Emetteur : JPF

Objet : Journal de bord TP Xefi Academy 07/03/2023

**Préambule**

L’objectif est de développer en ASP.NET une application web de gestion d’une base de connaissances qui sera utilisée par la communauté des développeurs Stacktim du groupe Xefi.

Architecture de l’application

* Une API REST afin de réaliser le CRUD des entités liées à l’application
* Une application web style SPA simple afin de consommer l’API

1. **Install .Net 7**

<https://dotnet.microsoft.com/en-us/download/dotnet/7.0>

<https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-7.0.201-windows-x64-installer>



Intéressant les liens vers les ressources après l’Install

Ressources

• Documentation .NET sur https://aka.ms/dotnet-docs

• Documentation de kit SDK sur https://aka.ms/dotnet-sdk-docs

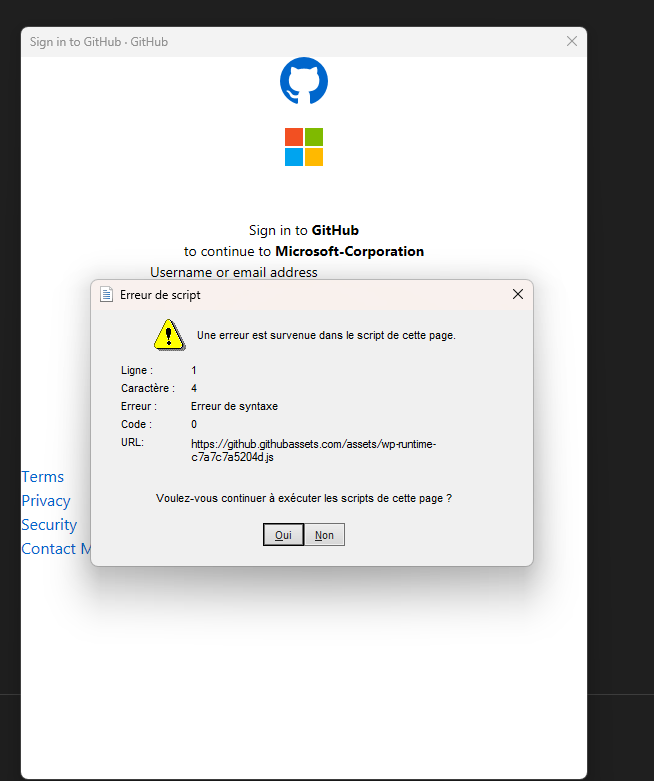
• Notes de publication sur https://aka.ms/dotnet7-release-notes

• Tutoriels sur https://aka.ms/dotnet-tutorials

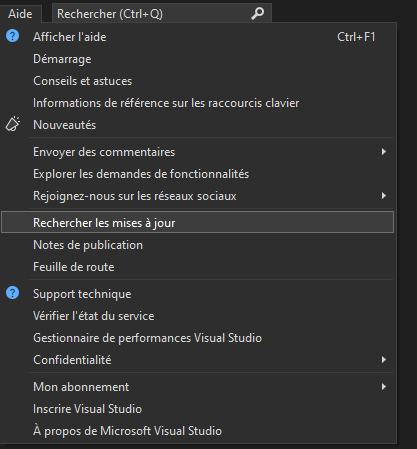
1. **Mise à jour Visual 2022**

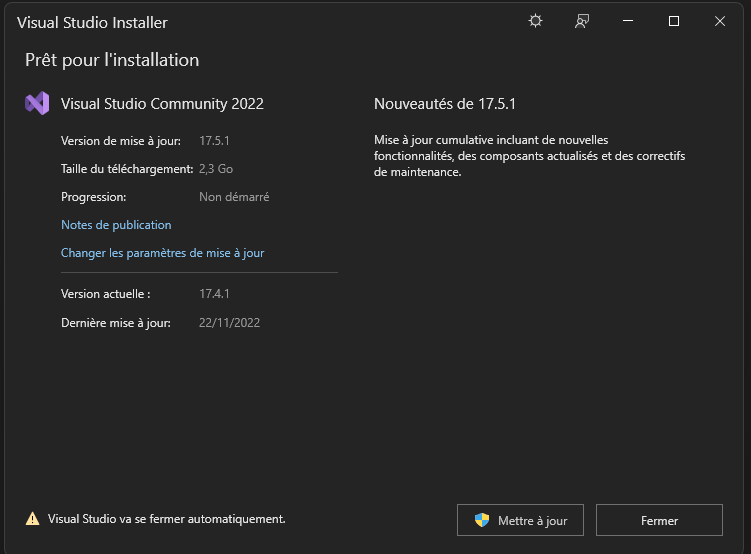
Je vais utiliser VS 2022 Community en m’identifiant avec un compte GitHub

Il a du mal, le script de connexion GitHiub plante.

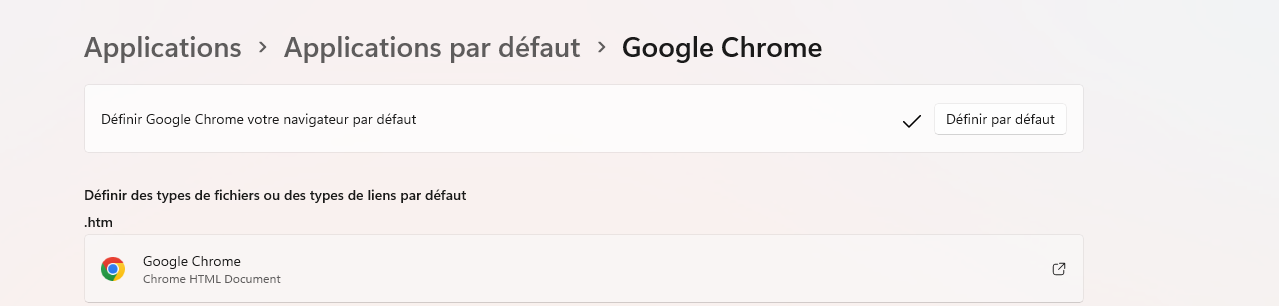


Donc d’abord mettre à Jour VS22





L’erreur js sur la page de login GitHub est peut-être due au navigateur par défaut. Je mets Chrome



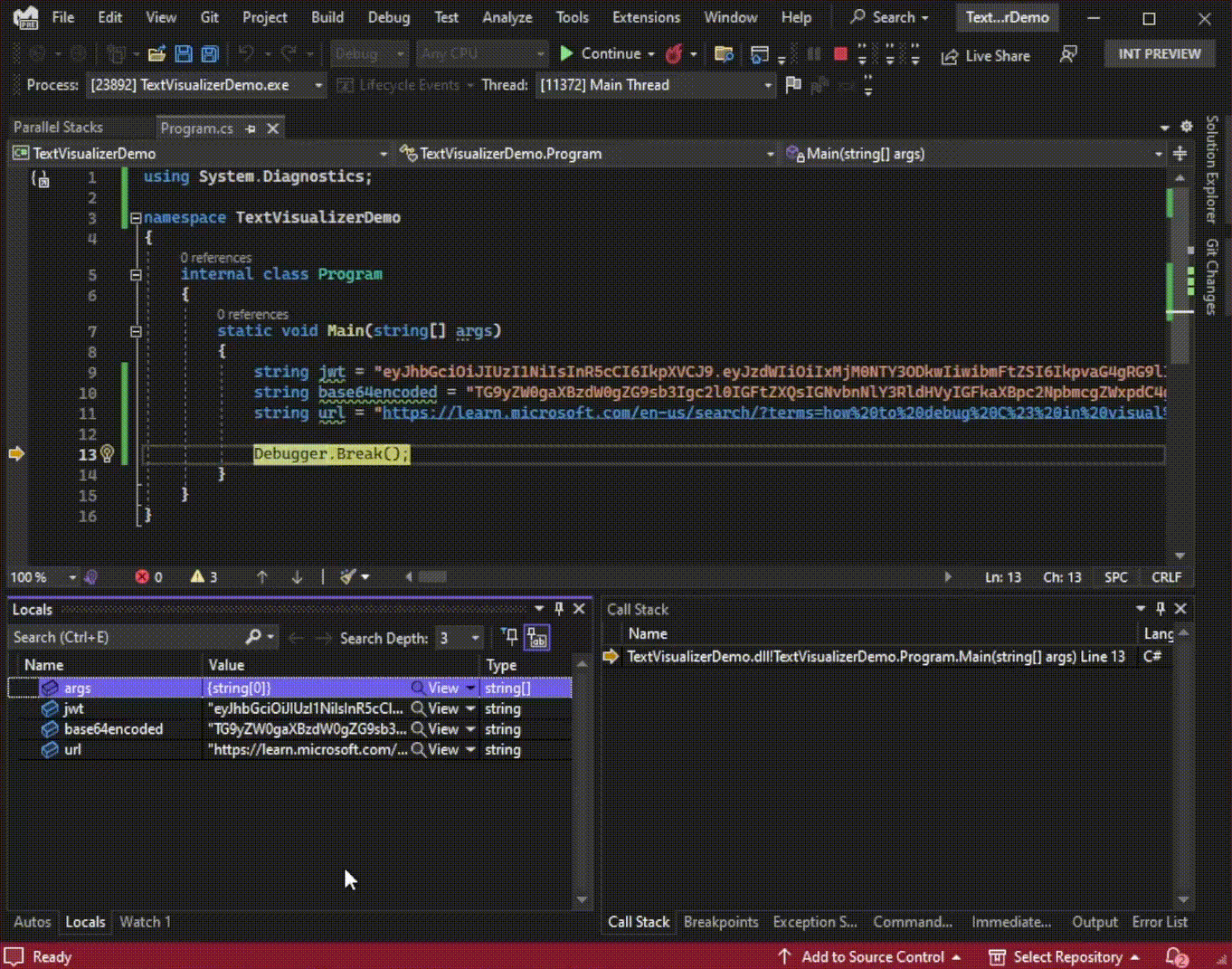
Pendant la mise à jour VS22 on consulte les notes de mise à jour

<https://learn.microsoft.com/en-us/visualstudio/releases/2022/release-notes#17.5.1>

Déjà une modification qu’impactera nos habitudes :

* ASP.NET Core applications launched in Visual Studio now redirect output to the Integrated Terminal Tool Window instead of an external console Window.

Ça, c’est sympa aussi 😊

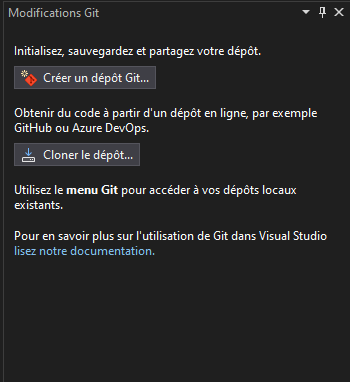


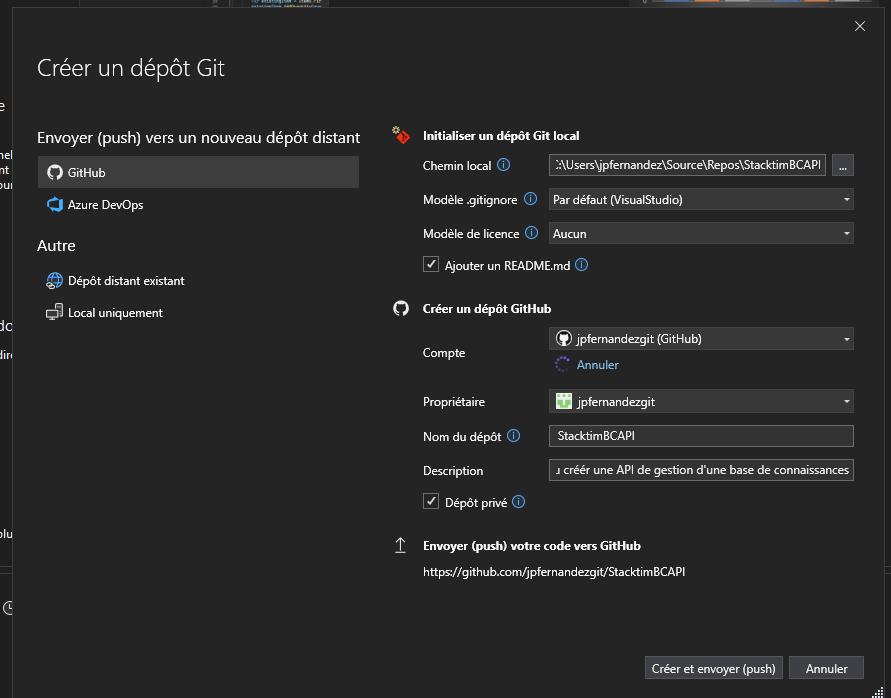
Suite à la mise à jour la connexion avec mon compte perso Github ne marche toujours pas.

Je me connecter donc avec mon Comptes Microsoft @Dailybiz

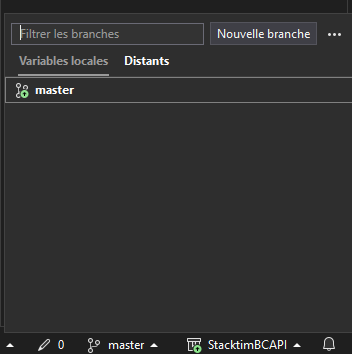
1. **Création d’un repo Github**

Afin de partage le code avec vous je créé un dépôt Git avant toute chose

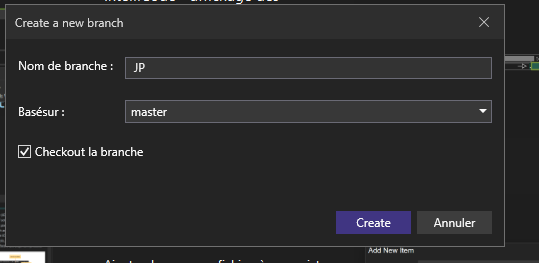


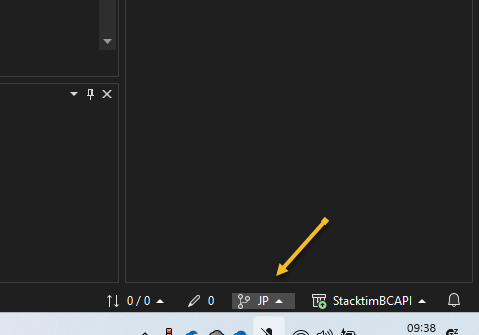


Ensuite je créé une branche « JP » pour ne pas travailler sur Master :



Et on se positionne dessus (checkout)

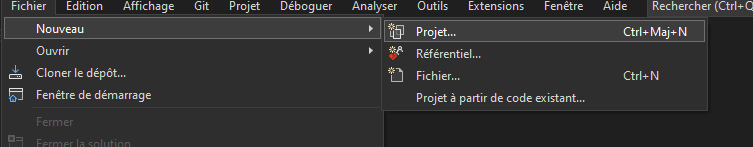


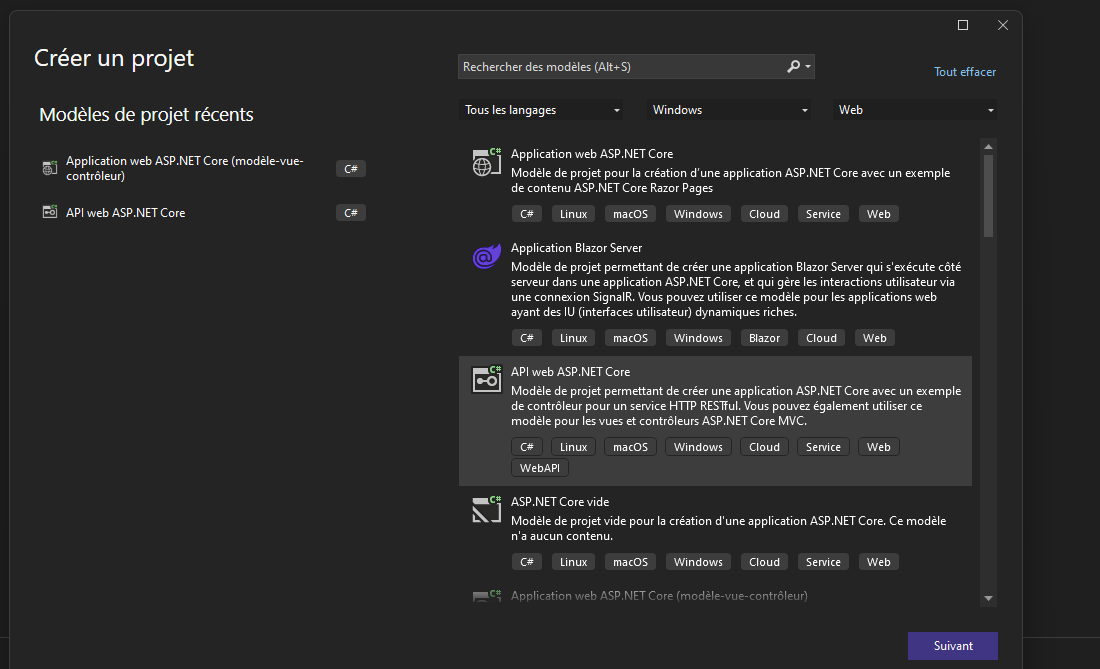


**Le repo a été mis en public**

[**https://github.com/jpfernandezgit/StacktimBCAPI.git**](https://github.com/jpfernandezgit/StacktimBCAPI.git)

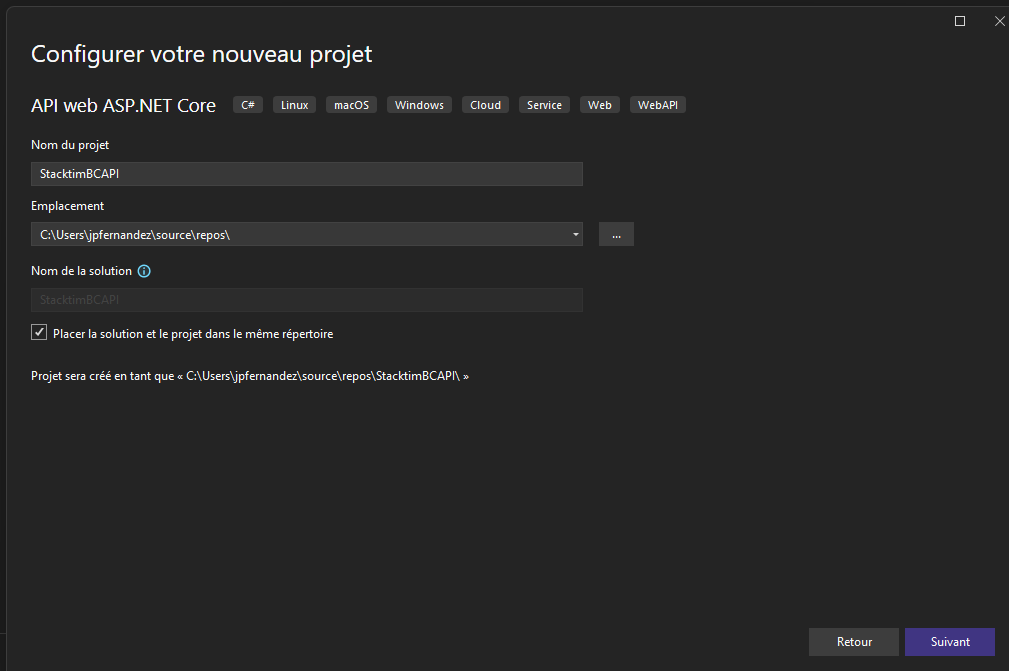
1. **Création du projet API**





Oui, Visual studio parle encore de ASP.NET **Core**, même si « Core » n’est plus le nom officiel de .NET 7

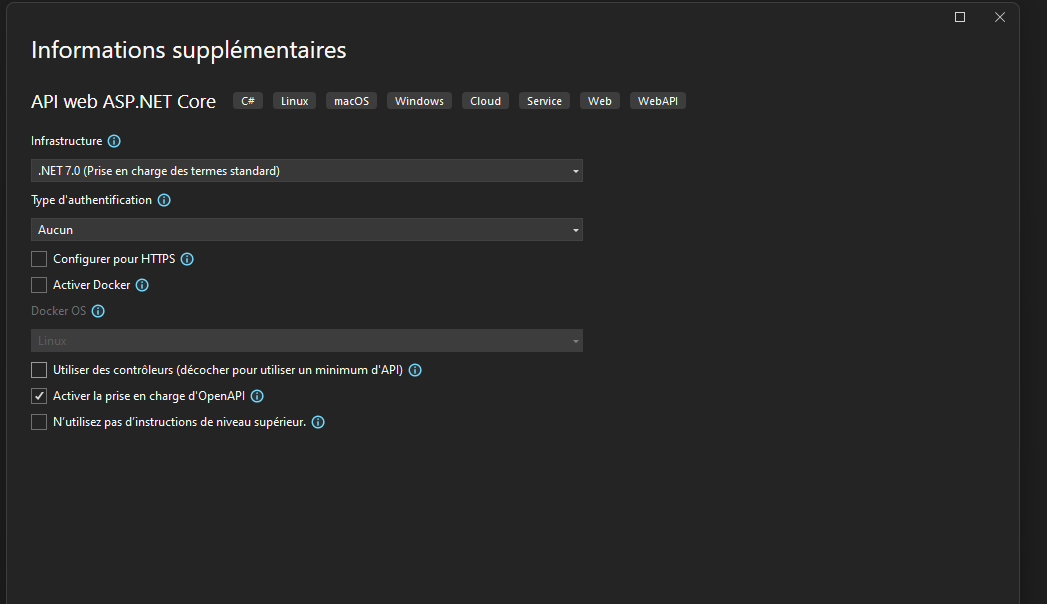
La solution aura 1 seule projet, donc on coche la case pour placer la solution et le projet dans le même dossier, c’est mieux



Autres choix :

htttps : Pas la peine (vue en cours pourquoi)

On décoche aussi l’option pour créér une « Minimal API »



Le projet est créé et apparait en tant que modifications dans ma branche Git

1. **Nettoyage du projet**

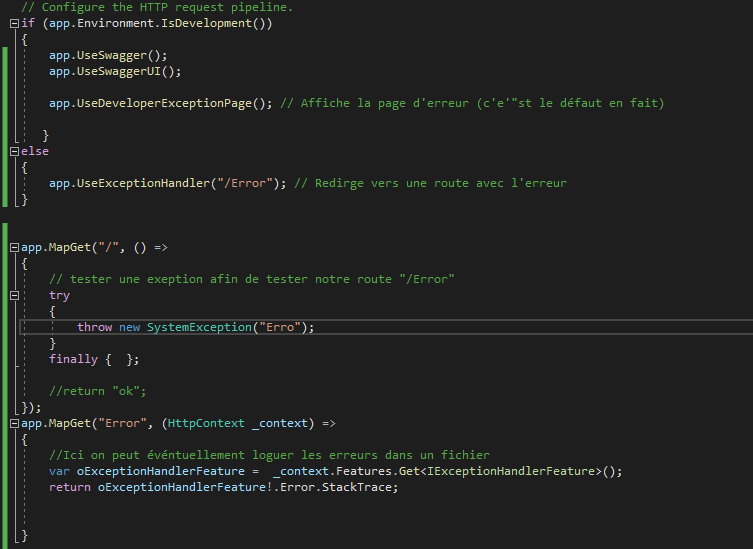
Le projet est créé à partir d’un Template, il faut donc nettoyer Programs.cs et supprimer tout ce qui ne nous intéresse pas (faisant référence à weatherforecast)



1. Gestion des erreurs

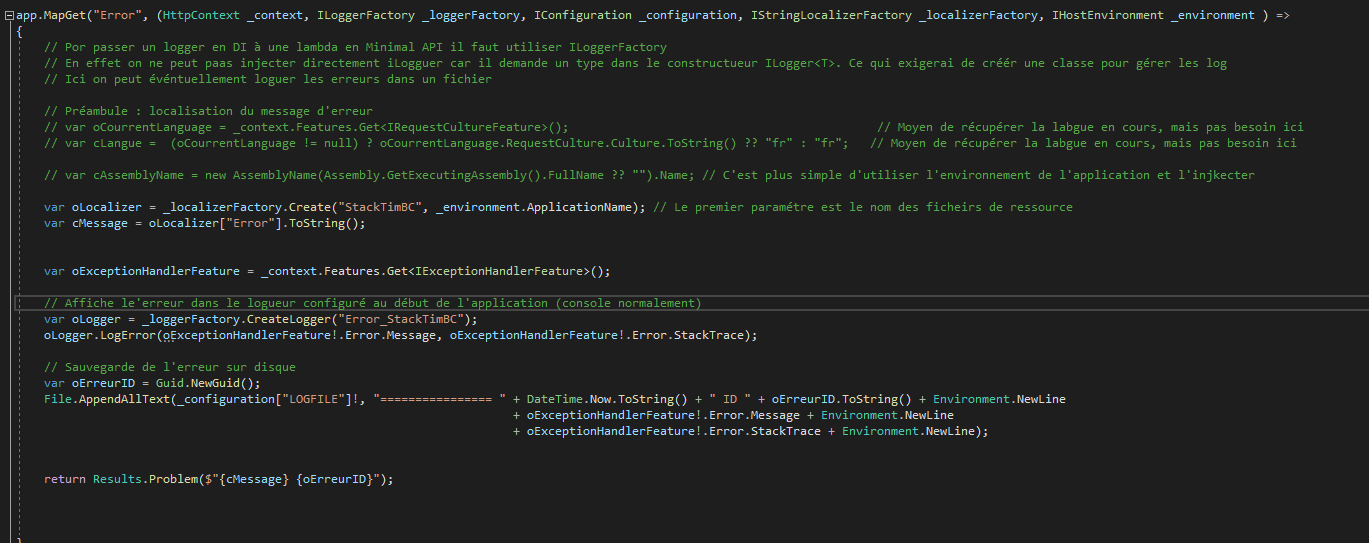
Une bonne pratique dans tous les projets c’est de se préoccuper de la gestion de l’affichage des erreurs à l’utilisateur en cas d’exception. ASP.NET nous aide là-dessus

<https://learn.microsoft.com/fr-fr/aspnet/core/fundamentals/error-handling?view=aspnetcore-7.0>



Dans la nouvelle route « Error » on va loguer les erreurs sur disque

Doc : <https://learn.microsoft.com/en-us/dotnet/core/extensions/logging?tabs=command-line>



Voir le code complet sur le Github

1. Localisation

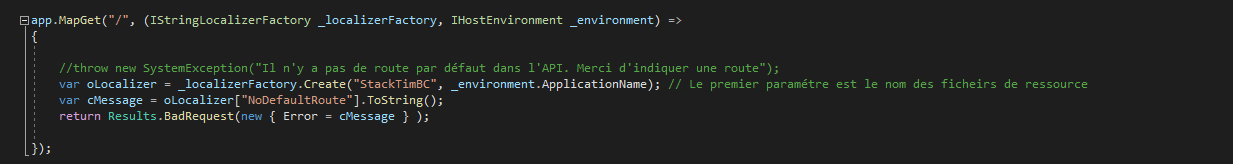
Pour les messages affichés / logués on souhaite que cela soit traduit automatiquement selon la langue de l’utilisateur si jamais il teste l’API dans un navigateur. Sinon ça sera « FR » par défaut.

Pour ça on va créér des fichiers de ressources dans un dossier « Ressources » + utiliser iLocalizerFactory pour aller chercher les messages. Les services

AddLocalization

AddRequestLocalization

Doivent être ajoutés



Voir github pour code complet

1. Création des de la base SQL

On va faire les requêtes dans SQL et les ajouter au projet

On fait le choix de ne pas utiliser Entity Framework

On mettre les requêtes SQL dans un dossier « Migration »

Voir le script dans le dossier

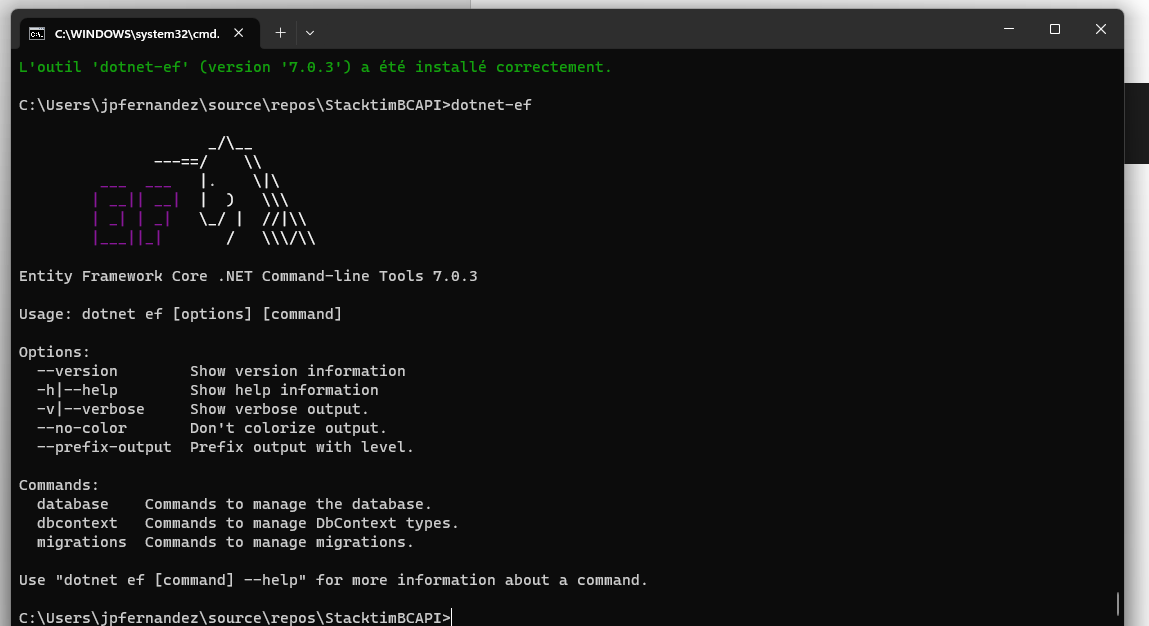
Ensuite on va utiliser EF juste pour générer les classes

Doc : <https://learn.microsoft.com/fr-fr/ef/core/get-started/overview/install>

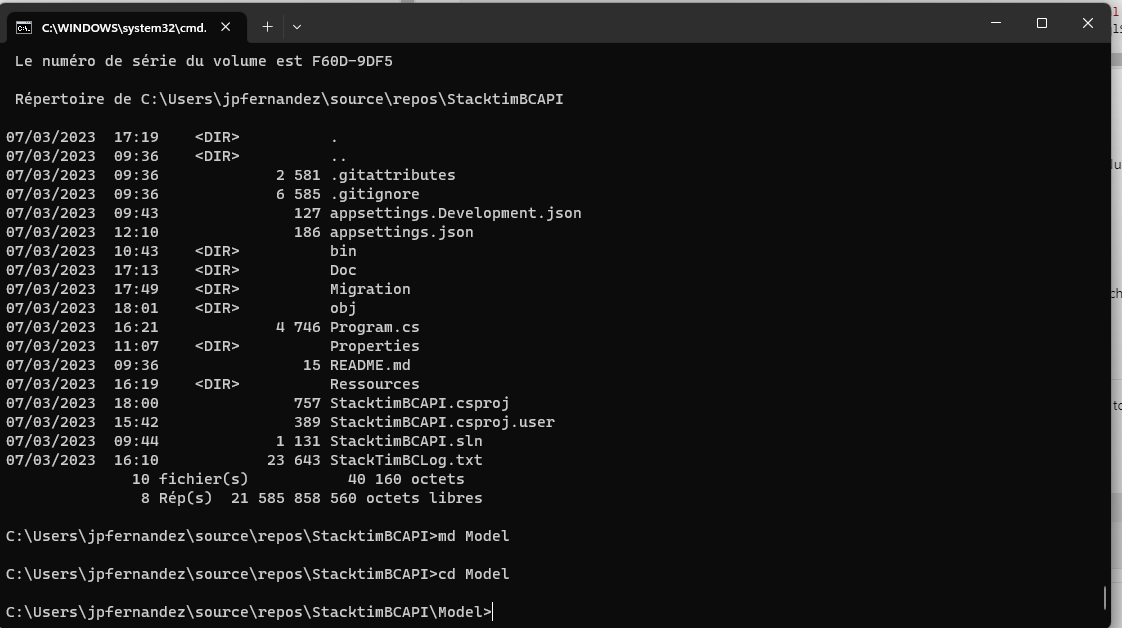
Dans une terminal

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet tool install --global dotnet-ef



On va ajouter un dossier Model au projet



Puis on va demander a EF de gérer les classes à partir de la base, dans ce dossier

dotnet-ef dbcontext scaffold "Data Source= JPF-LEGION\SQL2019;Trusted\_Connection=true ;Initial Catalog=StackTimBC; Encrypt=false" Microsoft.EntityFrameworkCore.SqlServer --output-dir Model