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 Le cahier des charges 4](#_Toc501288817)

[2 Fonctionnalités 5](#_Toc501288818)

[2.1 Introduction 5](#_Toc501288819)

[2.2 L’application WEB 5](#_Toc501288820)

[2.2.1 La page d’accueil 6](#_Toc501288821)

[2.2.2 La gestion des accès utilisateur 6](#_Toc501288822)

[2.2.3 La page « Contact » 7](#_Toc501288823)

[2.2.4 La page « à propos de » 7](#_Toc501288824)

[2.2.5 La page « Produits Archi» 7](#_Toc501288825)

[2.2.6 La page « Bilans bobine» 9](#_Toc501288826)

[2.2.7 Pour aller plus loin 9](#_Toc501288827)

[3 Architecture du logiciel 10](#_Toc501288828)

[3.1 Introduction 10](#_Toc501288829)

[3.2 Architecture 10](#_Toc501288830)

[3.3 Le serveur Web 12](#_Toc501288831)

[4 Architecture de la base de données 13](#_Toc501288832)

[4.1 Configuration du serveur de base de données 13](#_Toc501288833)

[4.1.1 Compte pour l’application Web 13](#_Toc501288834)

[4.1.2 Compte pour le SSAS 14](#_Toc501288835)

[4.2 Introduction aux Data WareHouse 14](#_Toc501288836)

[4.3 Choix du schéma 15](#_Toc501288837)

[4.4 Alimentation du Dataware house 16](#_Toc501288838)

[4.4.1 Configuration de l’ETL avec SSIS 16](#_Toc501288839)

[4.4.2 Déploiement de l’ETL 20](#_Toc501288840)

[4.4.3 Exécution de l’ETL 22](#_Toc501288841)

[4.4.4 Pour aller plus loin 24](#_Toc501288842)

[4.5 Création et gestion du cube avec SSAS 24](#_Toc501288843)

[4.5.1 Utilisation du cube avec Excel 24](#_Toc501288844)

[4.6 Création des rapports avec SSRS 26](#_Toc501288845)

[4.7 Alimentation du cube 27](#_Toc501288846)

[5 La sécurité 28](#_Toc501288847)

[5.1 Cryptage des comptes utilisateurs 28](#_Toc501288848)

[5.2 Gestion des droits 28](#_Toc501288849)

[6 Qualité du code et gestion des versions 30](#_Toc501288850)

[6.1 Choix de la gestion des versions 30](#_Toc501288851)

[6.1.1 Copie et renommage de répertoires 30](#_Toc501288852)

[6.1.2 Utilitaire de gestion de code sources 30](#_Toc501288853)

[6.2 GitHub 31](#_Toc501288854)

[7 Conclusions 34](#_Toc501288855)

# 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 produits des bobines d’acier et de suivre leurs manutentions dans les halles du site.

# Fonctionnalités

## Introduction

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

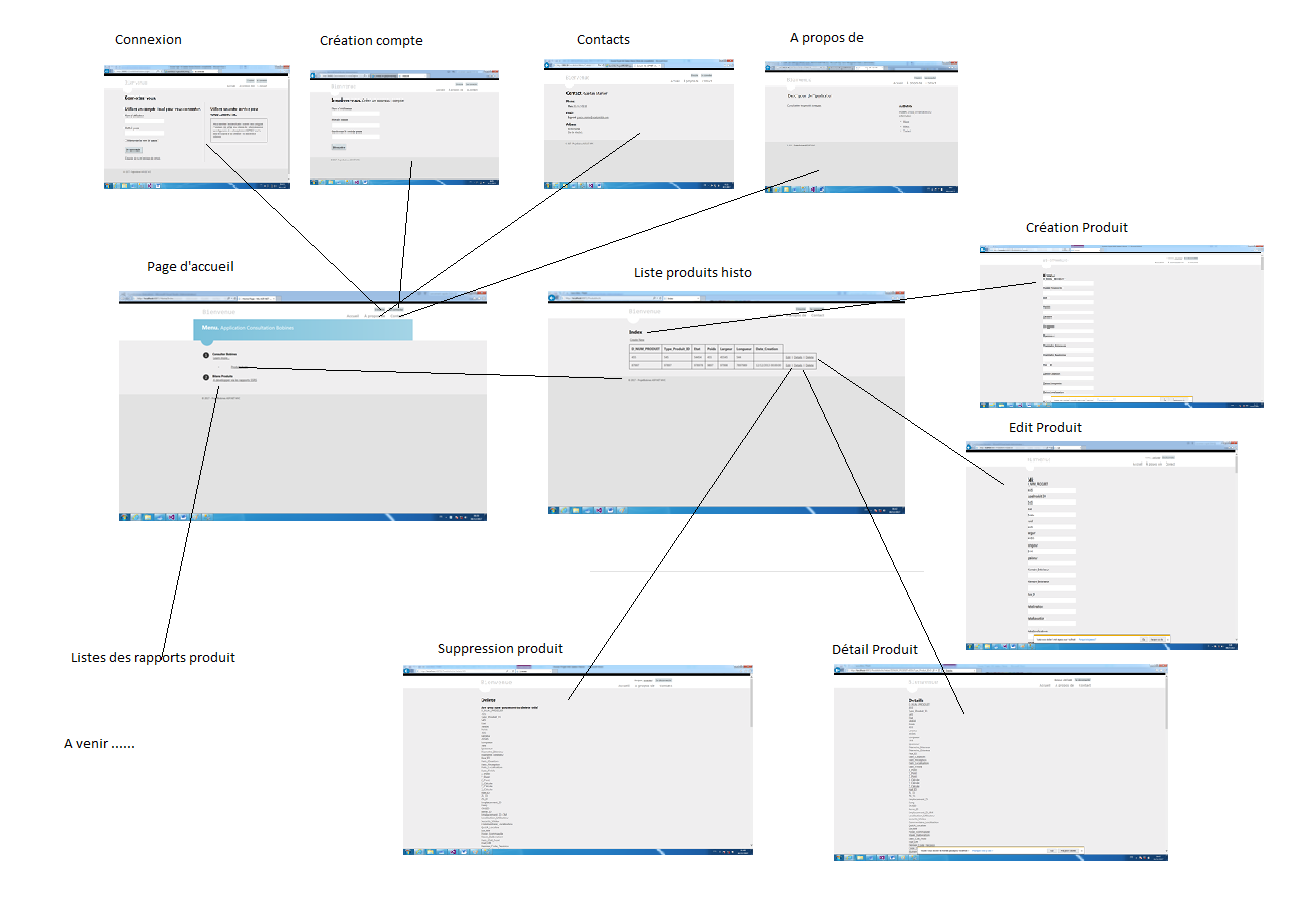
L’application serveur permet de consulter ces informations à partir de pages web. Il est possible de modifier ces fiches produit grâce à divers actions : création, modification et suppressions.

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, halle … TODO

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.

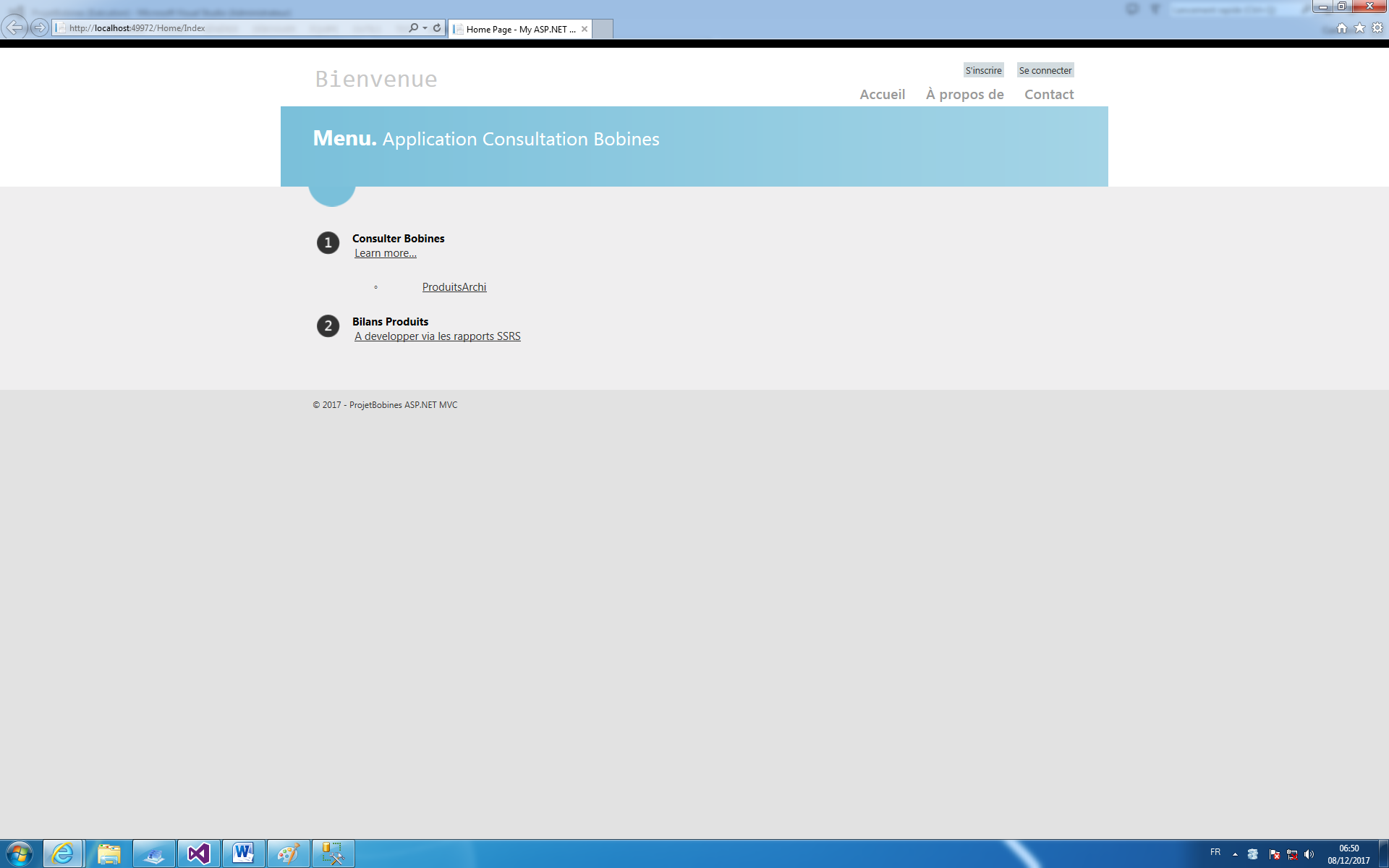
Une application mobile permet de consulter également ces informations par smartphone afin de pouvoir faire une vérification des produits directement dans les halles.

## L’application WEB



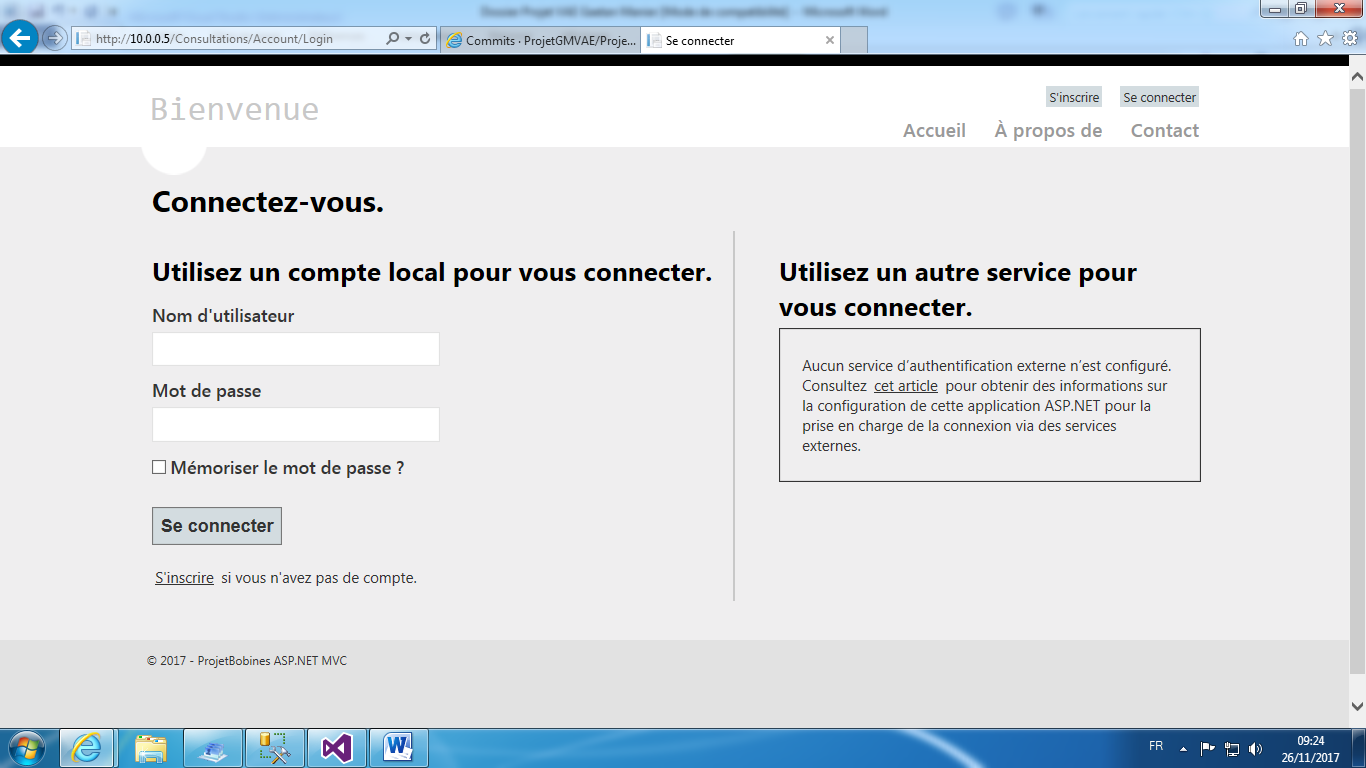
### La page d’accueil

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.

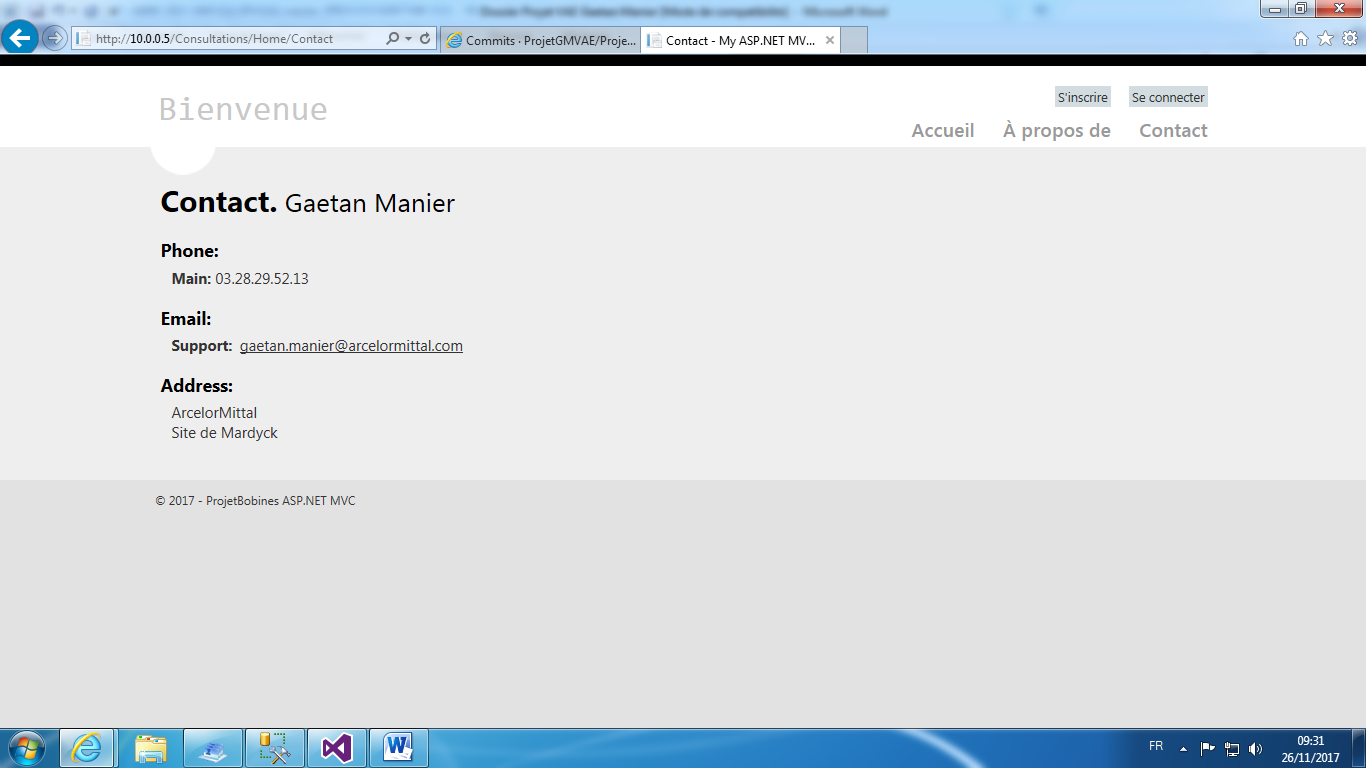
Lors de la création d’un nouveau compte, l’utilisateur doit envoyer un mail à l’administrateur du site afin de définir le rôle de son compte. L’accès aux différentes vues et aux fonctionnalités (création, suppression, mise à jour) se fait grâce à l’attribution de rôles sur les comptes.

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

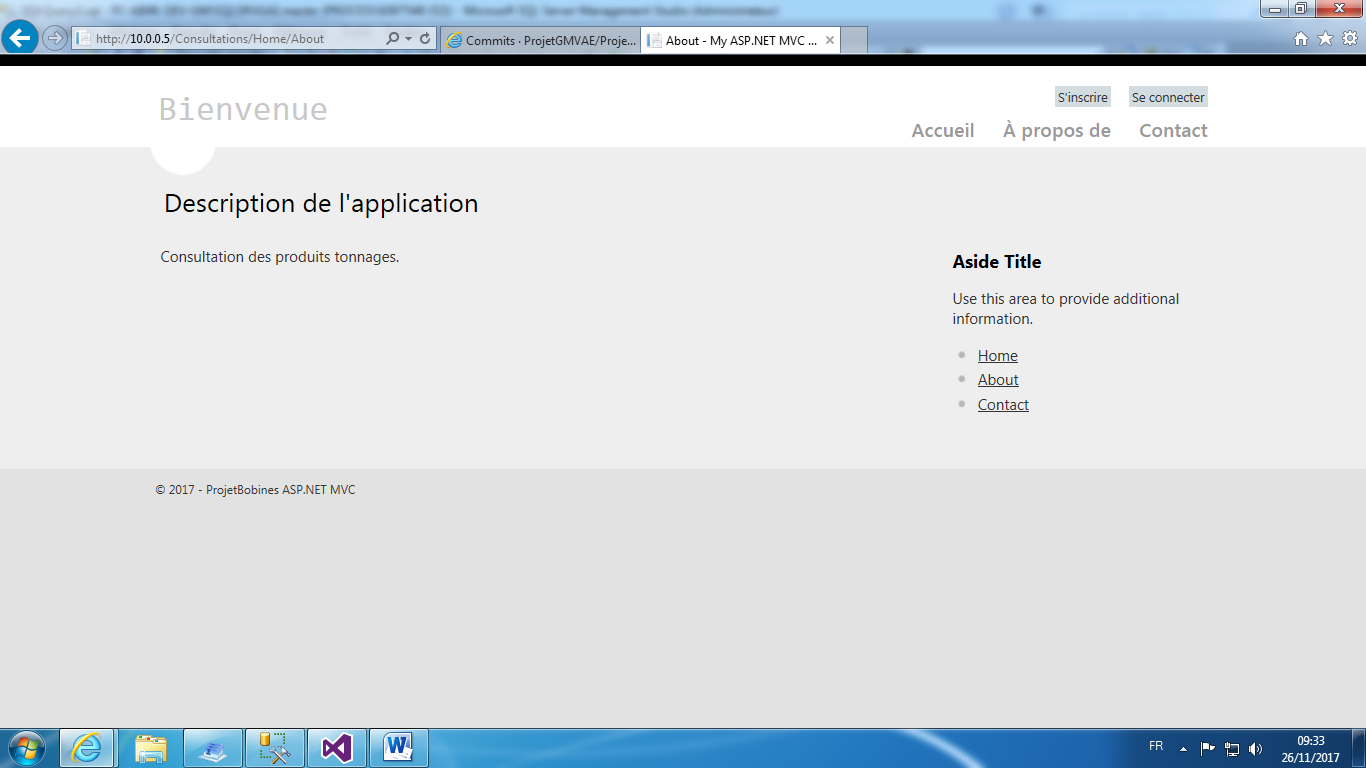
### La page « Contact »

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.



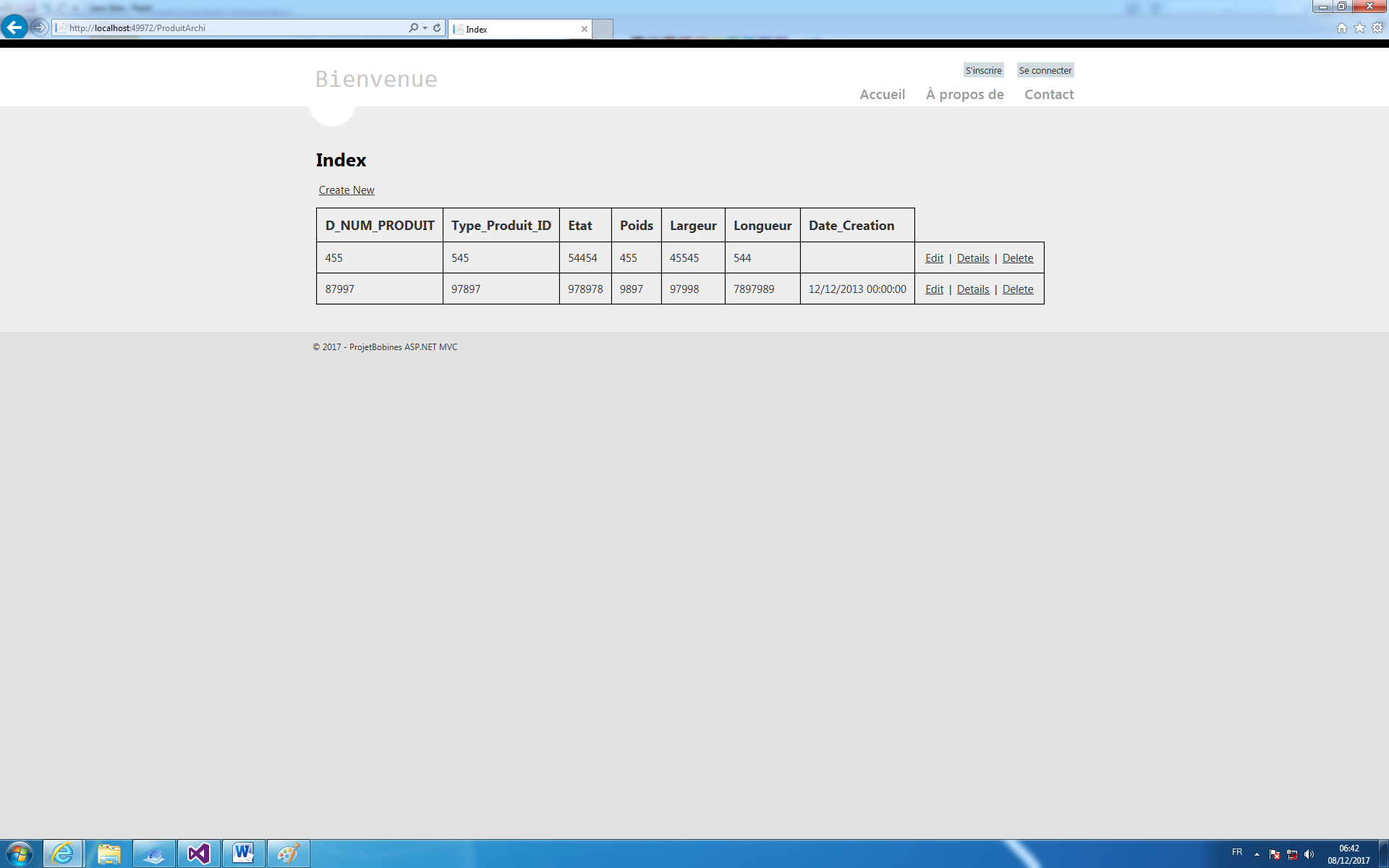
### La page « à propos de »

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



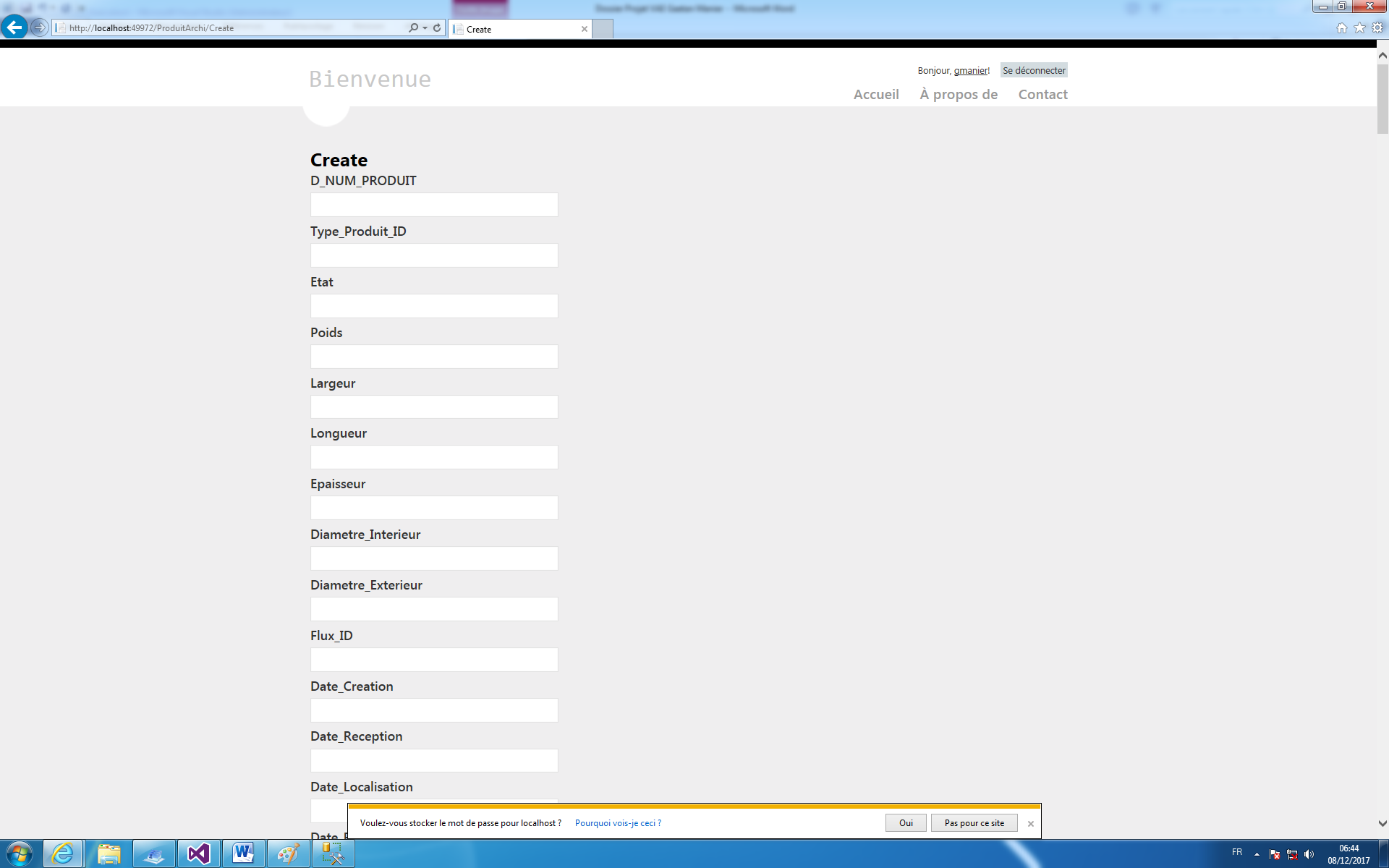
### La page « Produits Archi»

Cette page permet de gérer les données des produits en base de données.

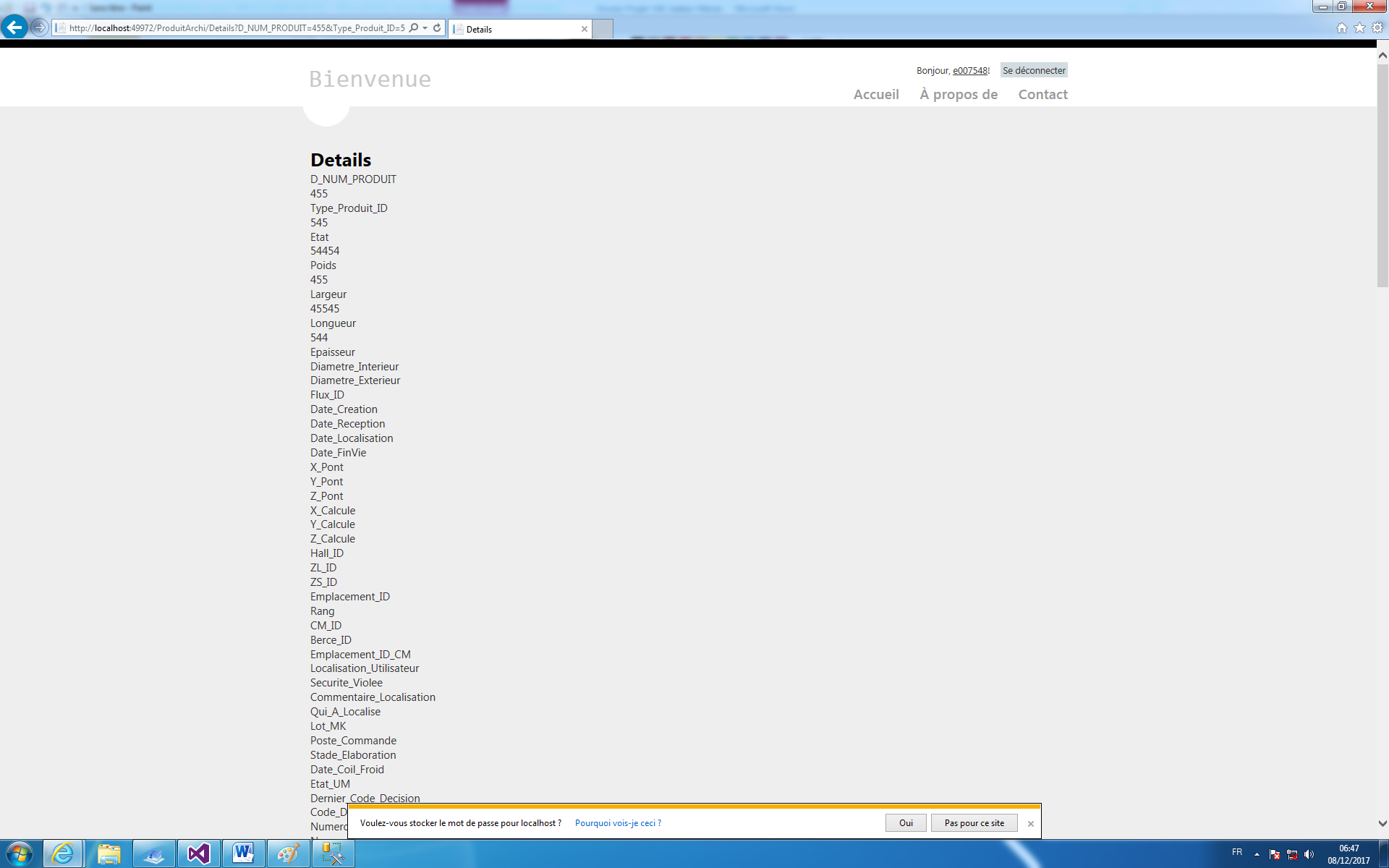


Elle se présente sous la forme d’une liste de produits présents dans la table « produits » sur laquelle l’utilisateur (suivant le rôle qui lui est attribué) peut :

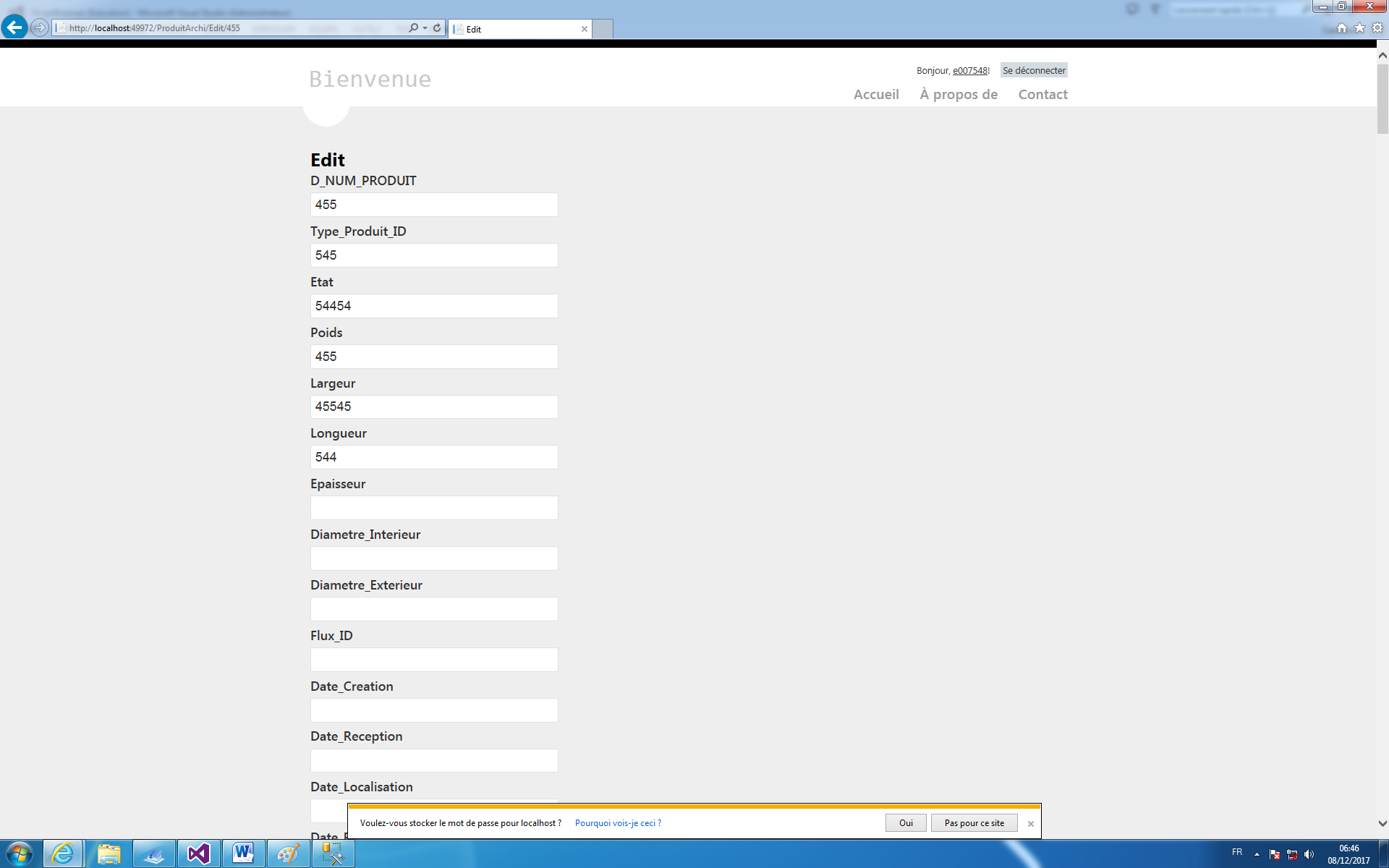
* Créer un nouveau produit



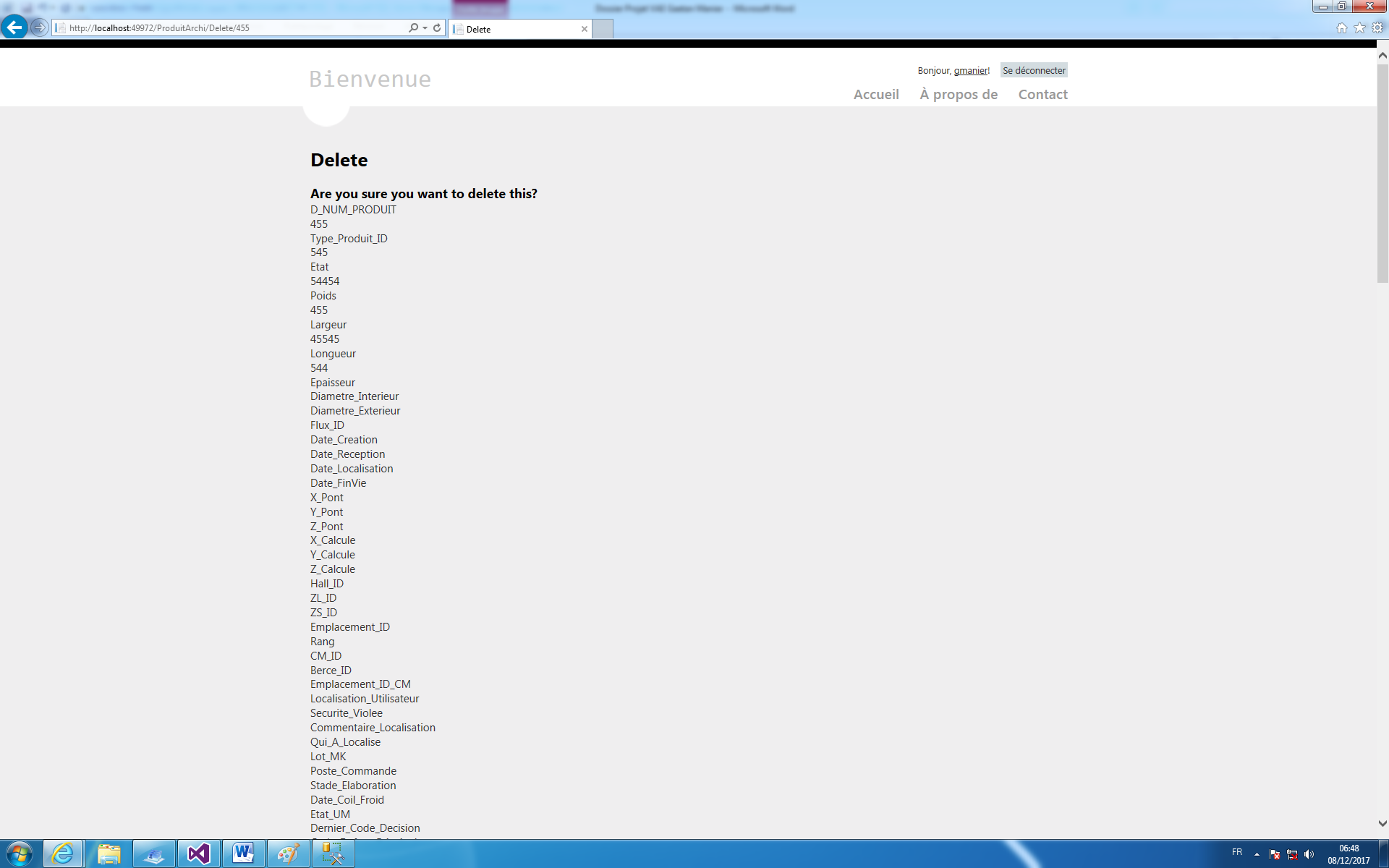
* Consulter le détail d’un produit



* Modifier un produit



* Supprimer un produit



Ces données produit servent pour alimenter le modèle en étoile du data warehouse.

### La page « Bilans bobine»

Cette page permet de consulter différents bilans bobines basé sur des rapports créés en automatiques et se basant sur les cubes du data warehouse.

### Pour aller plus loin

Il est bien sur possible d’aller plus loin dans notre 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 des fonctionnalités de consultation de données.
* de rajouter de nouveaux rapports consultables

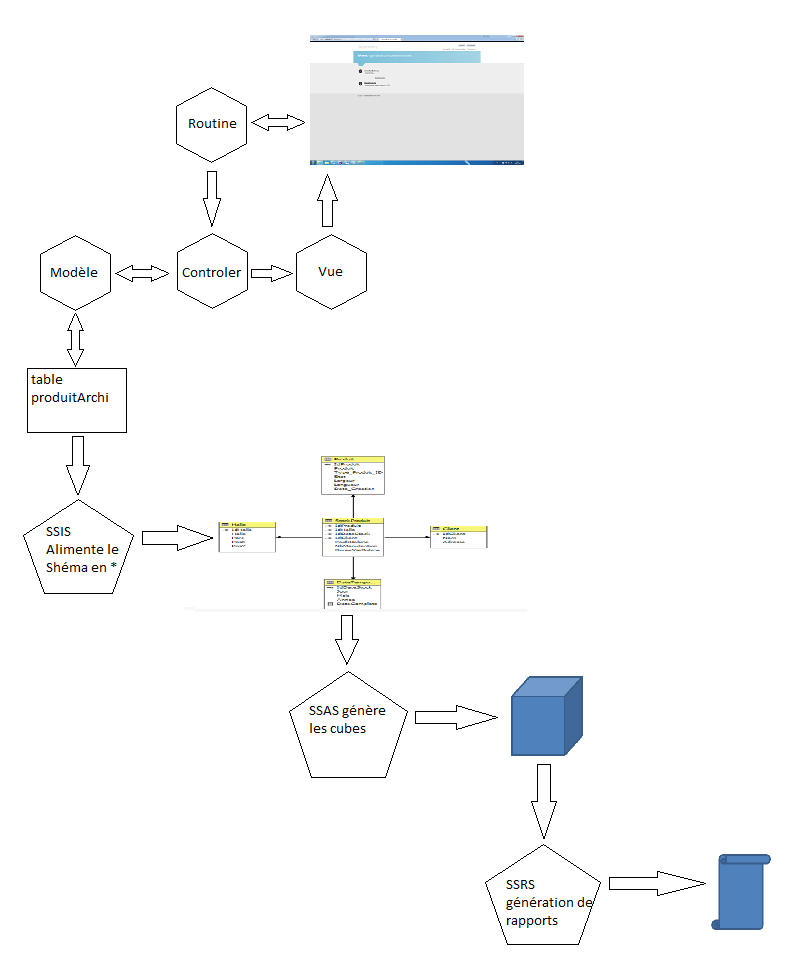
# Architecture du logiciel

## Introduction

L’application information de consultation de produits utilise les éléments suivants pour son architecture logicielle :

* Visual Studio 2013 (VS2013) sera la plateforme de développement aussi bien pour l’application serveur, que pour la base de données. Les développements informatiques seront en C# et ASP.Net. Les déploiements logiciels se feront directement à partir de VS2013.
* Le serveur web sera hébergé sur un PC bureautique dans notre cas.
* La base de données mise en place sera une SqlServeur Express 2012.

## Architecture



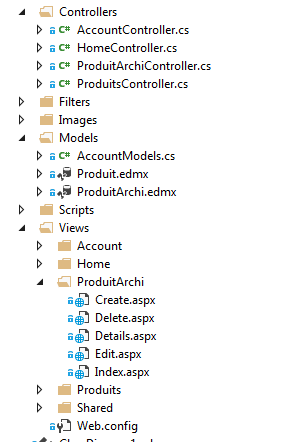
index.php?query=item/ranger-courses
point
d’entrée
Kernel::run()
$cont = new ItemController
 

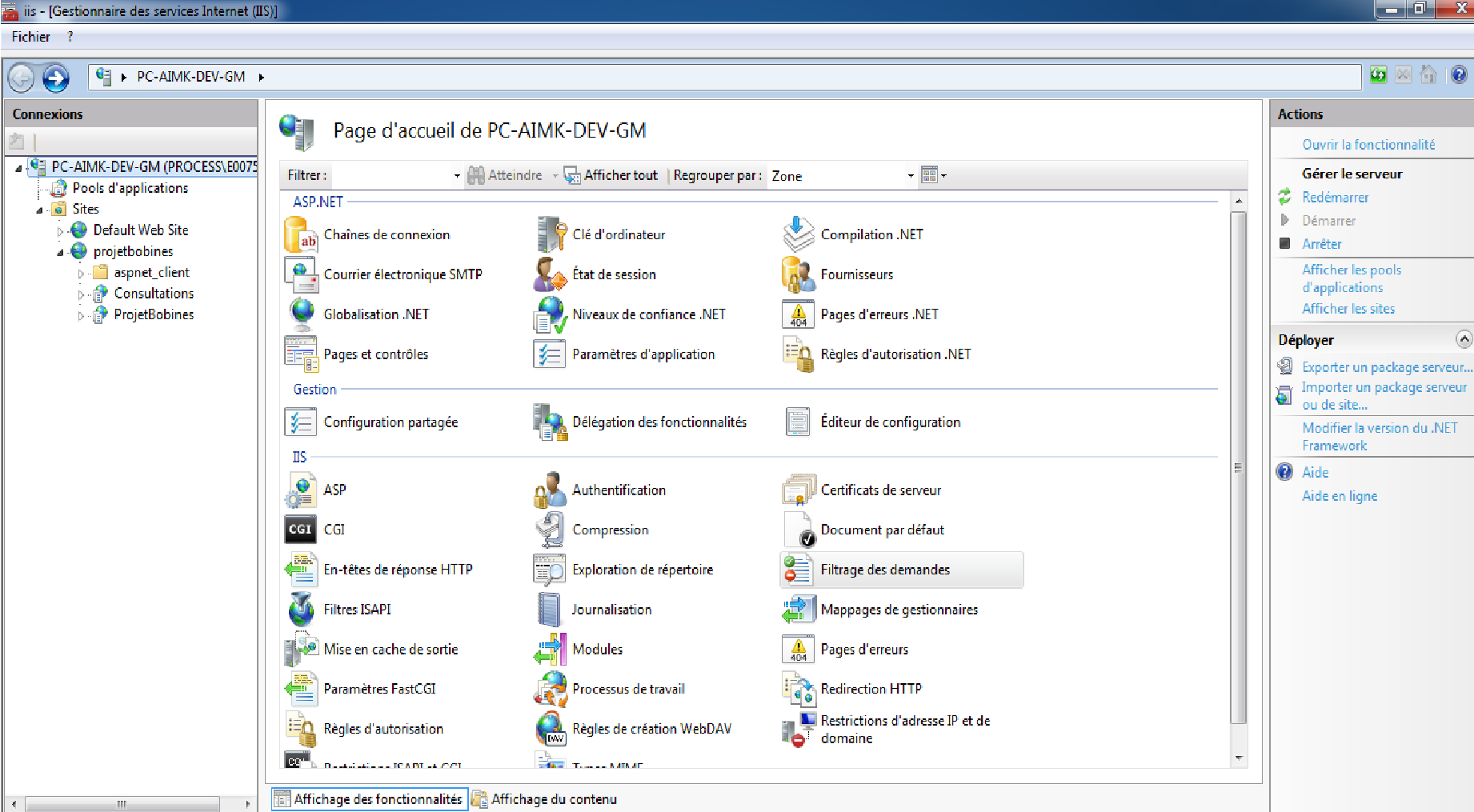
 ("ranger-cours...



## 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 sur le site de ArcelorMittal de Mardyck.

Pour 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 déploiment 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 deploiement (*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.

# 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.

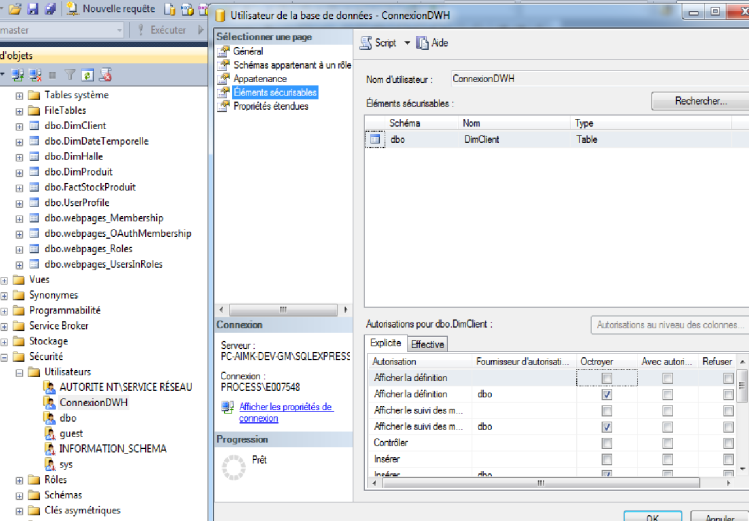
La configuration et l’utilisation de ces différents services seront vues plus en détail dans les chapitres suivants.

### 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.



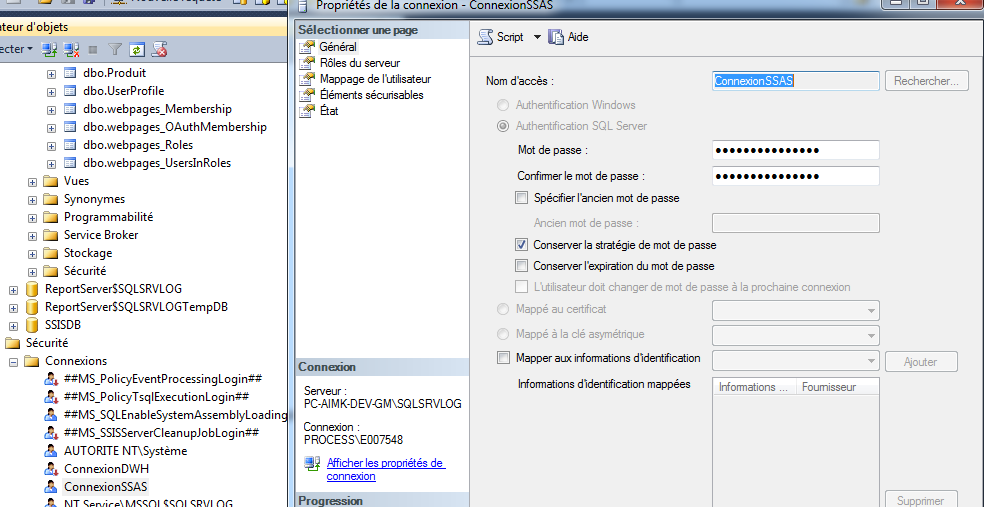
Pour cela il a été créé une connexion au niveau du serveur de base de données qui s’appelle « ConnexionDWH ».



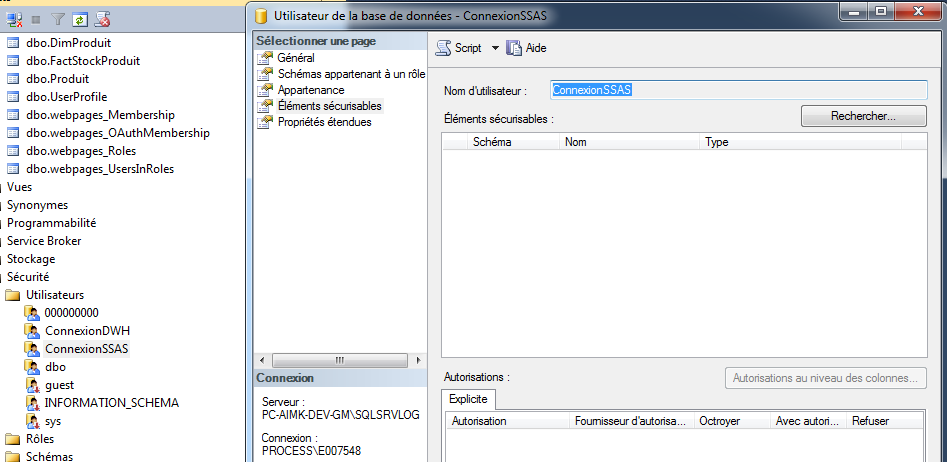
Ensuite un compte utilisateur sur la base de données « ConnexionDWH».

### 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.

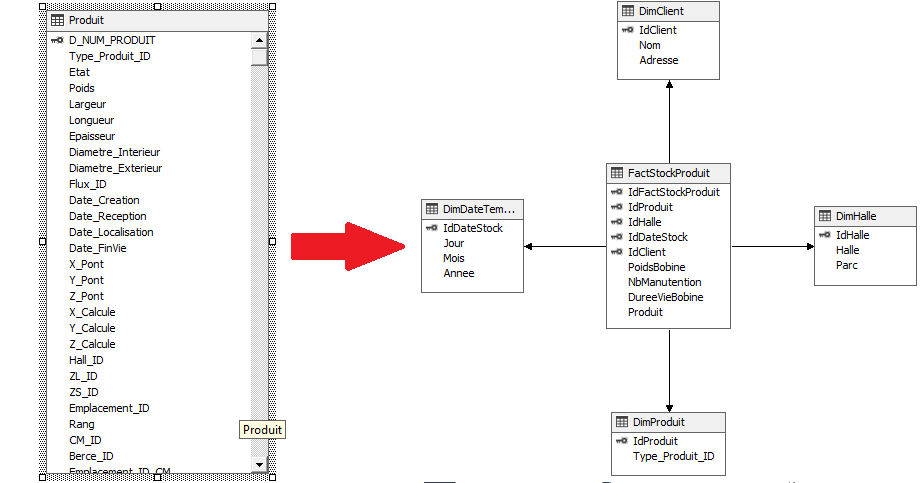
En temps normal j’utilise des bases de données de type SGDB (système de gestion de base de données relationnelles) dans nos applications car elles répondent aux besoins de stockage de données bidimensionnelles de production.

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

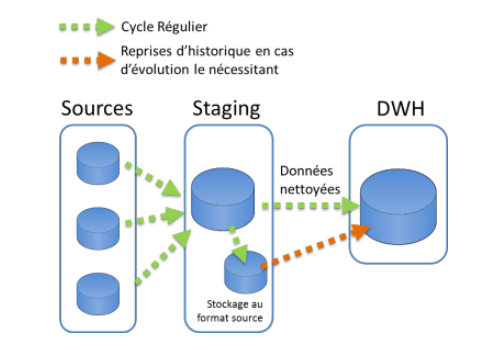
Pour pouvoir alimenter le data warehouse j’ai programmée un ETL (Extract-Transform-Load) en utilisation une solution « integration services » du module business intelligence de Visual Studio 2013.

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

### Configuration de l’ETL

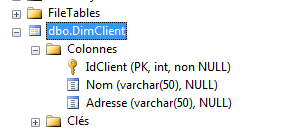
A partir des données de la table produit, l’ETL permet d’alimenter la table FactStockProduit (tables des faits du schéma en étoile) en effectuant différentes recherches dans les tables de dimension.

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 :

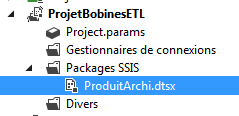
* Création de la table



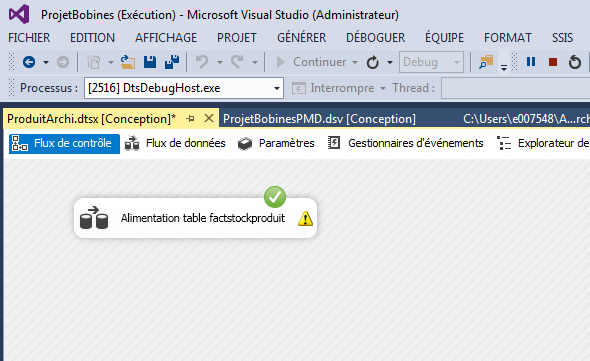
* Alimentation de la table à partir d’un fichier excel (suppression des blancs) qui se base sur le select distinct des champs de la table produits.

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

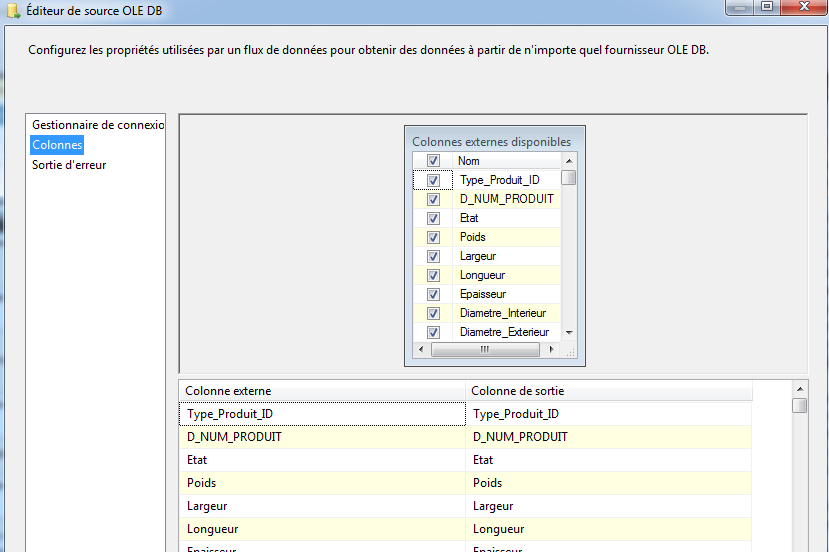
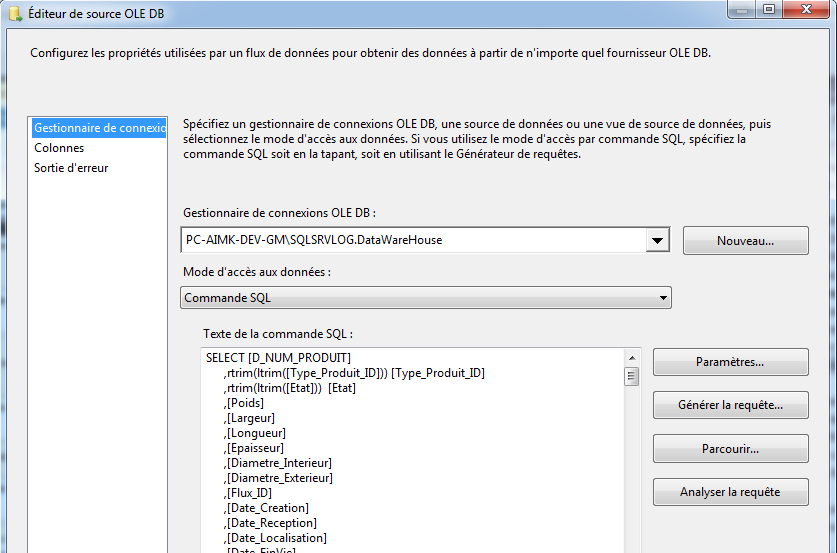
* Dans la solution il faut créer un package SSIS



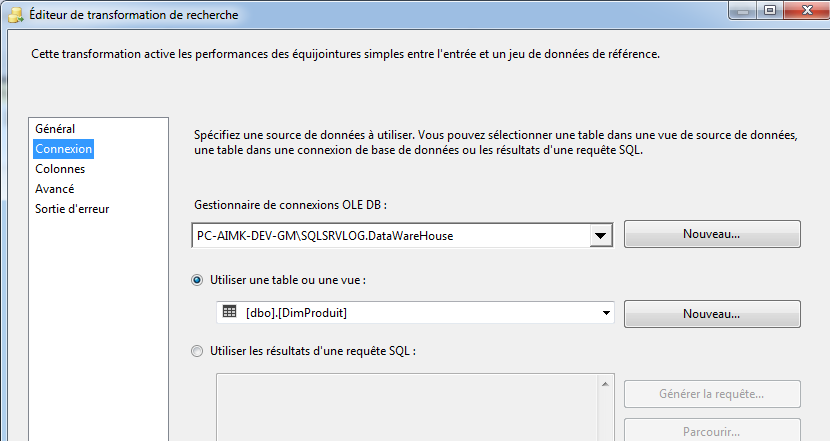
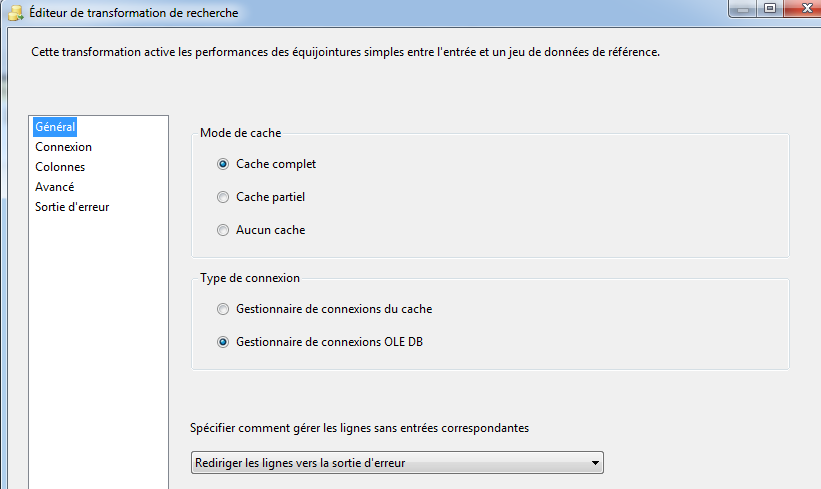
* On crée ensuite un flux de contrôle.

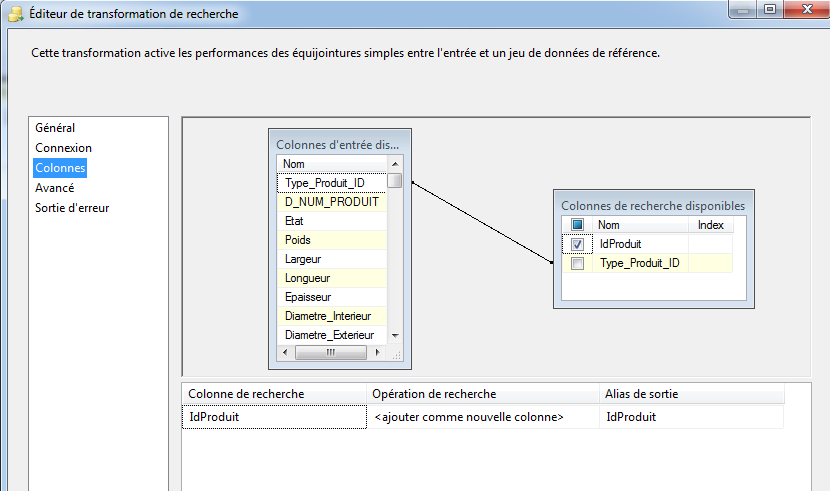


* Dans le flux de contrôle on double clic pour créer le flux de données.
* La source de données est configurée de telle façon que l’on récupère l’ensemble des données de la table produit. Dans la commande SQL les données de type chaînes de caractères utilisées pour effectuer des requêtes dans les tables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

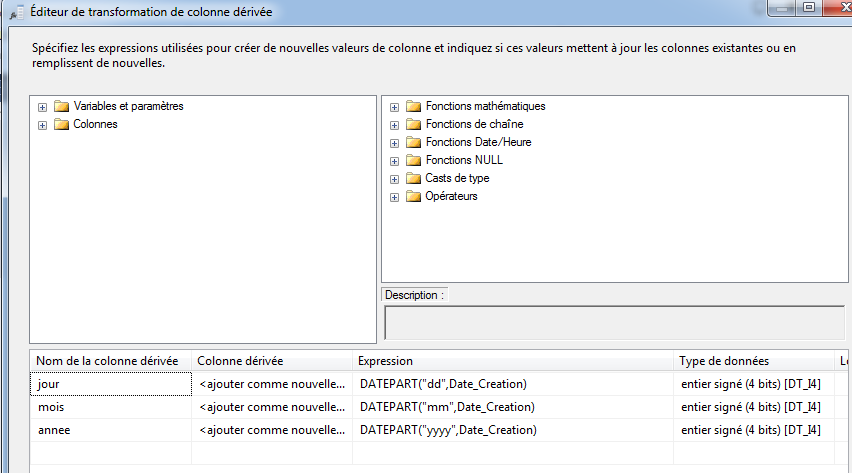


* En sortie de la source de données on vient rajouter un composant de recherche du produit qui va permettre de recherche l’ID de la table de dimension DimProduit en fonction du type de produit.

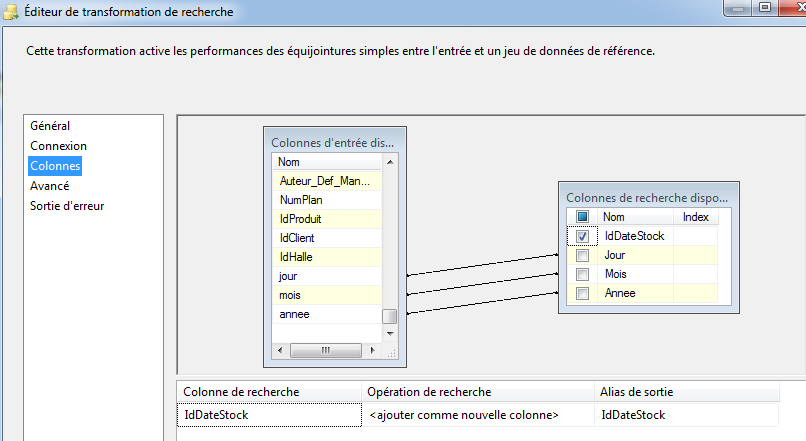




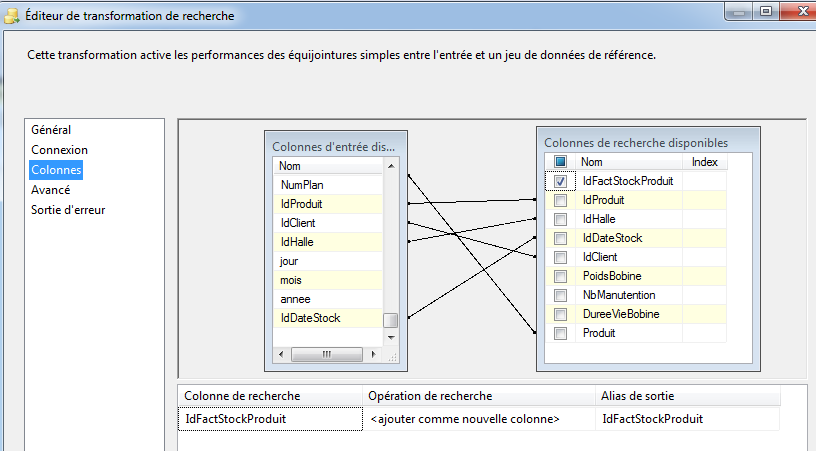
* Lorsque l’on relie la sortie d’un composant de recherche vers un autre, il faut spécifier si la recherche se fait avec ou sans correspondance. Dans notre cas ce fut par correpondance.
* On effectue la même chose pour retrouver les ID clients et Halles des tables DimClient et DimHall.
* Pour la recherche 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.



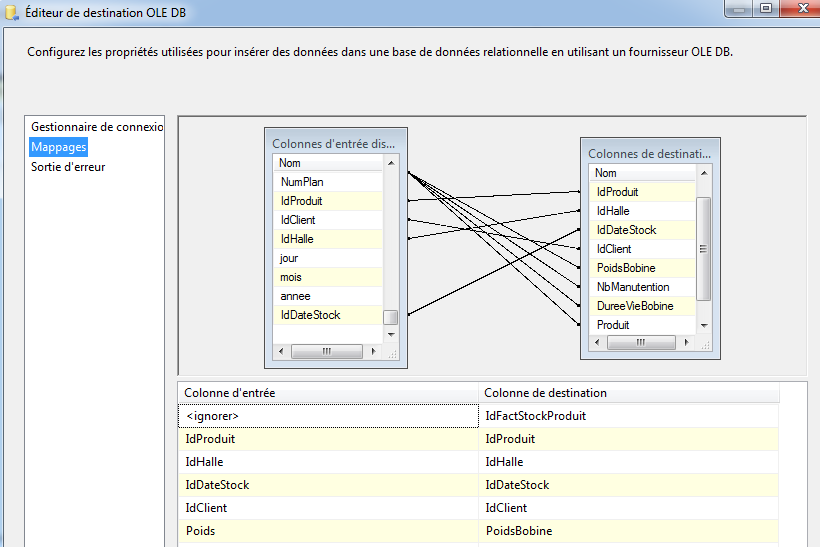
* Les nouvelles colonnes ainsi crées m’ont permis de les affecter au composant de recherche de la table de dimension DimDateTemporelle.



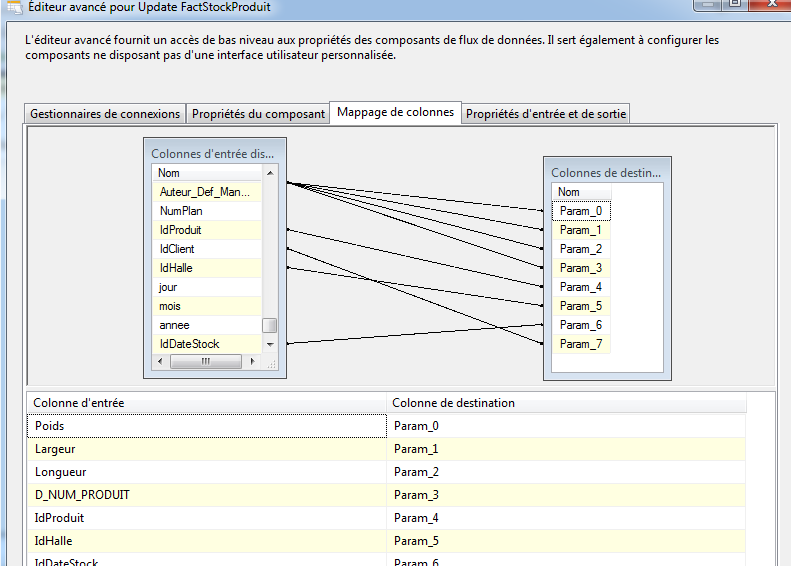
* Avec d’envoyer les données vers la table de destination, il fallait tester si les données étaient déjà présentes dans la table. Pour cela un composant de recherche permet de vérifier si les données pour un même enregistrement IDclient, IdHall, IDDateStock et IDProduit existent dans la table. Si la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



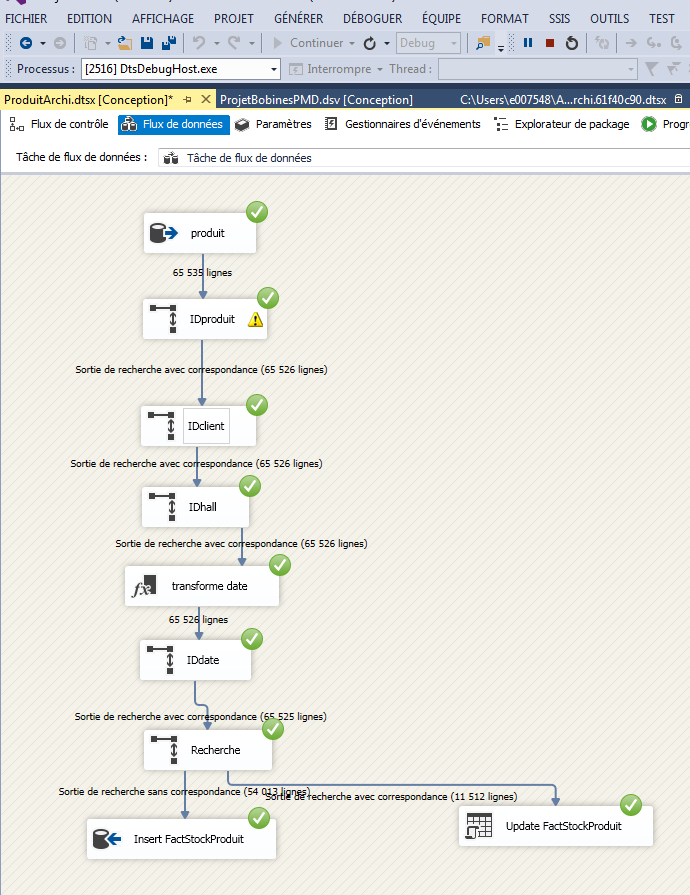
* La sortie sans correspondance du composant de recherche permet d’insérer un nouvel enregistrement dans la table FactStockProduit.



* 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.

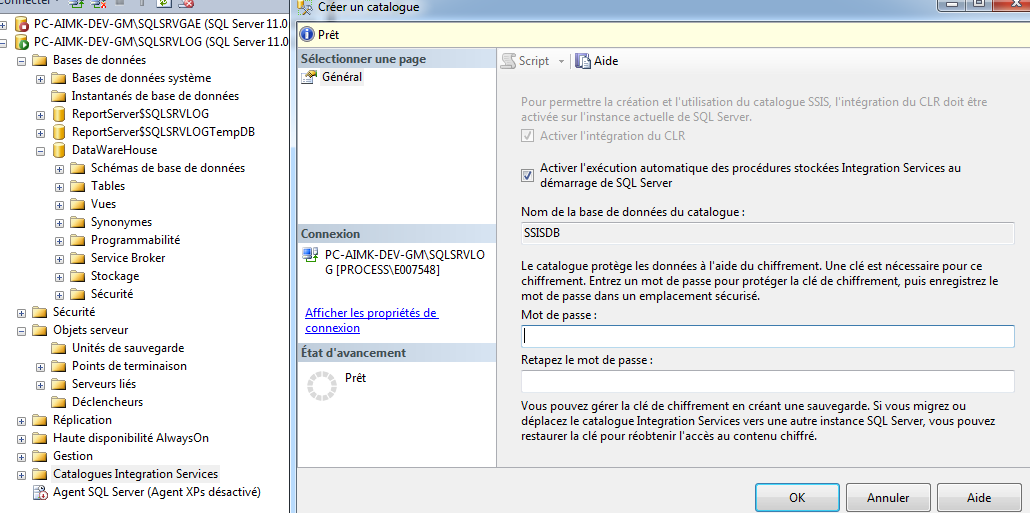
### Déploiement de l’ETL

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

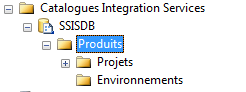
Pour cela un clic droit permet d’afficher la fenêtre d’invite et de remplir les différents éléments.



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

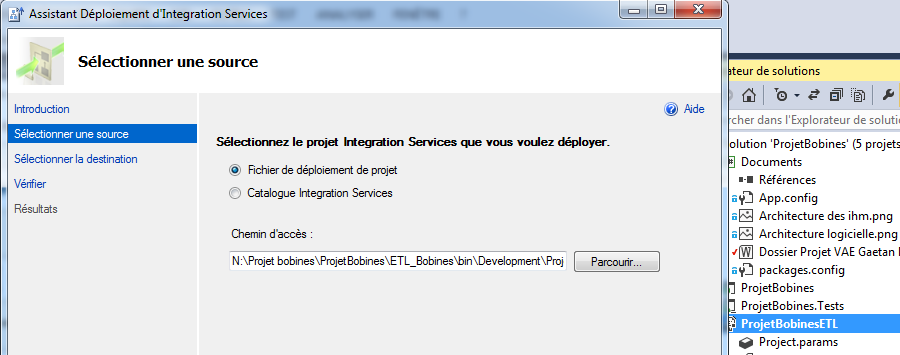
Le Common Language Runtime (CLR) est au cœur de Microsoft .NET Framework et fournit l'environnement d'exécution pour tout le code .NET Framework. Le code qui s'exécute dans le CLR est appelé code managé. Le CLR fournit divers services et fonctions requis pour l'exécution du programme, notamment la compilation 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.

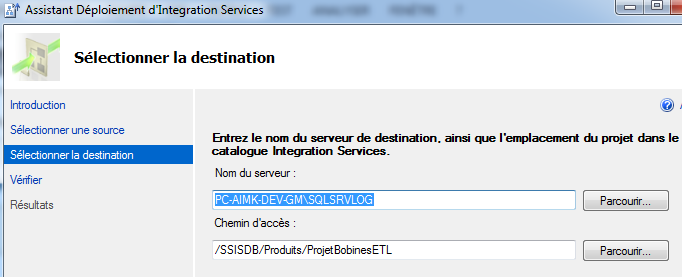


Sous Visual Studio, le clic droit sur le projet ProjetBobinesETL permet de sélectionner «déployer » et d’avoir l’assistant de déploiement d’Integration Services.

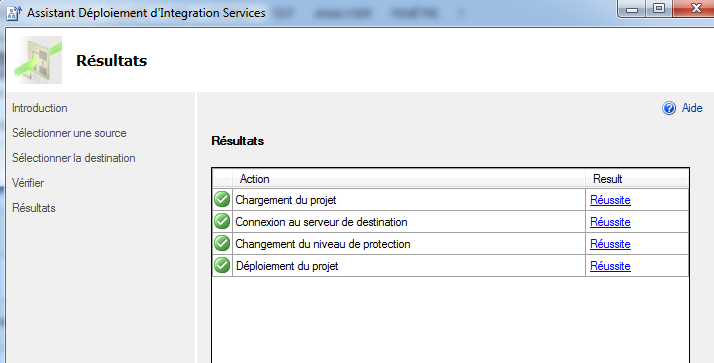
Il faut sélectionner le projet d’Integration Services à déployer. Dans mon cas il s’agit du choix par défaut correspondant au projet sélectionné.



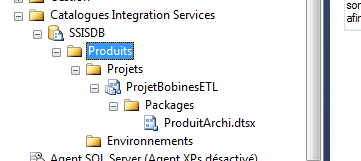
Il faut ensuite sélectionner la destination en sélectionnant le nom du serveur de base de données et le chemin d’accès. Dans notre cas on retrouve le projet précédemment créé dans Sql Server Management Studio.



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



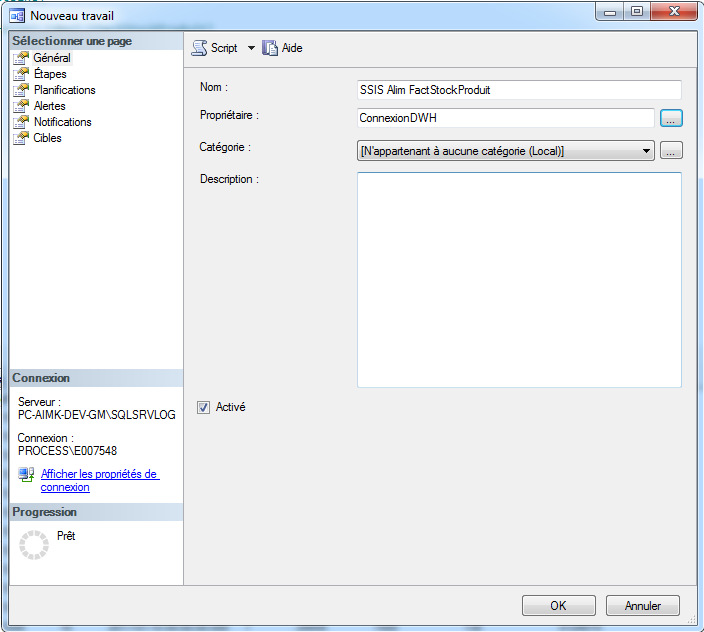
Dans Sql Server Management Studio on retrouve le projet déployé.



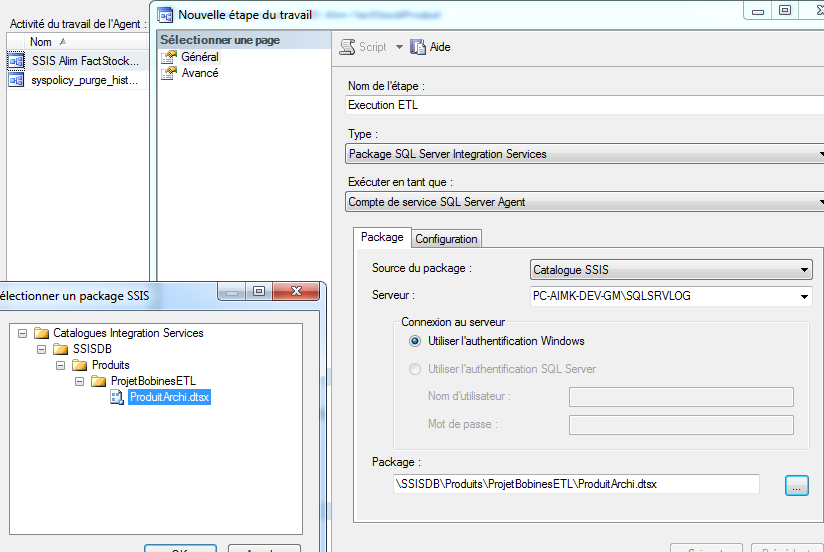
### Exécution de l’ETL

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.

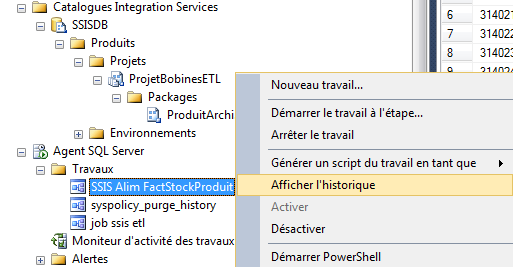
Pour cela il faut nommer le nouveau travail à faire exécuter et affecter un propriétaire.



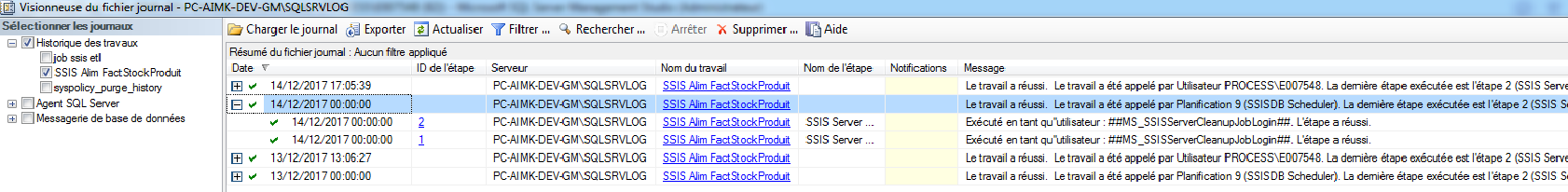
Puis il faut définir les étapes que l’on souhaite faire exécuter.



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.



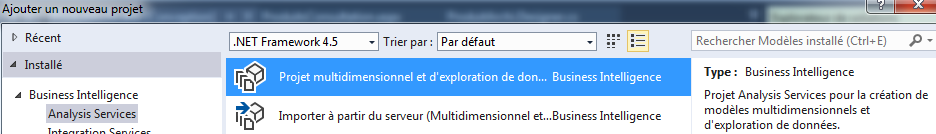
### Pour aller plus loin

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.

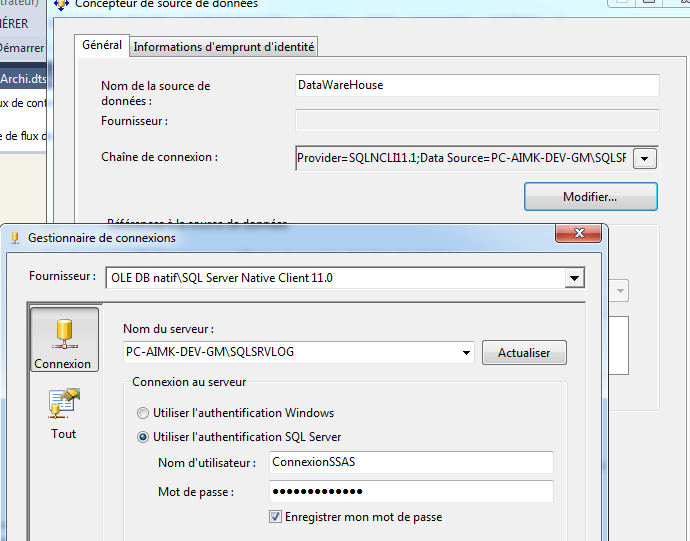
## Création et gestion du cube avec SSAS

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.



Créer le cube

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.

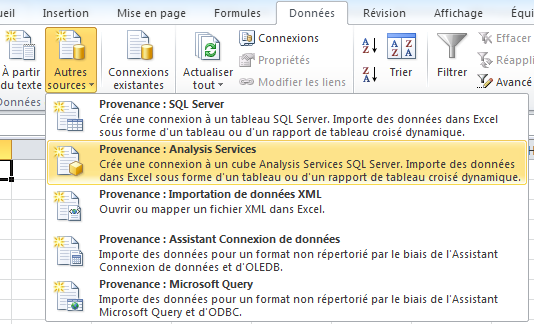


Le tester

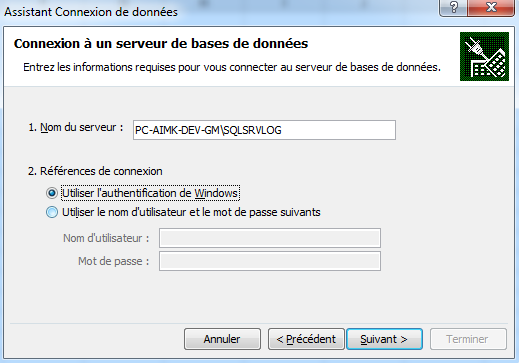
Le déployer

### Utilisation du cube avec Excel

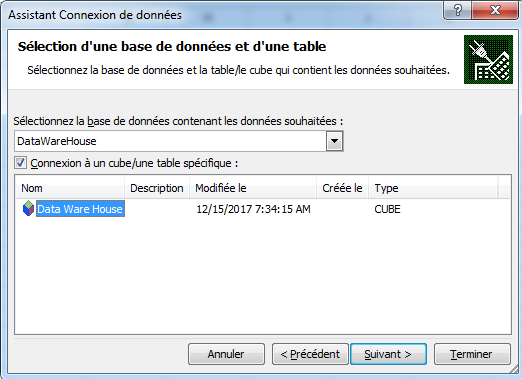
Il est possible d’utiliser directement le cube avec un fichier Excel afin de générer des rapports.



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

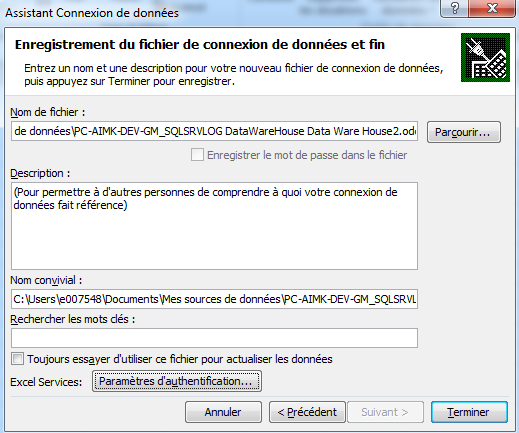


Déclarer la connexion au serveur de la base de données

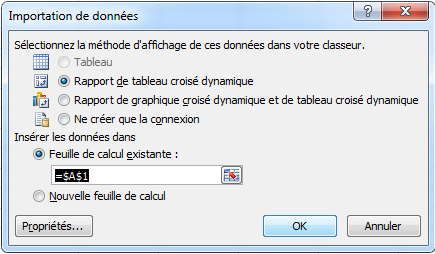


Enregistrer le fichier de connexion à la base de données. Dans notre cas j’ai pris une identification Windows mais il est possible créer une identification SSS (Secure Store Service) est un service partagé qui fournit le stockage et le mappage des informations d'identification telles que les noms de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

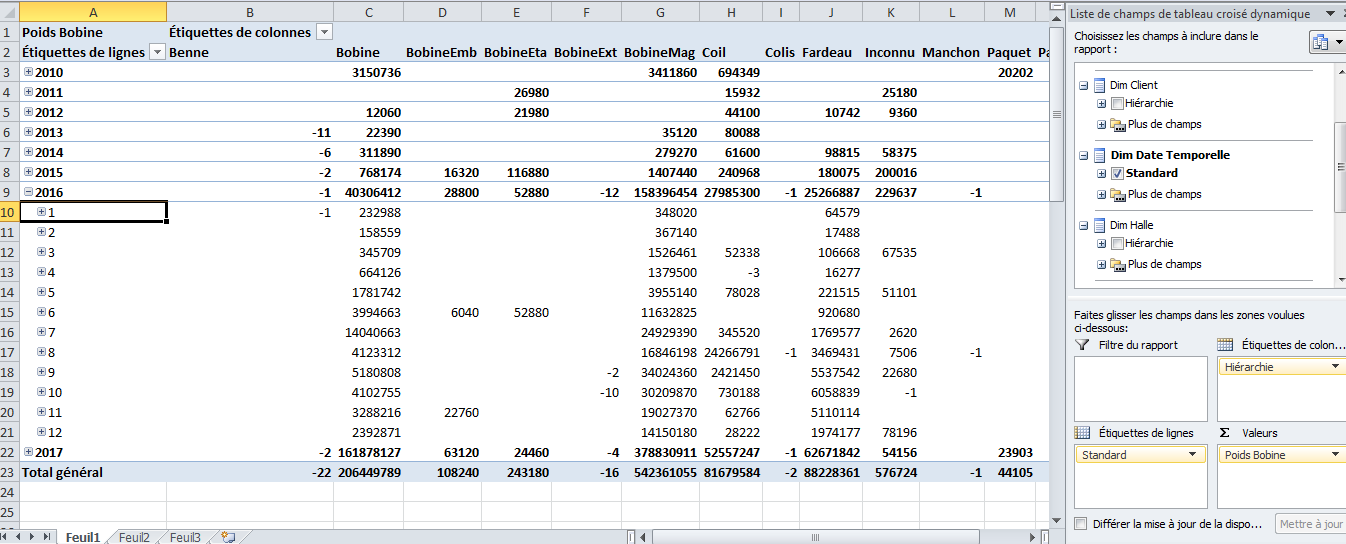
Puis sélectionner la base de données et le cube qui a été précédemment déployé.



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.

## Création des rapports avec SSRS

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.

## Alimentation du cube

# La sécurité

## Cryptage des comptes utilisateurs

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

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.

## 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 :

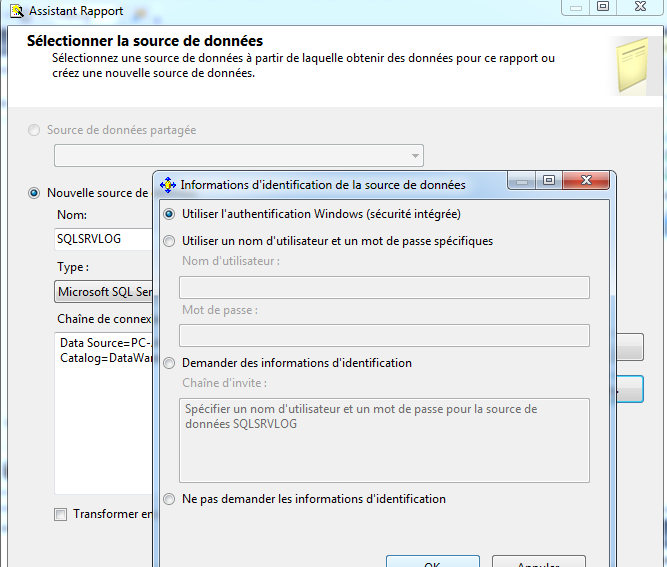
[Authorize(Roles="admin")]

public ActionResult Create() {

...

}

SSRS



# Qualité du code et gestion des versions

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

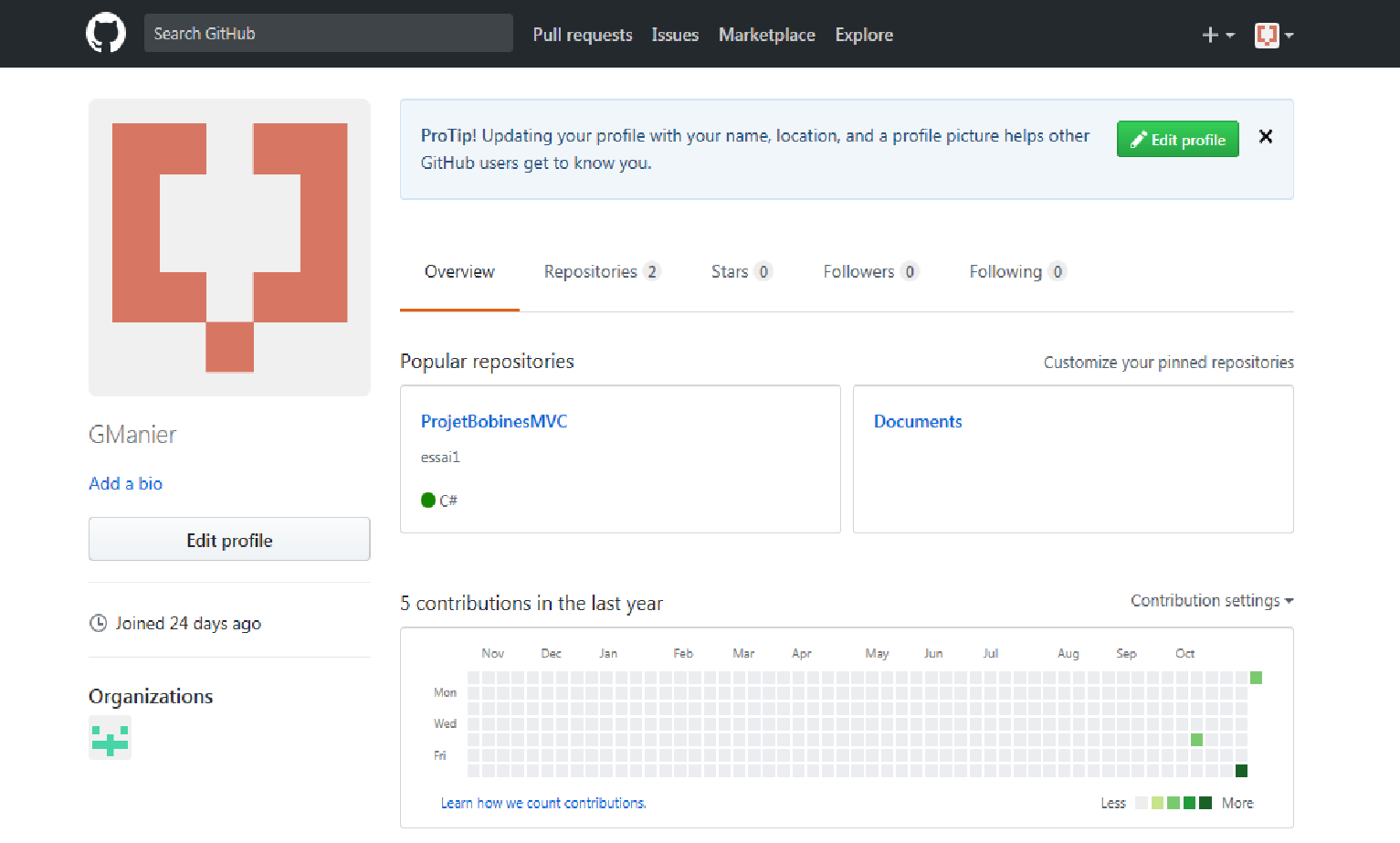
Sur le marché il existe plusieurs utilitaires de gestion de code sources et gestion des versions de fichiers. Pour n’en citer que quelques un, nous avons TFS (Team Foundation Server), Visual sourceSafe, Git, GitHub et BitBuchet.

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.

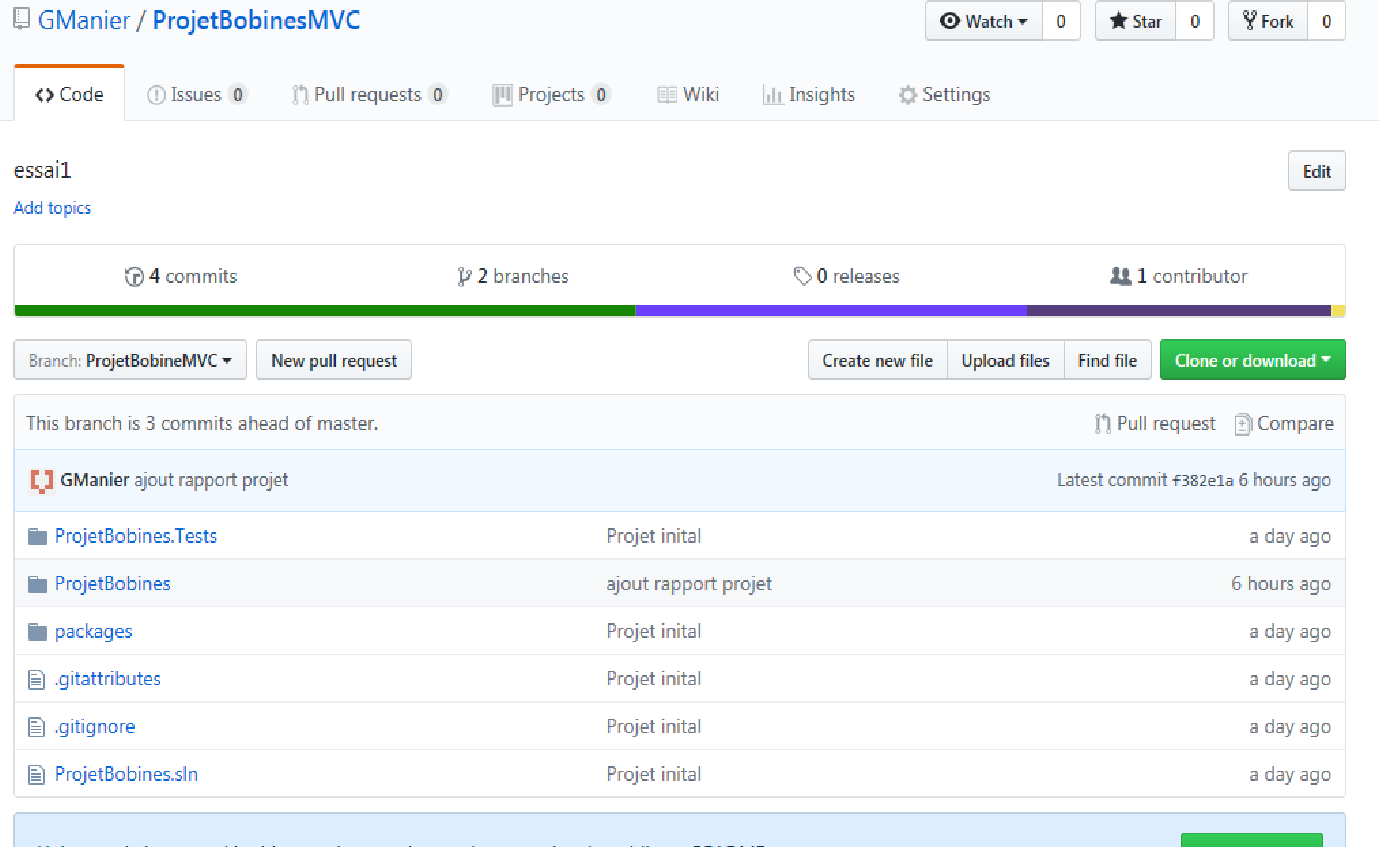
Comme Mon tuteur de projet devait également pouvoir au code source et aux documents que je créé, j’ai opté pour GitHub. L’avantage est qu’il s’interface directement avec Visual Studio 2013 (VS2013) et que l’extraction ou l’archivage des fichiers se fait naturellement avec la même souplesse que TFS.

## 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.

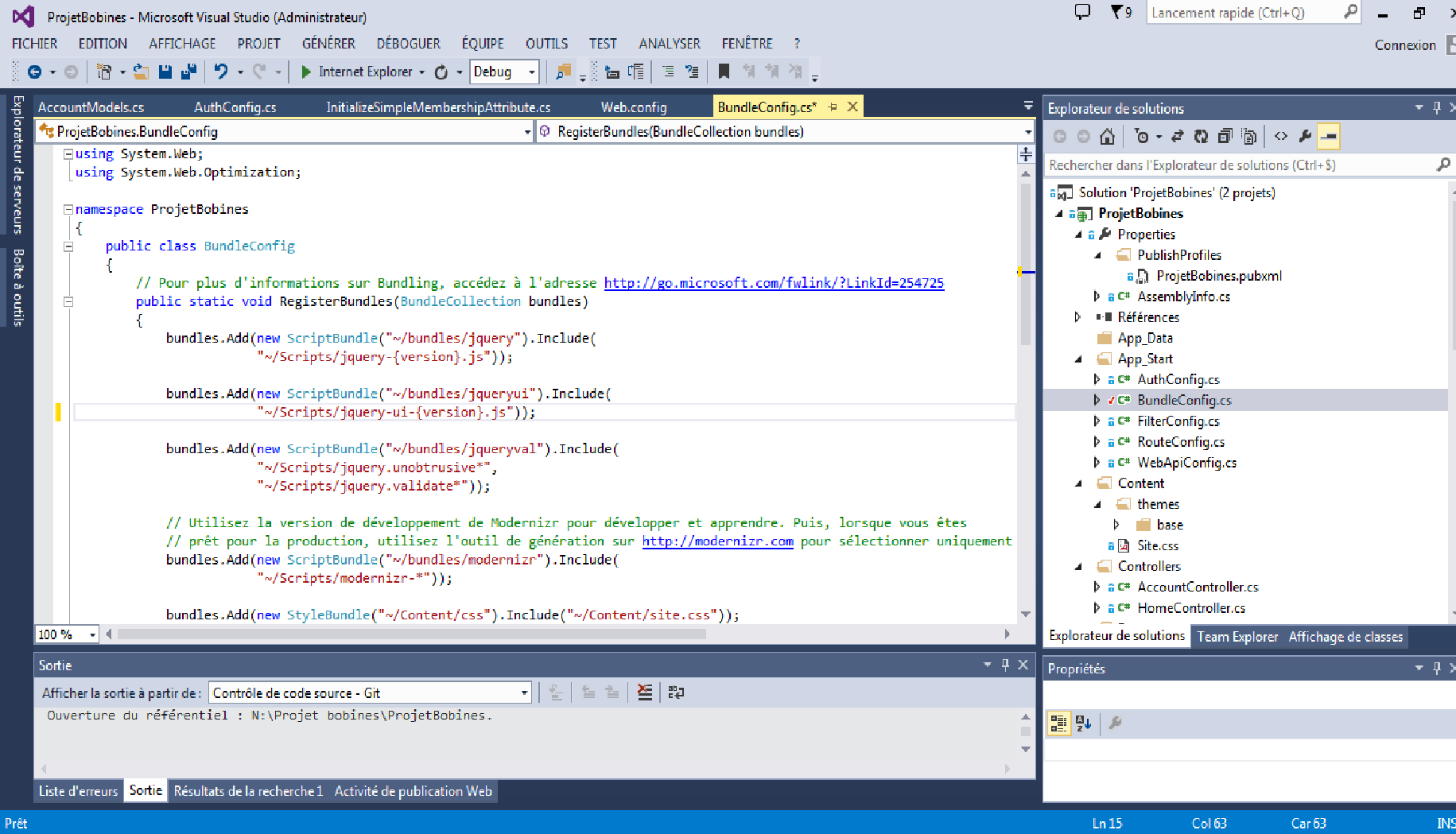


Une fois le compte créé, j’ai dû définir des « Repositories » afin de pouvoir y inclure les différentes solutions Visual Studio du projet. Il est possible ensuite de créer des branches de développements afin de gérer les versions en fonction soit des développements en cours, soit en recette, soit en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

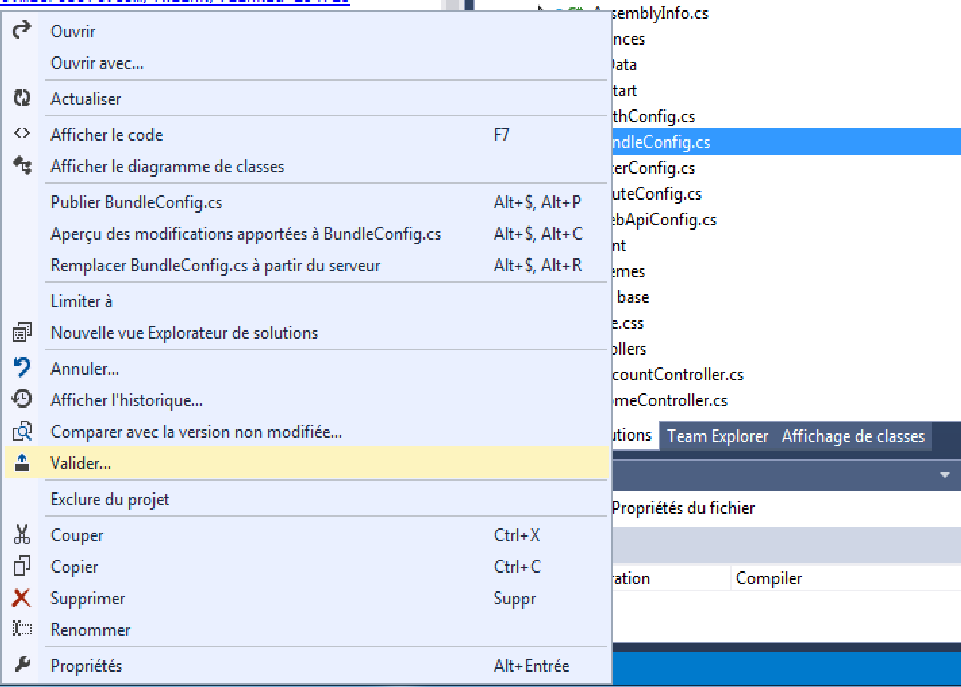


L’utilisation avec Visual Studio 2013 est assez simple car celui-ci propose naturellement 2 gestionnaires de code source, TFS ou Git.

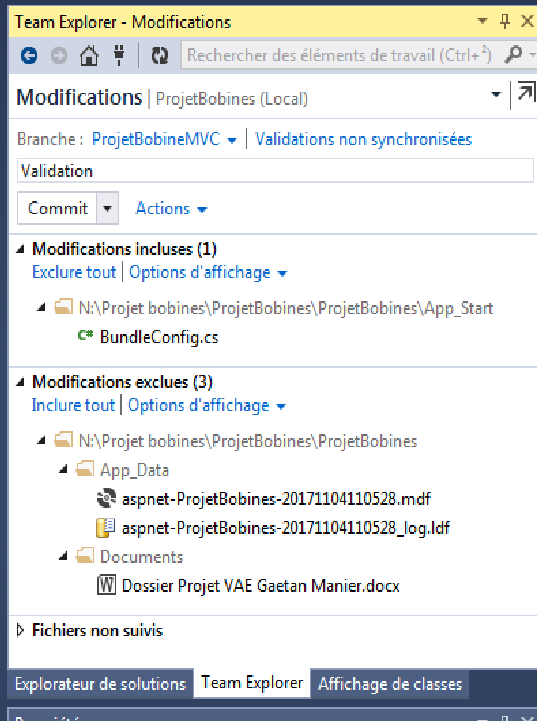
En sélectionnant Git, il faut déclarer son compte GitHub puis le « repositories » et branche sur laquelle on souhaite archiver ses projets.

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.



# 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.