Manier Gaetan

42 Rue Saint Firmin

59190 Morbecque

Mail : [gaetan.manier@sfr.fr](mailto:gaetan.manier@sfr.fr)

Dossier Projet de VAE

Master informatique parcours

Ingénierie des Systèmes Informatiques Distribués

Jury du 29 juin 2018

Université du Littoral Côte d’Opale

[1 Introduction 5](#_Toc502315988)

[2 Le cahier des charges 6](#_Toc502315989)

[3 Construction du projet 8](#_Toc502315990)

[3.1 Choix des outils 8](#_Toc502315991)

[3.2 Découpage du projet sous VS2013 8](#_Toc502315992)

[4 Fonctionnalités 10](#_Toc502315993)

[4.1 Introduction 10](#_Toc502315994)

[4.2 L’application WEB 10](#_Toc502315995)

[4.2.1 La page d’accueil 10](#_Toc502315996)

[4.2.2 La gestion des accès utilisateur 11](#_Toc502315997)

[4.2.3 La page « Contact » 11](#_Toc502315998)

[4.2.4 La page « à propos de » 12](#_Toc502315999)

[4.2.5 La page « Liste des produits» 12](#_Toc502316000)

[4.2.6 La page « Recherche un produit» 14](#_Toc502316001)

[4.2.7 La page « Bilans bobine» 15](#_Toc502316002)

[4.2.8 En conclusion 15](#_Toc502316003)

[4.3 Le Data Warehouse 16](#_Toc502316004)

[4.4 L’application mobile 16](#_Toc502316005)

[5 Architecture du logiciel 17](#_Toc502316006)

[5.1 Découpage du projet sous VS2013 17](#_Toc502316007)

[5.2 Architecture de l’application web 18](#_Toc502316008)

[5.3 Le serveur Web 19](#_Toc502316009)

[5.4 En conclusion 20](#_Toc502316010)

[6 Architecture de la base de données 21](#_Toc502316011)

[6.1 Configuration du serveur de base de données 21](#_Toc502316012)

[6.1.1 Compte pour l’application Web 21](#_Toc502316013)

[6.1.2 Compte pour le SSAS 22](#_Toc502316014)

[6.2 Introduction aux Data Warehouse 22](#_Toc502316015)

[6.3 Choix du schéma 23](#_Toc502316016)

[6.4 Alimentation du Data Warehouse avec SSIS 24](#_Toc502316017)

[6.4.1 Configuration de l’ETL 24](#_Toc502316018)

[6.4.2 Déploiement de l’ETL 28](#_Toc502316019)

[6.4.3 Exécution de l’ETL 30](#_Toc502316020)

[6.4.4 Pour aller plus loin 32](#_Toc502316021)

[6.5 Création et gestion du cube avec SSAS 32](#_Toc502316022)

[6.5.1 Connexion à la base de données 32](#_Toc502316023)

[6.5.2 Vue des sources de données 33](#_Toc502316024)

[6.5.3 Création des dimensions 34](#_Toc502316025)

[6.5.4 Création du cube 35](#_Toc502316026)

[6.5.5 Déploiement du cube 36](#_Toc502316027)

[6.5.6 Exécution du cube avec Excel 37](#_Toc502316028)

[6.5.7 Conclusion 39](#_Toc502316029)

[6.6 Création des rapports avec SSRS 39](#_Toc502316030)

[6.6.1 Création des rapports 39](#_Toc502316031)

[6.6.2 Configuration de Reporting Service 42](#_Toc502316032)

[6.6.3 Déploiement des rapports 43](#_Toc502316033)

[6.6.4 Consultation des rapports 44](#_Toc502316034)

[6.6.5 Conclusion 45](#_Toc502316035)

[7 La sécurité 46](#_Toc502316036)

[7.1 Les comptes utilisateurs 46](#_Toc502316037)

[7.2 Protection des pages web 46](#_Toc502316038)

[7.3 Gestion des droits 46](#_Toc502316039)

[7.4 Conclusion 47](#_Toc502316040)

[8 Qualité du code, gestion des versions et des fiches recettes 48](#_Toc502316041)

[8.1 Choix de la gestion des versions 48](#_Toc502316042)

[8.1.1 Copie et renommage de répertoires 48](#_Toc502316043)

[8.1.2 Utilitaire de gestion de code sources 48](#_Toc502316044)

[8.1.3 GitHub 49](#_Toc502316045)

[8.2 Les fiches de recettes 51](#_Toc502316046)

[8.2.1 Les fiches de recette 51](#_Toc502316047)

[8.2.2 Les tests unitaires 52](#_Toc502316048)

[8.2.3 Conclusion 52](#_Toc502316049)

[9 Conclusions 53](#_Toc502316050)

[10 Annexes 54](#_Toc502316051)

[10.1 Annexe 1 : Arborescence des vues 54](#_Toc502316052)

[10.2 Annexe 2 : Architecture globale 55](#_Toc502316053)

[10.3 Annexe 3 : Architecture de l’application 56](#_Toc502316054)

# Introduction

Ce document est mon rapport de projet de VAE pour le titre de Master informatique parcours Ingénierie des Systèmes Informatiques Distribués.

Mon projet consiste en la création d’une application informatique comme décrit dans le chapitre du cahier des charges.

Les chapitres décrits dans mon rapport vont nous amener à comprendre cette application de part :

* ses fonctionnalités
* son architecture logicielle
* son architecture de base de données
* sa sécurité
* la qualité de son code, des versions et des recettes

Pour chaque chapitre une conclusion donne un aperçu des autres outils existant sur le marché, des difficultés rencontrées ainsi que du temps passé pour réaliser chaque étape de ce projet.

Les concepts vus dans ce projet sont pour moi entièrement nouveaux, car ils ne répondent pas à ce jour aux standards utilisés dans ma société.

En fin de rapports vous trouverez la conclusion du projet ainsi que les différentes annexes.

# Le cahier des charges

L’application informatique devra répondre aux exigences suivantes : Modèle, Vue et Contrôleur.

Le langage de programmation sera de type JEE, mais pourra être également en C#, C++ ou DotNet suivant ce qui est utilisé dans l’entreprise mais devra dans tous les cas répondre à l’architecture MVC. L’utilisation des API Rest (get, delete, post) est demandée.

La base de données SQL devra être de type Data Warehouse et structurée comme telle sur le serveur. Le client aura une connexion web au serveur où il faudra gérer la sécurité de la connexion.

Le schéma de l’application sera le suivant :



La gestion du suivi des versions se fera avec l’un des outils suivants : git/sun, github ou bitbucket.

Au niveau du rapport et des attendus pour la soutenance orale, les points suivants seront demandés :

* Une explication des fonctionnalités de l’application
* Une architecture du logiciel au format MVC
* Une architecture de la base de données ainsi que sa structure
* Quels sont les points clés au niveau sécurité informatique qui ont été mis en œuvres ou qui auraient pu être mis en places.
* La qualité du code avec la gestion des versions et des fiches de recettes (Test applicatif)

Le choix du sujet de l’application à développer est laissé à l’étudiant.

Il s’agira d’une application permettant de consulter les fiches des bobines d’acier et de suivre leurs manutentions dans les halles du site.

# Construction du projet

## Choix des outils

Afin de répondre au cahier des et à l’architecture logicielle demandée, j’ai dû faire différents choix.

D’abord pour l’application, j’avais le choix entre le langage de programmation Java et le c#. C’est deux types de langage permettent de développer en respectant l’architecture MVC (modèle Vue Contrôleur). J’ai donc choisi le c# par commodité car c’est l’environnement de développement informatique que j’utilise au quotidien avec Visual Studio 2013. Je n’avais donc pas de nouveau langage informatique à apprendre hormis les concepts imposés par l’architecture MVC et les applications web.

Visual Studio 2013 permet également de développer des applications Web API et des applications mobiles utilisant les API Rest (get, delete, post).

Pour la base de données j’ai choisi SQL Serveur 2012 car c’est l’environnement de base de données qui est utilisé dans mon entreprise. De plus l’ensemble les fonctionnalités BI (Business Intelligence) est natif avec cette base de données qui seront utiles pour traiter la partie Data Warehouse et alimenter le cube.

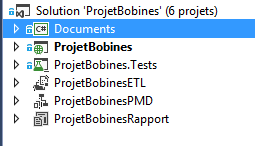
Pour la partie web, j’utilise les fonctions IIE pour l’hébergement de mon application Web moyennant l’ajout des fonctions ASP.Net.

Enfin pour la gestion de l’archivage et l’historisation des fichiers sources au cours du développement, j’ai choisi GitHub parmi la multitude d’applications existantes qui s’interface directement avec Visual Studio 2013. Cela m’a permis de retrouver la souplesse de ce que je connais avec TFS (Team Foudation Services) au quotidien.

Bien sur d’autres outils existent sur le marché et l’architecture logicielle aurai été différentes. Je reviendrais sur ces différents points dans les conclusions des différents chapitres de ce rapport.

## Découpage du projet sous VS2013

La solution est constituée de plusieurs projets distincts se trouvant dans la solution « ProjetBobines » sous Visual Studio.



Le projet « Documents » contient les documents comme ce rapport, un exemple de fichier Excel mappé sur le cube du projet

Le projet « ProjetBobines » contient l’ensemble des sources de l’application web au format MVC.

Le projet « ProjetBobines.Tests » contient l’ensemble des sources de l’application web au format MVC pour la gestion des tests et des recettes de l’application.

Le projet « ProjetBobinesETL» contient l’ensemble des fichiers de l’application Integration Services permettant la gestion des flux et l’intégration de données hautes perfomances. L’extraction, la transformation et l’entreposage (ETL) des données pour le Data Warehouse est décrits dans ce projet.

Le projet « ProjetBobinesPMD» contient l’ensemble des fichiers de l’application Analysis Services permettant la création des modèles multidimensionnels et d’exploration de données pour le cube de l’application.

Le projet « ProjetBobinesRapport» contient l’ensemble des fichiers de l’application Report Server pour la création des rapports utilisant le cube de l’application.

Web api … TODO

Appli mobile TODO

# Fonctionnalités

## Introduction

L’application permet aux utilisateurs de consulter des fiches de bobines à partir d’une interface web, de rapports BI et d’une application sur mobile.

L’application serveur permet de consulter ces informations à partir de pages web. Il est possible de modifier ces fiches produit grâce à divers actions : création, modification et suppressions.

Ces fiches contiennent les informations relatives à leur historique et leurs localisations dans les halles. L’utilisateur a la possibilité de faire des recherches des bobines soit par numéro à partir d’une liste déroulante et ainsi effectuer les actions de création, dédition et de suppression du produit.

Pour pouvoir accéder aux différentes vues, l’utilisateur doit s’identifier via une interface. Ces données sont sauvegardées en base de données avec un cryptage de son mot de passe.

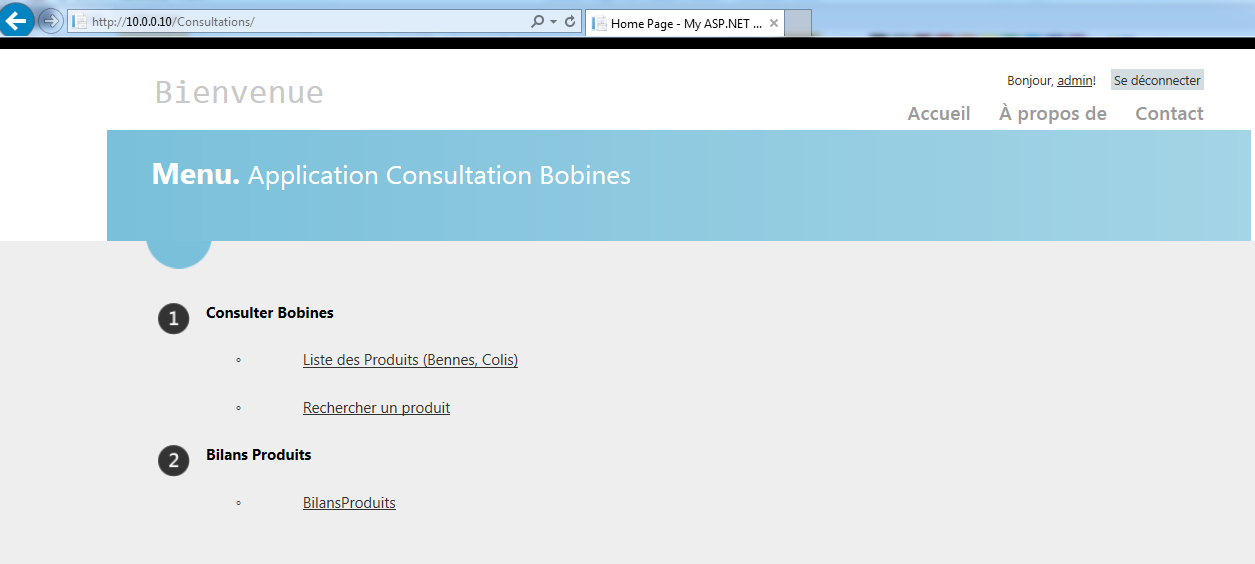
Une application mobile permet de consulter également ces informations par smartphone afin de pouvoir faire une vérification des produits directement dans les halles.

## L’application WEB

L’arborescence de l’application web est décrite en annexe 1 page TODO.

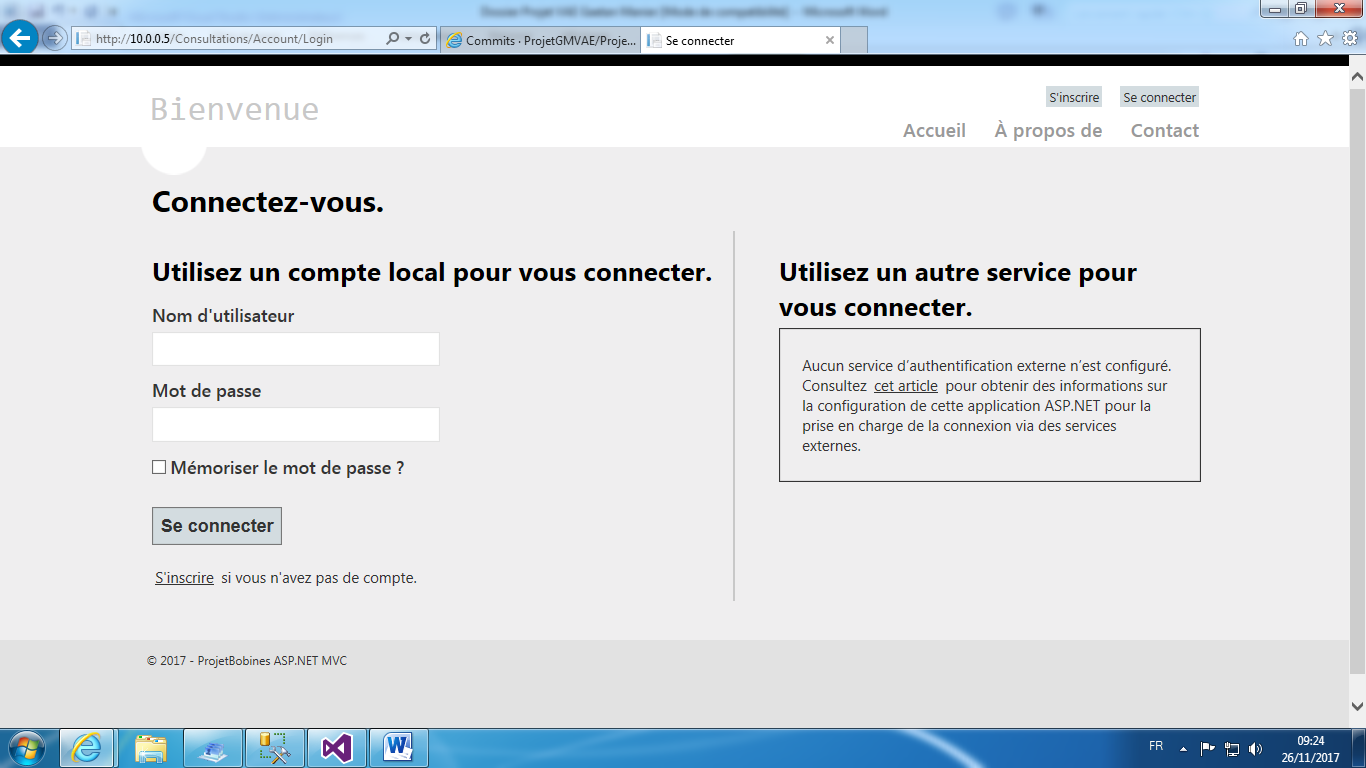
### La page d’accueil

Cette page s’ouvre lors de l’arrivée de l’utilisateur sur le site web. Elle permet d’accéder aux différentes fonctionnalités de l’application.



### La gestion des accès utilisateur

Lors de son arrivée sur la page d’accueil, l’utilisateur a la possibilité de s’inscrire ou de créer un nouveau compte.

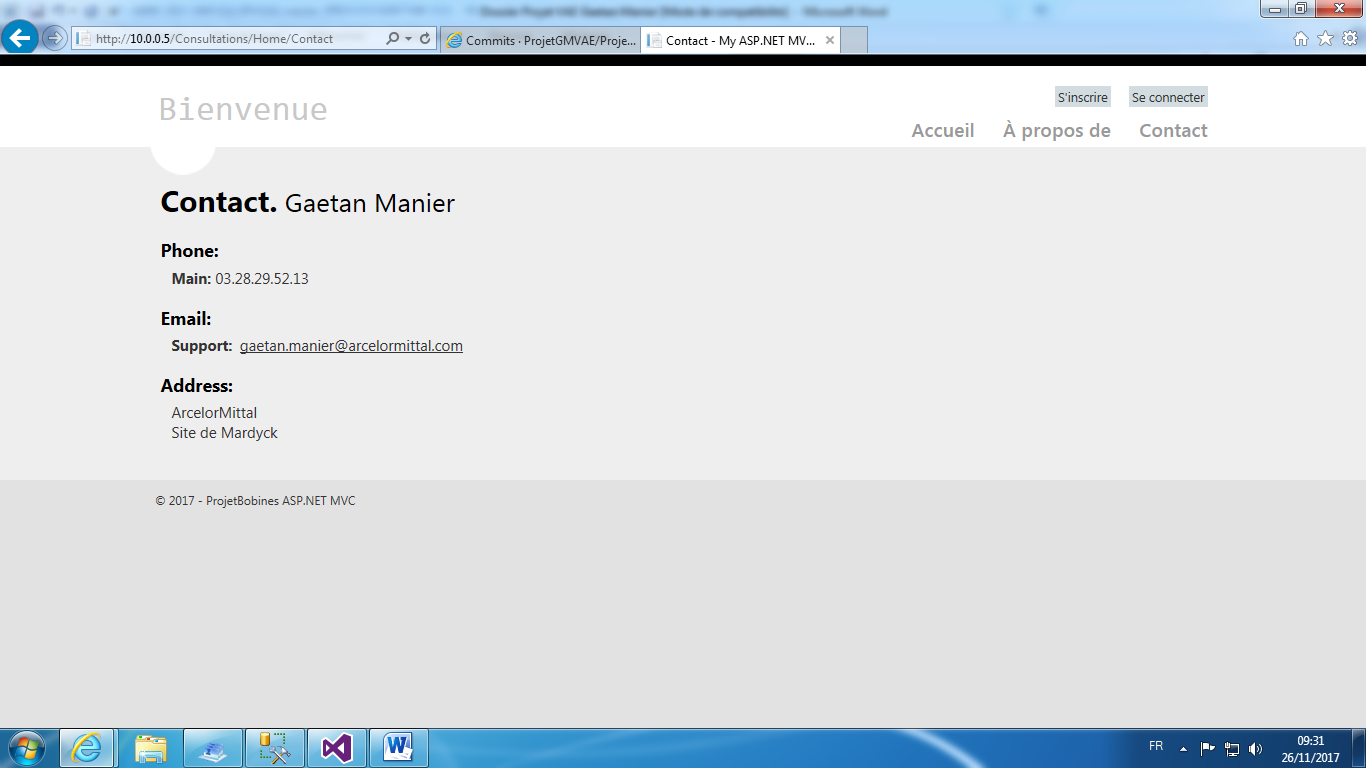
Lors de la création d’un nouveau compte, l’utilisateur doit envoyer un mail à l’administrateur du site afin de définir le rôle de son compte. L’accès aux différentes vues et aux fonctionnalités (création, suppression, mise à jour) se fait grâce à l’attribution de rôles sur les comptes.

Ces rôles sont les suivants :

* Admin : il permet d’avoir accès à l’ensemble des fonctionnalités du site
* Modificateur : il permet d’accès aux fonctionnalités de mise à jour des informations.
* Visiteur : il permet d’accéder uniquement en mode visualisation des vues

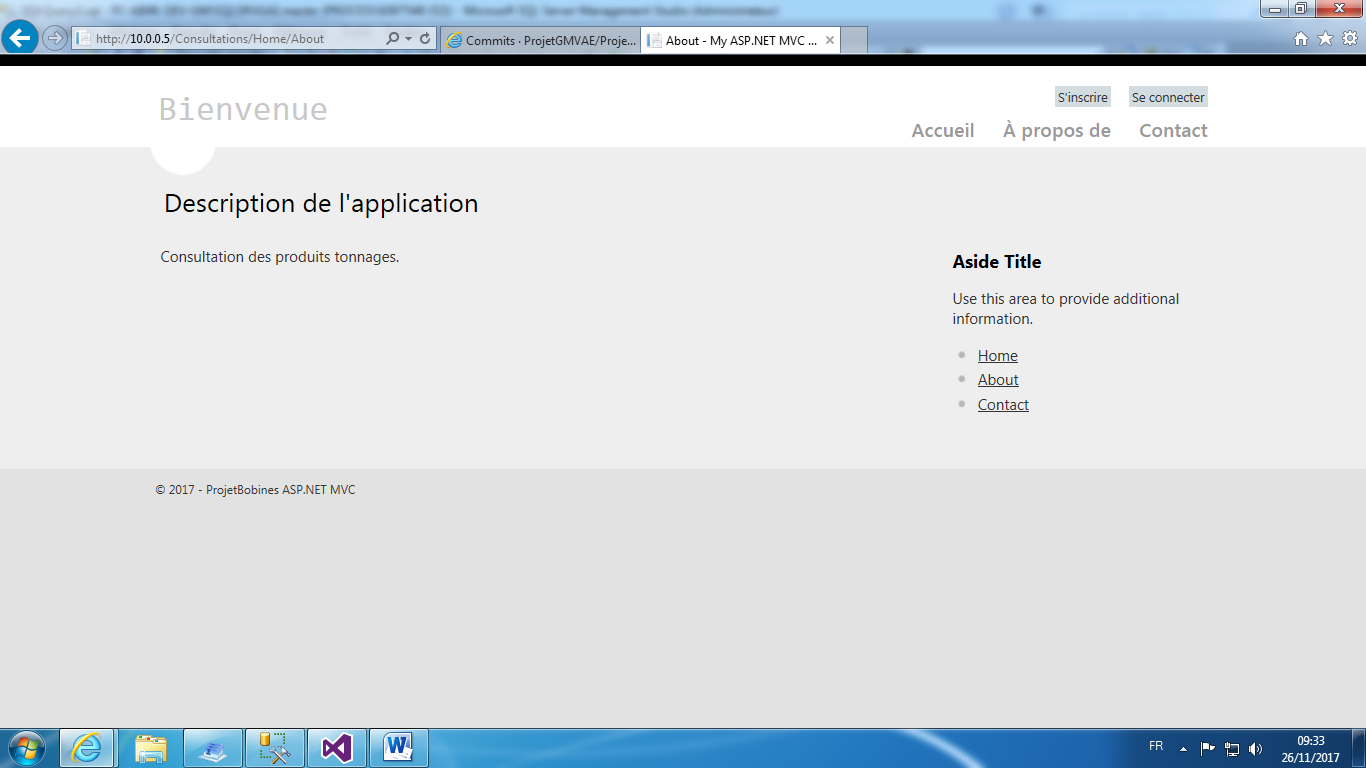
### La page « Contact »

Cette page permet d’avoir les coordonnées de l’administrateur du site afin de faire les demandes d’ajout de rôle sur le compte qui est créé.



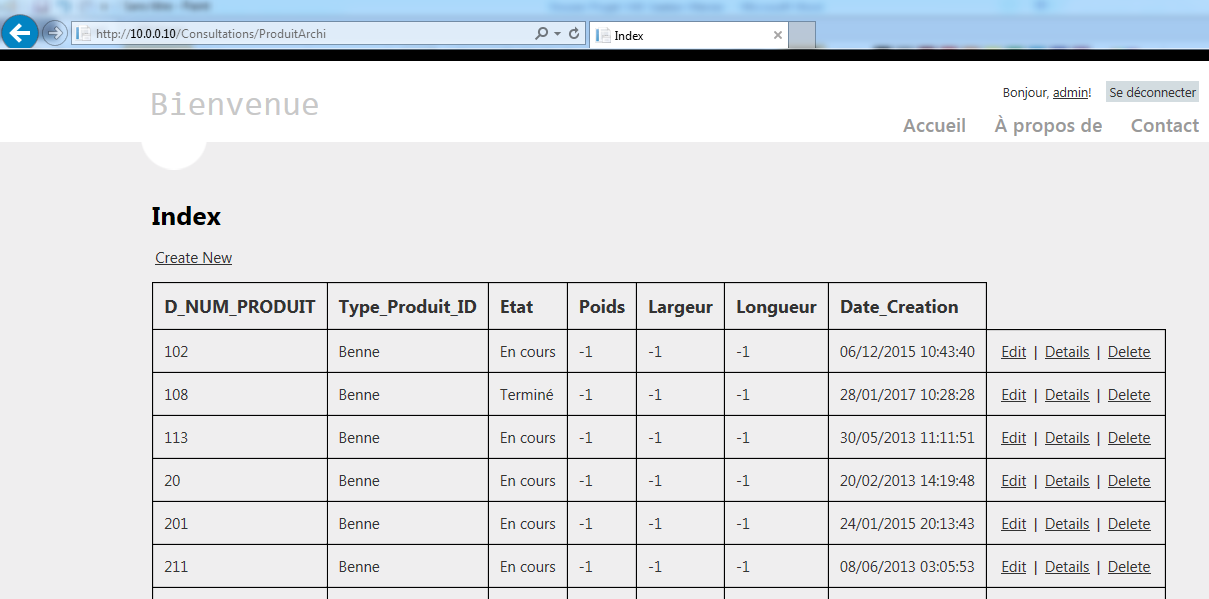
### La page « à propos de »

Cette page permet d’avoir un aperçu des fonctionnalités du site, de connaitre la version en cours.



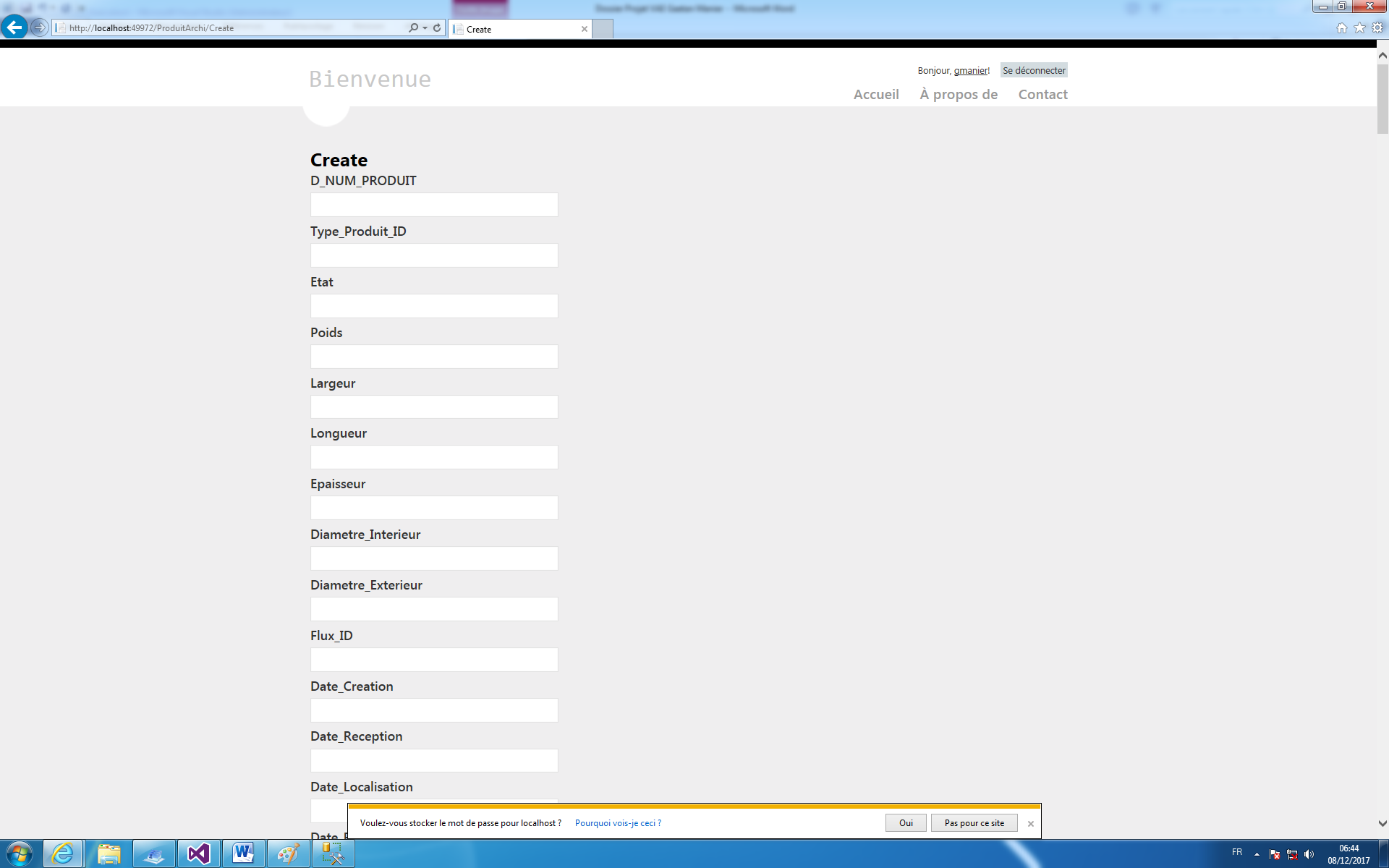
### La page « Liste des produits»

Cette page permet de visualiser les données des produits (bennes et colis) en base de données.

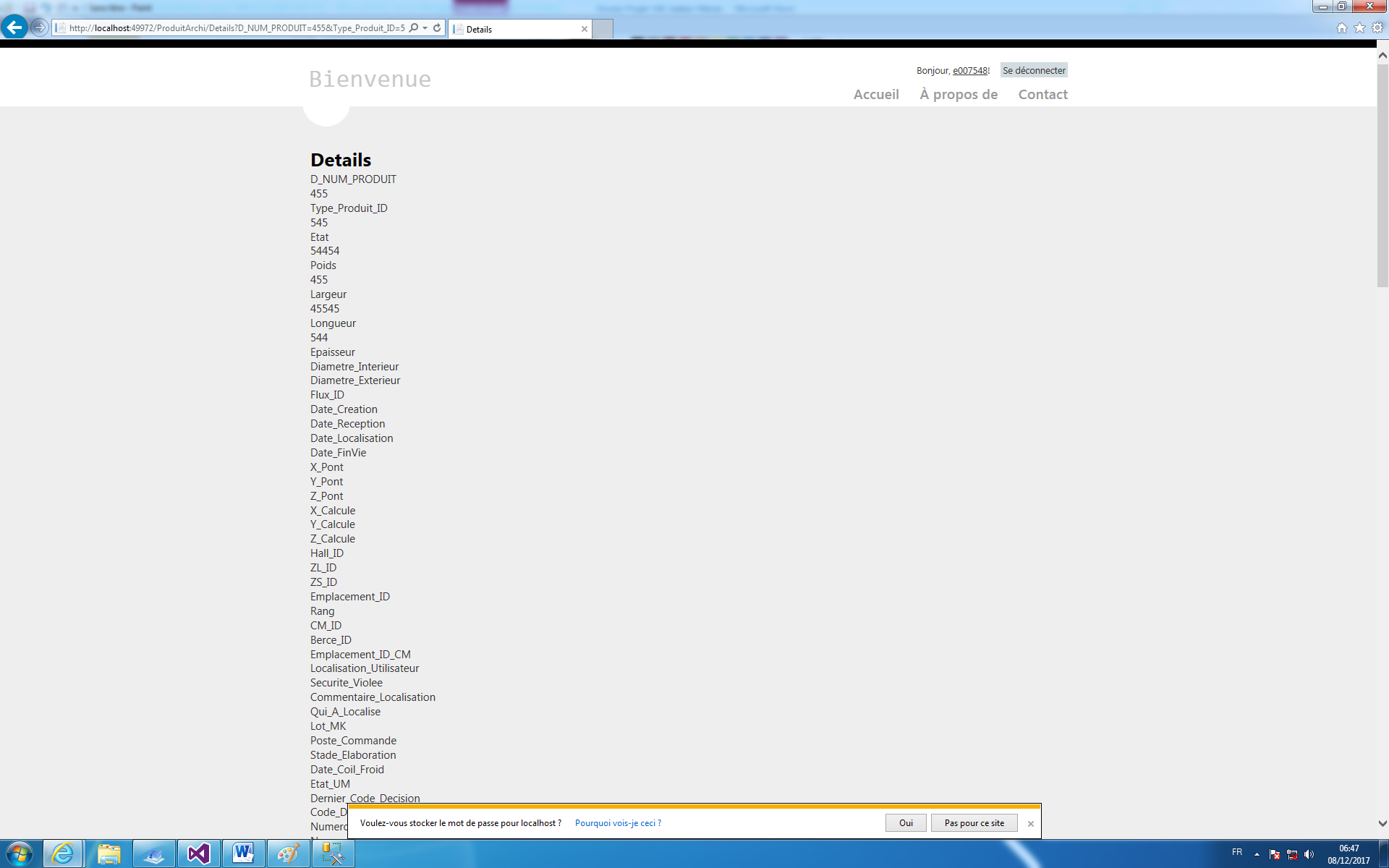


Elle se présente sous la forme d’une liste de produits présents dans la table « produits » sur laquelle l’utilisateur (suivant le rôle qui lui est attribué) peut :

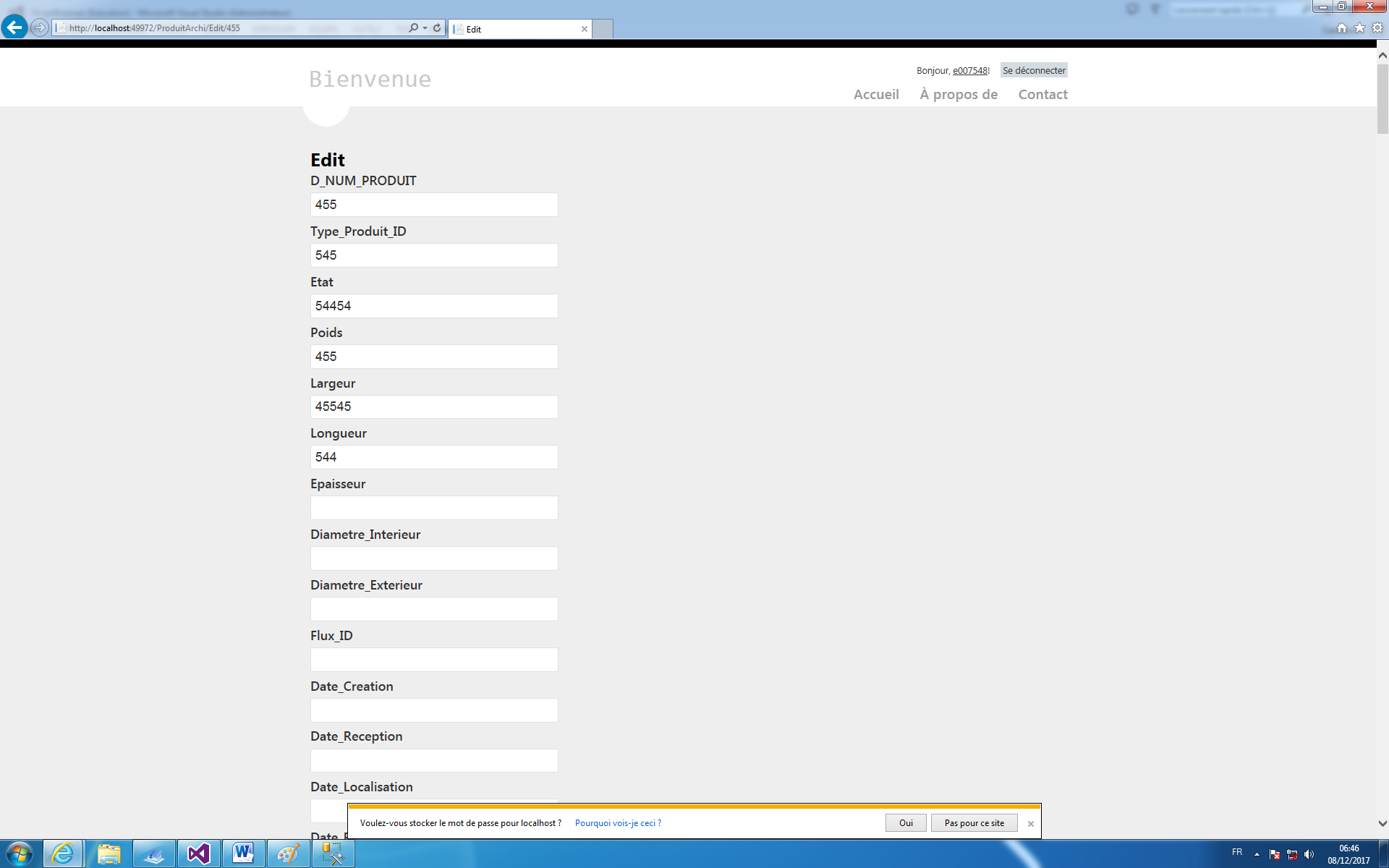
* Créer un nouveau produit



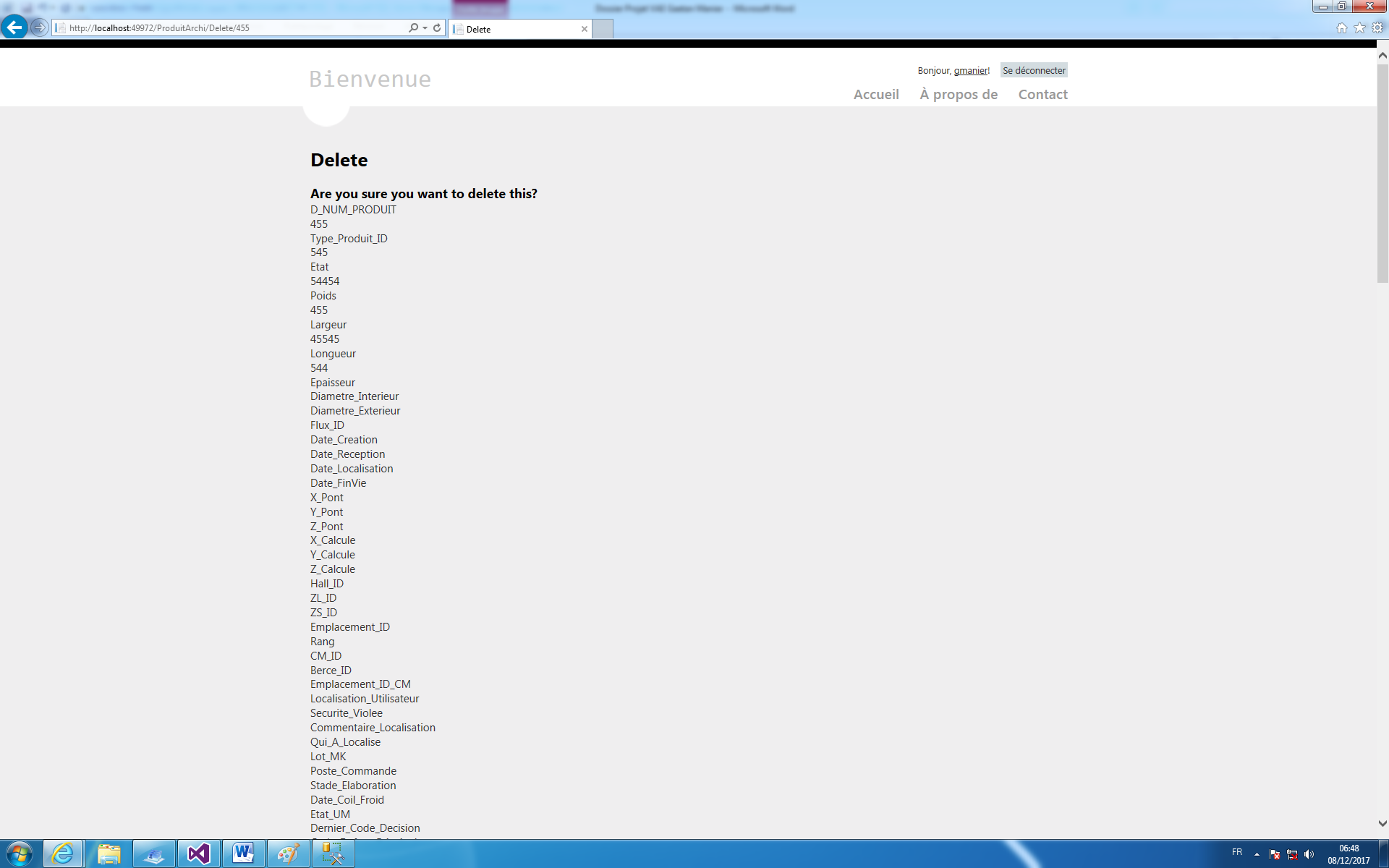
* Consulter le détail d’un produit



* Modifier un produit



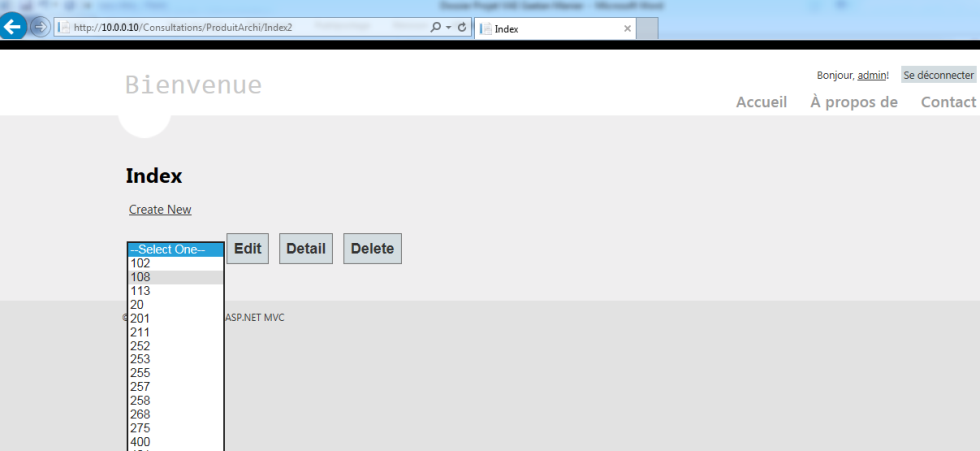
* Supprimer un produit



Ces données font parties de la table produits qui sert à l’alimentation du Data Warehouse par un ETL (décrit chapitre 4.4).

### La page « Recherche un produit»

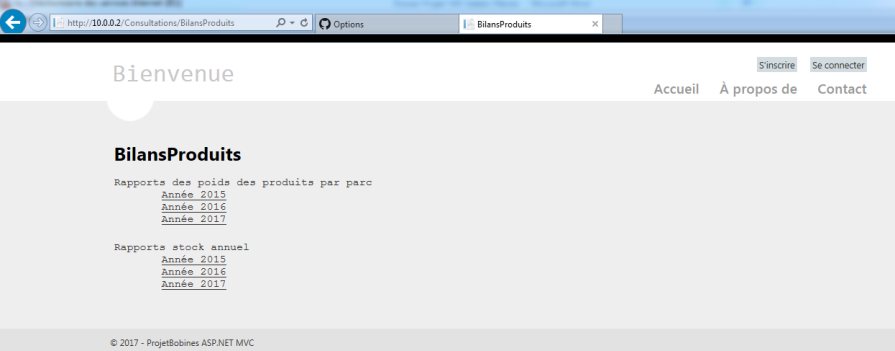
Cette page permet après avoir sélectionné un numéro de produit dans la liste déroulante d’effectuer les mêmes opérations que dans la page « Liste des produits ». C’est à dire la modification, la visualisation et la suppression d’un produit.



Ceux sont d’ailleurs les mêmes écrans et les mêmes méthodes qui sont appelées. Ceci grâce au découpage de l’application au format MVC qui permet la réutilisabilité de ces éléments.

### La page « Bilans bobine»

Cette page permet de consulter différents bilans bobines basés sur des rapports créés en automatiques et se basant sur les cubes du Data Warehouse.





### En conclusion

La création de cette application web a été pour moi une nouveauté à divers points de vue.

D’abord la création de l’architecture logicielle ne fait pas parti de mon métier de base car en temps normal j’ai un service informatique de développement qui s’en charge. Cela m’a permis de créer une application informatique de zéro en partant du concept jusqu’à sa réalisation.

Il est bien sur possible d’aller plus loin dans mon application :

* améliorer l’ergonomie des IHM
* ajouter des fonctionnalités de création de produit ou de mise à jour des tables de dimensions de notre schéma en étoile du Data Warehouse.
* ajouter de nouveaux rapports consultables

## Le Data Warehouse

normal

## L’application mobile

TODO.

# Architecture du logiciel

Le projet concerne l’application web (repère en A sur le schéma) qui respecte le standard MVC (Modèle Vue Contrôleur). Elle permet à l’utilisateur de consulter les données via une page web qui a pour effet d’envoyer une requête à une routine qui interroge le contrôleur. Celui par l’intermédiaire du modèle récupère les données pour les envoyer à la vue et ensuite les afficher sur la page web. L’accès à la base de données du modèle se fait en utilisant Entity Framework qui est un ensemble de technologie ADO.NET.

Le second projet concerne la base de données (repère en B sur le schéma) devant être un Data Warehouse afin de répondre au cahier des charges. Elle possède une table produits dans laquelle on retrouve de façon brute les données de bobines provenant d’une base de données classique de type SGBDR (systèmes de gestion de base de données relationnel).

Afin de créer le Data Warehouse, la table produit a ensuite été décortiquée afin d’etre sur un schéma de base de données de type OLAP (OnLine Analytical Processing). Une table de fait et plusieurs tables de dimension ont ainsi été créées sur le principe d’un schéma en étoile.

Ensuite un ETL (Extarct Transform Load) a été implémenté en utilisant SSIS (SQL Server Integration Services) afin de récupérer et de transformer les données de la table produit pour alimenter le Data Warehouse.

Un cube OLAP (Online Analytical Processing) a été créé en utilisant SSAS (SQL Server Analysis Services). Il permet de :

* Analyser des données déjà agrégées et stockées selon les besoins de l’utilisateur.
* Modéliser les données sous forme de dimensions et de mesures.
* Capacité à manipuler de gros volumes de données agrégés en croisant différentes dimensions.
* Créer des rapports directement dans Excel en utilisant toute la puissance du cube.
* Utiliser les fonctions classiques d’agrégation : Min, Max, Count, Sum, mais peut être enrichi avec des calculs spécifiques complexes.

Enfin la création et l’édition de rapports et de bilans est rendu possible par l’utilisation de SSRS (SQL Server Reporting Services). Les rapports se basent sur le cube précédemment créé et sont déployés sur un site web. Les rapports sont consultables depuis l’ application web en utilisant les raccourcis adéquats.

--------------------------------------------------------

Le troisième projet concerne l’application mobile permettant à l’utilisateur de consulter les données des produits sur son téléphone. Cela s’appuie sur une application WebAPI …..TODO

## Architecture de l’application web

L’architecture de mon application web est basée sur le Modèle Vue Contrôleur (MVC) qui est un patron d’architecture logicielle pour implémenter des interfaces utilisateurs et des applications web par exemple. Le découpage de l’application sous Visual Studio se trouve en annexe 3 page TODO .

Elle est composée de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

* Le Modèle

Élément qui contient les données ainsi que de la logique en rapport avec les données: validation, lecture et enregistrement. Il peut, dans sa forme la plus simple, contenir uniquement un simple texte, voire des données beaucoup plus compliquées[]. Le modèle représente l'univers dans lequel s'inscrit l'application[2].

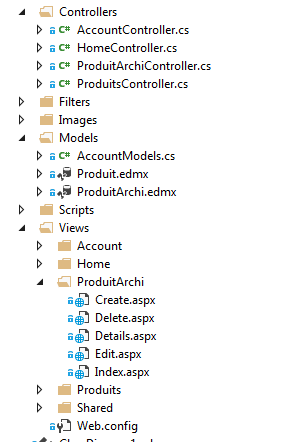
* La Vue

Partie visible d'une [interface graphique](https://fr.wikipedia.org/wiki/Interface_graphique). La vue se sert du modèle, et peut être un diagramme, un formulaire, des boutons, etc. Une vue contient des éléments visuels ainsi que la logique nécessaire pour afficher les données provenant du modèle. Dans une application web une vue contient des balises [HTML](https://fr.wikipedia.org/wiki/Hypertext_Markup_Language).

* Le Contrôleur

Module qui traite les actions de l'utilisateur, modifie les données du modèle et de la vue.

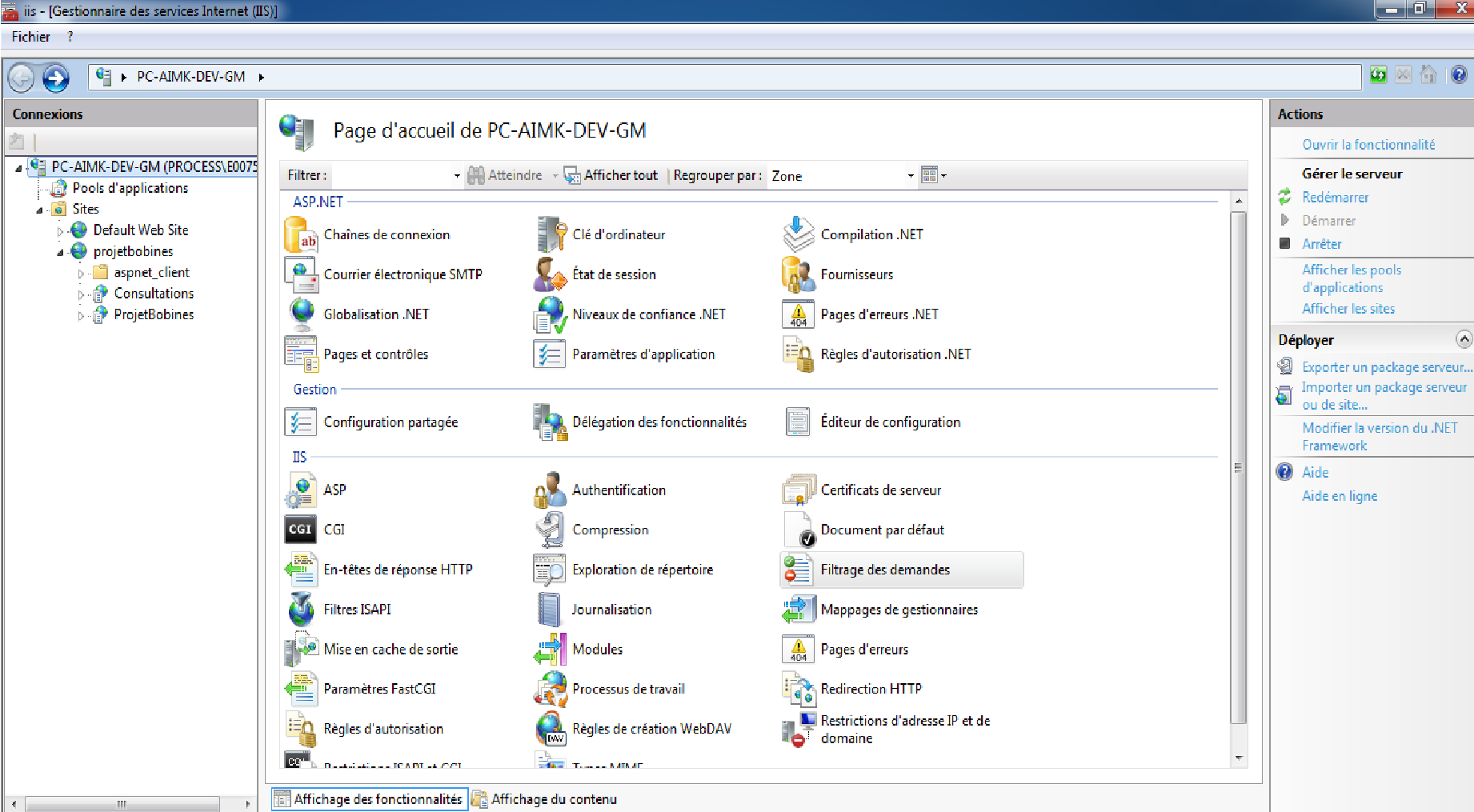
On retrouve donc ce découpage dans notre solution sous Visual Studio:



## Le serveur Web

Afin de pouvoir répondre au cahier des charge il m’a fallu créer un nouveau site web en utilisant IIS (Internet Information Services) sur mon PC. Une première pour moi car cela ne fait pas parti de la technologie que nous utilisons actuellement dans nos applications informatiques.

Après mettre documenté et effectué une série d’essais plus ou moins infructueux, j’ai créé le site web « projetbobines » qui contient les applications « ProjetBobines » et «Consultations ».



Lors de mes différents essais, j’ai été confronté à des problèmes liés au déploiement de mon application C# vers le service IIS.

* L’application développée en C# utilisait le Framework 4.5 alors que le site web et le projet d’application pointé sur le Framework 2.0 nativement. J’ai donc dû modifier ces paramètres dans IIS.
* La version de ASP n’était pas la bonne (*erreur : HTTP Error 500.21 - Internal Server Error Handler "ScriptHandlerFactory" has a bad module "ManagedPipelineHandler" in its module list*) , j’ai dû mettre la version de ASP en téléchargement la version ASP.NET v4.
* Une fois ces points réglés, je ne pouvais toujours pas déployer mon application C# de VS2013 vers IIS. Ceci aussi bien en utilisant l’option de publication que l’importation d’un package de déploiement (*erreur : No option to Import site package using IIS 6.1 & web deploy 3.5*). Pour cela j’ai dû mettre à jour la version web deploy 3.5.

## En conclusion

D’abord la création de l’architecture logicielle ne fait pas parti de mon métier de base car en temps normal j’ai un service informatique de développement qui s’en charge. Cela m’a permis de créer une application informatique de zéro en partant du concept jusqu’à sa réalisation.

# Architecture de la base de données

## Configuration du serveur de base de données

Le choix de la base de données s’est porté sur SQL Server 2012 qui est utilisé en standard dans nos applications informatiques.

Les bibliothèques de business intelligence ont été rajoutées afin de pouvoir utiliser l’ensemble des outils nécessaires à la construction et la gestion du Data Warehouse.

* Integration Services : permet l’alimentation des tables du Data Warehouse à partir des tables « brutes » de l’exploitation.
* Analysis Services : permet de créer et de gérer le cube qui s’interface sur le Data Warehouse.
* Reporting Services : permet de générer des rapports automatiques à partir de la base de données en se basant sur le cube.

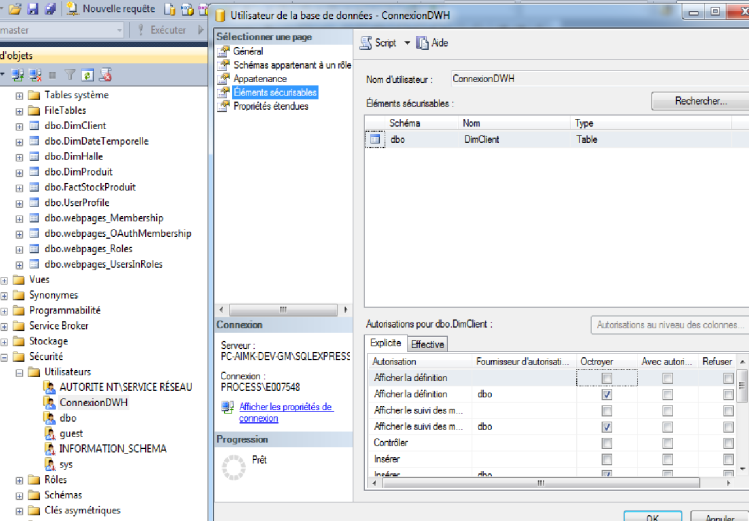
La configuration et l’utilisation de ces différents services seront vues plus en détail dans les chapitres suivants.

### Compte pour l’application Web

L’application web se connecte à la base de données en utilisant un compte spécifique sans utiliser les comptes Windows par défaut.



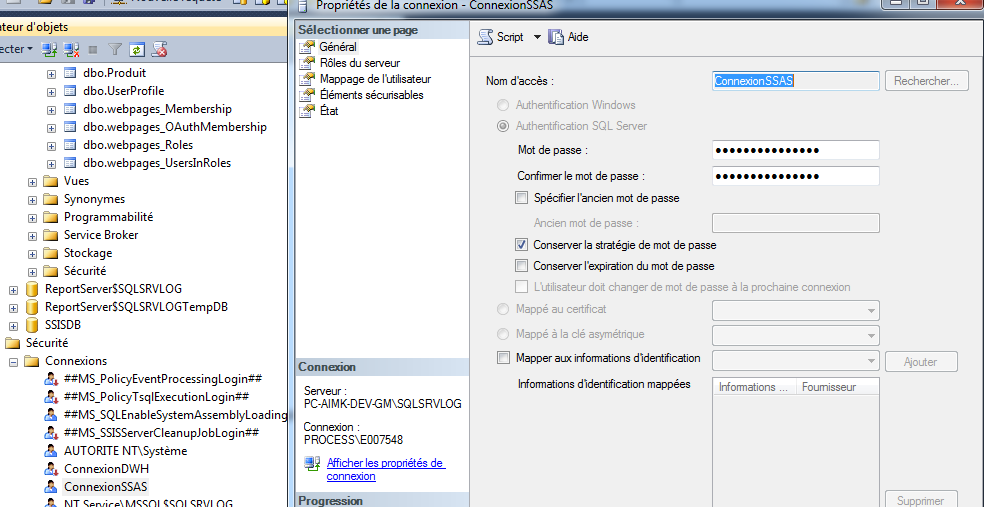
Pour cela il a été créé une connexion au niveau du serveur de base de données qui s’appelle « ConnexionDWH ».



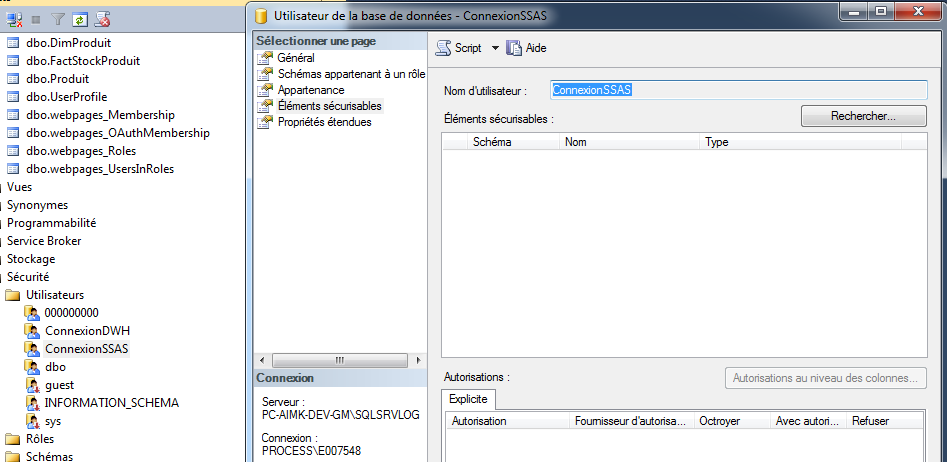
Ensuite un compte utilisateur sur la base de données « ConnexionDWH».

### Compte pour le SSAS

Dans les bonnes pratiques de configuration d’un service SSAS, il est conseillé de configuré un compte de base de données dédié à cela.



Pour cela il a été créé une connexion au niveau du serveur de base de données qui s’appelle « ConnexionSSAS ».



Ensuite un compte utilisateur sur la base de données « ConnexionSSAS».

## Introduction aux Data Warehouse

Dans le cahier des charges, il m’a été demandé de réaliser un Data Warehouse (entrepôt de données) afin de structurer l’archivage des données et d’en faciliter l’interrogation.

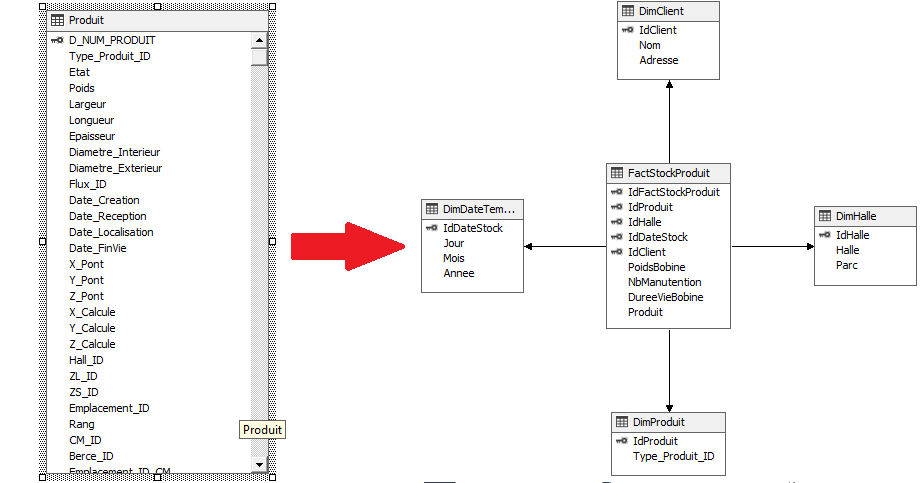
En temps normal j’utilise des bases de données de type SGDB (système de gestion de base de données relationnelles) dans nos applications car elles répondent aux besoins de stockage de données bidimensionnelles de production.

Ce type de base de données est fortement utilisé dans ce cas, car les mises à jours de données sont régulières et en temps réel.

L’utilisation du Data Warehouse modifie en profondeur la composition de la base de données. D’abord la base de données est gérée par un serveur OLAP (Online analytical processing).

## Choix du schéma

A partir de la table produit (contenant l’ensemble des informations bobines), j’ai créé le schéma en étoile suivant. Bien sûr cela ne représente qu’une portion de ce qui est aurait pu être faite à partir de la table d’origine.



L’utilisation du Data Warehouse modifie en profondeur la composition de la base de données. D’abord la base de données est gérée par un serveur OLAP (Online analytical processing).

Pour pouvoir faire ce choix de schéma en étoile, j’ai d’abord analysé les besoins de recherche et de reporting que l’on peut avoir besoin lorsque l’on souhaite manipuler les données de notre base de données actuelle.

Ce que j’en ai déduit est que généralement on souhaitait faire des recherches à partir des halles ou des clients pour en déduire les stocks de produits existant.

J’ai donc créé une table de fait (FactStockProduit) dans laquelle on retrouve les différentes clés en lien avec les tables de dimensions, ainsi que les colonnes comme le poids, le nom des produits ou encore le nombre de manutention. L’alimentation de cette table sera fera lors de l’exécution de l’ETL (Extract-Transform-Load) décrit plus bas.

Pour les tables de dimension, je me suis contenté pour notre exemple des tables clients, produit, halle et date. L’alimentation de ces tables a été réalisée en prenant les données de la table produit lors de la phase de construction de la base de données.

## Alimentation du Data Warehouse avec SSIS

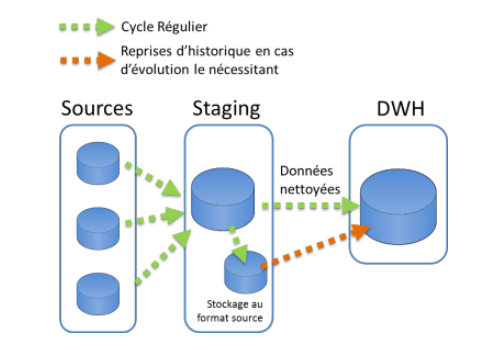
Pour pouvoir alimenter le Data Warehouse j’ai programmée un ETL (Extract-Transform-Load) en utilisation une solution « integration services » du module business intelligence de Visual Studio 2013.

L’ETL permet de synchroniser et de stocker des informations massives provenant principalement de bases de données.

### Configuration de l’ETL

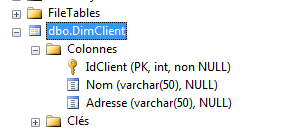
A partir des données de la table produit, l’ETL permet d’alimenter la table FactStockProduit (tables des faits du schéma en étoile) en effectuant différentes recherches dans les tables de dimension.

Les tables de dimension contiennent les index pour les données halle, clients, produit, date. Elles ont été alimentées manuellement en insérant les données nécessaires provenant de la table produit avec en plus une indexation automatique.



Par exemple pour la table DimClient j’ai procédé ceci :

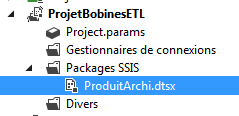
* Création de la table



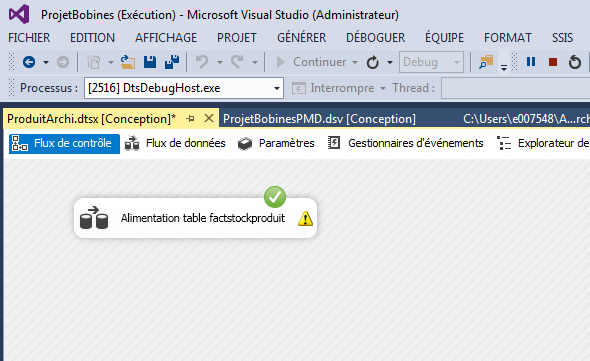
* Alimentation de la table à partir d’un fichier excel (suppression des blancs) qui se base sur le select distinct des champs de la table produits.

Pour alimenter la table FactStockProduit l’ETL est configuré comme ceci :

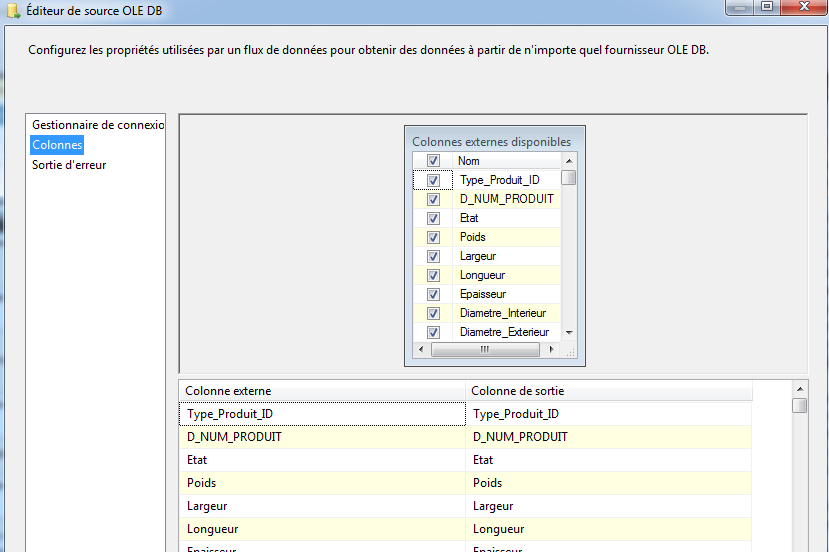
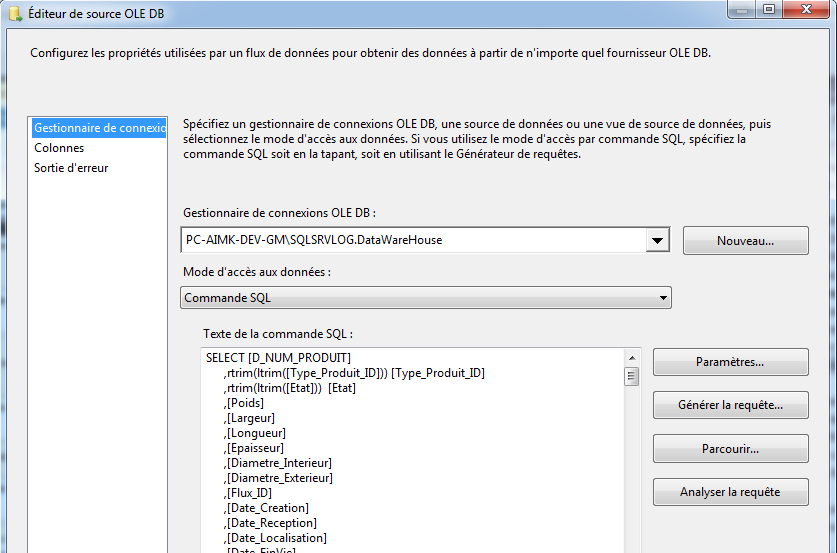
* Dans la solution il faut créer un package SSIS



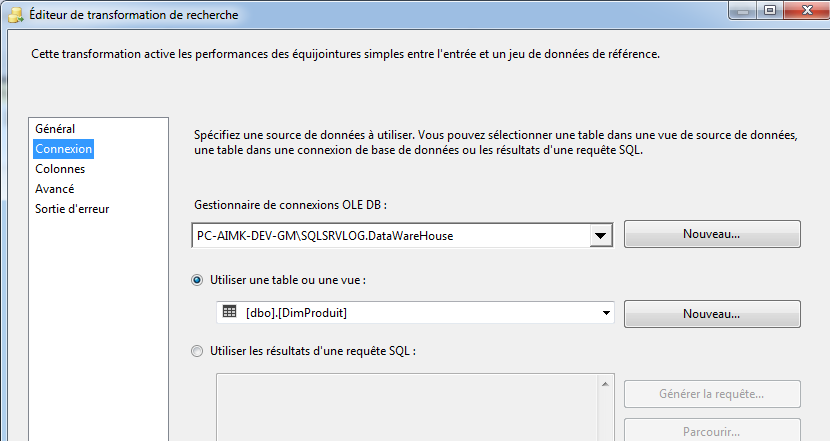
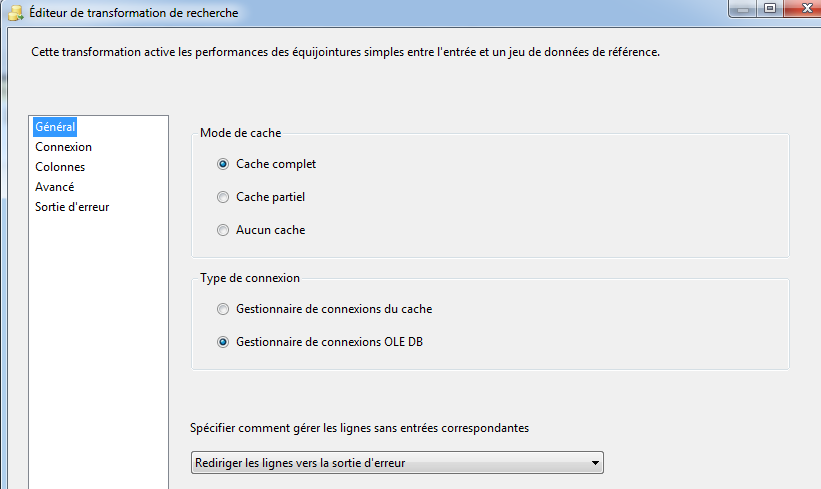
* On crée ensuite un flux de contrôle.

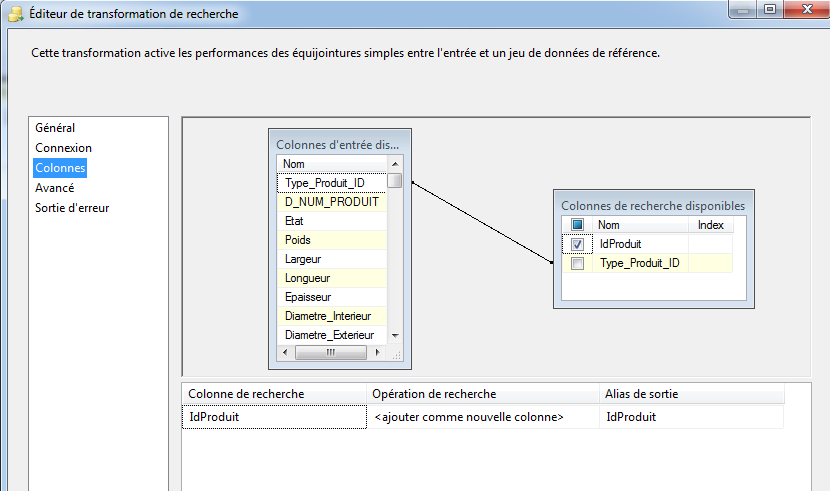


* Dans le flux de contrôle on double clic pour créer le flux de données.
* La source de données est configurée de telle façon que l’on récupère l’ensemble des données de la table produit. Dans la commande SQL les données de type chaînes de caractères utilisées pour effectuer des requêtes dans les tables de dimensions sont traitées de façon à supprimer les blancs devant et derrière les données.

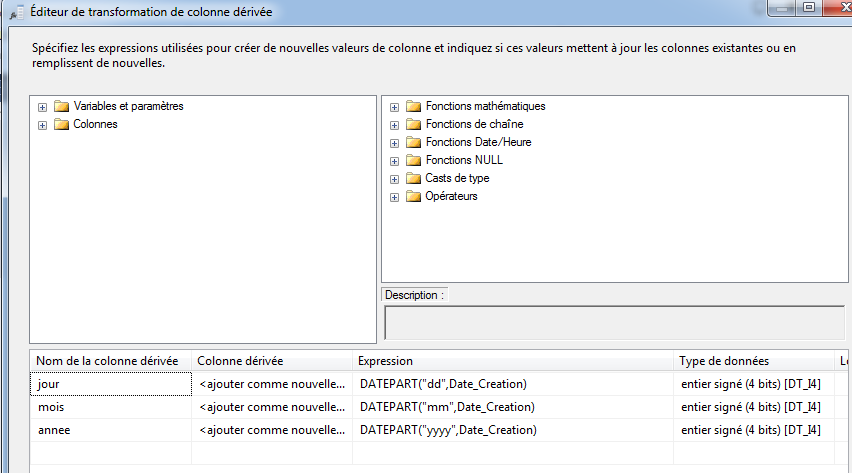


* En sortie de la source de données on vient rajouter un composant de recherche du produit qui va permettre de recherche l’ID de la table de dimension DimProduit en fonction du type de produit.

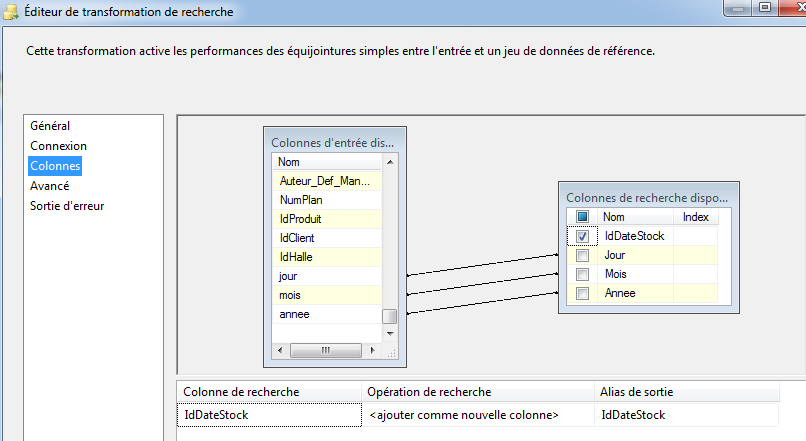




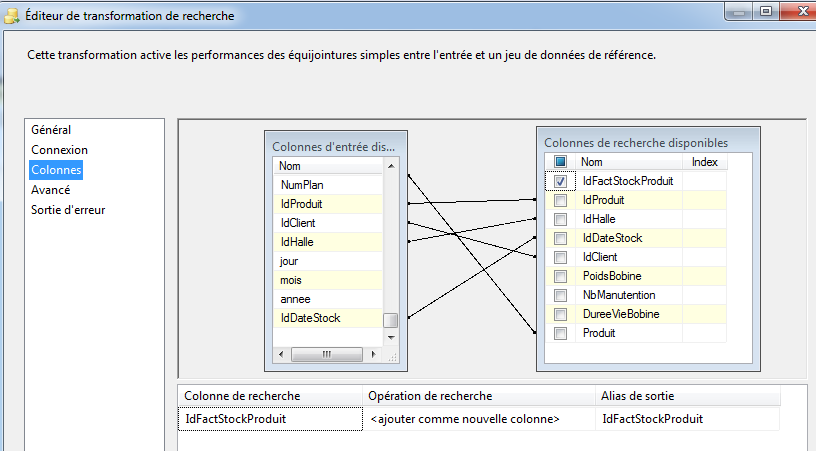
* Lorsque l’on relie la sortie d’un composant de recherche vers un autre, il faut spécifier si la recherche se fait avec ou sans correspondance. Dans notre cas ce fut par correpondance.
* On effectue la même chose pour retrouver les ID clients et Halles des tables DimClient et DimHall.
* Pour la recherche de l’identifiant de la table de dimension DimDateTemporelle, j’ai procédé d’abord en utilisant un composant de colonne dérivée. Il m’a permis de sortir de nouvelles colonnes jour, mois et année.



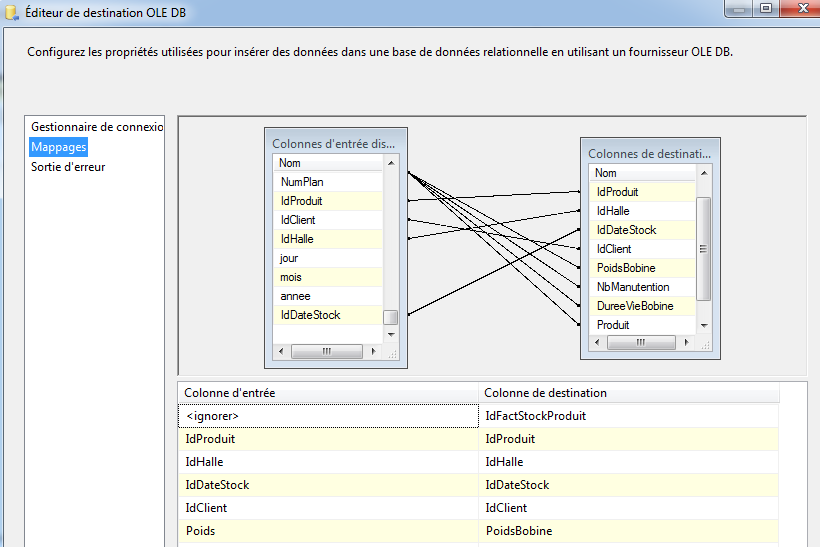
* Les nouvelles colonnes ainsi crées m’ont permis de les affecter au composant de recherche de la table de dimension DimDateTemporelle.



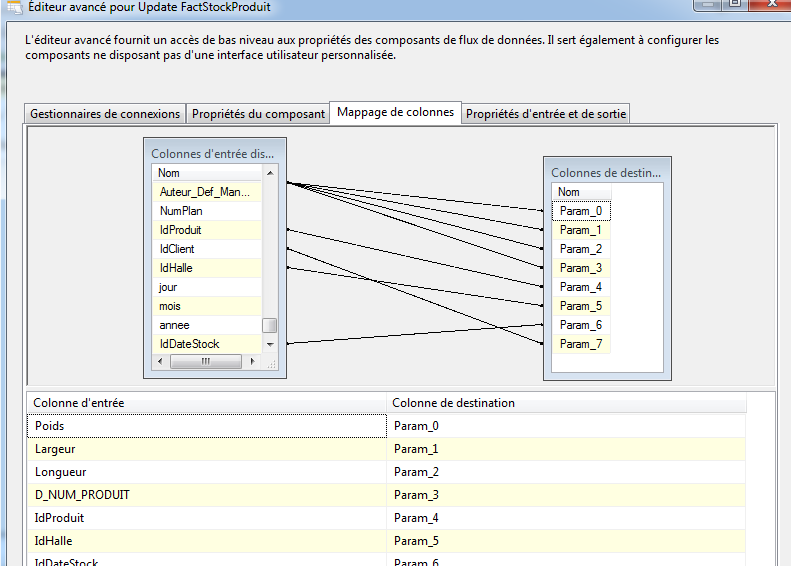
* Avec d’envoyer les données vers la table de destination, il fallait tester si les données étaient déjà présentes dans la table. Pour cela un composant de recherche permet de vérifier si les données pour un même enregistrement IDclient, IdHall, IDDateStock et IDProduit existent dans la table. Si la recherche donne un IDFactStockProduit cela indiqué qu’il existait déjà un enregistrement.



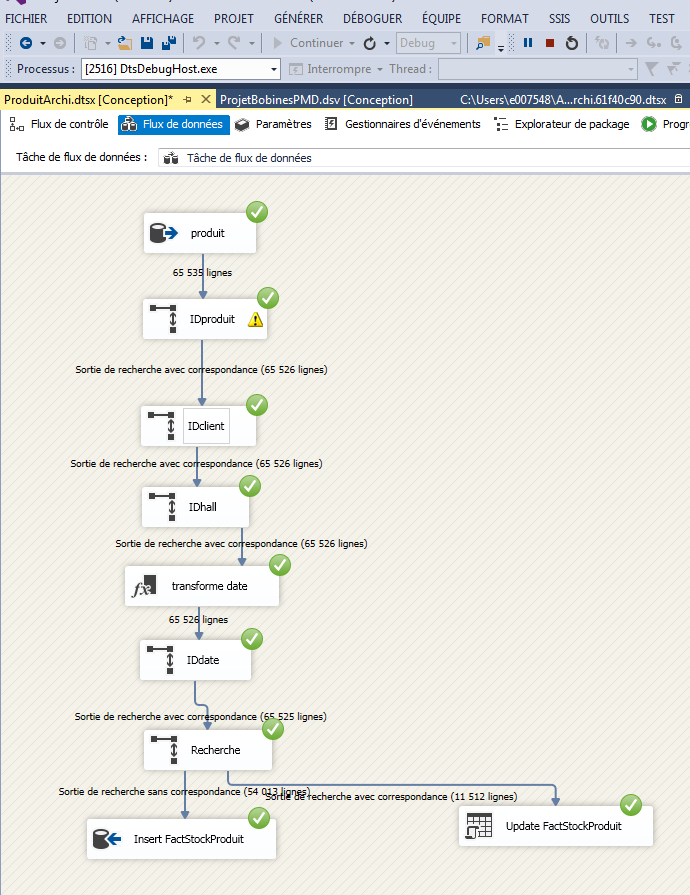
* La sortie sans correspondance du composant de recherche permet d’insérer un nouvel enregistrement dans la table FactStockProduit.



* En sortie avec correspondance du composant permet de mettre à jour la table FactStockProduit.



Lors de l’exécution de l’ETL dans Visual Studio nous avons les différentes étapes qui passent au vert indiquant ainsi que la programmation a bien été effectuée.



Comme nous pouvons constater dans le schéma ci-dessus, l’exécution de l’ETL s’est bien déroulée. Sur les 65525 enregistrements, 54013 enregistrements ont été insérés dans la table FactStockProduit et 11512 ont été mis à jour. Une fois le projet créé, je l’ai déployé sur le serveur de base de données.

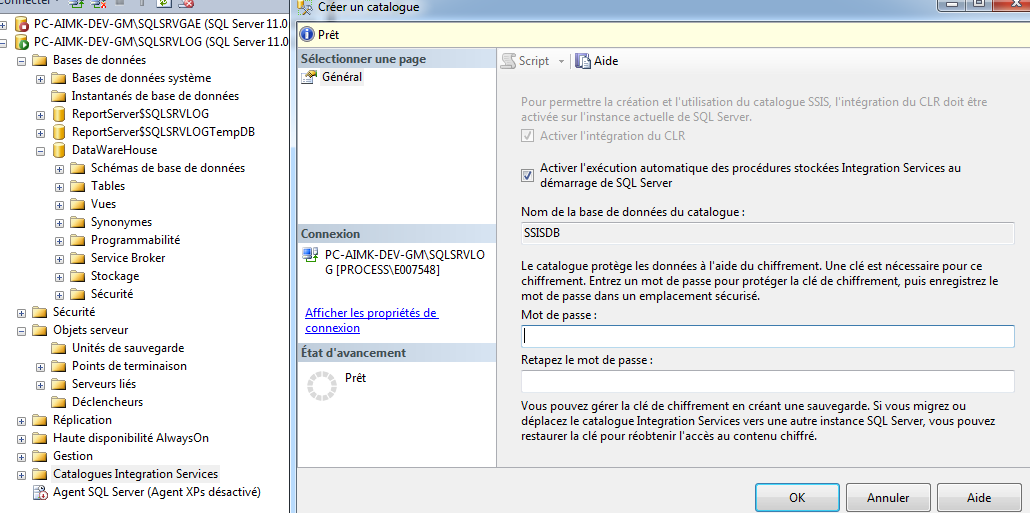
### Déploiement de l’ETL

Sur le serveur j’ai créé le catalogue **SSISDB** afin de pouvoir déployer les différents packages de ma solution de l’ETL que j’ai mis en place.

Pour cela dans SQL Server Management Studio il faut créer un catalogue SSISDB « Catalogues Integration Services ».

Le catalogue SSISDB est l’élément central pour l’utilisation des projets Integration Services Integration Services (SSIS) que j’ai déployé sur le serveur Integration Services Integration Services. Ainsi, c'est dans ce catalogue que j’ai définis les paramètres de projet et de package, j’ai configuré les environnements pour spécifier des valeurs d'exécution pour les packages, exécuté et résolue les problèmes relatifs aux packages, et géré les opérations du serveur Integration Services Integration Services

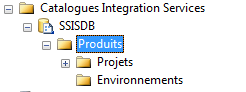
Pour cela un clic droit permet d’afficher la fenêtre d’invite et de remplir les différents éléments.



Il faut cocher les 2 cases dont l’intégration du CLR (Common Language Runtime).

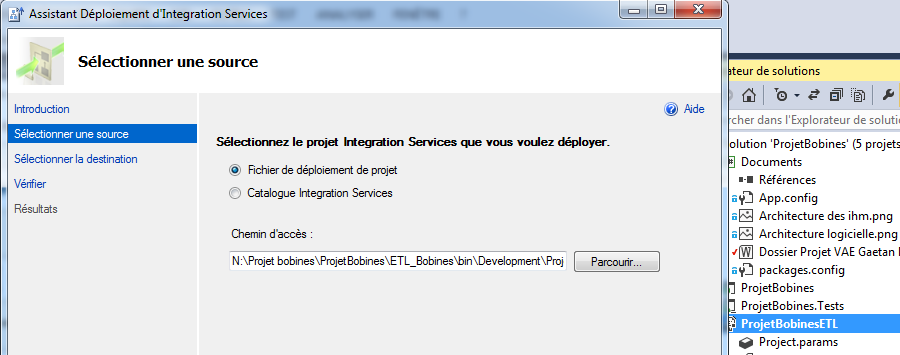
Le Common Language Runtime (CLR) est au cœur de Microsoft .NET Framework et fournit l'environnement d'exécution pour tout le code .NET Framework. Le code qui s'exécute dans le CLR est appelé code managé. Le CLR fournit divers services et fonctions requis pour l'exécution du programme, notamment la compilation juste-à-temps (JIT), l'allocation et la gestion de mémoire, l'application de sécurité de type, la gestion des exceptions, la gestion des threads et la sécurité.

En faisant un clic droit sur SSISDB, j’ai créé un projet avec le nom « produits » qui va contenir les différents objets déployés.

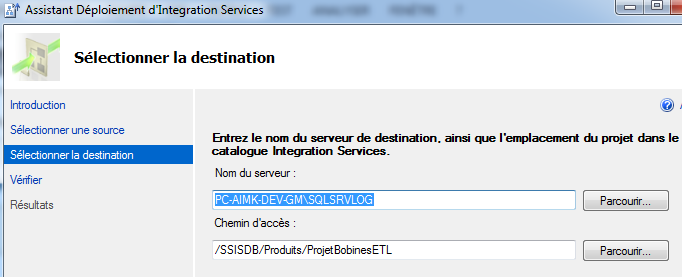


Sous Visual Studio, le clic droit sur le projet ProjetBobinesETL permet de sélectionner «déployer » et d’avoir l’assistant de déploiement d’Integration Services.

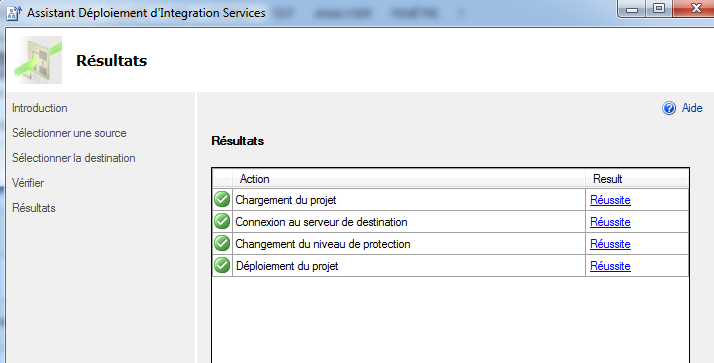
Il faut sélectionner le projet d’Integration Services à déployer. Dans mon cas il s’agit du choix par défaut correspondant au projet sélectionné.



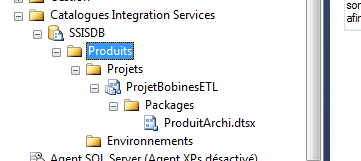
Il faut ensuite sélectionner la destination en sélectionnant le nom du serveur de base de données et le chemin d’accès. Dans notre cas on retrouve le projet précédemment créé dans Sql Server Management Studio.



En cliquant sur le bouton « suivant », on finalise le déploiement et si tout se passe correctement nous avons l’écran suivant :



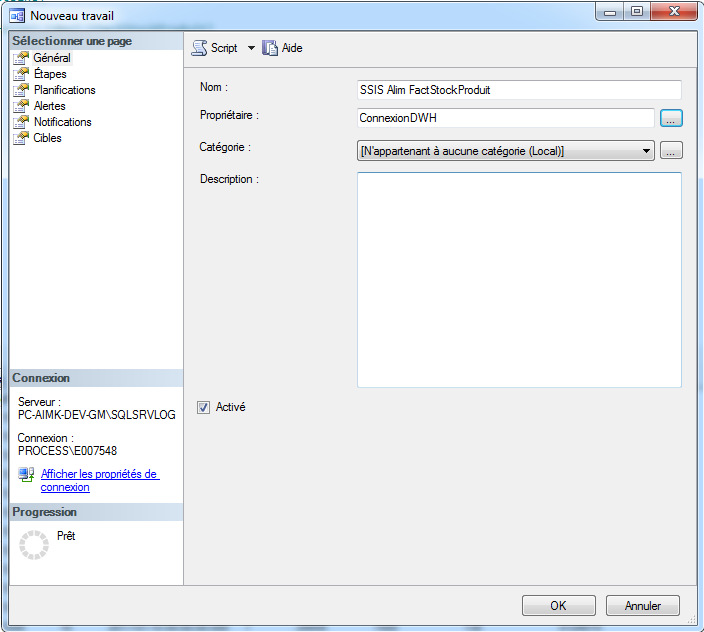
Dans Sql Server Management Studio on retrouve le projet déployé.



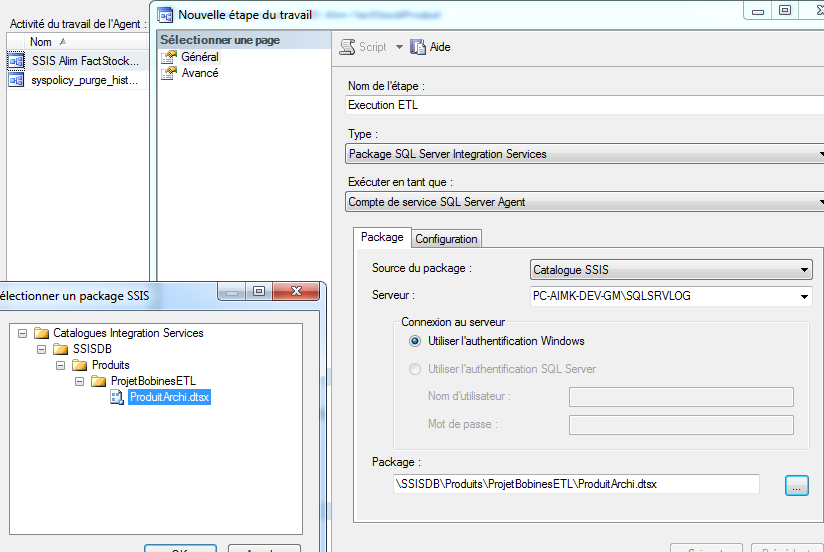
### Exécution de l’ETL

Une fois l’ETL déployé dans le serveur de base de données, j’ai configuré l’agent SQL Server afin de le faire exécuter en automatique.

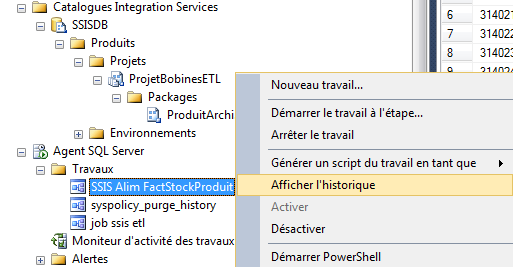
Pour cela il faut nommer le nouveau travail à faire exécuter et affecter un propriétaire.



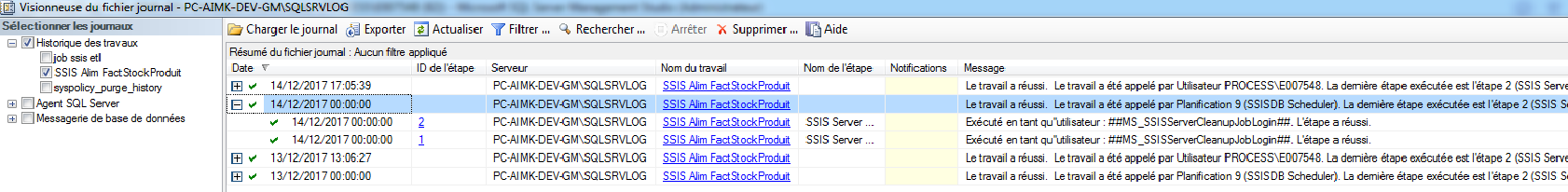
Puis il faut définir les étapes que l’on souhaite faire exécuter.



Afin de vérifier le bon fonctionnement du déclenchement de l’ETL, j’ai vérifié son historique.



On constate que la tâche d’exécution de l’ETL se déroule normalement en manuel et en automatique.



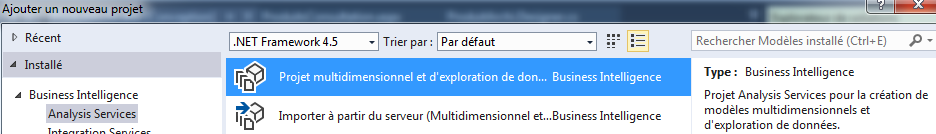
### Pour aller plus loin

Pour aller plus loin dans mon projet il aurait été possible de rajouter les points suivants :

* En ajoutant un flux de contrôle avant l’alimentation de la table FactStockProduit on aurait pu alimenter en automatique les tables de dimensions.
* Pour la gestion du stock, il est possible de rajouter une transformation de dimension à variation lente afin de gérer les écarts de stock ou les calculs de durée de vie des produits sur l’historique.
* Ajouter d’autres tables de dimensions afin de proposer d’autres possibilités de recherche et d’autres rapports aux utilisateurs.
* La table produit est suffisante à elle-même pour poursuivre les développements de ce projet et de proposer d’autres tables de faits et de dimensions comme le suivi des manutentions de produits.

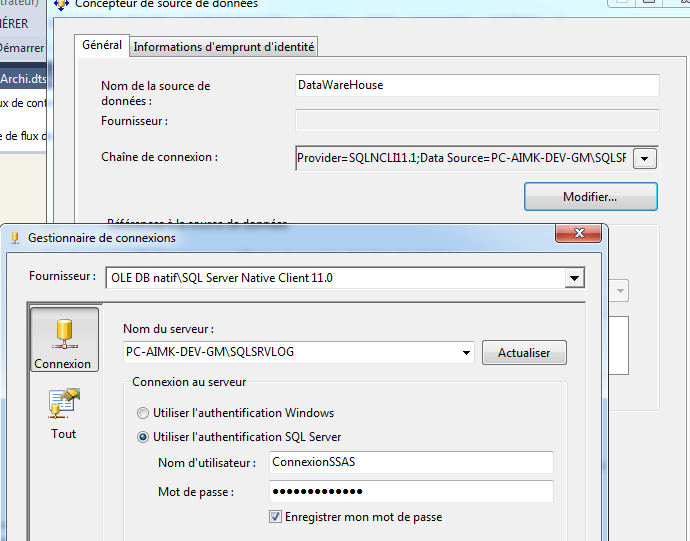
## Création et gestion du cube avec SSAS

Pour le créer le cube, j’ai utilisé les outils SSAS (SQL Server Analysis Services) fournis dans Visual Studio. Pour cela il faut créer un nouveau projet multidimensionnel et d’exploration de données dans notre solution.



### Connexion à la base de données

Il faut d’abord créer la connexion à la base de données. Dans les bonnes pratiques conseillées par Microsoft, il faut créer un compte dédié au cube. C’est le cas avec le compte connexionSSAS vu précédemment.



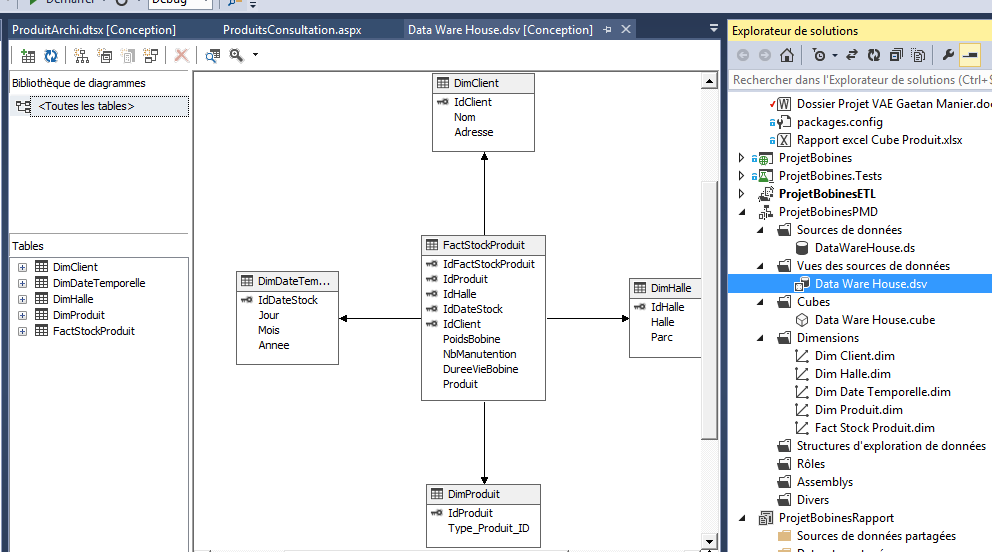
### Vue des sources de données

J’ai ensuite créé la vue des sources de données qui contient le modèle logique du schéma utilisé par le cube, les dimensions et les structure d’exploration de données. Elle représente la définition (stocké au format XML) des métadonnées de ces éléments de schéma par le modèle UDM (Unified Dimentional Model) et par les structures d’exploration de données.

Le rôle d'un UDM consiste à établir une passerelle entre l'utilisateur et les sources de données. Un UDM est construit sur une ou plusieurs sources de données physiques. L'utilisateur émet des requêtes sur cet UDM, à l'aide d'une large palette d'outils clients, tels que Microsoft Excel.

L'UDM ne constitue qu'une fine couche au-dessus de la source de données et offre à l'utilisateur final les avantages suivants : modèle de données plus simple et plus lisible, indépendance par rapport aux sources de données d'arrière-plan hétérogènes et optimisation des performances des requêtes de type résumé. Dans certains scénarios, il est possible de construire automatiquement un simple UDM. Lorsque l'UDM est construit avec des investissements plus conséquents, les avantages peuvent décupler grâce à la richesse des métadonnées que le modèle peut fournir.

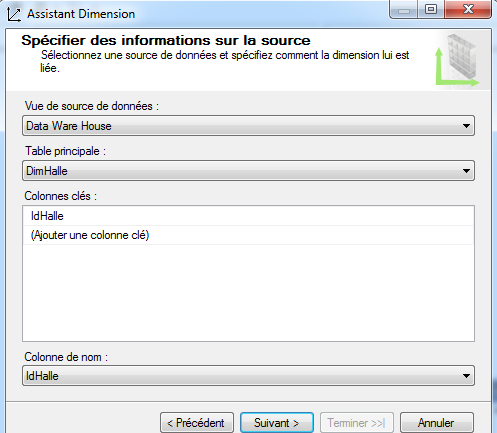
Un assistant de création permet de sélectionner la base de données puis de choisir les différentes tables de dimension et la table de fait.



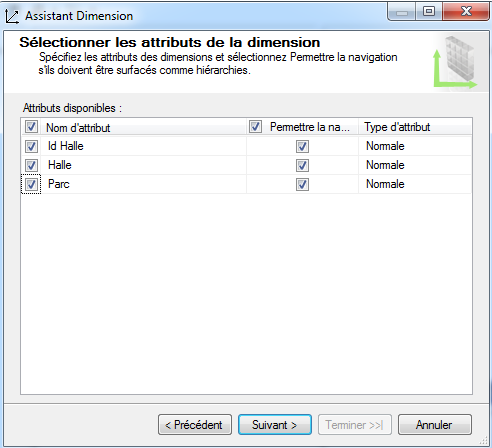
### Création des dimensions

J’ai ensuite créé les dimensions à partir des tables de dimension de la base de données. L’assistant de création de dimensions permet également de générer des tables type comme les tables temporelles, de production, de suivi client etc…

Pour les 4 tables de dimension de mon projet j’ai donc procédé de la même façon. Dans la vue de source de données j’ai sélectionné une table de dimension, la table DimHalle pour notre explication



J’ai ensuite sélectionné les attributs de la table de dimension.

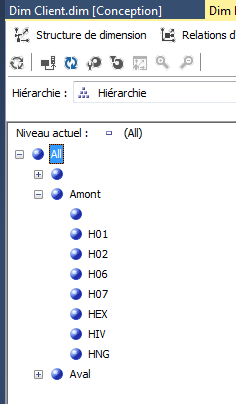


Puis dans l’interface j’ai pu définir la hiérarchie des données pour cette table. Dans notre cas nous avons le parc (amont, aval) puis les halles (E2, P2…).



Le bouton « traiter »  permet de déployer sur le serveur la dimension ainsi créée et de vérifier son bon fonctionnement en l’exécutant.

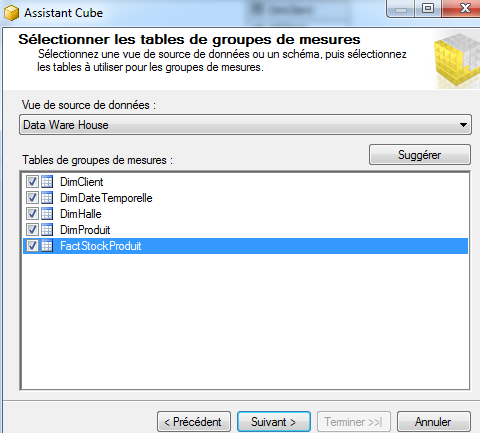
Grace au bouton « navigateur »  il est possible de visualiser le résultat de la création de notre dimension.



On retrouve bien dans notre exemple la hiérarchie souhaitée pour la dimension des halles. Comme dans la table DimHalle il y avait des données avec des nuls pour certains parcs nous les retrouvons donc également.

### Création du cube

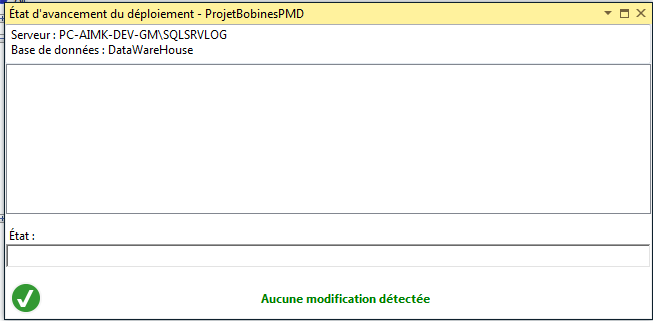
Pour la création du cube à proprement dit, j’ai utilisé l’assistant de création en le paramétrant de la façon suivante. J’ai dû donc choisi la création du cube à partir des tables existantes et en les sélectionnant dans la vue des sources de données précédemment créée.



### Déploiement du cube

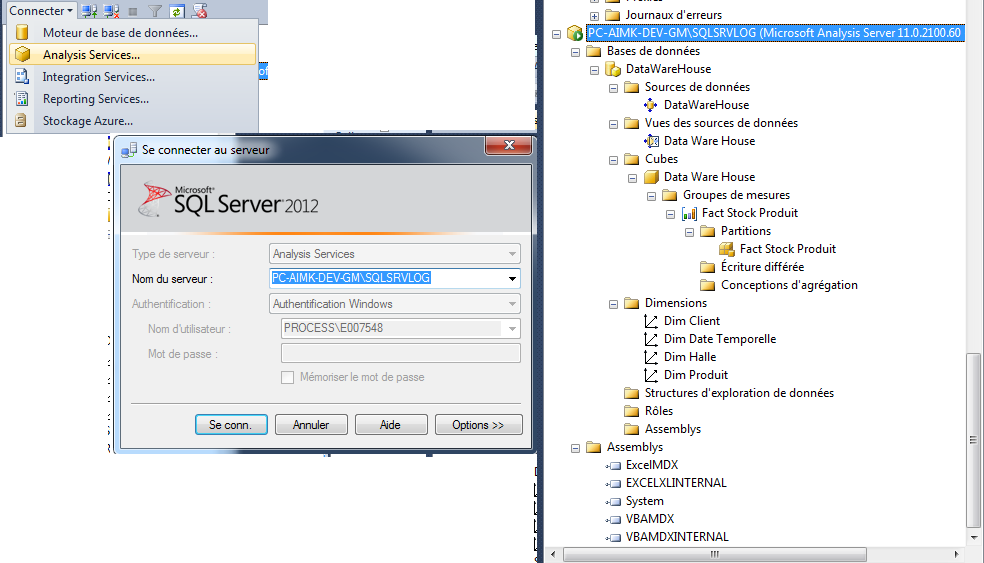
Pour déployer le cube dans le serveur de base de données il suffit de cliquer sur déployer au niveau du projet « projetbobinePDM ».

Une popup apparait pour indiquer que cela s’est bien passé.



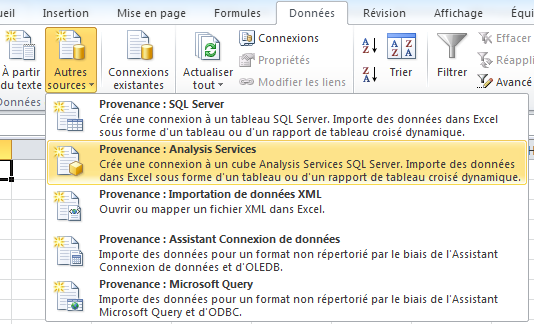
Ensuite dans SQL server management studio il est possible de visualiser les objets déployés en se connectant à la base de données Analysis Services et en sélectionnant le serveur de base de données du projet.

On retrouve donc l’ensemble des objets du cube de nous avons créés précédemment.

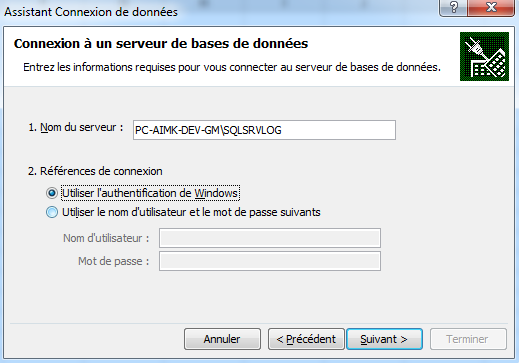


### Exécution du cube avec Excel

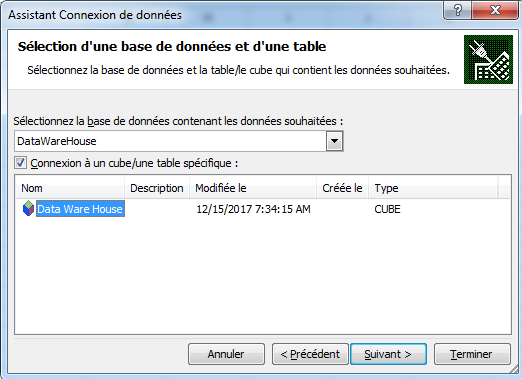
Il est possible d’utiliser directement le cube avec un fichier Excel afin de générer des rapports.



Pour cela il faut créer un nouveau fichier Excel et dans l’onglet « données » sélectionner autres sources puis la provenance : Analysis Services.

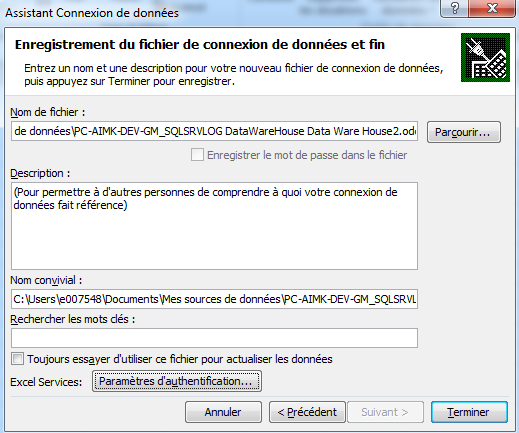


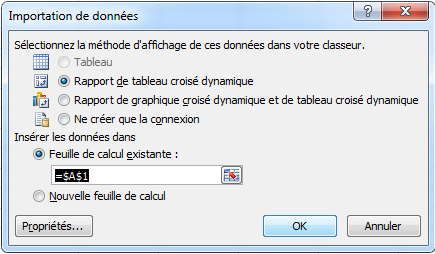
Déclarer la connexion au serveur de la base de données



Enregistrer le fichier de connexion à la base de données. Dans notre cas j’ai pris une identification Windows mais il est possible créer une identification SSS (Secure Store Service) est un service partagé qui fournit le stockage et le mappage des informations d'identification telles que les noms de compte et les mots de passe.

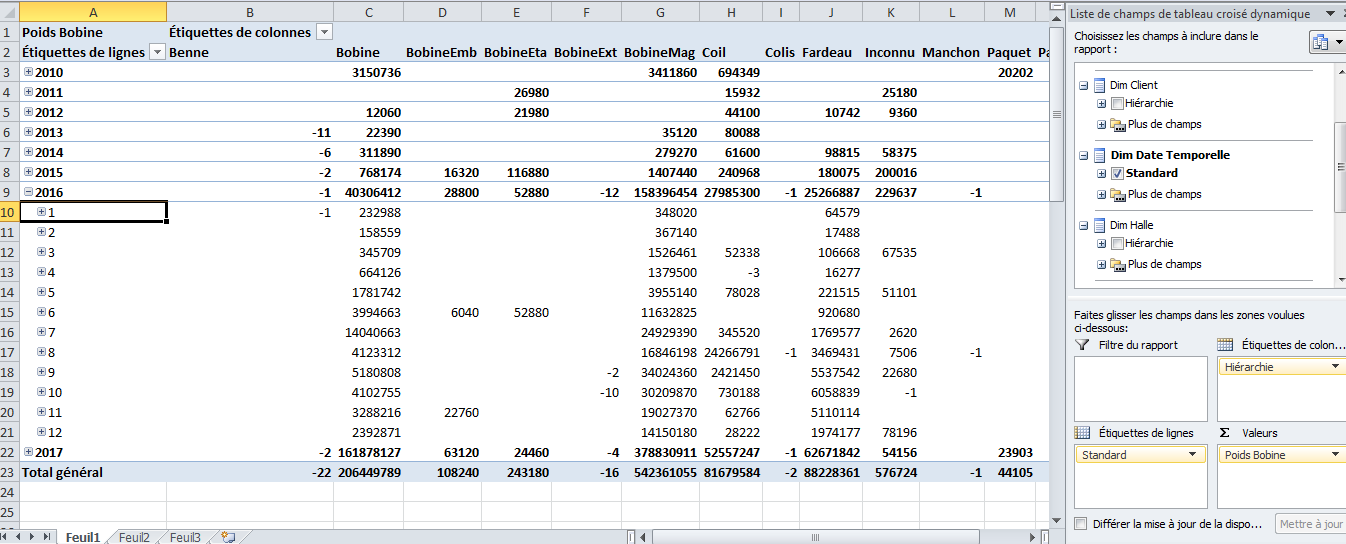
Puis sélectionner la base de données et le cube qui a été précédemment déployé.





On choisit ensuite où on veut importer les données et sous quelle forme. Dans notre nous prenons le tableau croisé dynamique.

La constitution du rapport reste en soit assez simple car on utilise ici les standards des tableaux croisés dynamique. Dans l’exemple ci-dessous j’ai sélectionné pour l’affichage des lignes le champ de date standard créé dans le cube afin d’avoir la hiérarchie des années, des mois puis des jours. Pour les colonnes j’ai pris la hiérarchie de la table de dimension DimProduit afin d’avoir la vision complète de l’ensemble des produits par type.



Enfin en valeur, j’ai pris le poids des bobines. Ainsi en sélectionnant les jours, mois et années on a automatiquement le rapport qui se met à jour.

Il peut ensuite être sauvegardé afin de refaire une visualisation des données en fonction des mises à jour de la base de données.

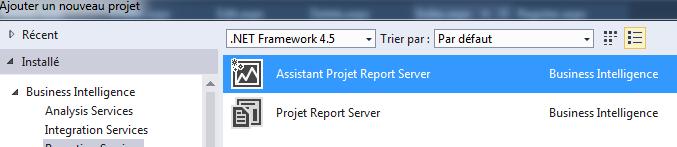
### Conclusion

Pour aller plus loin dans mon projet de créer d’autres cubes reprenant les données de calculs des temps de transfert des produits dans les halles par exemple.

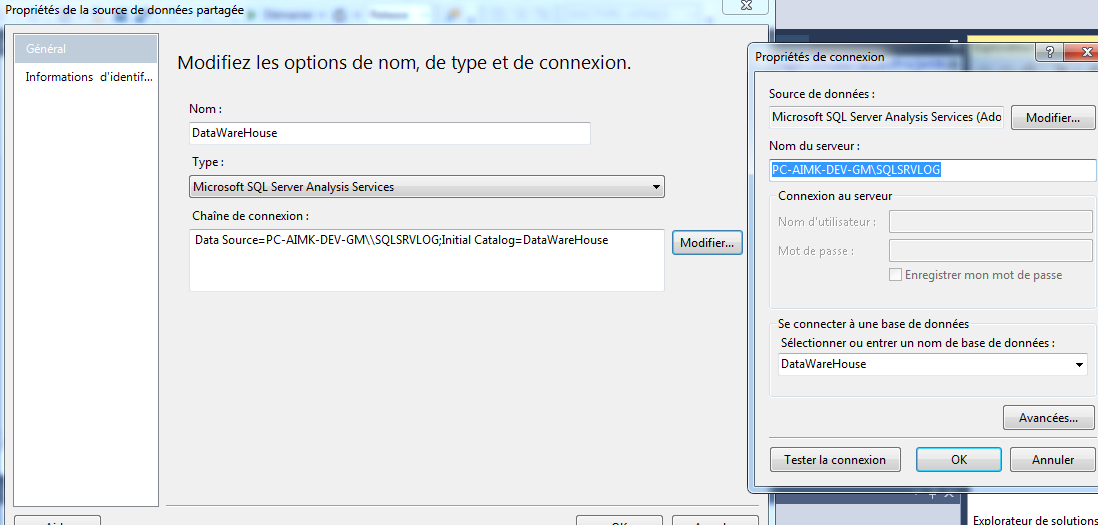
## Création des rapports avec SSRS

### Création des rapports

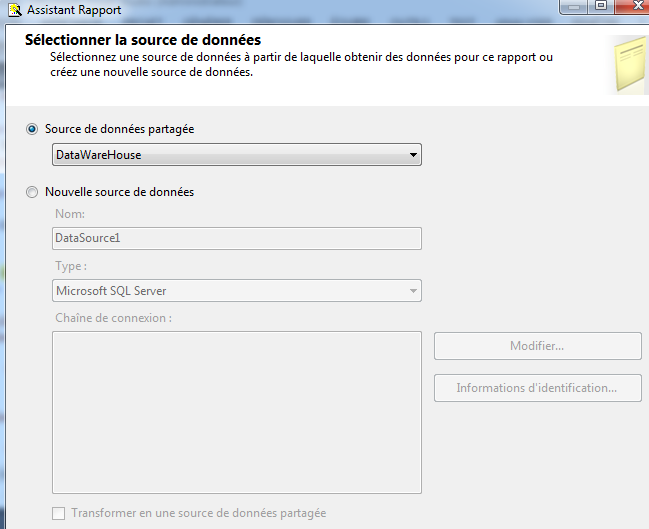
Visual Studio permet de créer des rapports automatique grâce à la couche SSRS (SQL Server Reports Services) qui s’interface directement avec le cube précédemment créé. Pour cela il faut créer un nouveau projet dans Visual Studio.



Il faut ensuite configurer la connexion à la base de données.



Enfin on peut créer un rapport qui nous donne le poids des produits en fonction du parc en utilisant l’assistant et en le configurant comme ci-dessous.



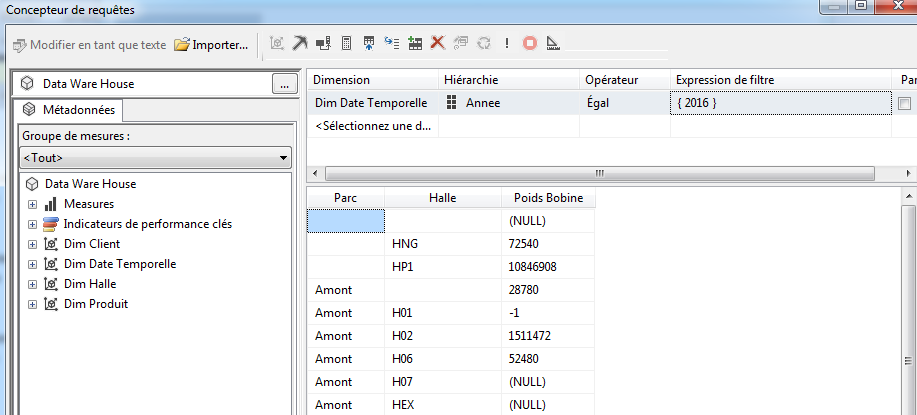
Pour notre rapport des poids bobines en fonction des parcs et des halles on sélectionne les différentes données de façon à les placer à droite de l’interface.

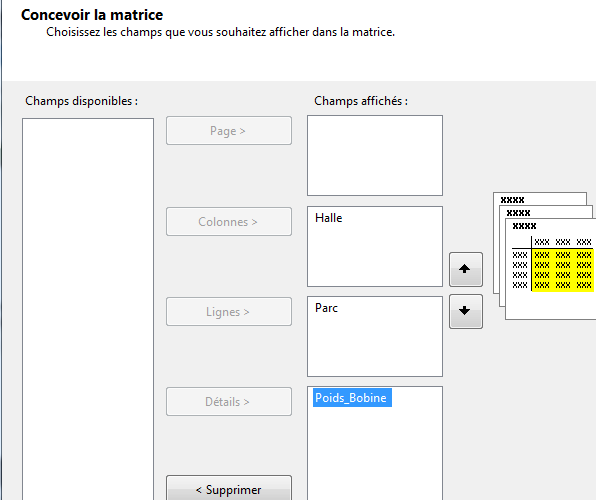
Dans notre nous avons sélectionné l’année 2016.

Puis on sélection le type d’affichage entre tabulaire et matriciel (notre exemple).

Pour la sélection de la source de données on prend DataWareHouse créé précédemment.

Ensuite il faut utiliser le gestionnaire de requête.



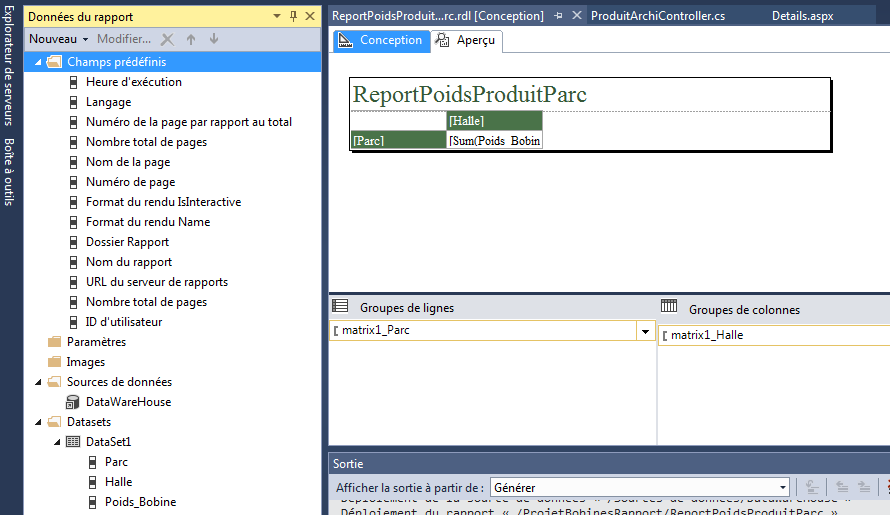


On remplit ensuite les différentes parties de la matrice (sur le même principe que le tableau croisé dynamique sous Excel).

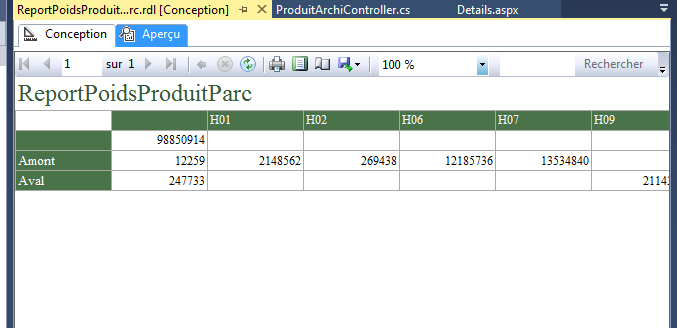
En colonne on met la halle, en ligne le par cet dans le détail on retrouve le poids bobines.

On peut ensuite choisir le style près définie. Dans notre exemple j’ai choisi le style forêt.

Une fois le rapport créé, il est possible d’agrémenter le rapport avec d’autres éléments prédéfinis.

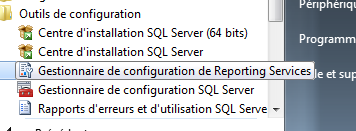


Le mode aperçu permet de visualiser le contenu du rapport et de se donner une idée du résultat final.

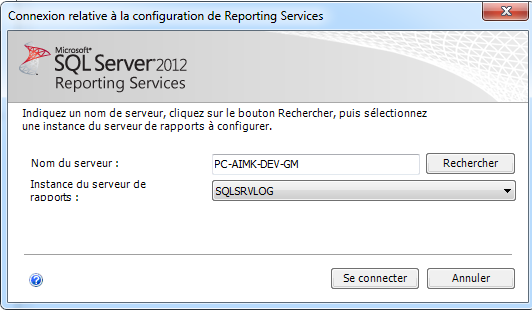


Avant de pouvoir déployer le rapport sur le serveur il faut configurer le Reporting service.

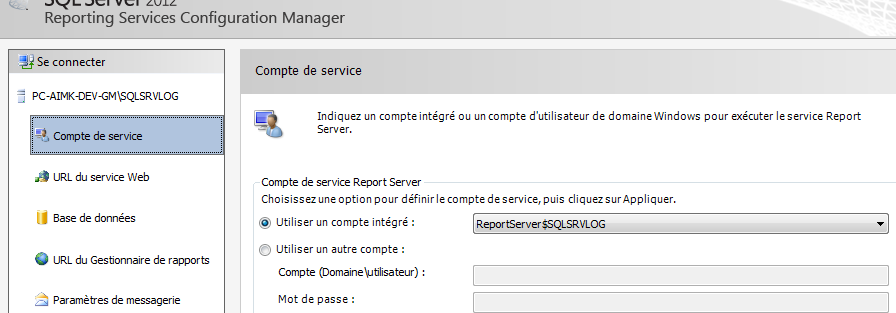
### Configuration de Reporting Service



Avant de pouvoir déployer le rapport sur le serveur il faut configurer le Reporting service en sélectionnant l’utilitaire dans les outils de configuration de SQL Server 2012.



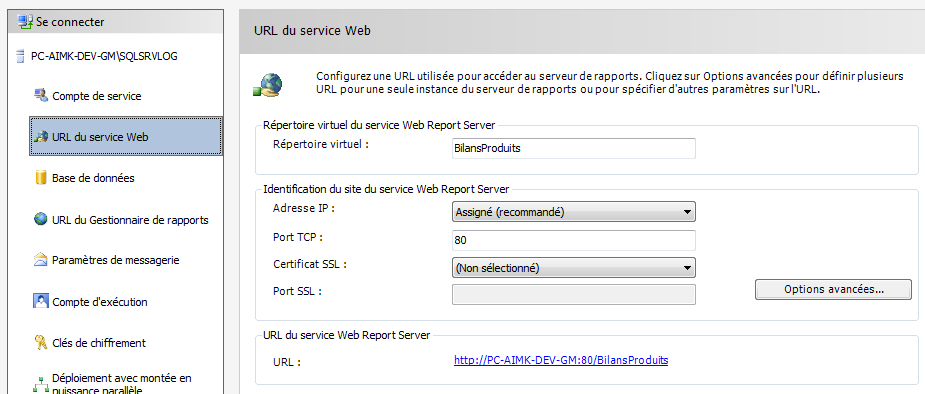
On sélectionne ensuite la configuration à la base de données souhaitée. Dans notre cas nous nous connectons à l’instance SQLSRVLOG.

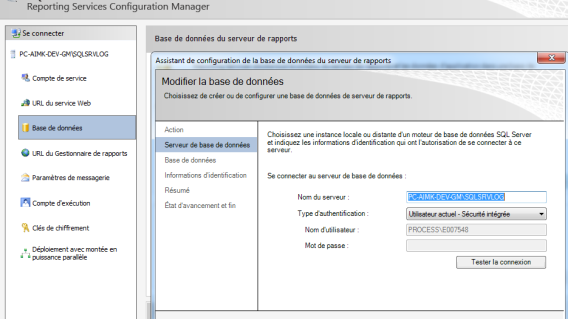


La configuration du compte de service permet de créer le compte qui va exécuter le service report server. Par défaut j’ai pris le compte intégré qui est conseillé.

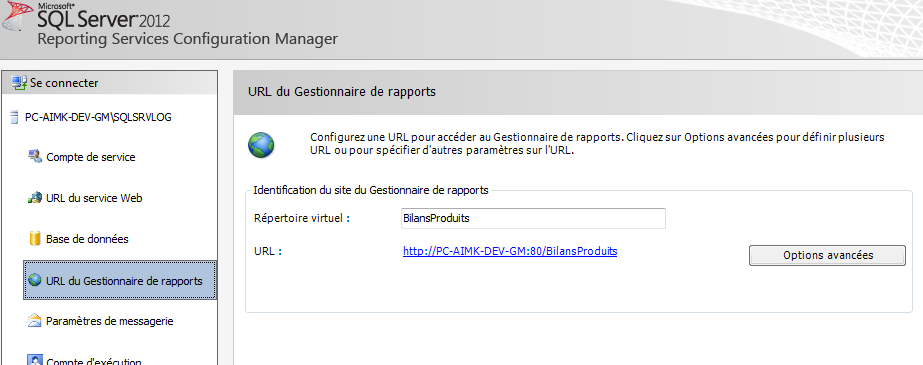
Afin de pouvoir déployer les rapports sur notre serveur web il faut configurer le service web.

Pour cela on renseigne le nom du répertoire virtuel ainsi que le port du service. On récupérer ainsi la future adresse web où l’on pourra consulter les rapports. Les options avancées permettent également de gérer la couche SSL (TODO) pour la sécurisation du site.





J’ai ensuite configuré la base de données sur laquelle le service report server se connectera afin de récupérer les rapports.

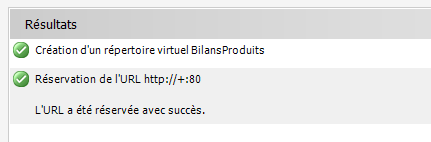


La configuration de l’URL du gestionnaire peut de configurer le site. Par défaut j’ai pris le même répertoire que le site web configuré précédemment.

Pour les autres onglets de paramètre j’ai laissé la configuration par défaut.

Pour la partie sécurité, Reporting Services gère une clé symétrique de chiffrement pour les données d’identification, les chaînes de connexion et les données sensibles de la base de données Report server. J’ai sauvegardé la clé de chiffrement dans un fichier afin de l’archiver. TODO expliquer clé symétrique.

A chaque fois pour chaque onglet, la validation des modifications met à jour le server. Cela permet de savoir au fur et à mesure si les paramètres sont renseignés correctement et ceci grâce aux différents messages affichés en bas des onglets.

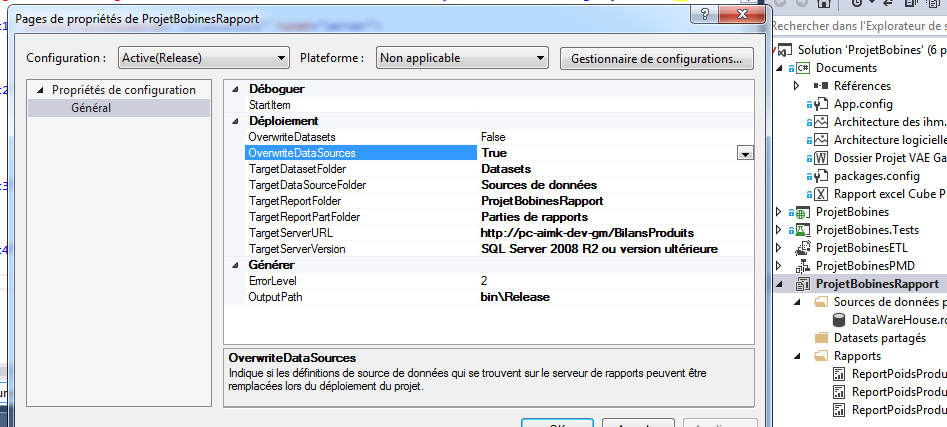


Après avoir appliqué les paramètres le serveur web reporting services est configuré. On peut maintenant passer au déploiement des rapports dans Visual Studio.

### Déploiement des rapports

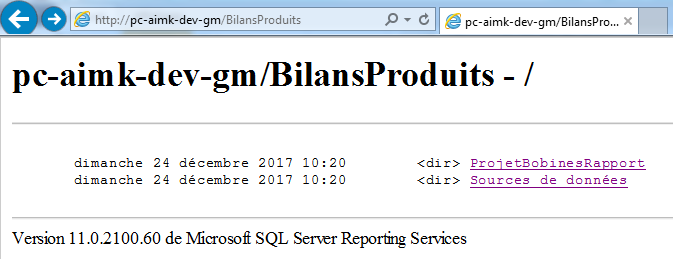
Avant de déployer les rapports il faut configurer les propriétés du projet ProjetBobinesRapport que nous avons créé au début de ce chapitre sur la création de rapports avec SSRS.

Dans ces propriétés j’ai donc modifié le OverwriteDataSources afin d’autoriser le remplacement des rapports présents sur le serveur lors du déploiement et le TargetServerURL afin de spécifier l’adresse et le répertoire de destination des rapports à déployer. Cette adresse web correspond à celle crée lors de la configuration du Reporting Services.

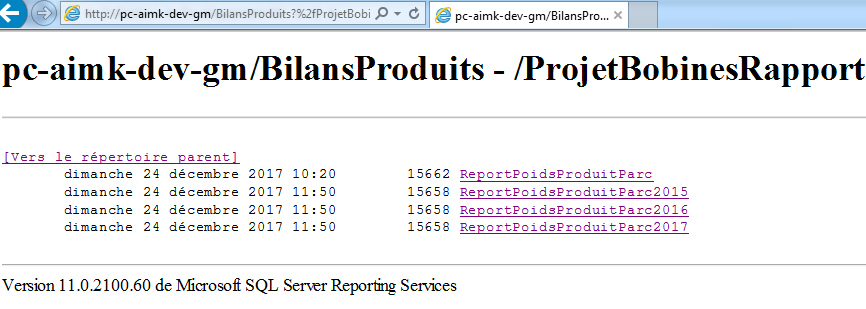


### Consultation des rapports

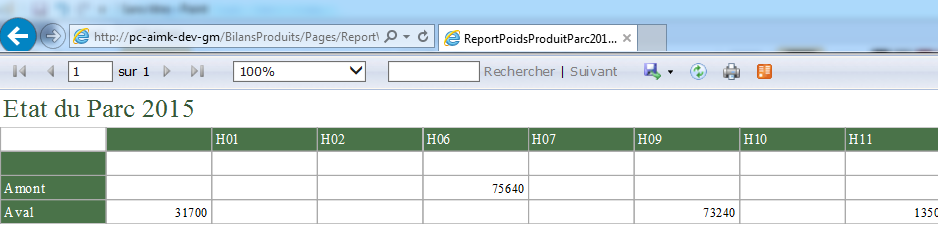
Il est possible de consulter les rapports directement avec internet explorer en mettant l’URL suivante <http://pc-aimk-dev-gm/BilansProduits>.



On peut ainsi accéder aux rapports en cliquant sur le lien ProjetBobinesRappots et à la consultation de la source de données.



Si l’on veut consulter le rapport des poids de produit dans les parcs en 2015 il suffit de cliquer sur le lien en rapport. Cela permet d’afficher le rapport correspondant qui a été créé au tout début du chapitre.



L’interface web permet à l’utilisateur d’imprimer sont rapport mais aussi de le sauvegarder dans différents format. Pour des tableaux de données par exemple, la gestion des boutons des pages suivantes/précédentes par exemple est réalisée automatiquement.

Pour mon application l’accès aux rapports pour les utilisateurs se fera directement via le site web que j’ai créé afin de leurs éviter de changer de page web et d’environnement.

### Conclusion

Pour mon application l’accès aux rapports pour les utilisateurs se fera directement via le site

# La sécurité

## Les comptes utilisateurs

Lorsque l’utilisateur créé son compte via la page web, certaines informations sont cryptées en base de données comme son mode de passe.

Pour cela il a été utilisé la Classe WebSecurity qui fournit des fonctionnalités de sécurité et d'authentification pour les applications ASP.NET Web Pages, y compris la possibilité de créer des comptes d'utilisateur, de connecter et de déconnecter les utilisateurs, de réinitialiser et de modifier les mots de passe, ainsi que d'exécuter les tâches associées.

## Protection des pages web

L’application utilise le jeton anti-contrefaçon pour protéger l’application contre la falsification de requête intersites. Pour utiliser cette fonctionnalité, la méthode AntiForgeryToken est appelée à partir des controlers.

## Gestion des droits

Une meilleure solution consiste donc à contrôler les droits par rapport à des “rôles” et à associer les utilisateurs à ces rôles soit en passant par une base de données soit par l’intermédiaire de l’Active Directory (la liste des utilisateurs concernés étant ainsi définie à l’extérieur du code). ASP.NET propose une API pour gérer des rôles et fourni également un ensemble de fournisseurs pour les stocker (dont un pour SQL et un pour Active Directory), le tout permettant de simplifier l’association entre utilisateurs et rôles. Avec cela, nous pourrions adapter notre code pour autoriser uniquement les utilisateurs appartenant au rôle “admin” à accéder à l’URL /Dinners/Create :

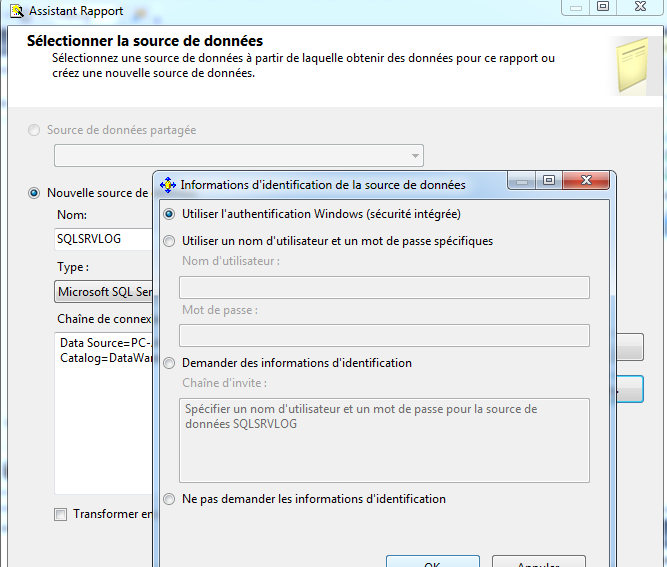
[Authorize(Roles="admin")]

public ActionResult Create() {

...

}

SSRS



## Conclusion

Il existe une multitude de solutions permettant de protéger des applications informatiques ou gérer des droits utilisateurs.

Pour ma part nous utilisons par habitude la gestion des comptes utilisateurs par un domaine en utilisant active directory. Les comptes sont donc associés à des groupes d’utilisateurs en fonctions de privilèges déterminés.

Les machines sont également inscrites dans ce domaine afin de profiter de la centralisation des groupes de fonctionnalités.

# Qualité du code, gestion des versions et des fiches recettes

Lors de la réalisation du projet, il était important de gérer le versionning du code de l’application et de la documentation (comme ce rapport).

Pour cela il existe différentes méthodes :

* Copier et renommer le ou les répertoires des sources avant développement.
* Utiliser un utilitaire de gestion de code sources.

## Choix de la gestion des versions

### Copie et renommage de répertoires

Cette méthode de versionning obligée chaque développeur à copier les sources soit en local soit sur un serveur, puis de remettre à jour manuellement les fichiers qu’ils avaient modifiés. De plus si plusieurs programmeurs utilisés le même fichier à un moment donné, rien ne leur indiqué s’il était utilisé par un autre membre de l’équipe projet.

Autrefois utilisée elle a montré ces limites depuis l’arrivée des projets en nouvelles technologies où les équipes projets sont relativement importante voir même distante de plusieurs centaines de kilomètres.

Bien évidemment cette solution n’a pas été retenue pour mon projet.

### Utilitaire de gestion de code sources

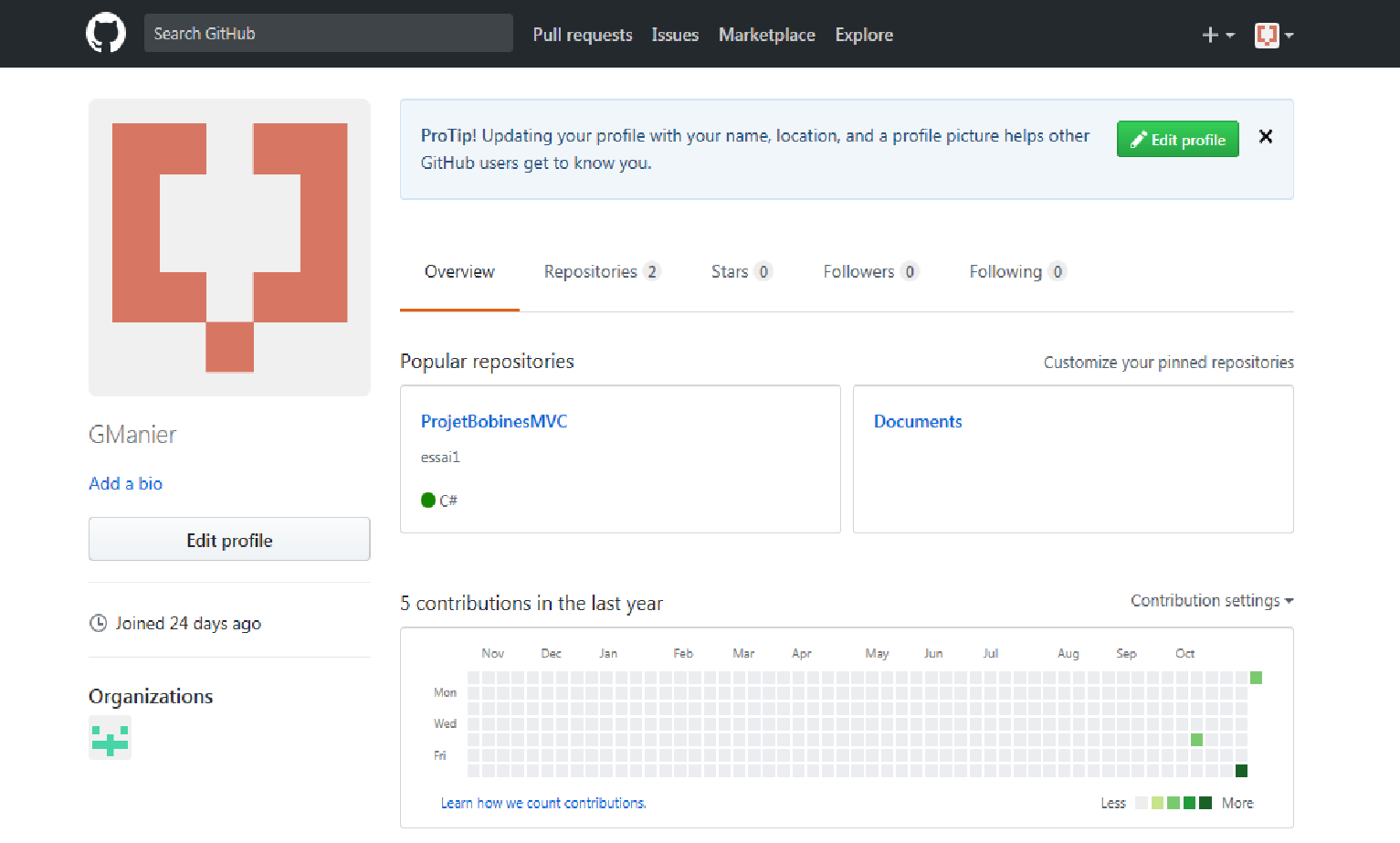
Sur le marché il existe plusieurs utilitaires de gestion de code sources et gestion des versions de fichiers. Pour n’en citer que quelques un, nous avons TFS (Team Foundation Server), Visual sourceSafe, Git, GitHub et BitBuchet.

Dans mon entreprise nous utilisons TFS (Team Foundation Server) qui permet de gérer les différents projets des différentes usines. Ce serveur est géré par nos équipes supports d’infrastructure et de développement logiciel. Pour des raisons d’accès à nos serveurs et à nos réseaux d’entreprise je n’ai pas choisi cet utilitaire pour la gestion des sources de mon projet.

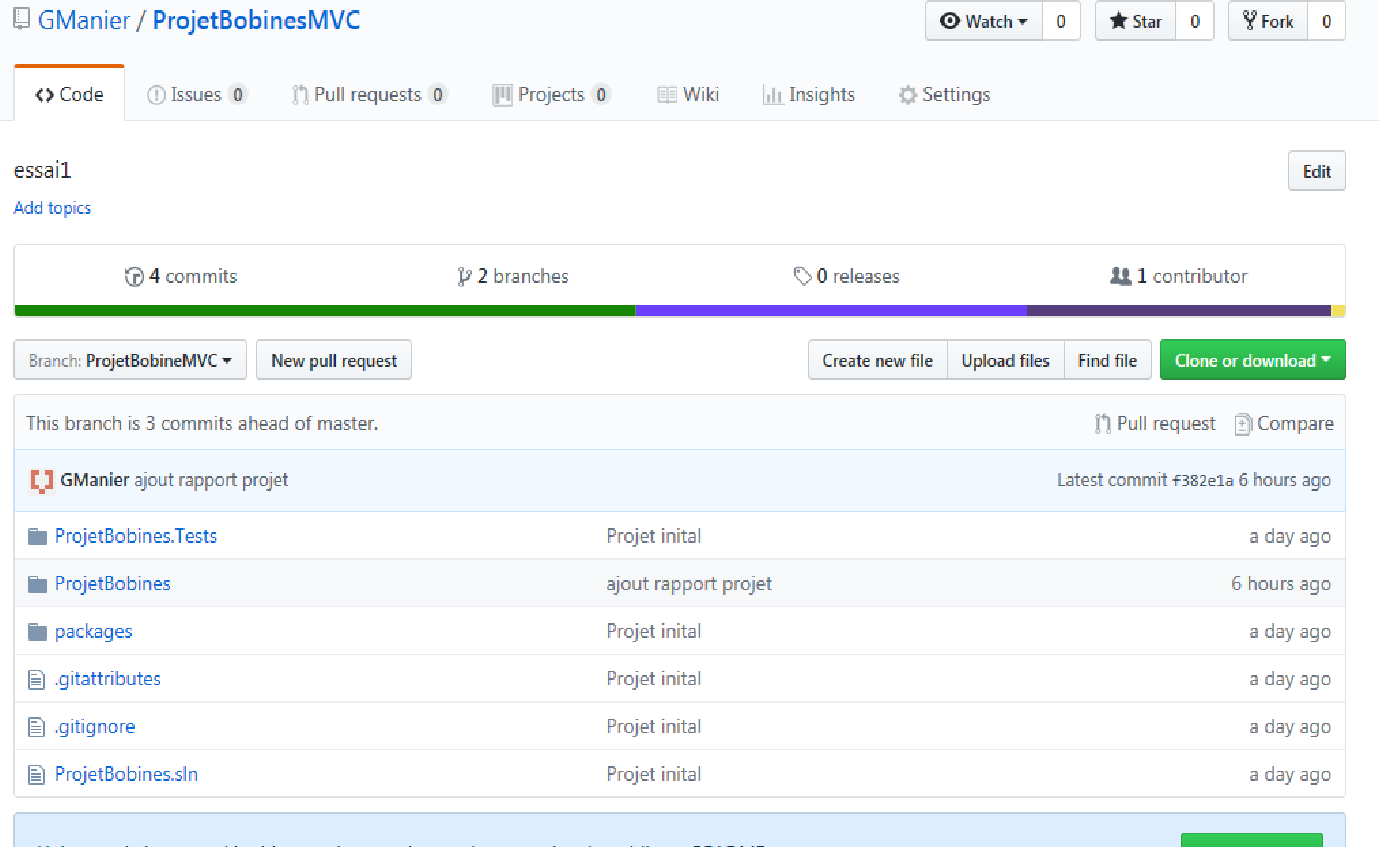
Comme Mon tuteur de projet devait également pouvoir au code source et aux documents que je créé, j’ai opté pour GitHub. L’avantage est qu’il s’interface directement avec Visual Studio 2013 (VS2013) et que l’extraction ou l’archivage des fichiers se fait naturellement avec la même souplesse que TFS.

### GitHub

Le choix s’est donc porté sur ce service web d’hébergement et de gestion pour le développement de logiciel. Pour pouvoir utiliser ces services j’ai donc créé un compte pour lequel j’ai dû spécifier s’il s’agissait d’une utilisation privée (Attention maintenant) ou public.



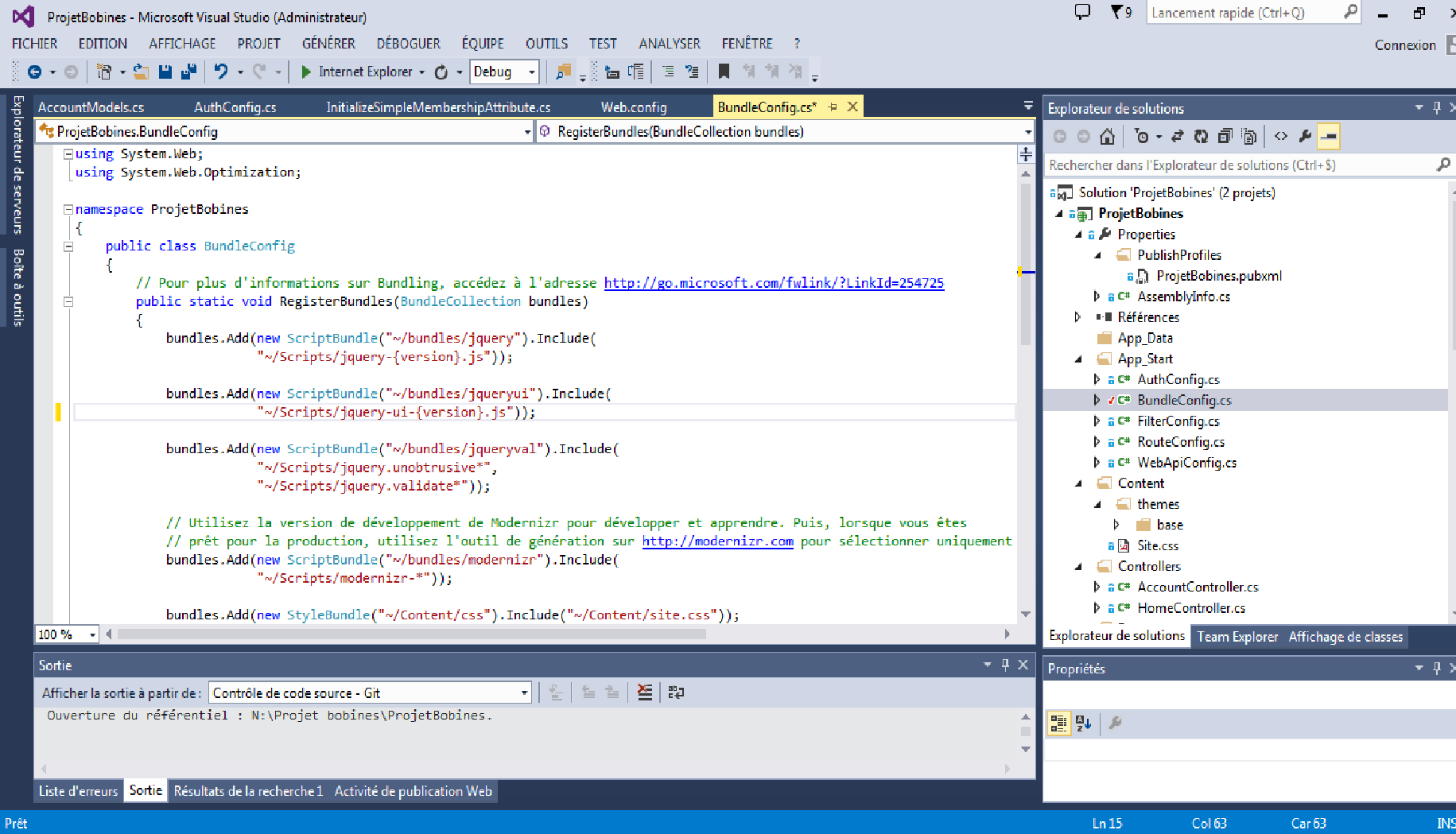
Une fois le compte créé, j’ai dû définir des « Repositories » afin de pouvoir y inclure les différentes solutions Visual Studio du projet. Il est possible ensuite de créer des branches de développements afin de gérer les versions en fonction soit des développements en cours, soit en recette, soit en production.



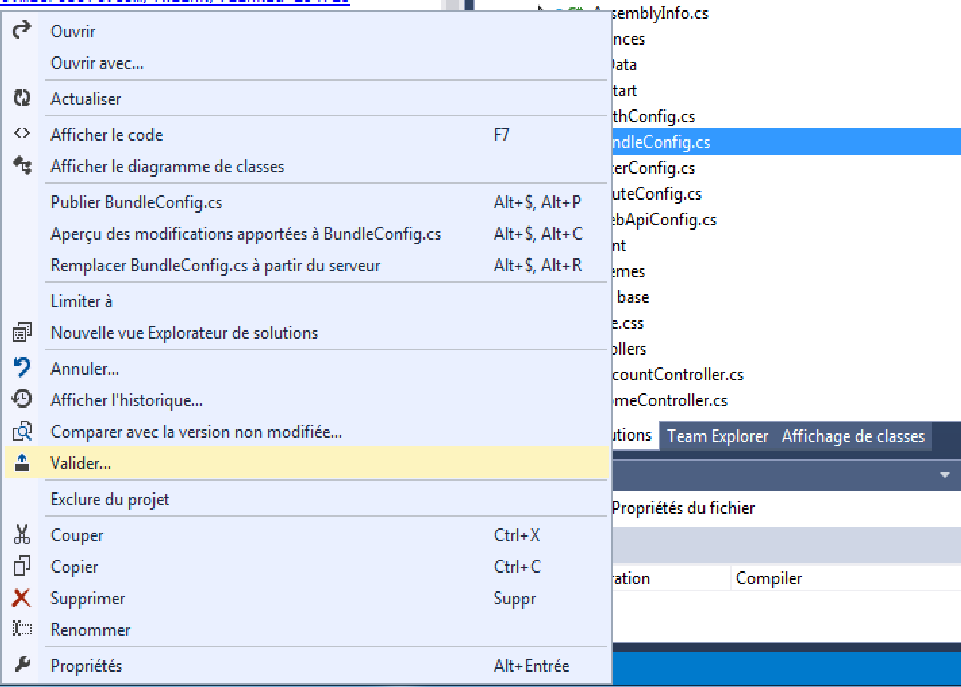
L’utilisation avec Visual Studio 2013 est assez simple car celui-ci propose naturellement 2 gestionnaires de code source, TFS ou Git.

En sélectionnant Git, il faut déclarer son compte GitHub puis le « repositories » et branche sur laquelle on souhaite archiver ses projets.

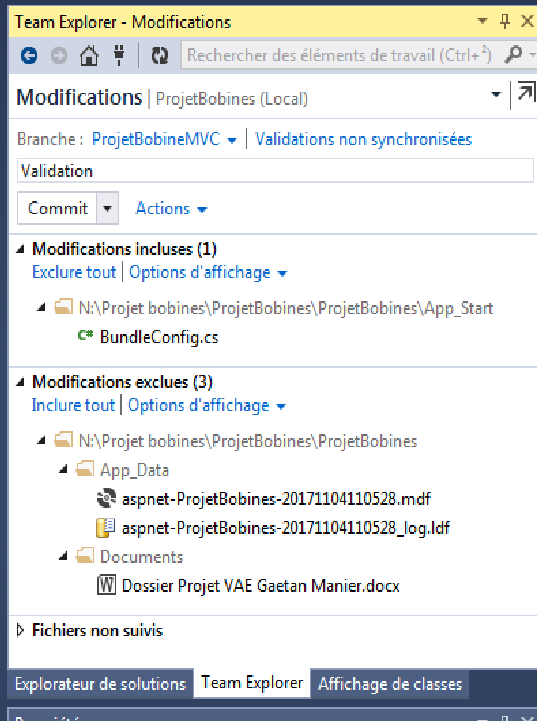
Dans VS2013 les fichiers extraits sont indiqués par une coche en rouge et ceux non extraits sont avec un cadenas bleu.



Pour archiver ses modifications il faut cliquer sur le fichier extrait, ou l’ensemble du projet et sélectionner « valider ».



Cette action nous envoie directement sur l’onglet « Team explorer » où il faut mettre un commentaire et cliquer sur commit.



Cela a pour effet d’archiver localement les modifications. Pour pouvoir les archiver sur le service web GitHub et partager ces modifications il valider la synchronisation.



## Les fiches de recettes

Pour tester les applications informatiques plusieurs méthodes existent comme les fiches de recette, les tests unitaires.

### Les fiches de recette

Elles permettent au fournisseur et au client de valider le bon fonctionnement de l’application en fonction des attendus souhaités et de ce qui a été écrit dans le cahier puis dans l’analyse fonctionnelle du projet. Les fiches de recette sont principalement écrites par le client dans lesquelles plusieurs scénarios sont écris.

Le client passe alors en revue l’ensemble de l’application afin de tester la globalité de son comportement. Cela peut paraît parfois fastidieux surtout si le client reprend l’intégralité des fiches de recettes des qu’il a repéré un ensemble de non conformités dans le souci de s’assurer qu’il n’y ait pas d’effet de bord suite aux dernières corrections.

Dans mon projet j’ai réalisé les recettes suivantes par exemple :

* Consultation des données des produits :
  + Action  : Sélection d’un produit dans la liste
  + Résultat  : Affichage des différentes colonnes du produit
* Création d’un produit :
  + Action  : Accès à l’écran de création en respectant les droits requis

Remplissage des différents champs et validation des données

* + Résultat  : Vérifier que le produit a bien été créé en base de données.

### Les tests unitaires

L’utilisation du modèle MVC permet d’implémenter des tests unitaires dans les applications WEB. Lors de la création de mon application j’ai également créé un projet de test unitaire qui permet de tester dans un premier temps le contrôleur de compte et le contrôleur Home.

Les tests unitaires ASP.NET.MVC appellent directement les méthodes des contrôleurs MVC créés dans l’application.

### Conclusion

Il existe évidemment d’autres utilitaires permettant de réaliser des tests d’application informatique comme par exemple Postman et Swagger.

# Conclusions

Difficultés vue les techno exmployée

Surmonté les pb doc internet et travail

Apport sur la vision que l on peut avoir pour construire nos nouvelles applications. Proposer des solutions technqiues aux equipes supports.

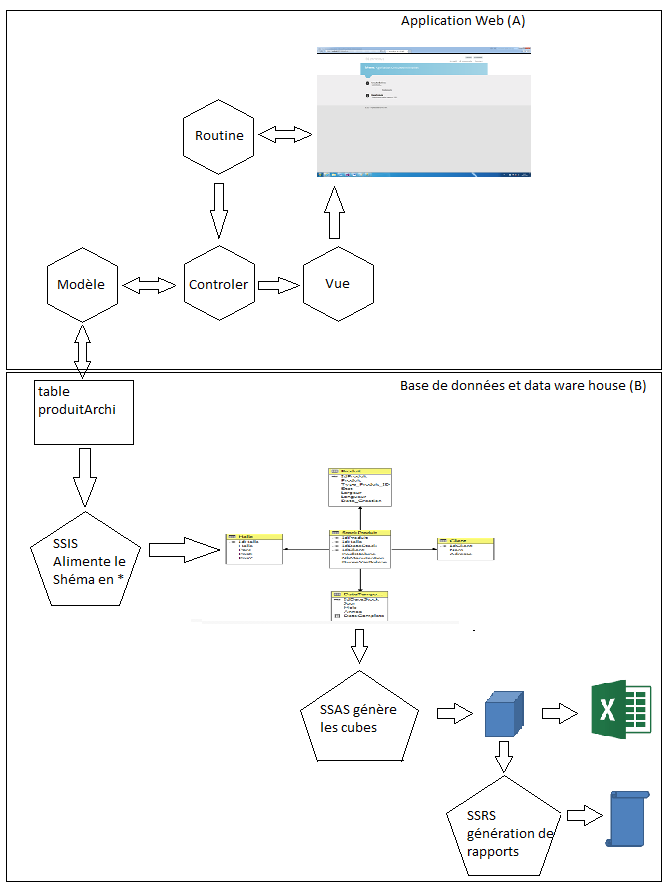
Nos clients on de plus en plsu besoin de rapports divers afin d’améliorer leurs process. Nous utilisons principalement SAPBi4 pour la partie reporting mappée sur une base de données de type serveur de données technique. L’apporche faite dans ce projet donne une autre dimension à ce que je peux proposer en terme de reporting.

# Annexes

## Annexe 1 : Arborescence des vues



## Annexe 2 : Architecture globale



## Annexe 3 : Architecture de l’application