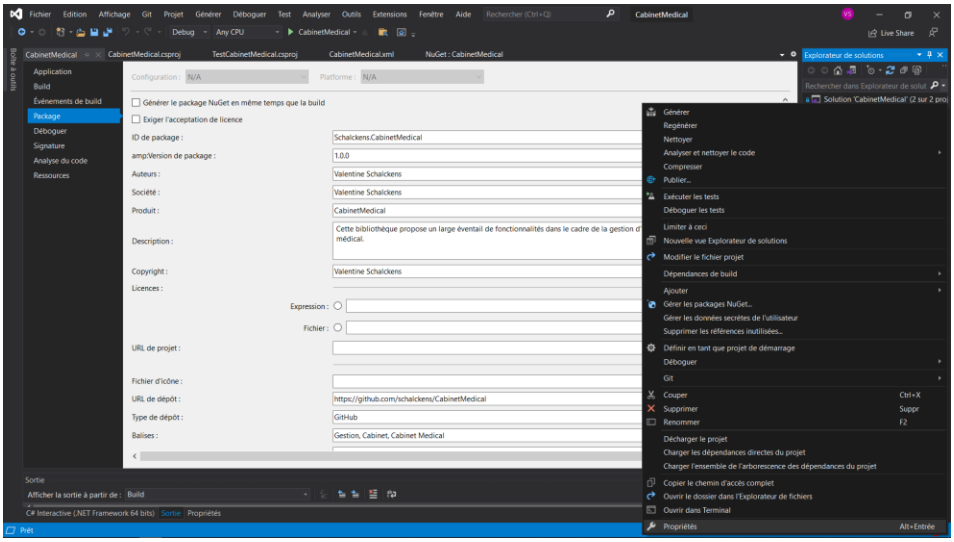
TP 3 : PRESTATIONS SOINS

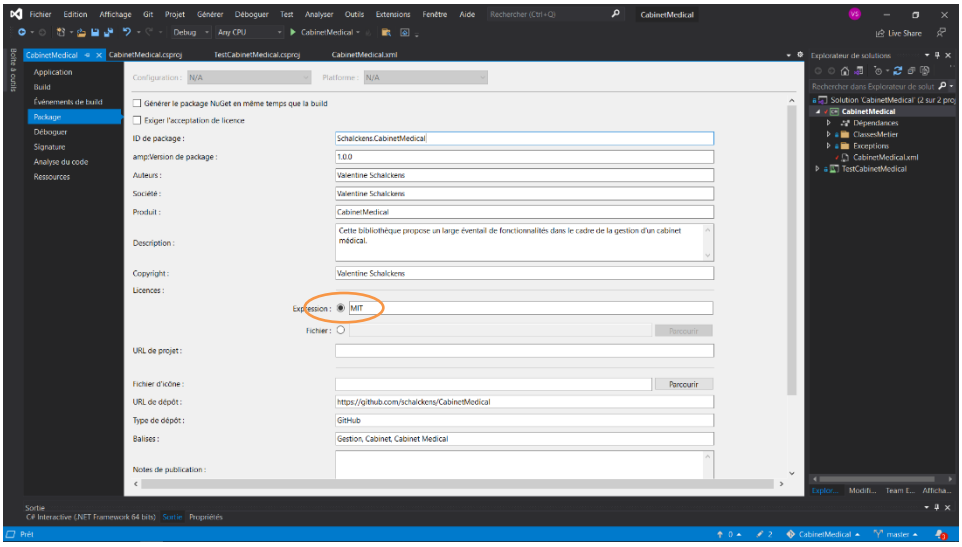
CREATION D'UN PACKAGE ET AJOUT DE FONCTIONNALITE

SOMMAIRE

TUTORIEL: CREATION ET PUBLICATION D'UN PROJET EN TANT QUE PACKAGE NUGET.

1. Création d'un compte sur le site de NuGet.
2. Se rendre dans les propriétés du projet afin de compléter les métadonnées nécessaires à la création d'un package NuGet.



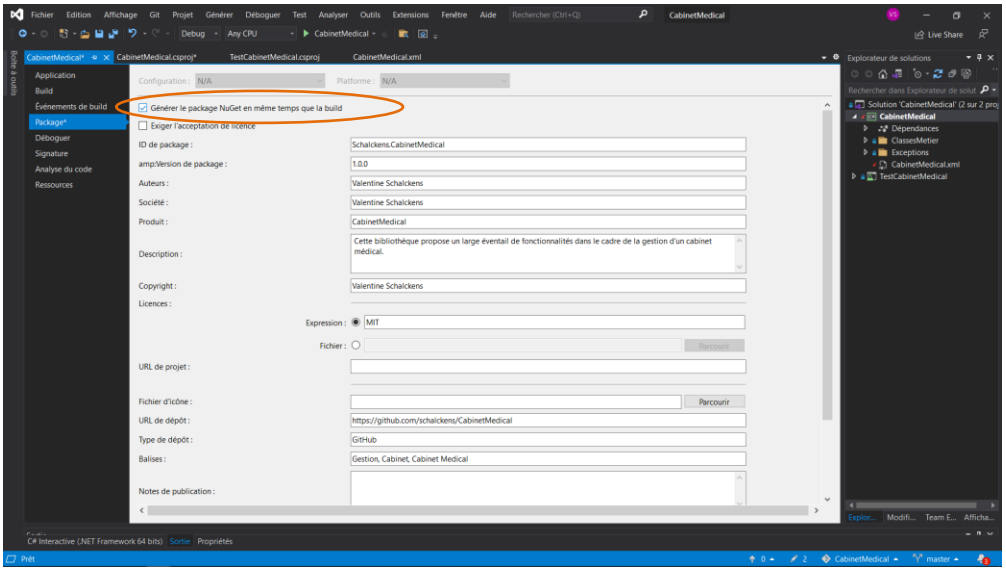


Ici on ajoute une licence afin de ne pas avoir d'erreur lors de l'ajout sur le site NuGet.

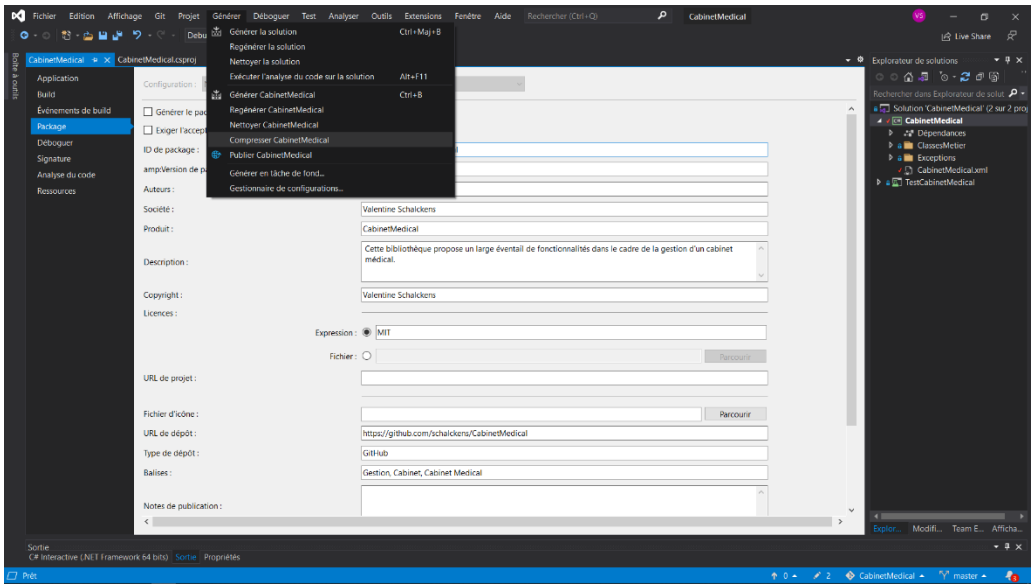
J'ai choisi la licence MIT car elle est très peu contraignante ce qui est parfait pour tester la mécanique de publication d'un package NuGet et me permettra de le réutiliser par la suite facilement.

3. Compresser le projet, pour cela deux manières :

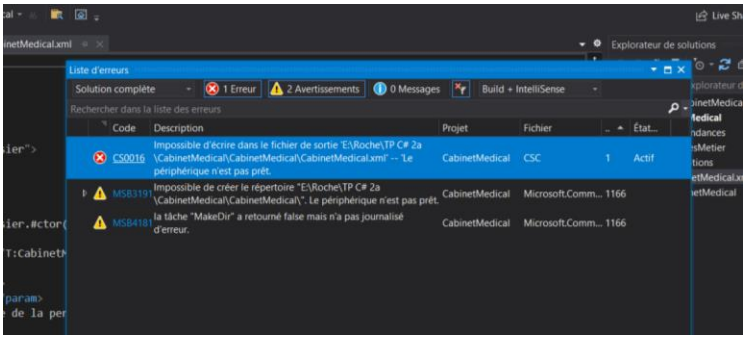
    a. Automatique à chaque génération du projet : cocher la case « Générer le package NuGet en même temps que la build » .



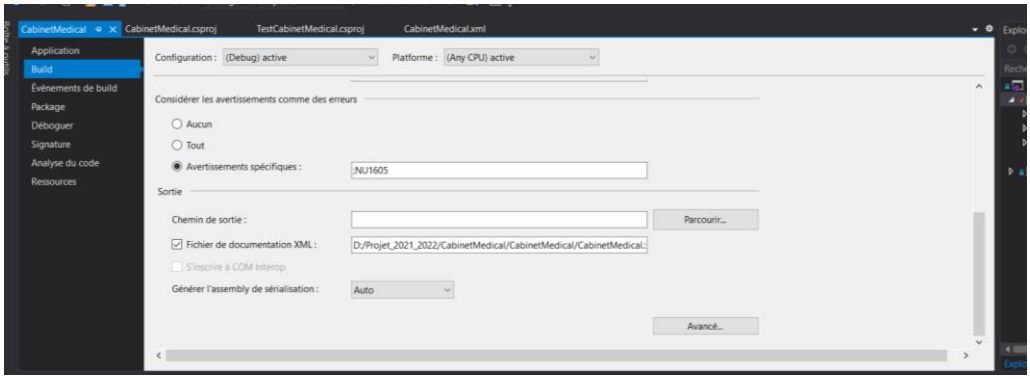
    b. De manière manuelle : aller dans « Générer → Compresser <nom du projet> ».



Problème rencontré et résolu :

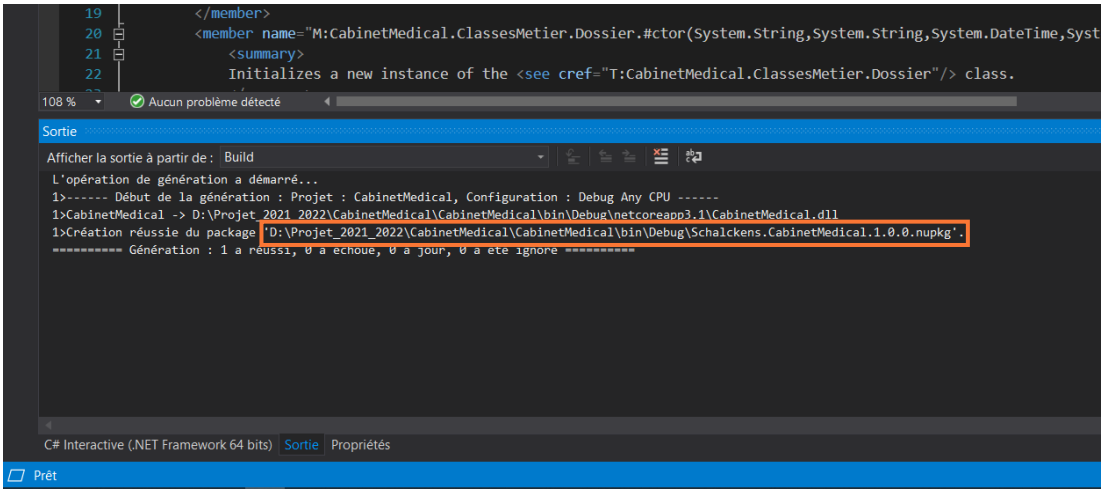


A cause de la perte de ma clé USB, j'ai dû faire un clone de mon projet depuis GitHub pour récupérer mon projet. L'URL du projet a donc changé et cela m'a soulevé une erreur. Pour la résoudre je suis allé dans les propriétés du projet dans « Build » et j'ai modifié le chemin d'accès au fichier XML.

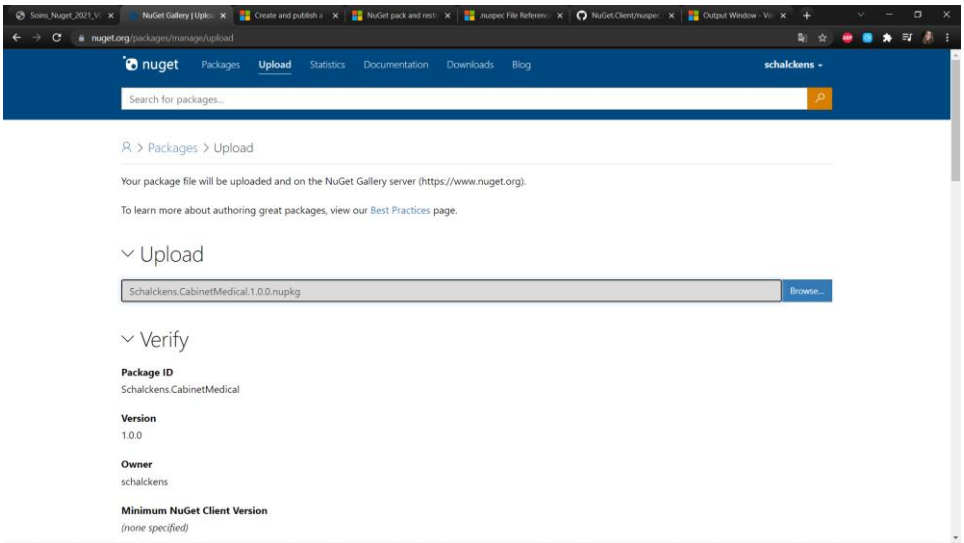


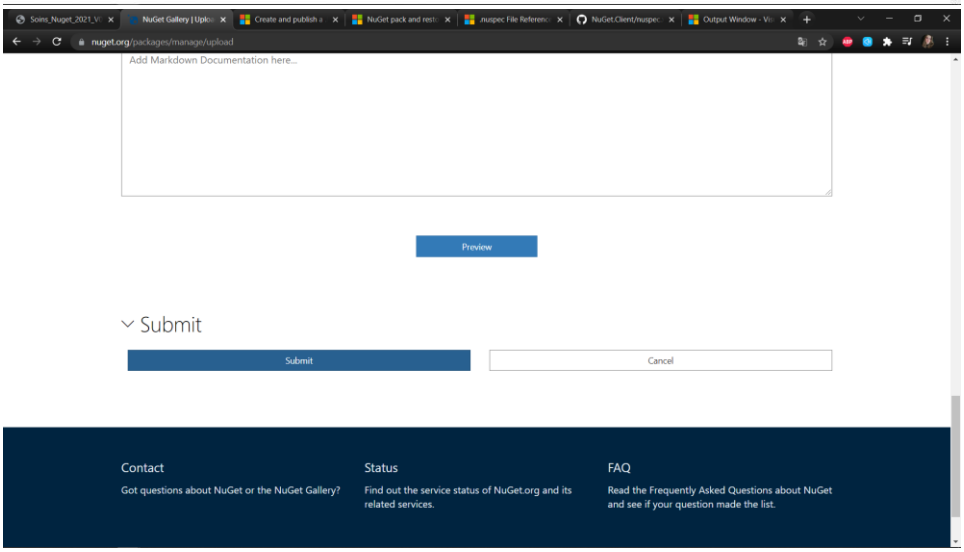
---

4. Si réussite, elle s'affiche dans l'onglet « Output » de Visual Studio :

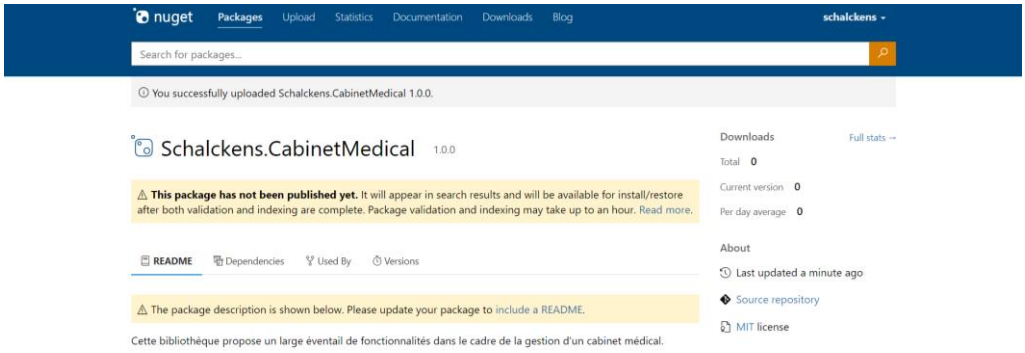


5. Se rendre sur NuGet dans la rubrique « Upload », nous pouvons maintenant lui donner le chemin d'accès au package que nous venons de créer. Après vérification des informations nous pouvons cliquer sur « Submit » en bas de la page :





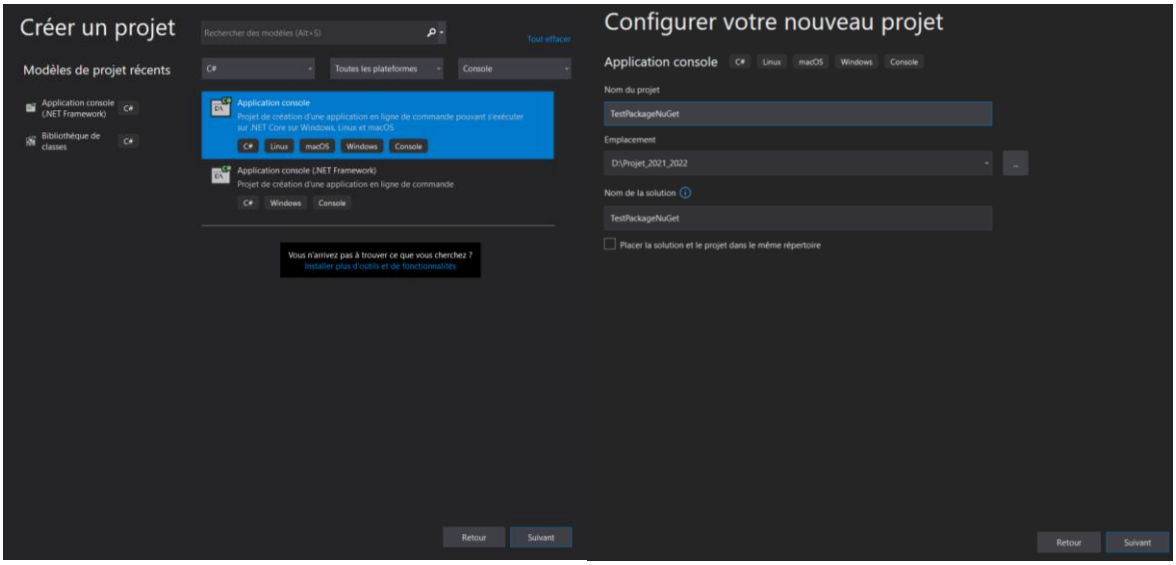
Félicitation ! Le Package est maintenant créé et publié sur NuGet !



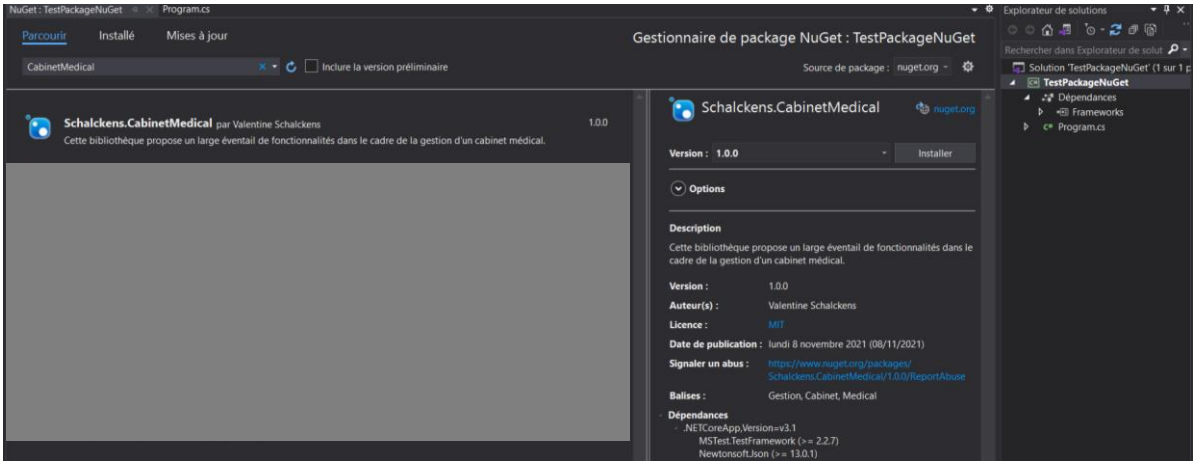
Il ne reste plus qu'à attendre sa validation qui se fait en quelques minutes !

PARTIE I: TEST DU PACKAGE CREER

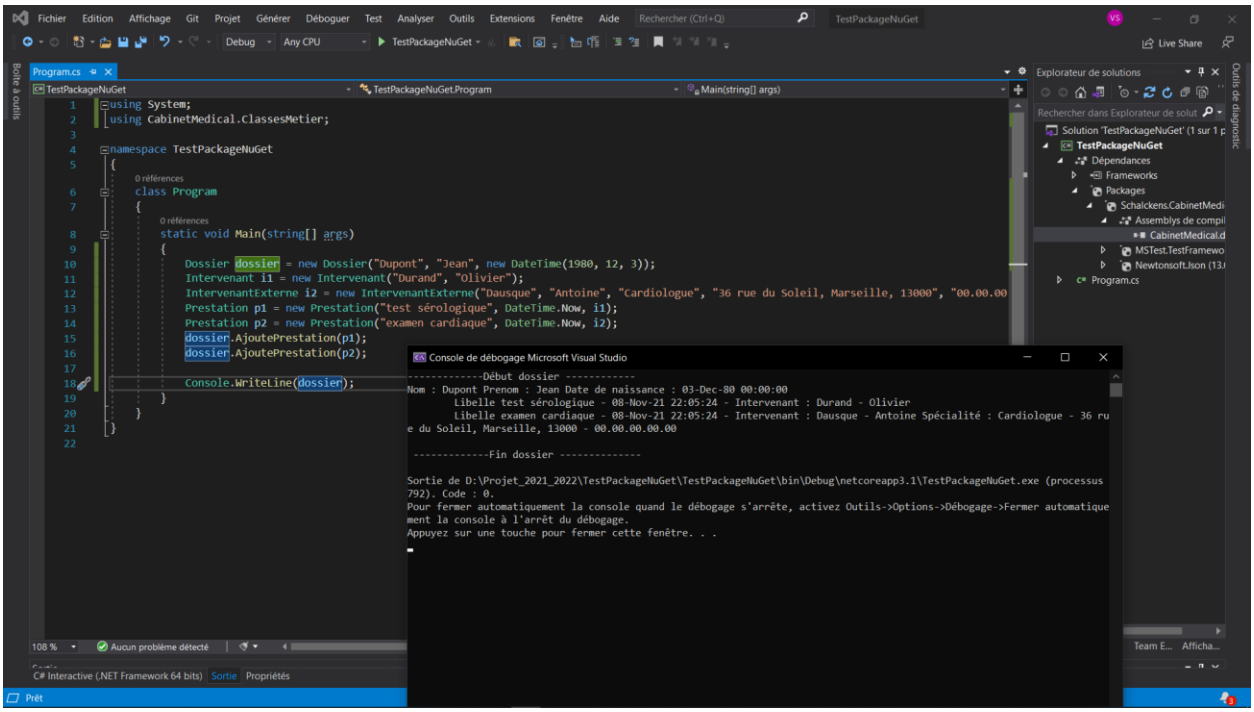
1. Création d'un nouveau projet :



2. Installation du package NuGet « Schalckens.CabinetMedical » : Il faudra accepter les conditions de la licence et si tout s'est bien passé on ne doit pas avoir d'erreur à la fin de l'installation.



3. Test du package : On devrait maintenant avoir accès aux fonctionnalités du package !



PARTIE II: AJOUT D'UNE FONCTIONALITÉ (+TUTORIEL)

1. Ajout de la classe Cotation, modification des autres classes et implémentation de test :

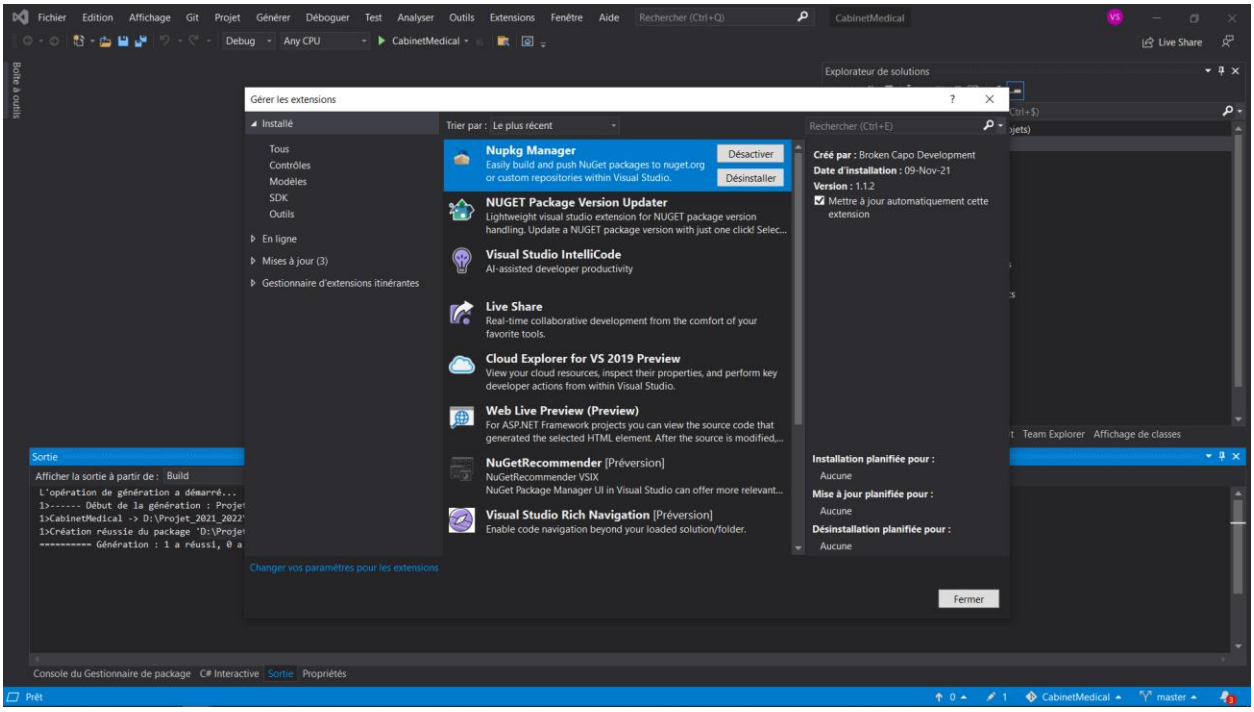
```
namespace CabinetMedical.ClassesMetier
{
    using CabinetMedical.Exceptions;

    /// <summary>
    /// Classe Cotation.
    /// </summary>
    2 références
    public class Cotation
    {
        private string id;
        private string libelle;
        private int indice;

        /// <summary>
        /// Initializes a new instance of the <see cref="Cotation"/> class.
        /// Constructeur de la Classe Cotation.
        /// </summary>
        /// <param name="id"> id de la cotation. </param>
        /// <param name="libelle">l libellé de la cotation. </param>
        /// <param name="indice"> indice de cotation. </param>
        0 références
        public Cotation(string id, string libelle, int indice)
        {
            this.id = id;
            this.libelle = libelle;
            if (indice > 0)
            {
                this.indice = indice;
            }
            else
            {
                throw new CabinetMedicalException("Indice négatif ou égale à 0.");
            }
        }

        /// <summary>
        /// Gets id.
        /// </summary>
        0 références
        public string Id { get => this.id; }

        /// <summary>
        /// Gets or Sets libelle.
        /// </summary>
        0 références
        public string Libelle { get => this.libelle; set => this.libelle = value; }

        /// <summary>
        /// Gets or Sets indice.
        /// </summary>
        0 références
        public int Indice { get => this.indice; set => this.indice = value; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using CabinetMedical.ClassesMetier;
using CabinetMedical.Exceptions;

namespace TestCabinetMedical
{
    [TestClass]
    0 références
    class CotationTest
    {
        [TestMethod]
        0 références
        public void TestInstanceCotationOK()
        {

            Cotation cotation = new Cotation("3e","cotation 1", 2);
            Assert.IsInstanceOfType(cotation, typeof(Cotation));
        }

        [TestMethod]
        [ExpectedException(typeof(CabinetMedicalException))]
        0 références
        public void TestInstanceCotationKO()
        {
            Cotation cotation = new Cotation("3g","cotation 2", -1);
        }
    }
}
```

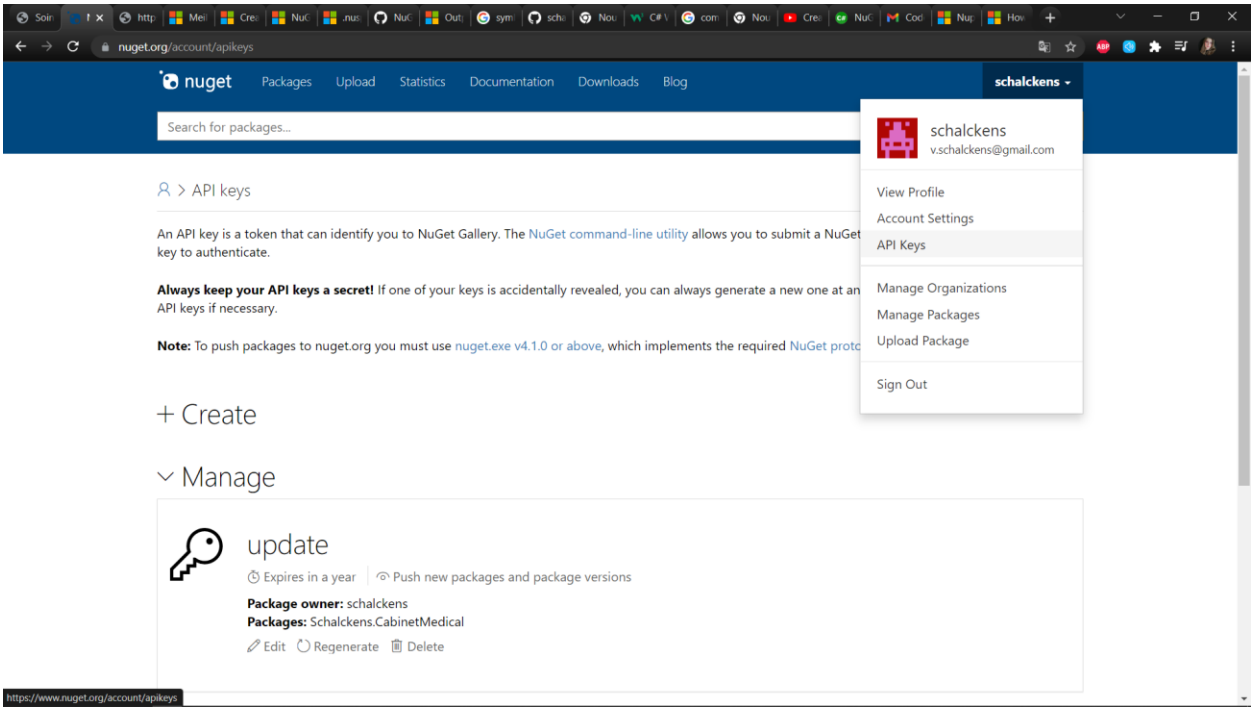
```
public class Prestation
{
    private int prixFixe = 100;
    private int prixCourant;
    private Cotation cotation;

    // Méthodes

    /// <summary>
    /// Initializes a new instance of the <see cref="Prestation"/> class.
    /// </summary>
    /// <param name="libelle">Libéllé.</param>
    /// <param name="dateHeureSoin">Date et heure de la prestation.</param>
    /// <param name="intervenant">Objet de la classe Intervenant.</param>
    /// <param name="cotation">Objet de la classe Cotation.</param>
    18 références
    public Prestation(string libelle, DateTime dateHeureSoin, Intervenant intervenant, Cotation cotation)
    {
        this.Libelle = libelle;
        if (DateTime.Compare(DateTime.Now.Date, dateHeureSoin.Date) >= 0)
        {
            this.DateHeureSoin = dateHeureSoin;
        }
        else
        {
            throw new CabinetMedicalException(" Date non conforme ");
        }

        this.Intervenant = intervenant;
        this.cotation = cotation;
        this.prixCourant = cotation.Indice * this.prixFixe;
    }
```

```
    public int PrixFixe { get => this.prixFixe; set => this.prixFixe = value; }

    /// <summary>
    /// Gets or Sets prixCourant.
    /// </summary>
    0 références
    public int PrixCourant { get => this.prixCourant; set => this.prixCourant = value; }

    /// <summary>
    /// Gets or Sets cotation.
    /// </summary>
    0 références
    public Cotation Cotation { get => this.cotation; set => this.cotation = value; }
```

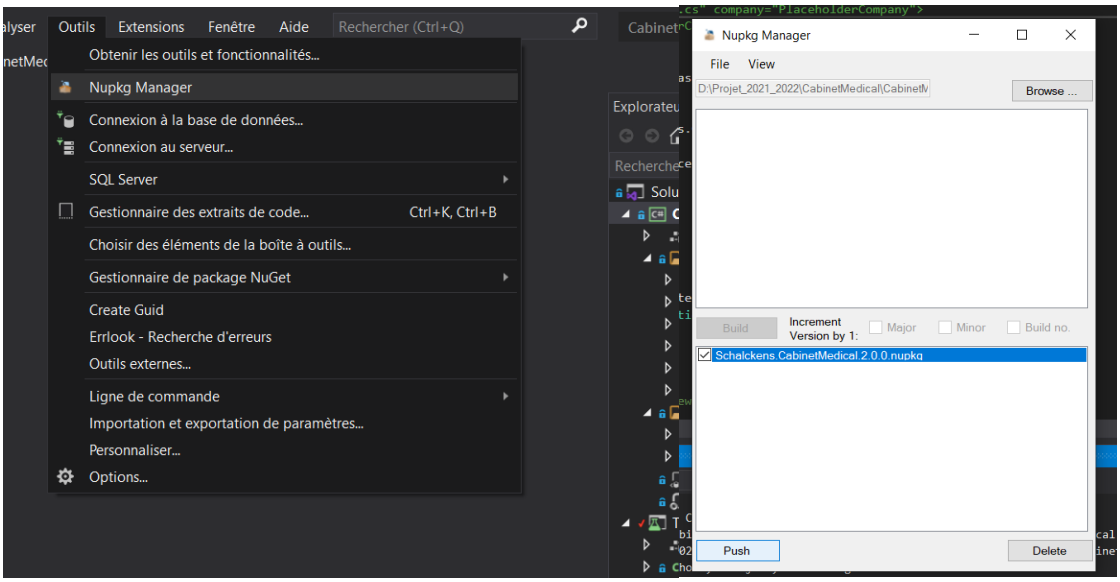
2. Mise à jour du package NuGet une fois le projet re-compressé et la version changée : TUTORIEL
    a. Installation de de l'extension Nupkg Manager : Extension → Gérer les extensions…



    b. Création d'une API keys sur NuGet.org.

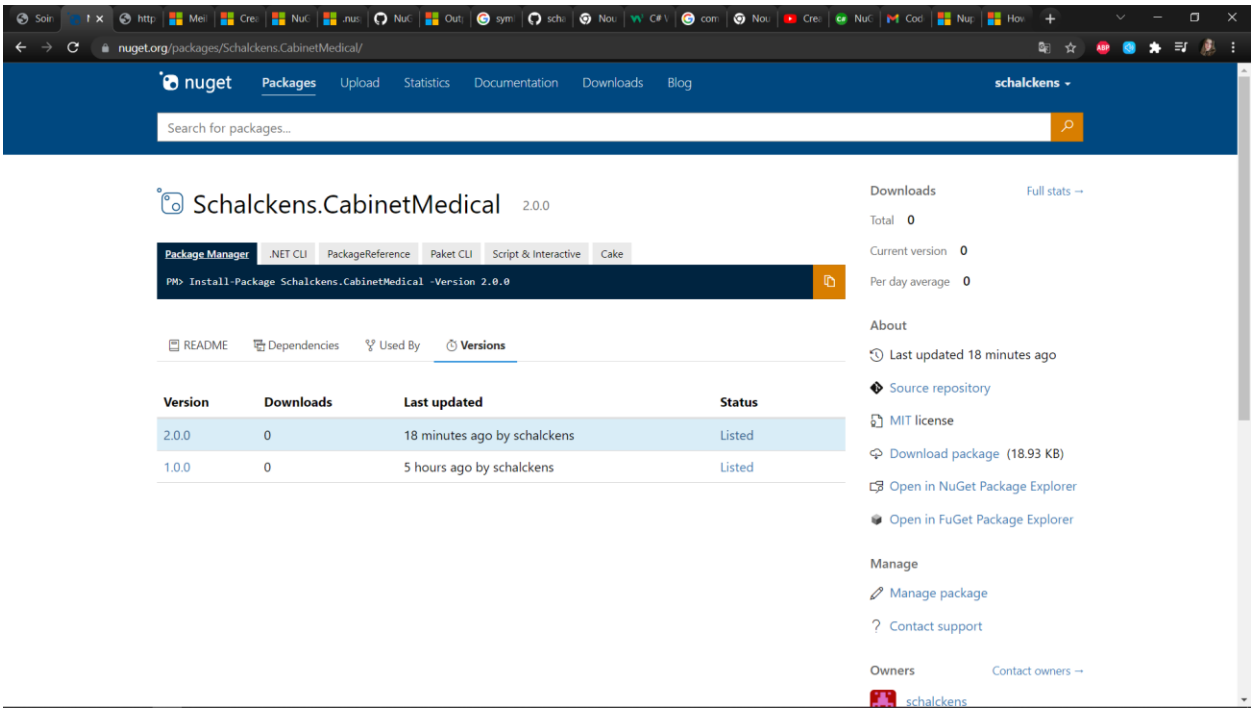


c. Utilisation de Nupkg Manager pour push la nouvelle version sur NuGet.org.



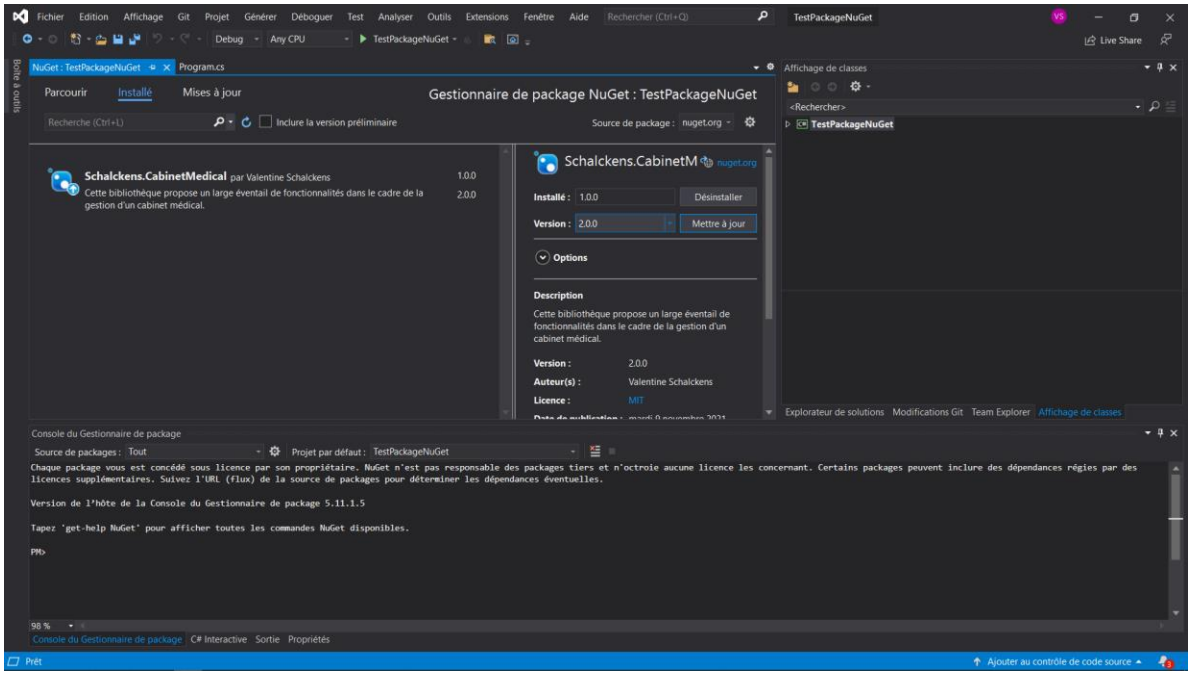
En appuyant sur push la première fois il va vous demander de renseigner l'API keys que vous avez créé au préalable et le serveur NuGet : https://api.nuget.org/v3/index.json.

Et voilà ! Après validation votre package NuGet est passé à la version supérieure !



PARTIE III: MISE EN OEUVRE DE LA NOUVELLE FONCTIONNALITÉ

1. Mettre à jour le package Schalckens.CabinetMédical :



2. Et voici son utilisation :