Rapport PPE 2 – Gestion Comptes Rendus

Jacques Bourbonnais

Sommaire

BTS Services informatiques aux organisations **SESSION 2024**

ANNEXE 9-1-B : Fiche descriptive de réalisation professionnelle (recto)

Épreuve E5 - Conception et développement d'applications (option SLAM)

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE	**N° réalisation : 02**
Nom, prénom : Bourbonnais Jacques	**N° candidat : 02345657150**
Épreuve ponctuelle ☒ **Contrôle en cours de formation** ☐	**Date :** 13 / 06 / 2024

Organisation support de la réalisation professionnelle
Laboratoire GSB

Intitulé de la réalisation professionnelle
PROJET Service Web

Période de réalisation : 04/24 ..... **Lieu :** EPSI Lyon .....
Modalité : ☒ **Seul(e)** ☐ **En équipe**

Compétences travaillées

☒ Concevoir et développer une solution applicative

☒ Assurer la maintenance corrective ou évolutive d'une solution applicative

☒ Gérer les données

Conditions de réalisation[1] (ressources fournies, résultats attendus)

- Ressources fournies : description du contexte, expression des besoins, exemple de page
- Résultats attendus : Réalisation d'une solution Frontend web MVC qui utilise les API développées lors de la première réalisation, sécurisation de certaines pages via un formulaire d'authentification pseudo/mot de passe (informations stockées sur une base de données locale), format de sortie json

Description des ressources documentaires, matérielles et logicielles utilisées[2]

- SGBD : SQL Server et SQL Management Studio
- Environnement de développement : Visual Studio
- Bibliothèque de développement : .NET
- Langages : HTML, CSS, Javascript, C#, Jquery
- Gestion de version : Github
- Tests des comportement API : Tests unitaires et tests d'intégrations manuels via navigateur web

Modalités d'accès aux productions[3] et à leur documentation[4]

- GitHub

[1] En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

[2] Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

[3] Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.* ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

[4] Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

Reverse Engineering – gestion des comptes rendus (Contexte GSB)

L'application CR est développée sous Access avec des Macro Excel en VBScript

Le but de cette réalisation est de de faire évoluer cette application en la récréant en mode web MVC

- Définition du besoin en examinant l'ancienne application
- Reverse conception et Migration de la BDD Access vers SQL Server (SSMA)
- MCD
- Création d'une application web MVC en https avec Visual Studio 2022
- Utilisation de bootstrap pour implémenter le pattern Design Responsive
- Mise en place d'un système d'authentification basé sur une gestion de compte individuels (Service Identity de Microsoft)
- Création d'une couche d'accès aux données (ORM Object Relationnal Mapping) basé sur les Entity Framework
- Création des objets model
- Développement des contrôleurs et des vues html
- Mise en place du HTTPS via un certificat auto signé
- Éxecution via IIS (serveur web intégré à Visual Studio)
- Gestion de versions
- Publication GitHub

Définition du besoin en examinant l'ancienne application

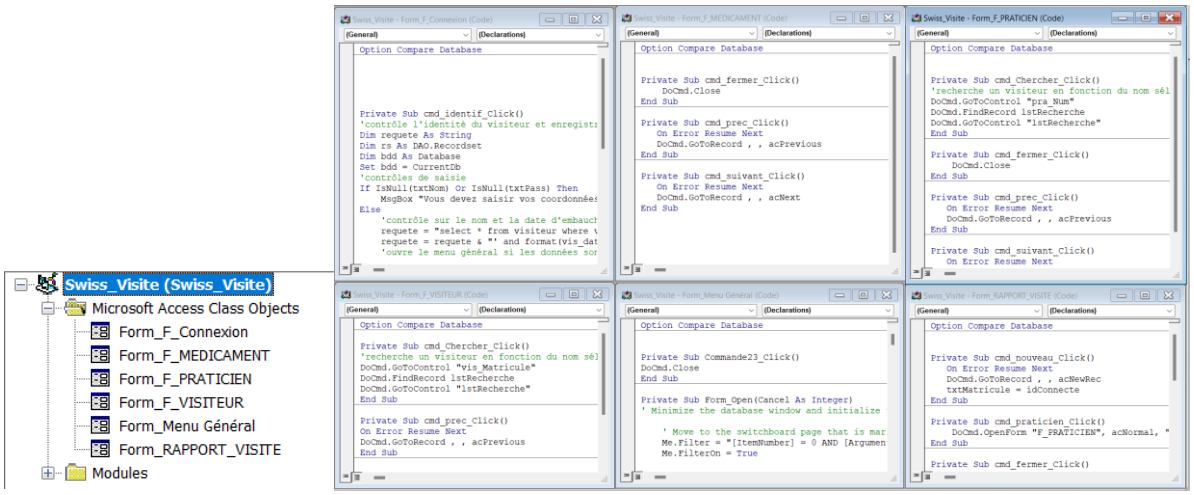
Le laboratoire Galaxy Swiss Bourdin (GSB) désire mettre à disposition des visiteurs médicaux une application Web permettant de centraliser les comptes-rendus de visite. Cette base d'information sera utilisée à des fins d'élaboration de la démarche de communication auprès des praticiens et donnera une vision individuelle et synthétique de l'activité de représentation.

Pour permettre une aide au renseignement des rapports, l'application fournira une description des produits du laboratoire, les coordonnées précises des praticiens et des informations détaillées les concernant.

L'objectif de ce projet est de moderniser l'application en la reconstruisant en tant qu'application web MVC. À l'origine développée sur Access avec des macros Excel en Visual Basic Script, elle sera repensée pour une architecture web plus dynamique et plus moderne.

Voici à quoi ressemblait l'application sur Access :

Et le script Visual Basic des macros derrière qui permet une telle interface et utilisation de l'application :



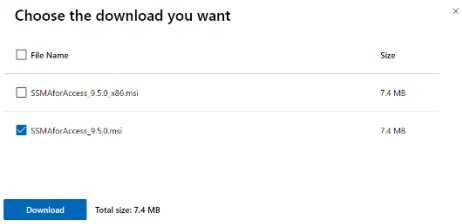
Voici une représentation visuelle approximative de la nouvelle application web conçue pour gérer les comptes rendus :

Reverse conception et migration de la BDD Access vers SQL Server (SSMA)

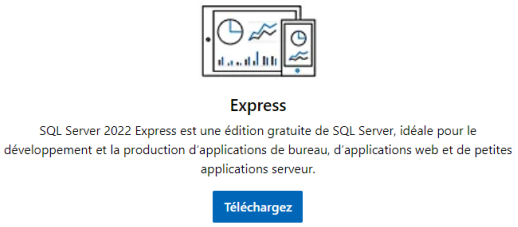
Pour récupérer la base de données de l'application Access, nous devons utiliser le logiciel SSMA (SQL Server Migration Assistant) pour l'extraire vers une base de données que nous créerons dans le logiciel SSMS (SQL Server Management Studio).

Pour installer SSMA et SSMS :

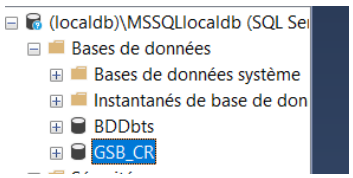
SSMA : https://www.microsoft.com/en-us/download/details.aspx?id=54255



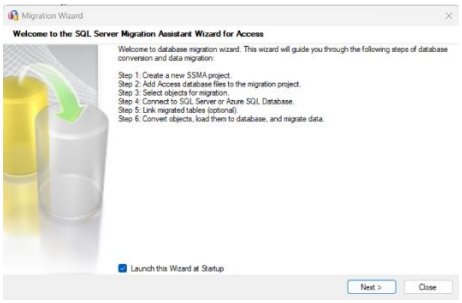
SSMS : https://learn.microsoft.com/fr-fr/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16



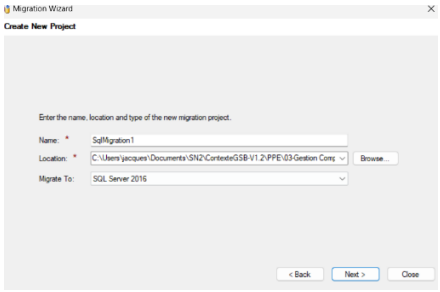
Dans SSMS, il faut créer une base de données qui pourra stocker la migration de la base de données de Access.



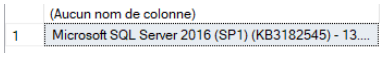
Puis ouvrir SSMA et créer un nouveau projet en cliquant sur « Next ».



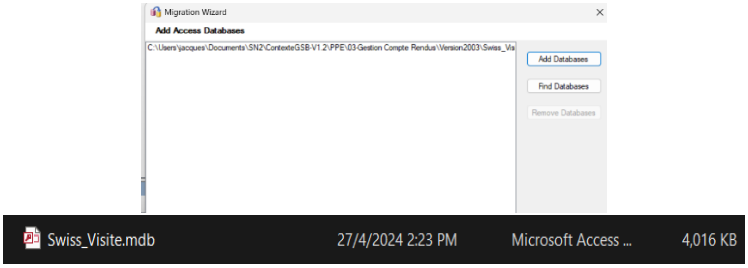
Il faut s'assure de mettre la bonne version de SQL Server, en général c'est la version SQL Server 2022 quand on est sur SSMS 2022, mais pour ma part j'utilise SSMS 2018 donc j'ai la version SQL Server 2016.



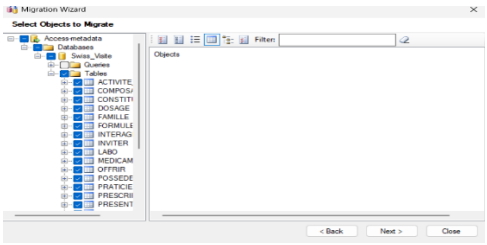
Pour savoir quelle version vous avez, il faut aller dans SSMS, puis nouvelle requête et écrire « select @@VERSION » puis exécuter.



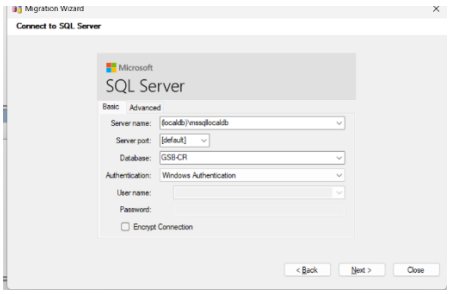
Ensuite il faut ajouter la BDD Access.



Sélectionner toutes les tables.



Et ensuite se connecter à la base de données SQL qu'on a créé précédemment sur SSMS.

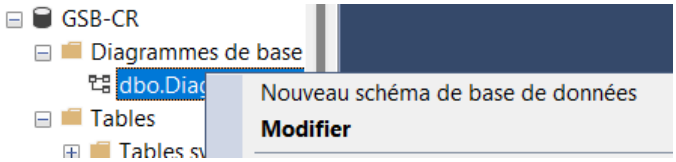


Ensuite il faut attendre que SSMA Convert, Load et Migrate.

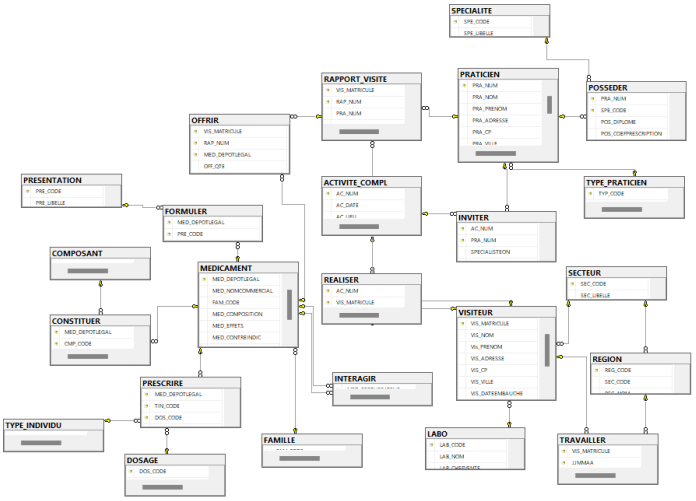
Une fois fini, dans la base de données sur SSMS, toutes les tables seront rajoutées.

MCD

Pour le MCD, il y a une option spécifique à cela.

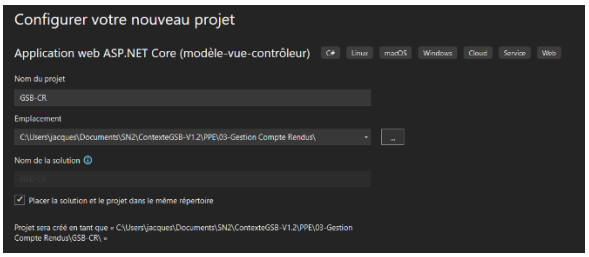


Appuyer sur « Nouveau schéma » et sélectionner toutes les tables et générer le MCD.

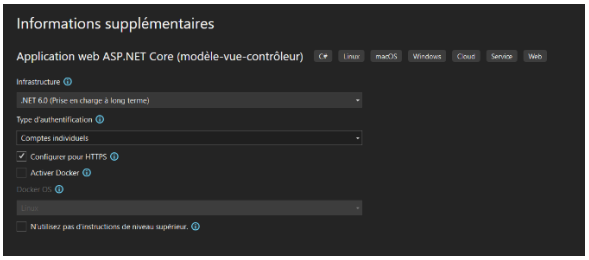


Création d'une application web MVC en https avec Visual Studio 2022

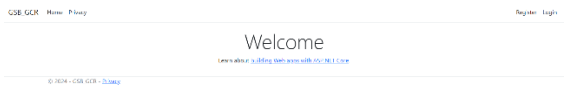
Sur Visual Studio 2022, il faut sélectionner un projet ASP.NET Core (MVC).



Lors des paramètres, il faut bien mettre la version .NET 6.0 pour l'installation des paquets NuGets. Il faut mettre en place un système d'authentification qui sera basé sur une gestion de compte individuels (Service Identity de Microsoft). Et aussi configurer le certificat HTTPS (version sécurisée du protocole HHTP) en cochant la case.

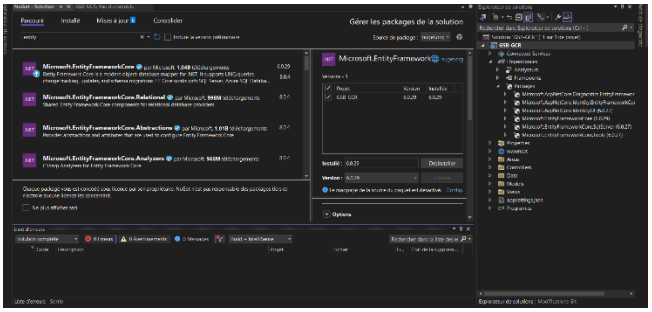


Voici à quoi ressemble l'application au lancement, l'application est de base en anglais, mais dans les fichiers html, on peut la modifier entièrement pour qu'elle soit personnalisé à ma façon et en français.

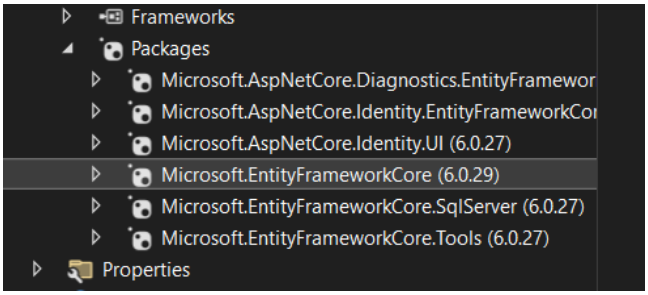


Création d'une couche d'accès aux données (ORM Object Relationnal Mapping) basé sur les Entity Framework

Pour cette partie, il faut ajouter les packages NuGet, voici un des packages qu'il faut installer. (Comme nous avons la version .NET 6.0 il faut télécharger la dernière version 6.0 des différents packages).



Et voici les différents packages qu'on aura installé.



Création des objets model

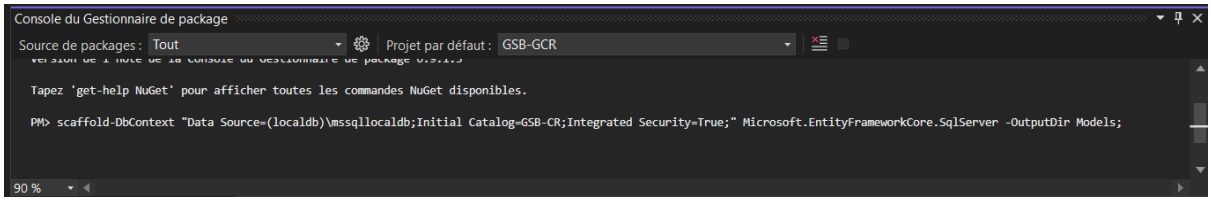
L'installation des packages nous permet donc de créer les différents objets model class. Pour cela, il faut ouvrir la Console du Gestionnaire de package dans 'onglet « Outils » de Visual Studio 2022.

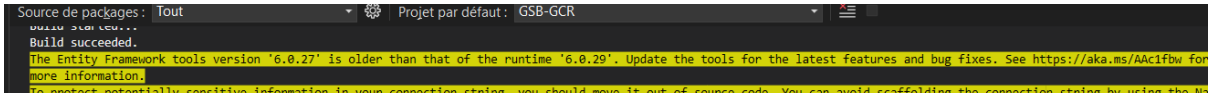
Puis il faut insérer cette commande :

scaffold-DbContext "Data Source=(localdb)\mssqllocaldb;Initial Catalog=NomBDD;Integrated Security=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models;

Pour le NomBDD, je dois mettre la mienne donc « GSB-CR », donc dans la console pour ma part, je dois mettre :

scaffold-DbContext "Data Source=(localdb)\mssqllocaldb;Initial Catalog=GSB-CR;Integrated Security=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models;

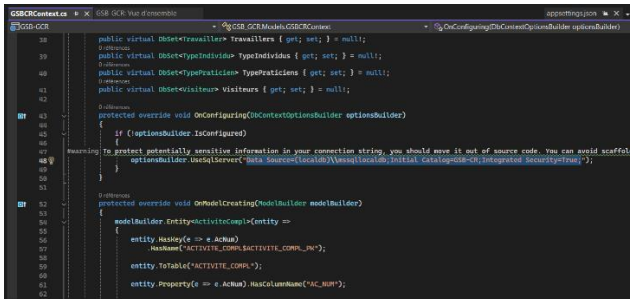




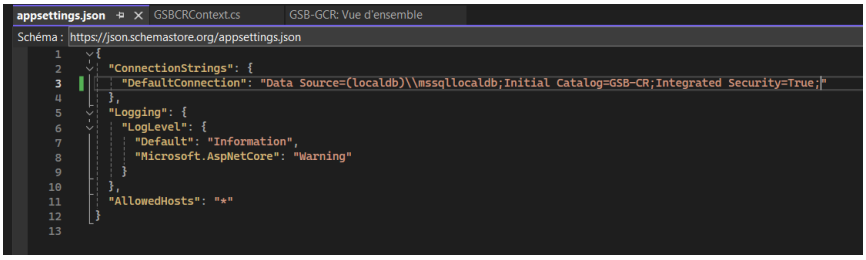
Si c'est écrit « Build succeeded », cela veut dire que tous vos objets class model de la base de données se sont créés.

Juste après cela, il faut ajouter la chaine de connexion dans le fichier « appsettings.json » pour établir une connexion à la base de données. Pour faire cela, il faut aller dans l'objet model class Context.

Et chercher et copier la chaine de connexion vers la ligne 50.

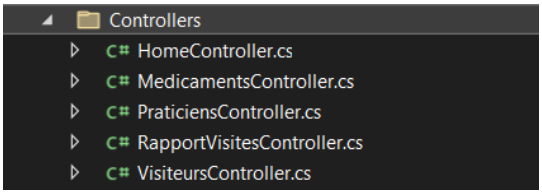


Et remplacer la chaine de connexion initial par la chaine de connexion de la base de données.



Développement des contrôleurs et des vues html

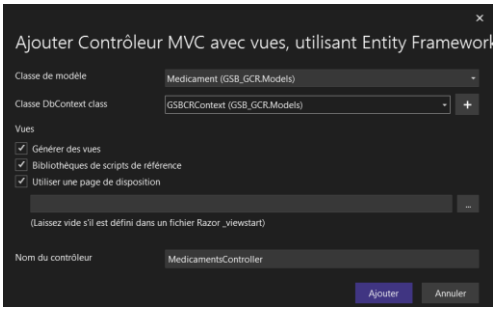
Pour le projet, nous avons créés 4 contrôleurs Medicaments, Praticiens, RapportVisites et Visiteurs.



Je vais montrer la création d'un seul contrôleur parce que c'est répétitif et il faut faire la même chose pour les 4. Il faut d'abord faire un clic droit sur le dossier « Controllers » et Ajouter.

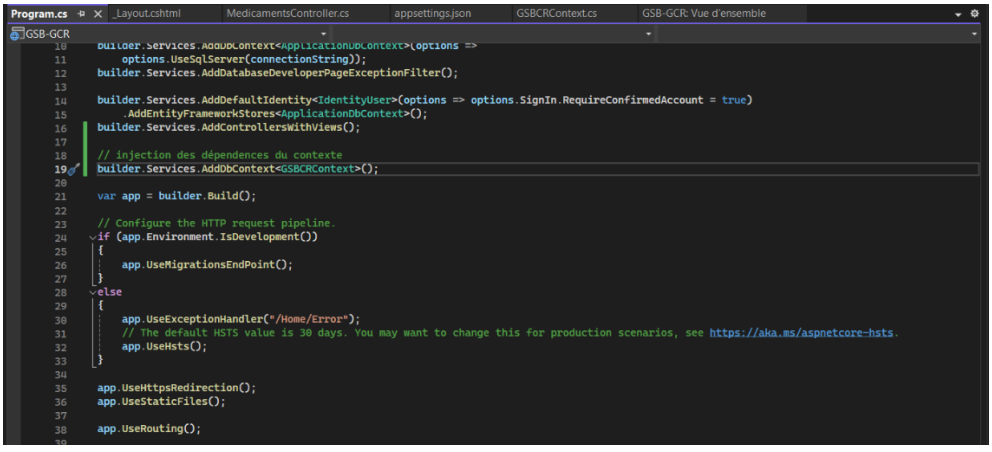
Puis il faut sélectionner contrôleur MVC.

Ensuite sélectionner la classe de modèle Medicament et le DbContext qui est pour moi GSBCR.



Il faut refaire ceci 3 fois mais pas avec la classe Medicament mais avec les classes Praticiens, RapportVisites et Visiteurs.

Ensuite, afin de pouvoir effectuer des opérations telles que la lecture, l'écriture et la modification de données, il faut que le Service Web (l'application) puisse interagir avec la base de données, donc pour cela, il faut injecter les dépendances dans le context du modèle GSBCR.



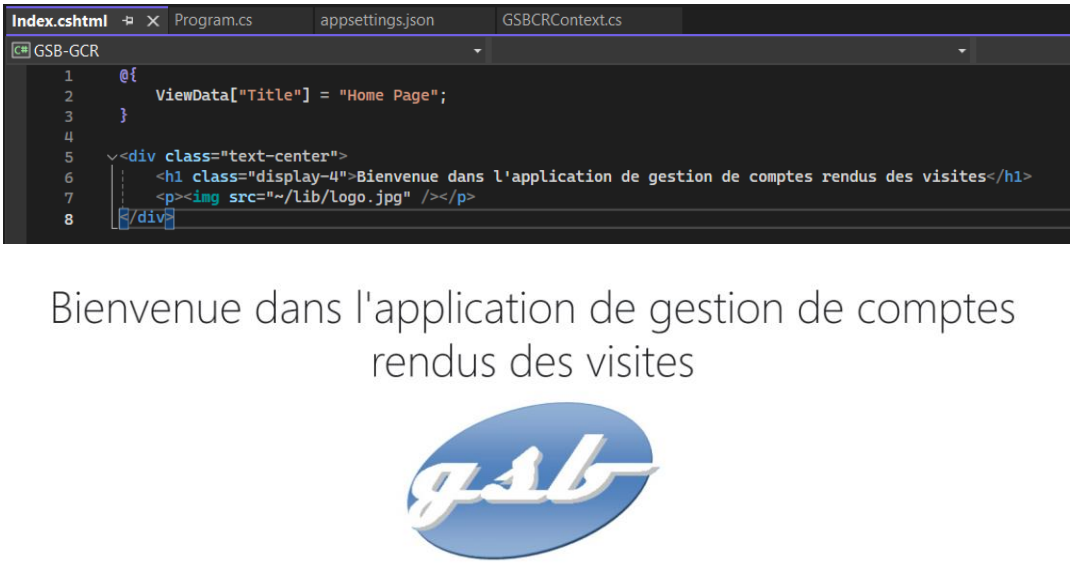
Builder.Services.AddDbContext<GSBCRContext>() ;

Cette ligne nous permet, par exemple, d'accéder à la liste de médicaments, de pouvoir ajouter un nouveau médicament, d'en modifier un, de regarder les détails d'un et d'en supprimer un.

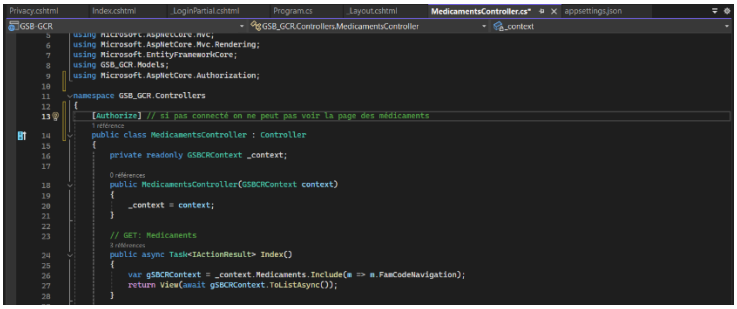
Et maintenant que tout est fait, il manque plus qu'à modifier le front-end (vues html). C'est des actions assez répétitives mais voilà quelques modifications/ajouts que j'ai fait à mon application.

J'ai ajouté dans mon header une barre de navigation.

La page d'accueil.

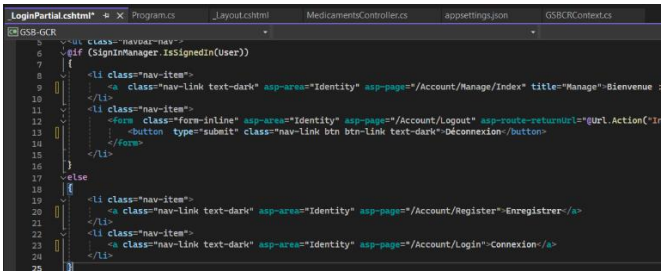


Sur cette application, il y un système d'authentification donc j'ai ajouté une fonctionnalité qui permet de voir la liste des médecins seulement si on est connecté, pour cela il suffit tout simplement d'ajouter « [Authorize] » dans le contrôleur Medicaments.

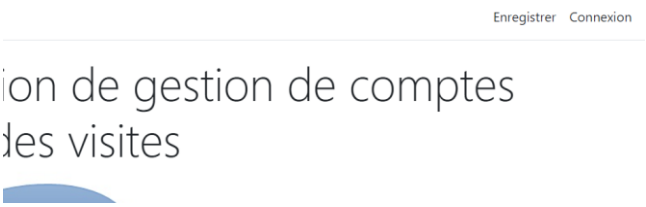


```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using GSB_GCR.Models;
using Microsoft.AspNetCore.Authorization;

namespace GSB_GCR.Controllers
{
    [Authorize] // si pas connecté on ne peut pas voir la page des médicaments
    public class MedicamentsController : Controller
    {
        private readonly GSBCRContext _context;

        public MedicamentsController(GSBCRContext context)
        {
            _context = context;
        }

        // GET: Medicaments
        public async Task<IActionResult> Index()
        {
            var gSBCRContext = _context.Medicaments.Include(m => m.FamCodeNavigation);
            return View(await gSBCRContext.ToListAsync());
        }
```

Et pour le reste du front-end, il faut savoir s'y trouver dans les différents vues html pour modifier la langue (Anglais → Français). Comme par exemple, le mot Register → Enregistrer et Login → Connexion.

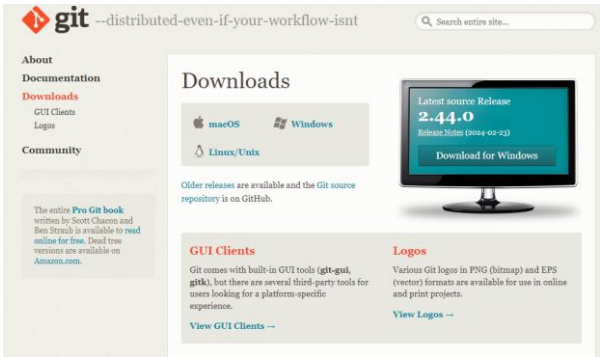


```
@if (SignInManager.IsSignedIn(User))
{
    <li class="nav-item">
        <a  class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Bienvenue
    </li>
    <li class="nav-item">
        <form  class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("
            <button  type="submit" class="nav-link btn btn-link text-dark">Déconnexion</button>
        </form>
    </li>
}
else
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Enregistrer</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Login">Connexion</a>
    </li>
}
```

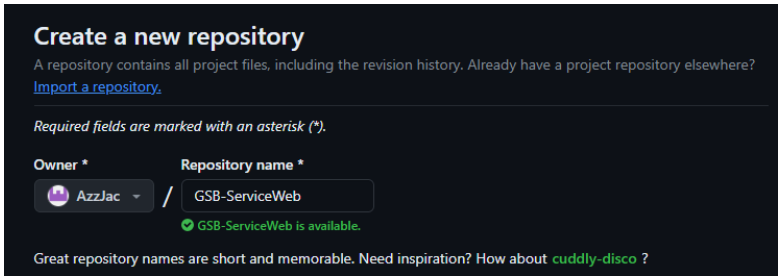


Publication GitHub

Pour déposer le projet sur GitHub, nous devons d'abord commencer par installer Git.



Ensuite, il faut se rendre sur notre compte GitHub et créer un nouveau repository dans lequel nous allons déposer le projet.



Ensuite il faut faire un clic droit sur le dossier contenant tous les fichiers du projet.

Et sélectionner « Afficher plus d'options » → « Ouvrir Git Bash ».

Puis une fenêtre va s'ouvrir et il faudra initialiser le dossier comme étant un dossier git en tapant la commande « git init ».

```
jacques@MSI MINGW64 ~/Documents/SN2/ContexteGSB-V1.2/PPE/03-Gestion Compte Rendu
s
$ git init
Initialized empty Git repository in C:/Users/jacques/Documents/SN2/ContexteGSB-V
1.2/PPE/03-Gestion Compte Rendus/.git/
```

Puis il faut se connecter son dépôt local avec le dépôt GitHub.

```
jacques@MSI MINGW64 ~/Documents/SN2/ContexteGSB-V1.2/PPE/03-Gestion Compte Rendu
s (master)
$ git remote add origin https://github.com/AzzJac/GSB-ServiceWeb.git
```

Ensuite, si vous avez créé un dossier exprès pour le dépôt du projet dans GitHub, il faut ajouter les fichiers dans le dossier et taper la commande « git add -A », pour ma part les fichiers sont déjà dans le dossier car j'ai utilisé le même dossier.

On va effectuer notre premier commit, pour cela, il faut écrire la commande « git commit -m [commentaire du commit] ».

```
jacques@MSI MINGW64 ~/Documents/SN2/ContexteGSB-V1.2/PPE/03-Gestion Compte Rendu
s (master)
$ git commit -m "first push"
[master (root-commit) 96ed483] first push
 466 files changed, 101488 insertions(+)
 create mode 100644 CR/cherchePraticien.php
 create mode 100644 CR/classConnexion.php
```

Une fois le commit fait, on a plus qu'à envoyer les fichiers sur GitHub en faisant un push.

```
jacques@MSI MINGW64 ~/Documents/SN2/ContexteGSB-V1.2/PPE/03-Gestion Compte Rendu
s (master)
$ git push origin master
Enumerating objects: 605, done.
Counting objects: 100% (605/605), done.
Delta compression using up to 20 threads
Compressing objects: 100% (554/554), done.
Writing objects: 100% (605/605), 34.65 MiB | 6.80 MiB/s, done.
Total 605 (delta 195), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (195/195), done.
To https://github.com/AzzJac/GSB-ServiceWeb.git
 * [new branch]      master -> master
```

Une fois terminé, on peut aller vérifier que le projet existe sur GitHub.