PPE : GSB gestion de la clôture

APPLICATION C# : SERVICE WINDOWS

Ressources et documents du projet

- Fiche de situation professionelle (PDF)
- Archive contenant l'exécutable et instructions d'installation (sur GitHub)
- Code source (GitHub repository)
- Base de données (fichier .sql et instructions d'installation en local)
- Documentation technique
- Compte rendu du projet (PDF)

Ressources fournies :

- Description des tâches à réaliser (PDF)
- Contexte de l'entreprise GSB (fichier Word)
- Cahier des charges (fichier Word)

Compétences associées

- A1.1.1 , Analyse du cahier des charges d'un service à produire
- A1.2.4 , Détermination des tests nécessaires à la validation d'un service
- A1.3.4 , Déploiement d'un service
- A1.4.1 , Participation à un projet
- A4.1.8 , Réalisation des tests nécessaires à la validation d'éléments adaptés ou développés
- A4.1.9 , Rédaction d'une documentation technique

Présentation du projet

Contexte de la situation professionelle et description du besoin

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy et le conglomérat européen Swiss Bourdin. Après avoir fait évoluer l'application web du laboratoire, servant pour la gestion de frais des visiteurs médicaux par les comptables, il faut intégrer un service mettant à jour l'état des fiches de manière automatique.

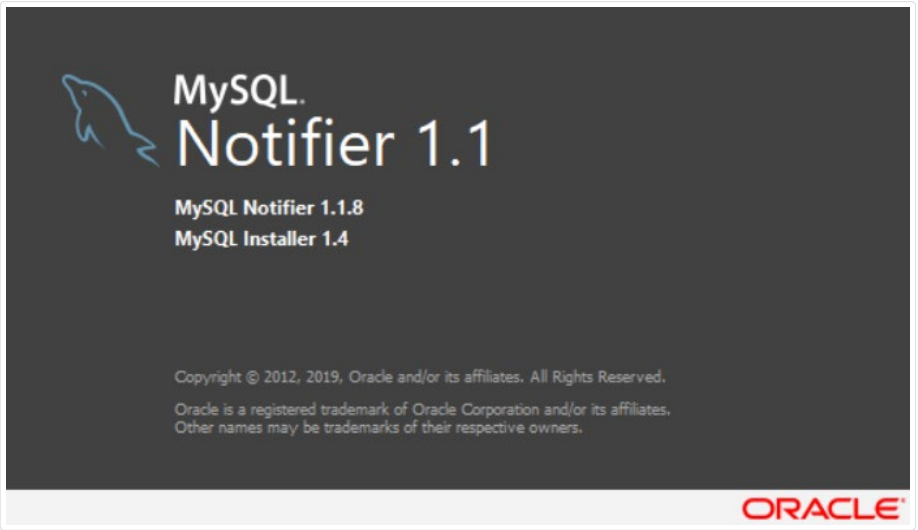
Description du besoin

Le cahier des charges de l'application Frais GSB stipule que la fiche d'un visiteur est clôturée au dernier jour du mois. Cette clôture sera réalisée par l'application selon la modalité suivante : au début de la campagne de validation des fiches par le service comptable, un script est lancé qui clôture toutes les fiches non clôturées du mois qui va être traité. D'autre part, il est dit que la mise en paiement est faite au 20 du mois suivant la saisie par les visiteurs.

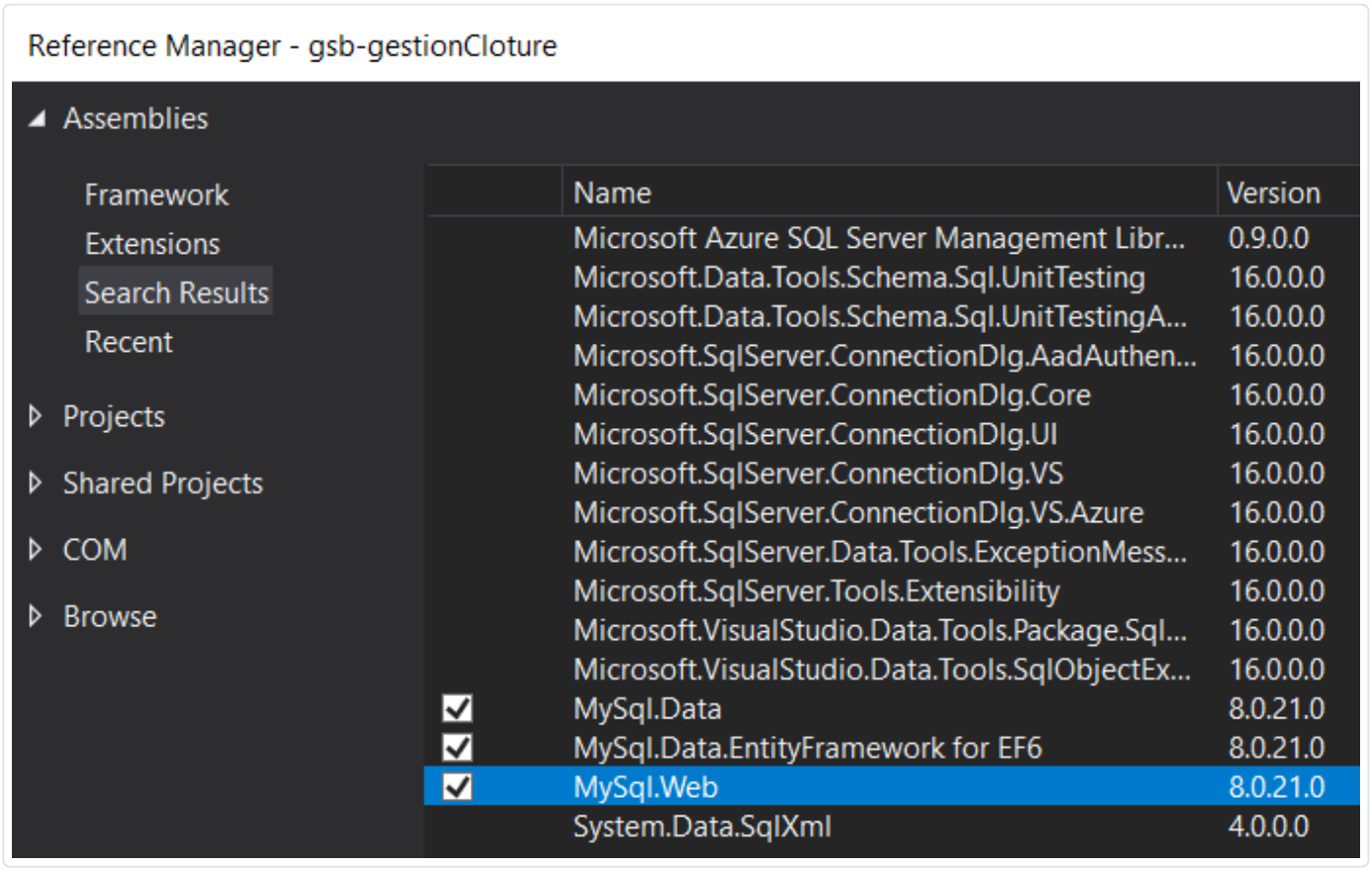
Il faut répondre à ces deux objectifs en développant une application C# avec VS.Net, puis en créant un service s'exécutant à intervalle régulier en tâche de fond. Cette application va devoir permettre, au début de la campagne de validation, c'est-à-dire à partir du 1er jour du mois N, la clôture de toutes les fiches créées le mois N-1. Elle permettra, d'autre part, à partir du 20e jour du mois N la mise en remboursement des fiches créées le mois N-1.

Préparation du projet

Création d'un nouveau projet C# de type application console, sous Visual Studio.
Installation de MySQL Installer, qui permet de bénéficier des frameworks utiles pour la connexion à la base données, puis ajout aux références du projet.



Installation de MySQL Installer



Ajout de MySQL aux références du projet

Classe d'accès aux données

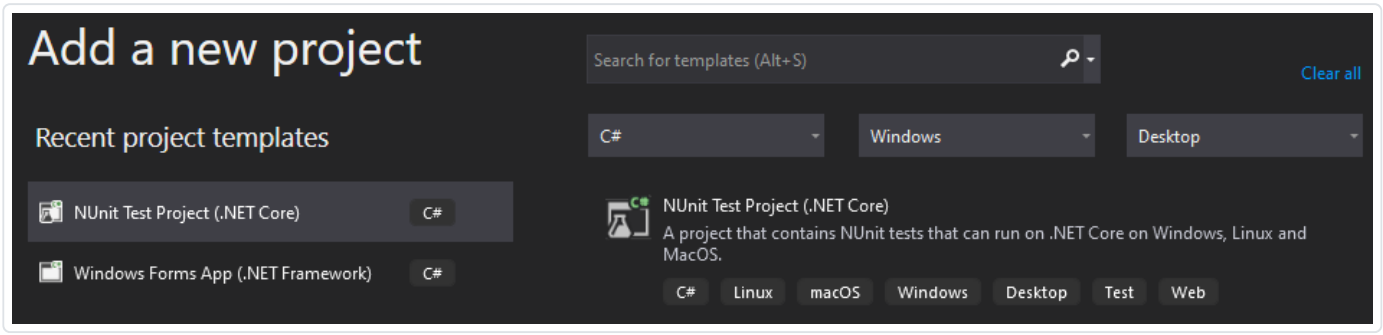
Création de la classe DataAccess qui est utile à l'accès et la modification des données d'un base MySQL.
Le constructeur de cette classe instancie la classe MySqlConnection, permettant de gérer une connexion à un serveur MySQL. La classe DataAccess contient des méthodes qui permettent le lecture, la modification, la mise à jour et la suppression de données.
Cette classe est réutilisable, elle a été conçue pour pouvoir être importée dans d'autres projets en relation avec une base de données MySQL.

Classe de gestion des dates

Création de la classe abstraite DateGestion, qui comporte des méthodes statiques, ayant pour but de retourner des dates utiles à la gestion de la clôture des fiches de frais.
Classe optimisée de manière à éviter les répétitions de code, par le biais d'une méthode privée "calculDate". Cette méthode calcule la date, par rapport à l'objet DateTime et le nombre passés en paramètre.

Réalisation de tests unitaires

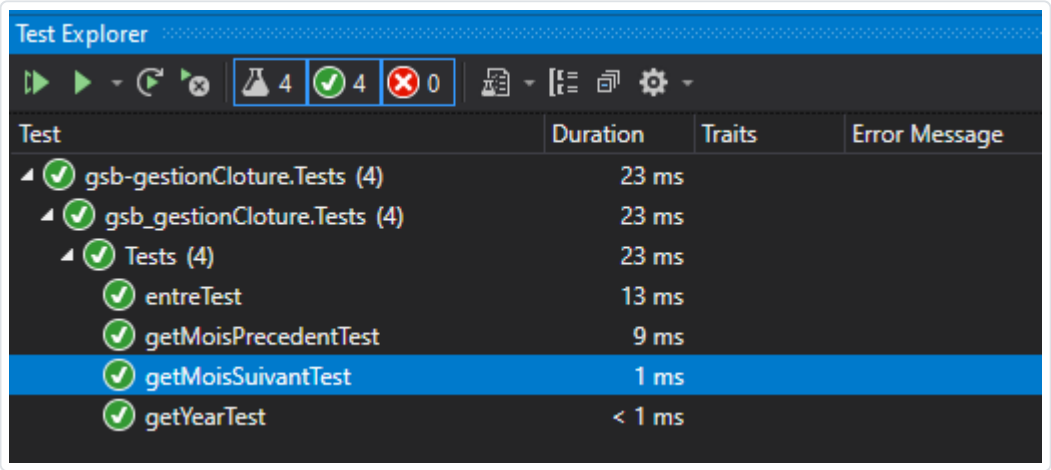
Ajout d'un nouveau projet NUnit à la solution, pour la réalisation de tests unitaires sur les méthodes de la classe DateGestion.
La plupart des tests comportent plusieurs assertions afin de pourvoir tester différents cas de figure.



Création du projet NUnit

```
public class Tests
{
    [SetUp]
    0 references
    public void Setup()
    {
    }

    [Test]
    0 references
    public void getMoisPrecedentTest()
    {
        Assert.AreEqual("08", DateGestion.getMoisPrecedent());
        Assert.AreEqual("08", DateGestion.getMoisPrecedent(DateTime.Today));
    }

    [Test]
    0 references
    public void getMoisSuivantTest()
    {
        Assert.AreEqual("10", DateGestion.getMoisSuivant());
        Assert.AreEqual("10", DateGestion.getMoisSuivant(DateTime.Today));
    }

    [Test]
    0 references
    public void entreTest()
    {
        Assert.IsTrue(DateGestion.entre(01, 20));
        Assert.IsTrue(DateGestion.entre(01, 20, DateTime.Today));
    }

    [Test]
    0 references
    public void getYearTest()
    {
        Assert.AreEqual("2020", DateGestion.getYear(DateTime.Today));
    }

}
```

Création des différents tests sur la classe DateGestion



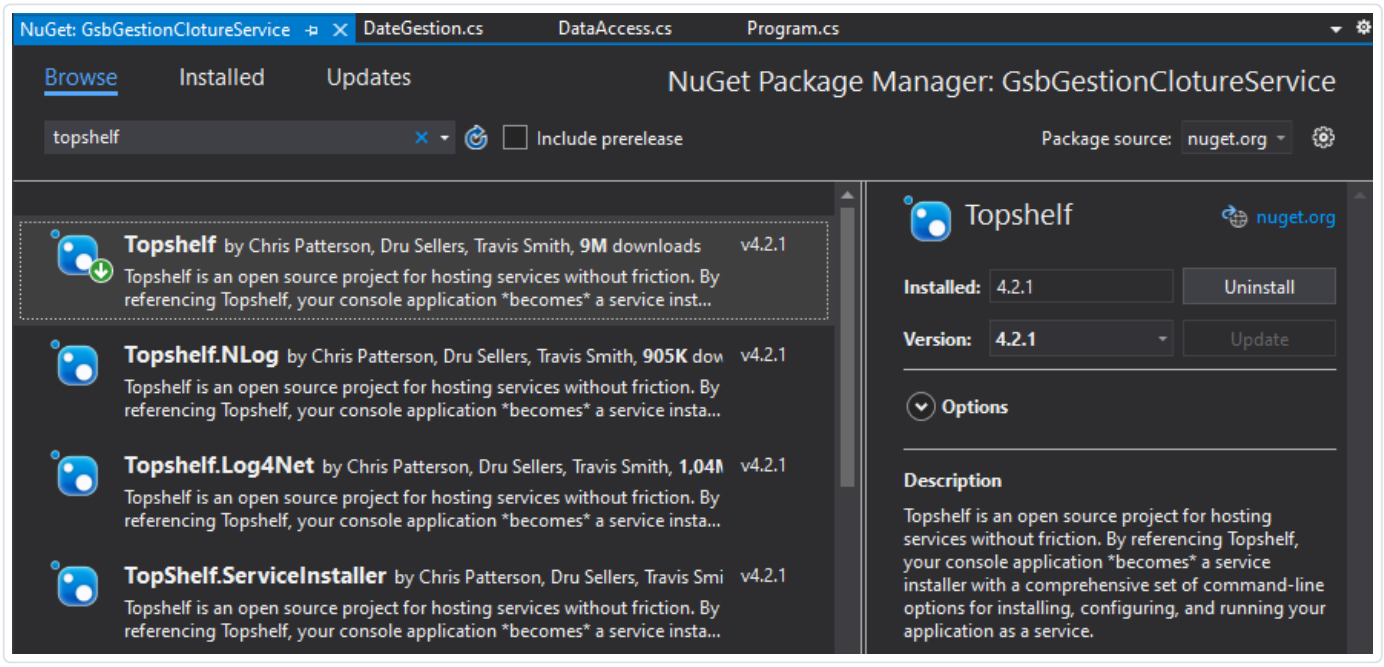
Résultat des tests

Classe de gestion du Timer et des requêtes SQL

Elle sert d'intermédiaire entre les classes d'accès aux données / de gestion des dates, et le "Main" qui est la fonction principale du programme. La classe GestionCloture instancie la classe "Timer" permettant de définir un intervalle de temps, qui lance automatiquement une méthode, et qui est donc utile dans le cadre du déploiement d'un service.
Lorsque l'intervalle de temps défini est atteint, le "Timer" exécute une méthode qui à pour but de se connecter à la base de données MySQL et, selon la date du jour, envoie les requêtes mettant à jour l'état de fiches de frais.

Création et installation du service Windows

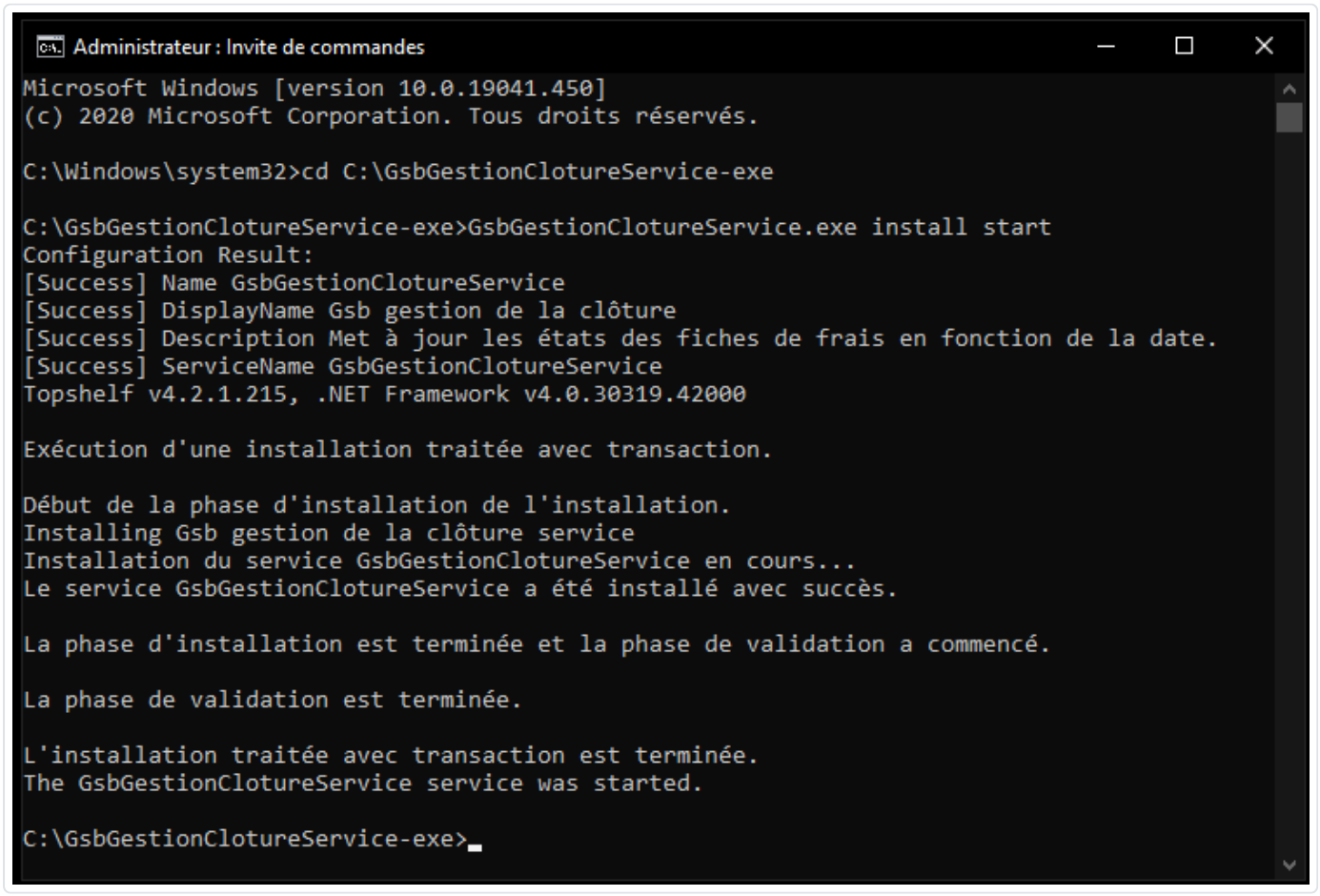
Installation du package Topshelf permettant le déploiement d'un service Windows, via la classe "HostFatory".
Package installé via le gestionnaire de package NuGet, intégré à Visual Studio.



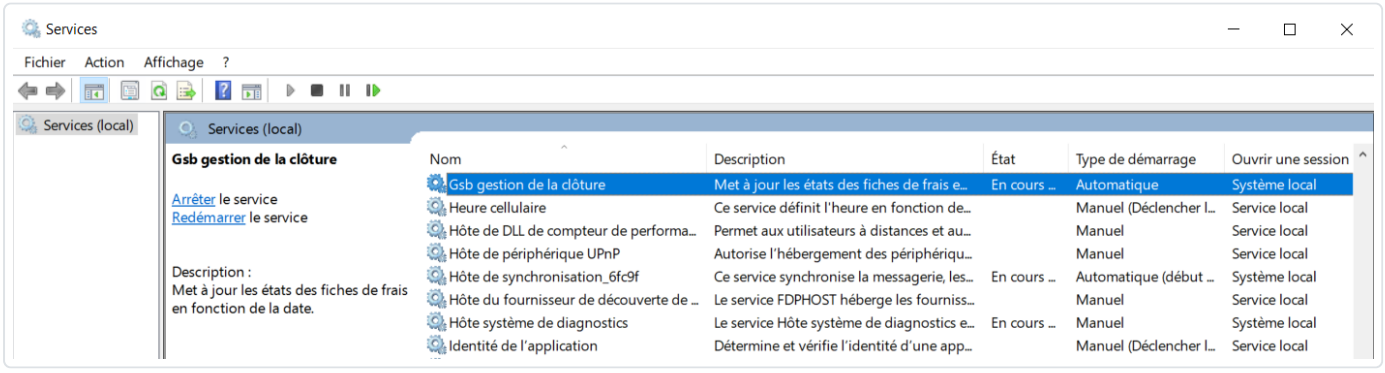
Installation du package Topshelf

Codage du Main (fonction principale du programme), qui instancie la classe HostFatory, fait appel à la classe GestionCloture et défini les différents paramètres du service Windows.
Création de l'exécutable et installation du service en ligne de commande, via la commande suivante :

```
GsbGestionClotureService.exe install start
```



Installation du service Windows en ligne de commande



Service en cours d'exécution

Conclusion

Ce projet m'a permis de découvrir la création d'un service Windows et de me familiariser avec la création de classes et de tests unitaires.
La création et le déploiement d'un tel service, automatisant des tâches répétitives, peut permettre à une entreprise de gagner en productivité.