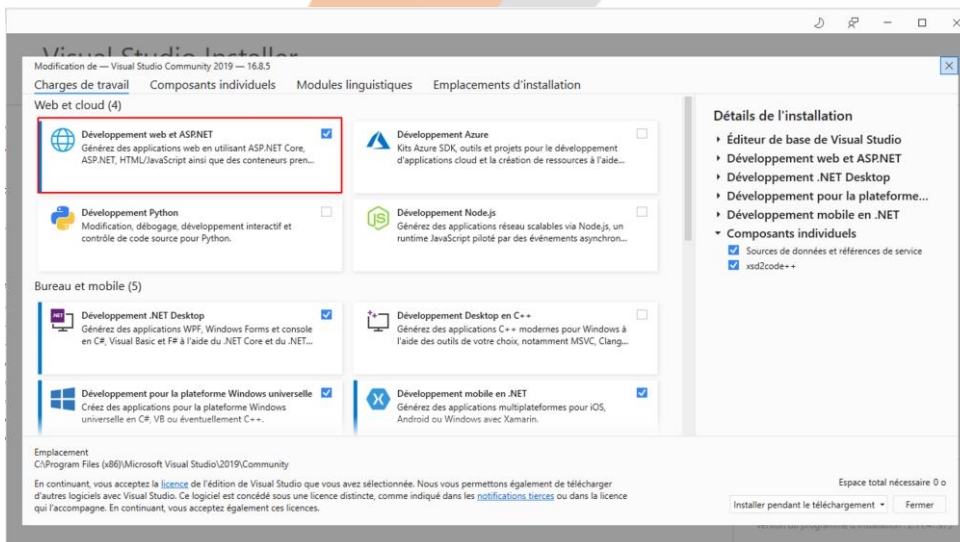


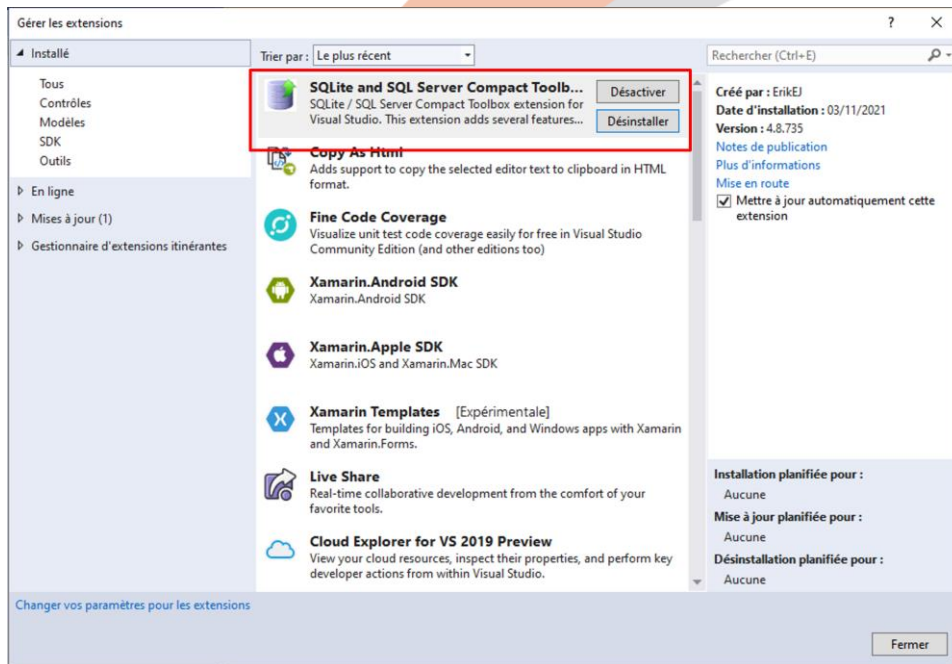
Workshop Backend

Prérequis :

Installation **Développement Web et ASP.NET**



Installation extension **SQLite and SQL Server Compact Toolbox**



SDK **.Net Core 5.0**

<https://dotnet.microsoft.com/download/dotnet/5.0>

I. Architecture N-Tiers backend

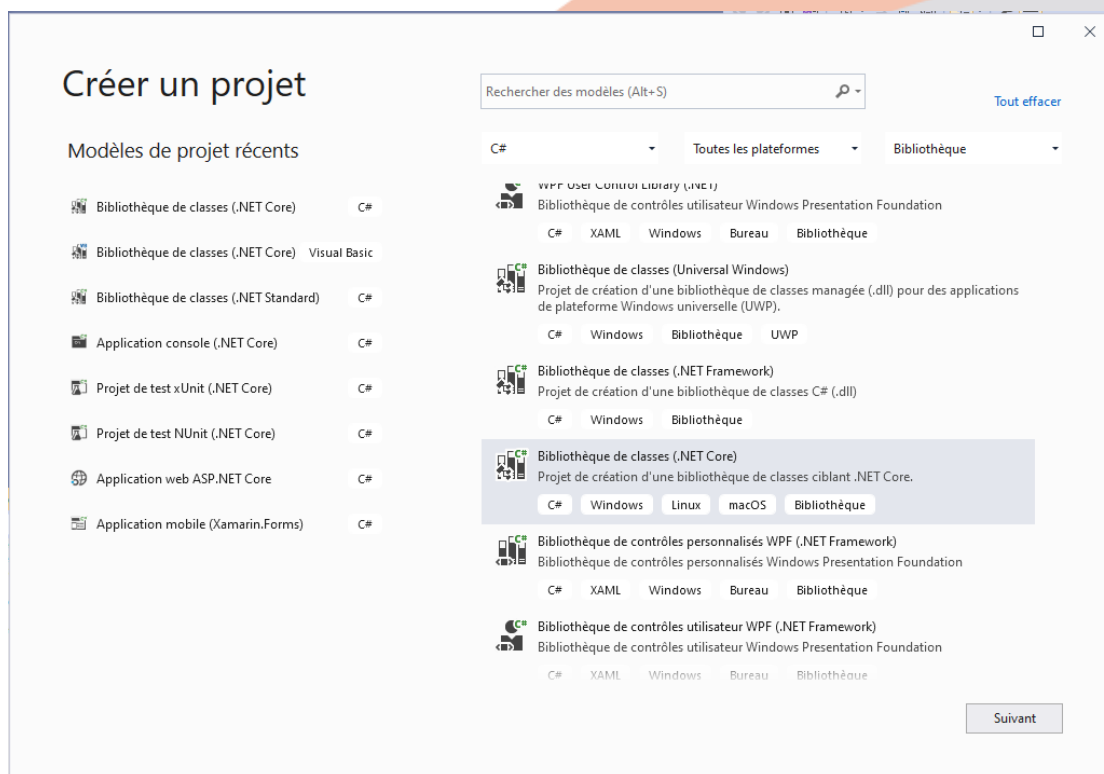
Ouvrir Visual Studio 2019

Voici un aperçu sur la solution à créer :

- **Solution** : iit.crossplateforme.Solution
 - **Bibliothèque de classe** : iit.crossplateforme.Data
 - **Bibliothèque de classe** : iit.crossplateforme.Domain
 - **Web ASP NET Core** : iit.crossplateforme.webApi

Ci-dessous les étapes de création de la solution :

Créer un projet Bibliothèque de classes .NetCore



Nommer le projet **iit.crossplateforme.Data** et la solution **iit.crossplateforme.Solution**

Configurer votre nouveau projet

Bibliothèque de classes (.NET Core) C# Windows Linux macOS Bibliothèque

Nom du projet

Emplacement

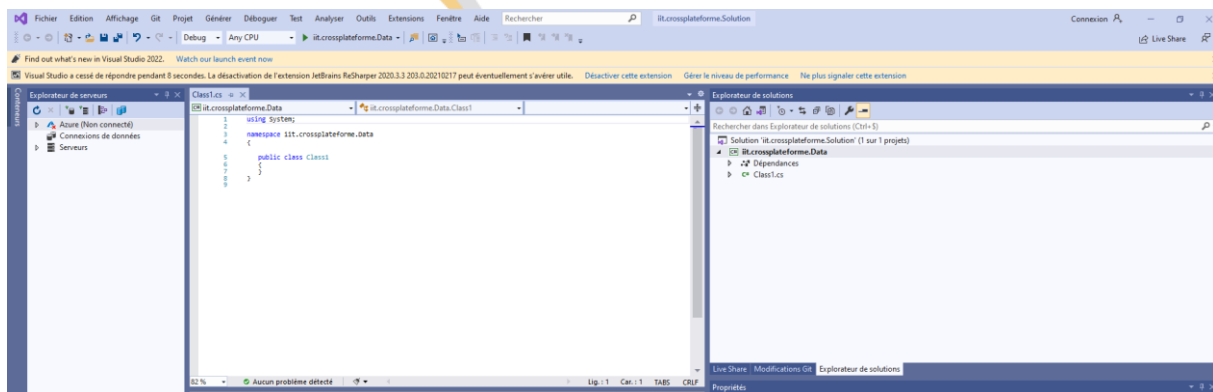
Solution

Nom de la solution ⓘ

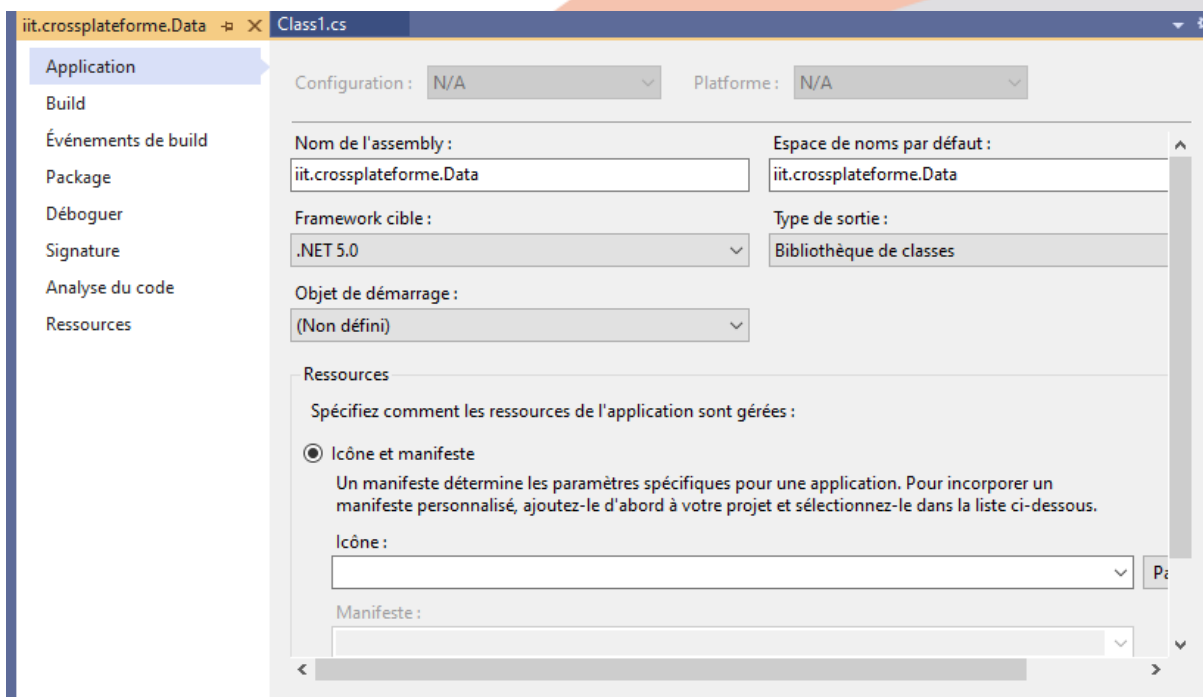
☐ Placer la solution et le projet dans le même répertoire

Retour Créer

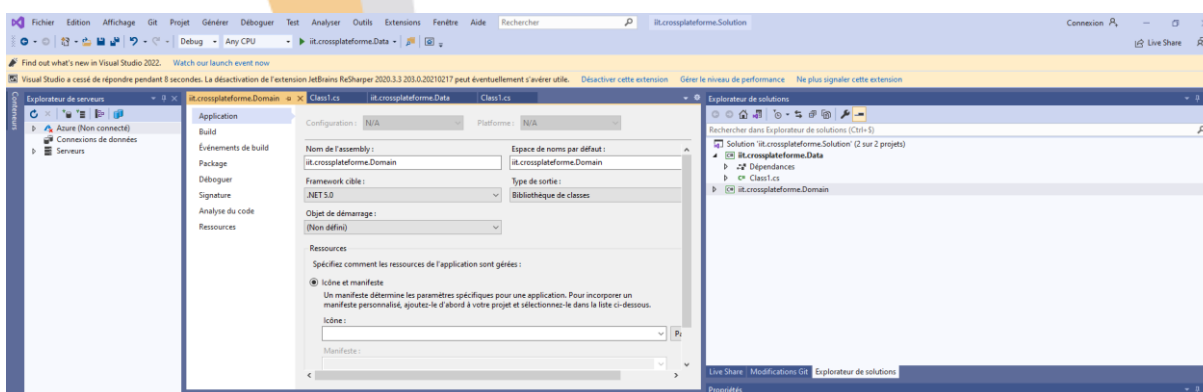
Voici le résultat de cette action :



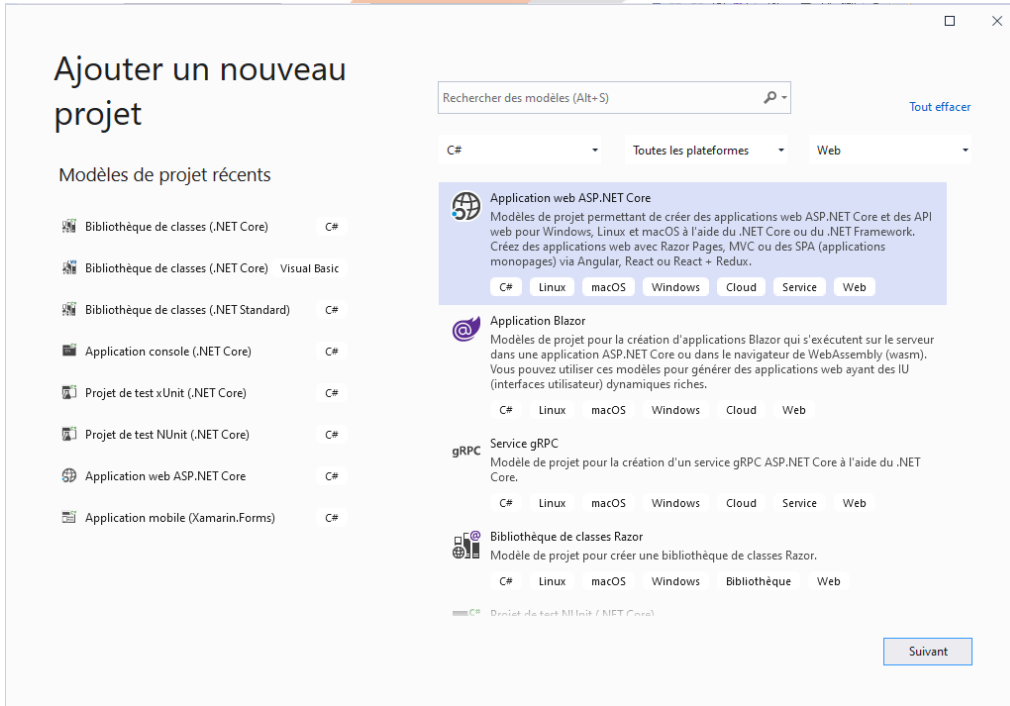
Accéder aux propriétés du projet crée et définir le Framework cible (.Net 5.0)



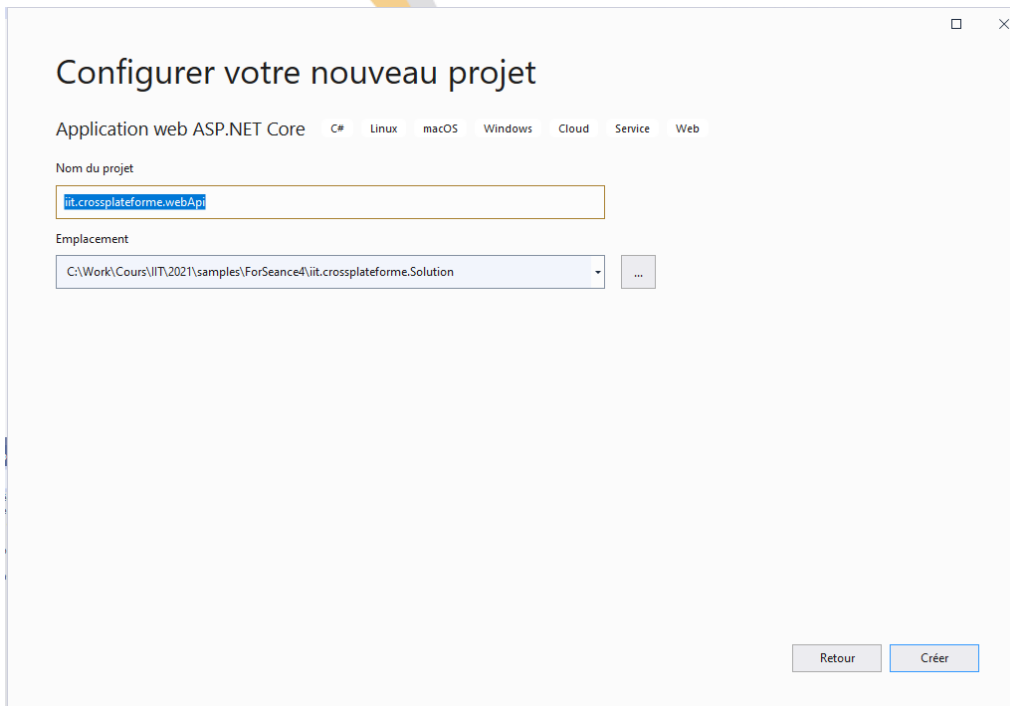
Ajouter la couche domain en tant que projet Bibliothèque de classes sous la solution précédemment créée de la même manière que le projet Data et changer dans les propriétés le Framework cible à .Net5.0.



Ajouter la couche webApi en tant que projet ASP Net Core

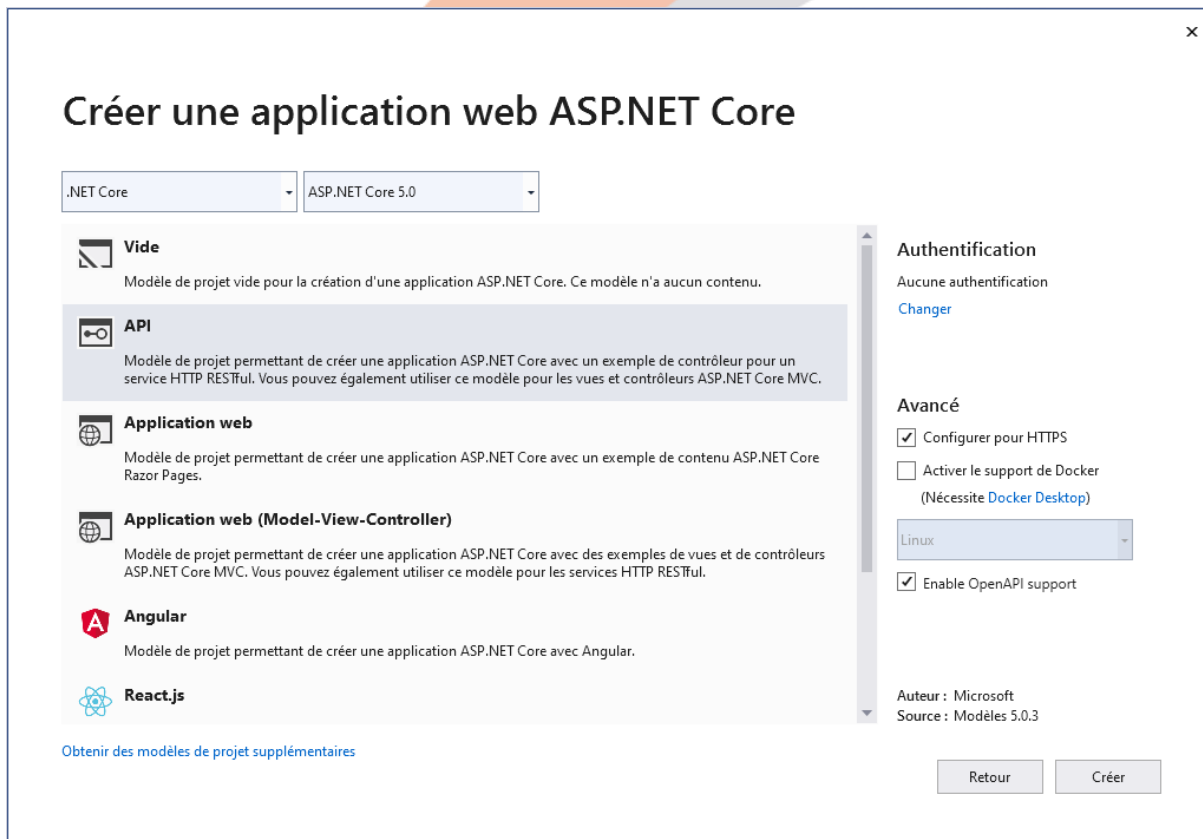


Nommer le projet **iit.crossplateforme.webApi**



Choisir le framework cible **ASP .NET Core 5.0**

Sélectionner l'option API



Créer une application web ASP.NET Core

.NET Core ASP.NET Core 5.0

Vide
Modèle de projet vide pour la création d'une application ASP.NET Core. Ce modèle n'a aucun contenu.

API
Modèle de projet permettant de créer une application ASP.NET Core avec un exemple de contrôleur pour un service HTTP RESTful. Vous pouvez également utiliser ce modèle pour les vues et contrôleurs ASP.NET Core MVC.

Application web
Modèle de projet permettant de créer une application ASP.NET Core avec un exemple de contenu ASP.NET Core Razor Pages.

Application web (Model-View-Controller)
Modèle de projet permettant de créer une application ASP.NET Core avec des exemples de vues et de contrôleurs ASP.NET Core MVC. Vous pouvez également utiliser ce modèle pour les services HTTP RESTful.

Angular
Modèle de projet permettant de créer une application ASP.NET Core avec Angular.

React.js

[Obtenir des modèles de projet supplémentaires](#)

Authentification
Aucune authentification
[Changer](#)

Avancé
☒ Configurer pour HTTPS
☐ Activer le support de Docker (Nécessite [Docker Desktop](#))
 Linux
☒ Enable OpenAPI support

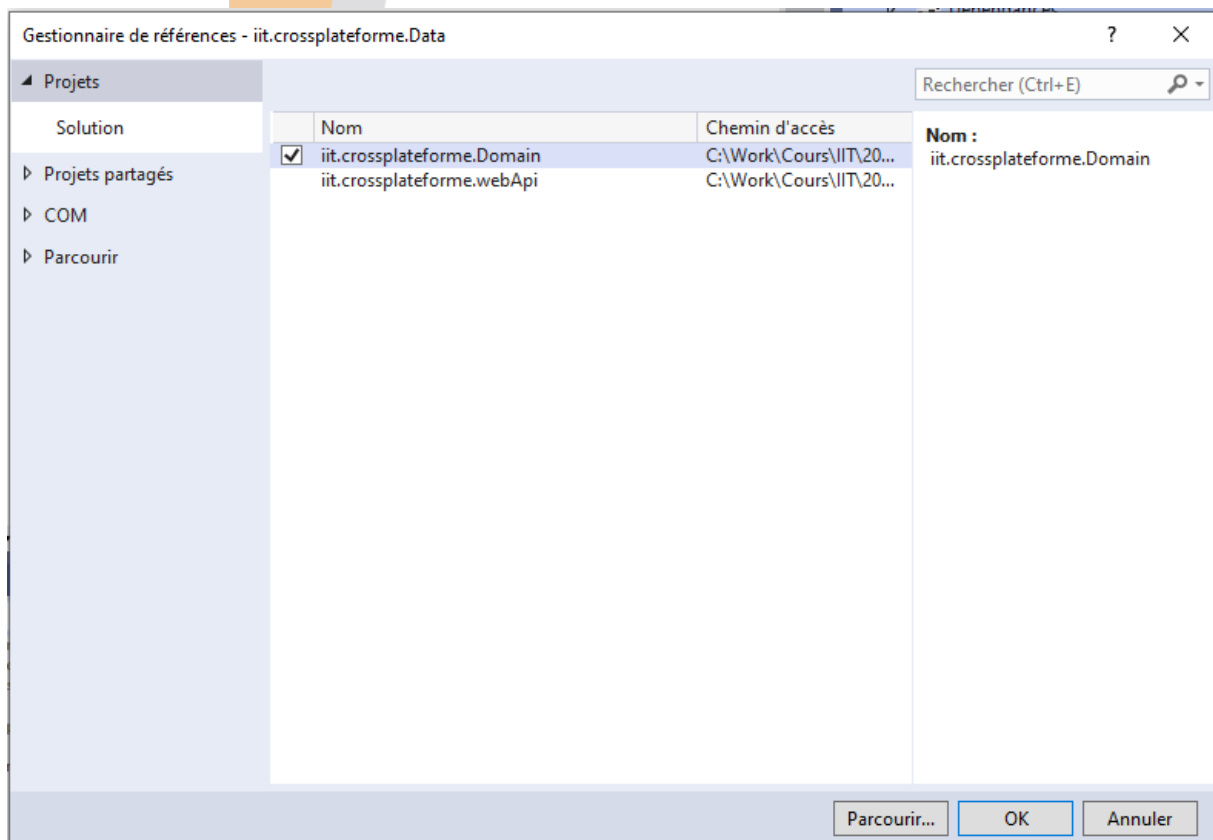
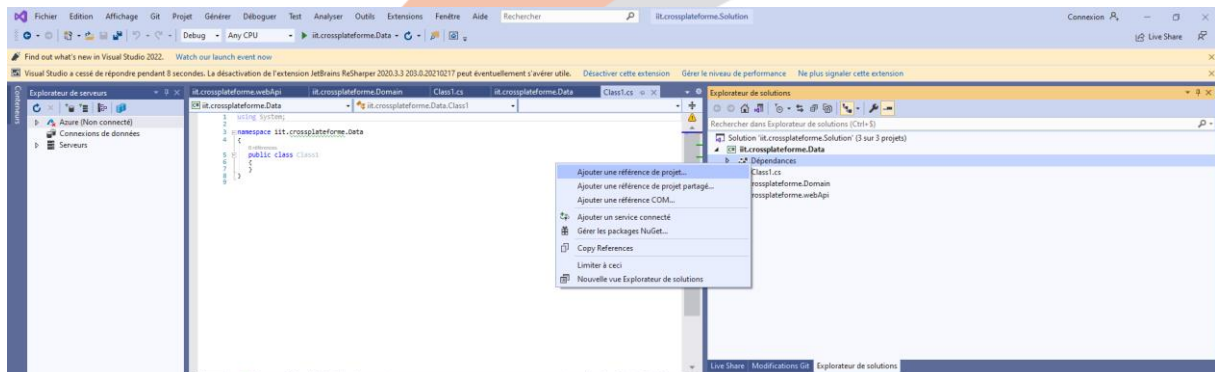
Auteur : Microsoft
Source : Modèles 5.0.3

[Retour](#) [Créer](#)

Ajouter les références entre les couches

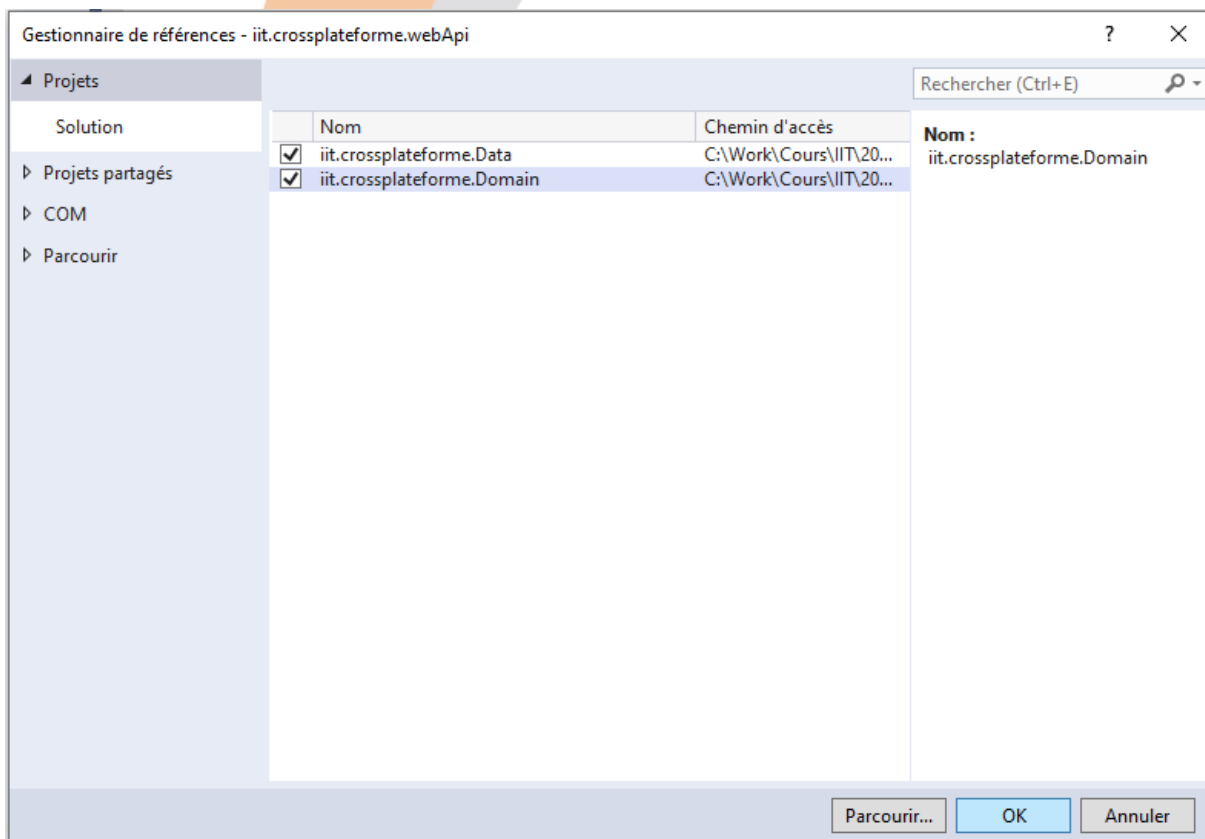
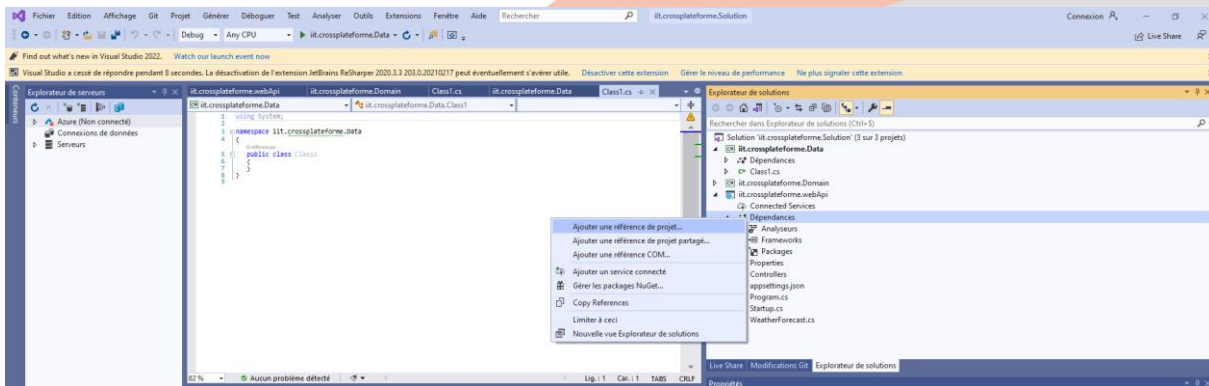
- Domain ne référence aucune couche (couche centrale – cœur métier)
- Data référence la couche Domain

Via menu contextuel sur le projet **iit.crossplateforme.Data** choisir « Ajouter une référence de projet »



- WebApi référence la Couche Data et la couche Domain (Attention - Remarque)

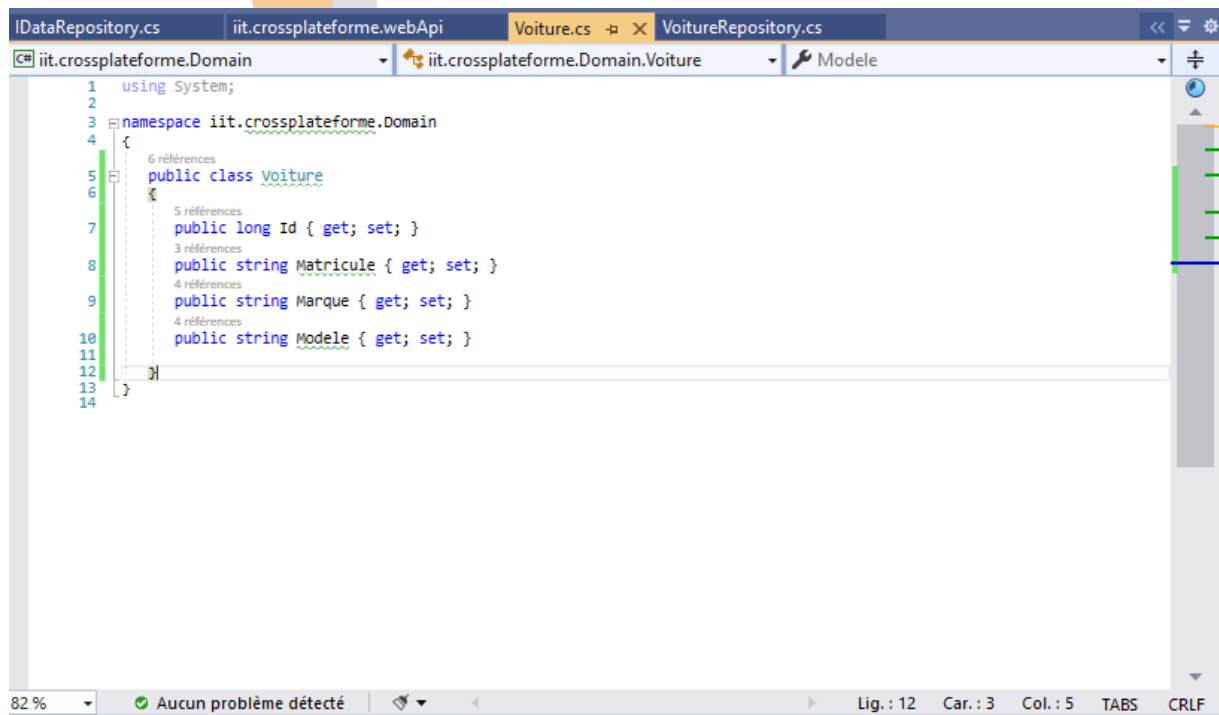
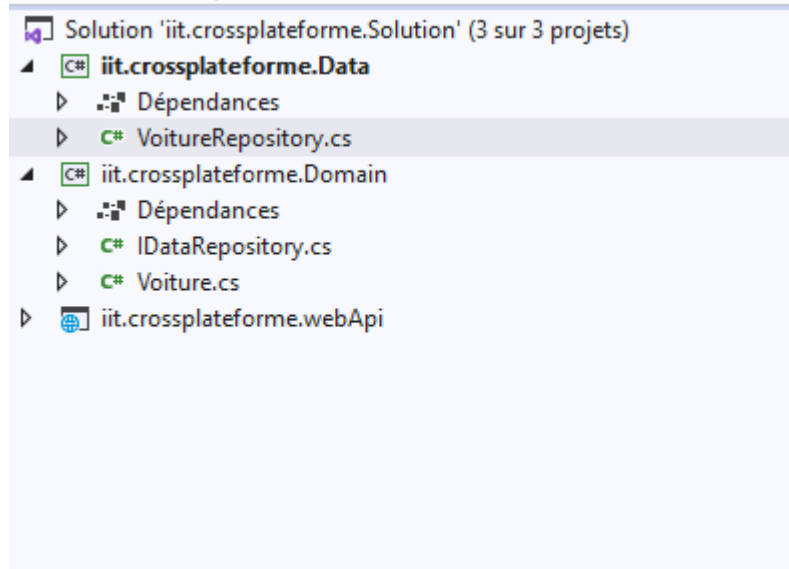
Via menu contextuel sur le projet **iit.crossplateforme.webApi** choisir « Ajouter une référence de projet »



Dans la couche Domain Ajouter une entité « Voiture » et une Interface de repository pour gérer les entités du projet.

Puis dans la couche Data réaliser une implémentation de Repository pour la gestion des voitures.

rechercher dans l'explorateur de solutions (ctrl+g)



```

1 using System.Collections.Generic;
2
3 namespace iit.crossplateforme.Domain
4 {
5     1 référence
6     public interface IDataRepository<TEntity>
7     {
8         1 référence
9         1 référence
10        1 référence
11        1 référence
12        IList<TEntity> GetAll();
13        TEntity Get(long id);
14        void Add(TEntity entity);
15    }
16 }
  
```

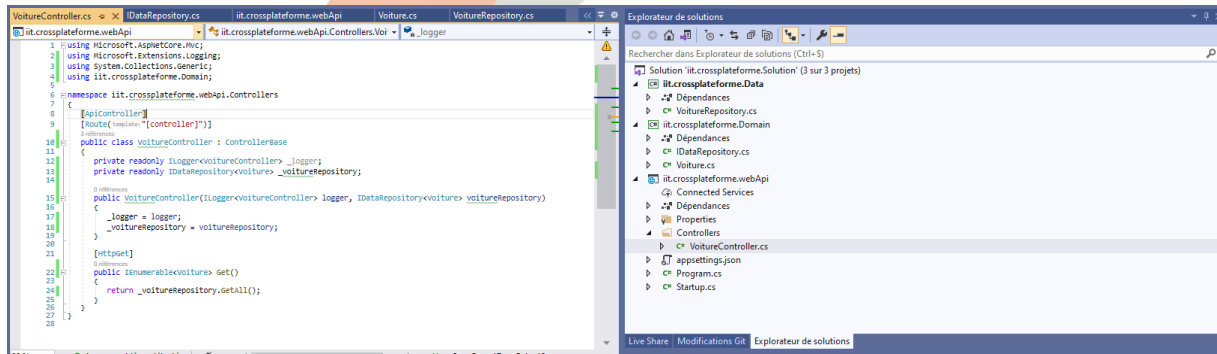
82 % | Aucun problème détecté | Lig.: 8 Car.: 10 Col.: 14 TABS CRLF

```

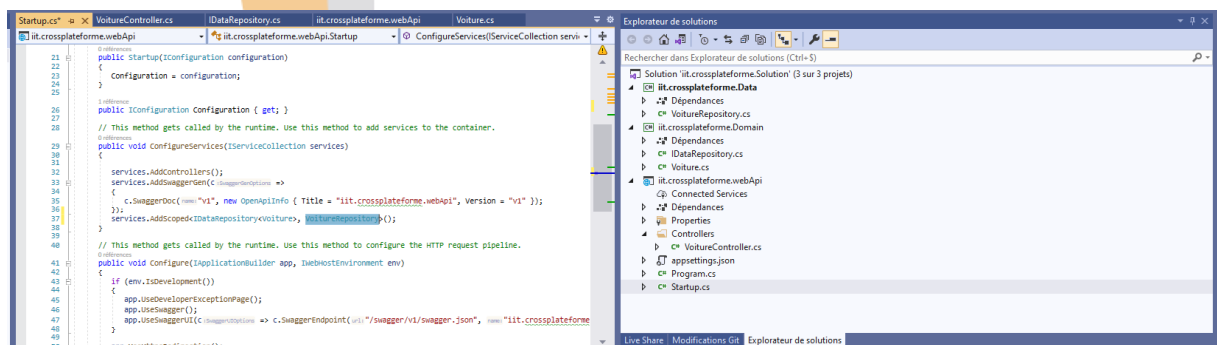
1 using System.Collections.Generic;
2 using iit.crossplateforme.Domain;
3
4 namespace iit.crossplateforme.Data
5 {
6     0 références
7     public class VoitureRepository: IDataRepository<Voiture>
8     {
9         1 référence
10        public IList<Voiture> GetAll()
11        {
12            throw new NotImplementedException();
13        }
14
15        1 référence
16        public Voiture Get(long id)
17        {
18            throw new NotImplementedException();
19        }
20
21        1 référence
22        public void Add(Voiture entity)
23        {
24            throw new NotImplementedException();
25        }
26    }
27 }
  
```

82 % | Aucun problème détecté | Lig.: 12 Car.: 7 TABS CRLF

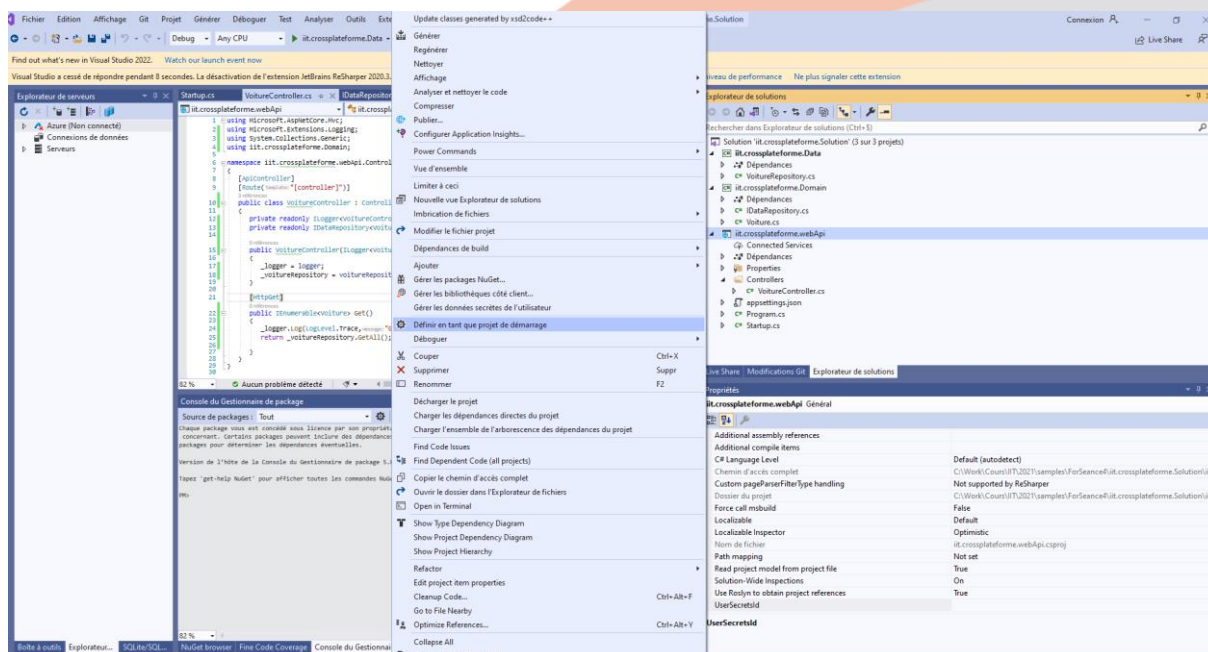
Dans la couche webApi, créer un contrôleur pour l'entité voiture et réaliser un API permettant de récupérer la liste de voitures disponibles.



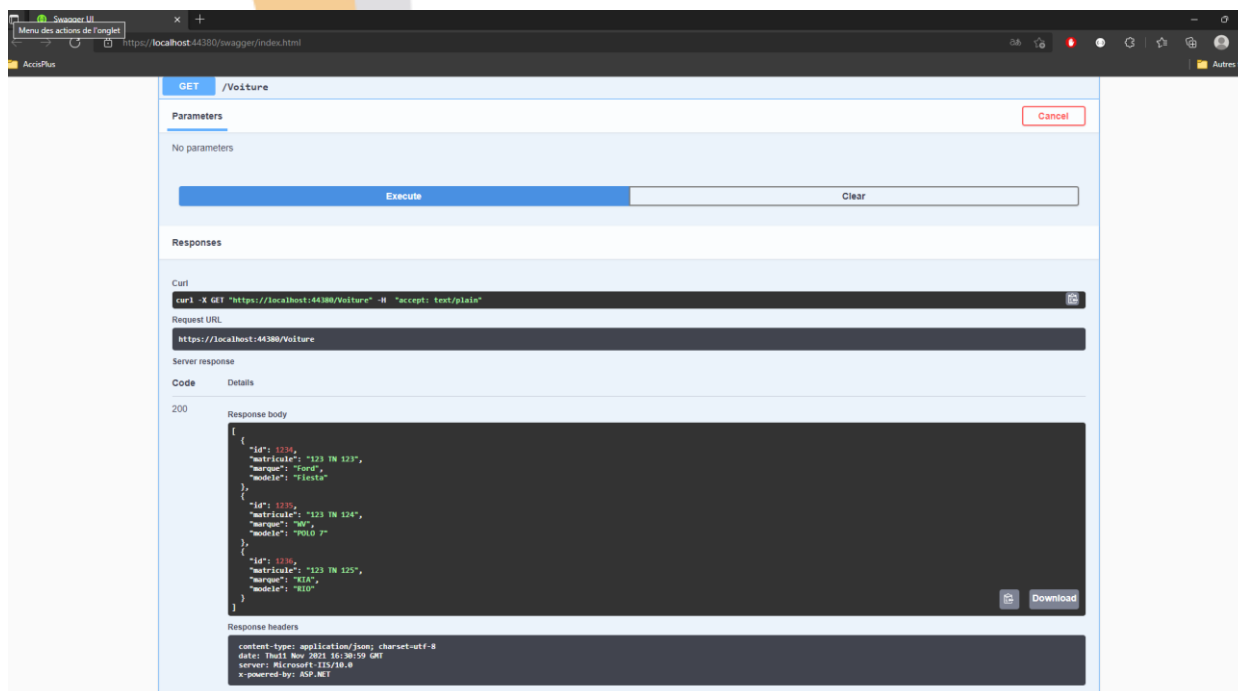
N'oubliez pas de définir l'injection de dépendance du repository dans la classe startup du projet webApi.



Définir le projet **iit.crossplateforme.webApi** comme projet de démarrage puis exécuter l'application.



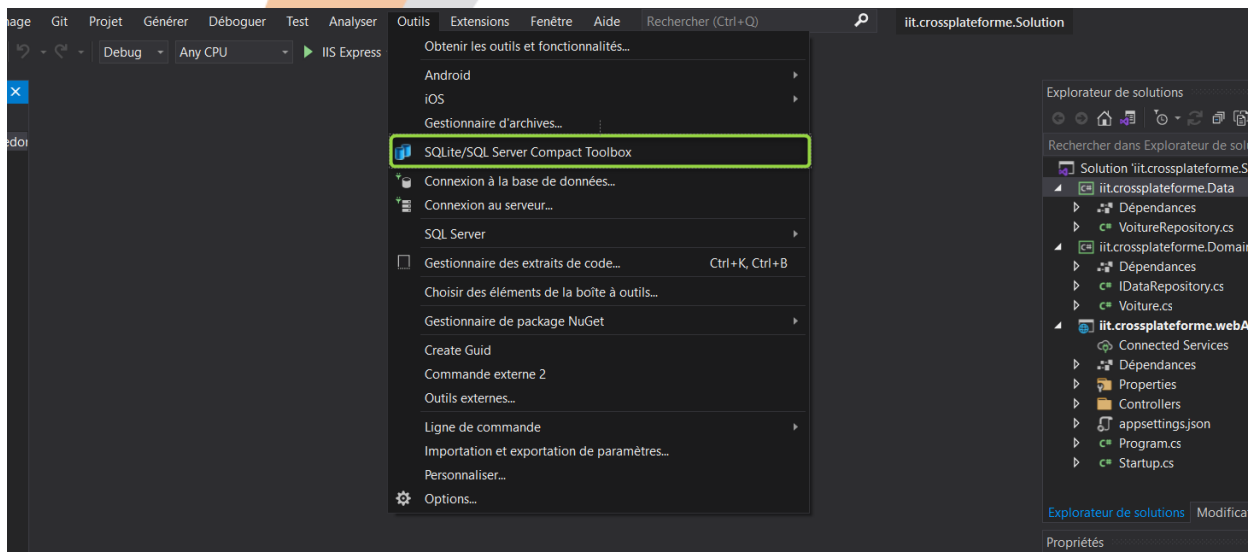
Voici le résultat de l'appel au web service :



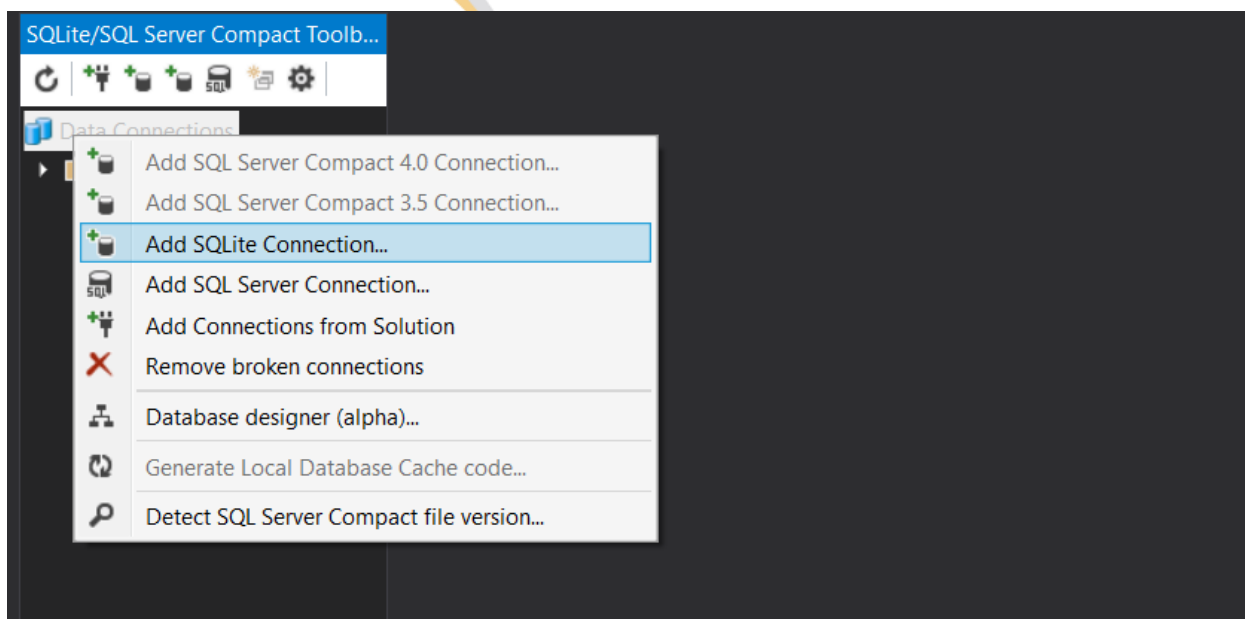
II. Intégration Entity Framework

1. Création Sqlite DB

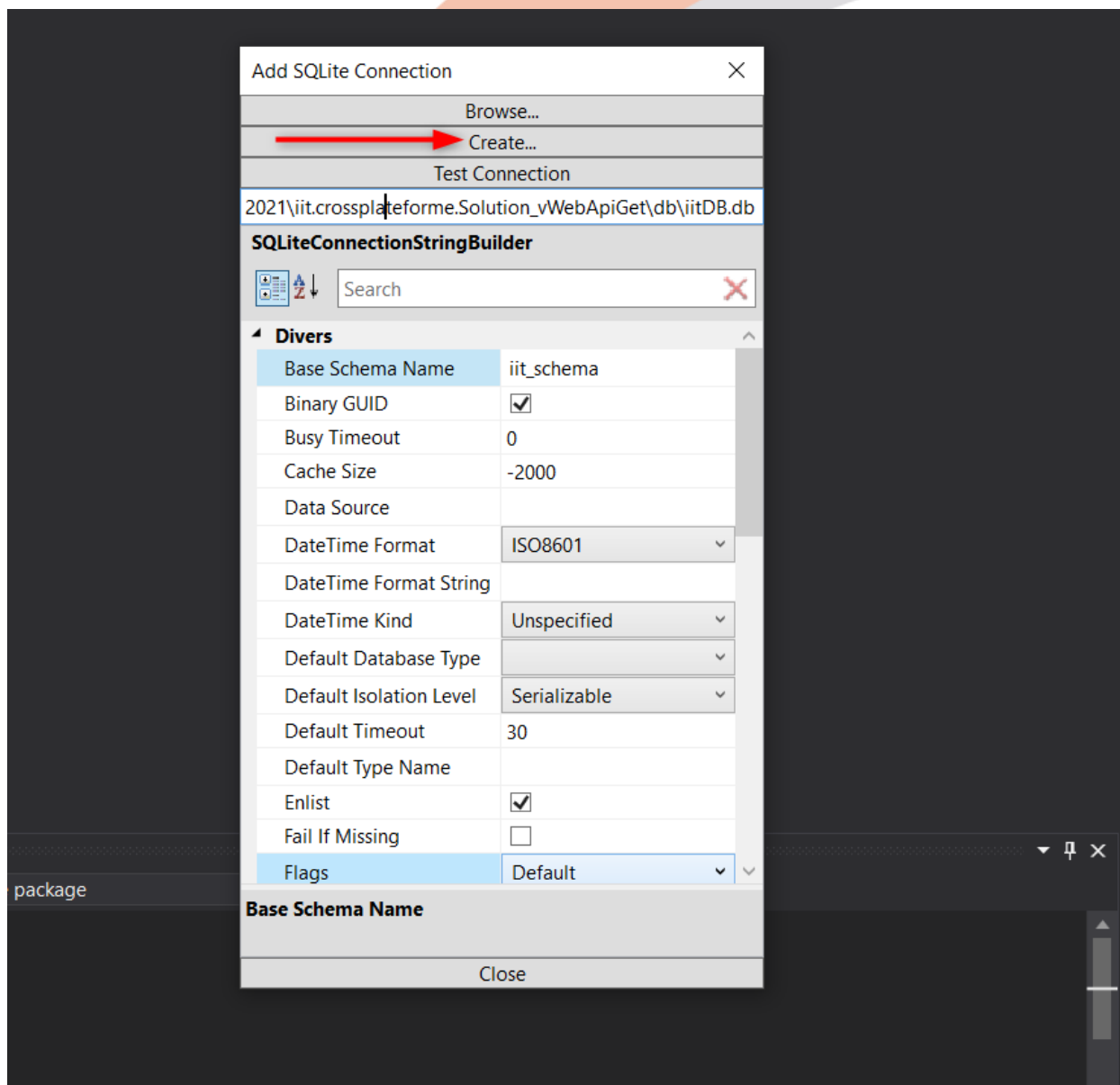
Cliquer sur le menu Outils > SQL Server Compact/SQLite Toolbox.



Sur la fenêtre qui s'affiche, via le menu contextuel choisir « Add SQLite Connection »

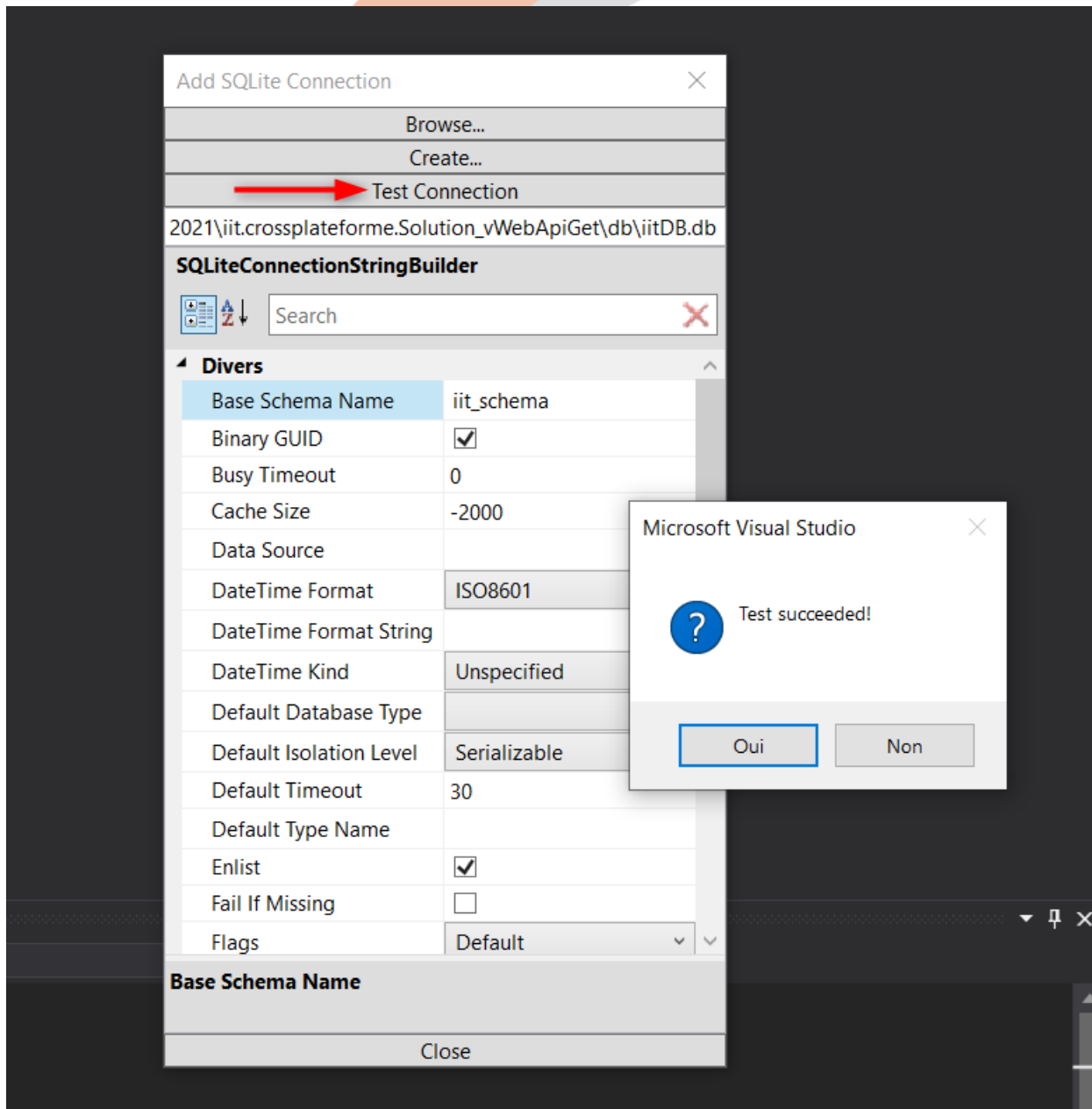


Cliquer sur le bouton « Create »

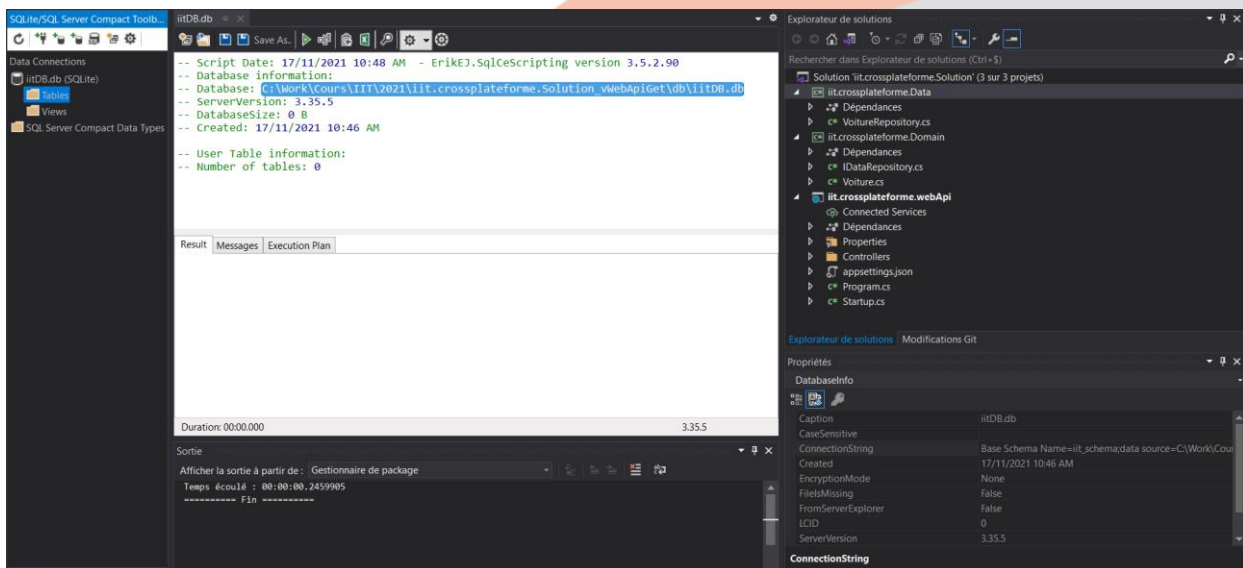


Taper le nom de la base de Données et sauver.

Cliquer sur le bouton "Test Connection" puis cliquer sur OUI.



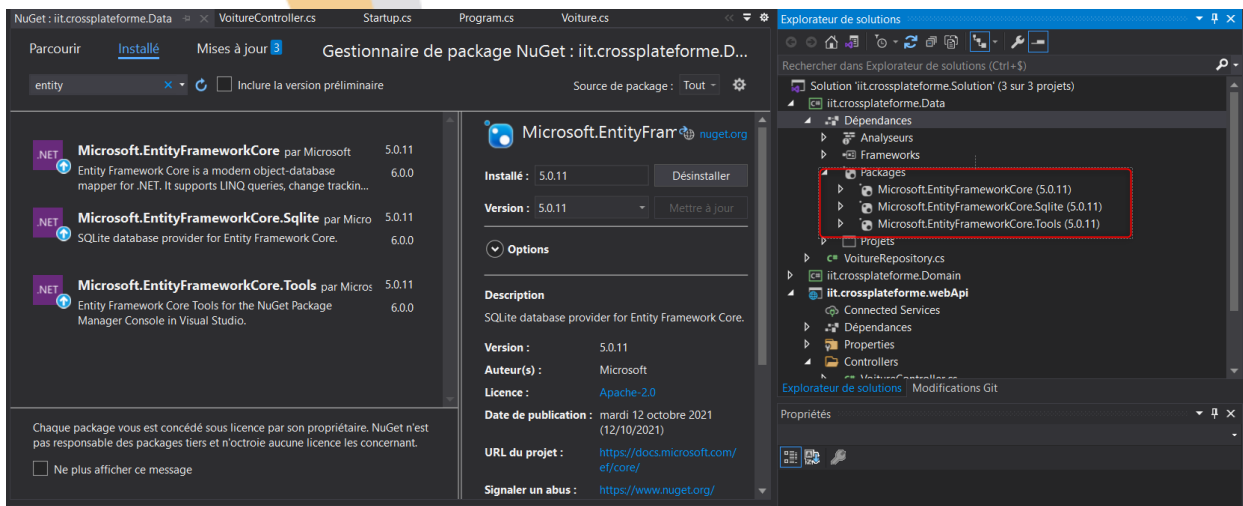
Cliquer sur "iitDB.db (SQLite) (tree view item)" pour consulter le contenu de la nouvelle DB.



2. Entity Framework & Migrations

Ajouter au projet **iit.crossplateforme.Data** les 3 references Nuget suivantes :

- Microsoft.entityFrameworkCore(5.0.11)
- Microsoft.entityFrameworkCore.Sqlite(5.0.11)
- Microsoft.entityFrameworkCore.Tools(5.0.11)



Ajouter la classe context pour l'entité voiture à persister. (mettre dans la commande `optionsBuilder.UseSqlite` la chaine de connexion de votre DB sqlite créée).

```
using iit.crossplateforme.Domain;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
namespace iit.crossplateforme.Data
{
    public class VoitureContext : DbContext
    {
        public DbSet<Voiture> Voitures { get; set; }
        public IConfiguration Configuration { get; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            => optionsBuilder.UseSqlite("data
source=C:\\Work\\Cours\\IIT\\2021\\iit.crossplateforme.Solution_vWebApiGet\\db\\iitDB.
db");

            protected override void OnModelCreating(ModelBuilder modelBuilder)
            {
                {
                    modelBuilder.Entity<Voiture>().HasData(new Voiture
                    {
                        Id = 1,
                        Marque = "FORD",
                        Modele = "Fiesta",
                        Matricule = "123 TN 1234"
                    }, new Voiture
                    {
                        Id = 2,
                        Marque = "KIA",
                        Modele = "Rio",
                        Matricule = "125 TN 1235"
                    });
                }
            }
        }
    }
}
```

Au niveau de la classe startup du projet **iit.crossplateforme.webApi** ajouter le context au container d'application (injection de dépendances)

```

211118103934_iitDb_V1.cs  VoitureContext.cs  Startup.cs  Program.cs  VoitureRepository.cs  VoitureController.cs  Voiture.cs  IDataRepository.cs
iit.crossplateforme.webApi
12 public class Startup
13 {
14     0 références
15     public Startup(IConfiguration configuration)
16     {
17         Configuration = configuration;
18     }
19
20     1 référence
21     public IConfiguration Configuration { get; }
22
23     // This method gets called by the runtime. Use this method to add services to the container.
24     0 références
25     public void ConfigureServices(IServiceCollection services)
26     {
27         services.AddControllers();
28         services.AddSwaggerGen(c =>
29         {
30             c.SwaggerDoc("v1", new OpenApiInfo { Title = "iit.crossplateforme.webApi", Version = "v1" });
31         });
32         services.AddDbContext<VoitureContext>();
33         services.AddScoped<IDataRepository<Voiture>, VoitureRepository>();
34     }
35
36     // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
37     0 références

```

Annoter la propriété Id de l'entité Voiture pour définir cette propriété comme identifiant de la table Voiture.

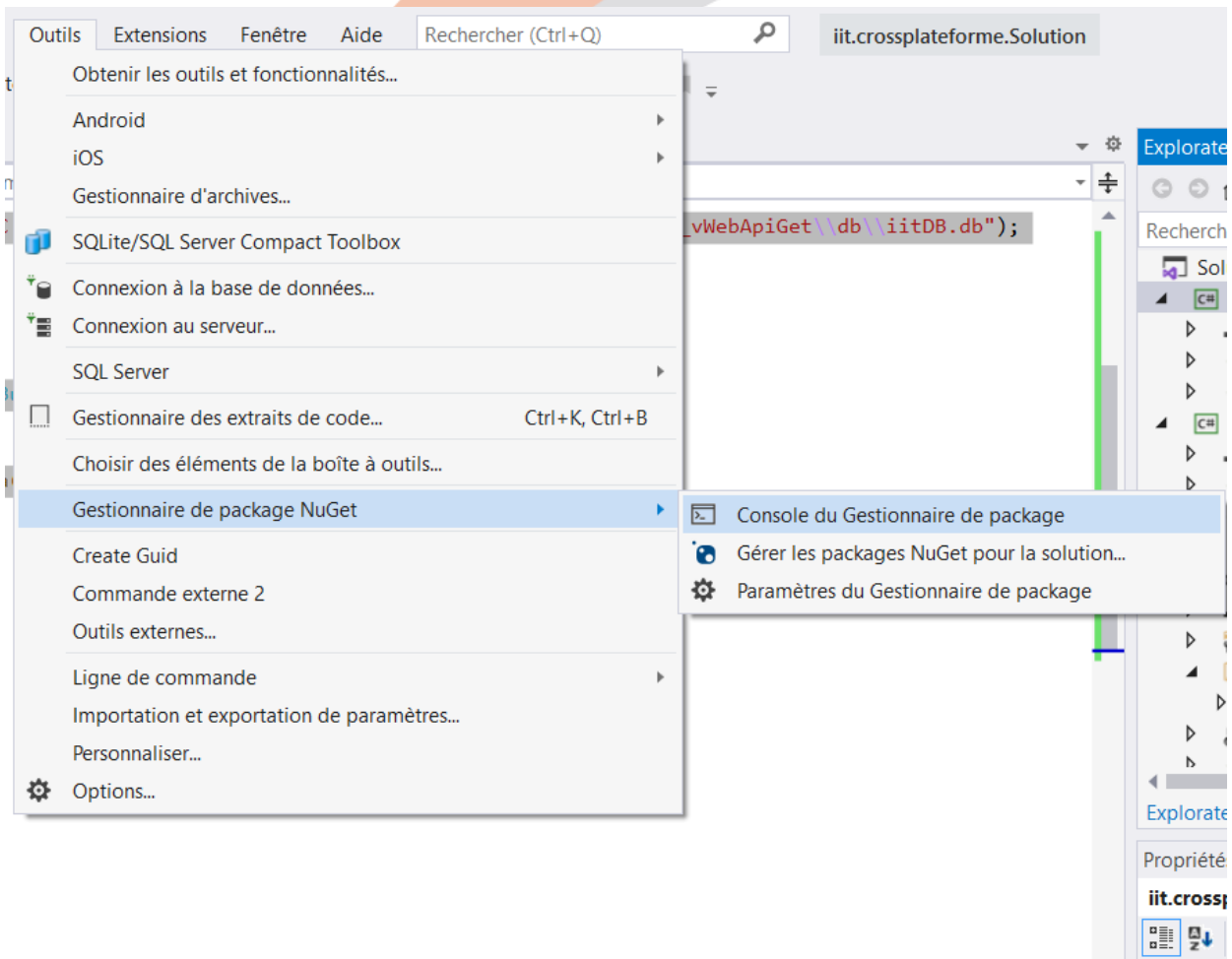
```

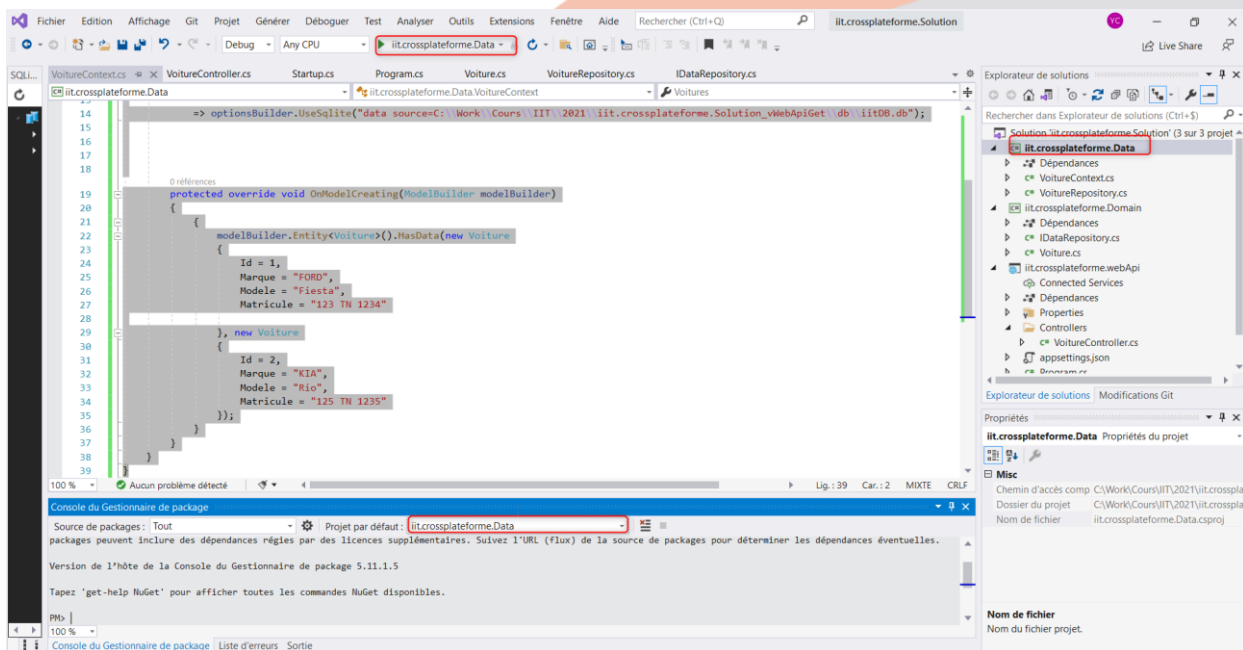
1 using System.ComponentModel.DataAnnotations;
2 using System.ComponentModel.DataAnnotations.Schema;
3
4 namespace iit.crossplateforme.Domain
5 {
6     17 références
7     public class Voiture
8     {
9         [Key]
10         [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
11         public long Id { get; set; }
12         5 références
13         public string Matricule { get; set; }
14         6 références
15         public string Marque { get; set; }
16         6 références
17         public string Modele { get; set; }
18     }
19 }

```

Définir le projet **iit.crossplateforme.Data** comme projet de démarrage.

Ouvrir la console de gestionnaire de package.

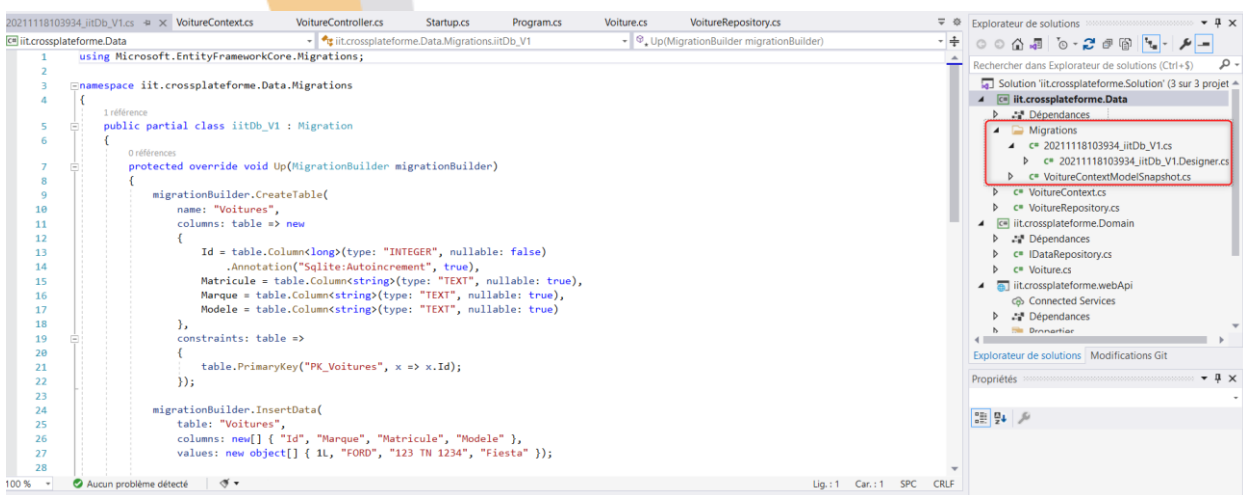




Exécuter les commandes suivantes :

- Add-Migration nom_migration

=>Des classes de migration pour la création de la BD sont générée.



- Update-Database

```
PM> Update-Database
Build started...
Build succeeded.
Applying migration '20211118103934_iitDb_V1'.
Done.
PM> |
```

La table Voiture est crée avec des données initiales dedans.

Id	Matricule	Marque	Modele
1	123 TN 1234	FORD	Fiesta
2	125 TN 1235	KIA	Rio

Adapter **VoitureRepository** pour lire les données depuis la BD créés (au lieu de données statiques) : La classe aurait l'allure suivante :

```
using System.Collections.Generic;
using System.Linq;
using iit.crossplateforme.Domain;

namespace iit.crossplateforme.Data
{
    public class VoitureRepository: IRepository<Voiture>
    {
        private readonly VoitureContext _voitureContext;

        public VoitureRepository(VoitureContext context)
        {
            _voitureContext = context;
        }
    }
}
```

```
public IList<Voiture> GetAll()
{
    return _voitureContext.Voitures.ToList();
}

public Voiture Get(long id)
{
    return _voitureContext.Voitures.FirstOrDefault(v=>
id.Equals(v.Id));
}

public void Add(Voiture entity)
{
    _voitureContext.Voitures.Add(entity);
    _voitureContext.SaveChanges();
}
}
```

Définir le projet **iit.crossplateforme.webApi** comme projet de démarrage.

Exécuter l'application et utiliser l'endpoint créée.