Microsoft® Visual Studio®

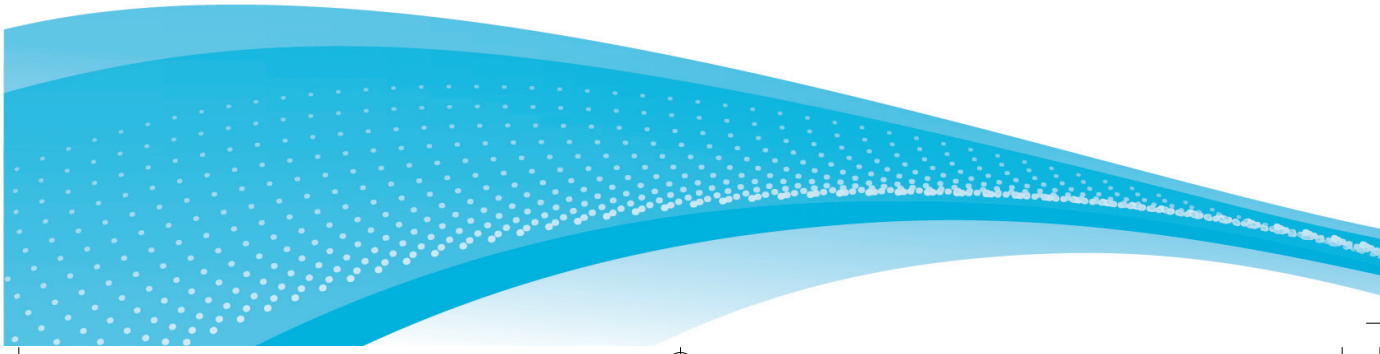
Développer avec

VISUAL STUDIO POUR WINDOWS AZURE

Microsoft®

SOGETI

Développer avec Visual Studio pour Windows Azure



Sommaire

Sommaire 2

Avertissement 3

Introduction 4

Contexte général, choix de Windows Azure 4

1. Contexte général – La société MCMG 4
2. Stratégie globale de MCMG 5
3. Les outils qui permettent d'y aller 7
4. L'application BDTHEQUE 13
5. Comment ouvrir son compte Windows Azure 13

Développement en local et déploiement sur Windows Azure 16

1. Création du projet et configuration 18
2. Déploiement sur Windows Azure 26

Architecture et design de la solution BDTHEQUE 34

1. Considérations propres aux applications Windows Azure 35
2. Architecture générale 41
3. Implémentation de la solution 52

ALM & Azure : Améliorer la qualité logicielle et la productivité 69

1. Le projet « BDTHEQUE » dans Team Foundation Server 2010 69
2. Build & déploiement automatisés de la solution 72
3. BDTHEQUE et les tests : la qualité omniprésente 78

Pour aller plus loin 85

Avertissement

Ce document est fourni uniquement à titre indicatif. MICROSOFT N'APPORTE AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, À CE DOCUMENT. Les informations figurant dans ce document, notamment les URL et les références aux sites Internet, peuvent être modifiées sans préavis. Les risques d'utiliser ce document ou ses résultats sont entièrement à la charge de l'utilisateur. Sauf indication contraire, les sociétés, les entreprises, les produits, les noms de domaine, les adresses électroniques, les logos, les personnes, les lieux et les événements utilisés dans ce document sont fictifs. Toute ressemblance avec des entreprises, noms d'entreprise, produits, noms de domaine, adresses électroniques, logos, personnes ou événements réels serait purement fortuite et involontaire.

Remerciements

Thomas Conté, Architecte, Microsoft France - http://blogs.msdn.com/thomasconte/
Benjamin Guinebertière, Architecte, Microsoft France - http://blogs.msdn.com/benjguin/
Benoit Launay, Chef de Produit Outils de développement - http://www.visualstudio.fr

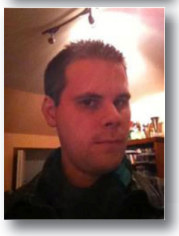
Auteurs

François MERAND – www.cloud-francois.com

National Practice Leader au sein de l'alliance Microsoft chez SOGETI France, en charge du pilotage et du développement des offres applicatives Microsoft. Responsable de l'équipe des .Net Rangers.

Jason DE OLIVEIRA - www.jasondeoliveira.com

Manager au sein de la BU Transformation Consulting de SOGETI France. Jason est MVP, et architecte en système d'information. Il pilote pour SOGETI France l'initiative « Azure Privilege Club ».



Laurent GAUTIER

Ingénieur d'étude & développement, BU Industrie Distribution & Services, SOGETI France. Diplômé de SUPINFO, Laurent a déjà travaillé sur plusieurs projets mettant en œuvre Windows Azure.

Introduction

La plateforme Windows Azure vous permet d'héberger, d'exécuter et de gérer vos applications et données dans le ou les centres de données Microsoft de votre choix. Grâce à Windows Azure, vous n'avez plus à gérer l'infrastructure liée à vos applications. En outre, la facturation de la plateforme Windows Azure est basée sur l'usage. Ainsi, vous payez à l'heure d'exécution de l'instance, au Go stocké par mois et à la bande passante consommée. Vous pouvez donc vous concentrer sur votre application et oubliez l'infrastructure.

Développer avec Visual Studio pour Windows Azure : c'est l'objet de ce livre blanc ! Nous avons décidé d'écrire ce livre blanc avec un objectif simple : permettre à chacun de comprendre comment appréhender un projet ciblant la plateforme Windows Azure et toutes ses composantes. Nous n'expliquerons donc pas en détail ce qu'est Windows Azure, il existe déjà de nombreuses sources d'informations sur le sujet, mais nous vous ferons découvrir de façon opérationnelle comment identifier un projet candidat pour Azure, le développer, le tester et le mettre en production.

Pour vous accompagner de façon très concrète, nous allons illustrer notre propos par une solution exemple : une société fictive, MCMG, qui à travers l'un de ses départements BD-MANIAC décide, après réflexion, de mettre en place une solution sur la plateforme Windows Azure, en migrant un de ses produits phares destiné aux collectionneurs de bandes dessinées (BDTHEQUE).

Ce livre blanc sera composé de plusieurs chapitres qui seront publiés régulièrement sur ce site. Le code source de l'application exemple est disponible ici http://bdtheque.codeplex.com

Contexte général, choix de Windows Azure

1. Contexte général – La société MCMG

La société MCMG est un groupe spécialisée dans l'édition de logiciels destinés au grand public. Son catalogue de produits et services couvre de nombreux domaines, sa spécialité étant la mise à disposition de solutions simples et performantes permettant aux collectionneurs du monde entier d'assouvir leurs passions. MCMG développe donc depuis

de nombreuses années des logiciels qui font le bonheur de ces collectionneurs, des numismates aux philatélistes en passant par les éphécalarophiles.

Son modèle de distribution pour ses applications repose sur la mise à disposition de DVD ou le téléchargement en ligne, ses produits se présentant sous la forme d'applications Windows traditionnelles. Le modèle commerciale est lui aussi classique les produits étant commercialisés sous forme de licences. Une incursion dans le monde en ligne a été tentée il y a un an, avec la mise à disposition d'une base de données en ligne pour les philatélistes. Les réflexions en cours au sein du groupe tendent à privilégier le modèle en ligne pour le futur des solutions du groupe. En complément, la société souhaite prendre en compte les aspects communautaires, les collectionneurs de tous bords étant avides d'échanges et de partages, pas seulement des objets constituants leurs collections, mais aussi de bonnes affaires, d'adresses et de conseils…

Il s'agit pour groupe d'une réflexion globale, à la fois concernant le futur de ses produits et de leurs cycles de développement, de son mode de distribution mais aussi de son modèle commerciale. MCMG a déjà entendu parler du Cloud et donc de Windows Azure. La société se pose de nombreuses questions sur l'impact que peut avoir le Cloud sur :

- Le développement de ses applications. Vaut-il mieux créer de nouvelles solutions ou migration est-elle une option « rentable » ?
- Les coûts, la sécurité d'une solution PaaS.
- Le cycle de vie d'une telle solution, la gestion des versions
- L'impact sur la stratégie commerciale, la communication
- Le modèle économique : comment passer d'un modèle de licence à un modèle de services : que facturer à quelle périodicité ?

Il est rapidement devenu évident pour le groupe que ces questions n'auraient des réponses tangibles et palpables que suite à la mise en œuvre d'un projet concret et réel.

2. Stratégie globale de MCMG

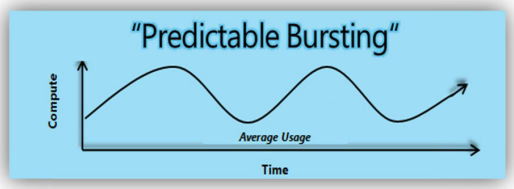
Avant de se lancer dans l'aventure, la direction du groupe a identifié quatre exigences principales, combinant les axes technologiques et métiers, en face desquels elle place un niveau d'attente en termes d'offres et de services attendus : comment Windows Azure réponds à ces exigences.

Association exigence métier et exigence technologique	Comment Windows Azure répond à ces exigences
Réduire les **coûts** en s'appuyant sur des capacités d'innovation technologiques	• **Une solution à moindre coût, ou du moins à coût maîtrisable** pour gérer les ressources IT • **Diminuer et rationaliser l'infrastructure** à acquérir, configurer et maintenir • **Contrôler et réduire le TCO** • Mettre en place une structure de **coûts prévisibles** • Adapter au plus près la consommation de ressources informatique aux besoins réels : **élasticité**
Rester **compétitif** au travers de **l'innovation**	• **Attention portée** sur le logiciel en tant que différentiateur sur le marché • **Monétiser de nouvelles offres** plus rapidement sans problème de facturation • **Se focaliser sur l'application**, pas sur l'infrastructure nécessaire à son exploitation
Générer rapidement de **nouveaux revenus**, en mettant en place des processus agiles	• **Accélérer le cycle de développement** • **Simplifier** le déploiement • **Commercialisation** plus rapide
Réduire les risques en s'assurant une fiabilité sans faille de son système	• **Qualité de services** : SLA • Prise en compte des problématiques de **Sécurité** • **Disponibilité de la solution.**

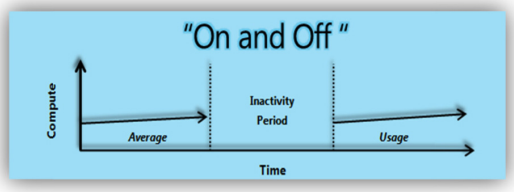
Les éléments de ce tableau constituent pour MCMG sa feuille de route pour son projet Cloud. Fort de ces éléments, le groupe MCMG retient que Windows Azure permet de valider les scénarios d'usage suivant :

Un scénario de **croissance rapide et difficilement prévisible** dans le temps pour l'ensemble des futures solutions en ligne

Un scénario de **pointes de charges prévisibles**, lors des lancements de nouveautés

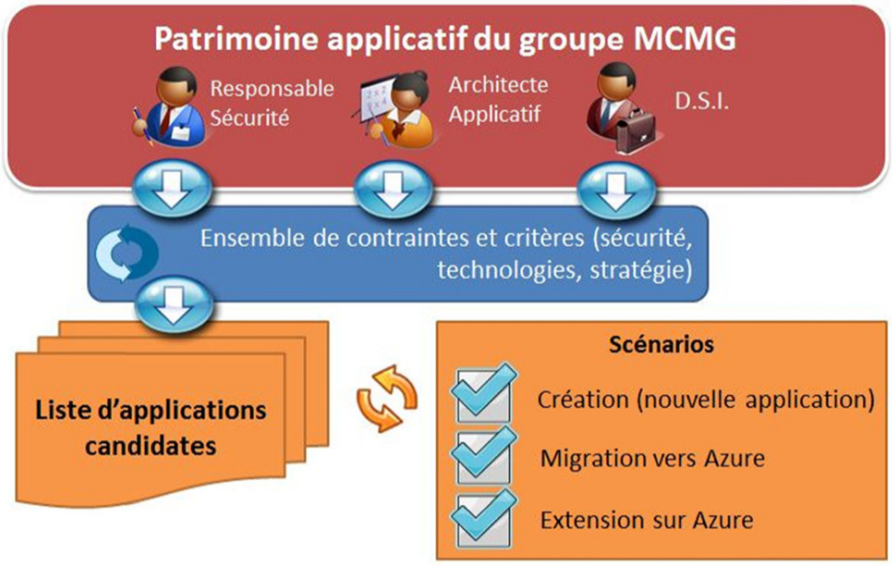


Enfin un troisième et dernier scénario de **mise à disposition de ressources ponctuelles** lors des grandes campagnes marketing par exemple



3. Les outils qui permettent d'y aller

Nouveau services en lignes, intégration des aspects communautaires… pour sécuriser et pérenniser ses choix, le groupe MCMG, décide de faire appel à un partenaire possédant un niveau d'expertise reconnu dans ces domaines. Le groupe possédant des équipes de développement compétentes sur la plateforme Microsoft .Net, la cible Windows Azure est rapidement identifié comme future plateforme pour héberger les applications 2.0 du groupe. L'occasion de mettre en place très rapidement un « Windows Azure Assessement » avec ce partenaire, permettant d'identifier dans le patrimoine applicatif de la société les applications candidates pour la plateforme Azure :

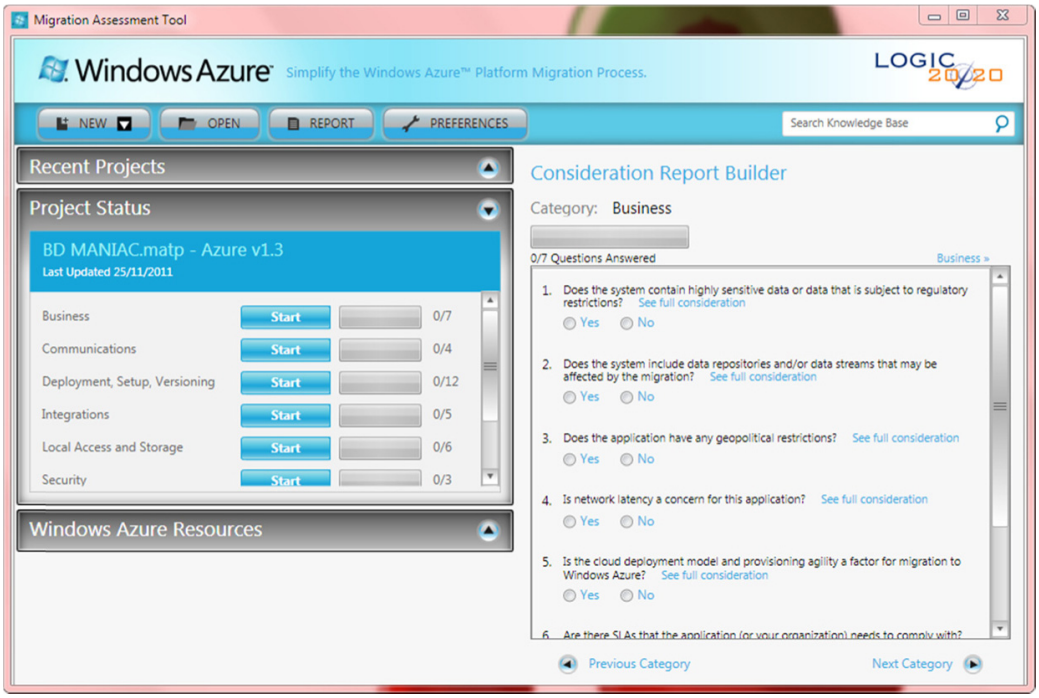


Déroulement d'un « Windows Azure Assessment »

Cet assessment constitue une étape importante car il permet d'impliquer les acteurs indispensables à la réussite d'un projet Windows Azure :

- La **D.S.I**. pour assurer la cohérence globale, alignée sur la stratégie du groupe.
- Un **responsable de la sécurité** applicative, afin de prendre en compte les contraintes de la société mais aussi les prérequis pour se lancer sur une plateforme PaaS.
- Un **architecte technique** maîtrisant bien le portfolio applicatif, à même de préciser et valider les éléments techniques mis en œuvre dans les solutions.

Dans le cadre de notre client MCMG, il est déjà acté que le choix portera sur la migration d'une application existante, la société n'ayant pas dans les mois à venir, l'intention de lancer de nouveaux produits sur le marché. Cette réunion, d'une ½ journée ou d'une journée se déroule dans un contexte outillé. Deux outils peuvent être utilisés lors de cette journée de travail. Le premier des deux est le MAT - Migration Assessment Tool http://matclickonce.blob.core.windows.net/app/publish.htm - proposé par Microsoft et disponible en ligne. Il cible spécifiquement AZURE.

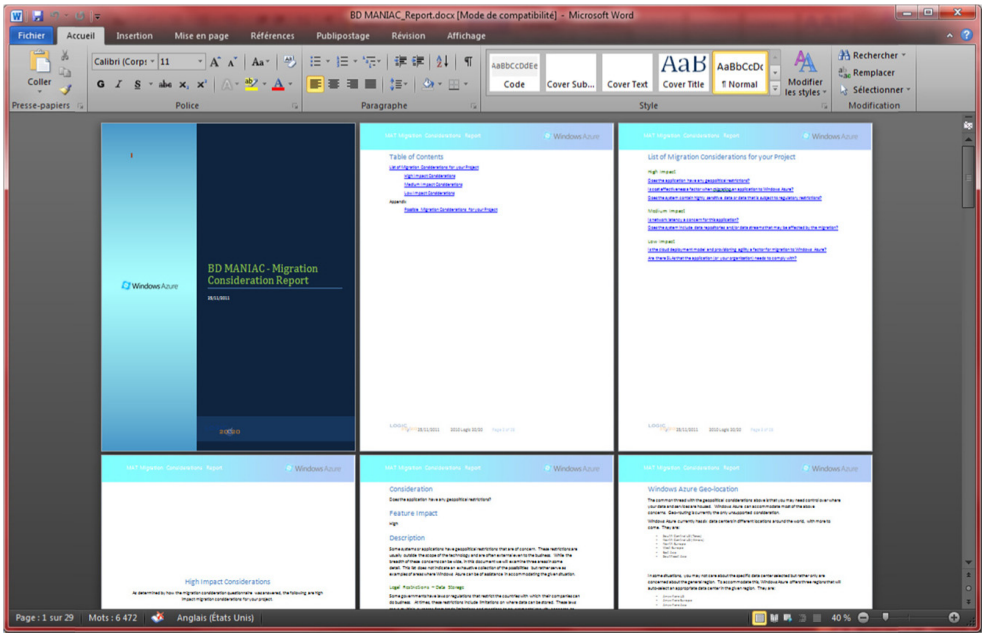


MAT – Migration Assessment Tool en action

Il permet de structurer et guider les échanges autour de 8 thèmes :

1. Métier
2. Communication
3. Installation, déploiement et gestion des versions
4. Intégration
5. Stockage et accès local aux ressources
6. Sécurité
7. SQL
8. Spécificité lié au Web

Chaque section est constituée d'une série de questions, et au final, une fois l'ensemble des questions traitées, un document de synthèse est produit. Il peut servir de socle pour rédiger un document de recommandations finales.



MAT – Migration Assessment Tool en action – document de restitution finale

L'avantage principal d'un tel outil est de passer en revue toutes les questions pertinentes avant de se lancer dans le choix des applications à migrer vers Windows Azure.

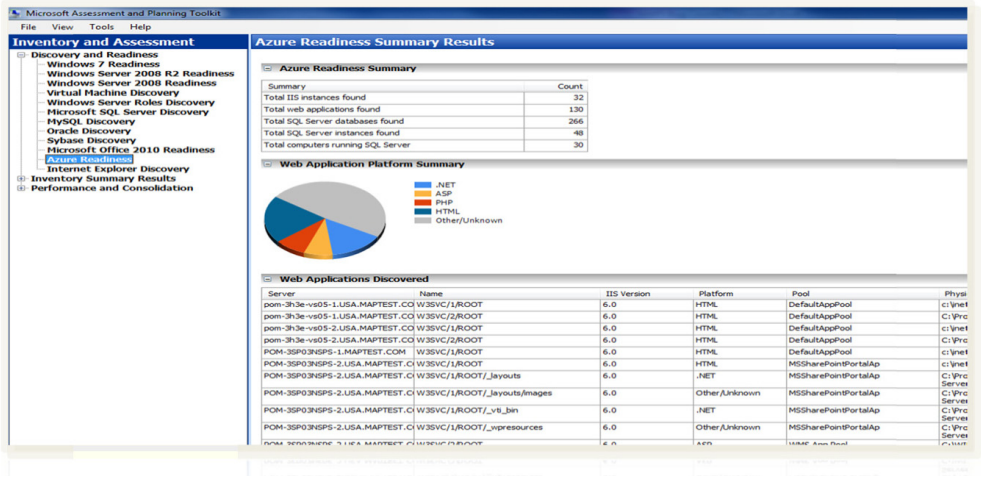
Pour MCMG, cette étape à permit de sélectionner plusieurs applications potentiellement aptes à migrer vers Windows Azure.

Le deuxième outil est le MAP – Migration Assessment & Planning Tool http://technet.microsoft.com/en-us/solutionaccelerators/dd537566 – proposé par Microsoft et disponible en ligne lui aussi. Il cible de façon générale les processus de migrations applicatives vers de solution Microsoft, et la version 6.0 inclus de nouvelles fonctionnalités pour simplifier la migration vers le Cloud (et donc Windows Azure).

Pour les projets de types PaaS : MAP 6 propose une évaluation des applications et de leur capacité à être hébergées sur la plate-forme Windows Azure, l'analyse inclut :

- Un inventaire des applications.
- Une estimation des besoins en termes de capacité au niveau de la plate-forme Windows Azure pour chaque application.

- Une évaluation de la difficulté de migration de chaque application.
- Un classement des applications en fonction de leur adéquation à la migration.
- Une analyse du retour sur investissement (TCO/ROI) de la migration de chaque application.



MAP – Migration Assessment & Planning Tool en action

Cette étape franchie, plusieurs applications sont retenues par le groupe comme solutions candidates à la migration vers Windows Azure. C'est au final pour son département bandes dessinées, **BD-MANIAC**, que le groupe MCMG décide de se porter son choix. **BD-MANIAC** édite déjà un logiciel destiné aux collectionneurs amateurs, **BDTHEQUE**, leur permettant de gérer leur collection au travers d'une solution « on-premise ». Le produit actuel devenant obsolète techniquement, BD-MANIAC souhaite changer de modèle en réalisant une solution innovante, adaptée aux besoins de ses utilisateurs et intégrant des fonctionnalités collaboratives avancées.

Il s'agit d'une migration / réécriture de la solution actuelle, basée sur un client riche Windows et une base de données locale. Le produit actuel dont le succès ne se dément pas souffre cependant d'une érosion de sa croissance, des produits plus novateurs faisant leur apparition chez des concurrents. La solution actuelle sera donc portée sur Windows Azure, et deviendra à la fois une solution disponible intégralement en ligne, mais aussi à terme via un client riche Windows consommant les nouveaux services Windows Azure disponibles en ligne.

CRITÈRES FINANCIERS

Dernier point important avant de se lancer dans l'aventure : mesurer et estimer les coûts de la solution Windows Azure par rapport à un modèle applicatif traditionnel (hébergement sur site ou chez un prestataire).

Là encore Microsoft permet d'estimer ces éléments à l'aide d'un outil, le **TCO et ROI Analyser Tool** http://www.microsoft.com/windowsazure/tools/#tcoCompare-LB, il permet de :

- Déterminer la valeur "quantitative" de Windows Azure, en jouant sur les nombreux paramètres et options proposés
- Estimer le TCO du développement et des opérations sur Windows Azure comparé à un hébergement traditionnel (incluant les coûts des amortissements du matériel et des licences et les coûts humains)
- Estimer le coût de fonctionnement sur Windows Azure

Questionnaire | Windows Azure Services | Windows Azure Costs | ROI Comparison

Windows Azure Instances

Number of Windows Azure instances:*	3
Average use hours per day:*	24.0
Average use days per year:*	365

Translate on-premise usage needs into Windows Azure instances

Proposed Windows Azure Platform for jim

*Based on the information you provided, and including growth (see details), the Windows Azure Platform price and ownership costs would be as follows, analyzed over the selected Three Year Analysis Period:***

		Usage**		Rate***		Initial / Month	Initial / Year
Windows Azure	Computing	2,190	hrs / month	$0.12	/ hour	$262.80	$3,154
	Storage (unstructured) (GB)	100.0	GB	$0.15	/ GB / month	$15.00	$180
	Storage transactions (tx)	1,000	10K tx / month	$0.01	/10K tx	$10.00	$120
SQL Azure		Usage**		Rate ***		Initial / Month	Initial / Year
	1GB databases	1	units	$9.99	/ month	$9.99	$120
	10GB databases	0	units	$99.99	/ month	$0.00	$0
.NET Services		Usage**		Rate***		Initial / Month	Initial / Year
	Service Bus	0	100K msgs / month	$0.15	/ 100K msgs	$0.00	$0
	Access control tokens	50	100K msgs / month	$0.15	/ 100K msgs	$7.50	$90
Bandwidth		Usage**		Rate***		Initial / Month	Initial / Year
	Inbound *	2.50	avg. GB / hr	$0.10	/ GB hour	$182.50	$2,190
	Outbound *	5.00	avg. GB / hr	$0.15	/ GB hour	$547.50	$6,570
Total						Initial / Month	Initial / Year
						$1,035.29	$12,424

** Average usage, counts, rates and cost per month over analysis period. Initial defaults based on Profile selections.

*** This is not a pricing proposal, but an estimate for comparison. Please contact Microsoft or a certified Microsoft Partner to obtain an official pricing quote. All pricing subject to change.

Windows Azure TCO & ROI Analyser Tool

Une fois ces étapes passées, le département BD-MANIAC possède toutes les informations utiles pour prendre la décision de partir sur ce projet.

4. L'application BDTHEQUE

Le choix étant maintenant arrêté, le département BD-MANIAC finalise son choix en divisant son application web en trois parties principales :

- Une partie permettant aux utilisateurs de créer, consulter et gérer sa propre collection. A terme cette fonctionnalité sera disponible via un client riche, puis un client mobile.
- Une partie permettant aux utilisateurs de rechercher une bande dessinée à partir d'une base de données. Cette base de données sera alimentée par les utilisateurs du site, sur un mode communautaire participatif, et d'ajouter ainsi une nouvelle bande dessinée si celle-ci n'apparait pas dans la base de données actuelle.
- Enfin une troisième partie permettant aux différents administrateurs de générer différents des rapports sur l'utilisation du site.

DESCRIPTION GÉNÉRALE DE L'APPLICATION

L'entreprise veut réaliser un site Internet simple et efficace, composé d'une page d'accueil composée d'une fenêtre d'authentification. L'ensemble des pages sera composé d'un menu permettant à l'utilisateur de naviguer facilement entre les différents éléments du site.

5. Comment ouvrir son compte Windows Azure

Dernière étape avant de se lancer : obtenir des ressources Windows AZURE.

Pour le besoins de notre scénario BDTHEQUE nous allons simplement vous expliquer comment ouvrir votre premier compte Azure.

Prérequis : il vous faut un Windows Live ID. Si ce n'est pas le cas, allez ici : www.live.com

Ensuite deux options :

VOUS N'ÊTES PAS ABONNÉ MSDN

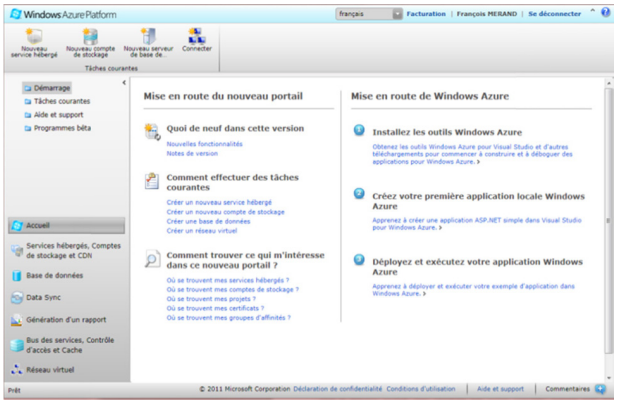
Rendez-vous sur le site : http://www.microsoft.com/france/windows-azure/ c'est votre point d'entrée pour obtenir un compte Azure.

Cliquez sur le bouton suivant au milieu de la page :

L'offre actuelle (12/2011) vous permet d'obtenir gratuitement 90 jours de ressources Windows Azure comprenant :

- **Calcul :** 750 heures d'une petite instance de calcul
- **Stockage :** 20 Go avec 50 000 transactions de stockage
- **Transferts de données :** Données sortantes : 20 Go ; données entrantes : illimité
- **Base de données relationnelle :** 1 Go de base de données SQL Azure Web Edition
- **Contrôle d'accès :** 100 000 transactions
- **Bus des services :** 2 connexions
- **Mise en cache AppFabric :** 128 Mo de cache

Suivez pas à pas les instructions, pour vous enregistrer. Une fois la confirmation reçue, vous pourrez vous connecter à l'adresse précédente et cliquer sur le bouton « Se connecter » en haut à droite. L'écran de management d'Azure devrait ressembler à celui-ci :



VOUS ÊTES ABONNÉ MSDN PREMIUM OU MSDN ULTIMATE

Vous disposez alors gratuitement de 750h par mois de Windows Azure, soit l'équivalent d'une instance tournant en permanence !

Attention : *vous devez au préalable avoir un profil enregistré sur Microsoft Online Services pour activer votre Windows Azure. Si ce n'est pas le cas, connectez-vous sur https://mocp.microsoftonline.com/site/default.aspx puis remplissez les informations du profil.*

Services par niveau d'abonnement (sans frais supplémentaire)*		Premium, Ultimate & BizSpark
Windows Azure™	Petite instance de calcul	750 heures / mois
	Stockage	10 Go
	Transactions	1 000 000 / mois
AppFabric	Connexions de bus des services	5 / mois
	Transactions de contrôle d'accès	1 000 000 /mois
SQL Azure™	Bases de données Web Edition (1 Go)	750 heures / mois
Transferts de données	Europe et Amérique du Nord	7 Go en entrée / mois 14 Go en sortie / mois
	Asie Pacifique	2,5 Go en entrée / mois 5 Go en sortie / mois

Windows Azure pour les abonnées MSDN

Créer votre compte Windows Azure en 8 étapes :

- **Etape 1 :** Connectez-vous à votre compte MSDN. Allez directement sur la page « Mon Compte » et cliquez sur « Plateforme Windows Azure ».
- **Etape 2 :** Accédez au portail Microsoft Online Services : Vous serez automatiquement redirigé vers le portail Microsoft Online Services. Complétez les informations demandées puis cliquez sur« Suivant ».
- **Etape 3 :** Validez votre panier : Il contient « Plateforme Windows Azure MSDN Premium » en formule tarifaire. Cochez la case « En activant cette case à cocher, le client reconnaît… », puis « Valider ».
- **Etape 4 :** Choisissez votre mode de paiement : Cette démarche est obligatoire pour activer son compte Windows Azure. Vous ne paierez que si vous dépassez le quota Windows Azure inclus par défaut avec votre abonnement MSDN.

- **Etape 5 :** Activez votre contrat d'abonnement : Il est obligatoire de cocher la case « En activant cette case à cocher et en votre nom, vous reconnaissez… ». Tapez votre nom et votre fonction. Puis cliquez sur « J'accepte ».
- **Etape 6 :** Validez et Confirmez la commande
- **Etape 7 :** Validation du service : Vous accédez à la suite de la validation du service :
  - Nom de l'abonnement (ex. : Cloud Azure).
  - Administrateur du service (par défaut ce sera vous).
  - Résumé de la validation.
  - Puis, confirmez la validation du service.
- **Etape 8 :** Une fois votre abonnement Windows Azure actif, dans la section Actions, sélectionnez « refuser renouv. Auto » (renouvellement automatique) et cliquez sur « OK ».

Vous pouvez maintenant accéder à votre portail Windows Azure : http://windows.azure.com

Vous pourrez effectuer le suivi de votre consommation (mise à jour toutes les 24h) :

- Allez sur le portail Microsoft Online.
- Authentifiez-vous avec votre Windows Live ID.
- Cliquez sur « Afficher mes factures » en haut à droite.

Nous reviendrons sur les différentes fonctionnalités offertes au fur et à mesure de leurs utilisations par notre application.

Dans le prochain module, nous passerons aux choses concrètes, puisque nous nous intéresserons à la mise en place de l'environnement de développement (Visual Studio 2010, Azure SDK, etc.) au développement de l'application elle-même et à son déploiement sur Azure.

Développement en local et déploiement sur Windows Azure

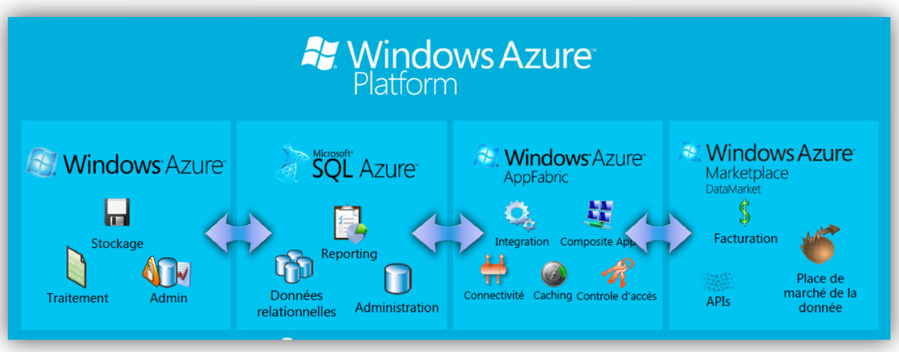
Le périmètre initial étant maintenant bien défini, l'équipe projet BD-MANIAC désignée pour réaliser l'application BD-THEQUE se met en place. Elle va commencer par installer la

plateforme de développement puis créer et configurer son premier projet Windows Azure. Ce projet sera ensuite immédiatement déployé sur la plateforme Windows Azure, afin de s'assurer que le cycle de production et de mise en exploitation de la solution est opérationnel de bout en bout.

L'objectif de cette seconde partie de ce livre blanc est de décrire les différentes étapes pour créer, configurer un projet de type Cloud sous Visual Studio et d'effectuer un premier déploiement sur Windows Azure.

Vous découvrirez à partir de cet exemple, expliqué en mode « pas à pas », que le déploiement du projet en lui-même est extrêmement simple **et ne nécessite aucune ligne de code.**

Avant de créer ce premier projet, prenons quelques lignes pour préciser ce qu'est Windows Azure ; simplement pour avoir un rappel synthétique :



La plateforme Windows Azure et ses principales composantes

La plateforme Windows Azure permet d'héberger, exécuter et gérer des applications et des données dans le ou les centre de données Microsoft de votre choix (dont 2 en Europe). BD-MANIAC peut utiliser les services suivants pour créer ses applications, hébergées sur des machines virtuelles :

- De la puissance de calcul via les notions de **« Worker Roles »** et **« Web Roles »**, mais aussi de **« VM Roles »** (actuellement en CTP)
- Différents types de stockage et de la base de données relationnelle (SQL Azure)
- Du réseau de diffusion de contenu (CDN)

- Du service de cache
- Du contrôle d'accès
- Un bus de service
- Du réseau virtuel (actuellement en CTP)
- Du décisionnel à venir (Actuellement la partie « reporting services » est disponible en CTP).
- Et une place de marché, Application Market pour les applications et Data Market pour les données

Pour des informations plus complètes, le meilleur point d'entrée est ici : http://windowsazure.fr

1. Création du projet et configuration

Pour commencer, l'équipe projet s'intéresse à son environnement de développement et souhaite découvrir comment créer et configurer un projet de type « Cloud », en mode local.

PRÉREQUIS – ENVIRONNEMENT DE DÉVELOPPEMENT

La première étape consiste à installer et configurer le socle applicatif qui permettra à l'équipe de développer sereinement la solution Windows Azure. Plusieurs logiciels sont nécessaires pour travailler avec Windows Azure, afin de développer l'application en local :

- Une version de Visual Studio 2010, avec le SP1 installé.
- Windows Azure SDK, le kit de développement pour .NET
- SQL Server 2008 R2 Express, l'émulateur Windows Azure ayant besoin de SQL Server 2008 R2 Express

L'option la plus simple consiste à laisser le kit d'installation de Visual Studio faire le nécessaire de façon automatique. Concernant le choix de la version de Visual studio, plusieurs options sont offertes à l'équipe de développement BD-THEQUE.

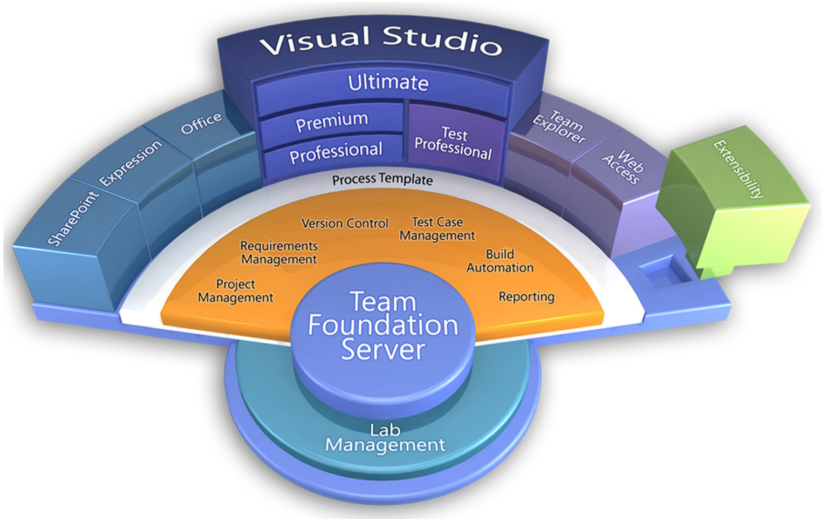
- **Visual Studio Web Developer 2010**, version gratuite et allégée de la plateforme de développement de Microsoft, disponible ici : http://msdn.microsoft.com/fr-fr/express/aa975050 . Cette version est suffisante pour tester l'outil et Windows Azure, mais n'est pas adaptée aux développements professionnels en entreprise.
- **Visual Studio 2010**, en version PRO, PREMIUM ou ULTIMATE. Vous trouverez toutes les informations utiles sur les fonctionnalités de ces différentes versions sur cette page : http://www.microsoft.com/france/visual-studio/default.aspx
- L'utilisation optionnelle de **Team Foundation Server**, outil indispensable pour mettre en place une filière de développement .Net industrialisée et performante. Vous trouverez toutes les informations utiles ici : http://msdn.microsoft.com/fr-fr/vstudio/ff637362
- Enfin l'équipe décide d'installer et d'utiliser un SQL Server 2008 R2 Express

Au final, l'équipe projet BD-MANIAC sélectionne son environnement de développement, composé de :

- Visual Studio 2010 ULTIMATE avec SQL Server 2008 R2 Express
- Elle décide aussi de connecter ses postes de développement à un serveur TFS – Team Foundation Server.

MCMG étant éditeur de logiciels, les processus de développement sont bien maîtrisés par l'ensemble des équipes, et les fondamentaux de l'ALM (Application Lifecycle Management) sont bien ancrés dans les usages au sein du groupe : **des outils** performants, au service d'**équipes** alignées sur des **processus** de développement adaptés au contexte de chaque projet. Ce triptyque invariant pour MCMG, sera mis à l'épreuve pour ce projet Cloud / Windows Azure:

- Des équipes focalisées sur le pilotage et la transparence des projets (responsabilisations à tous les niveaux, mesure et amélioration de la qualité en continu en favorisant l'excellence)
- Des outils performants, fournissant tous les services pour produire du logiciel de qualité optimale (outils intégrés, extensibles, productifs et ayant une courbe d'apprentissage réduite)
- Des processus agiles, orientés collaboration (accompagnement méthodologique, personnalisable et non intrusif)



L'environnement de développement complet de l'équipe BD-MANIAC

Du côté des prérequis techniques, il est fort utile de connaître les bases de la programmation en C#. En complément, des notions ASP.Net seront les bienvenues.

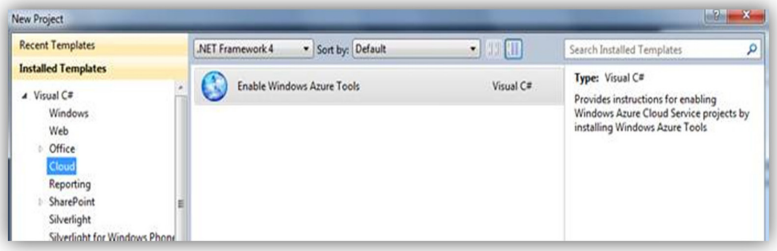
Enfin, dernière étape avant de se lancer, il est nécessaire d'installer le **« Windows Azure SDK »**. Ce kit vient compléter utilement les Visual Studio 2010 de l'équipe, en ajoutant tous les outils et compléments qui permettrons de créer, développer et configurer simplement les solutions Windows Azure. C'est aussi l'installation de ce kit qui permettra de développer et tester en local (sans connexion réel à Windows Azure) la solution.

Vous pouvez télécharger ce SDK ici : http://msdn.microsoft.com/fr-fr/windowsazure. L'installation vérifie ce qui est présent ou non sur votre machine et installe donc ce qui

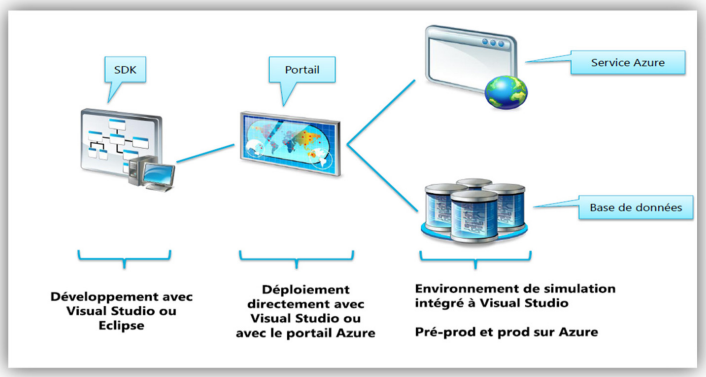
manque, au bon endroit et avec les bonnes versions. Le SDK dans sa version 1.6 intègre :

- Des modèles de programmation Windows Azure
- Les APIs pour lancer les machines virtuelles
- Un environnement de simulation Azure en local
- Les outils de packaging et de déploiement automatique
- Des exemples d'applications et un kit de formation disponible en complément dans le Windows Azure Platform Training Kit (http://www.microsoft.com/download/en/details.aspx?id=8396)

ASTUCE : Il existe une autre façon simple pour installer ce kit dans sa dernière version : il vous suffit d'ouvrir votre Visual Studio, de passer par le menu **« créer un nouveau projet »** de type **« Cloud »**. Si le SDK n'est pas installé, Visual Studio vous propose de le faire immédiatement, en téléchargeant et configurant la dernière version disponible en ligne. Cette installation est prise en charge par le « Web Platform Installer ». Il est important de noter que ce même Web Platform Installer installera Visual Studio Express si vous n'avez pas Visual Studio Ultimate installé sur votre poste.



Installation du SDK Azure automatiquement depuis Visual studio



Modèle de développement et de déploiement avec Windows Azure

CRÉATION DU PROJET SOUS VISUAL STUDIO

La création du projet est très simple, après avoir installé Visual Studio et Windows Azure Tools, lancez Visual Studio en **mode administrateur**. C'est important, car si vous ne lancez pas Visual Studio en mode administrateur, l'émulateur Windows Azure (qui vous permet de tester en local vos applications) ne se lancera pas.

L'équipe se lance donc pour créer son premier projet Windows Azure !

Une fois dans Visual Studio, sur la page d'accueil cliquez sur « New Project » ou allez dans **« File »** à **« New »** à **« Project »**

L'équipe peut donc ensuite créer son premier projet de type Cloud : **bdtheque.azure.**

L'étape suivante consiste pour l'équipe à ajouter des **rôles** à son projet. Windows Azure dispose de machines virtuelles appelées **« rôles »** pour exécuter les composants applicatifs. La notion de rôle est nouvelle et propre aux projets de type Windows Azure. Pour tirer parti des bénéfices d'Azure en termes de disponibilité et de montée en charge, une application doit suivre les règles suivantes :

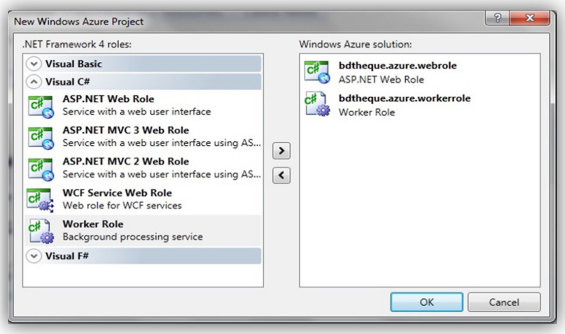
- Une application Windows Azure est construite sur un ou plusieurs rôles
- Chaque rôle doit être déployé au minimum sur deux instances
- Une application Azure doit se comporter correctement si une instance tombe. Il est important de conserver un mode dit « stateless » : les applications Windows Azure ne doivent pas gérer d'état pour fonctionner de façon optimale.

<u>L'application BD-THEQUE va implémenter :</u>

- Un **Web Role** Basé sur IIS, qui exécute les applications Web et utilise ASP.NET, WCF, PHP, etc.
- Un **Worker Role** utilisé pour les traitements en arrière-plan

Il est important de noter que Silverlight fonctionne sur Windows Azure.

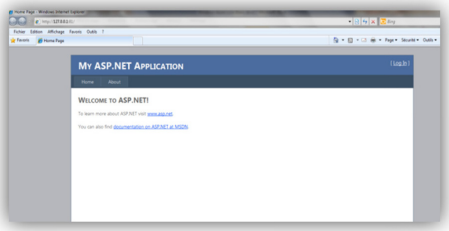
L'équipe ajoute donc un **« ASP.Net Web Role »** qu'elle renomme **bdtheque.azure.webrole** et un **« Worker Role »** qu'elle renomme **bdtheque.azure.workerrole**. Ce dernier servira pour le traitement des images dans la solution.



Une fois ces deux étapes passées, le projet est créé, et déjà prêt à être déployé. Un projet Windows Azure contient en fait plusieurs projets accessibles depuis l'explorateur de projets :

- Un projet global décrivant les rôles de l'application et leurs configurations.
- Autant de projets que de rôles créés.

Ensuite, le développement du site se fait de manière classique, sauf sur certains points (par exemple s'il on a besoin d'uploader des fichiers dans le Cloud, il faudra faire appel au stockage Azure). Il est tout à fait possible, grâce à l'émulateur Windows Azure, de lancer notre application en local et la débugger en appuyant sur F5. L'équipe effectue un premier test et obtient le résultat suivant :



Pourquoi cet émulateur ? Pour plusieurs raisons :

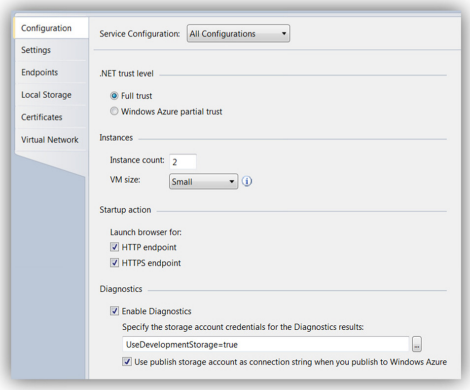
- Grâce à celui-ci il est possible d'exécuter notre projet en local
- Il simule le stockage Azure, cela permet donc de tester et débugger le code utilisant celui-ci. Il utilise SQL Server installé en local pour simuler SQL Azure.
- Il permet donc de tester notre application sans passer à chaque fois par un déploiement physique

Il est important de souligner que le déploiement dure un petit moment, donc le fait d'utiliser l'émulateur permet un gain de temps non négligeable.

Cette icône apparaît lors de l'exécution en local

Avant de déployer cette première application vide, il est important de s'arrêter sur la configuration de nos différents rôles. Le Worker Role et le Web Role ont à peu de choses près la même configuration. Pour voir la configuration de notre Web Role, il faut cliquer droit sur bdtheque.azure.webrole dans le dossier Roles de notre projet Cloud et sélectionner Properties.

La configuration se divise en plusieurs parties.



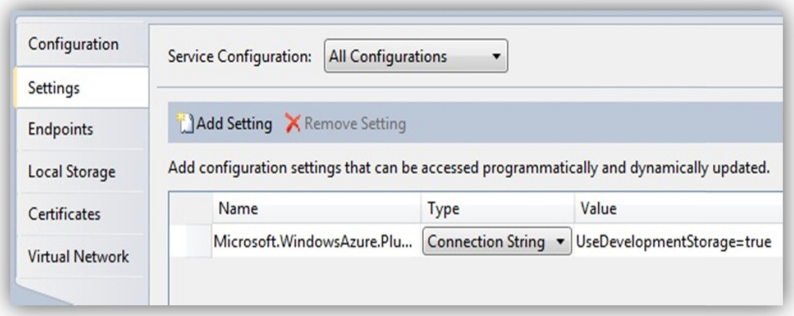
Configuration : La première concerne la configuration générale du Web Role. C'est ici que l'on va définir le nombre d'instances que l'on va utiliser. Ce paramètre est très important, car c'est lui qui va contrôler la notion de « Load Balancing » et de haute disponibilité de la future solution BD-THEQUE. Par exemple si l'équipe positionne ce paramètre à deux, une copie de l'application sera déployée sur deux instances différentes (deux machines virtuelles). Si un des serveurs rencontre des problèmes, la deuxième instance prend la relève. Il est donc conseillé de paramétrer plusieurs instances et non une seule. Microsoft ne garantit d'ailleurs un SLA élevé que si deux instances minimum sont mises en production.

Un autre paramètre important est la taille de la machine virtuelle, configurée selon la taille de l'application. Les informations relatives aux tailles des machine virtuelles sont disponibles ici : http://www.microsoft.com/france/windows-azure/offres/tarification.aspx . L'équipe

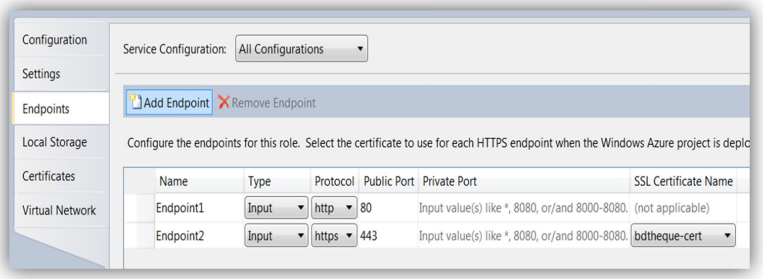
choisit de conserver dans un premier temps des machines de type « small ».

On peut également choisir de lancer l'application en http ou https (si des certificats SSL sont présents sur le portail Azure). L'équipe BD-THEQUE utilisera ces fonctionnalités ; elles seront expliquées dans le prochain chapitre.

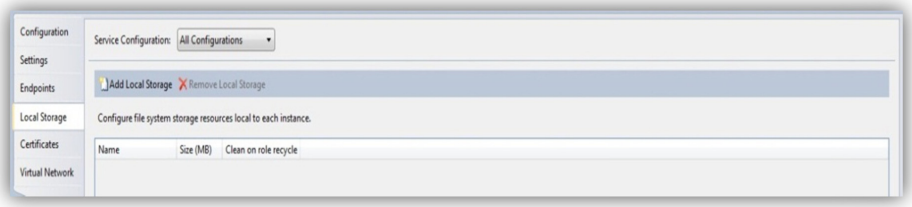
Settings : Cette deuxième partie permet de stocker différents paramètres de configuration (chaînes de connexion, emplacement du serveur de stockage Azure, Diagnostics …). C'est une IHM qui agit sur le contenu XML des fichiers **ServiceConfiguration** et **ServiceDefinition**



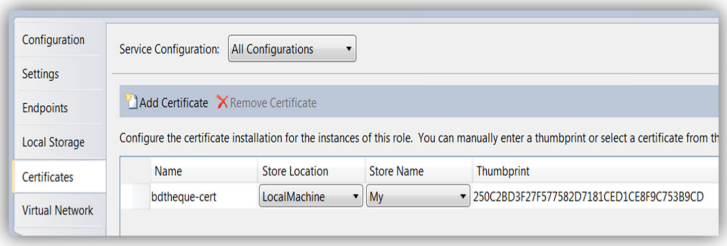
La partie «EndPoints », permet de définir les points de communication avec le navigateur (http, https, ports, …). L'équipe définit les EndPoints suivants pour tests :



Local Storage : Cette partie permet déclarer sur le File System local de l'instance en production sur Windows Azure des espaces de stockage. (Pour créer un système de cache par exemple). En l'état actuel du projet, l'équipe ne renseigne rien pour l'instant.



Certificates : Cette partie permet de rajouter des certificats SSL, qu'il faut avoir installé auparavant sur le serveur via le portail. Tout cela sera expliqué dans le prochain chapitre.



Virtual Network : Cette dernière partie permet de gérer l'utilisation d'Azure Connect, qui autorise votre application à se connecter sur un serveur interne à l'entreprise par TCP (SQL Server par exemple). Cette fonctionnalité n'est pas utilisée pour le projet BD-THEQUE.

Maintenant que l'équipe à fait le tour de la configuration du projet, elle peut effectuer son premier déploiement sur Windows Azure.

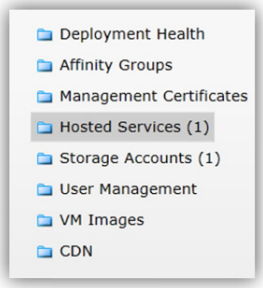
2. Déploiement sur Windows Azure

L'équipe BD-MANIAC va le découvrir : le déploiement sur Windows Azure, est très simple. C'est d'ailleurs une des forces de l'offre, il peut se faire en seulement quelques clics et aucun code n'est nécessaire. Il peut être effectué de deux manières différentes :

- Via le portail Windows Azure (généralement utilisé lors du premier déploiement)
- Via Visual Studio 2010, le processus de déploiement est complètement intégré à l'outil de développement. Il peut aussi être pris en charge dans le processus d'intégration de Team Foundation Server.

L'équipe BD-MANIAC décide d'expérimenter les deux options.

DÉPLOIEMENT VIA LE PORTAIL WINDOWS AZURE

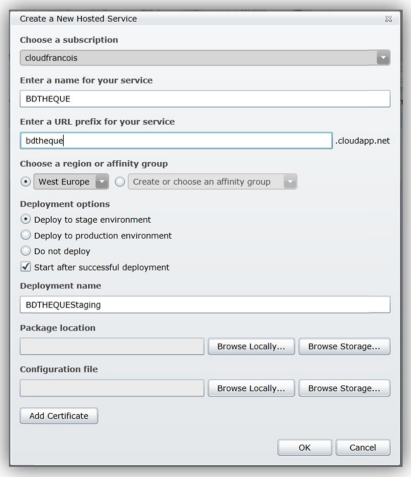


Cette méthode, très simple, est la moins rapide. Elle est utilisée généralement lors du premier déploiement de l'application sur Windows Azure, en même temps que la création de notre service hébergé sur le portail.

L'équipe BD-MANIAC se connecte avec son profil sur le portail Azure ; il faut aller dans la partie Services Hébergés, et créer un nouveau service hébergé. Sur la partie gauche de l'écran sont proposés dans un menu les différents services et actions possibles :

Pour créer le premier service, il faut cliquer tout simplement sur l'icône **« New Hosted Service »**.

Qui donne accès à l'écran de configuration ci-dessous :



Création et configuration initiale du site BD-THEQUE sur le portail Windows Azure

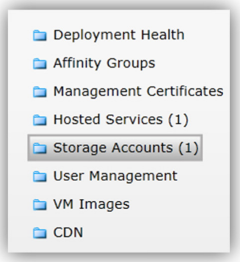
Lors de la création du service hébergé, on dispose de deux environnements :

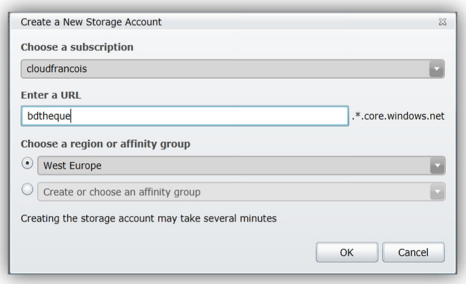
- Un environnement dit de « staging » afin d'effectuer les différents tests avant de passer en production finale. L'accès se fait via une URL généré automatiquement par le portail)
- Un environnement de production qui correspond à l'hébergement final, accessible via l'URL définie par l'équipe projet.

La mise en production consiste à permuter la version dite de « staging » avec la version en production. L'avantage de cette méthode est de pouvoir faire marche arrière : en cas de problème fonctionnel par exemple, il suffit de basculer à nouveau pour remettre en production la version précédente.

L'équipe conserve donc le nom **bdtheque** pour son service, qui sera donc accessible via l'URL **bdtheque.cloudapp.net** tant que les redirections DNS vers BDMANIAC.COM n'auront pas été installées. Pour ce premier déploiement c'est l'option **« stage environment »** qui est sélectionnée ; il s'agit bien de tester la solution, pas de la mettre en production. La solution sera déployée sur les datacenters Européens de la plateforme Windows Azure.

Ce point étant traité, il faut maintenant s'intéresser à la partie stockage des données. Après avoir cliqué sur **« Storage Accounts »**, puis l'icône **« New Storage Account »**, l'équipe configure très simplement un compte bdtheque dédié aux données de la future solution.

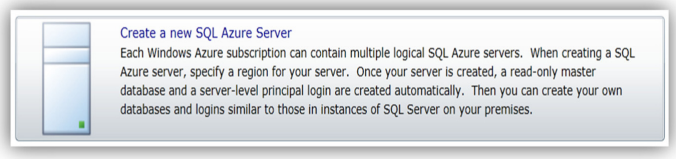




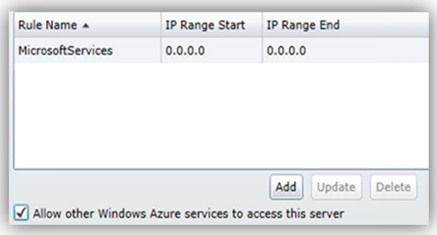
Une fois cette étape franchie, nous avons donc deux comptes bdtheque actifs : un pouvant héberger sur plusieurs machines virtuelles nos services applicatifs, et au second pouvant héberger nos machines virtuelles dédiées aux données.

En restant sur l'écran « Storage Account », l'équipe en profite pour configurer ses espaces de stockage, en commençant par la base de données SQL Server. Deux étapes simples pour

cela : il faut commencer par créer un serveur SQL Azure en cliquant sur la zone ci-dessous :

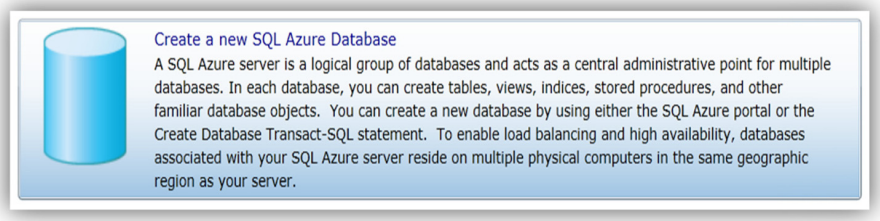


Précisez la région que vous voulez utiliser, et suivez ensuite les instructions. Préciser ensuite un **login** et un **mot de passe** qui vous servirons ensuite de compte administrateur pour cette machine SQL Server. Avant de continuer, il est important de créer des plages d'adresses IP autorisées à se connecter à ce serveur ; par défaut, même Windows Azure ne possède pas ces autorisations ! Pour le faire, commencez par ajouter une règle simple en ajoutant l'adresse 0.0.0.0 et en cochant la checkbox :

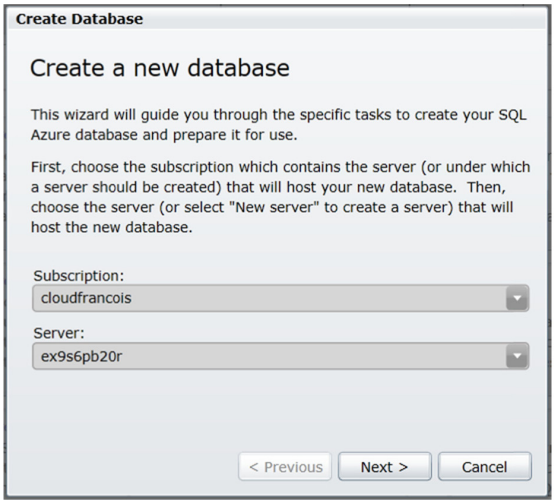


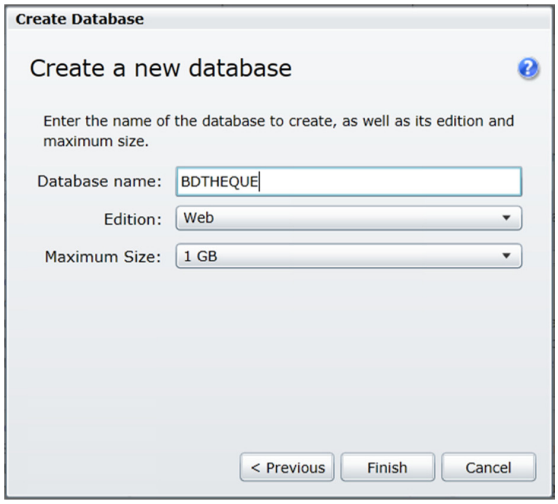
Procédez de même pour ajouter les adresses IP des postes ou machines qui devront accéder à ce serveur SQL.

Puis une fois ce serveur opérationnel, il reste à créer la base de données en cliquant sur cette zone :



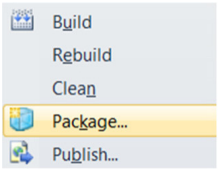
Le processus de création est très simple, il suffit de se laisser guide par les écrans :

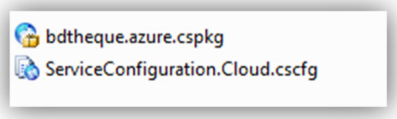




La taille des bases de données SQL Azure proposée est de 1GB ou 5GB jusqu'à 150GB max actuellement. Cette limitation n'est que théorique, les services annoncés de fédération de bases de données Azure (SQL Azure Federation) n'imposant pas de limite à la taille logique d'une base de données. Les étapes de création et de configuration des ressources Azure sont maintenant terminées, l'équipe va pouvoir tester la mise en ligne de la solution « vide ».

La dernière étape consiste à uploader l'application BD-THEQUE sur la plateforme Windows Azure, ainsi que la configuration des différents rôles. Pour cela, l'équipe projet a besoin de Visual Studio. Il suffit d'effectuer un clic droit sur le projet **bdtheque.azure** et de sélectionner Package.

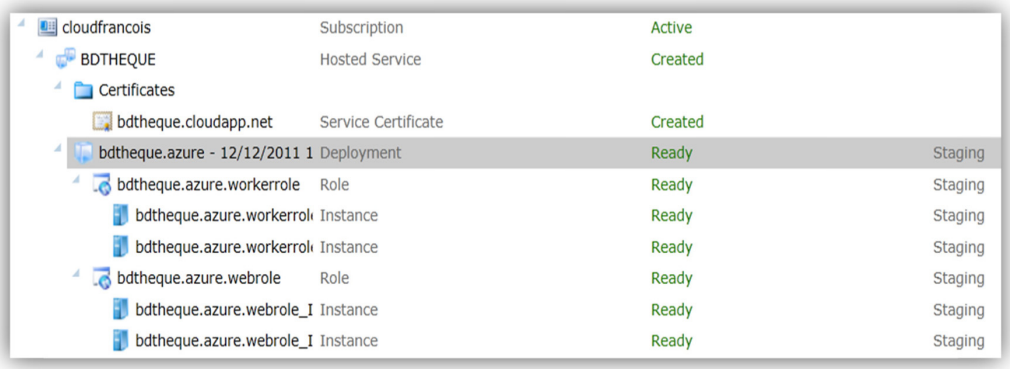




Packages à télécharger sur Windows Azure, produits par la compilation du projet BD-THEQUE

Les deux fichiers demandés par le portail sont générés automatiquement. Il ne reste plus qu'à copier le chemin des fichiers, les coller sur le portail et valider la création du service hébergé. Au bout de 10 à 15 minutes maximum, l'application BD-THEQUE sera disponible

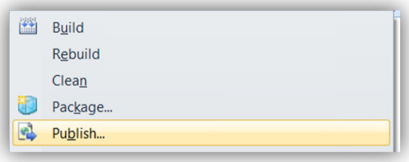
sur le cloud via l'URL de développement ou de production (selon votre choix précédemment effectué).



L'application BD-THEQUE est en production ! Avec un Web Role (2 instances) et un Worker Role (2 instances)

DÉPLOIEMENT VIA VISUAL STUDIO

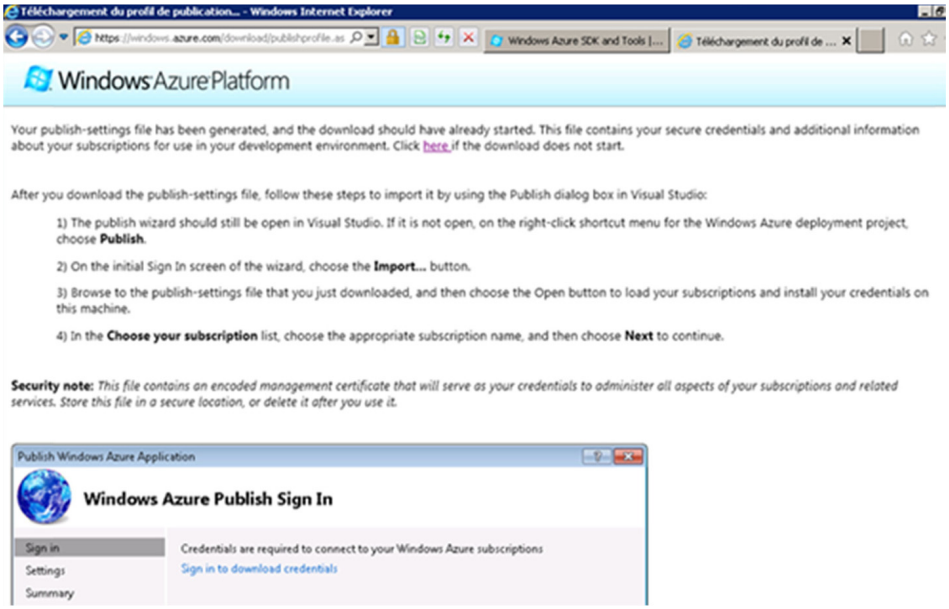
Il existe une deuxième méthode pour déployer une application sur Windows Azure. Via Visual Studio : au lieu de régénérer les packages de l'application et de les uploader sur le portail, quelques manipulations simples vont permettre à l'équipe BD-MANIAC de le faire depuis leur Visual Studio (nous verrons ultérieurement comment automatiser cette tâche à l'aide de TFS). L'équipe projet sélectionne le projet racine, puis clique sur le bouton droit pour sélectionner **« publish »**



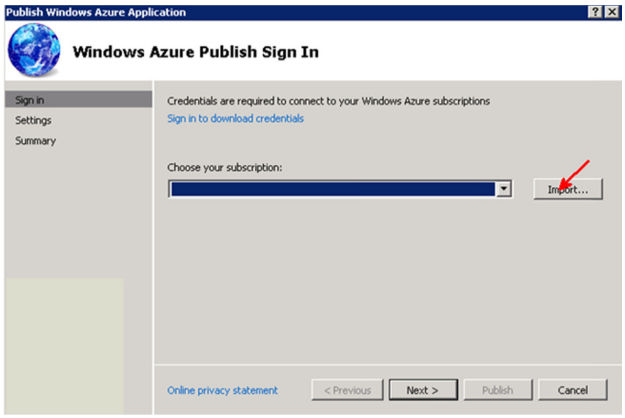
Pour le premier déploiement, une configuration initiale doit être effectuée. Il faut en effet que l'on s'authentifie sur le portail via Visual Studio. Ceci se fait via une authentification par certificats, ce qui est bien compréhensible. Avec le SDK 1.6 installé, il suffit de :

- Démarrer un déploiement de façon à charger un certificat qui est généré à la volée côté Windows Azure (dans la limite disponible):

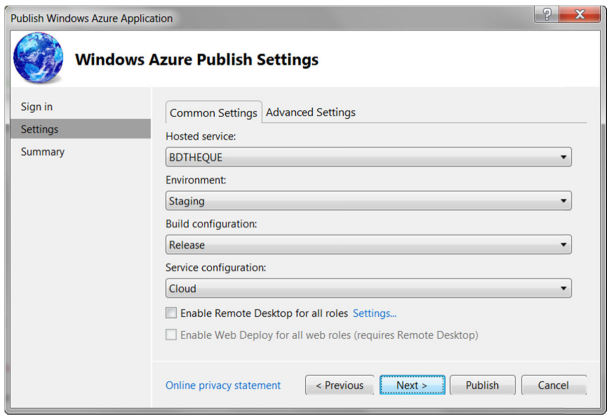
- Se connecter avec un compte Live ID qui a accès à un abonnement Windows Azure en cours de validité. Si vous n'en n'avez pas, vous pouvez aller à http://windowsazure.fr où on vous expliquera comment procéder. Un certificat est généré et installé dans les différents comptes associés au Live ID. Vous pouvez alors télécharger un fichier de type .publishsettings, qui contient ce certificat et la liste des IDs de ces abonnements.



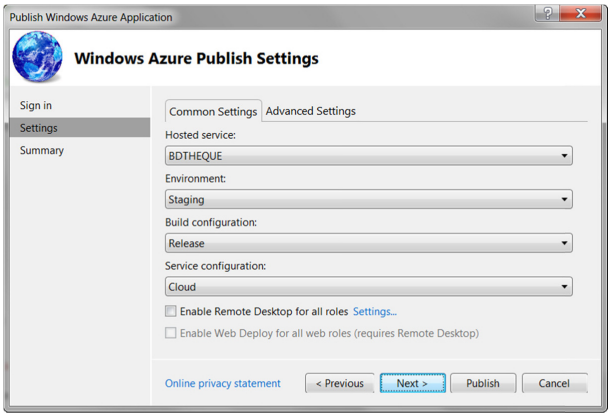
- Retourner à Visual Studio 2010, cliquer sur Import et choisir le fichier qui avait été téléchargé



Suivi des paramètres de mise en staging de l'application BD-THEQUE :



Avant d'obtenir votre accord final :



Architecture et design de la solution BDTHEQUE

L'équipe BDTHEQUE focalise maintenant son attention sur les considérations à prendre en compte en termes d'architecture applicative, du fait de la cible retenue pour sa solution : la plateforme Windows Azure. Elle identifie un certain nombre de points de contrainte pour lesquels elle veut identifier des solutions ou du moins avoir des réponses du point de vue architecture, lui permettant d'entamer les développements de façon sereine.

Les points généralement à considérer pour ce type d'application, sont :

- La gestion des états
- La montée en charge
- La gestion des dépendances
- La confidentialité des données
- Le niveau de SLA requis
- La prise en compte de la latence
- Le problème de réglementation
- L'extensibilité avec de nouvelles fonctionnalités
- La communication avec d'autres systèmes (internes/externes à l'entreprise)
- L'interopérabilité avec des technologies autres que Microsoft

L'expérience d'éditeur de logiciels de la société MCMG permet à l'équipe de s'appuyer sur les fondamentaux qu'elle maitrise :

- Un socle bâti sur une architecture classique « en couches »
- Un découplage entre les couches mais aussi dans l'implémentation des classes et des objets
- La prise en compte du cycle de développement de la solution afin de permettre une plus grande flexibilité dans la gestion des versions et leurs mises à disposition.

1.Considérations propres aux applications Windows Azure

De façon plus opérationnelle, l'équipe aborde la partie DESIGN de sa solution. Elle décide pour cela de partir des contraintes imposées par le modèle Windows Azure, afin d'adapter son DESIGN et donc son architecture applicative à ces contraintes.

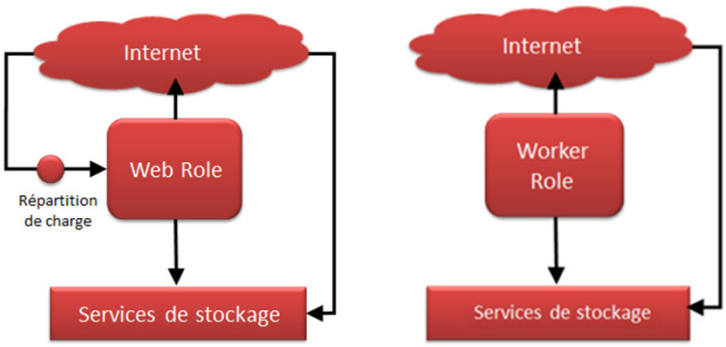
Les points identifiés sont les suivants :

- Une application Windows Azure est destinée à fonctionner sur un ou plusieurs rôles.
- Une application Windows Azure exécute une ou plusieurs instances dans chacun de ces rôles.
- La perte d'une instance ne doit pas être un problème pour une application Windows Azure.
- Les interactions avec le système d'exploitation sont sous « contraintes »
- Les interactions avec les systèmes de stockage, notamment ceux persistants, obéissent aussi à des contraintes spécifiques.
- La communication entre les différents rôles composants la solution suit aussi des principes à prendre en compte.

L' APPLICATION FONCTIONNE SUR UN OU PLUSIEURS RÔLES

Comme cela était expliqué dans le chapitre précédent, Windows Azure dispose de machines virtuelles appelées « rôles » pour exécuter les composants applicatifs. Deux rôles sont retenus pour la solution BDTHEQUE :

- un **« Web Role »**
- un **« Worker Role »**



Quels sont les impacts pour l'équipe BDTHEQUE ?

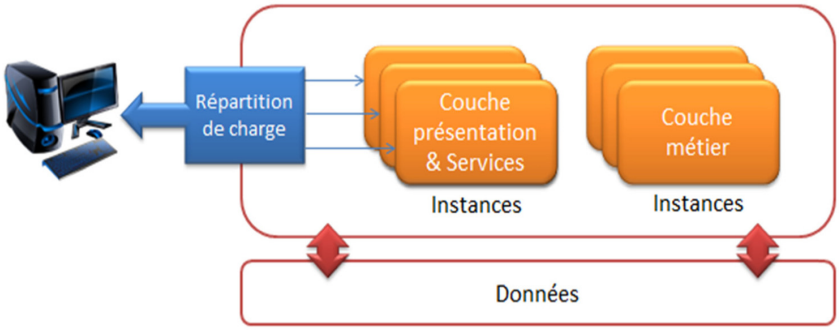
➔ L'application doit être divisée en composants logiques, sur un modèle à couplage faible (sans adhérence), de façon à pouvoir répartir ces composants dans les rôles cibles. Ce découpage doit se retrouver dans la solution Visual studio pour en faciliter le développement et surtout la maintenance.

➔ Les composants applicatifs doivent tenir compte des limitations de l'environnement du rôle cible auquel ils appartiennent. Un Web Role comme un Worker Role impose des cadres de développement et d'exécution auxquels l'équipe doit s'attacher. Le fait que Visual Studio offre des types de projets associés (Web Role et Worker Role) permet facilement à l'équipe d'intégrer ces contraintes en phase de développmenet.

L'APPLICATION EXECUTE UNE OU PLUSIEURS INSTANCES DE CES RÔLES

Les bénéfices attendus par l'équipe de ces multiples instances dans chaque rôle sont :

- Une meilleure distribution de la charge sur la ferme applicative de l'application finale
- Une résistance accrue au blocage d'une VM (cela ne doit pas être un point bloquant, qui doit continuer à fonctionner)
- Une montée en charge (et descente) optimisée et contrôlée (induisant une meilleure maîtrise des coûts)



BDTHEQUE déployée sur plusieurs instances

Les applications sont constituées d'une ou plusieurs instances de chaque rôle. Windows Azure doit lancer au moins deux instances de chaque rôle pour assurer la haute disponibilité requise. En entrée, la charge est répartie requête par requête. D'un point de vue technique, chaque instance exécute sa propre VM, répliquée si nécessaire.

Quels sont les impacts pour l'équipe BDTHEQUE ?

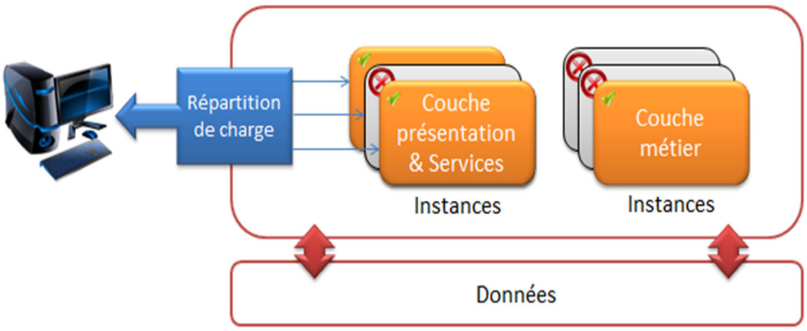
➔ Un seul point important ici : la persistance des contextes applicatifs qui doit être mise en œuvre et implémentée dans des mécanismes de stockage hors machine virtuelle. L'équipe prend en compte ce critère dans son implémentation et dans son architecture applicative.

LA PERTE D'UNE INSTANCE NE DOIT PAS ÊTRE UN PROBLÈME

Cette fois, les bénéfices attendus par l'équipe pour son application résident dans la capacité globale du système à :

- Résister aux pannes du système ou du matériel
- Etre « insensible » aux maintenances systèmes
- Ne pas être impacté par les mises à jour

La plateforme Windows Azure offre en standard de tels services (maintenance et mises à jour) qui s'exécutent sans incidence sur le socle applicatif déployé ni sur l'exécution des rôles en cours, garantissant bien sûr une continuité de service optimale. L'équipe BDTHEQUE prend en compte ce « confort » proposé, mais elle n'en n'oublie pas pour autant d'intégrer dans son architecture une dimension « continuité de service » et d'en tenir compte dans son implémentation.



BDTHEQUE et la perte d'une instance

Si une ou plusieurs instances ne sont plus accessibles, l'application Azure continue de fonctionner correctement.

Quels sont les impacts pour l'équipe BDTHEQUE ?

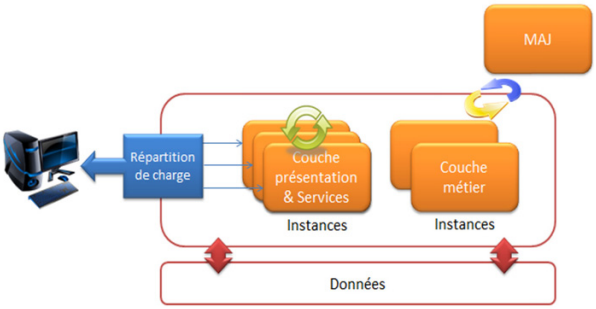
➔ Un seul point important ici aussi, qui concerne la persistance qui doit être externe à la VM: chaque rôle doit être « stateless » (sans gestion d'état) localement et stocker les informations de contexte sur un support externe (soit Windows Azure Storage ou SQL Azure).

LES INTERACTIONS AVEC L'OS SONT SOUS « CONTRAINTES »

L'équipe prend en compte le fait que dans Windows Azure, le « Fabric Controller » manipule les instances de façon optimisée, avec quelques incidences importantes :

- Des « Reboots » des machines et des recyclages des VM possibles en cours de vie de l'application. Les machines virtuelles sont régulièrement déplacées au sein d'un Data Center, pour des raisons de maintenance mais aussi d'optimisations énergétiques.
- Une gestion des patches et des mises à jour, aussi bien sur les Web Roles que les Worker Roles

En complément, et cela a été identifié lors de la phase de validation avec le MAT - « Migration Assessment Tool » (chapitre précédent) – certaines interactions avec le système d'exploitation hôte et ses services sont sous contraintes (l'utilisation de MSMQ par exemple, les communications utilisant UDP, intégration de composants COM, etc.



BDTHEQUE, les MAJ et les recyclages dans Windows Azure

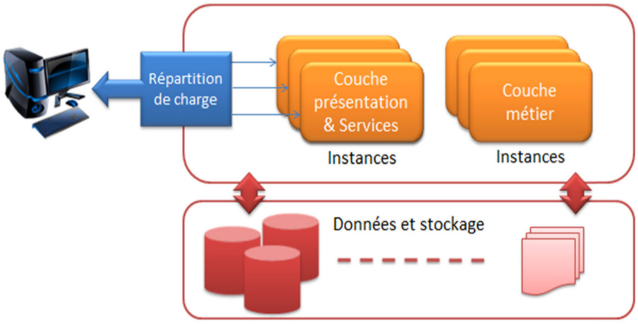
Quels sont les impacts pour l'équipe BDTHEQUE ?

Ces interventions « naturelles » sont sans conséquence pour les connectés sous réserve de respecter un certain nombre de points :

- ➔ Respecter une abstraction complète des couches basses
- ➔ Bien mettre en place les mécanismes de découplage des applications par rapport au serveur
- ➔ En fait, l'application ne doit pas tenir compte de l'OS sous-jacent

PRISE EN COMPTE DES INTERACTIONS AVEC LES SYSTÈMES DE STOCKAGES

Le stockage, qui est un stockage persistant, doit être conçu pour gérer la montée en charge imposée par les multiples instances des rôles de l'application. L'utilisation des services de stockage Windows Azure est la seule option retenue par l'équipe de façon à rester cohérent avec ces principes. En conséquence, l'équipe s'interdit toute écriture directe sur le système de fichier, dans la base de registre, en fait sur tout support « natifs » du système d'exploitation hôte.



Interaction avec le système de stockage dans BDTHEQUE

Quels sont les impacts pour l'équipe BDTHEQUE ?

Au moins trois points importants à prendre en considération par l'équipe :

- ➔ Les contextes applicatifs doivent être stockés en dehors des VMs (Windows Azure Blobs, Windows Azure Drives, Windows Azure Tables, SQL Azure, Windows Azure Caching).

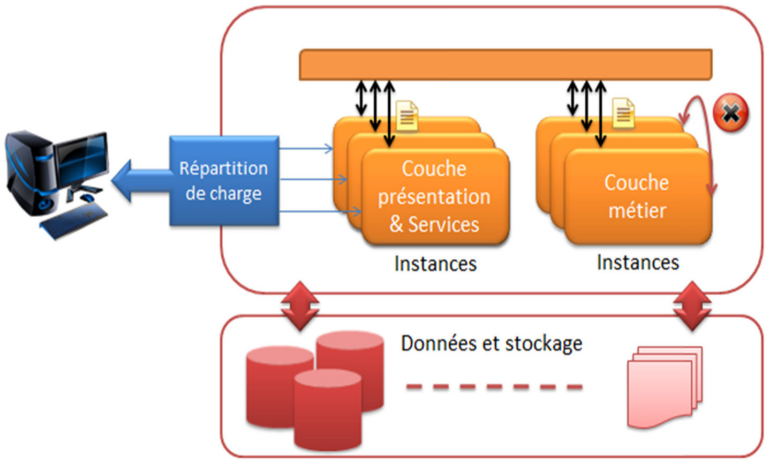
➔ L'équipe doit s'assurer que le stockage externe ne devient pas un point critique et sensible.

➔ Elle doit aussi s'assurer de la montée en charge du stockage externe.

LA COMMUNICATION ENTRE LES DIFFÉRENTS RÔLES

La séparation de l'application en plusieurs rôles implique bien sûr, à divers degrés, une communication entre ces rôles. L'équipe retient que :

- Les interactions doivent être sans état entre ces instances, quel que soit leur rôle.
- Les interactions doivent être hautement disponibles



Communication entre les rôles dans BDTHEQUE

Quels sont les impacts pour l'équipe BDTHEQUE ?

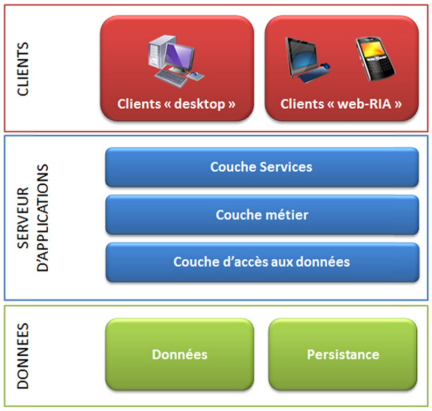
Appliquer les principes d'architecture applicative associés :

➔ Des communications asynchrones, sous forme de messages gérés par un bus de messages par exemple

➔ Utiliser les files d'attente (Windows Azure Queues) mises à disposition par la plateforme Windows Azure

2. Architecture générale

L'équipe BDTHEQUE s'appuie sur des fondamentaux biens connus et éprouvés : une architecture en couche :



Architecture en couche de la solution BDTHEQUE

Un exemple implémenté de ce type d'architecture est disponible sur CodePlex ici : http://layersample.codeplex.com/ .

Il décrit et implémente une solution exemple, mettant en œuvre tous les principes et tous les composants de la plateforme .Net 4.0 (WPF, WCF, WF, ASP.NET, etc.). Si vous souhaitez aller plus loin dans cette approche, un complément est disponible sur CodePlex, proposant sous forme d'une extension à Visual Studio 2010 : le **LAGS** ou « Layered Architecture Guidance Solution » : http://layerguidance.codeplex.com/ . C'est un ensemble d'outils qui génère à la fois du code et des projets structurés conformément aux principes présentés. Enfin, l'exemple initial a été porté, ou plutôt complété, pour cibler la plateforme Windows Azure : http://cloudsample.codeplex.com/ . C'est un bon point pour comprendre comment élaborer une architecture ciblant Windows Azure.

Pour l'équipe BDTHEQUE, ce découpage en couche se retrouve dans la structure au sein du projet Visual Studio, comme nous le découvrirons plus loin dans ce chapitre.

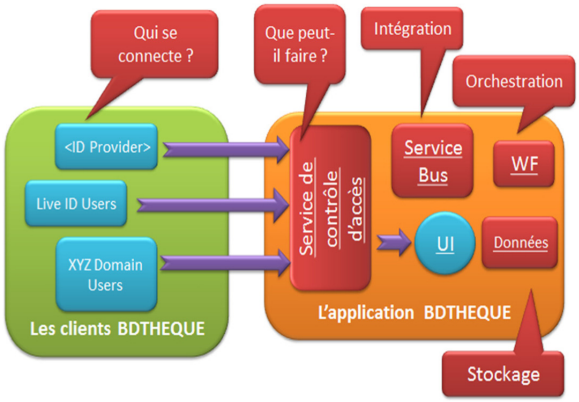
PRINCIPES FONDAMENTAUX DE LA SOLUTION

Dans un premier temps, l'équipe décide d'implémenter un client Web via une interface

HTML classique. Nous verrons que les choix d'architecture retenus autorisent de multiples clients (riches, mobiles) à se connecter à l'application.

En reprenant l'ensemble des services à sa disposition, l'équipe BDTHEQUE dispose d'une panoplie riche de services :

- Contrôle d'accès
- Un bus de services
- Un moteur de workflow, qui ne sera pas utilisé dans cette première version
- Des services d'intégration
- Des services d'orchestration, non mis en œuvre dans cette version
- Du stockage de données



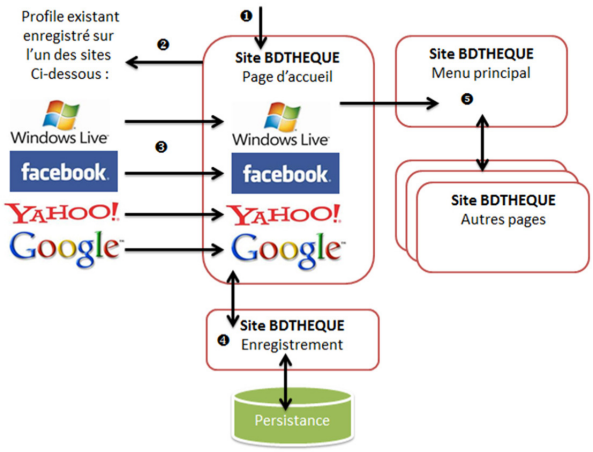
Modèle final de BDTHEQUE

Pour réaliser une première version de la solution, le choix se porte sur une solution Web simple :

- Un site web, qui sera ensuite relié à l'URL de la marque BDTHEQUE
- Une connexion via une fédération d'identité utilisant les principaux réseaux sociaux (Facebook, Live, Yahoo, etc.)
- Du stockage dans les BLOB AZURE pour les images
- Du stockage en base SQL AZURE pour toutes les autres informations
- Du caching dans Windows Azure Caching

L'équipe BDTHEQUE fait le choix de ne pas proposer de mécanisme d'authentification propre à sa solution. Pour se connecter il faudra impérativement posséder un compte chez l'un fournisseur de services suivant (la liste pourra évoluer dans le temps) :

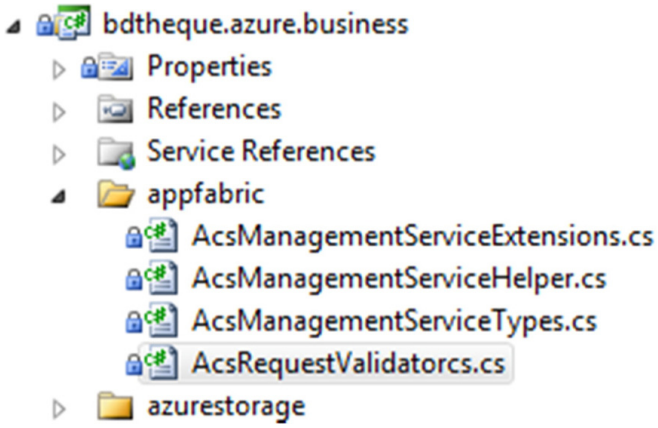
- Windows Live
- Facebook
- Yahoo
- Google



1. L'utilisateur arrive sur la page de l'application BDTHEQUE ; Il sélectionne un fournisseur de profile parmi ceux proposés
2. Si il n'est pas déjà authentifié, il doit alors le faire (la première fois cela passe bien sûr par l'autorisation pour l'application BDTHEQUE d'accéder aux informations chez le fournisseur de services sélectionné)
3. Le site BDTHEQUE récupère alors les informations qu'il peut récupérer.
4. Il propose la première fois de compléter le profile et le persiste en base de données SQL Azure. Il sera modifiable ultérieurement sur le site. Lors de la première connexion, l'utilisateur devra attendre la validation de son profile par un administrateur du site. L'approche communautaire de BDTHEQUE n'excluant pas une certaine forme de contrôle.
5. L'utilisateur, une fois authentifié peut accéder, en fonction de son profile, au différents menus et pages du site.

Pour mettre en œuvre ces mécanismes d'authentification, l'équipe BDTHEQUE s'est appuyée sur les services proposés par la plateforme Windows Azure, avec la mise en œuvre d'ACS – **Access Control Services (un composant de Windows Azure Active Directory)**, dans sa version 2.0.

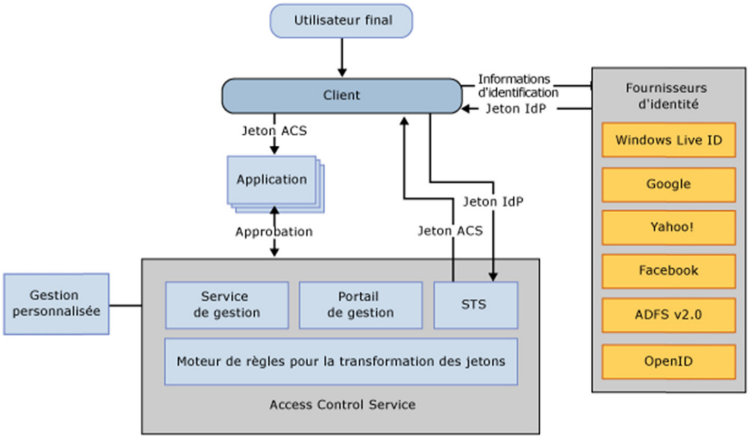
ACS 2.0 permet d'intégrer simplement un mécanisme centralisé de SSO - Single Sign On dans la solution Web BDTHEQUE. Il fonctionne avec de nombreuses solutions d'entreprise et s'intègre avec de nombreux fournisseurs d'identité.



Implémentation d'ACS dans BDTHEQUE

Le service de contrôle d'accès Windows Azure AppFabric permet de facilement authentifier et autoriser les utilisateurs pour qu'ils puissent obtenir un accès à notre application web tout en permettant à notre code de prendre en compte les fonctionnalités d'authentification et d'autorisation. L'intérêt, comme expliqué ci-dessus, est qu'au lieu d'implémenter un système d'authentification avec des comptes utilisateur propres à notre application, nous allons laisser ACS orchestrer l'authentification et la plupart des autorisations de nos utilisateurs. ACS intègre les fournisseurs d'identité standard, y compris les annuaires d'entreprises comme Active Directory et les identités Web telles que Windows Live ID, Google, Yahoo! et Facebook !

ACS est compatible avec les environnements de programmation et d'exécution les plus répandus et prend en charge de nombreux protocoles dont les protocoles Open Authorization (OAuth), OpenID, WS-Federation et WS-Trust.



Architecture générale d'ACS

Les fonctionnalités suivantes sont disponibles dans ACS :

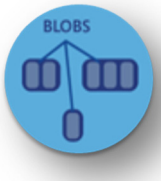
- Intégration avec Windows Identity Foundation (WIF)
- La prise en charge prête à l'emploi des fournisseurs d'identité Web répandus dont Windows Live ID, Google, Yahoo et Facebook
- Prise en charge prête à l'emploi d'Services de fédération Active Directory (AD FS) 2.0
- Prise en charge des protocoles OAuth 2.0 (draft 10), WS-Trust et WS-Federation
- Prise en charge des formats de jetons SAML 1.1, SAML 2.0 et SWT (Simple Web Token)
- Découverte de domaine Kerberos d'accueil intégré et personnalisable qui permet aux utilisateurs de choisir leur fournisseur d'identité
- Un service de gestion basé sur un protocole de données ouvertes (OData) qui fournit un accès par programme à la configuration d'ACS
- Un portail de gestion basé sur le navigateur qui autorise l'accès administratif à la configuration d'ACS

C'est donc la solution pérenne et la plus ouverte retenue par l'équipe BDTHEQUE !

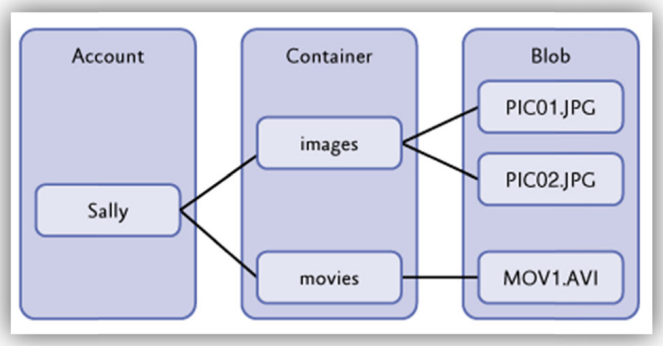
Vous trouverez plus d'informations ACS 2.0 ici : http://acs.codeplex.com/ et ici : http://msdn.microsoft.com/fr-fr/library/gg429788.aspx

STOCKAGE DES DONNEES POUR LA SOLUTION BDTHEQUE

Avant de développer ses premières lignes de code, l'équipe identifie puis isole les solutions de stockages proposées par Windows Azure qu'elle va utiliser dans sa solution. Pour rappel, les options disponibles sont :



Les Windows Azure Blobs constituent une structure hiérarchique simple, stockage de fichiers identifiés par un nom et chemin. Utilisables pour le stockage d'objets comparables à des fichiers, dans des containers, avec accès privé ou public. Ils peuvent être répliqués vers le service Windows Azure Content Delivery Network (CDN), pour une distribution globale et sont accessibles via les interfaces REST.



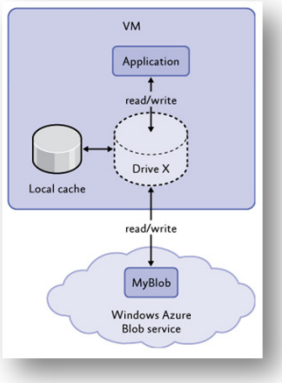
➔ **Pour BDTHEQUE, l'équipe décide d'utiliser les BLOBS pour stocker les images de la solution (couvertures, planches, photos, etc.)**

Les Windows Azure Drives sont vus comme des disques NTFS partagé entre de multiples instances en lecture uniquement (snapshots de blobs) avec l'objectif de fournir un volume NTFS persistant pour les applications Windows Azure

- Utilise les API NTFS existante pour accéder au disque persistant
- Permet de migrer dans le cloud des applications utilisant NTFS

Un Windows Azure Drive est une Page Blob

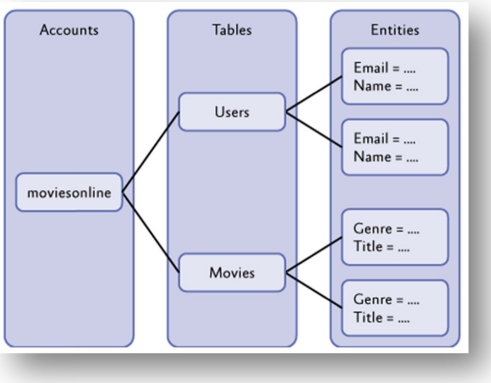
- Exemple : monter une page Blob comme X:\
- Toutes les écritures sur le disque sont persistées sur la page Blob



➔ **Pour BDTHEQUE, l'équipe décide de ne pas utiliser ce type de stockage, avant tout pour des raisons de simplicité de mise en œuvre.**

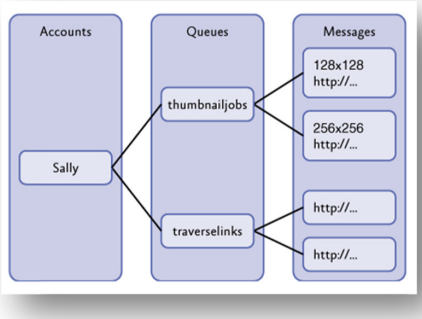
Windows Azure Tables : une structure tabulaire avec un ensemble d'entités définies par des propriétés.

Les tables fournissent un stockage structuré, sans schéma fixe et non relationnel. Ce concept de tables est massivement évolutif, hautement disponible et persistant. Le tout exposé via des API familières et facile d'utilisation.



➔ **Dans le cadre de l'application BDTHEQUE, l'équipe décide de ne pas implémenter dans la version initiale ces tables. Elle envisage par contre de tester et comparer la mise en œuvre de celles-ci avec la solution SQL Azure (performance, coûts, etc.)**

Les Windows Azure Queues, mises en œuvre selon le principe FIFO, pour la communication asynchrone entre instances. Elles sont performantes, hautement disponibles et fournissent un mécanisme fiable de distribution. Elles permettent des envois asynchrones simples, assurant qu'un message ne soit traité qu'une fois. Elles sont accessibles via REST.



➔ **Dans le cadre de l'application BDTHEQUE, l'équipe utilise ce mécanisme de files d'attente pour communiquer entre les instances des différents rôles retenus (Web Role et Worker Role)**

PATTERNS RECOMMANDES ET BONNES PRATIQUES

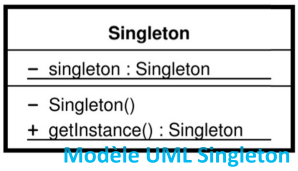
L'équipe projet BDTHEQUE après avoir défini son architecture globale identifie quelques bonnes pratiques qu'elle souhaite mettre en œuvre. Parmi ces bonnes pratiques, elle suit les recommandations de l'architecte qui préconise d'implémenter quelques patterns bien éprouvés :

- Le pattern **SINGLETON**
- Le pattern **FACADE**
- Le pattern **FACTORY METHOD**

D'autres patterns seront présentés dans les pages suivantes, associés au projet « bdtheque.data ».

Les explications de l'architecte :
Le pattern « SINGLETON », c'est quoi et ça sert à quoi ?

Le singleton est un design pattern qui a pour objectif de restreindre l'instanciation d'une classe à un seul objet (parfois à quelques objets seulement). On l'applique quand un seul objet est nécessaire pour coordonner des opérations dans un système. Le pattern peut aussi être utilisé pour son efficacité ou son efficience : le système occupe alors moins de mémoire avec peu d'objets instanciés plutôt que d'avoir de nombreux objets similaires.



Modèle UML Singleton

```
namespace bdtheque.business.bo
{
    public sealed class Factory : IFactory
    {
        // Singleton Implementation
        private Factory() { }

        // Singleton Implementation
        public static Factory GetInstance
        {
            get { return FactoryImpl.instance; }
        }

        // Factory Method
        public T Create<T>()
            where T : new()
        {
            return new T();
        }

        // Singleton Implementation
        private class FactoryImpl
        {
            internal static readonly Factory instance = new Factory();

            static FactoryImpl() { }
        }
    }
}
```

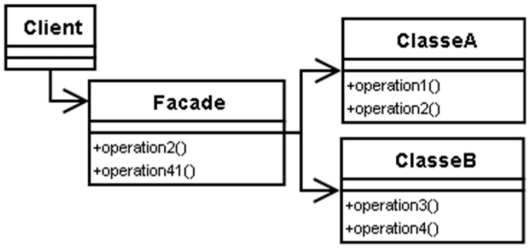
Implémentation Singleton dans BDTHEQUE

Pour implémenter un singleton il faut écrire une classe contenant une méthode qui crée une instance seulement s'il n'en existe pas encore une. Dans le cas contraire elle renvoie une référence vers l'objet instancié précédemment. Il faut veiller à ce que le constructeur de la classe soit privé, de façon à s'assurer que la classe ne puisse être instanciée autrement que par la méthode de création sous contrôle.

Voir http://msdn.microsoft.com/en-us/library/ff650316.aspx pour exemples et informations complémentaires.

Les explications de l'architecte :
Le pattern « FACADE », c'est quoi et ça sert à quoi ?

Le FACADE design pattern fournit une interface simple pour masquer une conception plus complexe et difficile à comprendre. La façade simplifiée propose ainsi des fonctionnalités réduites tout en exposant les fonctions nécessaires aux utilisateurs. La façade est un pattern éprouvé, qui encapsule la complexité des interactions entre les objets métiers.



Modèle UML du pattern Facade

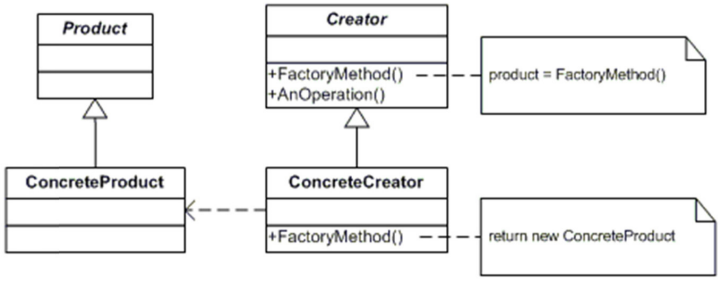
```
// Implementation of the Facade Pattern to encapsulate access to some of the Session Variables
public static class SessionFacade
{
    private const string authenticatedUserId = "AuthenticatedUserId";
    private const string authenticatedUserIsActive = "AuthenticatedUserIsActive";
    private const string authenticatedUserRole = "AuthenticatedUserRole";

    public static int? UserId
    {
        get
        {
            if(HttpContext.Current.Session[authenticatedUserId] == null)
                return null;
            else
                return int.Parse(HttpContext.Current.Session[authenticatedUserId].ToString());
        }

        set { HttpContext.Current.Session[authenticatedUserId] = value; }
    }
```

Implémentation du pattern Facade dans BDTHEQUE

Voir http://msdn.microsoft.com/en-us/library/orm-9780596527730-01-04.aspx pour exemples et informations complémentaires.

Les explications de l'architecte :
Le pattern « FACTORY METHOD », c'est quoi et ça sert à quoi ?

Ce troisième pattern, appelé FACTORY METHOD permet de définir une interface pour la création d'un objet en laissant à des sous-classes le choix des classes à instancier. Le rôle d'une Factory est donc d'instancier des objets de différents types sans avoir à spécifier la classe de l'objet qui sera créé. Son principal avantage est d'éviter la redondance entre ces classes. Les méthodes associées à ce pattern Factory sont en général implémentées via un Singleton.



Modèle UML du pattern Factory Method

```
public sealed class BdthequeUnitOfWorkFactory : IUnitOfWorkFactory
{
    // Singleton Implementation
    private BdthequeUnitOfWorkFactory() { }

    // Singleton Implementation
    public static BdthequeUnitOfWorkFactory GetInstance...

    public static string ConnectionString...

    // Factory Method
    public IUnitOfWork Create(string connectionString)
    {
        if (connectionString == null)
            throw new ArgumentNullException("ConnectionString");

        return new BdthequeUnitOfWork(connectionString);
    }
```

Implémentation du pattern Factory dans BDTHEQUE

Voir http://msdn.microsoft.com/en-us/library/ee817667.aspx pour exemples et informations complémentaires.

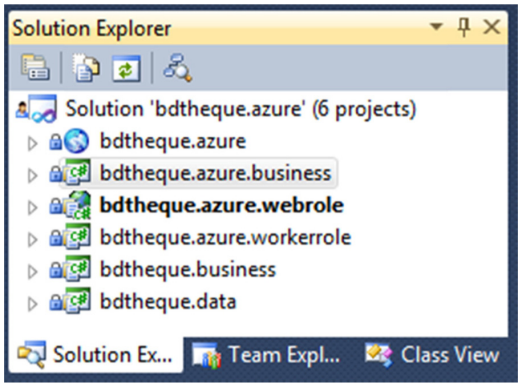
3. Implémentation de la solution

D'autres patterns existent et sont aussi retenus par l'équipe BDTHEQUE, assistée de son architecte logiciel. Ils vont être présentés en situation dans les pages suivantes pour une meilleure compréhension.

Regardons maintenant comment le projet est implémenté dans Visual Studio :

LE PROJET DANS VISUAL STUDIO

Le projet ouvert dans Visual Studio 2010 est celui récupéré sur CodePlex, ici : http://bdtheque.codeplex.com/



Structure globale de la solution BDTHEQUE

Structure globale du projet BDTHEQUE : l'équipe découpe la solution en 6 projets que vous retrouvez sous la solution globale **« bdtheque.azure »**. Ce découpage est retenu pour les raisons suivantes :

Il est en partie, du moins pour 3 des projets **(bdtheque.azure, bdtheque.azure.webrole et bdtheque.azure.workerrole)**, imposé par la structure même d'un projet de type Windows Azure dans Visual Studio.

Il correspond aux principes d'architecture évoqués ci-dessus, une architecture en couche, avec représentation de la couche métier (scindée en deux parties **bdtheque.business** et **bdtheque.azure.business** pour isoler la logique métier spécifique à Windows Azure) et de

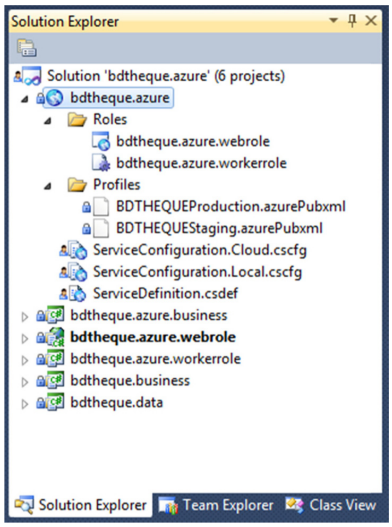
la couche d'accès aux données **(bdtheque.data)**. La couche UI (User Interface) est en fait représentée par **bdtheque.azure.webrole**.

Ce découpage permet un développement plus contrôlé de la solution en identifiant clairement les différents modules et leurs intervenants. La maintenance, la mise à jour et le déploiement s'en trouve donc facilités.

Passons rapidement en revue les 6 projets de la solution et ce qu'ils contiennent. Les 3 projets qui suivent sont ceux que Visual Studio a insérés dans la solution lorsque l'équipe a créé son projet Windows Azure :

LE PROJET BDTHEQUE.AZURE

Ce projet, créé par Visual studio lors de la création de la solution Windows Azure contient toutes les informations utiles à l'ensemble du projet (c'est le point central de configuration de la solution). On y trouve dans l'ordre :



Structure du projet racine BDTHEQUE

- Le référencement et la configuration des rôles de la solution BDTHEQUE. A savoir ici les deux rôles que l'équipe a ajoutés : un **« webrole »** et un **« workerrole »**. Ils contiennent pour l'essentiel des informations décrites dans le premier chapitre de ce livre. Les deux interfaces de saisies sont là pour faciliter la saisie des

informations qui sont pour l'essentiel contenues dans les fichiers XML ci-dessous.

- Deux profiles de configuration pour la publication des applications, un pour la version « staging » et un pour la version « production ». Ce sont des fichiers XML simples à comprendre, ils contiennent les noms des services, des espaces de stockage, etc.
- Des fichiers de configuration (.cscfg) du projet en local et sur les Data Centers Windows Azure. Au format XML, ils contiennent une section pour chacun des rôles de la solution, contenant des paramètres d'accès aux données, ACS, de sécurité, etc.

```
ServiceConfi....Local.cscfg •
<ServiceConfiguration serviceName="bdtheque.azure" xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration"
  <Role name="bdtheque.azure.webrole">
    <Instances count="2" />
    <ConfigurationSettings>
      <!-- Data Access & Connection Strings-->
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" value="UseDevelopmentStorage=true" />
      <Setting name="ComicsConnectionString" value="Server=localhost;Database=BDTHEQUE;Trusted_Connection=true;" />
      <Setting name="DataConnectionString" value="DefaultEndpointsProtocol=https; AccountName=bdtheque; AccountKey=[YOUR KEY]" />
      <Setting name="DropCreateDatabaseBdTheque" value="true" />
      <!-- Windows Azure ACS & Security Settings-->
      <Setting name="AcsServiceNamespace" value="https://bdtheque.accesscontrol.windows.net/" />
      <Setting name="AcsManagementServiceIdentityName" value="ManagementClient" />
      <Setting name="AcsManagementServiceIdentityKey=" value="[YOUR KEY]" />
      <Setting name="AcsHostUrl" value="accesscontrol.windows.net" />
      <Setting name="AcsManagementServicesRelativeUrl" value="v2/mgmt/service/" />
      <Setting name="AcsCustomLoginPage" value="Login.aspx" />
      <Setting name="AcsRealm" value="127.0.0.1%3a444" />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name="bdtheque-cert" thumbprint="[YOUR THUMBPRINT]" thumbprintAlgorithm="sha1" />
    </Certificates>
  </Role>
  <Role name="bdtheque.azure.workerrole">
    <Instances count="2" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

Exemple de configuration des rôles en local (ServiceConfiguration.Local.cscfg)

- Enfin un dernier fichier XML, ServicesDefinition.csdef, qui contient pour chaque rôle de la solution les définitions des configurations des services (un peu à la mode WCF). On y trouve les Bindings, les EndPoints, les services utilisés, etc.

LE PROJET BDTHEQUE.AZURE.WEBROLE

Le deuxième projet est le celui du webrole. Un projet ASP.Net relativement classique, qui contient nos pages ASPX et les fichiers de configuration habituels. L'équipe projet a décidé d'utiliser Entity Framework en version 4.2 en utilisant l'approche Code First, qui est détaillée dans les pages suivantes.

Ce rôle ayant vocation à héberger le site web de la solution finale, l'équipe BDTHEQUE en profite pour présenter et valider les principales pages de son future site web. Cette présentation constitue le socle et il sera enrichi ultérieurement d'autres fonctionnalités et donc d'autres pages web.

Avant tout, pour répondre aux besoins fonctionnels et aux aspects communautaires attendus, la solution retenue offrira 3 types de profiles susceptibles de se connecter :

1. Profile **Administrator** : Il a accès à tout le contenu du site, il peut activer les utilisateurs, ajouter des bandes-dessinées dans la base de données ainsi que des séries, créer des auteurs et éditeurs, il peut ensuite comme tout autre utilisateur, gérer sa propre collection.

2. Profile **Moderator** : Le modérateur peut créer des bandes-dessinées, des séries, gérer sa propre collection, créer des auteurs et éditeurs, il peut ensuite comme tout autre utilisateur, gérer sa propre collection. C'est un rôle clé du point de vue communautaire.

3. Profile **User** : L'utilisateur classique peut gérer sa propre collection, en effectuant des recherches dans la base de données (il ne peut ajouter à sa collection, que des BDs et/ou séries existante). A terme un utilisateur pourra créer des bandes-dessinées et séries, mais qui devront être validées par un modérateur avant de devenir visibles par tous sur le site.

Utilisation du site dans sa version initiale :

- L'utilisateur se connecte via Gmail, Facebook, Yahoo ou LiveID Lors de la première connexion, l'utilisateur remplit son profil.
- L'utilisateur ensuite saisit les infos de son profil (mail, …)
- L'utilisateur attend qu'un administrateur active son profil (Activate User) et lui attribue l'un des 3 rôles
- L'utilisateur peut ensuite naviguer sur le site (Home, Accueil)

Principaux écrans de l'application :

- **My Collection** se divise en deux sous-menus : **My Comics** permet de gérer ses bandes-dessinées, **My Series** permet de gérer les séries
- **Profile** permet de modifier le profil, **Admin** n'est visible que pour les administrateurs afin d'activer les utilisateurs
- **My Comics** affiche l'ensemble des bandes-dessinées d'un utilisateur (indépendant de ses séries), ainsi que des infos détaillées pour chaque BD, il y a un lien permettant d'ajouter une BD. Deux parties composent la page, une partie permet de rechercher une BD, et l'ajouter à sa propre collection (les utilisateurs classiques n'ont accès qu'à cette partie). La deuxième permet d'ajouter une BD inexistante dans la base (accessible uniquement pour les administrateurs et modérateurs)
- **My Series** répertorie les séries de l'utilisateur, ainsi que des infos détaillées pour chaque série, une fonction d'ajout de série est disponible, elle permet d'ajouter une série existante (utilisateur normal) ou une nouvelle série (modérateur ou administrateur), enfin il est possible d'afficher le détail des bandes-dessinées concernant la série, et d'ajouter une BD à la série (on est par conséquent redirigé vers le formulaire d'ajout de BD).

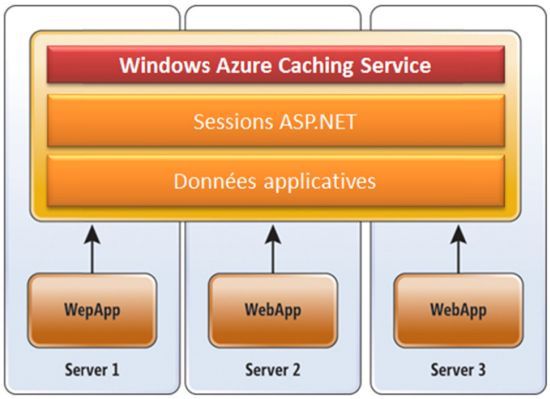
GESTION DES SESSIONS : OPTIMISATION AVEC WINDOWS AZURE CACHING

L'équipe profite de son travail sur le Web Role pour évoquer la problématique de gestion des sessions (ASP.NET) en environnement mutualisé. Par défaut le serveur IIS prend en charge la gestion de ces sessions de façon centralisée, mais le modèle de base doit être modifié de façon à externaliser la persistance de ces sessions. Plusieurs solutions sont possibles (SQL Azure, Windows Azure Table Storage et Windows Azure Caching Service) et l'équipe s'attache à évaluer les différentes alternatives offertes. C'est l'occasion de comprendre et voir si les services de caching proposés par la plateforme Windows Azure sont une bonne solution ?

Le choix de SQL Azure ou Windows Azure Table Storage impose une contrainte forte de gestion manuelle des sessions périmées. La solution Windows Azure Caching Service offre cette gestion mais aussi de nombreux autres avantages.

Windows Azure Caching Service :

- Un cache en mémoire distribué pour les applications Windows Azure et SQL Azure
- Un accès très rapide et échelonné aux données de l'application
- Une mise en cache qui dispose d'un modèle d'approvisionnement très simple : aucune infrastructure complexe à installer, à configurer, ni à gérer. Ces points sont pris en compte par le service lui-même. Il faut configurer uniquement deux options : les données à stocker et la taille du cache
- Pour les applications ASP.NET : une configuration déclarative très facile via le fichier Web.config



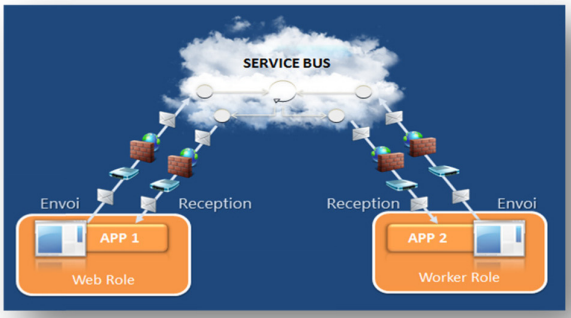
Le projet bdtheque.azure.workerrole

Le troisième projet est le celui du Worker Role. Ce projet a été sélectionné puis ajouté automatiquement lors de la création du projet Windows Azure par l'équipe projet. Un projet de ce type va contenir toutes les fonctionnalités de traitement d'arrière-plan (préparation des images, conversion du format, génération des icônes…).

Le Web Role et le Worker Role communiquent grâce à un mécanisme de communication asynchrone basé sur les files d'attente, mécanisme intégré à la plateforme Windows Azure, les Windows Azure Queue, dans lesquelles sont empilées les tâches (Messages) à traiter. Le Worker Role récupère ces tâches pour traitement de façon asynchrone et les met à disposition de l'application BDTHEQUE.

WINDOWS AZURE SERVICE BUS (COMMUNICATION AVANCÉE)

D'une façon générale, le Service Bus de Windows Azure AppFabric contient toutes les fonctionnalités utiles et attendues, qui vont au-delà de la simple communication entre les rôles d'une même application :



Les fondamentaux d'un Service Bus sont disponibles :

- Exposition de services au-delà des frontières du réseau
  - Traversée de pare-feu / NAT
  - Connectivité intermittente ou mobile
- Communication entre applications et services selon différentes topologies
  - One-way / Full-duplex
  - Unicast / Multicast
  - Orienté session / Peer-to-peer
  - Publish / Subscribe
- Enregistrement / annuaire pour la découverte de services
  - URIs stables
  - Support de REST et HTTP

Dans la première version de l'application, ce Worker Role n'est pas utilisé ; il le sera une fois le projet web déployé et opérationnel. L'équipe BDTHEQUE viendra alors implémenter dans

ce projet les traitements utiles. Nous vous invitons à suivre l'évolution de ce livre blanc et de l'application associée, en ligne, sur http://www.dotnetrangers.net ou sur http://visualstudio.fr/scenarios/developper-pour-le-cloud.aspx. La prochaine étape sera la mise en œuvre de ce Worker Role.

Les 3 autres projets de la solution sont ceux que l'équipe a créés pour développer son application conformément à ses principes d'architecture. Ils sont présentés ci-dessous dans l'ordre dans lequel ils ont été traités.

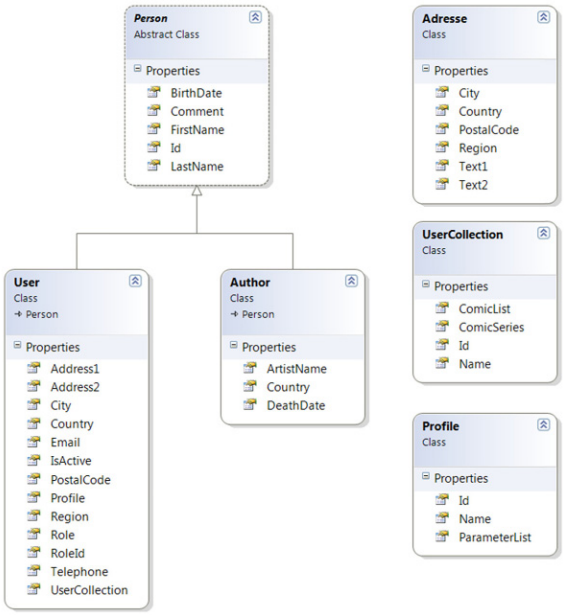
LE PROJET BDTHEQUE.DATA

Ce quatrième projet et les deux suivants sont entièrement créés ; ils ne sont pas issus de la création du projet Windows Azure par Visual Studio. Les données étant importantes, l'équipe commence par ce module. Elle range son code et ses différentes classes dans 4 dossiers :

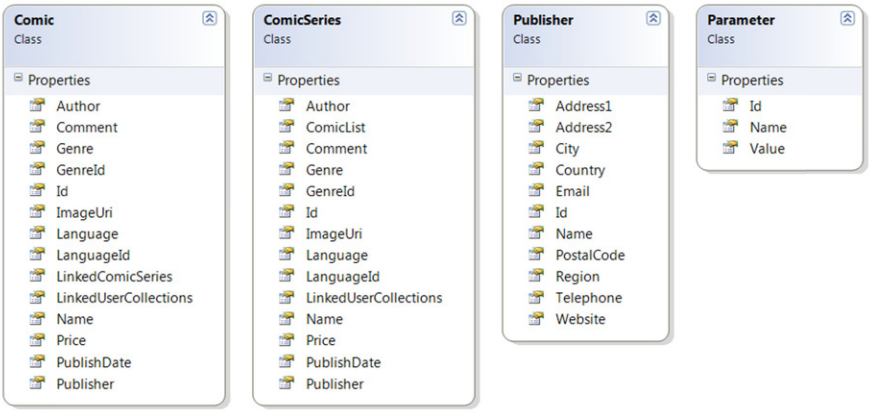
- **dataaccess** : accès, validation des données via Entity Framework
- **enums :** contient des énumérations utiles au projet
- **interfaces :** implémentées via les patterns UnitOfWork et Factory
- **poco :** contenant les classes simples des concepts manipulés.

Pour comprendre les données manipulées dans l'application, le plus simple est de consulter le diagramme des classes « POCO » :

Pour ce qui concerne les utilisateurs de la solution :

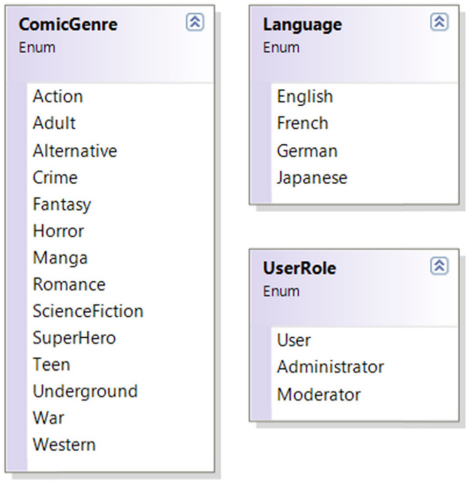


Et pour ce qui concerne les objets manipulés :



Et enfin quelques énumérations utiles pour le projet, correspondant aux énumérations décrites dans le dossier **enum** :

- Une associée aux différents types de bandes dessinées répertoriées
- Une autre associée aux langues supportées par la solution
- Une dernière associée aux profils possibles pour les utilisateurs



Les explications de l'architecte :
POCO, qu'est-ce que c'est et à quoi ça sert ?

Un Plain Old CLR Object ou encore POCO est un objet qui se veut être le plus simple possible et qui hérite donc de la classe "Object". Ses propriétés sont censées rester les plus basiques. Un POCO ne doit pas contenir de code spécifique à la couche d'accès aux données à laquelle il s'adresse, pas plus qu'à la couche UI à laquelle il fournit des informations. Le principe est d'avoir un objet découplé au maximum des couches de l'application. Cet objet est alors indépendant, il peut donc être utilisé sans modification à plusieurs endroits.

En voici un exemple qui définit une bande-dessinée (COMIC) :

```
public class Comic
{
    public int Id { get; set; }
    public string Name { get; set; }
    public virtual Author Author { get; set; }
    // Enum workaround for EF 4.2 limitations
    public ComicGenre Genre...
    public int GenreId { get; set; }
    public string Comment { get; set; }
    // Enum workaround for EF 4.2 limitations
    public Language Language...
    public int LanguageId { get; set; }
    public virtual Publisher Publisher { get; set; }
    public DateTime? PublishDate { get; set; }
    public decimal Price { get; set; }
    public string ImageUri { get; set; }
    public virtual List<UserCollection> LinkedUserCollections { get; set; }
    public virtual List<ComicSeries> LinkedComicSeries { get; set; }
}
```

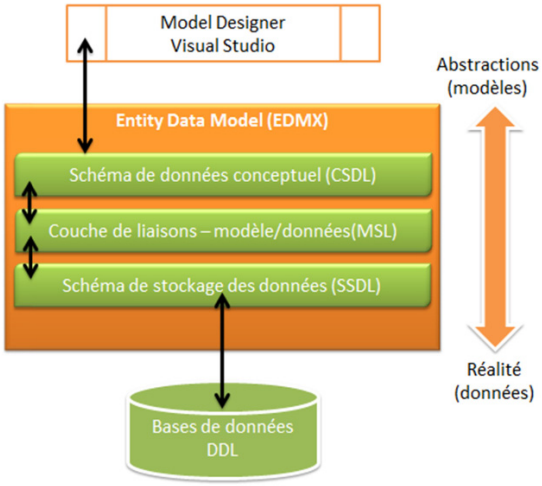
Un bon point d'entrée pour comprendre comment implémenter les principes POCO dans votre solution est de passer par le site MSDN : http://msdn.microsoft.com/fr-fr/library/dd456853.aspx

Les explications de l'architecte :
Entity Framework et Code First, qu'est-ce que c'est et à quoi ça sert ?

Au-delà des aspects cloud liés à Windows Azure, l'équipe BDTHEQUE est consciente de développer une solution relativement classique, manipulant des données et des objets des plus simples aux plus structurés, sans complexité algorithmique avérée. Par contre, forte de son expérience, elle souhaite utiliser ou produire un code applicatif pour accéder aux données, s'appuyant à la fois sur les meilleures pratiques du moment et si possible sur un

ensemble de classes existantes et éprouvées. Elle décide après une étude de s'appuyer sur **« Entity Framework »**.

ADO.NET Entity Framework permet aux développeurs de créer des applications d'accès aux données en programmant par rapport à un modèle d'application conceptuel au lieu de programmer directement par rapport à un schéma de stockage relationnel. L'objectif est de diminuer la quantité de code et les besoins en maintenance pour les applications orientées objet.



Les applications reposant sur Entity Framework présentent les avantages suivants :

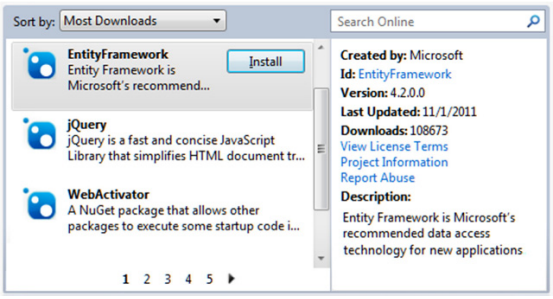
- les applications peuvent fonctionner par rapport à un modèle conceptuel plus centré sur les applications, comprenant notamment des types avec héritage, des membres complexes et des relations ;
- les applications sont libérées des dépendances codées en dur sur un moteur de données ou un schéma de stockage particulier ;
- les mappages entre le modèle conceptuel et le schéma spécifique au stockage peuvent être modifiés sans changer le code de l'application ;
- les développeurs peuvent utiliser un modèle objet d'application cohérent qui peut être mappé à différents schémas de stockage implémentés dans divers systèmes de gestion de bases de données ;

- plusieurs modèles conceptuels peuvent être mappés à un schéma de stockage unique ;
- la prise en charge de la fonctionnalité LINQ (Language-Integrated Query) permet d'effectuer la validation de la syntaxe au moment de la compilation par rapport à un modèle conceptuel.

Voir http://msdn.microsoft.com/en-us/library/bb399572.aspx pour exemples et informations complémentaires sur Entity Framework.

Entity Framework offre une approche centrée en priorité sur le code appelée Code First. Avec cette approche, on démarre par le code, il n'y a ni schéma, ni XML décrivant le modèle. L'équipe BDTHEQUE va créer ses entités et sa base de données à partir de simples classes POCO.

Concrètement, il faut commencer par référencer la librairie Entity Framewok dans le projet. L'équipe BDTHEQUE en profite pour essayer d'améliorer son processus de gestion des librairies et packages externes : par expérience, suivre et mettre à jour ces référence dans les différents projets s'avère vite fastidieux. Un produit permet de prendre en compte tous ces aspects : **Nuget**. C'est un gestionnaire de packages .NET disponible sous forme d'une extension pour les environnements de développement .NET Visual Studio (http://nuget.codeplex.com/).




```
PM> Install-Package EntityFramework -Version 4.2.0.0
```

Nuget est devenu une référence dans l'écosystème .NET ; il est d'ailleurs intégré par défaut dans le Framework Web ASP.NET MVC depuis sa version 3. C'est une solution qui a été rapidement adoptée par les développeurs. L'outil est actuellement à sa version 1.6 et bénéficie du support de Microsoft pour son développement.

Ensuite pour la création du modèle, comme son nom l'indique, l'approche Code First consiste à écrire en premier le code, car les tables de notre base seront représentées par de simples classes. Ce sont les classes POCO que l'équipe BDTHEQUE vous a présentées ci-dessus. Elles ne contiennent que des propriétés, équipées d'un « get » et d'un « set ».

Ce n'est pas la peine de chercher à ajouter des annotations ou même des attributs à nos classes, car Code First se chargera de tout. Il faut simplement créer une troisième classe, héritant de « System.Data.Entity.DbContext » afin de représenter notre base de données. C'est cette classe qui sera notre point d'entrée, qui représentera nos entités et qui nous permettra d'accéder à notre base.

```
public class BdthequeDataContext : DbContext
{
    // Public Constructors
    public BdthequeDataContext(string connectionString) : base(connectionString) { }
    public BdthequeDataContext() : base() { }

    // DbSet Definitions
    public DbSet<Author> AuthorDbSet { get; set; }
    public DbSet<Comic> ComicDbSet { get; set; }
    public DbSet<ComicSeries> ComicSeriesDbSet { get; set; }
    public DbSet<Parameter> ParameterDbSet { get; set; }
    public DbSet<Profile> ProfileDbSet { get; set; }
    public DbSet<Publisher> PublisherDbSet { get; set; }
    public DbSet<User> UserDbSet { get; set; }
    public DbSet<UserCollection> UserCollectionDbSet { get; set; }
```

Entity Framework et DbContext dans le projet BDTHEQUE

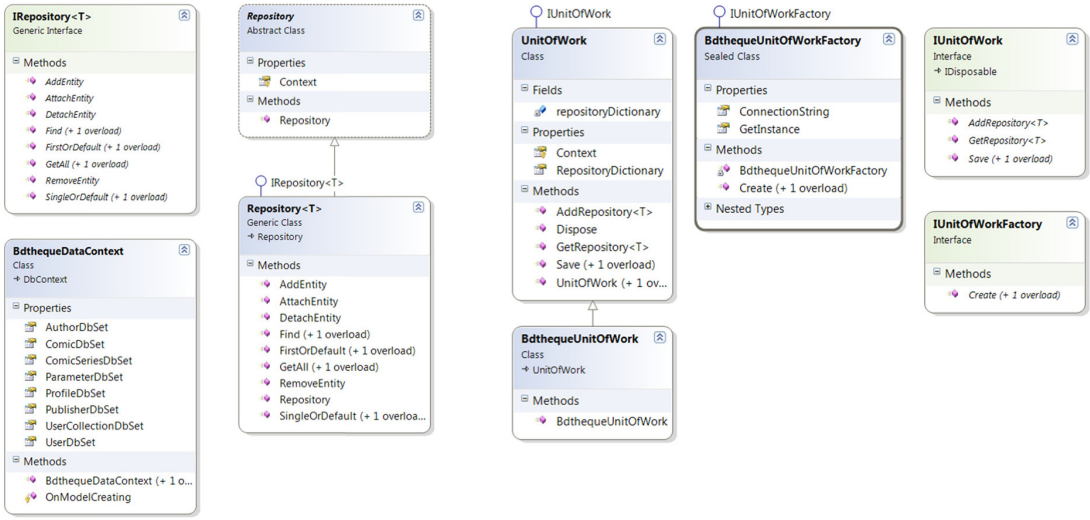
Voilà, ces quelques lignes suffisent à Entity Framework pour créer une base de données. Il suffit d'une classe héritant de DbContext et d'une propriété de type DbSet<T> pour chaque entité. Ensuite, l'équipe utilise la « Code First Fluent API » en implémentant la méthode OnModelCreating(…) de la classe dérivée et en définissant toutes les contraintes valables pour le domaine. De cette façon il est possible d'émettre des contraintes qui seront interprétées puis insérées dans la base de données cible.

```
// Model Restrictions
protected override void OnModelCreating(DbModelBuilder dbModelBuilder)
{
    // Author Restrictions
    dbModelBuilder.Entity<Author>().Property(a => a.Id).IsRequired();
    dbModelBuilder.Entity<Author>().Property(a => a.FirstName)
        .HasMaxLength(60)
        .IsRequired();
    dbModelBuilder.Entity<Author>().Property(a => a.LastName)
        .HasMaxLength(100)
        .IsRequired();
```

Voir http://blogs.msdn.com/b/adonet/archive/2011/09/28/ef-4-2-code-first-walkthrough.aspx pour exemples et informations complémentaires sur Code First.

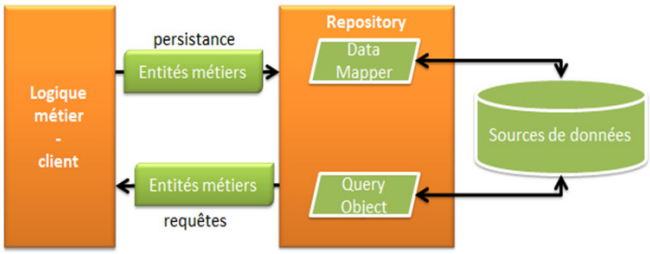
LES DIAGRAMMES DE CLASSES

Une façon simple pour l'équipe BDTHEQUE de présenter les concepts manipulés par l'application consiste à présenter les diagrammes de classes utilisés. Pour la couche d'accès aux données, l'équipe décide d'utiliser les patterns « Repository » et « UnitOfWork » afin de ne pas conserver un couplage fort avec Entity Framework (l'équipe souhaite se garder la possibilité d'optimiser si nécessaire ses accès aux données).



Les explications de l'architecte :
Le pattern « Repository », c'est quoi et ça sert à quoi ?

Le pattern Repository fournit une façade qui vient se greffer sur la couche d'accès aux données. Cette façade est une abstraction qui permet de se prémunir de la connaissance de la technologie sous-jacente, que ce soit Entity Framework, ADO.NET, NHibernate... Elle permet également de modifier l'implémentation de la couche d'accès sans avoir à impacter la couche métier. La dernière version d'Entity Framework inclut de multiples fonctionnalités pour implémenter le Repository Pattern très simplement. L'utilisation couplée des 2 patterns Repository et UnitOfWork améliore la maintenabilité des applications grâce à une architecture solide, à une réutilisation du code renforcée et à une capacité de test accrue.



Le pattern Repository

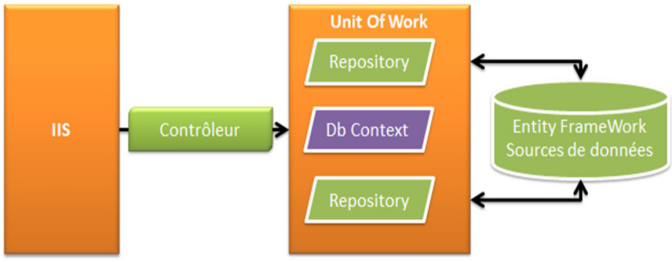
```
public interface IRepository<T>
    where T : class
{
    IEnumerable<T> GetAll();
    IEnumerable<T> GetAll(string includePath);
    IEnumerable<T> Find(Expression<Func<T, bool>> predicate);
    IEnumerable<T> Find(Expression<Func<T, bool>> predicate, string includePath);
    T FirstOrDefault(Expression<Func<T, bool>> predicate);
    T FirstOrDefault(Expression<Func<T, bool>> predicate, string includePath);
    T SingleOrDefault(Expression<Func<T, bool>> predicate);
    T SingleOrDefault(Expression<Func<T, bool>> predicate, string includePath);
    void AddEntity(T entity);
    void RemoveEntity(T entity);
    void AttachEntity(T entity);
    void DetachEntity(T entity);
}
```

Le pattern Repository implémenté dans BDTHEQUE

Voir http://msdn.microsoft.com/en-us/library/ff649690.aspx et http://blogs.msdn.com/b/ado-net/archive/2009/06/16/using-repository-and-unit-of-work-patterns-with-entity-framework-4-0.aspx pour exemples et informations complémentaires.

Les explications de l'architecte :
Le pattern « Repository », c'est quoi et ça sert à quoi ?

Le pattern UnitOfWork sert à coordonner les écritures successives de transactions. Il permet de préparer une série de modifications au sein de l'application et de les regrouper en une seule transaction atomique. De plus, il permet aussi de traquer les changements et gérer les problèmes de concurrence.



Le pattern UnitOfWork

```
public interface IUnitOfWork : IDisposable
{
    void AddRepository<T>(Repository<T> repository) where T : class;
    Repository<T> GetRepository<T>() where T : class;
    void Save();
    void Save(bool detectChanges);
}
```

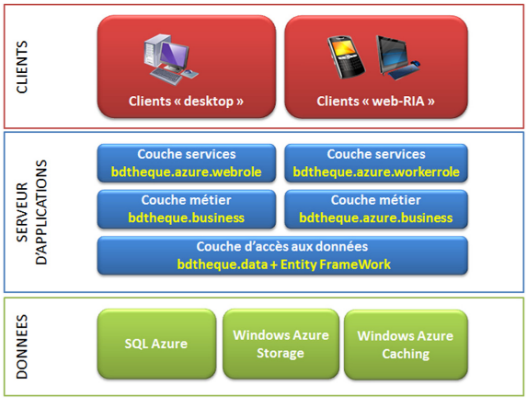
Le pattern UnitOfWork implémenté dans BDTHEQUE

Voir http://msdn.microsoft.com/en-us/magazine/dd882510.aspx et http://blogs.msdn.com/b/adonet/archive/2009/06/16/using-repository-and-unit-of-work-patterns-with-entity-framework-4-0.aspx pour exemples et informations complémentaires.

LE PROJET BDTHEQUE.BUSINESS

Le socle technique d'accès aux données est maintenant bien défini et implémenté, sur des bases solides mettant en œuvre les bons patterns et les bonnes pratiques en usage. Fort de ces acquis, l'équipe BDTHEQUE remonte maintenant vers la couche métier.

La lecture de l'architecture en couche associée aux différents projets de la solution est maintenant bien lisible



Le projet BDTHEQUE et ses couches

Dans ce découpage, l'équipe BDTHEQUE a choisi de découpler la couche métier en deux modules pour gagner en lisibilité et en maintenance :

- Un module contenant les objets métiers neutres, sans aucune adhérence au modèle Windows Azure : **bdtheque.business**
- Un second module contenant lui des objets métiers embarquant des spécificités liées à la plateforme Windows Azure : **bdtheque.azure.business**

Le projet bdtheque.business inclut la logique métier de l'application non specifique à Windows Azure.

L'équipe projet place donc ici, dans deux dossiers distincts **bo** et **interfaces** :

- L'ensemble des fonctionnalités dédié à la gestion des utilisateurs (login et activation des utilisateurs)
- La création des objets POCO via le Factory Method Pattern
- Des facilitateurs pour une autoriser une interaction avec WIF (Windows Identity Foundation) plus simple
- Les interfaces correspondantes aux fonctionnalités ci-dessus

LE PROJET BDTHEQUE.AZURE.BUSINESS

Ce projet bdtheque.azure.business inclut lui en complément la logique métier de l'application specifique à Windows Azure. Ici on trouve implémenté par l'équipe projet dans deux dossiers **appfabric** et **azurestorage** :

- Des facilitateurs pour une interaction avec ACS (un composant de Windows Azure Active Directory) plus efficace
- Des facilitateurs pour une interaction avec les services de stockage de Windows Azure (Windows Azure Storage : Windows Azure Blobs, Windows Azure Queues, Windows Azure Tables) plus rapide

ALM & Azure : Améliorer la qualité logicielle et la productivité

L'équipe BDTHEQUE possède maintenant un processus de production logicielle ciblant la plateforme Windows Azure, mais aussi un socle applicatif conçu sur une architecture performante et pérenne. Avant de rentrer en phase de production, elle souhaite **industrialiser son développement**, ou du moins mettre en place toutes les conditions lui permettant de travailler **efficacement** dans un **processus d'amélioration continue**. Elle désire identifier puis mettre en place tout ce qui permettra à la solution BDTHEQUE d'atteindre la **qualité optimale** en restant focalisée sur deux priorités :

- Une productivité mesurable, afin de conserver des ratios économiques optimaux en termes de développement et mise en production.
- Une agilité opérationnelle, de façon à pouvoir réagir efficacement aux demandes du marché et des utilisateurs. Le travail accomplie précédemment en terme d'architecture a déjà permis de poser les bases d'un solide socle applicatif agile.

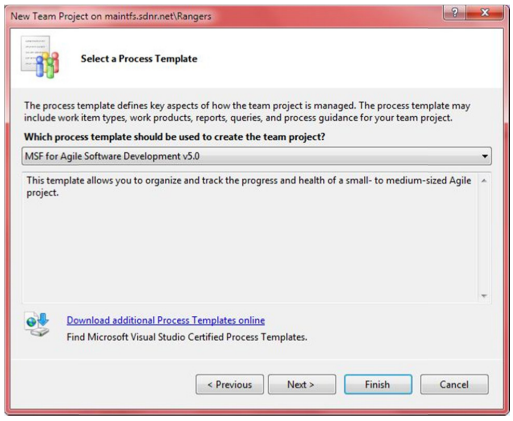
Elle s'attache de façon pragmatique à construire un socle de bonnes pratiques **ALM – Application Lifecycle Management**, en s'appuyant sur trois fondamentaux :

1. Une **méthodologie** : l'équipe souhaite donner un cadre méthodologique à sa production logicielle qui lui permette de tenir compte de son niveau de maturité actuel tout en tenant une trajectoire d'optimisation permanente. Elle décide pour cela de s'appuyer sur un cadre méthodologique agile.
2. Un outil évolutif et intuitif : chaque développeur connecte son Visual Studio sur un serveur de développement TFS 2010.
3. Une équipe motivée ! l'ALM ce n'est pas que des outils et des processus, c'est avant tout « travailler ensemble efficacement ».

1. Le projet « BDTHEQUE » dans Team Foundation Server 2010

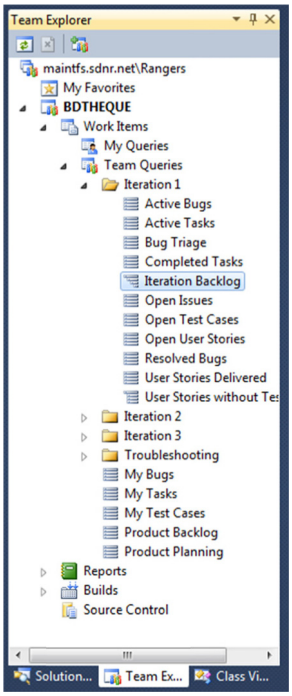
L'équipe se connecte donc sur son serveur TFS 2010, un serveur hébergé en externe, pour y créer son premier projet d'équipe. Elle s'identifie sur le serveur, puis crée un projet en sélectionnant un **« Process Template »** conforme à son choix orienté vers l'agilité : **« MSF Agile »**.

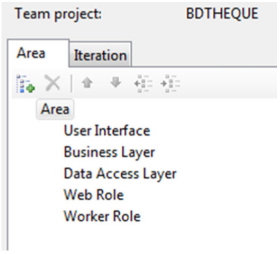
Elle configure les droits d'accès pour les membres de l'équipe, puis le module de contrôle de source et de gestion des branches, anticipant ainsi les futures versions du produit.



Création du projet BDTHEQUE dans TFS 2010

Le projet étant créé puis paramétré dans TFS 2010, l'équipe consacre un peu de temps pour formaliser les besoins et exprimer les principales exigences du projet BDTHEQUE. Afin de pouvoir ultérieurement piloter le projet par tableaux de bord, elle définit un découpage du projet en itérations (évolutions des fonctionnalités dans le temps) et en « area », associées aux différentes couches logiques de l'architecture retenue auparavant :





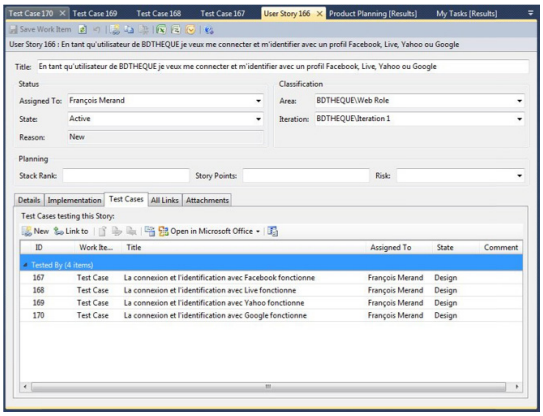
Ce découpage permettra de suivre l'évolution du projet plus finement aussi bien en phase de développement que de maintenance.

Elle décrit les scénarios d'usages de la solution dans des **Work Items** de type **User Stories**, qui sont traduit ensuite en tâches **(Tasks)** de développement par le chef de projet à destination des développeurs. Cette étape permet de constituer le **« Product Backlog »** qui contiendra l'ensemble des scénarios à produire pour finaliser le produit, ainsi que le premier **« Iteration Backlog »** pour démarrer le projet.

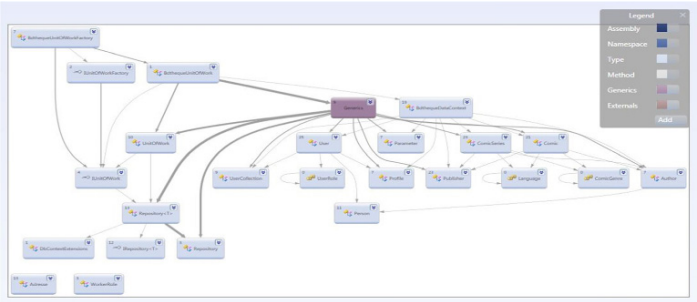
L'équipe BDTHEQUE attache une grande importance à la qualité du logiciel produit, aussi elle décide de mettre en place différents types de tests tout au long du développement :

- Des **tests unitaires**, associés à une couverture de code
- Des **tests fonctionnels**, qui seront automatisés
- Des **tests de montée en charge** de la future solution.

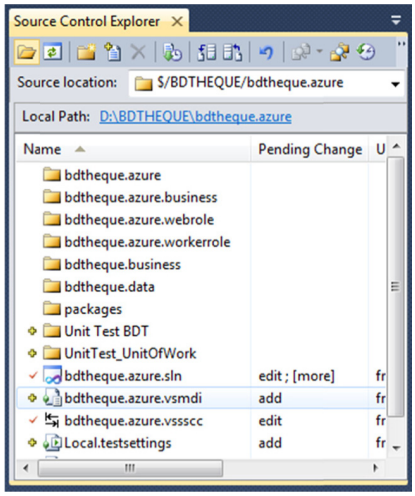
Afin de prendre en compte ces tests dès la conception du produit, l'équipe prend le temps de définir les différents tests fonctionnels (sous forme de campagne de tests) qui seront associés aux différentes exigences exprimées. L'objectif est de pouvoir suivre les exigences qui n'auraient pas été testées avant de partir en production.



Parmi les nombreuses fonctionnalités proposées, l'architecte de l'équipe en profite pour **valider et contrôler la cohérence** de son modèle grâce au **diagramme de dépendances :**



Enfin, la solution utilise le contrôle de source proposé par TFS 2010, qui permet à l'équipe de bien gérer les différentes versions, les mises à jour et les branches associées



2. Build & déploiement automatisés de la solution

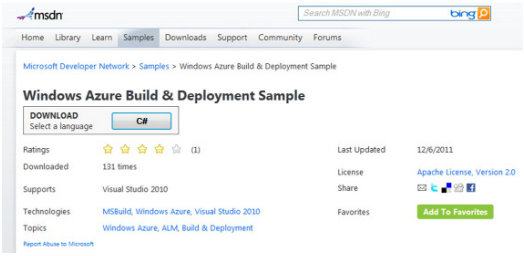
L'équipe BDTHEQUE se focalise maintenant sur l'automatisation du processus de build et de déploiement de la solution. Elle a testé dans le chapitre 2 le déploiement « manuelle » de son premier projet afin de valider la mise en ligne effective sur Azure de sa solution, mais à présent elle souhaite optimiser et automatiser une partie de ce processus.

Une fois le SDK Windows Azure installé sur les postes de développement, le déploiement des solutions est très simple. Le mode opératoire proposé est optimal pour un développeur seul. Les équipes projet préfèrent souvent activer un **« build »** puis déployer leur solution de façon centralisée, via un **« serveur de build »**. Cela leur permet d'être plus confortables dans ce processus build-déploiement, et surtout cela permet d'attacher des tâches complémentaires durant la phase de build, comme l'exécution de tests unitaires.

La version actuelle du SDK Azure, la 1.6, apporte de nouvelles fonctionnalités liées au déploiement afin de **faciliter la prise en compte des paramètres de publication et des informations de connexion** (tâches répétitives). Il n'est plus nécessaire désormais de fournir manuellement à chaque fois le code de votre abonnement, le nom du service hébergé et les certificats.

Pour permettre de gagner en efficacité, Microsoft propose un complément très simple à installer et configurer, à partir de cette URL :

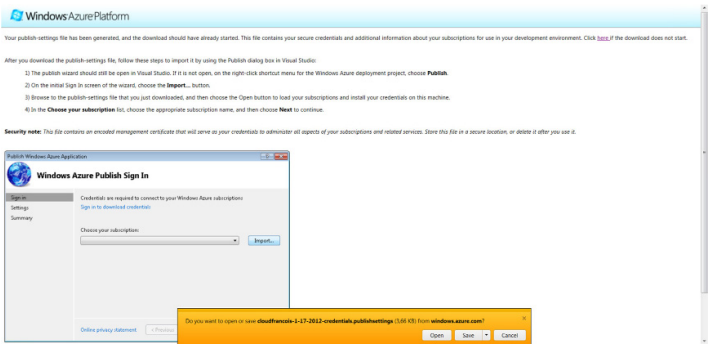
http://code.msdn.microsoft.com/windowsazure/Windows-Azure-Build-8cee065d



Page de téléchargement de l'exemple « build et déploiement Windows Azure » sur MSDN

Cet exemple contient des **scripts** et des **outils** qui simplifient le build et le déploiement centralisés des applications Windows Azure, en s'appuyant sur MSBuild et PowerShell. L'installation se fait en compilant le code fourni puis en suivant les quelques manipulations décrites sur la page web ci-dessus.

Ensuite, l'équipe suit le processus d'installation et elle se connecte sur la page https://windows.azure.com/download/publishprofile.aspx afin de récupérer sa configuration de publication automatiquement. Cette page, une fois authentifiée sur le portail Azure, génère un fichier contenant les informations sécurisées de connexion, ainsi que d'autres informations sur l'abonnement en cours, qui seront utilisées lors du déploiement des solutions Windows Azure depuis TFS 2010.



Page de récupération des informations de publication automatique

La poursuite du processus d'installation est ensuite très simple, depuis Visual Studio :

1) Il faut cliquer sur le menu "Publish" de la solution Azure dans Visual Studio

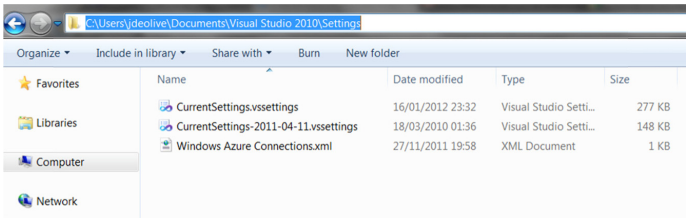
2) Sur l'écran initial (sign in), il faut cliquer sur le bouton « Import »

3) Sélectionner le fichier que vous venez de télécharger sur le portail ci-dessus,

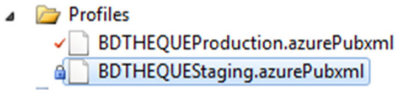
« cloudfrancois-1-17-2012-credentials.publishsettings » dans le cas de notre équipe BDTHEQUE.

4) Sélectionner ensuite votre abonnement et terminez la configuration classiquement de vos publications.

Outre l'installation du certificat sur le poste local, cette manipulation stocke les informations utiles à Visual studio dans le *« Windows Azure Connections.xml »* dans le dossier *%UserProfile%\Documents\Visual Studio 2010\Settings*. Il est conseillé **de supprimer le fichier récupéré .publishsettings**, ou de le stocker de façon sécurisée, car il contient le certificat et la clé privée qui donne accès aux compte Azure.



Une fois le paramétrage de la publication terminée dans Visual Studio, les informations utiles sont stockées dans un fichier au format XML, d'extension « .azurePubxml ». L'équipe BDTHEQUE possède deux configurations, une pour le site de production et une autre pour le site dit de « staging ».



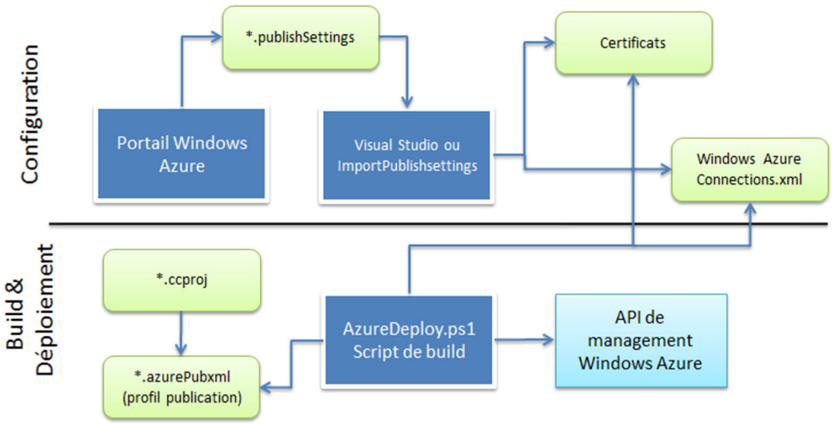

```xml
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <AzureCredentials>FMerand</AzureCredentials>
    <AzureHostedServiceName>bdtheque</AzureHostedServiceName>
    <AzureHostedServiceLabel>BDTHEQUE</AzureHostedServiceLabel>
    <AzureSlot>Staging</AzureSlot>
    <AzureEnableIntelliTrace>False</AzureEnableIntelliTrace>
    <AzureEnableProfiling>False</AzureEnableProfiling>
    <AzureEnableWebDeploy>False</AzureEnableWebDeploy>
    <AzureStorageAccountName>bdtheque</AzureStorageAccountName>
    <AzureStorageAccountLabel>bdtheque</AzureStorageAccountLabel>
    <AzureDeploymentLabel>bdtheque.azure</AzureDeploymentLabel>
    <AzureSolutionConfiguration>Release</AzureSolutionConfiguration>
    <AzureServiceConfiguration>Cloud</AzureServiceConfiguration>
    <AzureAppendTimestampToDeploymentLabel>True</AzureAppendTimestampToDeploymentLabel>
    <AzureAllowUpgrade>True</AzureAllowUpgrade>
    <AzureEnableRemoteDesktop>False</AzureEnableRemoteDesktop>
  </PropertyGroup>
</Project>
```

Contenu du fichier BDTHEQUEStaging.azurePubxml

La solution téléchargée par l'équipe comporte les éléments suivants (il ne faut pas oublier que Visual Studio n'est pas forcément installé sur le serveur de build) :

- Un fichier *« AzureDeploy.targets »*, qui est installé sur le serveur de build pour indiquer à MSBuild de préparer et déployer la solution sur Windows Azure
- L'outil *« ImportPublishSettings »* dont le rôle est d'importer un fichier ".publishsettings" sur le serveur de build
- Un script PowerShell *« AzureDeploy.ps1 »* qui s'appuie sur la librairie « AzurePublishHelpers »
- Une description TFS Build dont le rôle est de fournir les propriétés à MSBuild pour lancer le processus de build

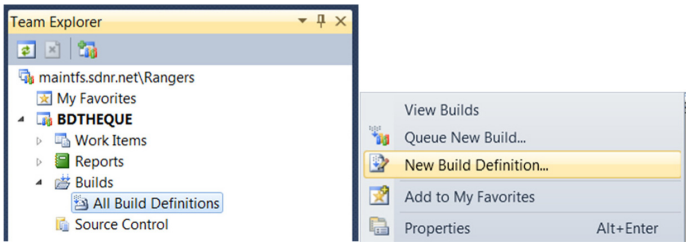
Le schéma de synthèse suivant repositionne tous les éléments de la solution :



Le fichier *« AzureDeploy.targets »* présenté ci-dessous doit être copié dans : « C:\Program Files\MSBuild\Microsoft\VisualStudio\v10.0\Windows Azure Tools\1.6\ImportAfter », pour être pris en compte automatiquement. Il exécute alors le script PowerShell en lui passant les 3 paramètres utiles.

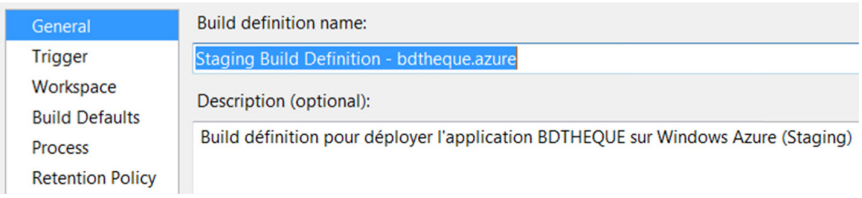
```
<Project ToolsVersion="4.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <PackageName>$(AssemblyName).cspkg</PackageName>
    <PackageForComputeEmulator>true</PackageForComputeEmulator>
  </PropertyGroup>
  <Target Name="AzureDeploy" AfterTargets="Build" DependsOnTargets="Publish" Condition="$(AzurePublishProfile)!=''">
    <Message Text="Executing target AzureDeploy from AzureDeploy.targets file"/>
    <Exec WorkingDirectory="$(MSBuildProjectDirectory)"
        Command="$(windir)\system32\WindowsPowerShell\v1.0\powershell.exe -f c:\builds\AzureDeploy.ps1 $(PublishDir)
        $(PackageName) &quot;Profiles\$(AzurePublishProfile)&quot;" />
  </Target>
</Project>
```

L'étape finale consiste maintenant pour l'équipe BDTHEQUE à configurer une première définition de build sur son serveur. Elle le fait à partir du menu « Builds » dans Team Explorer.

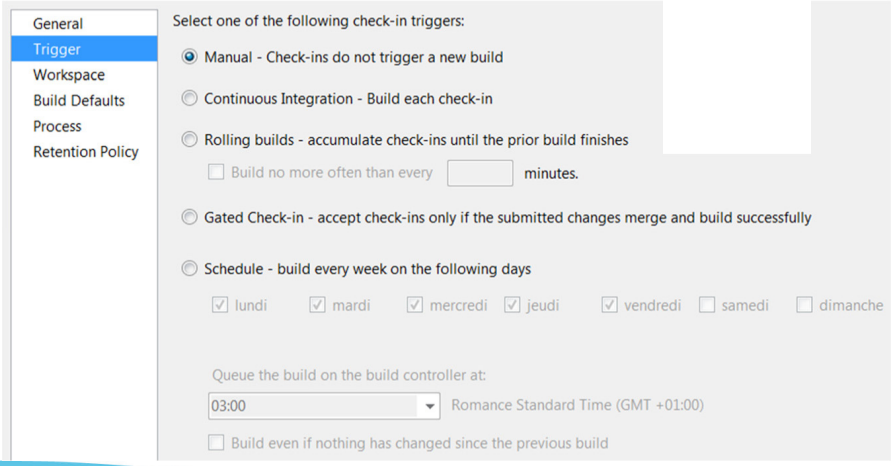


Création d'une nouvelle définition de build depuis Team Explorer

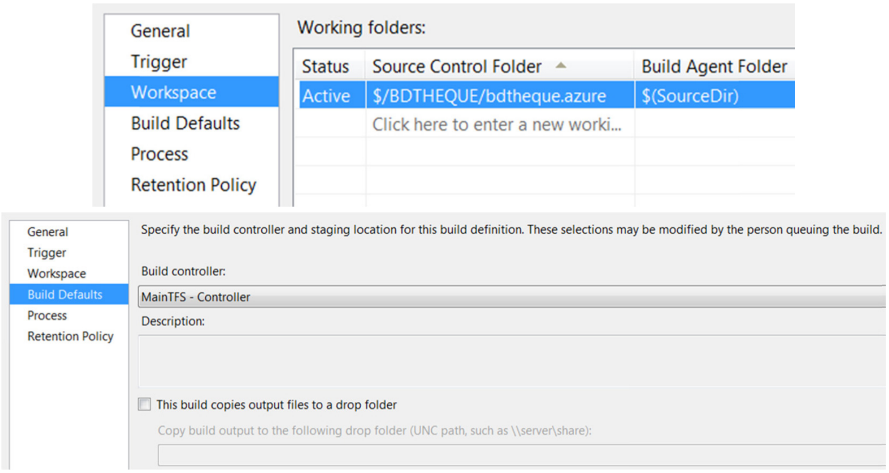
Il suffit alors de suivre les différents écrans proposés par l'assistant de configuration de build. Il faut commencer par donner un nom au build en cours de définition :



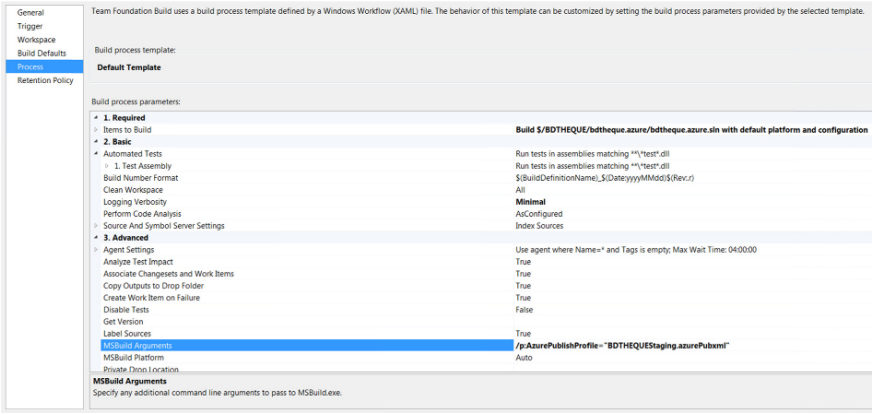
Puis l'équipe sélectionne quand le build doit se déclencher. Dans le cas de BDTHEQUE, elle sélectionne un build à la demande (manuel), ce qui lui semble plus adapté à sa situation actuelle, l'objectif étant de gagner en maturité et d'automatiser cette phase.



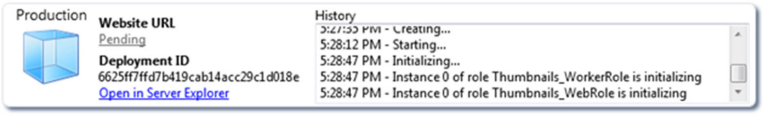
Elle précise ensuite où récupérer depuis le référentiel le code source à compiler, puis le contrôleur de build :



Enfin le plus important est de préciser dans l'onglet « Process » le paramètre « MSBuild Arguments » avec le nom du fichier « .azurePubxml » défini ci-dessus. En ce qui concerne l'équipe BDTHEQUE, elle publiera ainsi automatiquement via son serveur de build la solution sur ses serveurs de « staging » Windows Azure.



Voilà, une première étape franchie dans son processus ALM pour l'équipe BDTHEQUE, qui a maintenant à sa disposition un processus de build publiant automatiquement sa solution vers ses serveurs Windows Azure !



L'APPLICATION EST DÉPLOYÉE, QU'EN EST-IL DE DE LA MONTÉE EN CHARGE ?

L'élasticité proposée par la plateforme Windows Azure repose sur l'allocation de nouvelles VM ou la libération de VM non utilisées. Cette opération, bien que rapide, quelques minutes, demande une intervention manuelle. L'équipe BDTHEQUE souhaite à terme avoir une solution dite « auto-scalable ». En fait une application capable de s'auto-dimensionner par surveillance d'un certain nombre de paramètres (avec des garde-fous).

En complément de solutions du marché, Microsoft propose désormais une solution appelée **WASABi** ! (Windows Azure Integration Pack - Autoscaling Application Block). Disponible sur CodePlex, ce module est développé par l'équipe Microsoft Patterns & Practices qui s'occupe de l'Entreprise Library. Toute l'information utile se trouve sur la section Entlib de codeplex : http://entlib.codeplex.com/

La prochaine étape va maintenant consister à travailler sur la qualité du code, la qualité de la solution mise en ligne et aussi sa performance.

3. BDTHEQUE et les tests : la qualité omniprésente

L'équipe BDTHEQUE poursuit ses investissements autour de la mise en place d'une filière ALM lui apportant : qualité, flexibilité, automatisation, collaboration et efficacité. TFS 2010 offrant une panoplie complète d'outils aptes à répondre à ces exigences, elle décide de se focaliser sur la mise en place de tests :

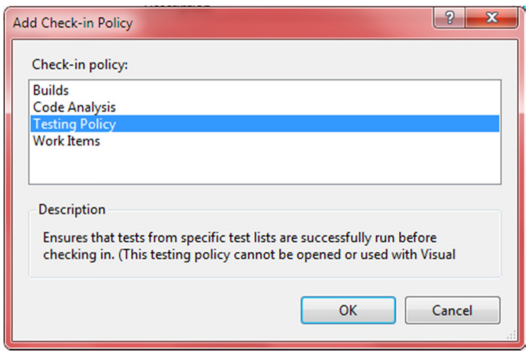
- Des **tests unitaires**, focalisés sur la qualité du code et mis en œuvre par les développeurs eux-mêmes. Ces tests sont couplés au module de couverture de code, afin de s'assurer qu'ils couvrent une part significative du code.

- Des **tests fonctionnels**, couvrant les scénarios (ou exigences) de la solution BDTHEQUE.

QUALITÉ DU CODE SOURCE : MISE EN PLACE DES TESTS UNITAIRES

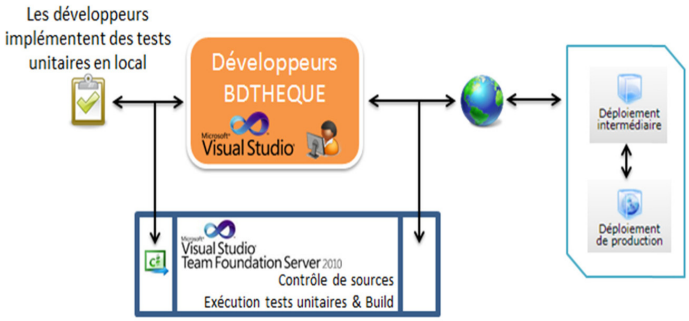
Les tests unitaires sont simples à comprendre : c'est du code qui teste que le code écrit par le développeur fournit les résultats attendus ! Il est donc impératif de s'appuyer pour cela sur un Framework de tests unitaires bien intégré à l'outil. Visual Studio propose MSUnit, et l'équipe BDTHEQUE décide d'utiliser cet outil pour des tests unitaire en local, sous contrôle de source.

Elle intègre donc ces tests dans sa chaine de production logicielle. A terme elle ajoutera une contrainte : pour pouvoir faire un « check-in » de son code source dans le référentiel, le développeur DEVRA avoir exécuté ses tests unitaires :

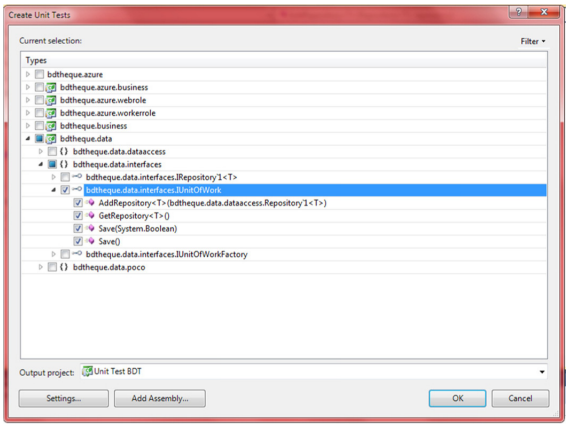


Check-in policy pour les tests unitaires

Le mécanisme de build définit dans les pages précédentes prends alors le relai pour publier le code sur le serveur intermédiaire Windows Azure :



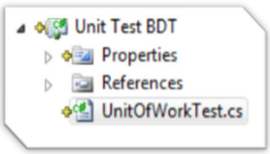
D'un point de vue pratique, les tests unitaires sont simples à implémenter dans Visual Studio 2010 et nativement disponibles : le développeur sélectionne dans son code source les fonctionnalités à tester par un clic droit sur la méthode à tester et l'outil fait le nécessaire :



Création d'un test unitaire depuis Visual Studio 2010

Visual Studio crée ensuite un projet de tests unitaires (possibilité de le mettre sous contrôle de sources). Il reste à préciser les valeurs pour les tests dans les fonctions créées et les valeurs attendues en retour.

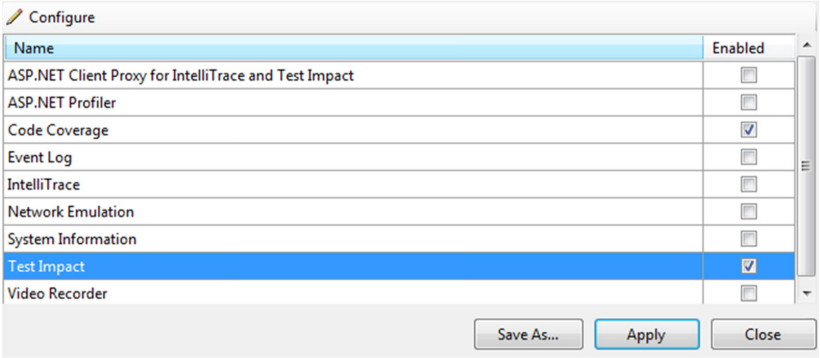
```
Additional test attributes

/// <summary>
///A test for UnitOfWork Constructor
///</summary>
[TestMethod()]
public void UnitOfWorkConstructorTest()
{
    string connectionString = string.Empty; // TODO: Initialize to an appropriate value
    UnitOfWork target = new UnitOfWork(connectionString);
    Assert.Inconclusive("TODO: Implement code to verify target");
}
```



Test unitaire généré et projet de tests généré automatiquement

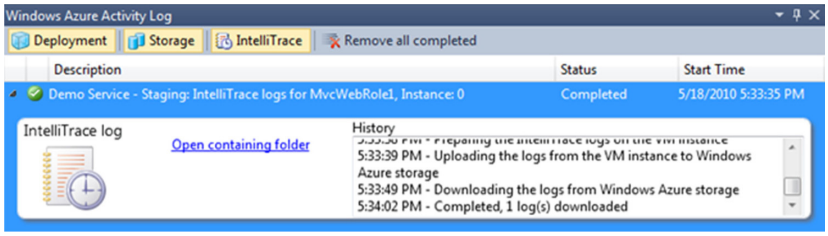
Tout cela est entièrement paramétrable depuis le menu « Test – Edit Test Settings ». L'équipe BDTHEQUE choisi d'activer la couverture de code, ainsi que les Tests impactés, afin d'optimiser l'impact des tests unitaires dans ses campagnes de tests (lors de la modification de code source, cette option permet de ne rejouer que les tests impactés par ce

changement, évitant ainsi de rejouer l'intégralité des tests.



Configuration des tests

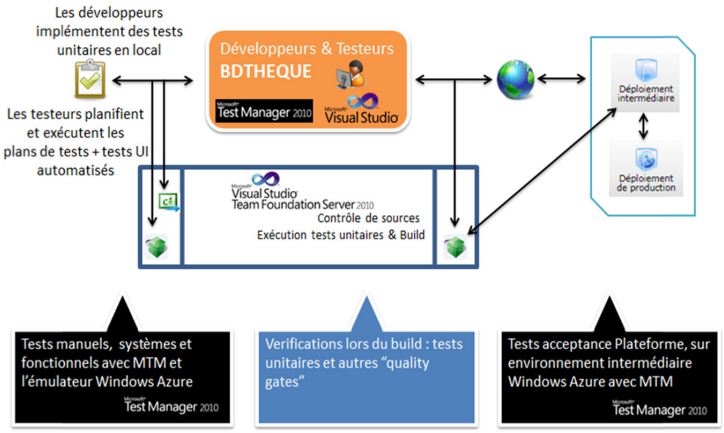
En complément, il est possible d'activer le mode debug en local via l'émulateur, et d'activer « IntelliTrace » sur les instances Windows Azure en cochant la bonne case lors du déploiement.



Récupération des log IntelliTrace en asynchrone depuis Windows Azure

QUALITÉ DE L'APPLICATION : TESTS FONCTIONNELS

Cette fois, c'est une équipe spécialisée dans le test qui intervient. Elle va travailler en collaboration avec l'équipe de développement, en s'appuyant sur un outil complémentaire et spécialisé : **Microsoft Test Manager (MTM)**. Le scénario de mise en place complète donc le précédent, puisque cette équipe va planifier et préparer en local ses campagnes de test avec MTM. Elle aura la possibilité de dérouler ses tests en local, via l'émulateur, mais aussi sur l'environnement intermédiaire (staging) hébergé sur Windows Azure.

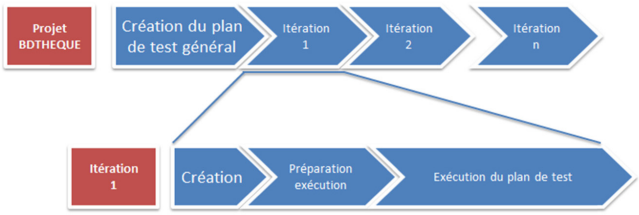


Circuit des tests pour Windows Azure avec MTM

Le challenge dans ce scénario consiste à bien répartir et exécuter ses tests au bon endroit du fait que l'environnement intermédiaire de Windows Azure est « facturé » à l'utilisation. Pour optimiser ces coûts, il existe deux autres environnements pour exécuter ses tests : celui dédié aux testeurs pour les tests dits « manuels » et celui du moteur de build pour les tests « automatisés ».

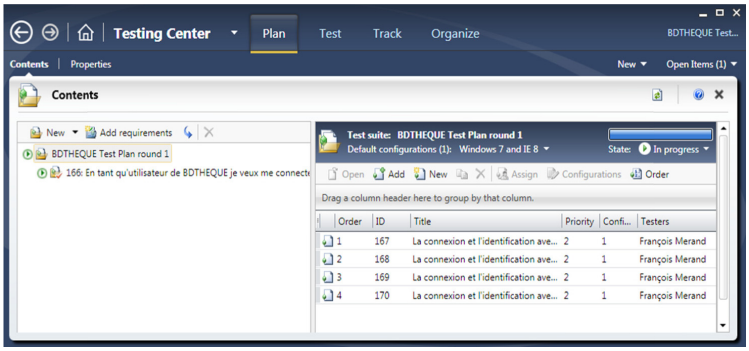
C'est le scénario retenu par l'équipe BDTHEQUE, qui effectue tous ses tests en local avec MTM, automatise tout ce qui peut l'être pour conserver une approche « industrielle » (y compris l'automatisation des tests UI, des tests unitaires via le moteur de build). Mais elle ne fait pas l'économie de jouer à chaque mise en production les mêmes plans de tests sur la plateforme intermédiaire.

La préparation et la configuration des campagnes de test suit une logique habituelle, découpée en plusieurs lots (itérations), calquées si possible sur les itérations même du projet BDTHEQUE. En termes de pilotage il devient possible de tester les exigences, et de suivre celles qui n'ont pas été testées.



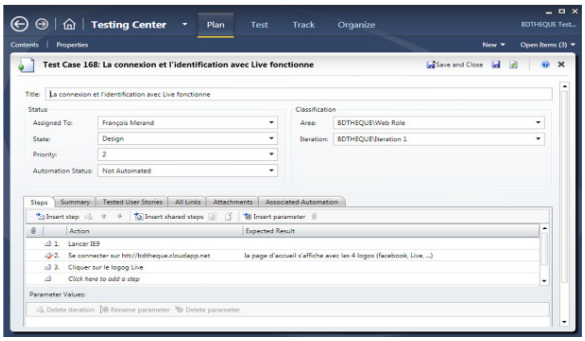
Elaboration de la campagne de tests BDTHEQUE

D'un point de vue pratique, l'équipe en charge des plans de tests pour BDTHEQUE exécute **Microsoft Test Manager :**



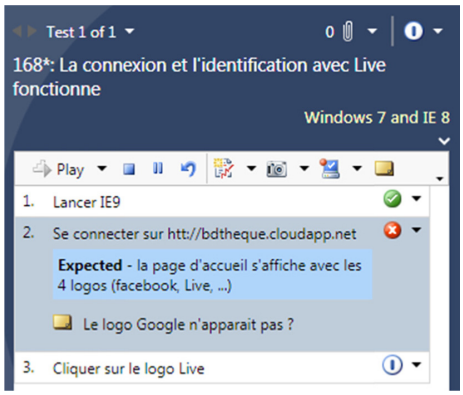
MTM en action : élaboration des plans de tests

Planification : Cette étape permet aux testeurs de retrouver les exigences stockées dans TFS 2010 et de bâtir les plans de tests détaillés. (Chaque suite et chaque test est en fait une fiche rattachée à un Work Item, avec toutes les caractéristiques utiles : responsable, priorité, etc.).



Test Manager : phase de planification des campagnes de tests

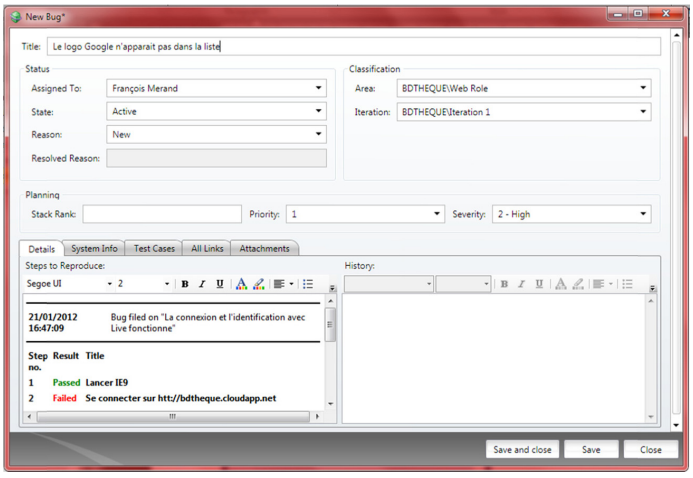
Les propriétés permettent de préciser les informations complémentaires collectées lors des tests : variables d'environnement, informations système, enregistrement d'une vidéo, etc.



Exécution des tests : cette phase permet de jouer les plans de tests. Si une anomalie est découverte, elle est remontée et saisie dans une fiche Work Item de type BUG et affectée à un développeur pour correction.

Ce déroulement est possible en local dans un premier temps, mais aussi au final sur la cible hébergée sur Windows Azure.

Il est possible de configurer des étapes communes ou de stocker des paramètres pour optimiser l'exécution de ces tests.



Une anomalie est remontée via MTM lors de l'exécution d'un test.
Cette fiche BUG sera lue par le développeur depuis son Visual Studio.
Sévérité et Priorité seront adaptées en fonction de la plateforme de tests : local ou sur Windows Azure !

Pour aller plus loin

L'équipe BDTHEQUE est maintenant bien armée pour entreprendre sereinement le développement de sa solution. L'automatisation du déploiement sur Windows Azure via le moteur de build, la mise en place des tests en tenant compte des spécificités de la plateforme cible lui permette d'entamer le processus d'amélioration continue associé à une démarche ALM de façon optimale.

Il reste d'autres pistes à explorer, notamment sur le plan des tests avec les tests de montée en charge, tests web, etc. Confiante dans ses choix, dans la motivation de ses équipes et la robustesse de ses outils, l'équipe BDTHEQUE aborde ce nouveau défi avec sérénité !

Un projet échoue au début, jamais à la fin ! Si vous échouez dans la planification… alors vous planifiez l'échec !

Sogeti est l'un des leaders des services informatiques et d'ingénierie de proximité, spécialisé dans la gestion des applicatifs et des infrastructures (application and infrastructure management), le conseil en technologies (high-tech engineering) et le testing.

De dimension internationale, le groupe apporte à ses clients des réponses simples, concrètes et efficaces, avec une gamme complète de solutions sur mesure et innovantes. Sur le territoire national, Sogeti France accompagne ses clients localement, grâce à une organisation décentralisée, souple et réactive.

Microsoft

SOGETI