 {
23     // > Pour la prod <
24     // options.UseSqlServer(builder.Configuration.GetConnectionString("PROD_BDD"));
25
26     // > Pour Tests : on écrit dans une Base Sqlite <
27     options.UseSqlite(builder.Configuration.GetConnectionString("DEV_BDD"));
28
29 });
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
136
```

JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 40 / 59

XIII –C Crée la base de données (si elle n'existe pas déjà) au lancement de l'appli

- Toujours dans le fichier « **Program.cs** », on peut demander à ce que la base de données soit créée (si elle celle-ci est inexistante).
- La base de données est alors créée au lancement de l'API.

**** Net 6 ****

Program.cs
Partie 1 du code

1

```

1  // > Pour "IParamRepository" & "DefaultParamRepository" <
2  using API.Demo.Param.CleanCode.Infrastructures.Repositories;
3  using API.Demo.Param.CleanCode.Domain;
4  using API.Demo.Param.CleanCode.Infrastructures;
5  ...
6  using Microsoft.EntityFrameworkCore;
7  ...
8  // -----
9  // ##### --- Program.cs --- #####
10 // ##### **** Programme de Démarrage **** #####
11 // ##### (.Net 6) #####
12 // -----
13 var builder = WebApplication.CreateBuilder(args);
14
15
16 // -----
17 // ##### *** Entity Framework Core .Net6 *** #####
18 // ##### Connexion du contexte ef Core à la chaîne de connexion #####
19 // -----
20 // > Voir :
21 // ==> https://stackoverflow.com/questions/68980778/config-connection-string-in-net-core-6
22 builder.Services.AddDbContext<DataContext>(options =>
23 {
24     // > Pour la prod <
25     options.UseSqlServer(builder.Configuration.GetConnectionString("PROD_BDD"));
26
27     // > Pour Tests : on écrit dans une Base Sqlite <
28     options.UseSqlite(builder.Configuration.GetConnectionString("DEV_BDD"));
29 });
30
31 // -----
32 // ##### *** Entity Framework Core .Net6 *** #####
33 // ##### On met en place le lien entre l'interface et la classe qui respecte ce contrat ici. #####
34 // ##### "builder.Services.AddScoped<IParamRepository, DefaultParamRepository>();" #####
35 // #####
36 // ##### Le Framework pourra alors effectuer l'injection d'une instance de la classe... #####
37 // ##### "DefaultParamRepository" qui respecte le contrat de l'interface... #####
38 // ##### ... "IParamRepository". #####
39 // -----
40 // -----
41 builder.Services.AddScoped<IParamRepository, DefaultParamRepository>();
42

```

**** Net 6 ****

Program.cs
Partie 2 du code

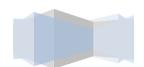
2

```

43  builder.Services.AddControllers();
44  // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
45  builder.Services.AddEndpointsApiExplorer();
46  builder.Services.AddSwaggerGen();
47
48
49  var app = builder.Build();
50
51  // Configure the HTTP request pipeline.
52  if (app.Environment.IsDevelopment())
53  {
54      app.UseSwagger();
55      app.UseSwaggerUI();
56  }
57
58  // > On se prépare à créer la BDD <
59  using (var scope = app.Services.CreateScope())
60  {
61      // -- Création des Tables si elles sont inexistantes -
62      var dbContext = scope.ServiceProvider.GetRequiredService<DataContext>();
63
64      // > Création des tables -
65      dbContext.Database.EnsureCreated();
66  }
67
68  app.UseHttpsRedirection();
69
70  app.UseAuthorization();
71
72  app.MapControllers();
73
74
75  app.Run();
76

```

Code pour lancer la création de la BDD au lancement de l'API.

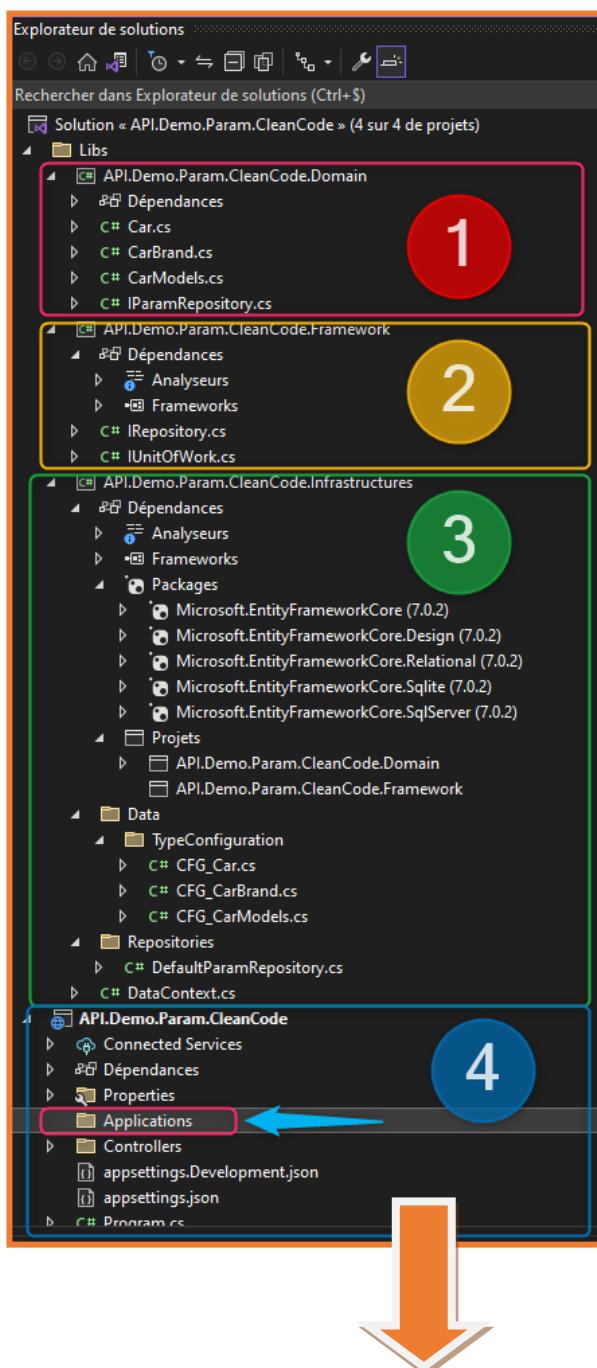


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 41 / 59

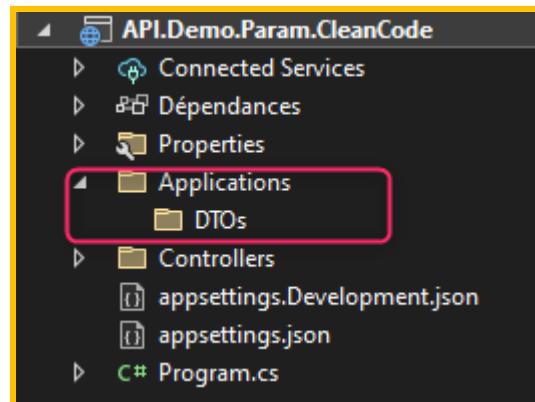
XIV – Les DTO (Data Transfert Object)

- Les DTO sont un type d'objets permettant les échanges entre l'API et l'extérieur.
- Il s'agit d'un objet qui définit la structure des informations à échanger.
- Les DTO's sont stockés dans le dossier « **Applications/DTOs** » du projet « **API.Demo.Param.CleanCode.Infrastructure** ».

XIV-A Créez le dossier « Applications/DTOs »

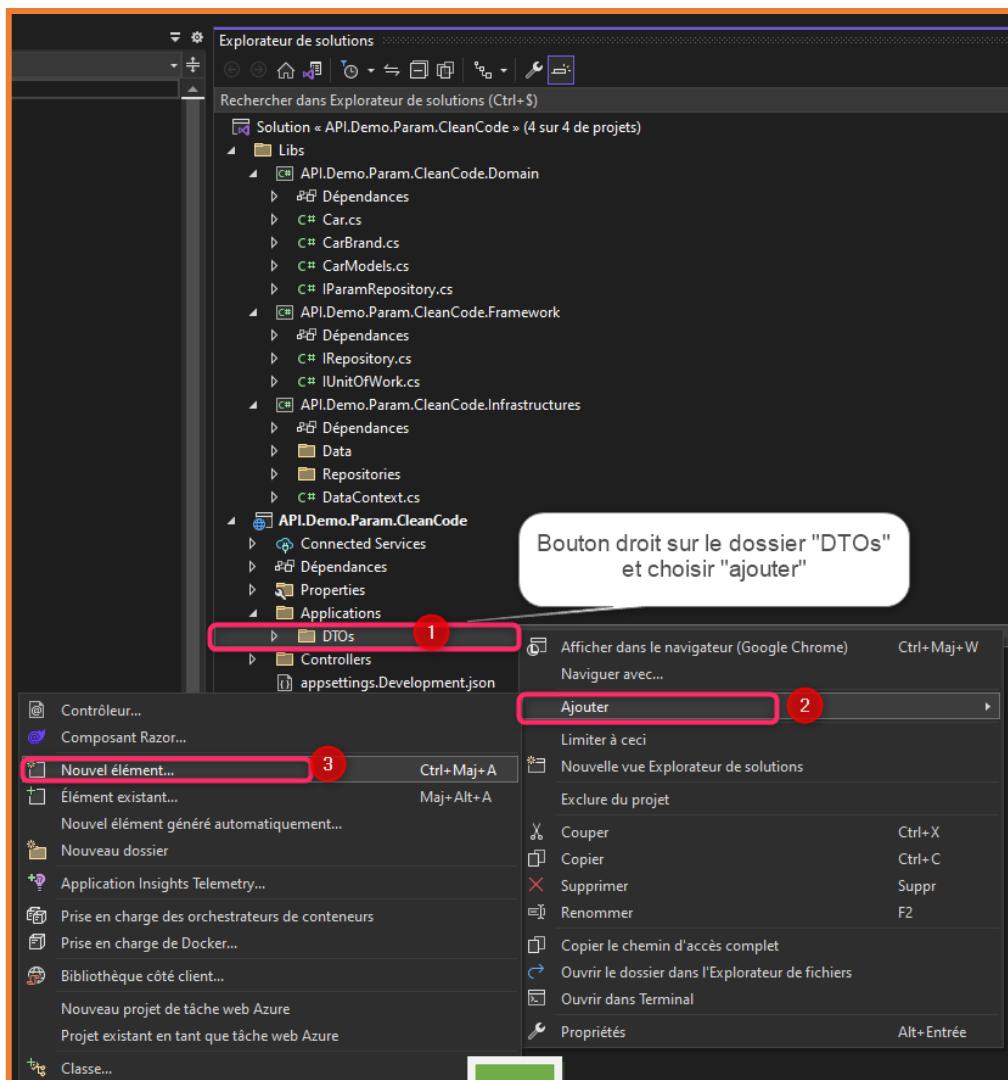


JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 42 / 59

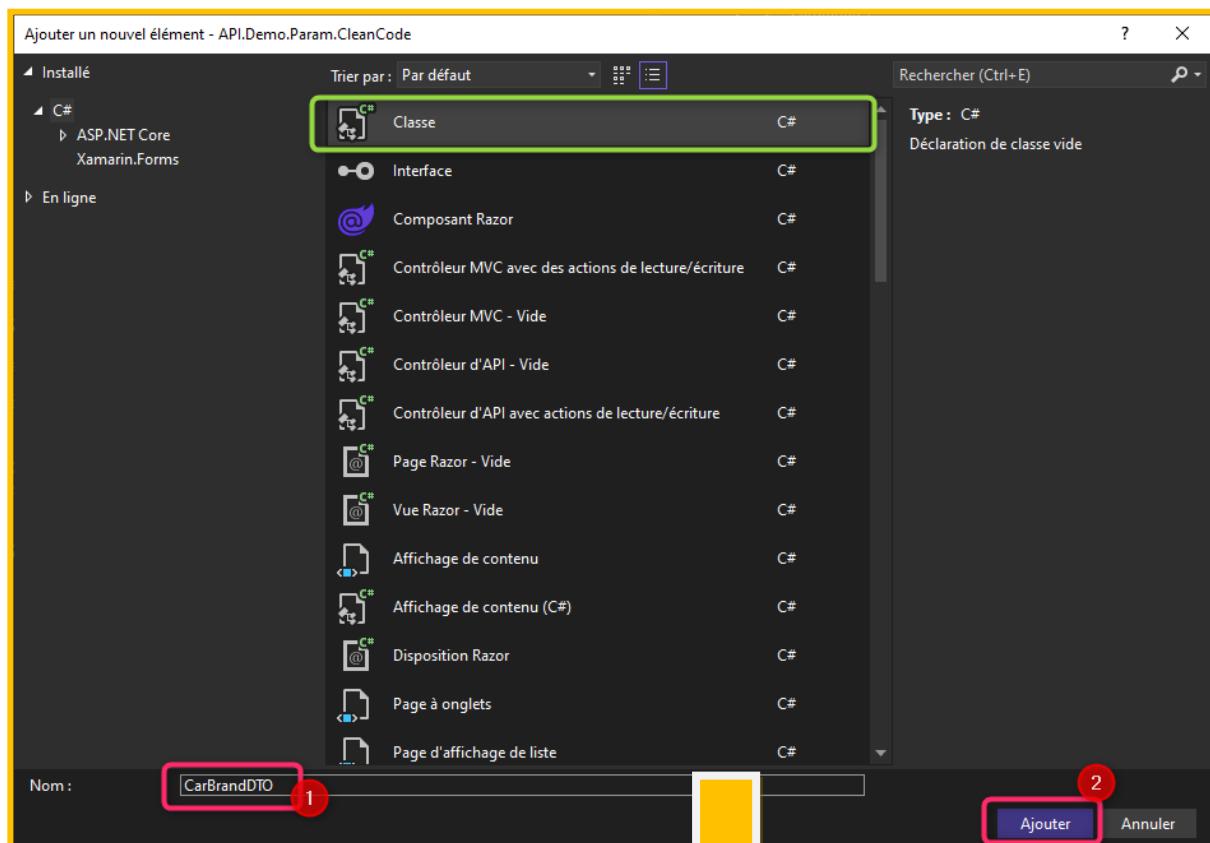


XIV-B Créez le DTO « CarBrandDTO »

- On crée le **DTO** pour la marque voiture.
- On ajoute un string pour renvoyer un **éventuel message d'erreur**.



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 43 / 59



```

namespace API.Demo.Param.CleanCode.Applications.DTOs
{
    /// <summary>
    /// Dto Marque voiture
    /// </summary>
    public class CarBrandDTO
    {
        #region properties
        // > ID généré par code <
        public string ID_CarBrand { get; set; }

        // > ID Généré par la base <
        public int ID_auto { get; set; }

        // > Libellé marque voiture <
        public string Libelle { get; set; }

        // > Message erreur pour l'appelant <
        public string ErrorMsg { get; set; }

        #endregion
    }
}

```

Explorateur de solutions

Solution « API.Demo.Param.CleanCode » (4 sur 4 de p)

- Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - C# Car.cs
 - C# CarBrand.cs
 - C# CarModels.cs
 - C# IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - C# IRepository.cs
 - C# IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructures
 - Dépendances
 - Data
 - Repositories
 - C# DefaultParamRepository.cs
 - C# DataContext.cs
- API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - DTOs
 - C# CarBrandDTO.cs
 - C# CarBrandResumeDTO.cs
 - C# CarDTO.cs
 - C# CarModelDTO.cs
 - Controllers
 - C# ApiController.cs
 - appsettings.Development.json
 - appsettings.json

- Program.cs

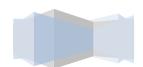
JC CERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 44 / 59

XIV-C Crée le DTO « CarBrandResumeDTO »

- On crée le **DTO** pour la marque voiture.
- Ce DTO comprend le nombre de modèles pour la marque.
- On ajoute un string pour renvoyer un **éventuel message d'erreur**.

The screenshot shows the Visual Studio IDE with the following details:

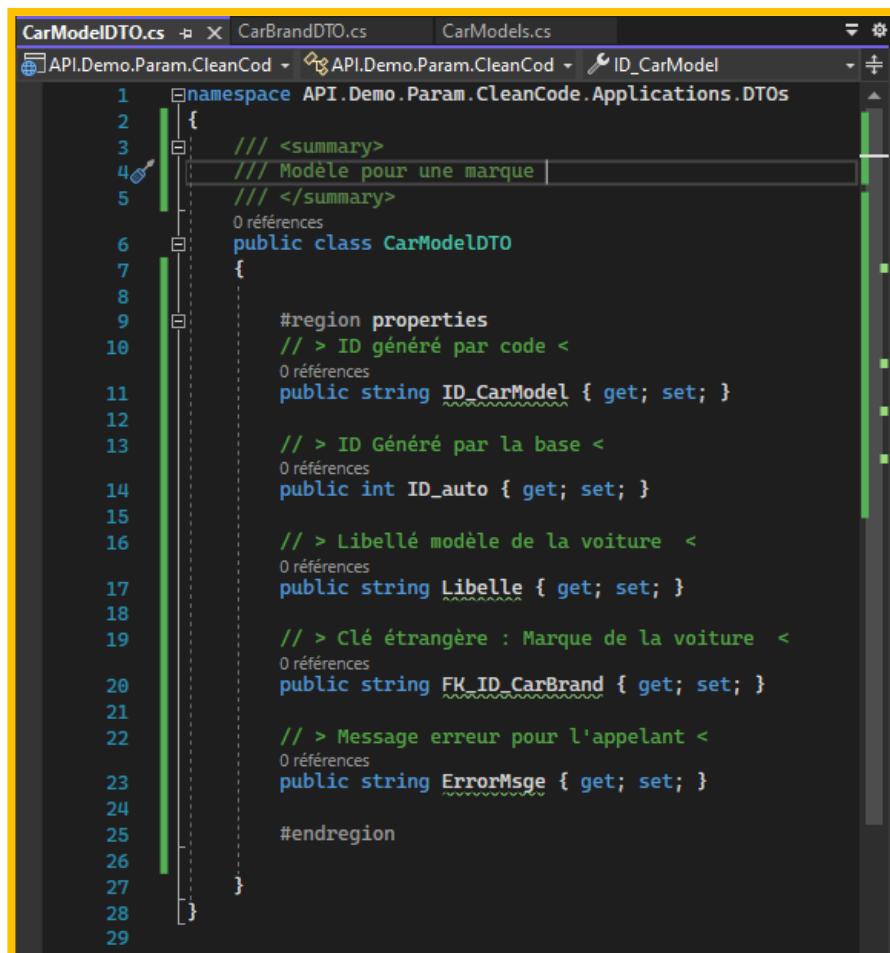
- Code Editor:** The file `CarBrandResumeDTO.cs` is open, displaying C# code for a DTO class. A callout bubble highlights the line `public int NbModels { get; set; }` with the text: "Dans ce modèle de données, on renvoie le nombre de modèles pour la classe."
- Solution Explorer:** Shows the project structure for `API.Demo.Param.CleanCode`, which includes four main components: `Libs`, `API.Demo.Param.CleanCode.Domain`, `API.Demo.Param.CleanCode.Framework`, and `API.Demo.Param.CleanCode.Infrastructure`. The `API.Demo.Param.CleanCode` component contains `Connected Services`, `Dépendances`, `Properties`, `Applications` (which contains `DTOs`), and `Controllers`.
- Task List:** Shows the following items:
 - API.Demo.Param.CleanCo -> API.Demo.Param.CleanCo -> NbModels
 - CarBrandResumeDTO.cs*
 - CarDTO.cs
 - CarBrandDTO.cs*
 - CarModelDTO.cs
 - ApiController.cs
 - appsettings.Development.json
 - appsettings.json
 - Program.cs



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 45 / 59

XIV – D Créer le DTO « CarModelDTO »

- On crée le **DTO** pour le modèle de la voiture



```

CarModelDTO.cs  X  CarBrandDTO.cs  CarModels.cs
API.Demo.Param.CleanCod  API.Demo.Param.CleanCod  ID_CarModel
namespace API.Demo.Param.CleanCode.Applications.DTOs
{
    /// <summary>
    /// Modèle pour une marque
    /// </summary>
    public class CarModelDTO
    {
        #region properties
        // > ID généré par code <
        public string ID_CarModel { get; set; }

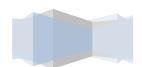
        // > ID Généré par la base <
        public int ID_auto { get; set; }

        // > Libellé modèle de la voiture <
        public string Libelle { get; set; }

        // > Clé étrangère : Marque de la voiture <
        public string FK_ID_CarBrand { get; set; }

        // > Message erreur pour l'appelant <
        public string ErrorMsge { get; set; }
        #endregion
    }
}

```



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 46 / 59

XIV – E Crée le DTO « CarDTO »

- Le **DTO** pour la voiture

The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `CarDTO.cs` file with the following content:

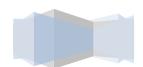
```

1  namespace API.Demo.Param.CleanCode.Applications.DTOs
2  {
3      public class CarDTO
4      {
5          #region properties
6          // > ID généré par code <
7          public string ID_Car { get; set; }
8
9          // > ID Généré par la base <
10         public int ID_auto { get; set; }
11
12         // > Clé étrangère : Marque de la voiture <
13         public string FK_ID_CarBrand { get; set; }
14
15         // > Clé étrangère : Model de la voiture <
16         public string FK_ID_CarModel { get; set; }
17
18         // > Message erreur pour l'appelant <
19         public string ErrorMsge { get; set; }
20     #endregion
21 }
22
23
24

```

The Solution Explorer on the right shows the project structure:

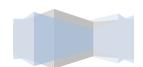
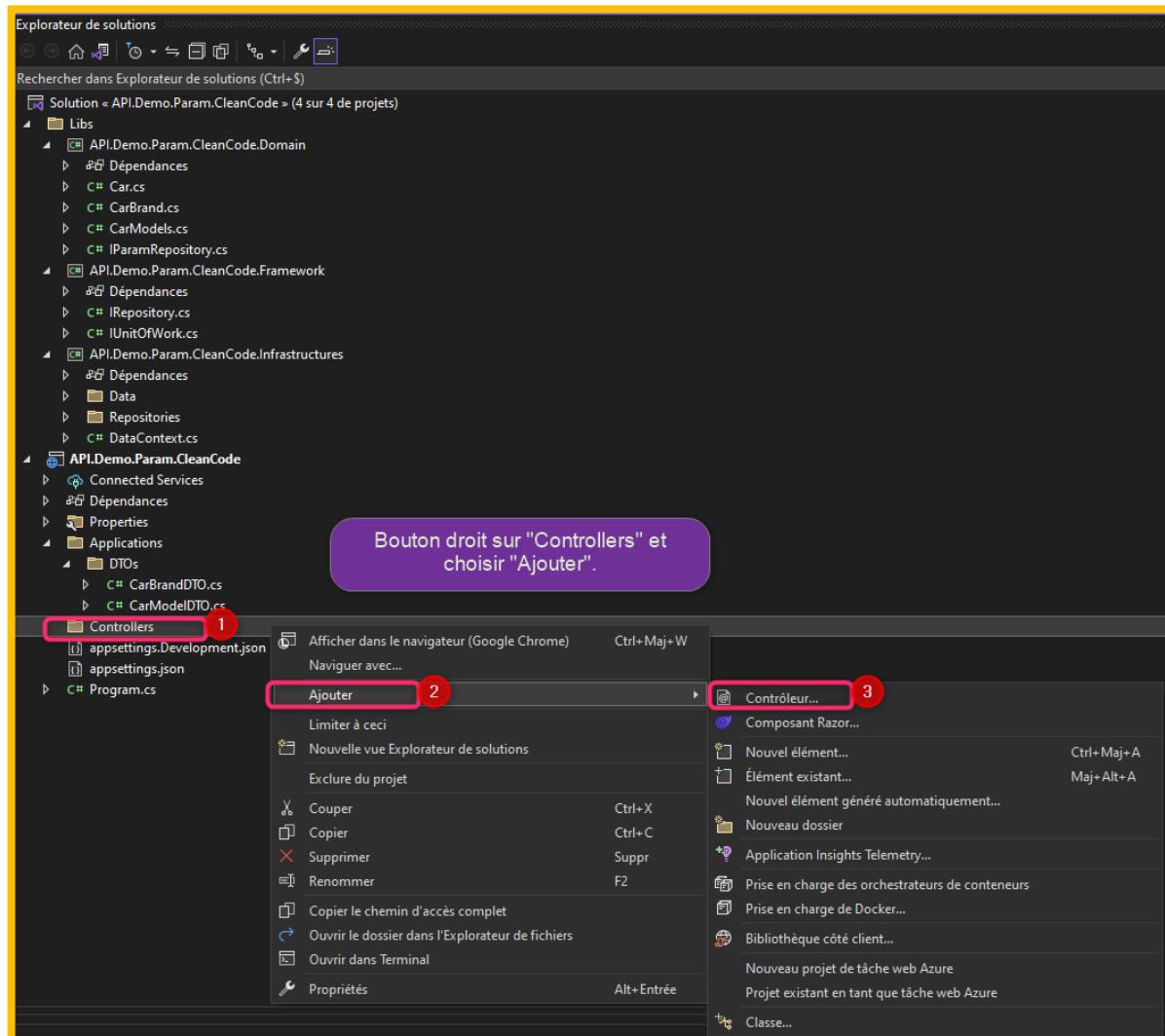
- Libs**:
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - Car.cs
 - CarBrand.cs
 - CarModels.cs
 - IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructures
 - Dépendances
 - Data
 - Repositories
 - DefaultParamRepository.cs
 - DataContext.cs
- API.Demo.Param.CleanCode**
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - DTOs
 - CarBrandDTO.cs
 - CarDTO.cs** (highlighted)
 - CarModelDTO.cs
 - Controllers
 - ApiController.cs
 - appsettings.Development.json
 - appsettings.json
 - Program.cs



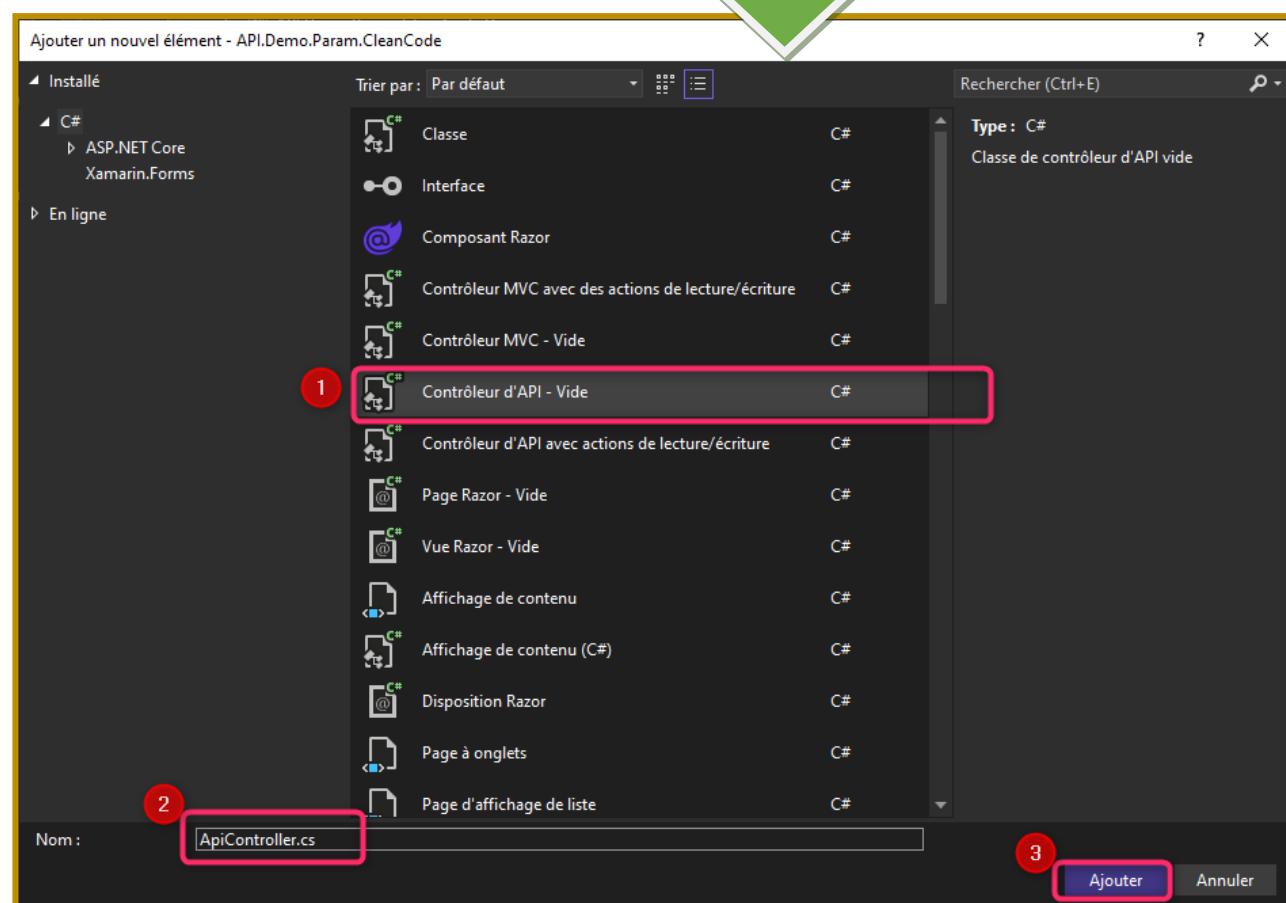
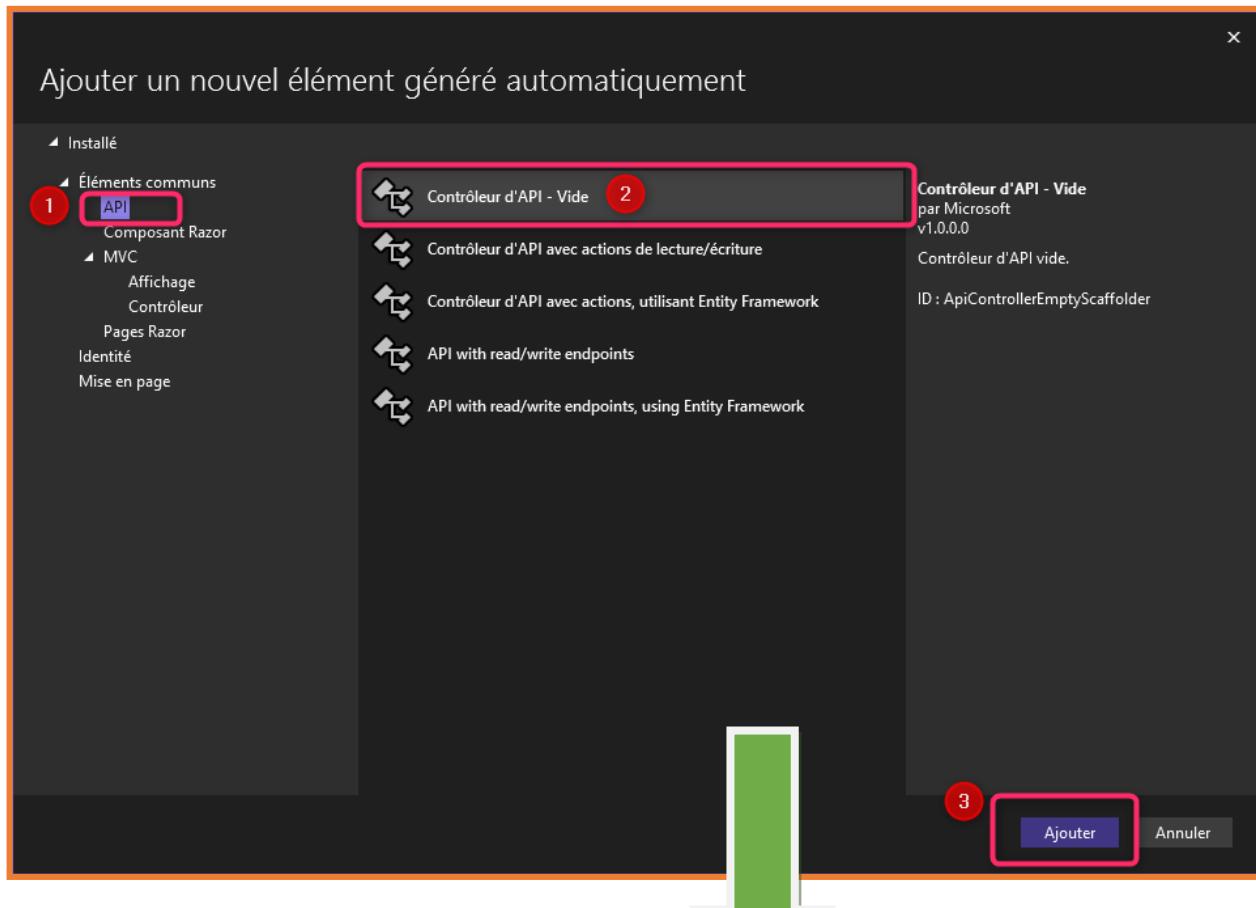
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 47 / 59

XV Création des méthodes dans le Contrôleur API

XV-A - Crédit de la création du contrôleur



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 48 / 59



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 49 / 59

The screenshot shows the Visual Studio interface. On the left, the code editor displays the `ApiController.cs` file:

```

1  using Microsoft.AspNetCore.Http;
2  using Microsoft.AspNetCore.Mvc;
3
4  namespace API.Demo.Param.CleanCode.Controllers
5  {
6      [Route("api/[controller]")]
7      [ApiController]
8      public class ApiController : ControllerBase
9      {
10
11  }
12

```

A pink box highlights the first few lines of the code. On the right, the Solution Explorer shows the project structure for `API.Demo.Param.CleanCode`:

- Libs
 - API.Demo.Param.CleanCode.Domain
 - API.Demo.Param.CleanCode.Framework
 - API.Demo.Param.CleanCode.Infrastructure
- API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - DTOs
 - CarBrandDTO.cs
 - CarModelDTO.cs
 - Controllers
 - ApiController.cs
 - appsettings.Development.json
 - appsettings.json
 - Program.cs

XV-B – Ajout du constructeur

The screenshot shows the Visual Studio interface. The code editor displays the `ApiController.cs` file with the following code:

```

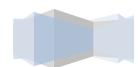
1  using Microsoft.AspNetCore.Http;
2  using Microsoft.AspNetCore.Mvc;
3
4  // > Pour "IParamRepository"
5  using API.Demo.Param.CleanCode.Domain;
6
7  namespace API.Demo.Param.CleanCode.Controllers
8  {
9      [Route("api/[controller]")]
10     [ApiController]
11     public class ApiController : ControllerBase
12     {
13         #region fields
14         // > Pour récupérer l'injection de dépendance <
15         // ( Voir la classe "Program.cs" )
16         private readonly IParamRepository _repository = null ;
17
18         // > On déclare un membre pour récupérer l'environnement d'exécution <
19         // ( ici on en aura pas besoin, mais c'est pour l'exemple )
20         private readonly IWebHostEnvironment _webHostEnvironment=null ;
21
22         #endregion
23
24         // > Constructeur <
25         // ( Installation injection dépendance &...
26         // ...récupération environnement exécution)
27         0 références
28         public ApiController(IParamRepository repository, IWebHostEnvironment webHost)
29         {
30             // > Récupération du repository <
31             _repository = repository;
32
33             // > Récupération environnement exécution <
34             _webHostEnvironment = webHost;
35
36         }
37
38     }
39

```

A purple box highlights the constructor definition. A callout bubble points to it with the text "Constructeur du contrôleur 'Api'".

The Solution Explorer on the right shows the project structure for `API.Demo.Param.CleanCode`:

- Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - Car.cs
 - CarBrand.cs
 - CarModels.cs
 - IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructure- API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - DTOs
 - CarBrandDTO.cs
 - CarModelDTO.cs
 - Controllers
 - ApiController.cs
 - appsettings.Development.json
 - appsettings.json
 - Program.cs



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 50 / 59

XV-C – La méthode « AddCarBrand » (ajout d'une marque de voiture)

- Méthode qui se charge d'ajouter une marque de voiture

```

41  #region Methods
42  /// <summary>
43  /// Add CarBrand
44  /// </summary>
45  /// <param name="BrandDto"></param>
46  /// <returns></returns>
47  [Route("PostCarBrand")]
48  [HttpPost]
49  public IActionResult AddCarBrand(CarBrandDTO BrandDto)
50  {
51      // > Par défaut, on est pessimiste <
52      IActionResult result = this.BadRequest();
53
54      // > (1B) ==> Crédit ID
55      Guid IDGuid = Guid.NewGuid();
56
57      // > On appelle la méthode "AddOneCarBrand" < du repository <
58      // ( On passe à la méthode "AddOneCarBrand"...
59      // ...un paramètre de type "CarBrand" ). .
60      CarBrand brand = this._repository.AddOneCarBrand(new CarBrand())
61      {
62          // > Libellé reçu dans le DTO <
63          Libelle = BrandDto.Libelle,
64
65          // > Génération de l'ID par programme <
66          ID_CarBrand = IDGuid.ToString()
67      };
68
69      // > On effectue l'ajout en base de données <
70      this._repository.UnitOfWork.SaveChanges();
71
72      // > On vérifie que l'ajout s'est bien passé <
73      // ( La méthode "AddOneCarBrand" renvoie l'objet inséré par l'ORM )
74      if (brand != null)
75      {
76          // > On récupère dans le DTO l'ID auto calculé par l'ORM <
77          BrandDto.ID_auto = brand.ID_auto;
78          // > On récupère aussi l'ID calculé par code <
79          BrandDto.ID_CarBrand = brand.ID_CarBrand;
80
81          // > renvoie un objet DTO <
82          result = this.Ok(BrandDto);
83
84      }
85
86      // > On renvoie le résultat <
87      return result;
88  }
89  #endregion

```

The screenshot shows the Visual Studio IDE with the code editor open to the ApiController.cs file. A callout bubble highlights the method signature `AddCarBrand(CarBrandDTO BrandDto)` with the text "Ajout de la méthode 'AddCarBrand' (ajout d'une marque)". The code is annotated with numerous comments explaining the logic and flow of the method. To the right, the Solution Explorer shows the project structure with several projects like API.Demo.Param.CleanCode.Domain, API.Demo.Param.CleanCode.Framework, and API.Demo.Param.CleanCode.Infrastructures.



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 51 / 59

XV-D – La méthode « UpdCarBrand » (Mettre à jour une marque de voiture)

The screenshot shows the Visual Studio IDE with the following details:

- Code Editor:** The main window displays the `DefaultParamRepository.cs*` file, specifically the `UpdCarBrand` method. A tooltip above the code reads: "La méthode "UpdCarBrand" (Mâj une marque de voiture)."
- Solution Explorer:** On the right, the Solution Explorer shows the project structure for `API.Demo.Param.CleanCode`, which includes four projects: `API.Demo.Param.CleanCode.Domain`, `API.Demo.Param.CleanCode.Framework`, `API.Demo.Param.CleanCode.Infrastructure`, and `API.Demo.Param.CleanCode`. The `API.Demo.Param.CleanCode` project contains several files like `ApiController.cs`, `appsettings.Development.json`, and `Program.cs`.
- Toolbars and Status Bar:** Standard Visual Studio toolbars are visible at the top, and the status bar at the bottom shows the path `API.Demo.Param.CleanCode>API.Demo.Param.CleanCode>Controllers>ApiController.cs`.



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 52 / 59

XV-E – La méthode « GetCarBrand » (Récupérer marque(s) de voiture)

- Méthode qui se charge de récupérer une marque OU une liste de marques de voiture.

```

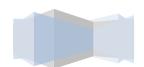
125  /// <summary>
126  /// Renvoyer liste marques de voiture
127  /// </summary>
128  /// <param name="ID_CarBrand"></param>
129  /// <returns></returns>
130  [Route("GetCarBrand")]
131  [HttpGet]
132  0 références
133  public IActionResult GetCarBrand([FromQuery] string ID_CarBrand)
134  {
135      // > Lance l'extraction de la liste <
136      // Si un ID est transmis, on ne renvoie que l'enregistrement...
137      // ...correspondant.
138
139      // Remarque : la clause "Where" sur le paramètre...
140      // ... "ID_CarBrand" se fait dans la méthode "GetAllCarBrand" du...
141      // ... _repository.
142      var BrandList = this._repository.GetAllCarBrand(ID_CarBrand);
143
144      // > On construit les colonnes de la liste d'objets que l'on renvoie <
145      var elements = BrandList.Select(item => new CarBrandDTO()
146      {
147          ID_auto = item.ID_auto,
148          ID_CarBrand = item.ID_CarBrand,
149          Libelle=item.Libelle,
150      }).ToList();
151
152      return this.Ok(elements);
153  }
154
155  #endregion
156
157  }
158
159

```

Ajout de la méthode qui renvoie une liste des marques de voitures.

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for the `GetCarBrand` method in `ApiController.cs`. The right pane shows the Solution Explorer with the project structure:

- Solution « API.Demo.Param.CleanCode » (4 sur 4 de projets)**
 - API.Demo.Param.CleanCode.Domain**
 - Dépendances
 - Car.cs
 - CarBrand.cs
 - CarModels.cs
 - IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework**
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructures**
 - Dépendances
 - Data
 - Repositories
 - DefaultParamRepository.cs
 - DataContext.cs
 - API.Demo.Param.CleanCode**
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - DTOs
 - CarBrandDTO.cs
 - CarBrandResumeDTO.cs
 - CarDTO.cs
 - CarModelDTO.cs
 - Controllers
 - ApiController.cs
 - GetCarBrand(string) : IActionResult
- appsettings.Development.json**
- appsettings.json**
- Program.cs**



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 53 / 59

XVI Erreurs Exécution (au premier lancement)

XVI – A Erreur à la création de la base de données

The screenshot shows the Visual Studio IDE with the following details:

- File Menu:** Fichier, Edition, Affichage, Git, Projet, Général, Déboguer, Test, Analyser, Outils, Extensions, Fenêtre, Aide.
- Toolbar:** Standard toolbar with icons for Open, Save, Print, etc.
- Status Bar:** Processus : [16060] API.Demo.Param.CleanCo, Événements du cycle de vie, Thread : [29272] Thread principal, Frame de pile : Program.<Main>\$.
- Code Editor:** Shows a C# file named Program.cs with the following code snippet:


```

46     builder.Services.AddEndpointsApiExplorer();
47     builder.Services.AddSwaggerGen();
48
49     var app = builder.Build();
50
51     // Configure the HTTP request pipeline.
52     if (app.Environment.IsDevelopment())
53     {
54         app.UseSwagger();
55         app.UseSwaggerUI();
56     }
57
58     // > On se prépare à créer la BDD <
59     using (var scope = app.Services.CreateScope())
60     {
61         // -- Création des Tables si elles sont inexistantes --
62         var dbContext = scope.ServiceProvider.GetRequiredService<DataContext>();
63
64         // > Création des tables --
65         dbContext.Database.EnsureCreated(); ✖
66     }
67
68     app.UseHttpsRedirection();
69
70     app.UseAuthorization();
71
72     app.MapControllers();
73
74     app.Run();
    
```
- Toolbox:** Standard toolbox with icons for Windows Forms, Web Forms, etc.
- Output Window:** Shows "Processus : [16060] API.Demo.Param.CleanCo".
- Task List:** Shows "Événements du cycle de vie".
- Diagnostic Tools:** Shows memory usage and CPU usage over time.
- Exception Dialog:** An error dialog is open at the bottom left, displaying:

Exception non gérée

Microsoft.Data.Sqlite.SqliteException : 'SQLite Error 1: near """: syntax error.'

Cette exception a été levée à l'origine dans cette pile des appels :

[Code externe]
Program.<Main>\$(string[]) dans Program.cs

Afficher les détails | Copier les détails | Démarrer la session Live Share...

Paramètres d'exception
 Arrêter quand ce type d'exception est levé
Sauf si elle est levée à partir de :
API.Demo.Param.CleanCode.dll

Ouvrir les paramètres d'exception | Modifier les conditions

The screenshot shows the Visual Studio IDE with the following details:

- File Menu:** Fichier, Edition, Affichage, Git, Projet, Général, Déboguer, Test, Analyser, Outils, Extensions, Fenêtre, Aide.
- Toolbar:** Standard toolbar with icons for Open, Save, Print, etc.
- Status Bar:** Processus : [16060] API.Demo.Param.CleanCo, Événements du cycle de vie, Thread : [29272] Thread principal, Frame de pile : Program.<Main>\$.
- Code Editor:** Shows a C# file named Program.cs with the same code snippet as the previous screenshot.
- Toolbox:** Standard toolbox with icons for Windows Forms, Web Forms, etc.
- Output Window:** Shows "Processus : [16060] API.Demo.Param.CleanCo".
- Task List:** Shows "Événements du cycle de vie".
- Diagnostic Tools:** Shows memory usage and CPU usage over time.
- Exception Dialog:** An error dialog is open at the bottom left, displaying:

Exception non gérée

Microsoft.Data.Sqlite.SqliteException : 'SQLite Error 1: near """syntax error.'

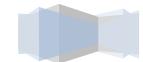
Cette exception a été levée à l'origine dans cette pile des appels :

[Code externe]
Program.<Main>\$(string[]) dans Program.cs

Afficher les détails | Copier les détails | Démarrer la session Live Share...

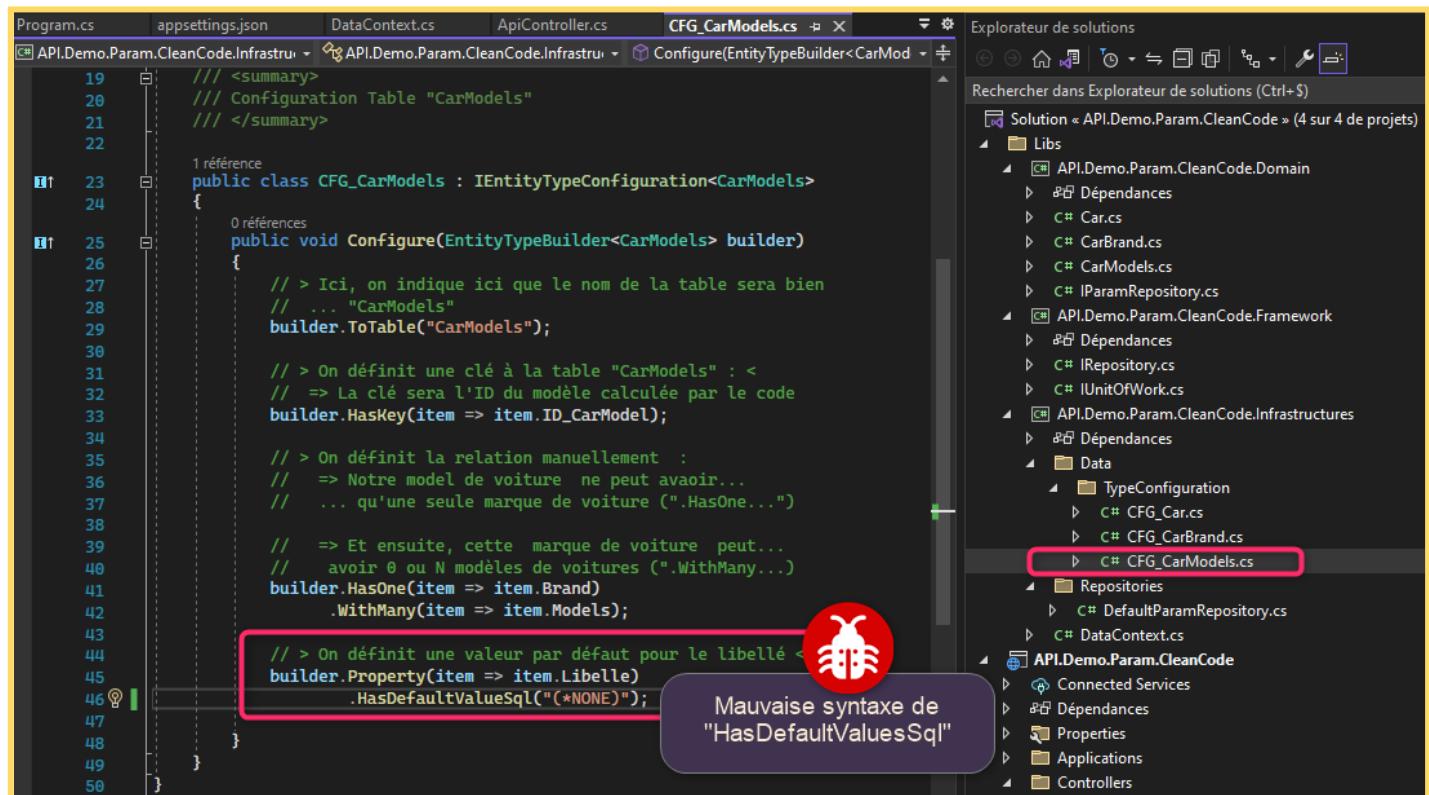
Paramètres d'exception
 Arrêter quand ce type d'exception est levé
Sauf si elle est levée à partir de :
API.Demo.Param.CleanCode.dll

Ouvrir les paramètres d'exception | Modifier les conditions



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 54 / 59

XVI – A-1 Mauvaise syntaxe de « HasDefaultValueSql » dans la classe de configuration



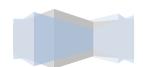
The screenshot shows a Visual Studio interface. The code editor displays a C# class 'CFG_CarModels' within a 'Configure' method of a 'EntityTypeBuilder<CarModels>' object. A tooltip is overlaid on the code, pointing to a line that uses the 'HasDefaultValueSql' method. The tooltip contains the text 'Mauvaise syntaxe de "HasDefaultValueSql"' and features a red bee icon.

```

19     /// <summary>
20     /// Configuration Table "CarModels"
21     /// </summary>
22
23     public class CFG_CarModels : IEntityTypeConfiguration<CarModels>
24     {
25         public void Configure(EntityTypeBuilder<CarModels> builder)
26         {
27             // > Ici, on indique ici que le nom de la table sera bien
28             // ... "CarModels"
29             builder.ToTable("CarModels");
30
31             // > On définit une clé à la table "CarModels" : <
32             // => La clé sera l'ID du modèle calculée par le code
33             builder.HasKey(item => item.ID_CarModel);
34
35             // > On définit la relation manuellement :
36             // => Notre model de voiture ne peut avoir...
37             // ... qu'une seule marque de voiture ("WithOne...")
38
39             // => Et ensuite, cette marque de voiture peut...
40             // avoir 0 ou N modèles de voitures ("WithMany...")
41             builder.HasOne(item => item.Brand)
42                 .WithMany(item => item.Models);
43
44             // > On définit une valeur par défaut pour le libellé <
45             builder.Property(item => item.Libelle)
46                 .HasDefaultValueSql("(*NONE)");
        }
    }

```

The Solution Explorer on the right shows the project structure for 'API.Demo.Param.CleanCode' with several files listed under 'Libs' and 'API.Demo.Param.CleanCode'. The file 'CFG_CarModels.cs' is highlighted with a red rectangle.



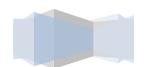
JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 55 / 59

XVI – A-2 Bonne syntaxe de « HasDefaultValuesSql » dans la classe de configuration

The screenshot shows the Visual Studio IDE with the following details:

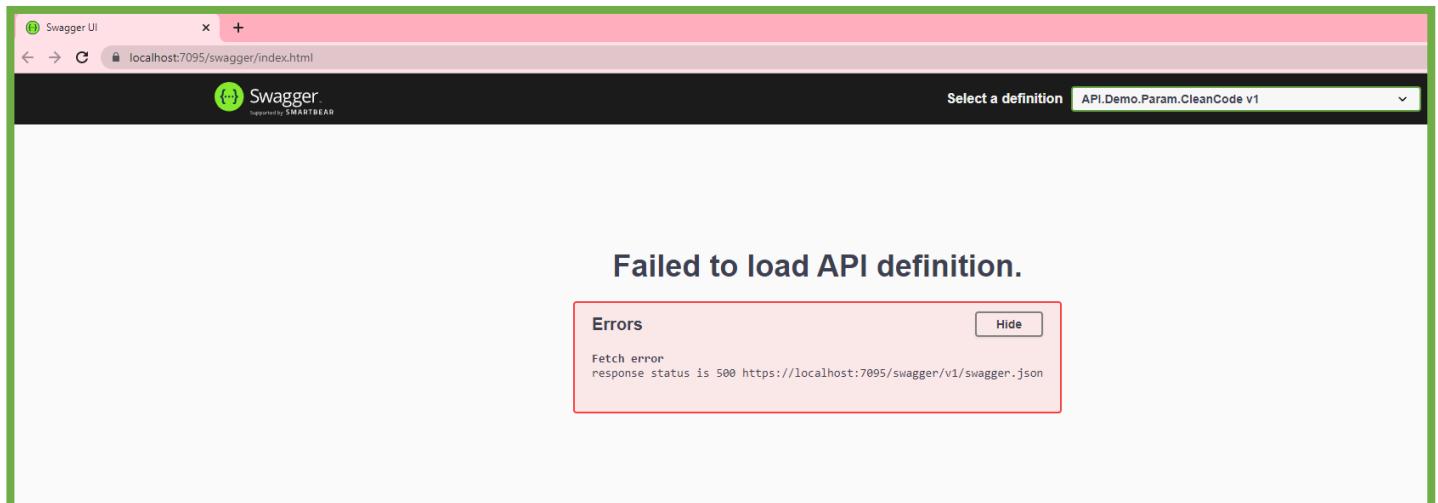
- Solution Explorer:** Shows the project structure under "API.Demo.Param.CleanCode". A file named "CFG_CarModels.cs" is selected and highlighted with a red box.
- Code Editor:** Displays the C# code for the "CFG_CarModels" class. A specific line of code is highlighted with a pink box and a green checkmark icon, indicating it is correct or good practice:


```
// > On définit une valeur par défaut pour le libellé <
builder.Property(item => item.Libelle)
    .HasDefaultValueSql("(*NONE*)");
```
- Text Overlay:** A callout bubble points to the highlighted code with the text: "Bonne syntaxe de 'HasDefaultValueSql' qui ne provoque de plantage à la création de la base."



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 56 / 59

XVI – B Swagger : Impossible de charger la définition



Screenshot of Visual Studio showing the code editor and Solution Explorer.

The code editor shows the `ApiController.cs` file:

```

133     /// <summary>
134     /// Renvoyer liste marques de voiture
135     /// </summary>
136     /// <param name="ID_CarBrand"></param>
137     /// <returns></returns>
138     [Route("GetCarBrand")]
139     [HttpGet]
140     public IActionResult GetListCarBrand([FromQuery] string ID_CarBrand)...
141
142
143     /// <summary>
144     /// Ajouter un Modèle d'une marque de voiture
145     /// </summary>
146     /// <param name="ModelDto"></param>
147     /// <returns></returns>
148     [Route("PostCarModel")]
149     [HttpPost]
150     public IActionResult AddCarModel([FromBody] CarModelDTO ModelDto)...
151
152
153     [Route("UpdateCarBrand")]
154     [HttpPost]
155     public IActionResult UpdCarModel([FromBody] CarModelDTO ModelDto)...
156
157
158     public IActionResult GetCarModels([FromQuery] string ID_CarBrand)...
159     #endregion
  
```

A callout bubble points to the last method with the text: "J'ai oublié d'indiquer la route [Route(MaRoute)]".

A callout bubble points to the last method with the text: "J'ai aussi oublié d'indiquer [HttpPost] ou [HttpGet]".

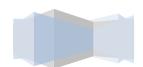
The Solution Explorer shows the project structure:

- Solution « API.Demo.Param.CleanCode » (4 sur 4 de pr)
 - Libs
 - API.Demo.Param.CleanCode.Domain
 - Dépendances
 - Car.cs
 - CarBrand.cs
 - CarModels.cs
 - IParamRepository.cs
 - API.Demo.Param.CleanCode.Framework
 - Dépendances
 - IRepository.cs
 - IUnitOfWork.cs
 - API.Demo.Param.CleanCode.Infrastructures
 - Dépendances
 - Data
 - TypeConfiguration
 - CFG_Car.cs
 - CFG_CarBrands.cs
 - CFG_CarModels.cs
 - Repositories
 - DefaultParamRepository.cs
 - DataContext.cs
 - API.Demo.Param.CleanCode
 - Connected Services
 - Dépendances
 - Properties
 - Applications
 - Controllers
 - ApiController.cs
- appsettings.Development.json
- appsettings.json
- DemoParam.db
- Program.cs



JC CERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 57 / 59

The screenshot shows the Swagger UI interface for a RESTful API. At the top, it displays the title "API.Demo.Param.CleanCode" with a version of "1.0 OAS3". Below the title, the URL "https://localhost:7095/swagger/v1/swagger.json" is shown. The main content area is divided into two sections: "Api" and "Schemas". The "Api" section lists several endpoints with their methods and URLs: POST /Api/PostCarBrand, POST /Api/UpdCarBrand, GET /Api/GetCarBrand, POST /Api/PostCarModel, POST /Api/UpdateCarBrand, and GET /Api/GetCarModel. Each endpoint entry has a dropdown arrow icon at the end. The "Schemas" section contains links to "CarBrandDTO" and "CarModelDTO", each preceded by a right-pointing arrow.



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 58 / 59

XVII Liens Divers

XVII – A dotnet efcore (liens documentation)

Voir lien :

<https://github.com/dotnet/efcore/issues/14635>

```
modelBuilder.Entity<Customers>(entity =>
{
    entity.HasKey(e => e.CustomerNumber)
        .HasName("PK_Customers")
        .ForSqlServerIsClustered(false);

    entity.HasIndex(e => e.Type)
        .HasName("CustomerTypes");

    //entity.HasIndex(e => e.CompanyName)
    //    .HasName("CustomersCompanyName");

    entity.Property(e => e.CustomerNumber)
        .IsUnicode(false)
        .ValueGeneratedNever();

    entity.Property(e => e.Type)
        .IsUnicode(false)
        .HasDefaultValueSql("('Standard')");

    entity.Property(e => e.AddedBy).IsUnicode(false);

    entity.Property(e => e.Address1).IsUnicode(false);

    entity.Property(e => e.Address2).IsUnicode(false);

    entity.Property(e => e.Address3).IsUnicode(false);

    entity.Property(e => e.Address4).IsUnicode(false);

    entity.Property(e => e.CompanyType)
        .IsUnicode(false)
        .HasDefaultValueSql("('Company')");

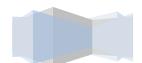
    entity.Property(e => e.CompanyName).IsUnicode(false);

    entity.Property(e => e.Email).IsUnicode(false);

    entity.Property(e => e.contactFirstName).IsUnicode(false);

    entity.Property(e => e.ContactLastName).IsUnicode(false);
```

Exemple d'utilisation



JC CHERID	VISUAL STUDIO C#	TEC_C#	Date
	Liste des thèmes abordés		02/02/2023
	MODULE / PROCESSUS	Visual Studio C# CLEAN-CODE (Synthèse)	Page 59 / 59

XVII – B Github (Dans Visual studio)

<https://learn.microsoft.com/fr-fr/visualstudio/version-control/git-make-commit?view=vs-2022>

