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

Sommaire

[Le cahier des charges 4](#_Toc499457444)

[Fonctionnalités 5](#_Toc499457445)

[Introduction 5](#_Toc499457446)

[Les différents IHM 5](#_Toc499457447)

[La page d’accueil 5](#_Toc499457448)

[La gestion des accès utilisateur 6](#_Toc499457449)

[La page « Contact » 6](#_Toc499457450)

[La page « à propos de » 7](#_Toc499457451)

[La page « Produits» 7](#_Toc499457452)

[La page « Produits Archi» 8](#_Toc499457453)

[La page « Bilans bobine» 8](#_Toc499457454)

[Architecture du logiciel 8](#_Toc499457455)

[Introduction 8](#_Toc499457456)

[Architecture 8](#_Toc499457457)

[Le serveur Web 9](#_Toc499457458)

[Architecture de la base de données 11](#_Toc499457459)

[Introduction aux Data WareHouse 11](#_Toc499457460)

[Choix du schéma 11](#_Toc499457461)

[Configuration du serveur de base de données 12](#_Toc499457462)

[Compte de connexion 12](#_Toc499457463)

[Alimentation du Dataware house 12](#_Toc499457464)

[Configuration de l’ETL 12](#_Toc499457465)

[Flux multiples 13](#_Toc499457466)

[Déploiement 14](#_Toc499457467)

[La sécurité 16](#_Toc499457468)

[Cryptage des comptes utilisateurs 16](#_Toc499457469)

[Gestion des droits 16](#_Toc499457470)

[DpapiProtectedConfigurationProvider classe 16](#_Toc499457471)

[Vue d'ensemble de la configuration protégée 16](#_Toc499457472)

[Utilisation de la configuration protégée 18](#_Toc499457473)

[La méthode du formulaire 20](#_Toc499457474)

[ValidateAntiForgery 21](#_Toc499457475)

[Clics multiples 21](#_Toc499457476)

[Le contrôle des données 22](#_Toc499457477)

[1) Contrôle des champs visibles 22](#_Toc499457478)

[2) Contrôle d'un champ caché 23](#_Toc499457479)

[Le captcha 25](#_Toc499457480)

[Qualité du code et gestion des versions 25](#_Toc499457481)

[Choix de la gestion des versions 26](#_Toc499457482)

[Copie et renommage de répertoires 26](#_Toc499457483)

[Utilitaire de gestion de code sources 26](#_Toc499457484)

[GitHub 26](#_Toc499457485)

[Conclusions 30](#_Toc499457486)

# 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 va permettre de consulter des fiches de bobines à partir d’une interface et de rapports BI.

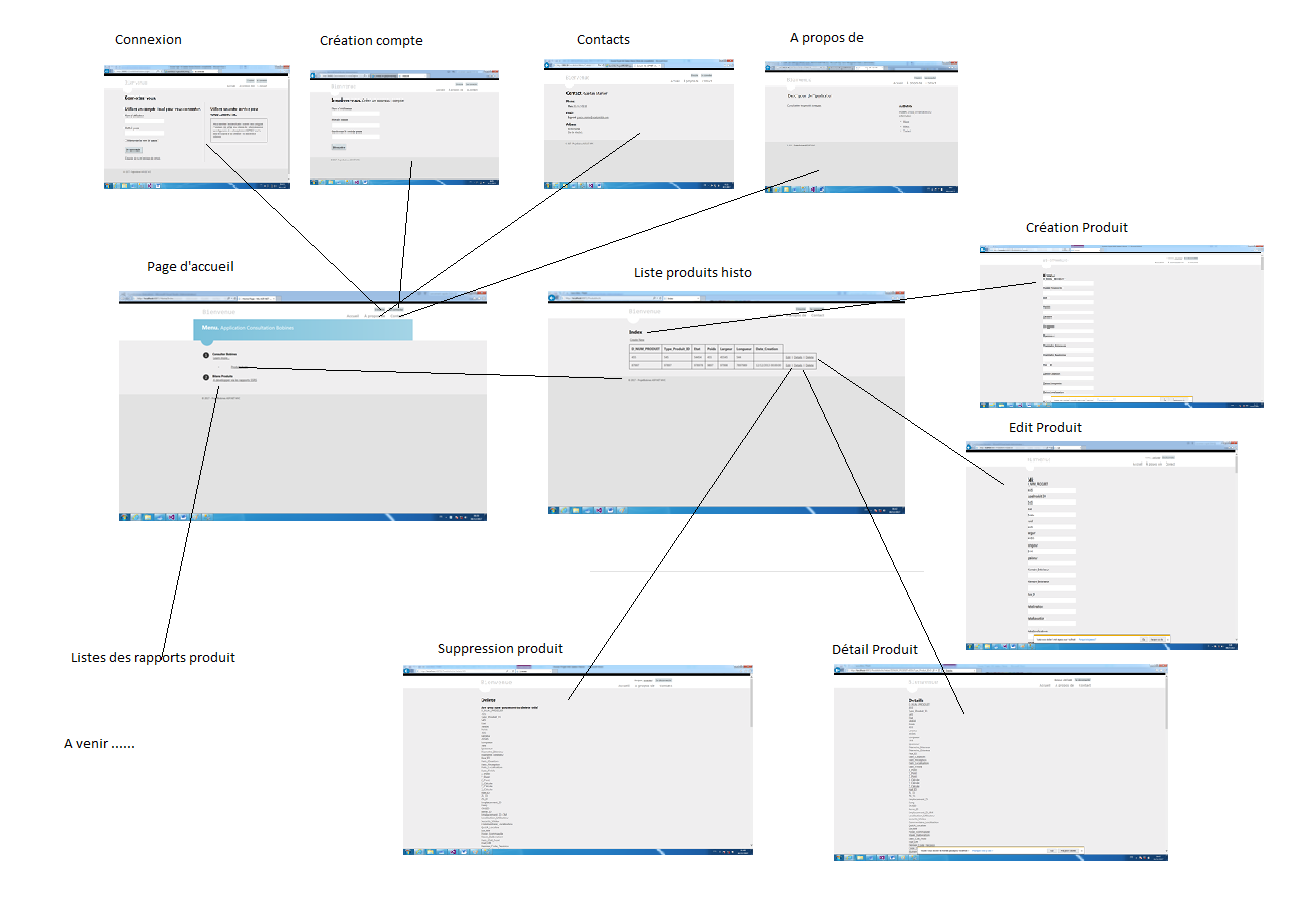
L’application serveur permettra de consulter ces informations à partir de pages web. Il sera possible de modifier ces fiches produit grâce à divers actions : création, modification et suppressions.

Ces fiches contiennent les informations relatives à leur historique et leurs localisations dans les halles. L’utilisateur aura la possibilité de faire des recherches des bobines soit par numéro, halle …

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

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

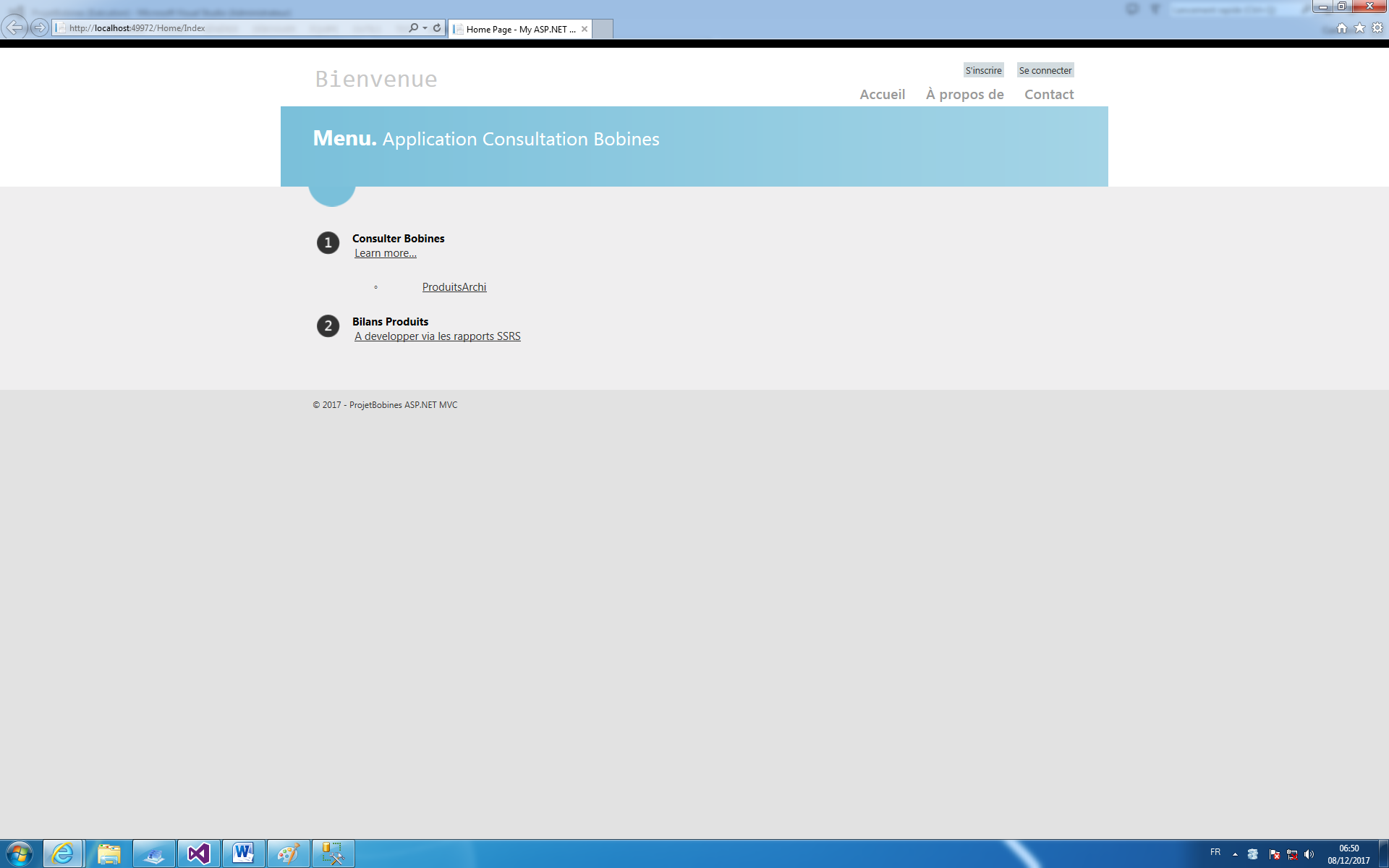
Les utilisateurs auront des droits afin de pouvoir réaliser des actions sur la base produits.



## Les différents IHM

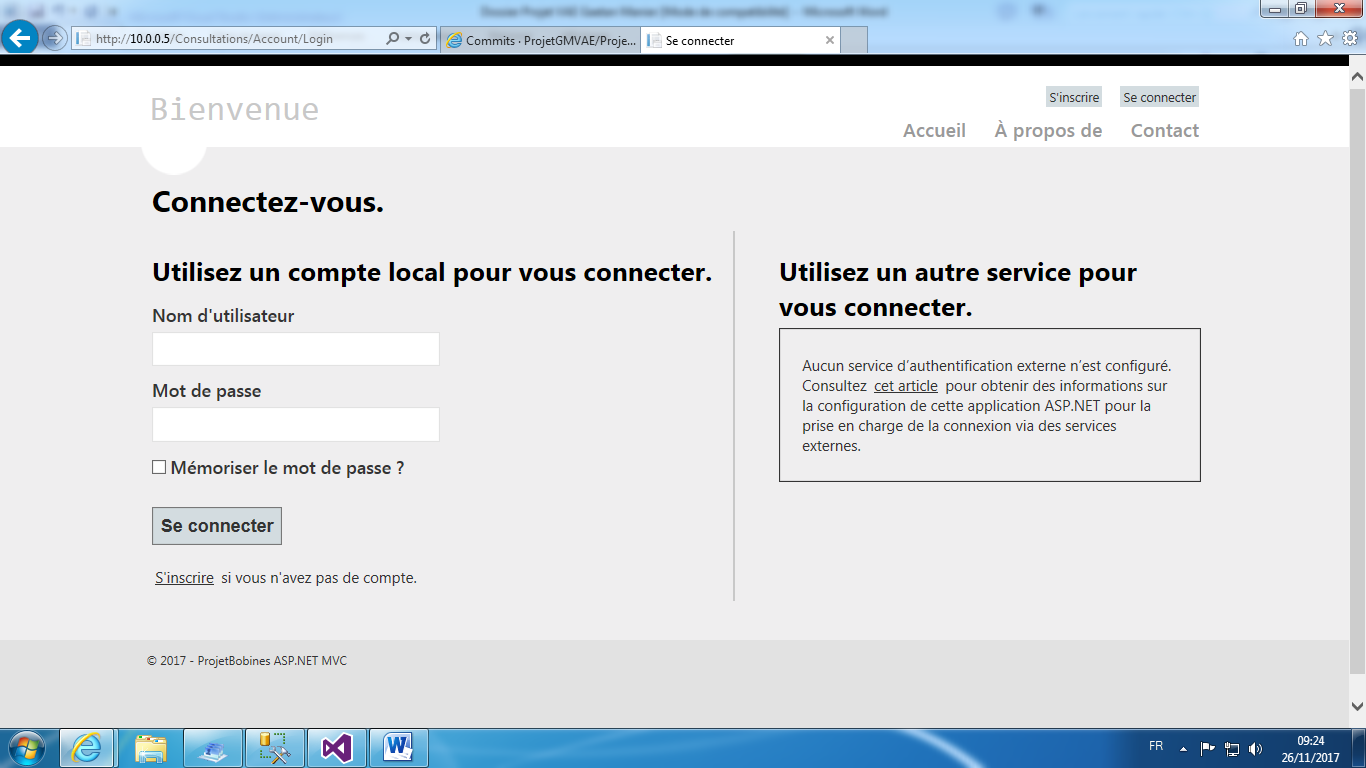
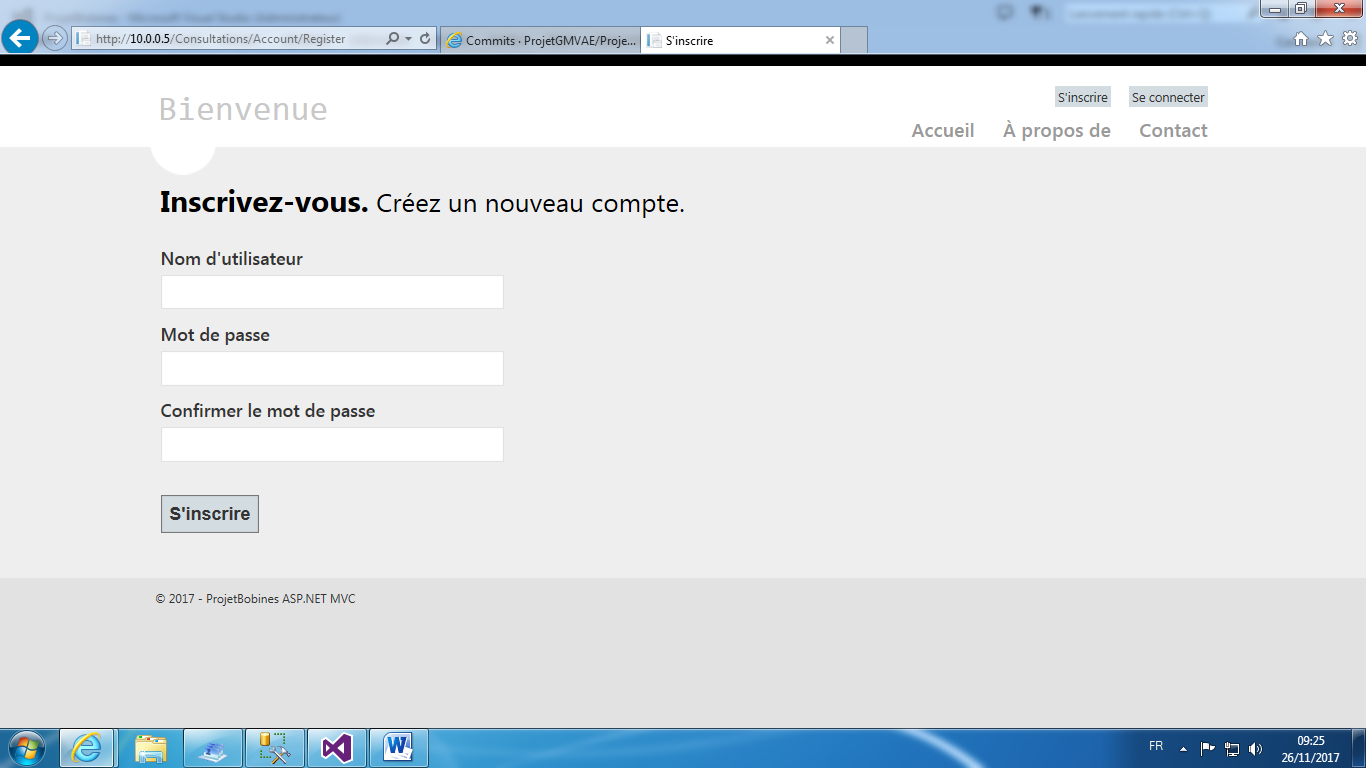
### La page d’accueil

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.

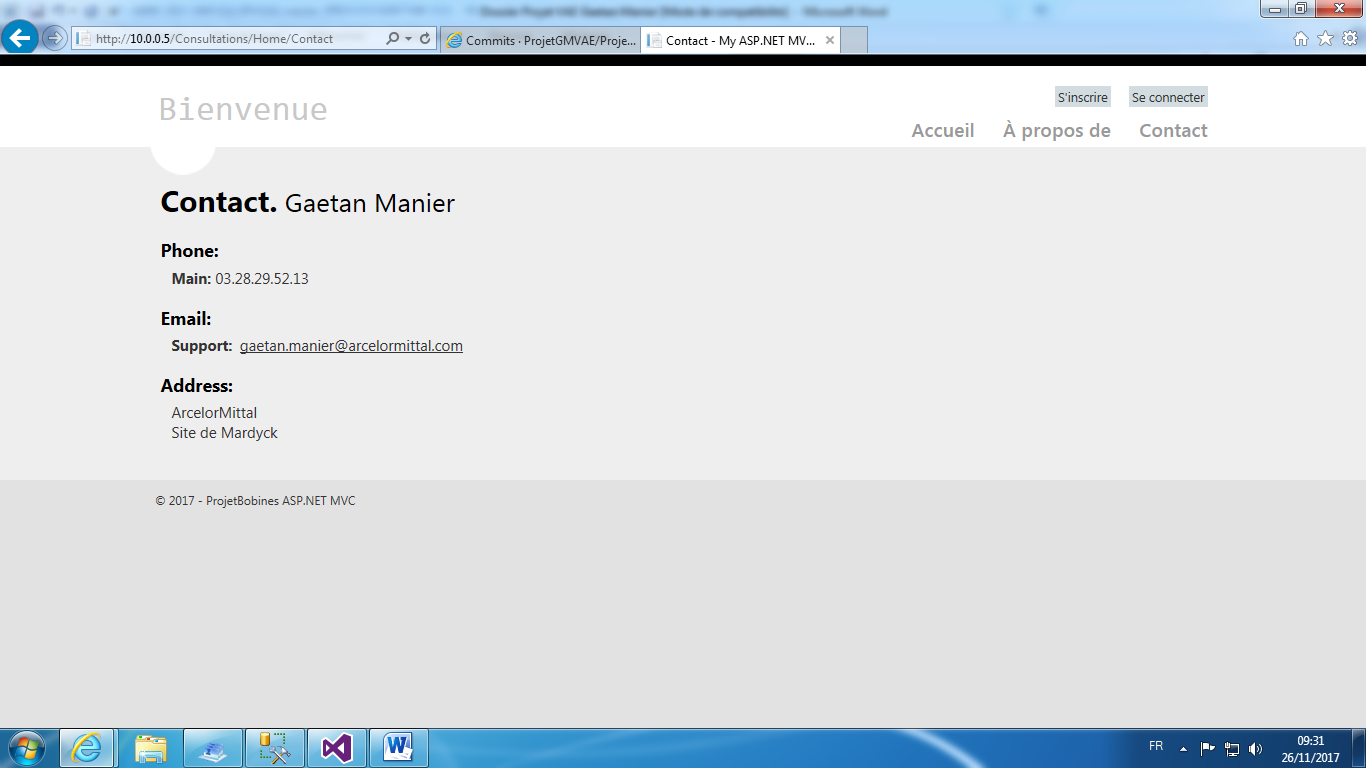
Lors de la création d’un nouveau compte, l’utilisateur doit envoyer un mail à l’administrateur du site afin de définir le rôle de son compte. L’accès aux différentes vues et aux fonctionnalités (création, suppression, mise à jour) se fait grâce à l’attribution de rôles sur les comptes.

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

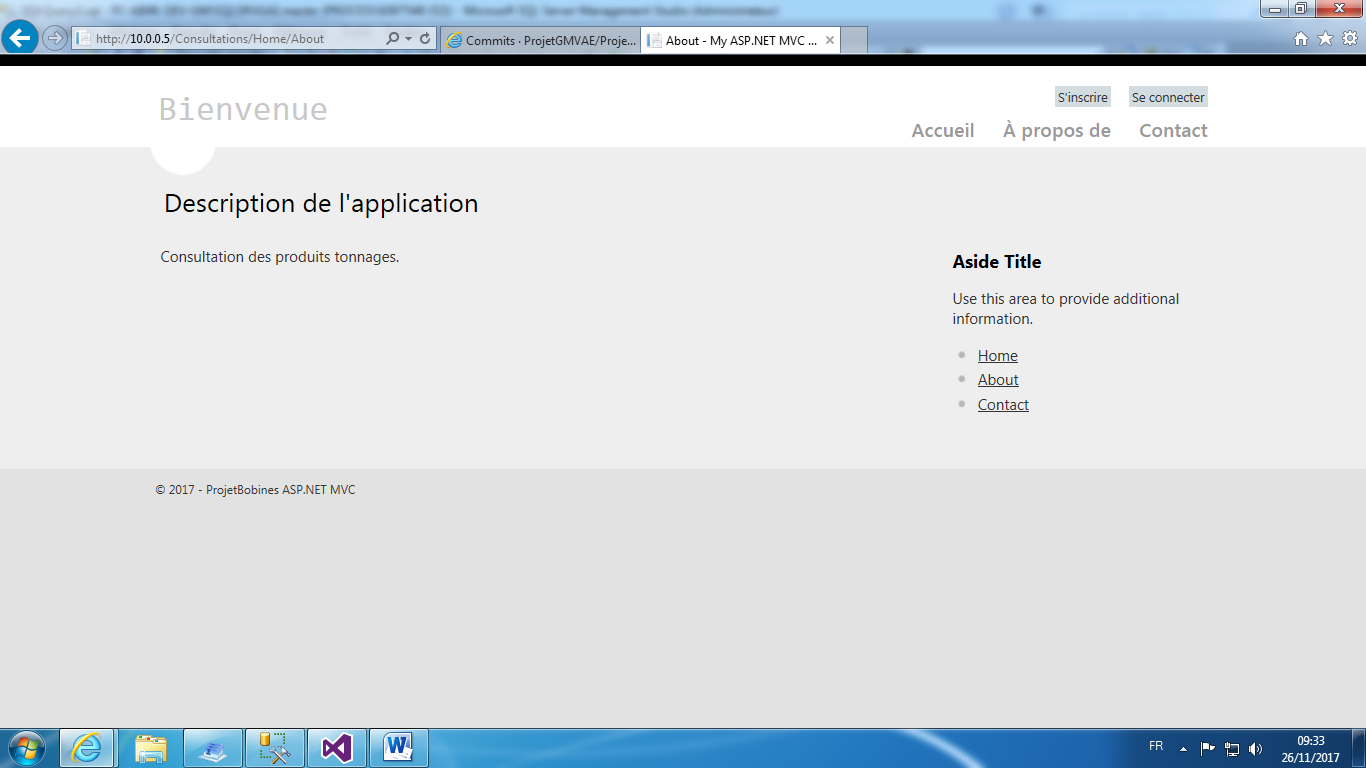
### La page « Contact »

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.

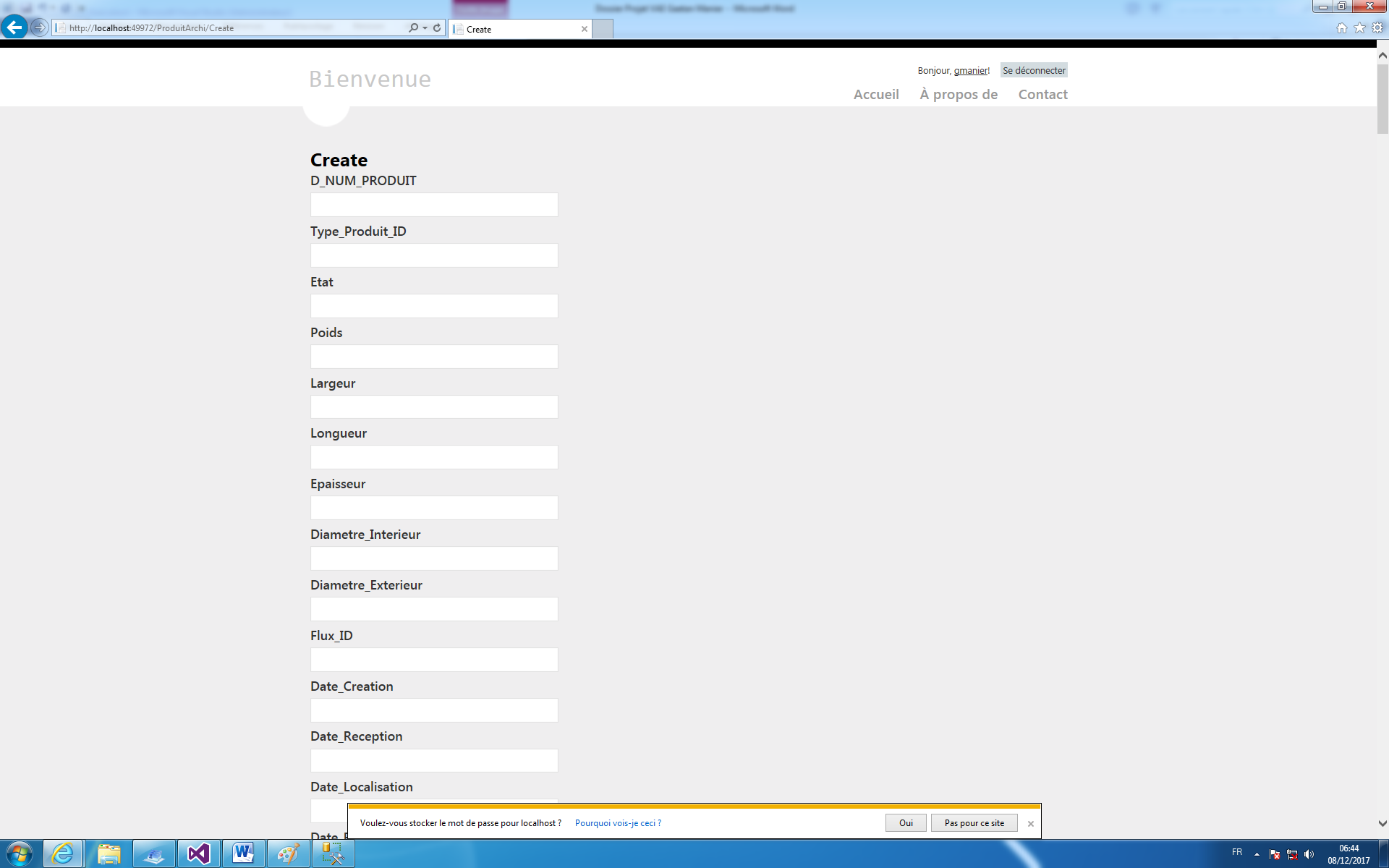


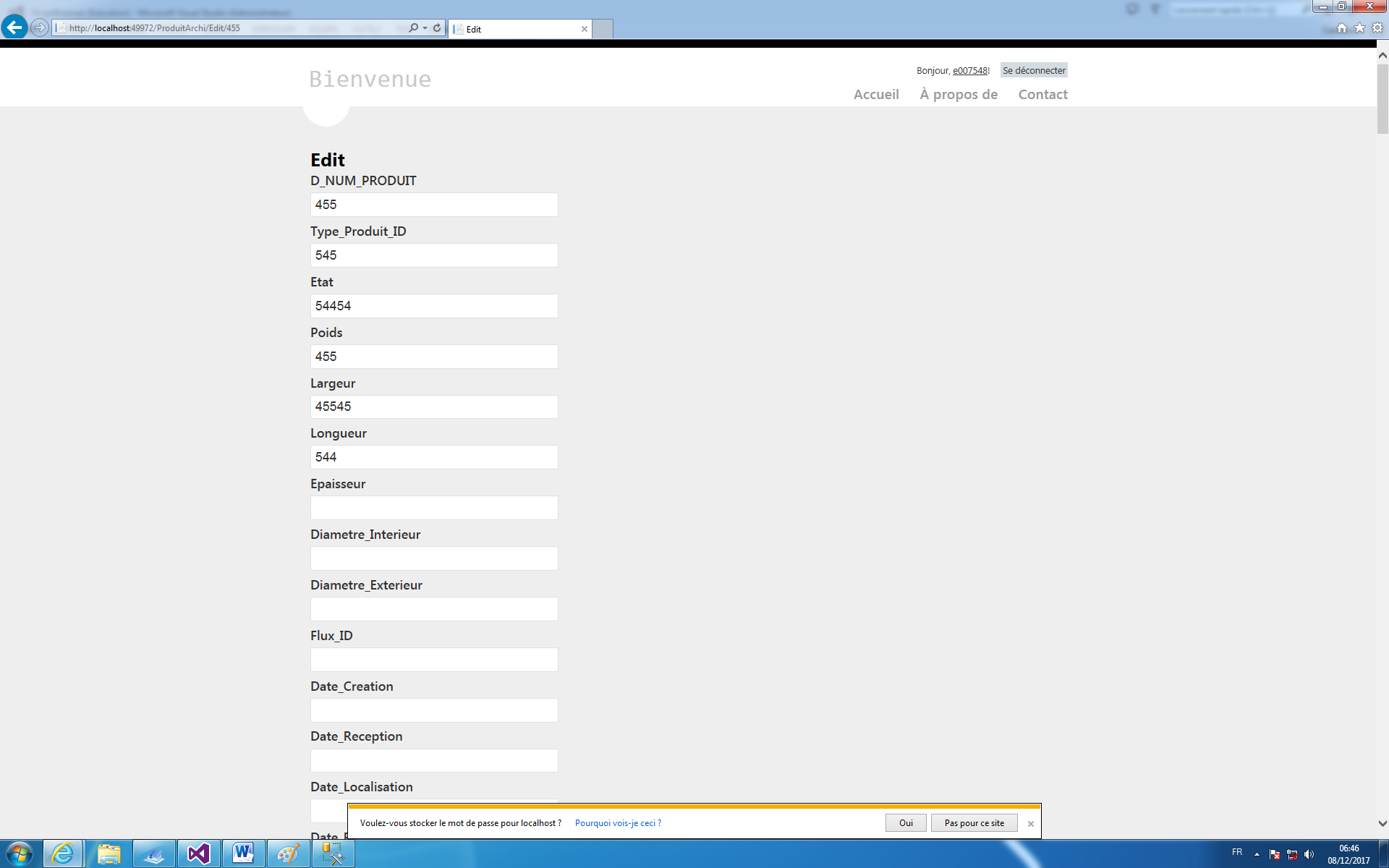
### La page « Produits Archi»

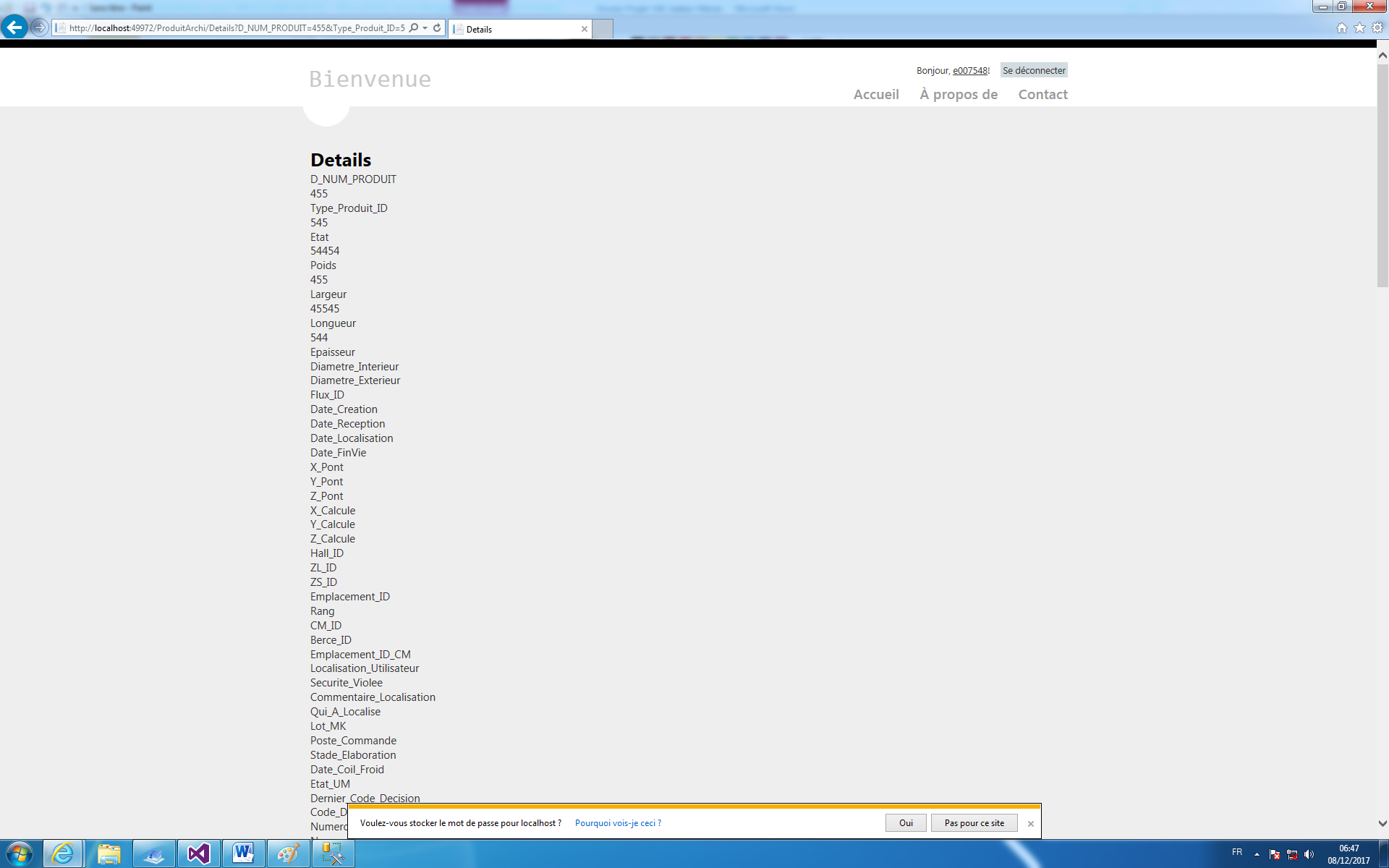
Cette page permet de gérer les données des produits en base de données. Elle se présente sous la forme d’une liste de produits présents dans la table « produits » sur laquelle l’utilisateur (suivant le rôle qui lui est attribué) peut :

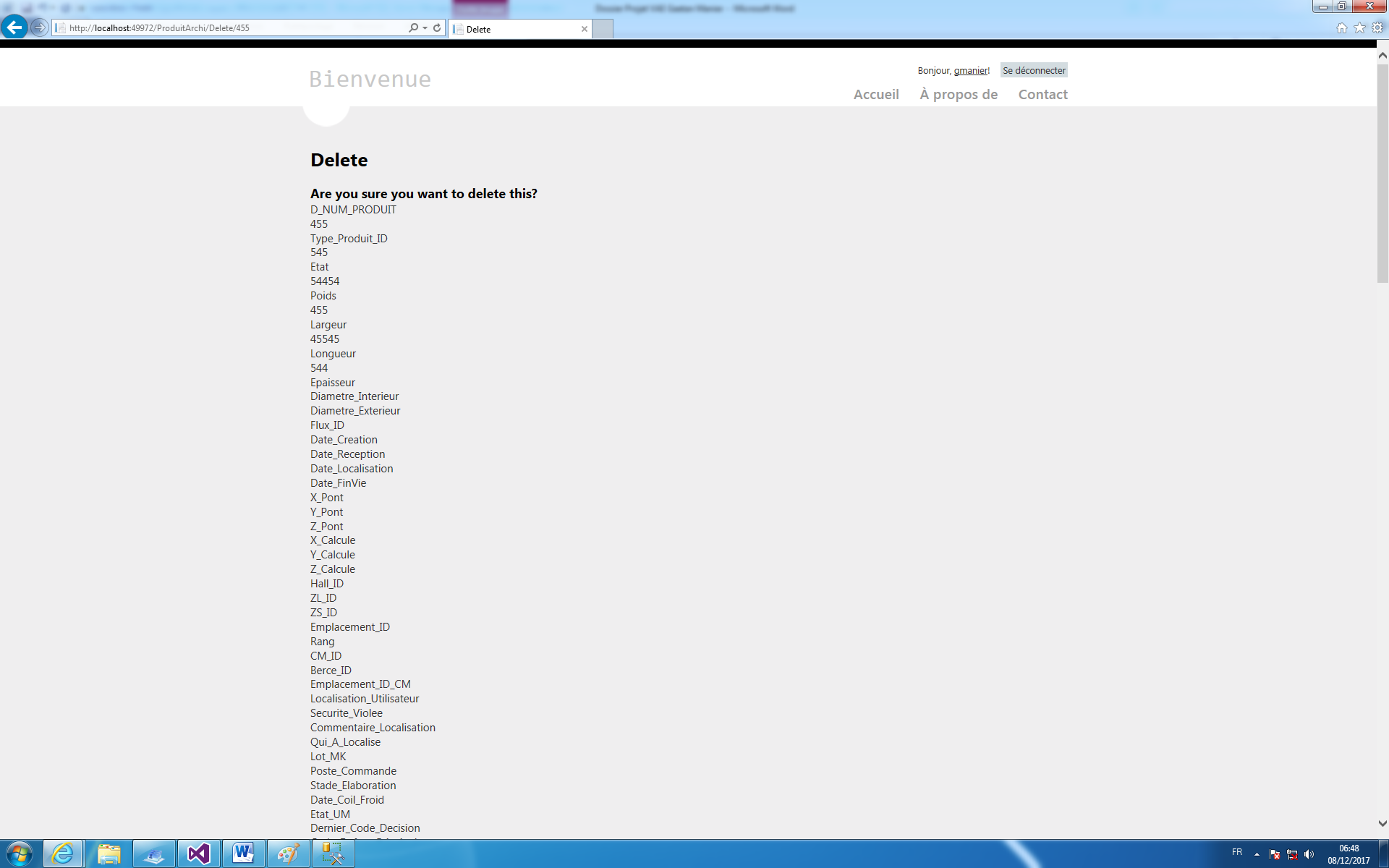
* Créer un nouveau produit
* Consulter le détail d’un produit
* Modifier un produit
* Supprimer un produit.











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

### 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 datawarehouse :

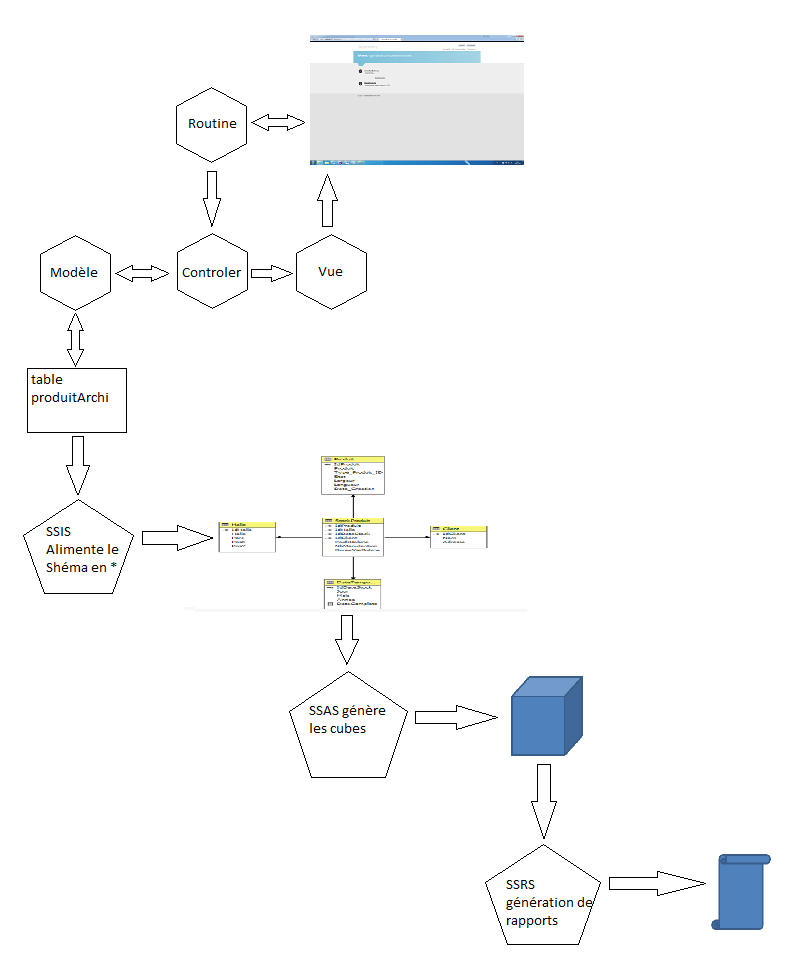
# Architecture du logiciel

## 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



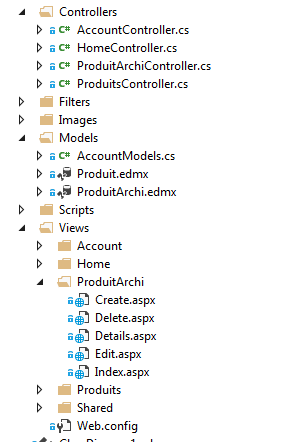
index.php?query=item/ranger-courses
point
d’entrée
Kernel::run()
$cont = new ItemController
 

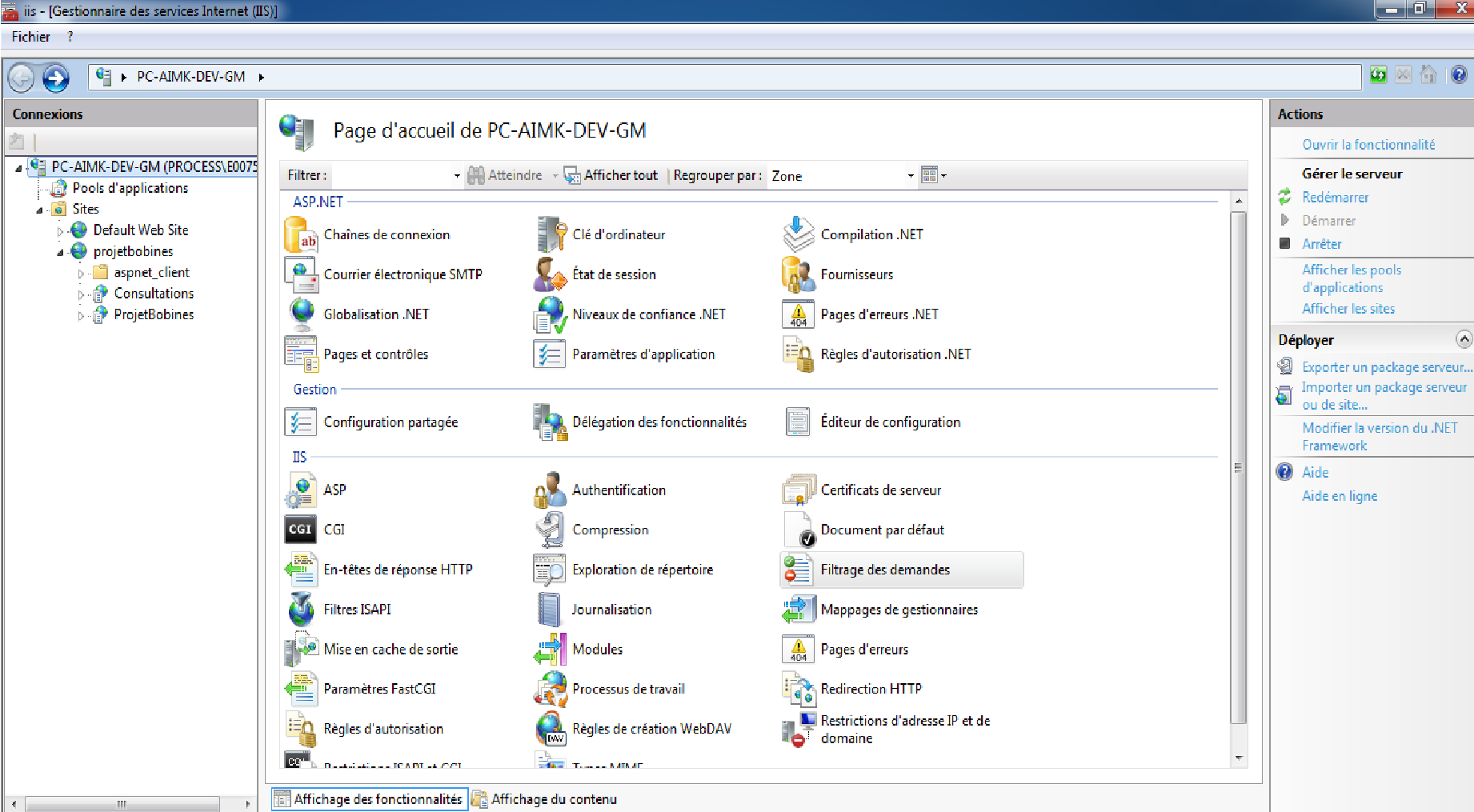
 ("ranger-cours...



## 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

## Introduction aux Data WareHouse

Dans le cahier des charges, il ma été demandé de réaliser un data warehouse afin de structurer l’archivage des données et d’en faciliter l’interrogation.

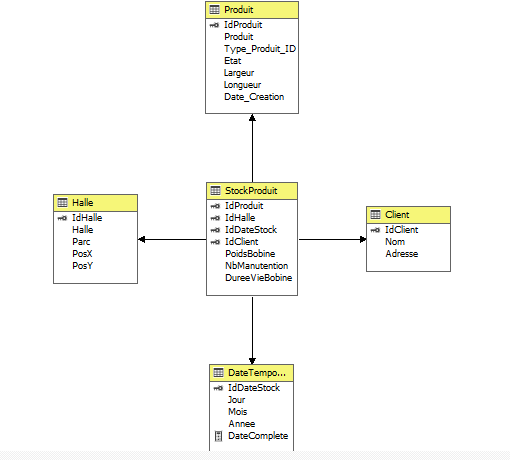
En temps normal j’utilise des bases de données de type SGDB (système de gestion de base de données relationnelles) dans nos applications car elles répondent aux besoins de stockage de données bidimensionnelles de production.

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

L’utilisation dudata whareHouse modifie en profoindeur la composition de la base de données. D’abord la base de données est gérée par une serveur OLAP (Online analytical processing).

## Choix du schéma

Dans notre cas, le choix a été fait pour avoir un schéma en étoile.



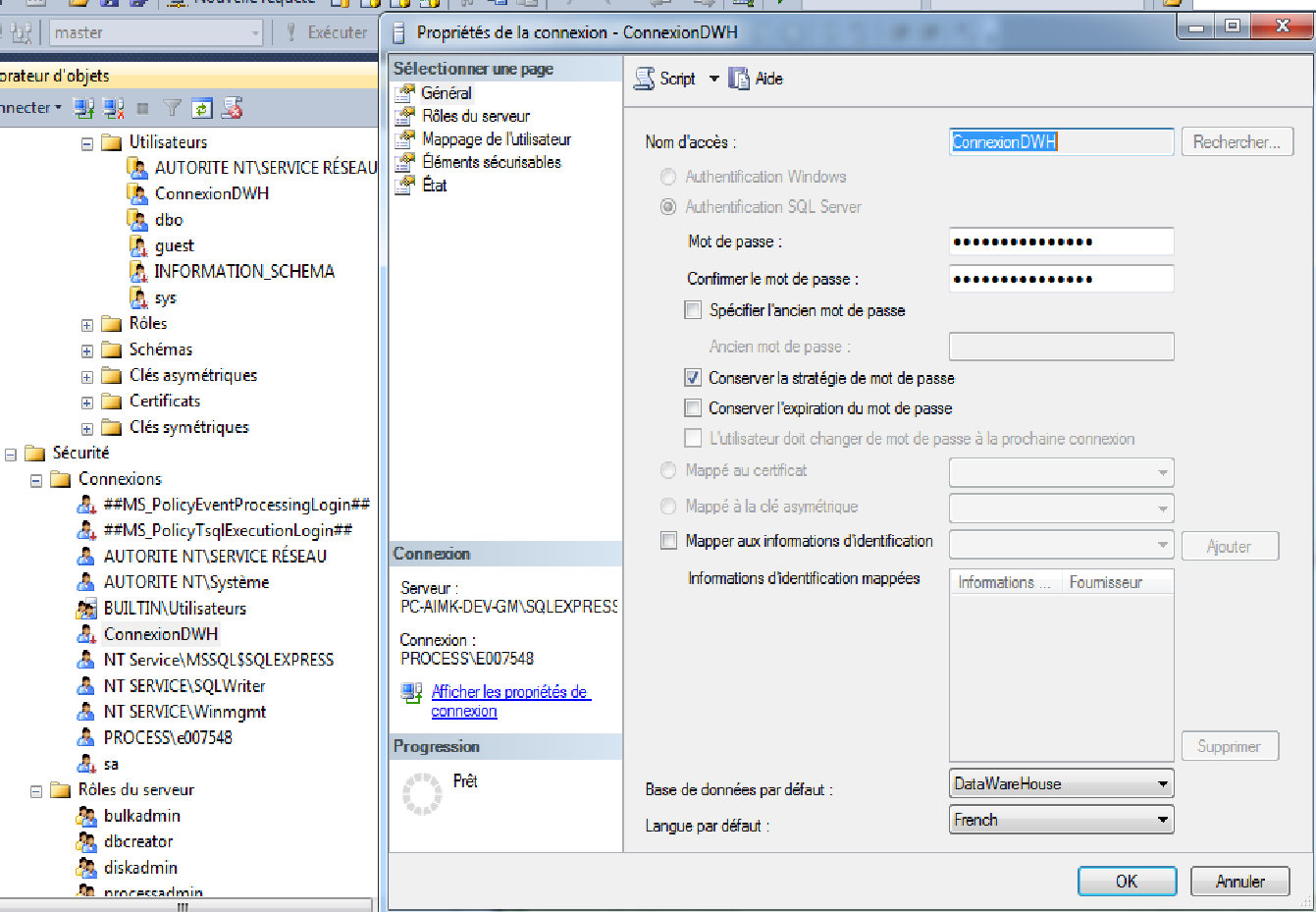
## Configuration du serveur de base de données

Afin de pouvoir utiliser les outils SSIS, SSAS et SSDT ont été rajoutés sur le serveur de la base de données car ils ne sont pas en standard dans les masters servant à construire les serveurs. Cette manipulation a été réalisée par l’équipe support d’infrastructure.

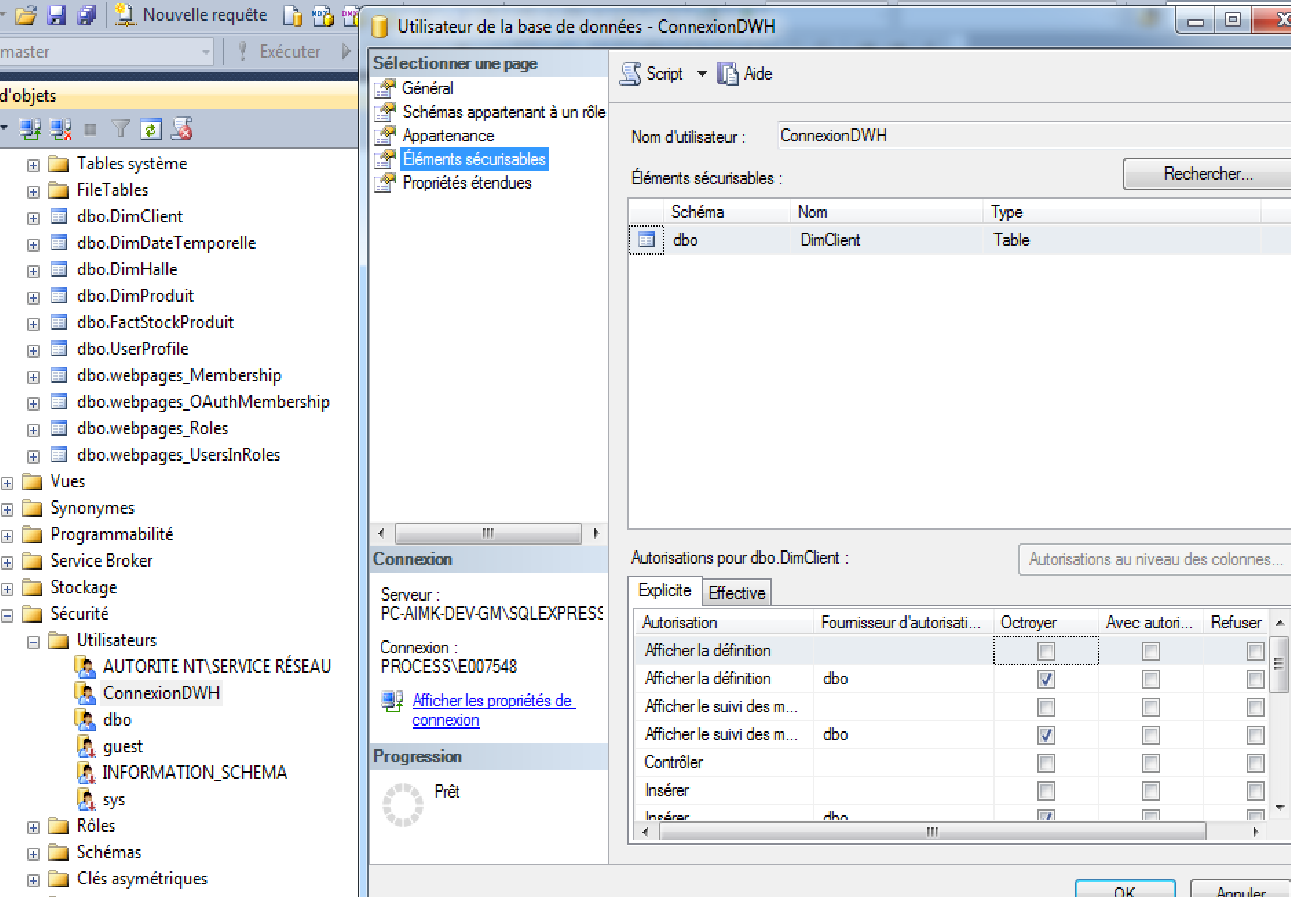
## Compte de connexion

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».



## Alimentation du Dataware house

### Configuration de l’ETL

Pour pouvoir aliementer mon dataware house j’ai programmée un ETL en utilisation une solution « integration services » du module business intelligence de visual studio 2013.

« La tâche de flux de données encapsule le moteur de flux de données chargé de déplacer des données entre les sources et les destinations et permet à l'utilisateur de transformer, nettoyer et modifier les données au cours de leur déplacement.The Data Flow task encapsulates the data flow engine that moves data between sources and destinations, and lets the user transform, clean, and modify data as it is moved. L'ajout d'une tâche de flux de données à un flux de contrôle de package permet au package d'extraire, de transformer et de charger des données.Addition of a Data Flow task to a package control flow makes it possible for the package to extract, transform, and load data.

Un flux de données est constitué d'au moins un composant de flux de données, mais il s'agit en général d'un ensemble de composants de flux de données connectés : des sources qui extraient des données, des transformations qui modifient, acheminent ou résument des données, et des destinations qui chargent des données.A data flow consists of at least one data flow component, but it is typically a set of connected data flow components: sources that extract data; transformations that modify, route, or summarize data; and destinations that load data.

Au moment de l'exécution, la tâche de flux de données crée un plan d'exécution à partir du flux de données et le moteur de flux de données exécute le plan.At run time, the Data Flow task builds an execution plan from the data flow, and the data flow engine executes the plan. Vous pouvez créer une tâche de flux de données qui n'a aucun flux de données, mais la tâche s'exécute uniquement si elle inclut au moins un flux de données.You can create a Data Flow task that has no data flow, but the task executes only if it includes at least one data flow.

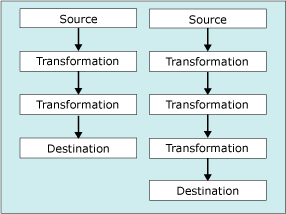
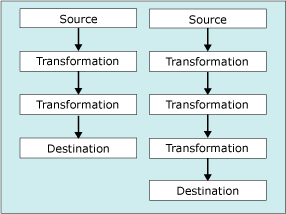
Pour insérer des données en bloc dans une base de données SQL ServerSQL Server à partir de fichiers texte, vous pouvez utiliser la tâche d'insertion en bloc plutôt qu'une tâche de flux de données et un flux de données.To bulk insert data from text files into a SQL ServerSQL Server database, you can use the Bulk Insert task instead of a Data Flow task and a data flow. Cependant, la tâche d'insertion en bloc ne peut pas transformer des données.However, the Bulk Insert task cannot transform data. Pour plus d’informations, consultez [Tâche d’insertion en bloc](https://docs.microsoft.com/fr-fr/sql/integration-services/control-flow/bulk-insert-task).For more information, see [Bulk Insert Task](https://docs.microsoft.com/fr-fr/sql/integration-services/control-flow/bulk-insert-task).

## Flux multiplesMultiple Flows

Une tâche de flux de données peut contenir plusieurs flux de données.A Data Flow task can include multiple data flows. Si une tâche copie plusieurs ensembles de données et si l'ordre dans lequel les données sont copiées n'est pas significatif, il peut être plus commode d'inclure plusieurs flux de données dans la tâche de flux de données.If a task copies several sets of data, and if the order in which the data is copied is not significant, it can be more convenient to include multiple data flows in the Data Flow task. Par exemple, vous pourriez créer cinq flux de données, chacun copiant des données à partir d'un fichier plat vers une table de dimension différente dans un schéma d'entrepôt de données en étoile.For example, you might create five data flows, each copying data from a flat file into a different dimension table in a data warehouse star schema.

Toutefois, le moteur de flux de données détermine l'ordre d'exécution lorsqu'une tâche de flux de données contient plusieurs flux de données.However, the data flow engine determines order of execution when there are multiple data flows within one data flow task. Par conséquent, lorsque l'ordre est important, le package doit utiliser plusieurs tâches de flux de données, chacune contenant un flux de données.Therefore, when order is important, the package should use multiple Data Flow tasks, each task containing one data flow. Vous pouvez ensuite appliquer des contraintes de précédence afin de contrôler l'ordre d'exécution des tâches.You can then apply precedence constraints to control the execution order of the tasks.

Le schéma suivant illustre une tâche de flux de données contenant plusieurs flux de données.The following diagram shows a Data Flow task that has multiple data flows.



«

### Déploiement

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

« Le catalogue SSISDB est l’élément central pour l’utilisation des projets Integration ServicesIntegration Services (SSIS) que vous avez déployés sur le serveur Integration ServicesIntegration Services.The SSISDB catalog is the central point for working with Integration ServicesIntegration Services (SSIS) projects that you’ve deployed to the Integration ServicesIntegration Services server. Ainsi, c'est dans ce catalogue que vous définissez les paramètres de projet et de package, configurez les environnements pour spécifier des valeurs d'exécution pour les packages, exécutez et résolvez les problèmes relatifs aux packages, et gérez les opérations du serveur Integration ServicesIntegration Services .For example, you set project and package parameters, configure environments to specify runtime values for packages, execute and troubleshoot packages, and manage Integration ServicesIntegration Services server operations.

Les objets stockés dans le catalogue SSISDB sont les projets, les packages, les paramètres, les environnements et l'historique opérationnel.The objects that are stored in the SSISDB catalog include projects, packages, parameters, environments, and operational history.

Vous inspectez les objets, les paramètres et les données opérationnelles stockés dans le catalogue SSISDB en interrogeant les vues de la base de données SSISDB .You inspect objects, settings, and operational data that are stored in the SSISDB catalog, by querying the views in the SSISDB database. Vous gérez des objets en appelant des procédures stockées situées dans la base de données SSISDB ou à l'aide de l'interface utilisateur du catalogue SSISDB .You manage the objects by calling stored procedures in the SSISDB database or by using the UI of the SSISDB catalog. Dans de nombreux cas, la même tâche peut être effectuée dans l'interface utilisateur ou en appelant une procédure stockée.In many cases, the same task can be performed in the UI or by calling a stored procedure.

«

# 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() {

...

}

**Codage des comptes et mots de passe de web.app**

# DpapiProtectedConfigurationProvider classe

**.NET Framework (current version)**

Fournit un objet [ProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.protectedconfigurationprovider(v=vs.110).aspx) qui utilise l’API de protection des données Windows (DPAPI) pour chiffrer et déchiffrer des données de configuration.

**Espace de noms:**   [System.Configuration](https://msdn.microsoft.com/fr-fr/library/system.configuration(v=vs.110).aspx)  
**Assembly:**  System.Configuration (dans System.Configuration.dll)

# Vue d'ensemble de la configuration protégée

**Visual Studio 2010**

* [Visual Studio 2008](https://msdn.microsoft.com/fr-fr/library/hh8x3tas(v=vs.90).aspx)
* [Afficher toutes les versions retirées](https://msdn.microsoft.com/en-us/mt703209?prefilter=visual%20studio%202005)

Mise à jour : novembre 2007

Vous pouvez utiliser la configuration protégée pour chiffrer des informations sensibles, y compris les noms d'utilisateurs et les mots de passe, les chaînes de connexion à une base de données et les clés de chiffrement, dans un fichier de configuration de l'application Web tel que le fichier Web.config. Le chiffrement des informations de configuration peut améliorer la sécurité de votre application dans la mesure où il sera très difficile à un intrus d'accéder à des informations sensibles même s'il parvient à accéder à votre fichier Web.config.

Par exemple, un fichier de configuration non chiffré peut contenir une section qui spécifie des chaînes de connexion à une base de données, comme l'illustre l'exemple suivant :

[Copier](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_5b30c7f4-3215-4089-b129-4297c72198ce');" \o "Copier dans le Presse-papiers.)

<configuration>

<connectionStrings>

<add name="SampleSqlServer" connectionString="Data Source=localhost;Integrated Security=SSPI;Initial Catalog=Northwind;" />

</connectionStrings>

</configuration>

Un fichier de configuration qui chiffre les valeurs des chaînes de connexion à l'aide de la configuration protégée n'affiche pas les chaînes de connexion en texte clair mais les stocke au lieu de cela sous une forme chiffrée, comme l'illustre l'exemple suivant :

[Copier](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_dc1675b0-0036-4493-bcba-4405871a9b33');)

<configuration>

<connectionStrings configProtectionProvider="RsaProtectedConfigurationProvider">

<EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"

xmlns="http://www.w3.org/2001/04/xmlenc#">

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />

<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

<EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5" />

<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

<KeyName>RSA Key</KeyName>

</KeyInfo>

<CipherData>

<CipherValue>RXO/zmmy3sR0iOJoF4ooxkFxwelVYpT0riwP2mYpR3FU+r6BPfvsqb384pohivkyNY7Dm4lPgR2bE9F7k6TblLVJFvnQu7p7d/yjnhzgHwWKMqb0M0t0Y8DOwogkDDXFxs1UxIhtknc+2a7UGtGh6Di3N572qxdfmGfQc7ZbwNE=

</CipherValue>

</CipherData>

</EncryptedKey>

</KeyInfo>

<CipherData>

<CipherValue>

</CipherValue>

</CipherData>

</EncryptedData>

</connectionStrings>

Lors d'une demande de la page, le .NET Framework déchiffre les informations de chaîne de connexion et les rend disponibles à votre application.

|  |
| --- |
| **hh8x3tas.alert_note(fr-fr,VS.100).gifRemarque :** |
| Vous ne pouvez pas utiliser la configuration protégée pour chiffrer la section configProtectedData d'un fichier de configuration. Vous ne pouvez pas non plus utiliser la configuration protégée pour chiffrer les sections de configuration qui n'emploient pas de gestionnaire de section ou des sections qui font partie de la configuration de chiffrement managée. La liste suivante répertorie les sections de configuration qui ne peuvent pas être chiffrées à l'aide de la configuration protégée : processModel, runtime, mscorlib, startup, system.runtime.remoting, configProtectedData, satelliteassemblies, cryptographySettings, cryptoNameMapping et cryptoClasses. Il est recommandé d'utiliser d'autres méthodes de chiffrement des informations sensibles, par exemple l'application console ASP.NET Set Registry (**Aspnet\_setreg.exe**), pour protéger les informations sensibles de ces sections de configuration. Pour plus d'informations sur l'application console ASP.NET Set Registry (**Aspnet\_setreg.exe**), consultez l'article Q329290 sur l'utilisation de l'outil ASP.NET pour le chiffrement des informations d'identification et des chaînes de connexion d'état de session, dans la Base de connaissances Microsoft du [site Web de support technique Microsoft](http://go.microsoft.com/fwlink/?LinkId=37114) |
| **hh8x3tas.alert_security(fr-fr,VS.100).gifNote de sécurité :** |
| Les informations de configuration chiffrées sont déchiffrées lors de leur chargement dans la mémoire utilisée par votre application. Si la mémoire de votre application est compromise, les informations sensibles de votre section de configuration protégée peuvent l'être aussi. |

## Utilisation de la configuration protégée

Vous gérez la configuration protégée à l'aide de l'outil ASP.NET IIS Registration (**Aspnet\_regiis.exe**) ou des classes de configuration protégée de l'espace de noms [System.Configuration](https://msdn.microsoft.com/fr-fr/library/system.configuration(v=vs.100).aspx).

L'outil **Aspnet\_regiis.exe** (installé dans le dossier %SystemRoot%\Microsoft.NET\Framework\NuméroVersion) inclut des options permettant de chiffrer et de déchiffrer des sections d'un fichier Web.config, de créer ou de supprimer des conteneurs de clé, d'exporter et d'importer des informations sur les conteneurs de clé et enfin de gérer l'accès à un conteneur de clé.

Le chiffrement et le déchiffrement du contenu d'un fichier Web.config sont exécutés à l'aide d'une classe [ProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.protectedconfigurationprovider(v=vs.100).aspx). La liste suivante décrit les fournisseurs de configuration protégée inclus dans le .NET Framework :

* [DpapiProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.dpapiprotectedconfigurationprovider(v=vs.100).aspx). Utilise l'API de protection des données (DPAPI) Windows pour chiffrer et déchiffrer des données.
* [RsaProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.rsaprotectedconfigurationprovider(v=vs.100).aspx). Utilise l'algorithme de chiffrement RSA pour chiffrer et déchiffrer des données.

Les deux fournisseurs offrent un chiffrement renforcé des données. Toutefois, si vous prévoyez d'utiliser le même fichier de configuration chiffré sur plusieurs serveurs, par exemple une batterie de serveurs Web, seul [RsaProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.rsaprotectedconfigurationprovider(v=vs.100).aspx) vous permet d'exporter les clés de chiffrement servant à chiffrer les données et de les importer sur un autre serveur.

Vous pouvez spécifier le [ProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.protectedconfigurationprovider(v=vs.100).aspx) que vous souhaitez utiliser en le configurant dans le fichier Web.config de votre application ou vous pouvez utiliser l'une des instances de [ProtectedConfigurationProvider](https://msdn.microsoft.com/fr-fr/library/system.configuration.protectedconfigurationprovider(v=vs.100).aspx) configurée dans le fichier Machine.config. Pour plus d'informations, consultez [Spécification d'un fournisseur de configuration protégée](https://msdn.microsoft.com/fr-fr/library/68ze1hb2(v=vs.100).aspx).

Une fois que vous avez spécifié le fournisseur à utiliser, vous pouvez chiffrer ou déchiffrer le contenu du fichier Web.config de votre application. Pour plus d'informations, consultez [Chiffrement et déchiffrement des sections de configuration](https://msdn.microsoft.com/fr-fr/library/zhhddkxy(v=vs.100).aspx).

**/////////////////////////////////////////////**

|  |
| --- |
| Lors de la création d'un formulaire, la sécurisation de celui-ci est bien souvent mise de côté. Si le formulaire n'est pas assez sécurisé alors l’attaquant pourra obtenir diverses informations sur notre site (mot de passe, schéma de base de données, etc. ). Cela peux donc engendrer des conséquences désastreuses et porter atteinte à l’image du site. Dans cet article nous allons voir les différents types d’attaques et les moyens mis à notre disposition pour les éviter. |

Voici la liste des différents types d’attaques auxquels nous pouvons être confrontés si on ne protège pas nos formulaires :

* **XSS** (Cross Site Scripting) : Injection de code indésirable dans des paramètres d’une URL ou dans un champ d’un formulaire.
* **CSRF** (Cross Site Request Forgery) : Elle consiste à effectuer des opérations sur un site sans le consentement d’un utilisateur. (Pour de plus amples informations je vous invite à regarder ce lien http :/www. cert. ssi. gouv. fr/site/CERTA-2008-INF-003/)
* **Injection SQL** : Modification d’une requête SQL existante grâce à un champ mal filtré. Ce type d’attaque permet de récupérer des informations sur la base de données, ‘bypasser’ l’authentification etc. L'attaquant se sert soit de l'url ou bien d'un formulaire.
* **Parameter Tampering :** Attaque basée sur un formulaire en passant certains paramètres dans l'URL ou changement d’un champ d'un formulaire sans l'autorisation de l'utilisateur.
* **OverPosting** : Poster des propriétés qui n’apparaissent pas dans le formulaire mais qui sont contenues dans le modèle du formulaire.

Il existe quelques techniques qui permettent d’éviter ou minimiser l'impact d’attaques :

* La méthode du formulaire
* ValidateAntiForgery
* Clics multiples
* Le contrôle des données
* La spécialisation un modèle de classe pour chaque vue.
* Captcha

# La méthode du formulaire

Lors de la création d’un formulaire, on a le choix de faire transiter l’information par méthode GET ou POST.

La méthode GET, privilégiée surtout pour la recherche, transmet les informations dans l'url, cependant la taille est généralement limitée à 260 caractères. On peut modifier cette limite dans le web. config en utilisant 2 attributs de l’élément HttpRuntime.

<httpRuntime maxRequestPathLength="260" maxQueryStringLength="2048" />

La méthode POST, privilégiée pour l’ajout, la modification et le formulaire de connexion, n’a pas de limite de caractères dans l’url car l’information se situe dans le corps du message.

**Note :** si on écrit le code ci-dessous, la méthode par défaut sera le GET, penser à mettre [HttpGet] ou à retirer le [HttpPost] sur l’action du contrôleur qui reçoit le submit.

<form action="/home/create"> ... </form>

Par contre si on met le code ci-dessous, la méthode par défaut sera le POST

@using(Html.BeginForm("Create","Home")){ ... }

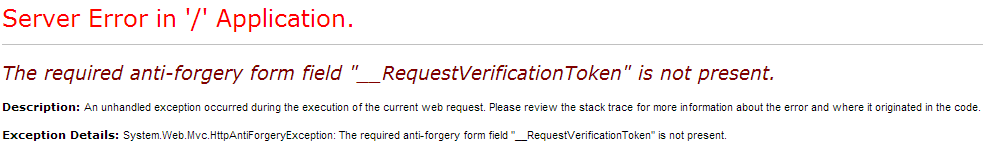
# ValidateAntiForgery

Pour se protéger d’attaques du type CSRF, le Framework MVC met à notre disposition l’attribut ValidateAntiForgeryToken.

Il suffit d’ajouter cet attribut dans la méthode d’action de notre contrôleur qui va recevoir le formulaire. Ensuite au niveau html dans notre formulaire, il suffit d’ajouter @Html. AntiForgeryToken () Au niveau du rendu de la page, cela va générer un token dans un champ caché qui sera récupéré lors de l’appel de notre action POST du formulaire.

<input name="\_\_RequestVerificationToken" type="hidden" value="hHPytP5nO6LULDPfHtD0-yrTiAxJrtGV29\_O7Figu\_\_gIR98E3t0aNYzSiWNGiO9YYESMQVXAduanp6QmMgAV7NuRrwVSCjV2dJyOmogJhY1">

Si le token est non présent lors de la création du formulaire et qu’il est mis sur l’action du contrôle alors par défaut il aura la page d’erreur ci-dessous :



# Clics multiples

Quand le formulaire met trop de temps à s'exécuter, les utilisateurs ont tendance à cliquer une nouvelle fois sur le bouton “Envoyer”. Cela a pour conséquence d'envoyer plusieurs fois la demande à notre méthode qui reçoit le POST et de potentiellement créer autant d'enregistrements. Pour corriger cela, il suffit d’ajouter un code JavaScript permettant sa désactivation lors du 1er clic :

<form action="/Home/Contact" id="idForm" method="post>

Nom : <input type="text" name="nom" />

Prenom : <input type="text" name="prenom" />

<input type="submit" id="btnSubmit" value="Envoyer" />

</form>

<script>

$("#idForm").submit(function()

{

if ($('#idForm').valid()) {

$("#btnSubmit").attr("disabled", "disabled");

return true;

}

});

</script>

# Le contrôle des données

### 1) Contrôle des champs visibles

Lors de l’envoi des informations d’un formulaire, l’attaquant va tenter d’ajouter un code dans un champ du formulaire pour ensuite récupérer des informations ou bien réussir à se connecter. Pour cela il faut faire des vérifications avant d’enregistrer les données en base. Et de même pour l’affichage.

**Le traitement des entrées**

Par défaut ASP.NET protège ce type d’attaque et envoie la page d’erreur ci-dessous. Pour certains besoins, on est obligé de retirer cette vérification, cela ce fait par le biais de l'attribut [ValidateInput (false)] que l'on ajoute sur l’action de notre contrôleur.

Une fois cet attribut mis, il faut ajouter notre propre validation en utilisant la bibliothèque **Sanitizer** qui contient les méthodes GetSafeHtml ou GetSafeHtmlFragment. Ces méthodes permettent de nettoyer le html pour pouvoir l’enregistrer en base de données.

Exemple :

string chaine1 = "<script>alert('test');</script>";

string chaine2 = "<h1>titre</h1>";

string safeHtmlFragment = Sanitizer.GetSafeHtmlFragment(chaine1);

// résultat : chaine vide

string safeHtmlFragment2 = Sanitizer.GetSafeHtmlFragment(chaine2);

// résultat : titre

string safeHtml = Sanitizer.GetSafeHtml(chaine1);

// résult : <html>\r\n<head>\r\n</head>\r\n</html>\r\n

string safeHtml2 = Sanitizer.GetSafeHtml(chaine2);

// résultat : <html>\r\n<body>\r\ntitre\r\n</body>\r\n</html>\r\n

**Traitement en sortie**

Pour le moteur de vue Razor, toutes les sorties qui font appel à un **HtmlHelper** sont encodées, on peut aussi utiliser Html. Encode. Pour certains cas dans lesquels on a besoin d'interpréter la sortie sans l'encodée, on utilisera @Html. Raw ().

Exemple :

@{

string message = "<script>alert('Cross Site Scripting Attack!');</script>";

}

@Html.Raw(message); // Affiche la message box

mon message : @(message) //<script>alert('Cross site scripting attack!');</script>

mon message : @Html.Encode(message) //&lt;script&gt;alert(&#39;Cross site scripting attack!&#39;);&lt;/script&gt;

### 2) Contrôle d'un champ caché

Pour les formulaires d'édition, on a l'habitude de mettre en champ caché l'Id de l'objet mais sans le protéger. Voici le résultat dans l'exemple suivant et les dégâts que cela peut occasionner.

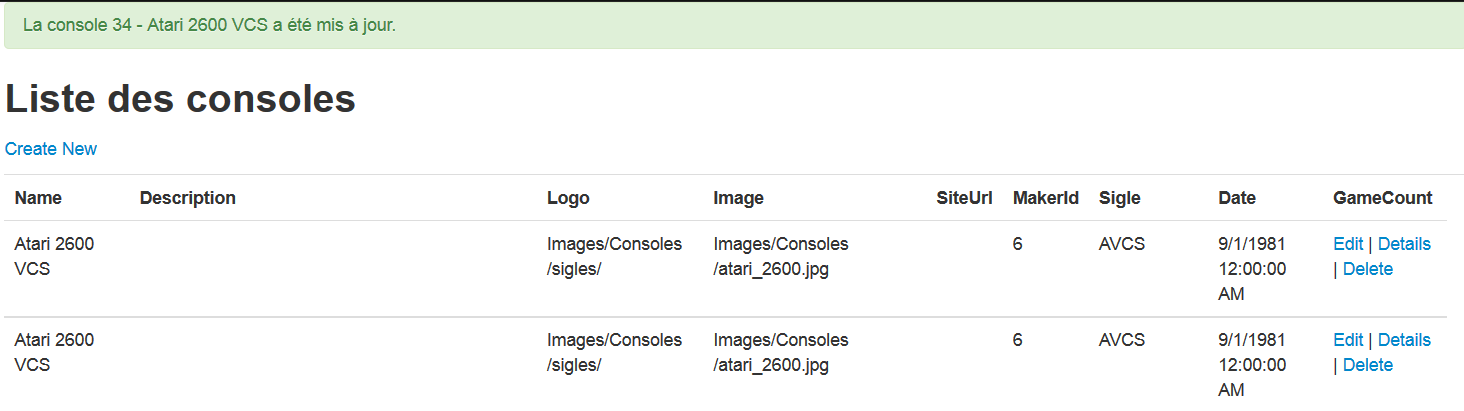
J'utilise Web Developper Tool dans FireFox pour afficher tous les champs de mon formulaire.



Je modifie directement le champ Id en ajoutant la valeur 34 (Capture ci-dessous)



Une fois mon formulaire validé, les informations contenues dans l'objet avec l'id 23 se retrouvent dans l'objet avec l'id 34.



Pour résoudre ce problème, il existe une solution qui consiste à ajouter notre champ caché Id et un champ caché contenant notre Id crypté. Lors du Post, on crée un **ActionFilter** qui va comparer l’id avec notre champ caché décrypté.

**La démarche**

J’utilise la Dll “Secure\_Coding. MvcSecurityExtensions” qui se trouve a cette adresse : http ://mvcsecurity. codeplex. com/

Dans mon formulaire en édition, on ajoute :

@Html. HiddenFor (model => model. Id)

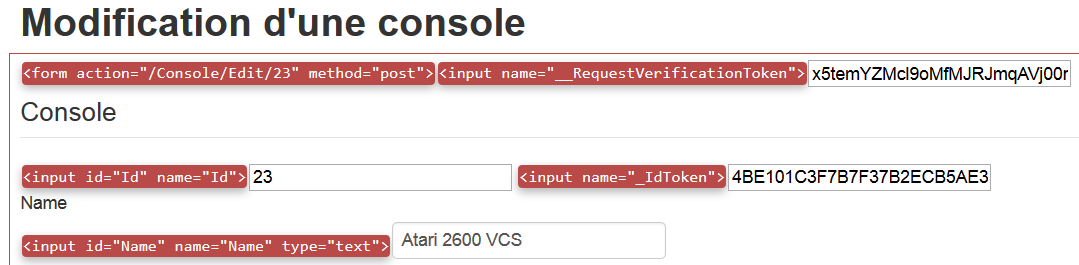
@Html. AntiModelInjectionFor (model => model. Id)

Ensuite dans mon action de mon contrôleur qui reçoit le formulaire en édition :

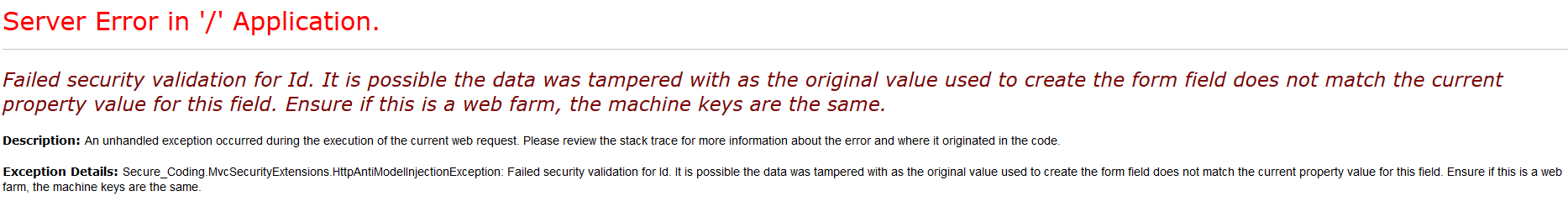
[ValidateAntiModelInjectionAttribute (”Id&rdquo ;)]

**Vérification en images**

Mon formulaire contient un champ caché nommé “\_IdToken” et un champ caché “Id”



Si je modifie le champ caché “Id”, j'ai un message d'erreur. (Capture ci-dessous)



# Le captcha

Le Captcha est une mesure de sécurité qui fait que la réponse ne peut pas être générée par un ordinateur mais par l’utilisateur. Cependant le Captcha n’est pas fiable à 100% car des personnes mal intentionnées peuvent utiliser des algorithmes de reconnaissance de caractère (OCR) pour obtenir le code. Il existe différentes manières de contourner un Captcha. Mais sa mise en place permet tout de même de se protéger du SPAM.

En ASP.NET MVC, il existe [reCaptcha](http://www.softfluent.fr/www.googlecom/recaptcha) ou encore CaptchaMvc.

# 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

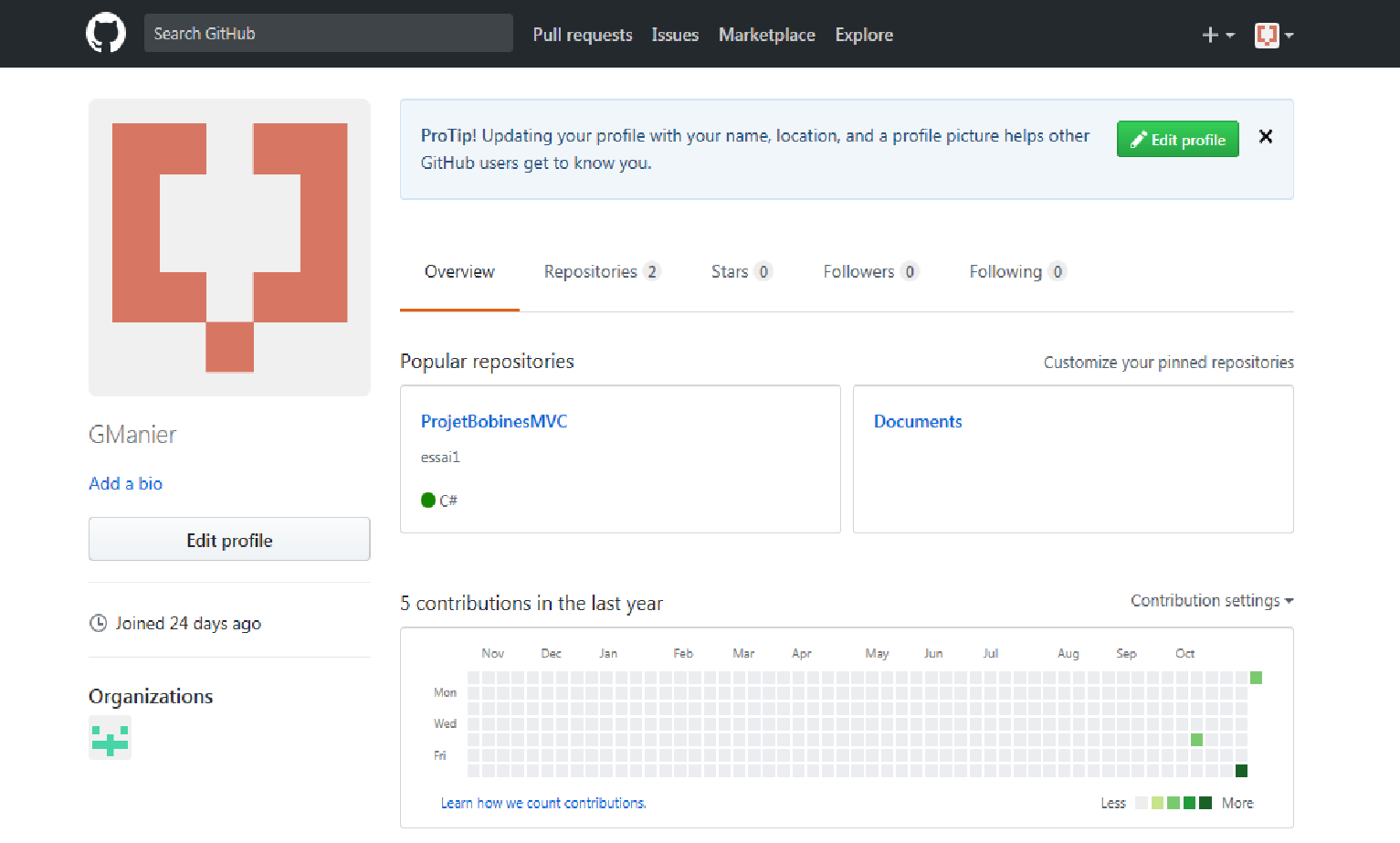
Sur le marché il existe plusieurs utilitaires de gestion de code sources et gestion des versions de fichiers. Pour n’en citer que quelques un, nous avons TFS (Team Foundation Server), Visual sourceSafe, Git, GitHub et BitBuchet.

Dans mon entreprise nous utilisons TFS (Team Foudation 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.

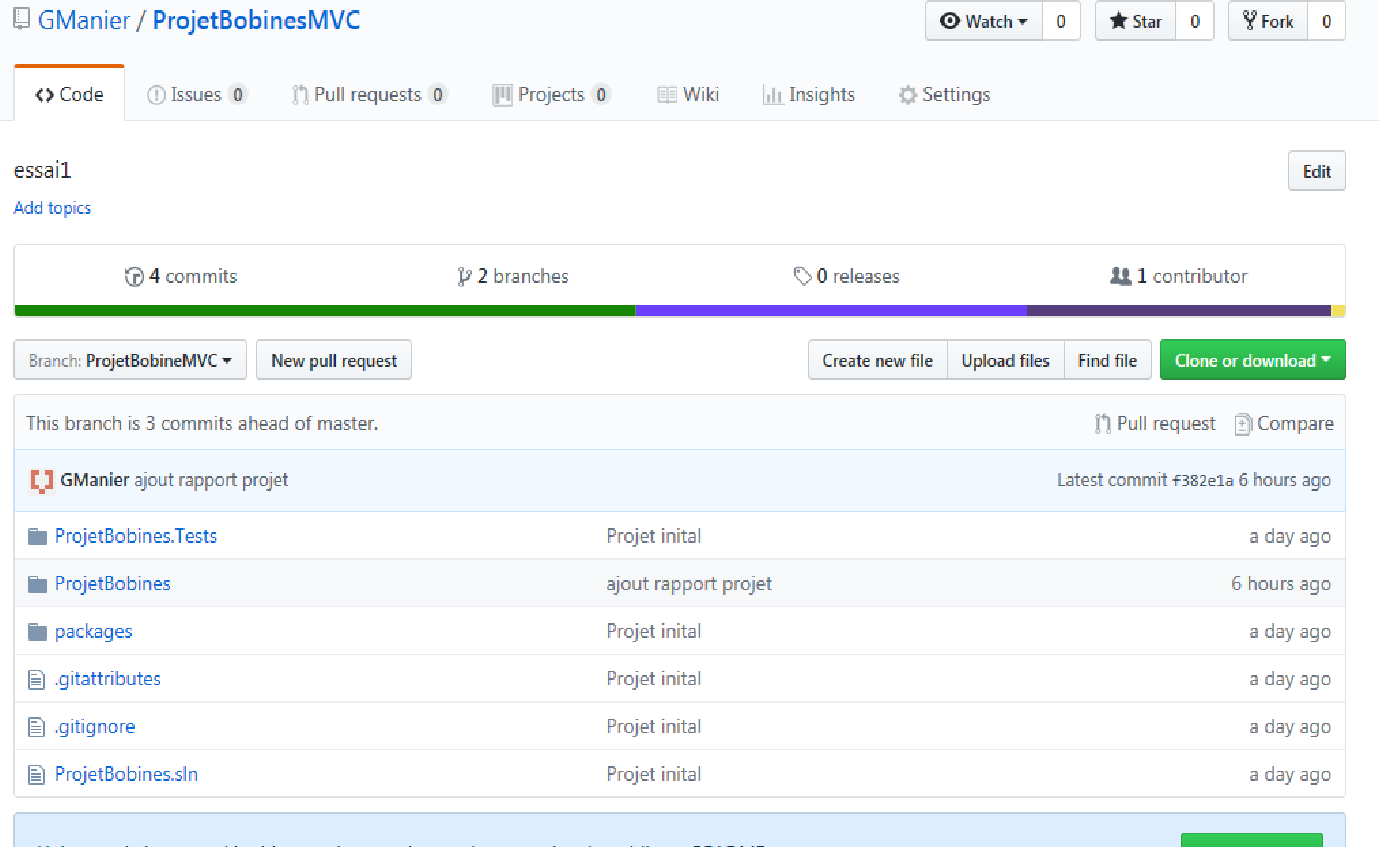
Comme Mon tuteur de projet devait également pouvoir au code source et aux documents que je créé, j’ai opté pour GitHub. L’avantage est qu’il s’interface directement avec Visual Studio 2013 (VS2013) et que l’extraction ou l’archivage des fichiers se fait naturellement avec la même souplesse que TFS.

## 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.

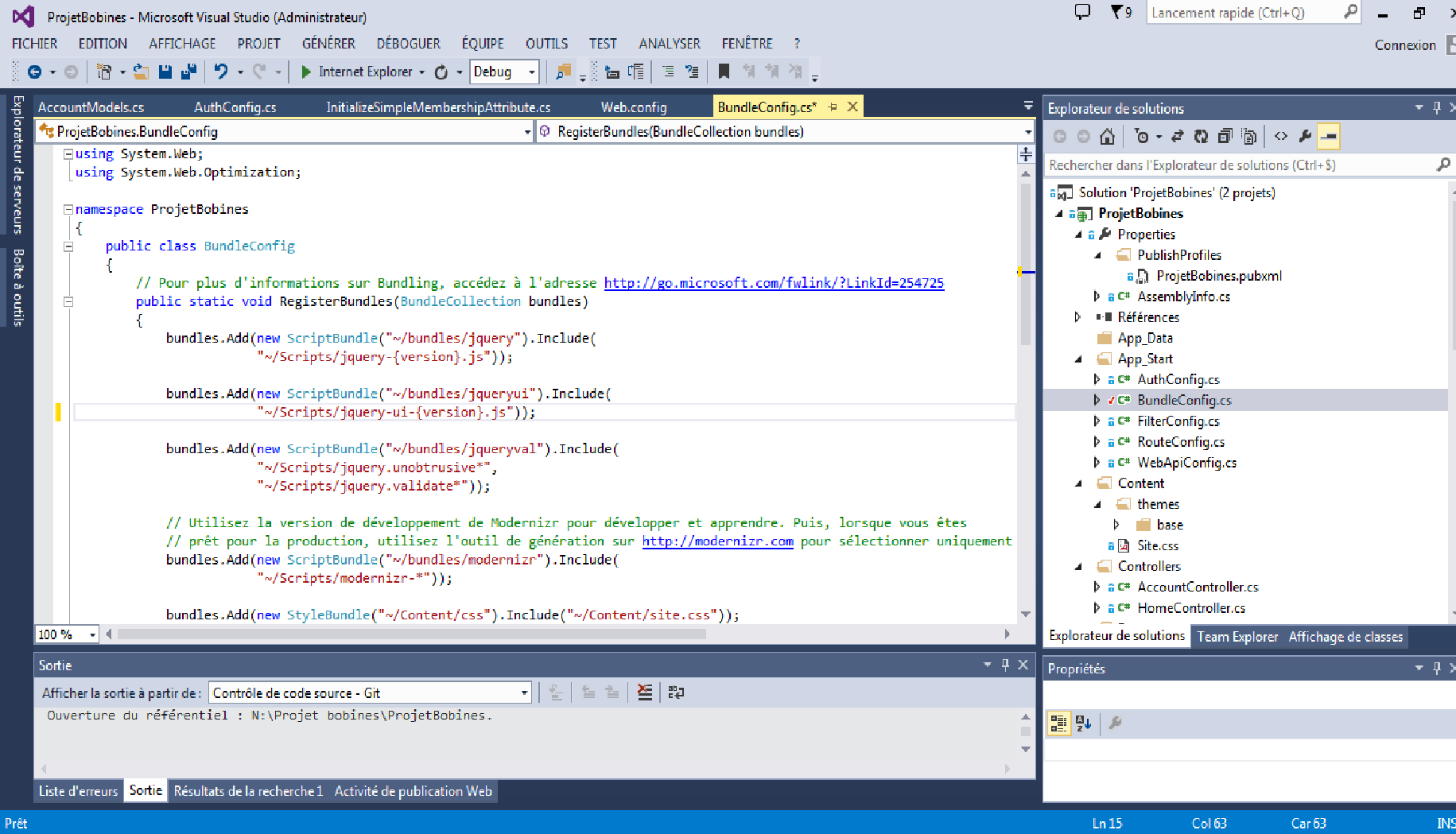


Une fois le compte créé, j’ai dû définir des « Repositories » afin de pouvoir y inclure les différentes solutions Visual Studio du projet. Il est possible ensuite de créer des branches de développements afin de gérer les versions en fonction soit des développements en cours, soit en recette, soit en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

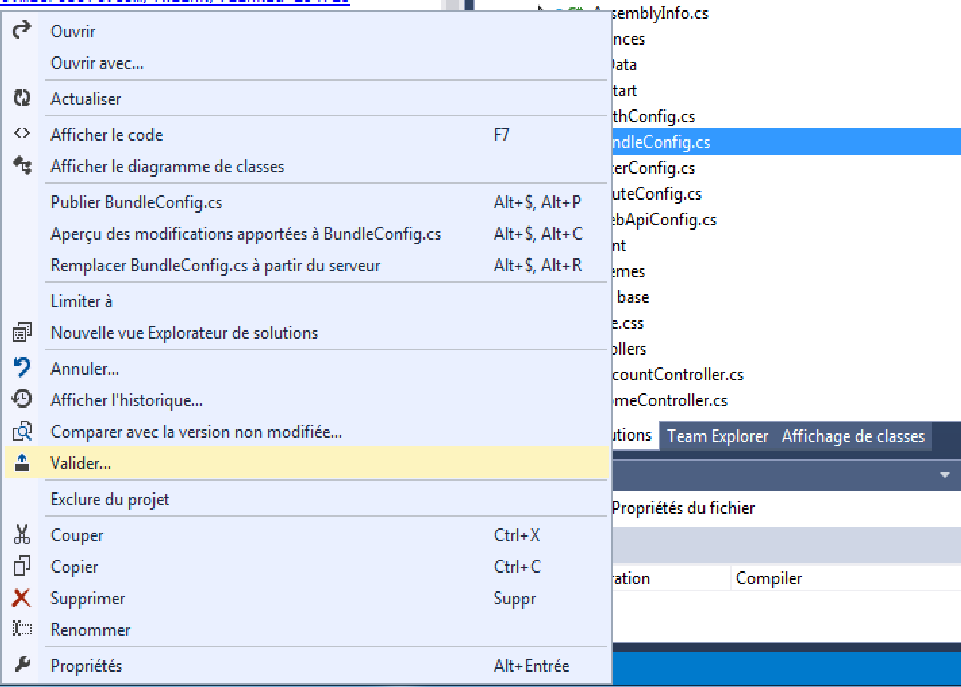


L’utilisation avec Visual Studio 2013 est assez simple car celui-ci propose naturellement 2 gestionnaires de code source, TFS ou Git.

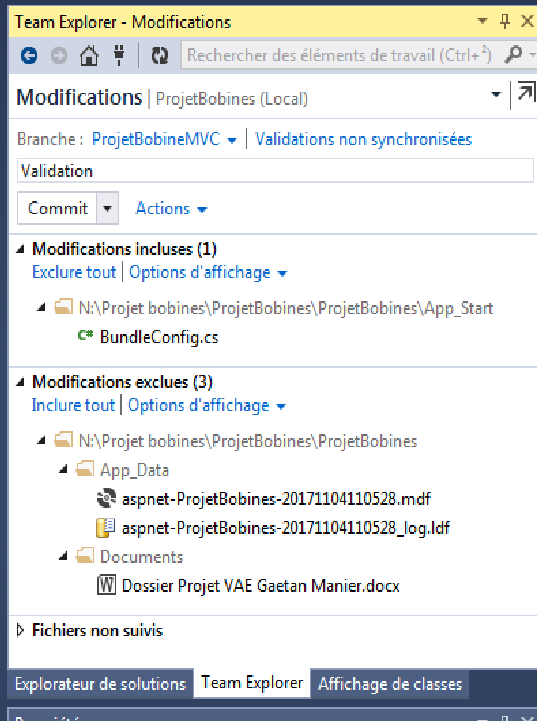
En sélectionnant Git, il faut déclarer son compte GitHub puis le « repositories » et branche sur laquelle on souhaite archiver ses projets.

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

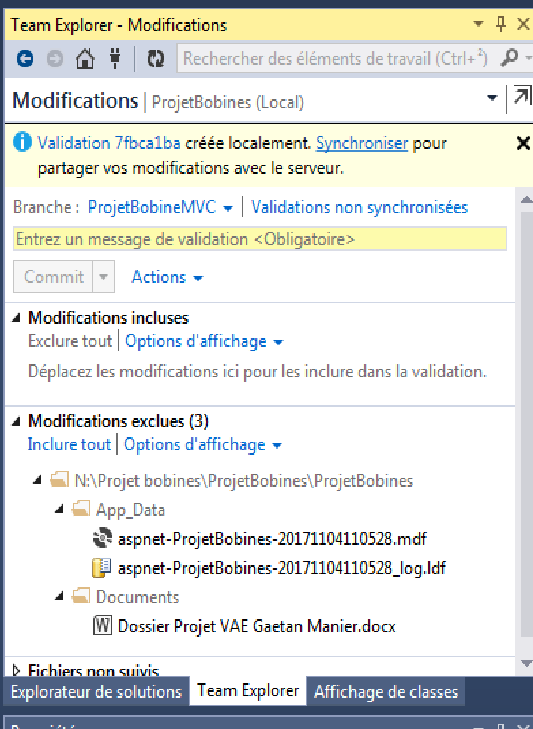
Pour archiver ses modifications il faut cliquer sur le fichier extrait, ou l’ensemble du projet et sélectionner « valider ».



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