INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Cloud Computing Sobre a Plataforma Windows Azure

PEDRO MIGUEL FERREIRA DUQUE GONÇALVES
Setembro de 2012

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

CLOUD COMPUTING SOBRE A PLATAFORMA WINDOWS AZURE

Pedro Miguel Ferreira Duque Gonçalves

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientador: Doutor Paulo Gandra de Sousa

Coorientador: Mestre António Rocha

Júri:

Presidente:

Vogais:

Porto, Setembro 2012

Dedicatória

Este trabalho é dedicado à minha família, especialmente aos meus pais e namorada.

Resumo

A utilização massiva da internet e dos serviços que oferece por parte do utilizador final potencia a evolução dos mesmos, motivando as empresas a apostarem no desenvolvimento deste tipo de soluções. Requisitos como o poder de computação, flexibilidade e escalabilidade tornam-se cada vez mais indissociáveis do desenvolvimento aplicacional, o que leva ao surgimento de paradigmas como o de *Cloud Computing*.

É neste âmbito que surge o presente trabalho. Com o objetivo de estudar o paradigma de *Cloud Computing* inicia-se um estudo sobre esta temática, onde é detalhado o seu conceito, a sua evolução histórica e comparados os diferentes tipos de implementações que suporta. O estudo detalha posteriormente a plataforma Azure, sendo analisada a sua topologia e arquitetura, detalhando-se os seus componentes e a forma como esta mitiga alguns dos problemas mencionados.

Com o conhecimento teórico é desenvolvido um protótipo prático sobre esta plataforma, em que se exploram algumas das particularidades da topologia e se interage com as principais redes sociais.

O estudo culmina com uma análise sobre os benefícios e desvantagens do Azure e através de um levantamento das necessidades da empresa, determinam-se as oportunidades que a utilização da plataforma poderá proporcionar.

Palavras-chave: Cloud Computing, Windows Azure, Redes Sociais.

Abstract

The massive use of the internet and the services it provides enables their progress, motivating companies to develop this type of solutions. Application requirements such as computing power, flexibility and scalability become key issues in application development, which leads to the emergence of paradigms such as Cloud Computing.

This was the context of this work. In order to study Cloud's Computing paradigm, its concept and historical evolution is analyzed, as well as the different types of implementations it provides. The study also analyzes Azure's platform, its topology and architecture, and how it mitigates some of the problems mentioned.

With the theoretical knowledge a prototype over Azure's platform is developed, exploring some of the specifications of the topology and interacting with the main social networks.

The study ends with an analysis over the benefits and drawbacks of Azure and the opportunities that the use of the platform may provide.

Keywords: Cloud Computing, Windows Azure, Social Networks.

Agradecimentos

Começo por agradecer à minha família, em especial aos meus pais, por sempre conseguirem reunir as condições necessárias para chegar até aqui e por toda a compreensão e incentivo que sempre me deram.

Agradeço também à minha namorada por todo o apoio e força que sempre me conseguiu dar.

À B-Simple e à Ambidata, pela oportunidade de desenvolvimento de um projeto na área de *Cloud Computing*, assim como a toda a equipa, que sempre me incentivou a desenvolver este estudo, mostrando-se sempre disponível para auxiliar no necessário.

Aos meus orientadores Doutor Paulo Gandra de Sousa e Eng.º António Rocha por toda a ajuda, dedicação e disponibilidade.

E por fim, gostaria de agradecer aos meus amigos, sem os quais não teria atingido muitas das coisas que até hoje concretizei.

A todos eles, o meu muito obrigado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

API	Application Programming Interface
BD	Base de Dados
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
E/S	Entrada / Saída
FIFO	First In First Out
HaaS	Hardware as a Service
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IE	Internet Explorer
IIS	Internet Information Services
IDE	Integrated Development Environment
LIMS	Laboratory Information Management System
NTFS	New Technology File System
PaaS	Platform as a Service
PMA	Procriação Medicamente Assistida
POS	Point Of Sale
QoS	Quality of Service
SaaS	Software as a Service

SLA	Service-Level Agreement
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer
TDS	Tabular Data Stream
T-SQL	Transact-SQL
VHD	Virtual Hard Drive
VM	Virtual Machine
WBS	Work Breakdown Structure
XML	Extensible Markup Language

1 Introdução

1.1 Problema

Na sociedade atual, o mercado é volátil modificando-se a cada dia que passa. A constante inovação nas diversas áreas de negócio potencia a evolução das soluções que tentam permanentemente dar resposta às necessidades impostas pelo mercado.

O desenvolvimento da *web* e dos serviços que oferece é o espelho desta evolução. A crescente procura deste tipo de soluções por parte do utilizador final motiva as empresas a investirem nesta área uma vez que se assume, cada vez mais, como um potenciador de negócio. Este dinamismo do mercado, conjugado com o aumento da procura deste tipo de serviços, levanta alguns problemas no momento da conceção de soluções.

O primeiro problema surge a nível infraestrutural. Tipicamente, aquando da estimativa de utilização de determinado serviço, existem várias questões às quais se tem que dar resposta: Quantas máquinas serão necessárias para correr o serviço? Como se vão interligar? Que tipo de mecanismos de otimização deverão ser implementados (distribuição de carga, *backup*, tolerância a falhas,..)?..

O segundo problema surge a nível do desenvolvimento aplicacional. Um cenário possível é o desenvolvimento de uma aplicação que suporte pedidos de uma dezena de utilizadores, outro completamente diferente, é o desenvolvimento de uma solução que suporte pedidos de milhares de utilizadores a acederem concorrentemente a um serviço.

Assim, a capacidade da plataforma e a topologia da aplicação, assumem-se como questões fulcrais na conceção de soluções. Tanto a plataforma como a aplicação terão que ser flexíveis

ao ponto de se poderem adaptar à variação da carga à qual estão sujeitos. Os recursos computacionais poderão aumentar ou diminuir significativamente, nunca podendo esta variação comprometer a resposta do serviço, tendo este que se manter acessível e disponível.

A partir dos problemas genéricos mencionados, facilmente se identificam aqueles que claramente se adaptam à realidade da empresa na qual vai ser desenvolvido este trabalho (B-Simple[1]):

- Possuir soluções que possam ser disponibilizadas como serviços (*Software as a Service* (SaaS) [26]) uma vez que, cada vez mais, os clientes não pretendem efetuar elevados investimentos iniciais;

- Questões financeiras da própria empresa: O fato da incerteza relativamente ao número de utilizadores neste tipo de soluções, aliada ao fato destas aplicações poderem ser usadas por períodos de tempo curtos (como é o caso deste *case study*), torna praticamente impossível adquirir infraestruturas que as suportassem.

O projeto a desenvolver insere-se no departamento de *Special Projects* e vai servir como entrada da empresa no desenvolvimento de soluções para a *cloud,* acompanhando as novas tecnologias lançadas pela Microsoft (sendo a B-Simple *Microsoft Gold Partner*). Desta forma, a empresa pretende manter a imagem de associação a "tecnologias emergentes", alargando a oferta do departamento de *Special Projects* a este tipo de soluções (SaaS). Para além de promover o crescimento da empresa, uma vez que passará a disponibilizar soluções de C*loud Computing* a grandes clientes que possuam elevadas necessidades de escalabilidade e elasticidade nas suas aplicações, a B-Simple pretende tirar partido deste tipo de soluções para reestruturar algumas das aplicações que possui atualmente.

[1] Na fase final do desenvolvimento deste estudo, toda a área de negócio em que se enquadrava este projeto foi transitada da B-Simple (www.b-simple.pt) para uma nova empresa (Ambidata®), também parceira da Microsoft, mantendo-se todos os objetivos que motivaram o desenvolvimento deste trabalho (www.ambidata.pt).

Expostos os problemas genéricos e específicos que motivaram o desenvolvimento deste trabalho, o problema que esta tese tenta abordar pode-se então resumir a:

> *Que oportunidades traria à empresa o desenvolvimento de aplicações segundo o paradigma de Cloud Computing sobre a plataforma Windows Azure?*

Partindo desta questão definiram-se os objetivos concretos que serão apresentados na secção seguinte.

1.2 Objetivos

O objetivo principal para o desenvolvimento deste trabalho foi o estudo do conceito de *Cloud Computing,* de forma a aprofundar os conhecimentos nesta área e avaliar benefícios para a empresa da adoção desse modelo. Este objetivo genérico pode ser subdividido num conjunto de objetivos específicos, que visam contribuir para a concretização do principal:

- Estudar o conceito de *Cloud Computing* e os tipos de soluções e aplicações existentes;
- Analisar concretamente a plataforma Azure e a forma como esta mitiga os típicos problemas de carga, escalabilidade e concorrência garantindo fiabilidade, elasticidade e elevados níveis de QoS (Quality of Service);
- Criar uma aplicação real e colocá-la na *cloud*, de forma a desenvolver um protótipo segundo este paradigma e analisar os resultados obtidos e benefícios/desvantagens da sua utilização;
- Analisar, mediante o contexto atual da empresa, o impacto que a adoção do Azure traria.

1.3 Enquadramento

1.3.1 Empresa

A empresa na qual o projeto foi desenvolvido (B-Simple) foi criada em 2002 desenvolvendo desde então, soluções de *software* para diferentes atividades de negócio, sendo constituída por seis departamentos de desenvolvimento:

- *Business* – Desenvolvimento de soluções para gestão de empresas;
- *Quality* – Desenvolvimento de soluções relacionadas com sistemas de qualidade em organizações;
- *Healthcare* – Desenvolvimento de soluções para a área da saúde;
- *Internet* – Desenvolvimento de soluções para Internet;
- *Mobile* – Desenvolvimento de soluções para a área da mobilidade;
- *Special projects* – Desenvolvimento de soluções de sistemas de informação de raiz à medida das necessidades do cliente.

Sendo *Microsoft Gold Partner* todas as aplicações que desenvolve utilizam as mais recentes tecnologias de desenvolvimento da Microsoft.

1.3.2 Projeto

Em Março de 2011 a Microsoft preparava o lançamento do seu novo *browser* Internet Explorer 9 (IE9) e necessitava de um veículo para divulgar e promover o contacto do público com o produto em Portugal. Surgiu assim a ideia de criar um concurso *online* para a promoção do IE9, antecedendo o seu lançamento oficial.

A ideia consistia num registo na página do concurso (através da conta do Facebook) e na divulgação do produto a amigos (através das redes sociais), tendo como objetivo obter-se o maior número de seguidores na página. No final do concurso, o concorrente com mais seguidores ganharia uma viagem e os amigos que o auxiliaram prémios de valor inferior. A integração das principais redes sociais na solução (Facebook e Twitter) era um dos requisitos

de maior importância, pois iria permitir uma divulgação eficaz do produto entre redes de amigos. Para participar no concurso, todos os concorrentes teriam que estar a utilizar o *browser* IE9, o que implicaria o seu *download*.

O concurso estaria apenas um mês *online*, terminando no dia do lançamento oficial do IE9. Como se tratava de uma solução concebida para uma grande empresa multinacional (Microsoft) aliado ao fato de ter prémios aliciantes, o sistema teria que estar preparado para suportar uma elevada carga de utilização. Por outro lado, a adesão do público era incerta, uma vez que dependeria da promoção por parte da Microsoft, face a outras ações de marketing em vigor e da predisposição dos participantes em instalar o IE9 para participar no concurso.

Mediante este cenário, a aplicação demonstrava grande potencial para correr na *Cloud*, nomeadamente devido ao seu tempo de vida curto e ao nível de carga desconhecido e eventualmente não linear, ficando assim estabelecido que se utilizaria a plataforma *Azure*.

1.3.3 Tecnologias Utilizadas

As tecnologias utilizadas para a realização deste projeto seguiram, dentro do possível, a linha de ferramentas utilizadas no desenvolvimento dos produtos atuais da empresa, de forma a simplificar a entrada na *Cloud* e futuras manutenções do *software*:

- *Microsoft Visual Studio 2010* utilizado como IDE (Integrated Development Environment) recorrendo-se à linguagem de programação VB.Net para o desenvolvimento aplicacional;
- *Microsoft SQL Server 2008 R2* para o suporte de BDs (Bases de Dados) a correr na *Cloud* (SQL Azure);
- *Windows Azure SDK 1.3* para simulação da plataforma da *Cloud* localmente, com ferramentas para desenvolvimento e testes aplicacionais;
- *Microsoft Visual SourceSafe* como sistema de cópias de segurança e controlo de versões de ficheiros;

- *Celebrata Cloud Storage Studio* e *Azure Storage Explorer* para exploração de conteúdo da conta de armazenamento Azure;
- *SQL Azure Migration Wizard* para migração da base de dados de forma integrada para SQL Azure.

1.4 Âmbito do Projeto

O tema do projeto "*Cloud Computing* sobre a plataforma Windows Azure" define claramente o âmbito deste trabalho. Estudar-se-á o paradigma de *Cloud Computing* e os diferentes tipos de implementações que suporta, porém será o Windows Azure [10] que assumirá uma posição de destaque relativamente às restantes soluções referenciadas ao longo do trabalho. É sobre esta tecnologia que o estudo irá incidir com mais detalhe, evidenciando-se as características principais da plataforma e dos componentes que a constituem. Será também desenvolvida uma aplicação prática sobre esta tecnologia, uma vez que a empresa na qual será desenvolvida é parceira da Microsoft, utilizando preferencialmente as suas ferramentas no desenvolvimento das soluções que comercializa. A parte prática constituirá uma pequena aplicação com o intuito de, correndo na *Cloud* da Microsoft, ser utilizada por um vasto número de utilizadores intensivamente, durante um determinado período de tempo. Pretende-se com este estudo avaliar as vantagens para empresa na adoção deste modelo para o desenvolvimento dos seus produtos.

1.5 Organização do documento

O documento encontra-se dividido em seis capítulos principais, que pretendem abordar sequencialmente todas as fases do estudo, desde a sua contextualização passando pelo estudo dos conceitos fundamentais, pelo protótipo prático, até à análise final do trabalho e suas conclusões. Descrevem-se de seguida as diferentes secções do documento.

No primeiro capítulo (Introdução) é contextualizado todo o trabalho que irá ser realizado, em que se enuncia o problema e respetivos objetivos, o seu enquadramento e o âmbito do projeto.

No segundo capítulo (Cloud Computing) é estudado o paradigma de *cloud computing*, mais concretamente a sua evolução histórica, o conceito e as suas distintas implementações, sendo analisadas as suas principais diferenças.

No terceiro capítulo (Windows Azure) é feita uma análise relativamente à plataforma Azure, sendo estudada a sua topologia e arquitetura detalhando-se os principais componentes que a constituem.

No quarto capítulo (Protótipo de estudo) é descrito o processo de desenvolvimento de um protótipo para Azure, sendo abordadas algumas particularidades relativas ao desenvolvimento para esta plataforma e detalhados processos de integração com as principais redes sociais.

No quinto capítulo (Análise no contexto da empresa) é efetuado um estudo relativamente às aplicações que a empresa possui, explorando-se oportunidades que surgiriam com a migração das soluções para Azure e o impacto que traria a nível de mercado e recursos humanos da empresa.

No sexto capítulo (Conclusões) é analisado o trabalho desenvolvido, avaliando-se o cumprimento dos objetivos propostos, as contribuições do trabalho para a empresa, as suas limitações e trabalho futuro.

2 Cloud Computing

2.1 Introdução ao cloud computing

2.1.1 Conceito

Cloud Computing constitui uma tecnologia que tira partido da maturidade da internet, das aplicações *web* e da interoperabilidade dos sistemas de computação atuais, para disponibilizar *hardware* e *software* como serviço.

Refere-se a um conceito de publicação/consumo de serviços via *web*, que engloba as aplicações que constituem estes serviços e o *hardware*/*software* existente nos *datacenters* que possibilitam o seu funcionamento [2]. De acordo com este modelo, os recursos computacionais são disponibilizados via internet permitindo uma rápida adaptação da infraestrutura face às necessidades, evitando-se elevados investimentos iniciais em recursos, o que tipicamente sucede com as arquiteturas de *hosting* locais.

O *cloud computing* recorre para isto a mecanismos de virtualização, de forma a criar um nível de abstração face às múltiplas camadas de *hardware*/*software* que implementa, desacoplando os recursos fornecidos da complexidade inerente aos mesmos.

Este mecanismo possibilita que recursos computacionais (CPU, espaço de disco, velocidade de acesso) sejam vendidos de acordo com as necessidades aplicacionais dos serviços, sendo apenas taxados os recursos requeridos (conceito de *Utility Computing* [2]). O regime de

pagamento deste tipo de soluções (*pay-per-use*), ao contrário de soluções de *hosting* existentes, é também inovador uma vez que não pressupõe um aluguer mensal, mas sim o pagamento dos recursos efetivamente consumidos, sendo igual o aluguer de um servidor durante 1000 horas, como de 1000 servidores durante uma hora.

Através da virtualização de recursos computacionais, os fornecedores deste tipo de soluções têm a capacidade de oferecer computação elástica, na qual o número e velocidade de processamento dos servidores varia de acordo com as necessidades do serviço. A par desta característica, também o armazenamento dinâmico é outra funcionalidade oferecida com múltiplos tipos de armazenamento de acordo com os dados a armazenar. Estes dados são replicados pelos múltiplos servidores, sendo garantida a sua durabilidade, mesmo em caso de falha.

O *Cloud Computing* constitui assim uma boa solução para múltiplas organizações (ex: *startups* com baixos recursos financeiros iniciais, empresas com elevada carga aplicacional,..) e para diferentes tipos de serviços: aplicações com elevadas necessidades de escalabilidade; com carga variável; com tempo de vida incerto e/ou que necessitem de elevado poder de computação para processamento de dados (cálculos, simulações, renderizações,..);.. Serviços como o Gmail, Facebook e Amazon, baseiam-se neste tipo de soluções para dar uma resposta eficaz à carga que diariamente estão expostos.

As soluções baseadas em *cloud computing* são tipicamente flexíveis (permitindo uma simples adaptação dos recursos face às necessidades), escaláveis, redundantes e possuem mecanismos de balanceamento de carga e tolerância a falhas [6]. Em termos de QoS, os fornecedores de serviços de *cloud computing* garantem a fiabilidade do serviço através de SLA's (*Service-Level Agreements*), com rácios de *uptime* que tipicamente rondam os 100%.

Outra das vantagens inerentes à utilização deste tipo de soluções é a redução dos custos/tempo gasto com manutenção de *hardware*, uma vez que esta passa a ser da responsabilidade do fornecedor do serviço. A carga de tempo gasto com a administração de infraestrutura é significativamente reduzida, libertando a equipa para a execução de tarefas que poderão trazer um valor acrescentado superior para a organização (novos desenvolvimentos, testes, promoção do produto,..).

2.1.2 Enquadramento histórico

A expressão *Cloud Computing* é relativamente atual, embora muitos dos conceitos que implementa possam ser revistos em abordagens no passado. A primeira oferta baseada no conceito de *Cloud Computing* surgiu em 2006 [2], com o lançamento da versão beta da *Elastic Compute Cloud* da Amazon (EC2) [21].

Com esta solução, a Amazon fornecia aos seus clientes a capacidade de criar e configurar instâncias virtuais via *Web Service*, de forma a adaptar a sua capacidade computacional à capacidade requerida pelas aplicações a suportar. Flexibilidade, elasticidade e *pay-per-use* eram alguns dos conceitos que melhor adjetivavam esta solução. Porém, conforme referido, muitas abordagens anteriores ao EC2 já adotavam alguns conceitos inerentes à *cloud*.

Uma destas abordagens baseou-se no conceito de obter elevado poder computacional através da interligação de múltiplos computadores de diferentes domínios, constituindo a abordagem de computação em grelha ("Grid Computing") [2].

Esta abordagem tinha como objetivo final levar a cabo tarefas que envolviam elevado processamento em paralelo, como cálculos científicos e simulações (climatéricas, económicas,..) [2]. Estas tarefas complexas eram divididas em subtarefas, sendo distribuídas posteriormente pelos diversos nós computacionais. A computação em grelha baseava-se assim num conceito de computador virtual distribuído de alto desempenho, utilizado para o processamento *batch* de tarefas computacionais. Contrastava com o conceito de *Cloud Computing* devido à inexistência de interatividade com o utilizador final e ao fato da oferta de recursos computacionais virtuais não ser fornecida a múltiplos utilizadores em simultâneo. Uma das empresas que iniciou a abordagem de diversos aspetos que atualmente se associam à *cloud* foi a IBM. Com a existência exclusiva no passado dos extremamente caros *mainframes* [22], a IBM vendia capacidade de processamento partilhando-a com múltiplos utilizadores e organizações (*utility-computing*), sendo utilizadas técnicas de virtualização nestes sistemas há décadas [2].

Também o conceito *autonomic-computing* [23] foi criado pela IBM [2]. Este conceito refere-se à capacidade dos sistemas computacionais serem geridos por si mesmo, de forma a mitigar problemas relacionados com o seu crescimento e respetivo aumento da complexidade de gestão. Num sistema autónomo, o operador humano não controla diretamente o sistema,

apenas define políticas de alto nível que constituem as linhas orientadoras de tomada de decisão por parte deste.

Para se gerirem infraestruturas com a magnitude das de *Cloud Computing* (também designadas como *Fabric*), é praticamente impensável não se recorrer a estes princípios.

Mover cargas de trabalho de um nó computacional para outro de forma transparente, inicializar imagens de máquinas (ex: Amazon EC2) ou instâncias (ex: Windows Azure) pré-configuradas, replicar dados e voltar a executar tarefas, são alguns exemplos de mecanismos presentes na *Fabric* que podem ser vistos à luz dos princípios de computação autónoma.

Os princípios de *hosting* [24] / *outsourcing* [25] existentes na *cloud* são também conceitos tradicionais que tipicamente levam à migração de aplicações e serviços da infraestrutura da empresa para uma outra, gerida e operada através de um fornecedor externo e mediada através de SLA´s (ex: *backups*, *renting* de servidores,..).

Porém, o conceito tradicional de *hosting* difere do da *cloud*. Como primeiro aspeto diferenciador, pode ser salientado o envio de aplicações para a infraestrutura externa e a sua gestão. No caso do *hosting* tradicional existe um conjunto de regras e procedimentos para o fazer, enquanto a abordagem da *cloud* é mais eficiente e simplista uma vez que o *developer* pode gerir, por si só, toda a infraestrutura na qual está a trabalhar, dotando este processo de enorme flexibilidade e rapidez.

Outro aspeto já referido mas extremamente inovador, consiste no pagamento dos serviços de *Cloud Computing* comparativamente com os de *hosting*. Contrariamente aos serviços de *hosting* tradicionais, o regime de pagamento das soluções de *Cloud Computing* não pressupõe um aluguer mensal, apenas o pagamento dos recursos efetivamente consumidos, potenciando uma redução nos gastos operacionais.

Pode-se então depreender que o conceito *Cloud Computing*, embora inovador, não se encontra completamente desligado do passado. Baseia-se em algumas abordagens já existentes, tirando partido das suas vantagens e mitigando os aspetos menos positivos destas. Surge como uma tecnologia que cria um novo conceito de computação, marcando a sucessão de alguns conceitos presentes no passado.

2.2 Tipos de Implementações

As ofertas de *Cloud Computing* são múltiplas e variadas. Existem várias empresas a fornecer soluções diferentes, embora todas estas possam ser categorizadas de acordo com o tipo de serviço oferecido. O nível de abstração e granularidade permitem classificar os três principais tipos de implementações de *Cloud Computing*, identificadas na Figura 1: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS).

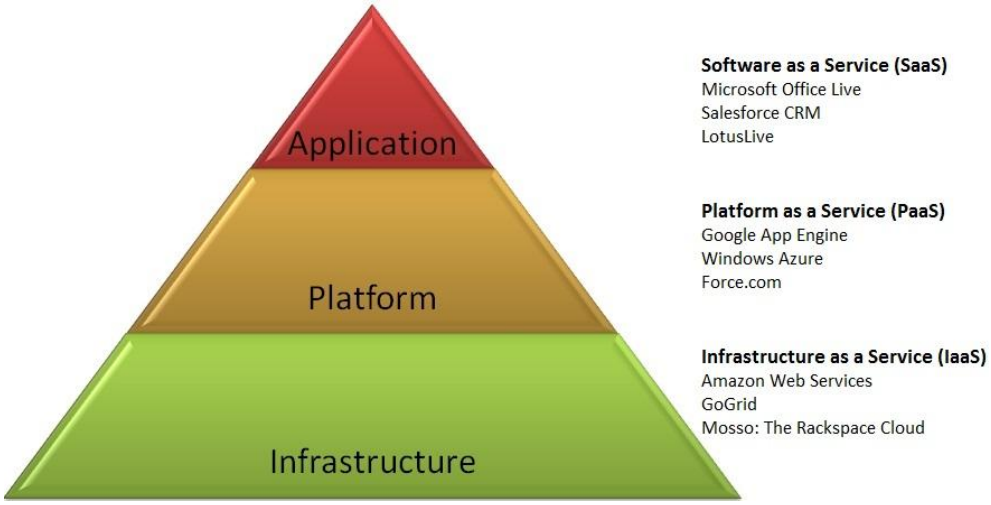

Figura 1 – Tipos de implementação de *Cloud Computing* (adaptado [26]])

2.2.1 Infrastructure as a Service (IaaS)

A classificação *Infrastructure as a Service* (IaaS), também designada por *Hardware as a Service* (HaaS), corresponde aos serviços fornecidos ao nível de infraestrutura, mais concretamente poder computacional, armazenamento e infraestrutura de rede (*firewalls* e *load balancers*).

Este tipo de serviços passou a ser disponibilizado pelos *datacenters* através de técnicas de virtualização, permitindo uma adaptação flexível das necessidades dos clientes face à oferta dos fornecedores, sendo apenas contabilizados os recursos utilizados pelo cliente (*utility computing*). Tipicamente, o cliente deste tipo de soluções é aquele que necessita de um ambiente computacional para correr as suas aplicações. Tendo a infraestrutura ao seu dispor, o cliente terá que instalar todo o *software* que necessita nas máquinas virtuais adquiridas, para o correto funcionamento das suas aplicações [26].

Tabela 1 – Exemplo de fornecedores de IaaS (adaptado [27])

Fornecedor	**Oferta IaaS**	**Ambiente de *Hosting***	**Armazenamento**	**Serviços *Cloud***
Amazon	Amazon Web Services [2] [28]	Elastic Compute Cloud	Elastic Block Storage	SimpleDB Simple Storage Services (S3) Cloud Front Simple Queue Services (SQS) Elastic MapReduce
ServePath	GoGrid [29]	GoGrid Cloud Hosting	GoGrid Cloud Storage	Nenhum
Rackspace	Mosso: The Rackspace Cloud [1] [30]	Cloud Servers Cloud Sites	Integrado com Cloud Servers	Cloud Files

2.2.2 Platform as a Service (PaaS)

As implementações *Platform as a Service* (PaaS), correspondem às *frameworks* e plataformas de programação fornecidas para o desenvolvimento de serviços para a *Cloud*. Fornecem um nível de abstração superior relativamente às IaaS, simplificando a gestão da infraestrutura.

Neste tipo de implementações, o *developer* não necessita de efetuar manutenções e *upgrades* de *software* a correr na *Cloud*, uma vez que é a própria infraestrutura (*Fabric*) que efetua automaticamente este tipo de tarefas.

Os ambientes PaaS fornecem um ambiente de execução para as aplicações, sendo apenas necessário o *upload* de *packages* contendo o código da aplicação ou uma versão compilada desse mesmo código. Esta interação é facilitada pela ligação direta entre os ambientes de desenvolvimento e o respetivo ambiente de execução na *cloud* suportado pelo fornecedor. Como exemplo, pode referir-se que a plataforma Microsoft Azure suporta a *framework* .NET e PHP e que o Google App Engine suporta Java e Python [26].

As implementações PaaS simplificam o desenvolvimento, *deployment* e configuração das aplicações, tornando-as extremamente escaláveis nas suas plataformas específicas. A grande desvantagem deste tipo de soluções é que uma aplicação desenvolvida para uma plataforma com determinada *framework* dificilmente será portável [27], apenas correndo no ambiente de execução para o qual foi desenvolvido [2].

Se a flexibilidade em termos de *software* é uma premissa importante, valorizando-se a independência face ao fornecedor, a melhor opção será a escolha de um ambiente IaaS. Tal como nas implementações IaaS, o cliente deste tipo de soluções é aquele que necessita de um ambiente computacional para correr as suas aplicações.

Tabela 2 – Exemplo de fornecedores de PaaS (adaptado [27])

Fornecedor	**Oferta PaaS**	**Ambiente de execução**	**Serviços *Cloud***
Google	Google App Engine [2] [31]	Java Runtime Environment Python Runtime Environment	Datastore Google Accounts Image Manipulation Mail Memcache URL Fetch
Microsoft	Azure Services Platform [1] [2] [32]	Windows Azure	Access Control Service SQL Services Workflow Services Services Bus Live Services
Salesforce.com	Force.com[1] [2] [33]	Apex Code(regras de negócio) Visualforce (*user interface*)	Database Services Web Service APIs

2.2.3 Software as a Service (SaaS)

A classificação *Software as a Service* (SaaS) corresponde à disponibilização de aplicações ou serviços existentes na *Cloud*, sendo apenas disponibilizadas as suas funcionalidades, encapsulando-se toda a complexidade inerente ao serviço. Como este tipo de aplicações corre sobre uma determinada plataforma e infraestrutura, são os fornecedores de SaaS que garantem que as aplicações (das quais são proprietários) se mantêm *online* e disponíveis, sendo os responsáveis pela manutenção da plataforma e infraestrutura que as suportam [26].

Ao contrário do que acontece em ambientes do tipo IaaS e PaaS, o cliente deste tipo de soluções é o utilizador final que consome as aplicações diretamente via *browser*.

Tabela 3 – Exemplo de fornecedores de SaaS (adaptado [27])

Fornecedor	Marca SaaS	Oferta
Microsoft	Microsoft Online Services [1] [2] [34]	Exchange Online SharePoint Online Dynamics CRM Online Office Live Meeting Office Communications Online
Salesforce.com	Salesforce CRM [1] [2] [35]	Sales Markting Service Partners
IBM	LotusLive [36]	LotusLive Engage LotusLive Connections LotusLive Meetings LotusLive Events LotusLive iNotes LotusLive Notes

2.2.4 Análise comparativa face ao tipo de implementação

A categorização das diferentes ofertas de *Cloud Computing* em IaaS, PaaS e SaaS permite perceber melhor toda a gama de serviços que a *Cloud* oferece. Os tipos de implementação variam mediante as funcionalidades que apresentam, adaptando-se aos diferentes objetivos e cenários existentes.

As implementações do tipo IaaS oferecem poder de computação, armazenamento e uma infraestrutura de rede disponibilizada como serviço, permitindo a sua adaptação face às necessidades de cada caso [27]. Neste tipo de implementações, as aplicações e todos os programas e componentes necessários, terão que ser instalados e configurados numa máquina virtual (VM – *Virtual Machine*), que irá ser posteriormente publicada na *Cloud* [27]. O cliente típico deste tipo de implementação é aquele que necessita de um ambiente computacional para disponibilizar o seu *software*, sendo o IaaS adequado para os seguintes cenários [88]:

- Para empresas com necessidade de utilização de infraestrutura muito variável (picos/quedas de utilização frequentes);
- Para novas empresas sem capital para investir em *hardware*;

- Para empresas com crescimento rápido em que a escalabilidade do *hardware* é extrema importância;

- Para necessidades temporárias de infraestrutura.

Por outro lado o IaaS não é adequado quando [88]:

- O armazenamento de dados fora da organização poderá gerar problemas legais ou regulamentares;

- São exigidos níveis de performance extremamente elevados (tempo-real) e a capacidade de resposta da infraestrutura existente corresponde a essas necessidades.

Tal como as implementações do tipo IaaS, as implementações PaaS oferecem também poder de computação, armazenamento e infraestrutura de rede via internet. Porém, adicionalmente oferecem um ambiente de execução de forma a ser possível correr código aplicacional compilado [27]. Desta forma, ao contrário das implementações do tipo IaaS, não é necessário a criação/configuração de uma VM de raiz, apenas a publicação do código aplicacional e inicialização do serviço. A infraestrutura garante a escalabilidade do *software* publicado, incluindo mecanismos de *load balancing* e *failover* (recuperação automática em caso de falha) [27]. Desta forma, a criação e publicação de *software* é simplificada, mitigando-se a complexidade inerente à compra e manutenção de *software*/*hardware* necessário para o suporte de aplicações. Tal como o IaaS, o cliente típico deste tipo de implementação é aquele que necessita de um ambiente computacional para disponibilizar o seu *software,* sendo o PaaS adequado para os seguintes cenários [88]:

- Quando se pretende agilizar a produção de *software*;

- Quando se necessita de uma plataforma que proporcione uma infraestrutura integrada para o desenvolvimento aplicacional;

- Quando se pretende automatizar testes e *deployment das* aplicações.

Por outro lado, o PaaS não é adequado quando [88]:

- A aplicação necessita de ser altamente portável;
- A performance da aplicação requer configurações específicas a nível de *hardware*/*software*.

Por fim, as implementações do tipo SaaS constituem aplicações finais disponibilizadas como serviços. Neste tipo de implementações as aplicações existentes na *Cloud* disponibilizam um conjunto de funcionalidades que serão consumidas via *browser* [27]. O *software* é gerido a partir de uma localização central e distribuído num esquema de "um para muitos", onde o cliente típico deste tipo de implementação (utilizador final) não necessita de lidar com qualquer tipo de atualização de *software* [88]. O SaaS pode operar sobre os ambientes de IaaS e PaaS, sendo adequado para os seguintes cenários [88]:

- Para aplicações onde existe uma interação significativa com o mundo exterior;
- Para *software* cujo acesso via *web* (ou móvel) é significativo;
- Para aplicações que serão utilizadas por um período de tempo curto;
- Para *software* sujeito a picos de utilização.

Por outro lado, o SaaS não é adequado para:

- Aplicações que exijam um tempo certo de execução (tempo real);
- Aplicações cuja legislação ou regulamentação impeça que os dados sejam armazenados externamente;
- Aplicações cujas soluções locais existentes já possuam capacidade de resposta face às necessidades impostas.

Tabela 4 – Análise comparativa de implementações ([27])

Tipo	**Cliente**	**Unidade de *deploy***	**Oferta**	**Esquema Típico de Faturação**
IaaS	Fornecedor Software	Imagem de Máquina Virtual	Ambiente de execução para máquinas virtuais Armazenamento na *Cloud (Storage)* Poderá ter serviços *Cloud*	Por período de faturação: Utilização de computação (por hora) Transferências de dados (por GB) Pedidos E/S (por milhão) Armazenamento (por GB) Transferências de dados - *Storage* (por GB) Pedidos E/S - *Storage* (por milhar)
PaaS	Fornecedor Software	Package de Aplicação	Ambiente de execução para código aplicacional Armazenamento na *Cloud (Storage)* Serviços Cloud	Por período de faturação: Utilização de computação (por hora) Transferências de dados (por GB) Pedidos E/S (por milhão) Armazenamento (por GB) Transferências de dados - *Storage* (por GB) Pedidos E/S - *Storage* (por milhar)
SaaS	Cliente Final	N/A	Aplicações finais	Por utilizador

3 Windows Azure

O Windows Azure, tal como outras soluções de *Cloud Computing* existentes, surge para tentar corresponder às necessidades aplicacionais das diferentes organizações garantindo escalabilidade, redundância e tolerância a falhas. A plataforma Azure consiste numa infraestrutura de *hardware*, *software*, rede e armazenamento que se gere automaticamente, garantindo elevada disponibilidade e performance às aplicações que suporta. Através da virtualização de recursos computacionais, o Windows Azure baseia-se no conceito de instâncias de computação que isoladamente tratam dos pedidos originários do *front-end* da aplicação (através dos *Web Roles*) e das tarefas a realizar em *background* (através dos *Worker Roles*). Baseia-se também num modelo de faturação *pay-per-use*, permitindo um ajuste dinâmico dos recursos de computação e de armazenamento de forma simplificada.

A plataforma Azure é flexível suportando múltiplas linguagens de programação podendo ser integrada com os ambientes de desenvolvimento comuns. Os *developers* podem usar o Microsoft Visual Studio (ou outros ambientes de desenvolvimento) conjuntamente com o "Windows Azure SDK", de forma a simular o ambiente da *cloud* localmente para o desenvolvimento aplicacional, sendo suportados os principais *standards*, linguagens de programação e protocolos entre os quais SOAP, REST, XML, Java, C#, PHP e Ruby [2]. A plataforma Azure é constituída pelos seguintes componentes principais [6]:

- *Fabric* – Coleção de servidores com múltiplas máquinas virtuais por servidor;
- *Fabric Controller* – Componente que gere o sistema operativo Windows Azure. Interage com cada máquina do *datacenter* através de um agente, monitorizando cada

máquina virtual e aplicação. Efetua o balanceamento de carga e otimiza a utilização do *hardware*;

- *Compute* – Computação virtualizada. Proporciona escalabilidade uma vez que permite a replicação de instâncias de computação, mediante necessidades aplicacionais;
- *Storage* – Serviço de armazenamento na *Cloud*, que proporciona um armazenamento não relacional de dados;
- *SQL Azure* – Serviço de suporte para bases de dados relacionais na *Cloud*, baseado no sistema de gestão de bases de dados da Microsoft (SQL Server).

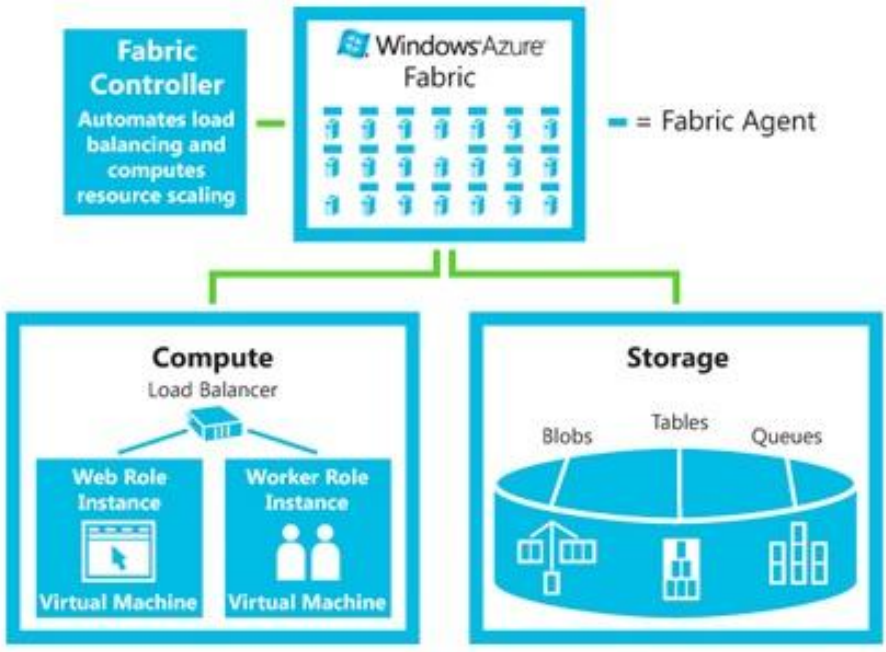

Figura 2 – Arquitetura Windows Azure [80]

3.1 Computação

O Windows Azure fornece capacidade de computação às aplicações de forma a permitir que estas corram na *cloud*, mitigando possíveis problemas de escalabilidade.

O Azure possui o conceito de instância de computação, representando uma máquina virtual que corre isoladamente e possui capacidades variáveis (Tabela 5).

Tabela 5 – Tamanhos e capacidade de instâncias de computação à data da escrita (25-02-2012 - adaptado [45])

Tamanho da instância de computação	**CPU**	**Memória (GB)**	**Espaço Armazenamento na instância (GB)**	**Desempenho E/S (Mbps)**
Extra - Pequena	Partilhados	0,768	20	5
Pequena	1	1,750	165	100
Média	2	3,5	340	200
Grande	4	7	850	400
Extra – Grande	8	14	1890	800

Embora a maior parte dos recursos das VM sejam dedicados, alguns associados ao desempenho de E/S (entrada/saída) são partilhados entre as instâncias no mesmo *host* físico no *datacenter* [46]. Ou seja, em períodos de acesso simultâneo, quem tiver um desempenho E/S alto (> =200Mbps) terá uma maior alocação de banda e por consequência um desempenho mais consistente [46].

Em Windows Azure, as aplicações utilizam recursos computacionais através de um ou mais componentes de computação designados por *roles* [47]. Existem três tipos distintos de *roles* [47]:

- *Web Role* – Disponibiliza um servidor de IIS (Internet Information Services) dedicado para o suporte do *front-end* das aplicações *web*;
- *Worker Role* – Otimizado para processamento de tarefas em *background*. É similar a um serviço do Windows que está permanentemente em execução, sendo particularmente utilizado para processamento assíncrono de tarefas, para tarefas que não necessitem de interação com os utilizadores e/ou tarefas que exijam tempo e poder de computação elevado;

- *Virtual Machine Role* – Permite o *deploy* de uma imagem de servidor com o sistema operativo Windows Server 2008 R2 para o Windows Azure. Este novo *role* (ainda em versão Beta) permite controlar todo o ambiente aplicacional, simplificando a migração de aplicações existentes para a *Cloud*.

Aplicações que utilizem *Web Roles* para o processamento de pedidos relacionados com o *front-end* e *Worker Roles* para o processamento de tarefas a realizar em *background*, desacoplam os diferentes componentes da aplicação, permitindo uma adaptação do número de instâncias de computação à carga verificada em cada um destes componentes (computação elástica) [46].

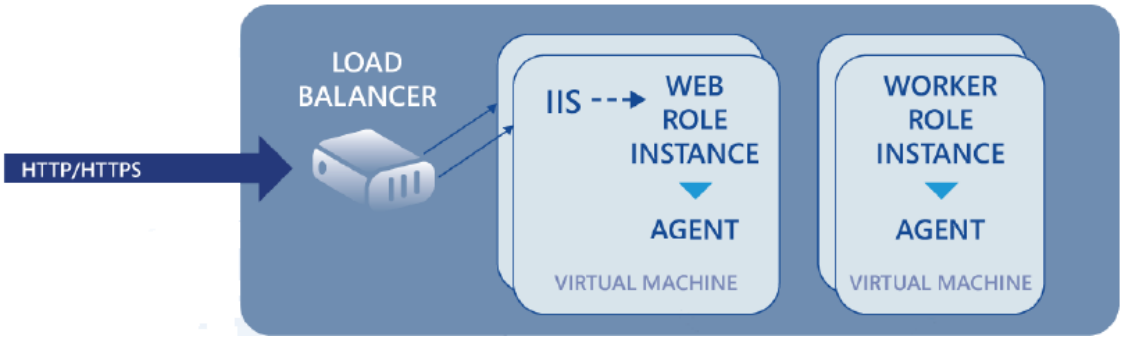

Figura 3 – Computação – principais componentes [6]

As horas de computação cobradas são calculadas a partir do momento em que é feito o *deploy* da aplicação para a *Cloud*, sendo multiplicadas pelo número de instâncias utilizadas [47]. O *deployment* para a *Cloud* é configurado através de dois ficheiros XML na aplicação: *ServiceDefinition.csdef* e *ServiceConfiguration.cscf* (Figura 4).

```
<Role name="BContestWeb">
  <Instances count="1" />
  <ConfigurationSettings>
    <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" value="UseDevelopmentStorage=true" />
    <Setting name="DataConnectionString" value="UseDevelopmentStorage=true" />
  </ConfigurationSettings>
</Role>
<Role name="BContestWorker">
  <Instances count="1" />
  <ConfigurationSettings>
    <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" value="UseDevelopmentStorage=true" />
    <Setting name="DataConnectionString" value="UseDevelopmentStorage=true" />
  </ConfigurationSettings>
</Role>
```

Figura 4 – Configuração da aplicação - *ServiceConfiguration.cscf*

O ficheiro de definição do serviço define os *roles* disponíveis para o serviço, especificando os seus *endpoints* e estabelecendo parâmetros de configuração do serviço. Por outro lado, o

ficheiro de configuração do serviço especifica valores de configuração para um ou mais *roles* do serviço (ex: definição da *connection string* para aceder à *Storage*).

3.2 Storage

O *Windows Azure Storage* é o componente arquitetural do Windows Azure que permite às aplicações armazenarem os seus dados na *Cloud* num modelo não relacional (*NoSQL*), endereçando os problemas de escalabilidade, concorrência, distribuição e descentralização dos dados. Problemas estes que tipicamente condicionam as aplicações baseadas em bases de dados relacionais, à medida que estas (ou o seu número de utilizadores) vão crescendo.

As abstrações de dados utilizadas a este nível são [4]:

- *Blobs (Binary Large Object)* – Proporcionam uma interface simplificada para o armazenamento de objetos, sendo estes guardados através do par (nome, valor);

- *Drives* – Disponibilizam um volume NTFS (*New Technology File System*) para o armazenamento de dados num disco persistente usando as APIs (*Application Programming Interface*) do NTFS, através das *Blobs*;

- *Tables* – Consistem em tabelas altamente escaláveis, sem esquema fixo, em que cada linha pode ter mais ou menos atributos que as restantes e cada atributo é armazenado através do par (nome, valor tipado). Cada linha da tabela possui obrigatoriamente uma *PartitionKey* e uma *RowKey*, sendo apenas permitido efetuar transações sobre a mesma partição;

- *Queues* – Consiste num mecanismo utilizado para troca de mensagens entre as diferentes partes da aplicação (*roles*), tornando-a desacoplada e dotando-a de forte escalabilidade.

A *Storage* garante a durabilidade e replicação dos dados, garantindo que estes nunca serão perdidos. As *Windows Azure Blobs*, *Tables* e *Queues* armazenados na *Storage* são replicadas três vezes no mesmo *datacenter* de forma a prevenir falhas de *hardware*, sendo esta replicação distribuída por três domínios de falha distintos, de forma a aumentar a disponibilidade dos dados [51]. Adicionalmente, as *Windows Azure Blobs* e *Tables* são

também replicadas geograficamente entre dois *datacenters*, com centenas de quilómetros de distância entre eles, de forma a prevenir a perda de dados em caso de desastres naturais [51].

Para a utilização deste sistema, através do portal do Windows Azure, o utilizador terá que criar uma conta na qual cria um espaço do tipo *Storage*. Após a criação, recebe uma chave de 256 *bits* que é utilizada para gerar um *hash*, funcionando como uma assinatura. É esta assinatura que é enviada em cada pedido feito à *Storage*, de forma a autenticar o pedido de acesso.

3.2.1 Tables

No âmbito da *Storage*, as *Windows Azure Tables* são a solução de armazenamento estruturado, fornecido pela plataforma Windows Azure.

Suportam tabelas de alta escalabilidade com capacidade de armazenar biliões de linhas e *terabytes* de dados. Estas tabelas têm a particularidade de não possuírem um esquema fixo, como as tabelas tipicamente utilizadas. Desta forma, cada propriedade de uma linha é guardada com o formato (nome, valor tipado).

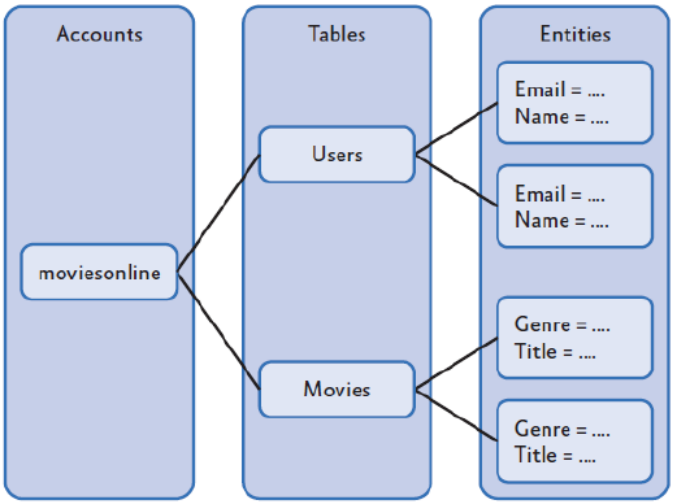

Figura 5 – *Windows Azure Tables* – modelo de dados [6]

O modelo de dados implica alguns conceitos enumerados de seguida [5]:

- *Table* – Suporta um conjunto de entidades. Podem existir múltiplas tabelas associadas a cada conta de *Storage*;

- *Entity (Linha)* – Entidades (linhas da tabela) são os elementos guardados nas tabelas. Cada entidade possui um conjunto de propriedades (máximo 255), sendo duas delas obrigatórias: *PartitionKey* e a *RowKey*. Esta chave múltipla é a que identifica univocamente cada entidade;

- *Property (Coluna)* – Representa um valor simples, com determinado tipo, associado a cada entidade;

- *PartitionKey* – A primeira chave das tabelas. É a partir desta chave que o sistema distribui as entidades pelos diversos servidores;

- *RowKey* – A segunda chave das tabelas, constituída por um ID único que identifica a entidade dentro da partição à qual pertence. A *PartitionKey* combinada com a *RowKey* identifica univocamente uma entidade na tabela;

- *Timestamp* – Propriedade mantida pelo sistema, que consiste na versão da entidade;

- *Partition* – Conjunto de entidades com a mesma *PartitionKey* agrupadas numa tabela;

- *Index* – Apenas existe um índex nas *Azure Tables*. Este índex é formado pela *PartitionKey* e pela *RowKey*. Assim sendo, *queries* efetuadas usando estas chaves serão mais eficientes, sendo os resultados ordenados pela *PartitionKey* e depois pela *RowKey*.

O acesso às tabelas pode ser feito a partir do seguinte URL:

http://<<nomeConta>.**table**.core.windows.net

Conforme já referido, as *Windows Azure Tables* permitem armazenar biliões de registos, pelo que questões de escalabilidade são de extrema importância. A técnica utilizada pelo sistema consiste na distribuição das entidades pelos diferentes servidores. Esta distribuição é efetuada

a partir do valor da *PartitionKey* de cada entidade, pelo que é necessária uma análise cuidada quando se agrupam as entidades para cada aplicação [5].

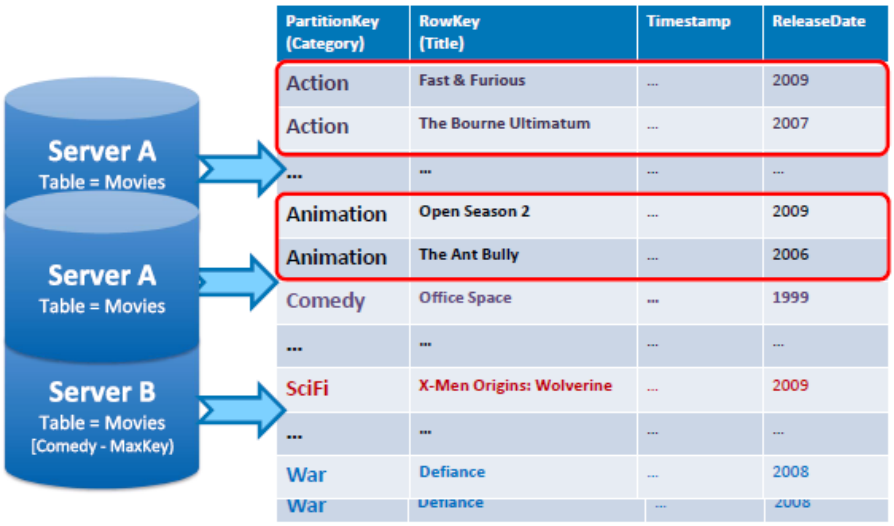

Figura 6 – *Windows Azure Tables* – partições [6]

O sistema monitoriza padrões de utilização de cada partição e efetua balanceamento de carga destas partições pelos vários nós servidores, espalhando o tráfego por eles.

Entidades com a mesma *PartitionKey* ficam juntas quando armazenadas, sendo servidas por um único nó [5]. Assim, logicamente (analisando o caso da Figura 6) uma *query* que retorne o nome de todos os filmes de ação, vai ser muito mais eficiente do que uma que retorne o nome de todos os filmes de Guerra e Animação (assumindo que o mesmo número de registos é retornado). Isto porque no segundo caso teriam que ser consultadas duas partições distintas, não estando as entidades no mesmo nó servidor como sucede no primeiro caso.

As transações entre entidades da mesma tabela e partição são atómicas [5]. As aplicações podem efetuar operações CUD (*Create*, *Update*, *Delete*) numa única operação. Se todas as operações da transação obtiverem sucesso, a operação termina com sucesso, senão toda a operação falha.

A escolha da *PartitionKey* torna-se então fulcral para o sucesso da escalabilidade de uma aplicação. É uma escolha que tem que ser feita com ponderação, uma vez que a eficiência das *queries* entre entidades da mesma partição contrasta com a escalabilidade da(s) tabela(s). Isto ocorre uma vez que quanto menos partições existirem, mais difícil é para o Azure distribuir a carga pelos diferentes servidores [5]. Quando as entidades são agrupadas, deve ter-se em

consideração as transações que existirão entre grupos de entidades, de forma a conseguir-se manipular todas elas na mesma transação. As *queries* efetuadas frequentemente, ou aquelas cuja latência é relevante, deverão sempre utilizar a *PartitionKey* como filtro de forma a obter-se uma performance elevada.

A manipulação das tabelas, bem como das entidades que as compõem, tem regras e processos próprios, endereçando os problemas já referidos (escalabilidade, concorrência,..). Esta manipulação pode ser feita via APIs do .NET, bem como através de REST, uma vez que os comandos da biblioteca de classes do .NET para a manipulação das tabelas/entidades resultam em transmissões de pacotes REST [15]. Um exemplo interessante é o processo *update* às entidades das tabelas. Em .NET o *update* deveria ser feito genericamente da seguinte forma:

- Criar um objeto do tipo *DataServiceContext* (classe que proporciona a API para as operações CRUD);
- Selecionar, através de uma *query*, a entidade da tabela para o *DataServiceContext* e modificar localmente o objeto;
- Adicionar o objeto ao mesmo *DataServiceContext* para fazer o *Update*;
- Através do método *SaveChanges* enviar o pedido para o servidor.

Analisando este processo de *update*, pode ser imediatamente identificado um problema. Se primeiro se seleciona a entidade do servidor, sendo esta alterada localmente e só depois é enviada de volta com as alterações, que garantias se tem que a versão original da entidade alterada ainda é a mesma que no momento do *update* se encontra no servidor?

Este problema é resolvido recorrendo ao sistema *optimistic concurrency* das *Windows Azure Tables* [5]. Conforme referido, cada entidade possui uma propriedade gerida pelo sistema (*Timestamp*). Sempre que é feito um *update* a uma determinada entidade, este *Timestamp* é alterado. Desta forma quando se obtém uma entidade, obtém-se também a sua versão através de uma *HTTP ETag*[2]. Quando o cliente envia o pedido de *update*, a *ETag* é enviada

[2] Mecanismo de validação do protocolo http, que permite ao cliente fazer pedidos condicionais ao servidor [20].

também. O servidor analisa a condição presente no *header* http e caso a versão da entidade seja igual à versão do servidor, o pedido é concluído com sucesso, sendo gerada uma nova versão da entidade. Caso contrário a alteração é rejeitada, sendo devolvido um erro para o cliente (*precondition failed*) [5]. No caso de receber este erro específico, a aplicação cliente deve efetuar o mesmo processo, até conseguir efetuar a operação pretendida.

Outro problema surge quando se pretende garantir consistência entre duas ou mais tabelas (ex: remover um determinado filme de uma tabela filmes e todos os detalhes associados a esse filme das restantes tabelas). Entre a mesma partição, as transações relativas a operações de manutenção estão garantidas. O sistema cria uma *snapshot isolation*, de forma a garantir que os dados sobre os quais se opera são consistentes no decorrer da transação. Mas o mesmo não ocorre em transações sobre partições diferentes. Para se garantir a consistência entre múltiplas tabelas, terão que se prever esses cenários e as falhas que possivelmente ocorrerão ao nível da aplicação. Uma das formas de o fazer é utilizando as *Windows Azure Queues* para preservar o estado da transação. Assim, mesmo que ocorram falhas durante o processo transacional, um *Worker Role* pode sempre completar a transação, uma vez que só quando esta for concluída com sucesso é que será eliminada da *Queue*.

3.2.2 Blobs

Contrastando com as *Windows Azure Tables*, as *Windows Azure Blobs* são a solução de armazenamento não estruturado fornecido pela plataforma Windows Azure.

As *Blobs* permitem o armazenamento de elevado volume de dados não estruturados, em formato texto ou binário por exemplo (imagens, ficheiros áudio/vídeo,..). Disponibiliza *containers* do tipo *Block* (para streaming de dados) ou do tipo Page (para operações de leitura/escrita simples), sendo os dados armazenados replicados pelo menos três vezes, tal como acontece com as *Tables* [51].

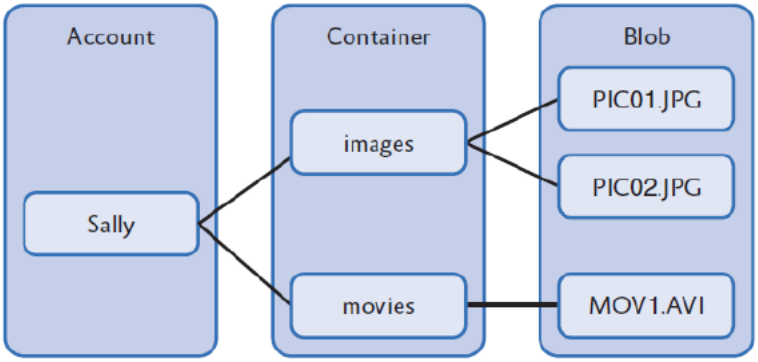

Figura 7 – *Windows Azure Blobs* – modelo de dados [6]

O modelo de dados é constituído pelos seguintes componentes [4]:

- *Blob Container* – Constituem grupos de *Blobs*, podendo cada conta possuir vários *containers*. É a este nível que se define as políticas de partilha: *Public Read* ou *Private*, mediante obrigatoriedade de autenticação para acesso ao seu conteúdo;

- *Blob* – Constituem ficheiros de qualquer tipo / tamanho. São armazenadas dentro dos *containers*, tendo obrigatoriamente que possuir um nome único nesse âmbito. Os dois tipos de *blobs* possíveis de armazenar são *Block Blobs* ou *Page Blobs*, sendo que a maior parte das utilizações são do tipo *Block Blobs* [77] . Cada *Block Blob* pode ter um tamanho até 200 GB, enquanto que uma *Page Blob* pode ter até 1 TB, sendo as *Page Blobs* mais eficientes quando os ficheiros são frequentemente modificados [77].

Tal como as *Azure Tables*, também as *Blobs* podem ser manipuladas via APIs do .NET bem como através de REST [15], sendo o acesso a uma determinada *blob* feito a partir do seguinte URL:

http://<<nomeConta>.**blob**.core.windows.net/<container>/<nomeBlob>

As *Page Blobs*, introduzidas em 2009-09-19 [79], são uma alternativa às *Block Blobs*. Enquanto que as *Block Blobs* são otimizadas para *streaming* de dados as *Page Blobs* são otimizadas para escritas/leituras aleatórias, uma vez que os dados são armazenados em páginas de 512 *bytes*, em vez de serem armazenadas sequencialmente em blocos de dados, conforme se verifica nas *Block Blobs*.

No momento de criação de uma *Page Blob*, através da instrução PUT *blob* define-se o seu tamanho máximo e através da instrução PUT *Page*, adiciona-se ou altera-se o conteúdo de uma página, ou de um conjunto de páginas. As escritas para *Page Blobs* são imediatamente submetidas na *Blob*, sendo o seu tamanho máximo 1 TB [79]. As *Page Blobs* constituem também o mecanismo de armazenamento das *Windows Azure Drives*, que serão abordadas na secção 3.2.3.

Por outro lado, as *Block Blobs* são constituídas por conjunto de *Blocks* (blocos de dados), cada um deles identificado por um "BlockID". No seu conjunto formam uma *blob* e são tipicamente utilizados para subdividir *blobs* com elevada dimensão. Cada *block* pode ter um tamanho variável até 4 MB, sendo o tamanho máximo da *Block Blob* 200 GB [78].

Blobs até 64Mb podem ser enviadas para a *Cloud* através de uma única operação PUT *blob* porém, para *blobs* de tamanho superior, terá que ser utilizada a interface dos *blocks*. Nestes casos, deverá particionar-se as *blobs* de forma a permitir o *upload* de múltiplos *blocks*. Tendo os *blocks* um ID ou nome único, esta operação é efetuada através de PUTs contínuos em URLs contendo o ID de cada *block*. Após esta operação, é enviada a lista ordenada de todos os *blocks* enviados, que constituem uma determinada *blob*, através de um PUT com o comando *blocklist* [4]. Esta lista ordenada de *blocks* constitui a *blob*, que pode ser lida através de um GET *blob*. As principais operações associadas às *Block Blobs* são então o PUT *block*, que permite o *upload* de um determinado *block* para uma *blob* e o PUT *blocklist*, que no fundo submete uma *blob*, através da especificação da lista de *blocks* que a constitui.

Esta forma de armazenamento traz múltiplas vantagens em termos de eficiência e tolerância a falhas. Ao ser subdividido um objeto grande em múltiplos objetos de menor tamanho é possível não só efetuar o *upload* dos *blocks* em paralelo, diminuindo o tempo necessário para armazenar a *blob* completa, como também permitir em caso de falha no envio de um *block*, o seu reenvio, em detrimento do reenvio da *blob* completa [4]. O *upload* dos *blocks*, como já foi referido, pode ser feito segundo qualquer ordem, uma vez que é a operação PUT *blocklist* que define qual vai ser a ordem correta para a constituição da *Blob* [4].

Tal como sucede com as *Windows Azure Tables*, para mitigar problemas associados com *updates* concorrentes, as *Windows Azure Blobs* também adotam o mecanismo de *optimistic concurrency*, baseando-se na data de última modificação da *blob* [4]. Assim sendo, caso a

ETag seja diferente da versão da *blob* que se está a modificar, é gerado um erro, correspondente à diferença de versões verificadas no momento do PUT.

Tal como existem operações de PUT condicionais, também são possíveis GET condicionais. Tipicamente são utilizadas em casos em que o cliente armazena *blobs* em cache local. Nestes casos, quando se pretende fazer uma atualização das *blobs* em cache, é invocado um GET condicional, sendo apenas efetuado o *download* (a partir da *Storage*) daquelas que efetivamente sofreram alterações desde uma determinada data [4].

Para prevenir casos em que se verifique um GET e um PUT concorrente sobre a mesma *blob*, é adotado o mecanismo de *snapshot isolation*, permitindo ao GET receber uma versão completa da *blob* e não parte de uma versão e parte de outra.

3.2.3 Drives

As *Windows Azure Drives* disponibilizam um volume NTFS persistente, utilizando-se as *Page Blobs* como mecanismo de armazenamento de dados.

As *drives* são implementadas como *Page Blobs* contendo uma *Virtual Hard Drive* (VHD) em formato NTFS [4]. Funcionam como discos persistentes, permitindo a utilização das APIs do NTFS para leituras e escritas através da *drive* mapeada (ex: X:\).

Um aspeto importante quando são utilizadas *Windows Azure Drives* é a possibilidade de se efetuar cache dos dados da *drive*, num disco local na VM. A cache dos dados na *drive* local reduz o tráfego gerado pela leitura de dados à *Page Blob*, reduzindo os custos associados a transações entre a VM e a *Storage* [4]. Isto sucede uma vez que leituras à cache do disco local não são cobradas, contrariamente ao que sucede quando estas são efetuadas às *Page Blobs* [4]. A Figura 8 ilustra este processo.

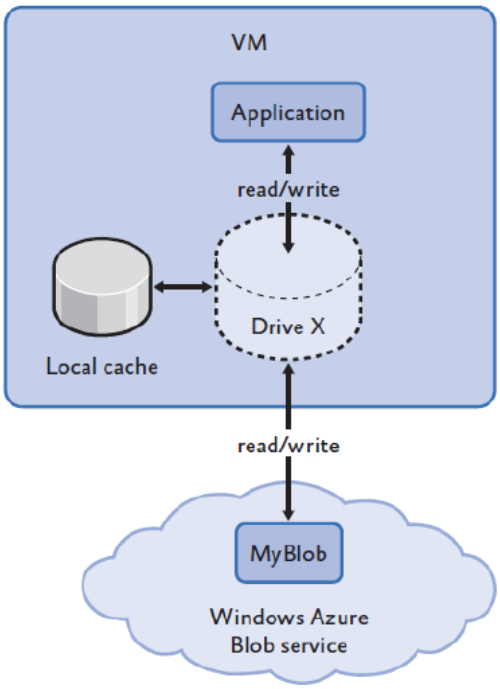

Figura 8 – *Windows Azure Drives* – cache dos dados na VM [6]

3.2.4 Queues

As *Windows Azure Queues* disponibilizam um mecanismo fiável para troca de mensagens na plataforma Azure. Este mecanismo proporciona uma entrega assíncrona de mensagens, interligando os diferentes componentes da aplicação.

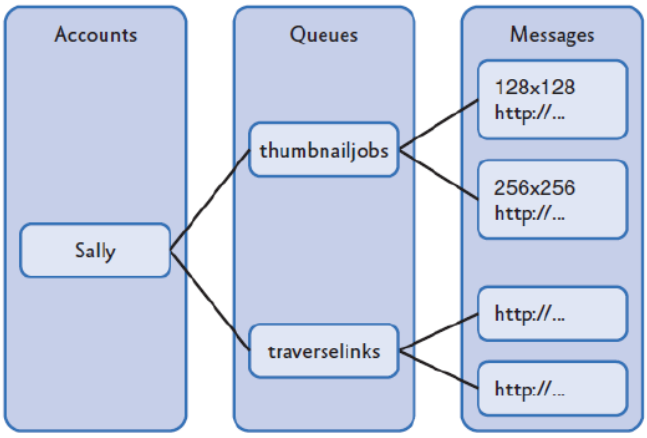

Figura 9 – *Windows Azure Queues* – modelo de dados [6]

O modelo de dados é constituído pelos seguintes componentes [7]:

- *Queue* – Constituem filas que contêm múltiplas mensagens. Não há limite quanto ao número de mensagens guardadas numa *queue*, sendo que mensagens armazenadas com mais de uma semana são automaticamente eliminadas pelo sistema. As *queues* possuem meta-dados associados sendo armazenados através do par <nome, valor>;

- *Messages* – Constituem as mensagens armazenadas nas *Queues*. Cada mensagem pode ter um tamanho até 64 KB (a partir da versão de 2011-08-18), caso contrário até 8 KB. Para armazenar mensagens maiores que estes limites, pode-se utilizar *blobs* ou *tables* para armazenar a informação necessária, colocando-se na mensagem a referência para os dados. As mensagens são tipicamente adicionadas ao final da fila e obtidas a partir do início, embora não seja garantido o comportamento FIFO (*First In First Out*).

As *Windows Azure Queues* podem também ser manipuladas via APIs do .NET, bem como através de REST [15], sendo o acesso a uma determinada *queue* feito a partir do seguinte URL:

http://<<nomeConta>.**queue**.core.windows.net/< nomeQueue>

3.3 Topologia

O Windows Azure, utilizando os componentes descritos anteriormente, possui uma topologia específica que no seu conjunto garante escalabilidade, redundância e tolerância a falhas, de forma a dar resposta às mais elevadas necessidades aplicacionais (Figura 10).

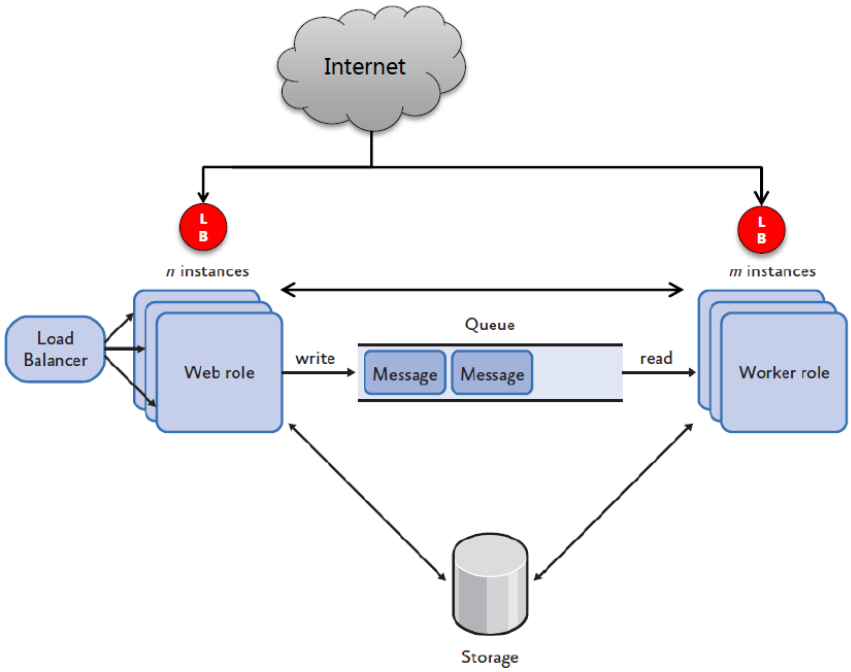

Figura 10 – Topologia Windows Azure [6]

Segundo esta topologia, todos os pedidos são processados por um *load balancer* (fornecido pelo *Fabric* do Windows Azure) antes de serem reencaminhados para um determinado *Web Role* ou *Worker Role*, de forma a haver um reencaminhamento uniforme do tráfego para as instâncias de cada tipo.

Os *Web Roles* processam os pedidos do *front-end* da aplicação, enquanto que os *Worker Roles* processam as tarefas em *background*, utilizando-se *Queues* para permitir a comunicação entre eles. A *Storage* e/ou o SQL Azure (que irá ser abordado no tópico 3.4) são componentes que garantem o armazenamento, permitindo salvaguardar o estado das aplicações. Porém, são as *Queues* que desempenham um papel central na topologia do Windows Azure. É através delas que é possível desacoplar as diferentes partes de uma aplicação, permitindo uma maior escalabilidade de acordo com as necessidades da mesma [7].

Num cenário típico de uma aplicação Azure, os *Web Roles* lidam com os pedidos *web* dos clientes. Se alguma tarefa exigir um processamento mais intensivo e/ou mais demorado, os *Web Roles* relegam o trabalho para os *Worker Roles*, colocando na *Queue* uma mensagem que de alguma forma identifica o trabalho a executar (ou este se encontra no conteúdo da mensagem, ou numa *blob*/*table*, sendo o conteúdo da mensagem a referência para esses dados) [7]. Quando surge uma nova mensagem na *Queue*, os *Worker Roles* "retiram-na" para processamento, ficando esta invisível para os restantes *Worker Roles* por um determinado período de tempo (propriedade *VisibilityTimout* da mensagem). Após o seu processamento, o *Worker Role* elimina a mensagem de forma a evitar o seu processamento em duplicado por parte de outro servidor de *back-end* [7]. Se por alguma razão a mensagem não for eliminada por parte do *Worker Role*, findo o *VisibilityTimout* volta a ficar visível.

A utilização desta topologia traz vários benefícios nomeadamente ao nível da escalabilidade. Analisando o tamanho da *queue*, consegue-se verificar como os *Worker Roles* estão a processar os pedidos requeridos. Se a fila crescer, pode concluir-se que os servidores de *back-end* não estão a ter o rendimento necessário e, neste caso, deverão ser acrescentadas instâncias de computação para consumir/processar mais rapidamente as mensagens. Se por outro lado, o tamanho da fila estiver praticamente a zero, deverão ser retiradas instâncias de computação dos *Worker Roles* pois não serão necessários tantos recursos [7].

Como as diferentes partes da aplicação se encontram desacopladas, a adaptação dos diferentes tipos de instâncias à carga a que estão sujeitas é independente. Estas podem assim ser ajustadas verticalmente (através do aumento da sua capacidade de processamento) e/ou horizontalmente (através do aumento do número de instâncias) sem afetar a lógica aplicacional [6].

Outra vantagem na utilização de troca de mensagens entre as diferentes partes da aplicação, caso estas sejam compreendidas pelas diferentes partes (ex: utilizando-se um formato standard - XML), é o fato dos componentes poderem ser alterados (nova tecnologia/linguagem de programação) com total transparência para o sistema, uma vez que as partes desacopladas apenas têm que compreender o conteúdo das mensagens, aumentando a flexibilidade e capacidade de extensibilidade da aplicação [7].

A utilização de *queues* nesta topologia permite também flexibilizar a utilização de recursos, uma vez que podem ser utilizadas filas distintas para tratar diferentes tipos de itens:

mensagens com diferentes prioridades, diferentes necessidades de processamento,.. Desta forma podem ser alocados recursos mediante a carga verificada em cada uma das filas de trabalho.

As *queues* mitigam também o impacto de falhas individuais a nível dos componentes [7]. Se um servidor de *back-end* falhar, o trabalho a executar não é perdido, ficando armazenado na fila. Este mecanismo de tolerância a falhas, permite que o sistema se mantenha disponível até os servidores voltarem a ficar operacionais.

3.4 SQL Azure

O SQL Azure é um serviço de suporte a bases de dados relacionais na *Cloud*, construído sobre a tecnologia SQL Server.

Proporciona um serviço de base de dados de alta disponibilidade, escalável e com mecanismos de tolerância a falhas. Com este serviço dispõe-se funcionalmente de um *datacenter* para correr instâncias de SQL Server sem necessidade de administração física do mesmo, conforme acontece com as instalações habituais de SQL Server. Os *developers* não têm assim que instalar, parametrizar e gerir nenhum tipo de *software*, podendo-se dedicar exclusivamente ao desenvolvimento aplicacional das BDs [2]. Este desenvolvimento é também simplificado uma vez que a linguagem *Transact-SQL* (T-SQL) é parcialmente suportada [13], bem como o protocolo de comunicação utilizado entre clientes/servidor do SQL Server: TDS (*Tabular Data Stream*). Em cada servidor SQL Azure podem ser criadas várias BDs com os objetos comuns (tabelas, views, stored procedures,..), porém, tanto os servidores como as BDs são objetos virtuais, não correspondendo a servidores e BDs físicos.

O SQL Azure replica múltiplas cópias de dados para diversos servidores físicos, de forma a preservar a disponibilidade destes e a continuação do negócio em caso de falha do sistema [2].

Relativamente à arquitetura do SQL Azure (Figura 11) as instâncias encontram-se instaladas e distribuídas pelas máquinas dos *datacenters* da Microsoft, encontrando-se o *hardware* separado daquele que é utilizado para suportar o *Azure Compute* e a *Storage* [8].

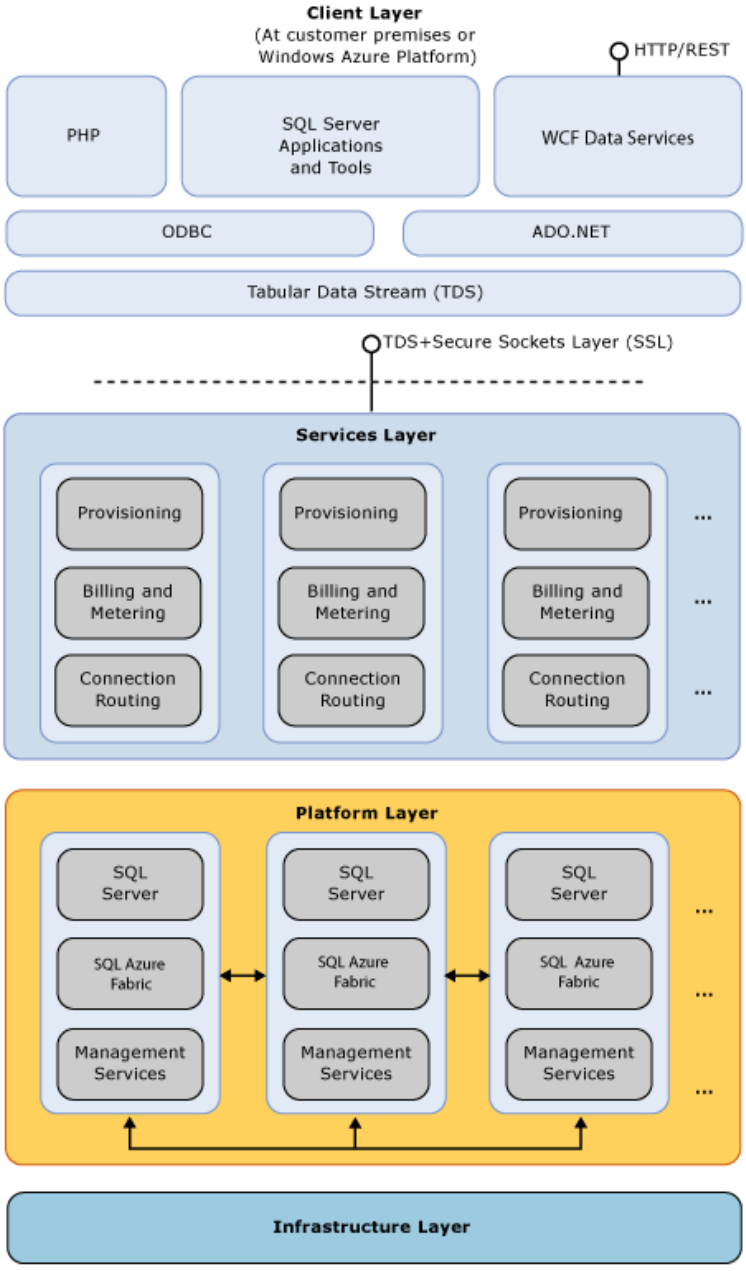

Figura 11 - Arquitetura SQL Azure [11]

A camada de plataforma comporta os servidores físicos e os serviços que suportam a camada de serviço. Consiste em várias instâncias SQL Server cada uma delas gerida pelo *SQL Azure Fabric*, que adereça as questões de *load balancing*, *failover* (recuperação automática em caso de falha) e replicação automática entre servidores no *datacenter*. Como o SQL Azure armazena múltiplas cópias das BDs, em caso de falha na máquina onde se encontra a BD, arranca automaticamente uma outra máquina idêntica com uma cópia da base de dados, de forma a manter o sistema disponível.

Na camada de serviço é feito o aprovisionamento, medição, faturação e *routing* de conexões. É através deste SaaS que o cliente tem a possibilidade de aceder à(s) BD(s), quer através do portal Azure, quer através dos serviços ODBC e ADO.NET via TDS [8]. O acesso via REST é também possível, mantendo-se assim compatibilidade com as mesmas linguagens de programação que o SQL Server suportava, não trazendo a migração de aplicações para SQL Azure problemas a este nível.

Tipicamente as aplicações que passam a utilizar SQL Azure possuem dois tipos de arquitetura possível: as aplicações locais que correm num *datacenter*/servidor local em que apenas a BD é migrada para SQL Azure (modificando-se apenas o local dos dados), ou aquelas em que a aplicação é também migrada para Windows Azure. No primeiro cenário, a transmissão de dados entre a aplicação e a BD é feita via TDS sobre SSL (*Secure Sockets Layer*), sendo a comunicação cifrada, tornando segura a transmissão de dados via internet [12]. Já no segundo, o utilizador acede via cliente *web* à aplicação na *Cloud* (Figura 12).

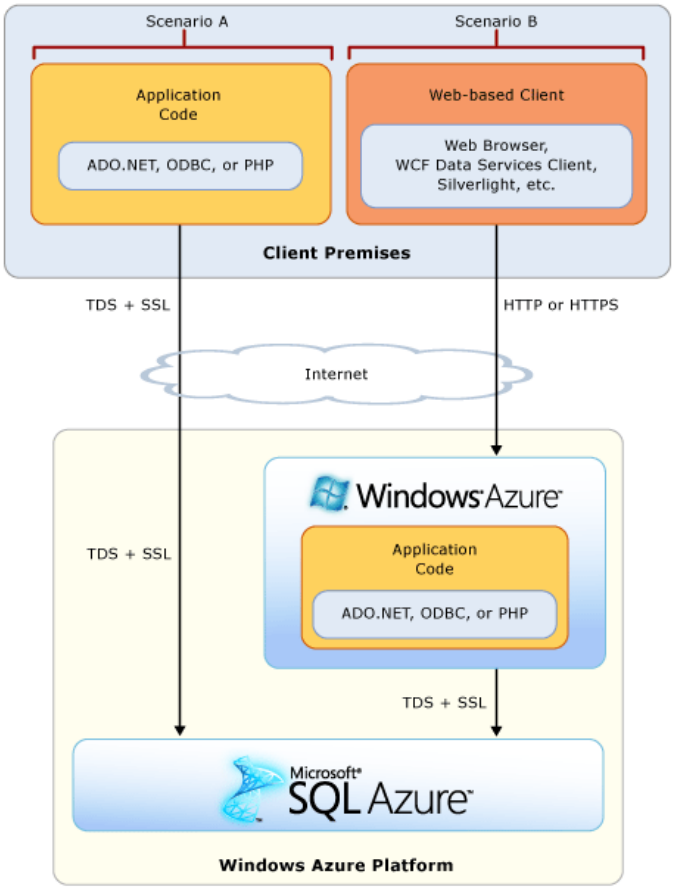

Figura 12 – Modelo de acesso aos dados SQL Azure [12]

No primeiro cenário devem ser consideradas as questões de latência de rede no acesso aos dados a partir da aplicação cliente, sendo estas mitigadas obviamente se a aplicação for migrada para Windows Azure (segundo cenário) [12].

Relativamente à taxação do serviço, é cobrada a utilização do SQL Azure e a base de dados utilizada. Existem dois tipos de BDs, *Web Edition* e *Business Edition*, podendo ambas as versões coexistir no mesmo servidor SQL Server [14]. A *Web Edition* é a indicada para pequenas aplicações *web*, uma vez que suporta base de dados com tamanho máximo de 1 a 5 GB. Para aplicações com maior necessidade em termos de armazenamento existe a *Business Edition*, suportando BDs com um tamanho máximo de 150 GB (taxadas até 50 GB com incrementos de 10 GB e a partir dos 50 GB com incrementos de 50 GB) [14]. O tipo de base de dados e tamanho máximo da mesma é indicado no momento da sua criação, podendo ser alterado posteriormente.

Para além do tipo/tamanho da(s) BD(s) utilizada(s), são também taxadas as transferências de dados que ocorrem com clientes fora da plataforma Windows Azure ou fora da região onde se encontra a BD [14]. O tamanho e número máximo de BDs de cada conta registado diariamente é aquele que irá ser contabilizado para a fatura mensal.

Tabela 6 – Análise comparativa entre SQL Server e SQL Azure (adaptado [81])

	SQL Server (Local)	**SQL Azure**
Edições Disponíveis	Express Workgroup Standard Enterprise	Web Edition Business Edition [82]
Armazenamento Dados	Sem limite	Web Edition – BDs com tamanho máximo entre 1 e 5 GB Business Edition – BDs com um tamanho máximo de 150 GB [83]
Conetividade	SQL Server Management Studio SQLCMD	SQL Server Management Studio (suportadas versões iguais ou superiores a 2008 R2) SQLCMD Portal de gestão SQL Azure [84]
Autenticação	SQL Authentication Windows Authentication	Apenas SQL Authentication
Suporte TSQL	Suportado	Parcialmente suportado: TSQL suportado ([85]) TSQL parcialmente suportado ([86]) TSQL não suportado ([87])
Database Mirroring	Suportado	Não suportado
SQL Agent	Suportado	SQL Agent e Jobs não disponíveis em SQL Azure [3]
SQL Server Integration Services (SSIS)	Suportado	SSIS não disponível na plataforma Azure [4]

[3] Em alternativa o SQL Server Agent pode correr no SQL Server local conectando-se à BD SQL Azure.
[4] Em alternativa o SSIS pode correr localmente conectando-se à BD SQL Azure.

4 Protótipo de estudo

4.1 Pré-requisitos

A ideia para o desenvolvimento de uma aplicação em Azure surgiu a partir de uma necessidade da Microsoft para divulgar e promover o lançamento do seu novo produto, o IE9 (Março de 2011). A ideia, conforme mencionado, seria criar um concurso *online* para a promoção do novo *browser*, antecedendo o seu lançamento oficial.

O protótipo a desenvolver teria que garantir genericamente que os utilizadores percorressem os passos representados no diagrama de atividades da Figura 13.

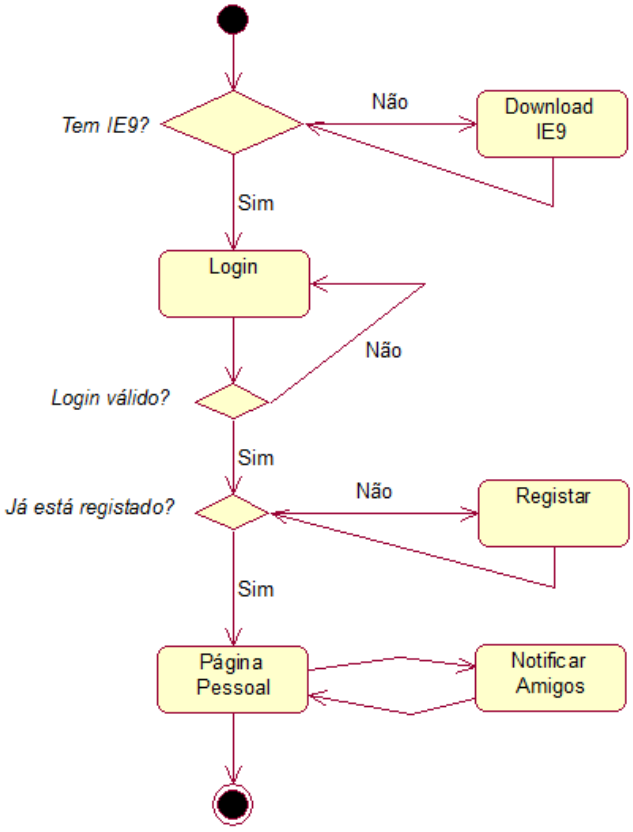

Figura 13 - Diagrama de atividades

Os utilizadores do concurso seriam subdivididos em três grupos: *Master*, *Follower* ou ambos. Tudo dependeria da forma como estes se relacionariam entre eles. A interação de um utilizador do tipo *Master* seria muito semelhante à de um *Follower*, como se descreve de seguida no processo de autenticação pretendido.

Passos para um utilizador do tipo *Master*:

- Acede à página do concurso diretamente;
- Escolhe participar no concurso;
- Se não aceder à página com o *browser* IE9, é redirecionado para uma página para efectuar o seu *download*;
- Autentica-se via Facebook: se for o primeiro acesso ao *site*, dá permissões à aplicação para aceder aos seus dados pessoais. A aplicação regista automaticamente o

concorrente com os dados obtidos via Facebook, sendo também gerado e registado um *link* pessoal para partilha.

- A aplicação regista os dados relativos ao seu acesso (ex: data e hora);
- O utilizador é reencaminhado para a sua página pessoal, tendo acesso à estatística de número de pessoas que o seguem, bem como a hipótese de partilhar o seu *link* pessoal via redes sociais ou *e-mail*.

Um utilizador do tipo *Follower*:

- Acede à página do concurso através de um *link* fornecido por um *Master*;
- Autentica-se via Facebook nos mesmos moldes que o *Master*, sendo redirecionado para uma página para o *download* do IE9 caso não esteja a aceder ao *site* através deste *browser*;
- Tratando-se do primeiro acesso, o concorrente é registado automaticamente na aplicação com os dados obtidos do Facebook, sendo também registado como *Follower* de um determinado *Master* em que a chave da relação é parte do *link* de acesso ao *site*. Caso contrário, apenas são armazenados dados relativos ao seu acesso;
- É reencaminhado para a sua página pessoal, que no caso dos *Followers* poderá ser uma página de agradecimento pela participação no concurso e incentivo à divulgação do *link* do *Master* via redes sociais ou *e-mail*.

Se um *Follower* pretender participar também no concurso para ganhar o prémio principal, bastará seguir os passos descritos para se tornar *Master*, passando a ficar com ambos os registos. O mesmo é válido para um *Master* que ajude um outro amigo, passando a ficar também com o registo de *Follower*.

A nível de concurso apenas não seriam válidas as seguintes operações (tendo estas que ser validadas pela aplicação):

- Se o utilizador já é *Master* não se pode inscrever novamente como *Master*;
- Um utilizador *Master* não pode ser seguidor de si próprio;

- Um utilizador só pode ser *Follower* uma vez (só pode seguir uma pessoa).

Definidas as ideias chave para o projeto, estabelecidos os pré-requisitos e os objetivos da aplicação começou-se a delinear o seu desenvolvimento, as principais metas a atingir e os principais problemas a mitigar, nomeadamente:

- *Processamento de pedidos dos Web Roles e Worker Roles* – que processos é que poderiam ser relegados para processamento em *background*, de forma a libertar os *Web Roles*?

- *Armazenamento da informação* – que tipo de armazenamento se utilizaria na aplicação entre as soluções disponíveis (SQL Azure, *Tables*, *Blobs* ou *Queues*)?

- *Sessões em aplicações distribuídas* – como garantir o bom funcionamento das sessões no ambiente Windows Azure?

- *Integração com Facebook* – como poderia a aplicação interligar-se com este sistema externo, de forma a obter dados dos utilizadores necessários ao seu registo no concurso e como funcionaria a sua autenticação?

- *Deploy para a Cloud* – desenvolvida a aplicação localmente quais seriam as implicações do seu envio para a *Cloud*?

4.2 Estrutura da aplicação

4.2.1 Funcionalidades requeridas

Os requisitos apresentados pressupunham um conjunto de funcionalidades a desenvolver, nomeadamente:

- Mecanismo de autenticação através da plataforma Facebook;

- Registo automático do concorrente na aplicação através de dados obtidos via Facebook;

- Registo de acessos dos utilizadores a páginas pessoais;

- Mecanismo para criação automática de *link* pessoal para partilha e possibilidade de partilha automática através das principais redes sociais (Facebook e Twitter) ou via *e-mail*;

- Estatística na página pessoal do *Master* indicando o número de *Followers* que possui.

4.2.2 Topologia

Após o estudo efetuado relativamente às funcionalidades requeridas à aplicação, adotou-se a topologia demonstrada na Figura 14.

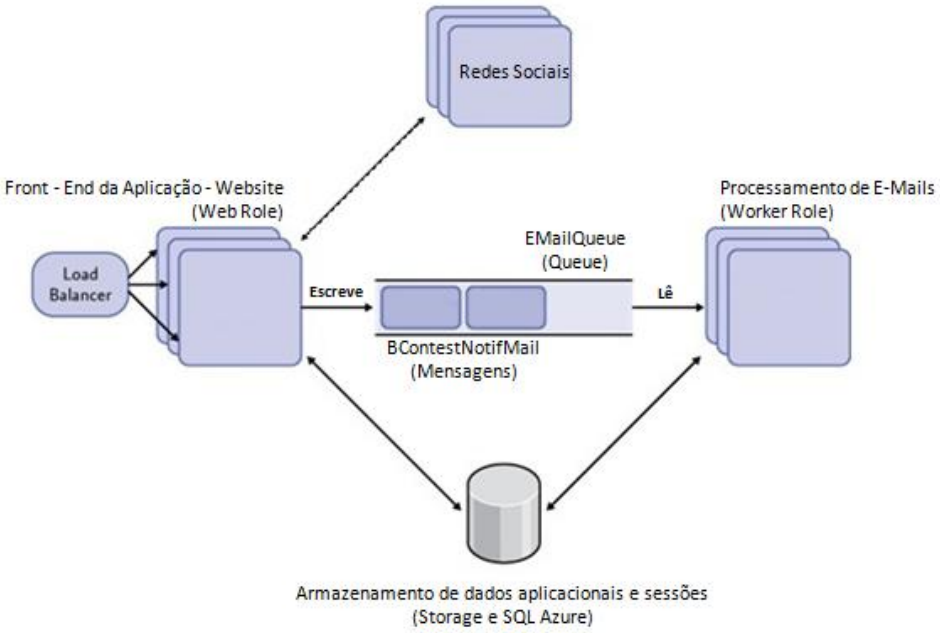

Figura 14 - Topologia da aplicação (adaptado [6])

Os diferentes componentes da topologia foram utilizados com os seguintes intuitos:

- *Web Roles* – *front-end* da aplicação. Todos os pedidos dos clientes via *browser* são processados por estes componentes;

- *Worker Roles* – processamento das tarefas a realizar em *background*. As operações de envio de *e-mails* são processadas pelos *Worker Roles* assincronamente;

- *Queues* – troca de mensagens entre *Web Roles* e *Worker Roles*, em que mensagens com objetos serializados (contendo informação para o envio de *e-mails*) são enviadas dos *Web Roles* para processamento posterior por parte dos *Worker Roles*;

- *Storage (Blobs e Tables)* – armazenamento de sessões (em *Blobs*) e de informação acerca destas (em *Tables*);

- *SQL Azure* – armazenamento de dados de utilizadores e da aplicação.

A partir da topologia apresentada, ficam evidenciados alguns aspetos funcionais da aplicação:

- O processamento assíncrono que vai envolver troca de mensagens entre os *Web Roles* e os *Worker Roles* via *Queues*, vai ser utilizado como mecanismo de envio de *e-mails*;

- Devido à particularidade deste tipo de topologias, em que poderão existir vários *Web Roles* a processar pedidos, as sessões serão armazenadas em *Blobs* e *Tables* para garantir a persistência das mesmas;

- Irá ser utilizado o SQL Azure como serviço de base de dados relacional na *Cloud*, devido às suas semelhanças com a tecnologia SQL Server, amplamente utilizada na empresa, que irão facilitar o desenvolvimento e futura manutenção da aplicação.

Conforme o descrito utilizam-se duas tecnologias de armazenamento distintas. Isto deveu-se ao fato da empresa não ter qualquer interesse em armazenar a informação temporária relativa às sessões dos utilizadores na BD. A *Storage* foi assim utilizada apenas como local de armazenamento de sessões, apoiando o mecanismo utilizado para garantir a persistência das mesmas na aplicação. Se a solução se baseasse numa arquitetura convencional, estas seriam armazenadas exclusivamente em memória. Já os dados relativos aos utilizadores/aplicação eram dados importantes que se pretendiam armazenar e com os quais possivelmente trabalhar no futuro (gerar mapas, consulta-los,..), pelo que desde o início se definiu que seriam armazenados em SQL Azure. Sendo armazenados desta forma simplificaria também a sua interligação com ferramentas de geração de *dashboards* (a partir de dados via SQL) que a empresa possui (Planos).

4.2.3 Modelo de Dados

A partir destes requisitos definiu-se o que seria necessário armazenar na base de dados (BD) para suportar a aplicação, sendo o modelo de dados o esquematizado no diagrama da Figura 15.

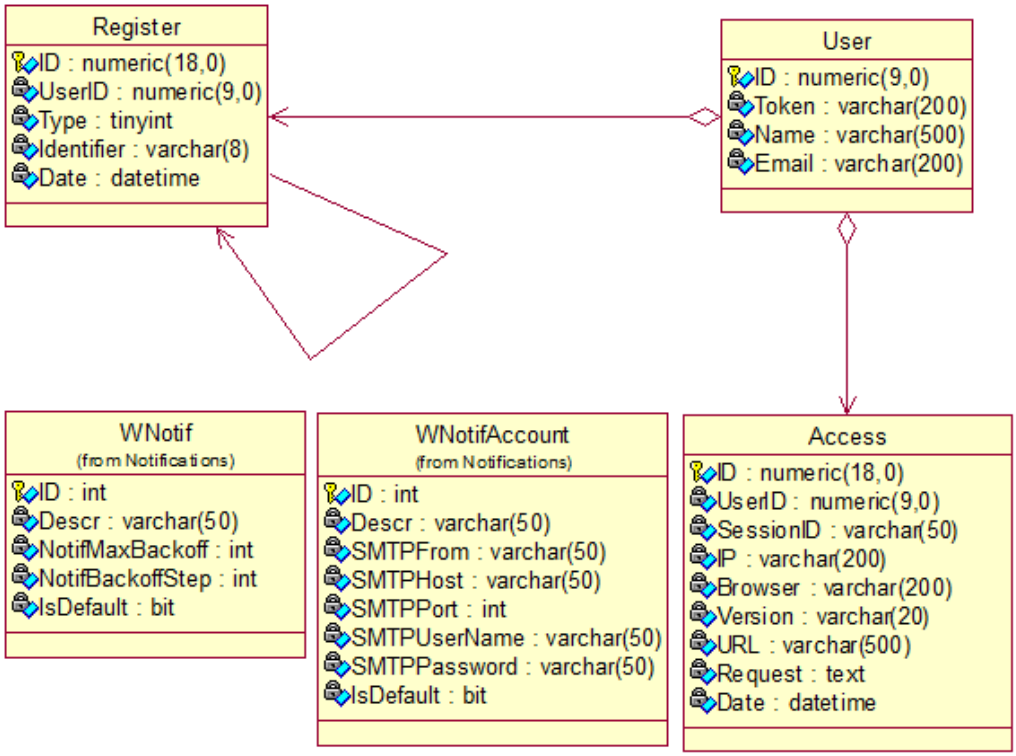

Figura 15 – Modelo de dados

As tabelas *User*, *Register* e *Access* estão diretamente associadas ao processo do concurso (registo e acesso dos utilizadores), enquanto que as tabelas *WNotif* e *WNotifAccount* estão ligadas ao processo de notificações. São de seguida detalhadas as tabelas referidas:

- *User* – Tabela utilizada para registar dados do utilizador devolvidos pelo Facebook (Identificador do utilizador ou *Token*, Nome e *E-mail*). Só quando se obtém um *token* de acesso é que é possível aceder a dados privados do utilizador e executar tarefas em seu nome, desde que devidamente autorizadas pelo mesmo.

ID	Token	Name	Email
8	1000[redacted]166[redacted]	Pedro [redacted]	p[redacted]@gmail.com
9	13109[redacted]6[redacted]	Ricardo [redacted]	m[redacted]@hotmail.com

Figura 16 – Dados armazenados na tabela *User*

- *Register* – Tabela utilizada para registar relações entre os utilizadores. É nesta tabela que se definem as relações entre utilizadores *Master* e *Follower*. Cada utilizador possui um identificador que o relaciona com a tabela *User* (*UserID*) e um identificador externo (*Identifier*), que constitui parte do *link* divulgado pelo *Master* aos seus *Followers*. O campo *Type* é utilizado para identificar o tipo de utilizador na relação (1 - *Master* ou 2 - *Follower*), sendo a chave da relação o identificador externo.

  Analisando a Figura 17 pode-se concluir que o concorrente com o *UserID* 1 é *Master* e tem como *Followers* os concorrentes 3 e 4.

ID	UserID	Type	Identifier	Date
1	1	1	5[redacted]65357	2011-03-15 09:29:26.697
2	2	1	0[redacted]19tbg	2011-03-15 09:34:11.150
3	3	2	5[redacted]65357	2011-03-15 13:19:32.303
4	4	2	5[redacted]65357	2011-03-15 14:05:42.790

Figura 17 - Dados armazenados na tabela *Register*

- *Access* – Tabela utilizada para registar informação acerca dos acessos dos utilizadores ao *site*. Esta tabela vai servir de prevenção contra fraudes nas participações, servindo para a Microsoft como prova de validação da participação vencedora. Nesta tabela são registados o *ID* da sessão, a data e baseado no pedido do cliente: o ip remoto, o *browser*, a versão do *browser* e o URL acedido. Todas as variáveis presentes na

coleção *Server Variables* [39] que fornece informações sobre o servidor, a conexão com o cliente e o pedido atual da conexão, são introduzidas no campo *Request*.

ID	UserID	SessionID	IP	Browser	Version	URL	Request	Date
1	1	qhv2k4hbemwqwaabtx1bk2pc	[redacted]17.35	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_ACCEPT:...	2011-03-15 09:29:27.953
2	2	wwedpyzvle1lffzufnnbb2w0	[redacted].222.136	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_ACCEPT:...	2011-03-15 09:34:12.330
3	1	qhv2k4hbemwqwaabtx1bk2pc	[redacted]17.35	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_ACCEPT:...	2011-03-15 09:48:46.247
4	1	qhv2k4hbemwqwaabtx1bk2pc	[redacted]17.35	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_ACCEPT:...	2011-03-15 10:36:16.937
5	1	qhv2k4hbemwqwaabtx1bk2pc	[redacted]17.35	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_ACCEPT:...	2011-03-15 10:37:08.337
6	1	qhv2k4hbemwqwaabtx1bk2pc	[redacted].127.12	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_PRAGMA...	2011-03-15 13:12:27.537
7	3	dfaywek0tmu1ixldlsa3o3g	[redacted].158.68	IE	9.0	http://www.voanaweb.com/Contest/Follower.aspx?v=5[redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_VIA:1.1 S...	2011-03-15 13:19:33.250
8	1	qhv2k4hbemwqwaabtx1bk2pc	[redacted].127.12	IE	9.0	http://www.voanaweb.com/Contest/Status.aspx [redacted]	ALL_HTTP : HTTP_CONNECTION:Keep-Alive HTTP_PRAGMA...	2011-03-15 14:04:32.750
9	4	dfudjnmybghqsfked3eqk2gd	[redacted].127.12	IE	9.0	http://www.voanaweb.com/Contest/Follower.aspx?v=5[redacted]	ALL_HTTP : HTTP_CACHE_CONTROL:no-cache HTTP_CON...	2011-03-15 14:05:44.120

Figura 18 - Dados armazenados na tabela *Access*

- *WNotifAccount* – Tabela de configuração que contém dados de acesso à conta de SMTP (*Simple Mail Transfer Protocol*) para o envio de notificações via *e-mail*. Contém o endereço de *e-mail* do remetente, o host e a porta utilizados para as transações STMP (no caso concreto, smtp.exchange.telepac.pt, porta 25), bem como as credenciais para autenticar o remetente. O registo marcado como pré-definido (*IsDefault*) será o utilizado para o envio de *e-mails*.

ID	Descr	SMTPFrom	SMTPHost	SMTPPort	SMTPUserName	SMTPPassword	IsDefault
1	B-Simple Principal	bquality@b-simple.pt	smtp.exchange.telepac.pt	25	[redacted]	[redacted]	1

Figura 19 - Dados armazenados na tabela *WNotifAccount*

- *WNotif* – Tabela de configuração que contém os detalhes para o mecanismo de *backoff* implementado a nível dos *Worker Roles*. Os *Worker Roles* utilizam um mecanismo de *polling* [42] para verificar com determinada frequência se existem mensagens na *queue*. Quando os *Worker Roles* não encontram mensagens para processar, incrementam as unidades definidas no *NotifBackoffStep* (segundos) diminuindo a frequência de verificação até um máximo predefinido em *NotifMaxBackoff* (segundos). Processo detalhado no tópico 4.8.2.

ID	Descr	NotifMaxBackoff	NotifBackoffStep	IsDefault
1	Normal	3600	1	1

Figura 20 – Dados armazenados na tabela *WNotif*

4.2.4 Esquema de Páginas

Relativamente ao esquema de páginas, o *site* do concurso possui duas *Master Pages* que definem a estrutura da aplicação*: o Site.Master e o Content.Master.*

O *Site.Master* contém a parte externa da página contendo o título, *links* úteis da Microsoft e os botões de partilha da página para Facebook e Twitter. O *Content.Master* que possui o *Site.Master* como *Master Page,* é a base de desenvolvimento das restantes páginas (Figura 21).

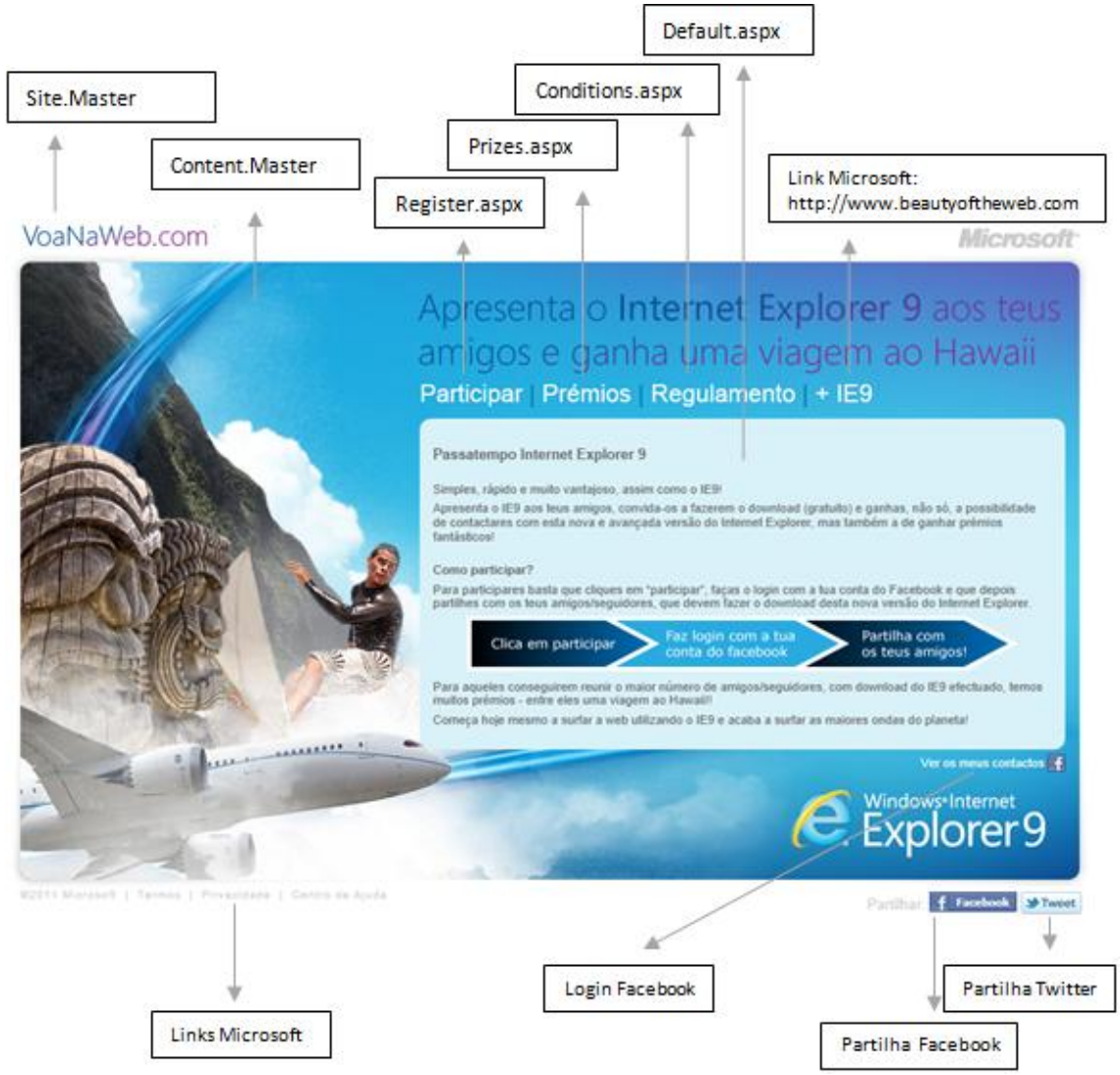

Figura 21 – Página Default.aspx

A solução possui um mecanismo genérico para registo estatístico da utilização do *site*, que se baseia no Google Analytics [43]. Este mecanismo permite, através de um bloco de código *JavaScript* introduzido nas páginas (neste caso no *Site.Master* - Figura 22), o registo de dados

dos visitantes e o seu envio para o servidor do Google Analytics. Tendo uma determinada conta associada, conseguem-se obter *dashboards* que representam toda a informação acerca da utilização do *site*[5].

```
<script type="text/javascript">
    var _gaq = _gaq || [];
    _gaq.push(['_setAccount', 'UA-217[redacted]2-1']);
    _gaq.push(['_trackPageview']);

    (function () {
        var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
        ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
        var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
    })();
</script>
```

Figura 22 – *Script* para Google Analytics (*Site.Master*) [44]

Tal como já referido, o *site* possui também um mecanismo próprio de registo de acessos por parte dos utilizadores, através do preenchimento da tabela *Access*. O registo desta tabela só ocorre caso o utilizador navegue para uma das páginas que exigem autenticação, ou seja a *Status.aspx* (página pessoal de um *Master*) ou a *Follower.aspx* (página pessoal de um *Follower*).

Para o funcionamento deste mecanismo, as páginas da aplicação herdam de uma página específica *BPage* que regista estas informações através do evento *Init*. Desta forma, para além de permitir avaliar de uma forma centralizada se o utilizador está a tentar aceder às páginas do concurso que exigem IE9 registam, no caso de acesso a uma das páginas referidas, dados informativos acerca do cliente (Figura 23). Estas informações contêm, conforme mencionado, o id da sessão, o ip remoto, *browser*, versão do *browser*, o URL a que o utilizador acedeu, bem como todas as variáveis presentes na coleção *Server Variables* [39].

[5] Por questões de privacidade não é possível divulgar as estatísticas do voenaweb.com. O Anexo 1 – Google Analytics contém um exemplo dos *dashboards* gerados pelo Google Analytics para um outro *site*, em que se pode analisar entre outros, o número de visitas, a variação do número de visitas, o número de visualizações de cada página, o local de acesso dos utilizadores, o browser e o *service provider* destes.

```
Dim bcontest As New BContest
Dim requestVariables As New StringBuilder
For Each serverVariable As String In Request.ServerVariables
    If requestVariables.Length = 0 Then
        requestVariables.AppendFormat("{0} : {1}", serverVariable, Request.ServerVariables(serverVariable))
    Else
        requestVariables.AppendFormat("{0}{1} : {2}", vbCrLf, serverVariable, Request.ServerVariables(serverVariable))
    End If
Next

bcontest.SetAccess(Session.SessionID, _
                   Request.ServerVariables("REMOTE_ADDR"), _
                   Request.Browser.Browser, _
                   Request.Browser.Version, _
                   Request.Url.ToString, _
                   requestVariables.ToString)
```

Figura 23 – Extrato de código - evento *Init* (*BPage*)

4.3 Desenvolvimento

4.3.1 Desenvolvimento Azure – particularidades

A criação de um projeto para a *Cloud* e mais concretamente para Windows Azure, tem algumas particularidades face a um projeto *web* comum. Isto deve-se ao fato da plataforma ter sido desenvolvida para suportar uma elevada carga de utilização não comprometendo o sistema e desacoplando os seus diferentes componentes.

Utilizando o IDE Microsoft Visual Studio para o desenvolvimento aplicacional, deverão ser adicionadas a nível do projeto as seguintes referências, para suporte da plataforma Azure [40]: *Microsoft.WindowsAzure.Diagnostics*, *Microsoft.WindowsAzure.ServiceRuntime* e *Microsoft.WindowsAzure.StorageClient*.

Adicionalmente no *Global.asax* deverão ser declarados os *namespaces* *Microsoft.WindowsAzure* e *Microsoft.WindowsAzure.ServiceRuntime*, colocando-se no *aplication_start* a seguinte instrução [40]:

```
CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName,
configSetter)
configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))
```

Esta instrução define o editor global de configuração para a conta de armazenamento, simplificando o acesso aos seus atributos. Colocando esta instrução no *Global.asax*, as aplicações apenas necessitam de o definir uma vez, podendo em qualquer parte da aplicação aceder à conta de armazenamento, através da indicação do atributo que pretendem obter (neste caso o valor da *DataConnectionString*):

```
CloudStorageAccount account =
CloudStorageAccount.FromConfigurationSetting("DataConnectionString");
```

Quando é utilizado este método (`CloudStorageAccount.FromConfigurationSetting`), o Azure procura um editor de configuração. Devido à chamada executada no *Global.asax*, `SetConfigurationSettingPublisher`, este método é encontrado e utilizando o `RoleEnvironment.GetConfigurationSettingValue`, obtêm-se os valores da conta de armazenamento [41], previamente definidos para a instância.

Quando uma aplicação Azure corre em ambiente de desenvolvimento, um emulador reproduz as funcionalidades da *Cloud* localmente, permitindo a observação do estado das instâncias de computação definidas na configuração da aplicação (Figura 24), bem como o controle do estado da *Storage* (Figura 25).

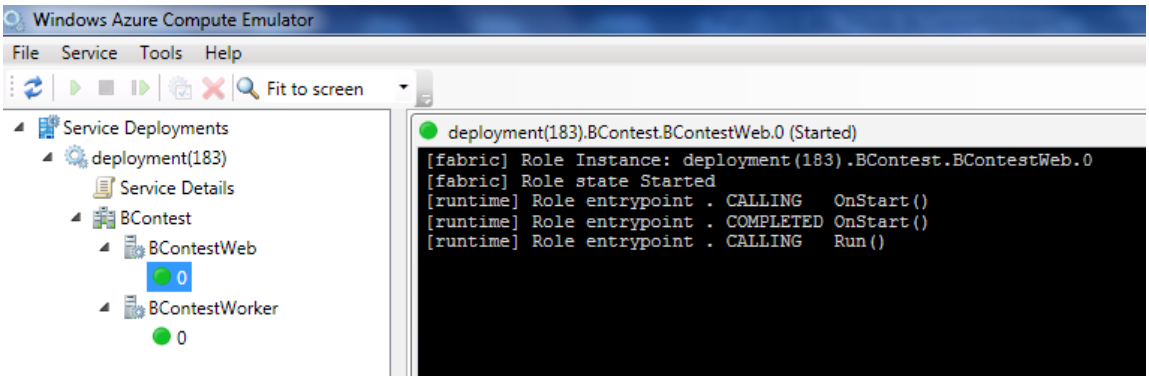

Figura 24 – Visualização das instâncias de computação inicializadas

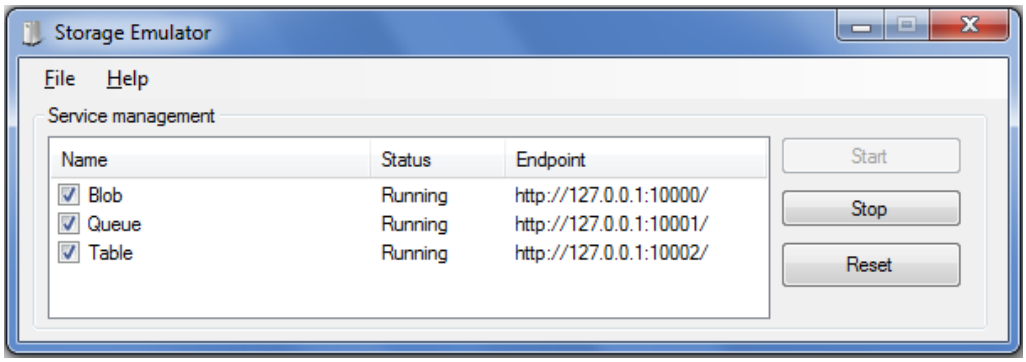

Figura 25 - Visualização do estado da *Storage* local

4.3.2 Estrutura da solução protótipo

No início do desenvolvimento, criou-se um novo projeto do tipo Windows Azure, indicando que se pretendia utilizar instâncias do tipo *Web Role* e instâncias do tipo *Worker Role*, conforme o ilustrado na Figura 26.

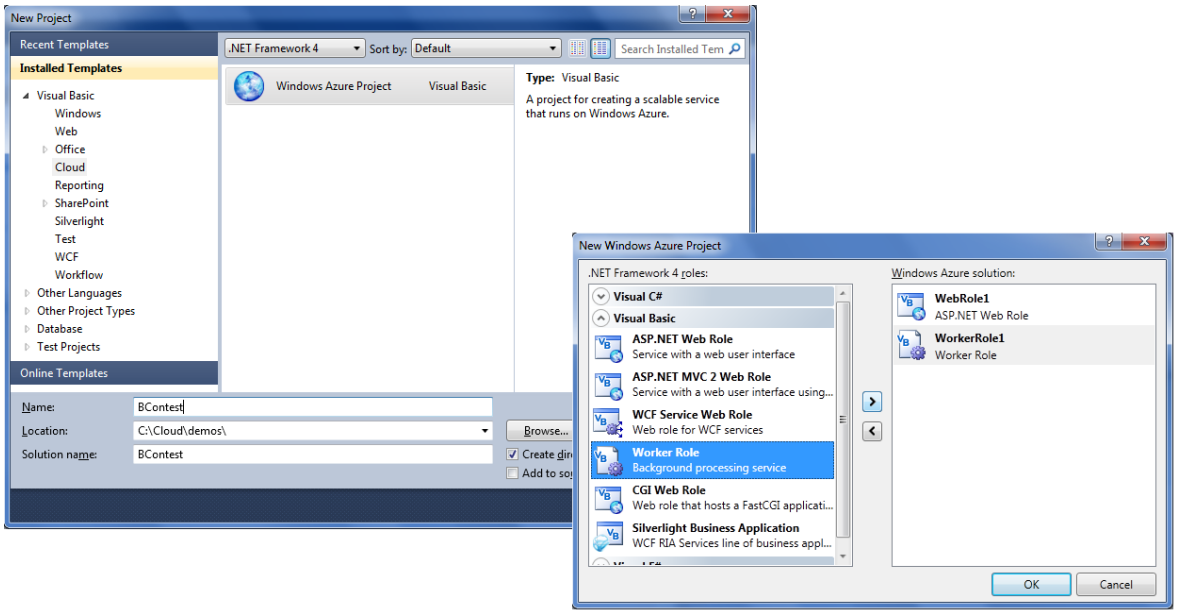

Figura 26 – Criação de um projeto Windows Azure

Desta forma, foi criada uma nova solução com a estrutura base para correr em Windows Azure (Figura 27), constituida por três projetos principais:

- *BContest* – que contém os *roles* da aplicação e a sua configuração através dos ficheiros *ServiceDefinition.csdef* e *ServiceConfiguration.cscfg* que determinam como a

aplicação é constituida e como irá correr na *Cloud* quando for feito o *deploy* da mesma;

- *BContestWeb* – que constitui o projeto *Web*, responsável por dar resposta ao *front-end* da aplicação;

- *BContest Worker* – que constitui o serviço responsável pelo processamento de tarefas em *backgroud*.

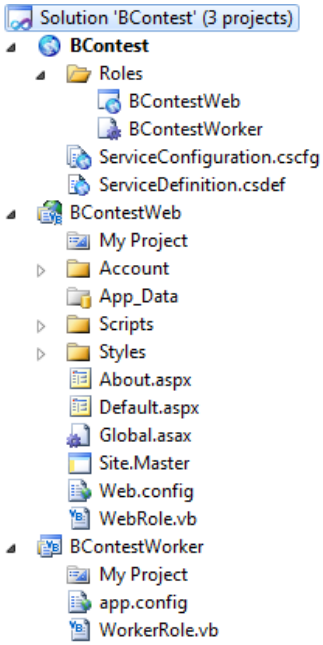

Figura 27 - Estrutura inicial de projeto em Windows Azure

Os *Web Roles* e *Worker Roles* podem ser facilmente configurados, de forma a adaptar a sua capacidade de processamento/armazenamento às necessidades de cada aplicação, bem como a sua ligação à conta de armazenamento na *Cloud*.

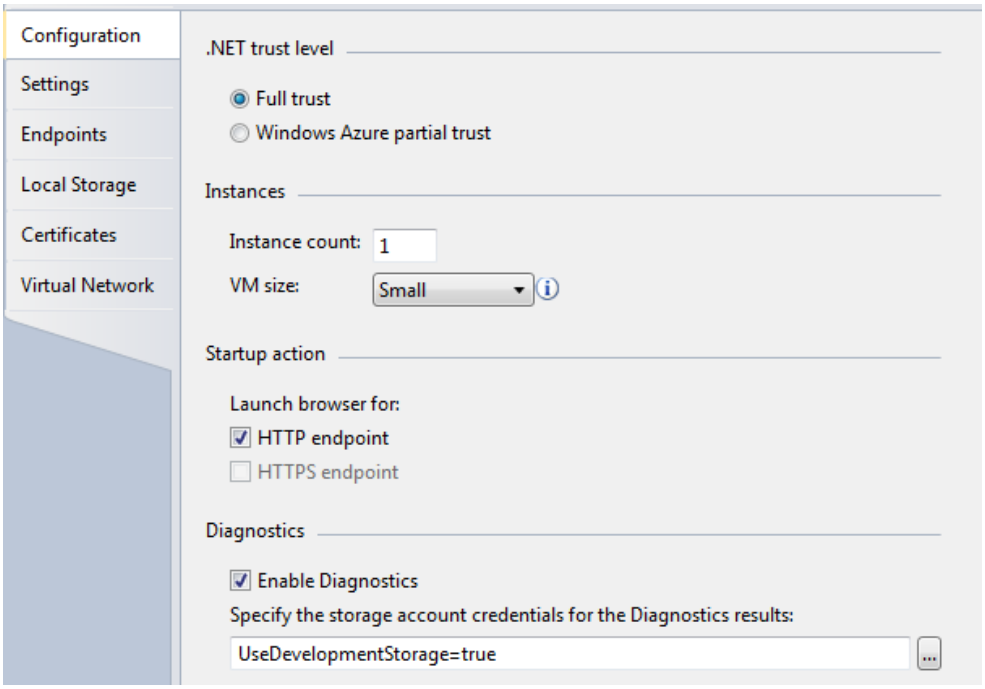

Figura 28 – Configuração do *Web Role* (*Configuration*)

Na Figura 28 é ilustrada a interface de configuração do *Web Role,* em que é possível definir o número de instâncias necessárias para o processamento de pedidos, bem como a capacidade de cada uma (Tabela 5). O modo de diagnóstico também se encontra ativo, permitindo obter um *log* das principais ocorrências registadas pelo *Web Role*, neste caso, em ambiente de desenvolvimento. *Web Roles* e *Worker Roles* possuem detalhes de configuração distintos, permitindo adaptar a capacidade de cada um dos componentes face à carga que cada um terá: se a aplicação possuir elevada carga a nível de *front-end*, os *Web Roles* deverão ter uma maior capacidade, caso contrário, se o processamento principal na aplicação recai sobre tarefas de *background,* deverão ser os *Worker Roles* aqueles que possuem maior capacidade.

Na configuração do *Web Role* (em *Settings*) podem também ser definidos os dados de acesso à conta de armazenamento na *Cloud* (Figura 29).

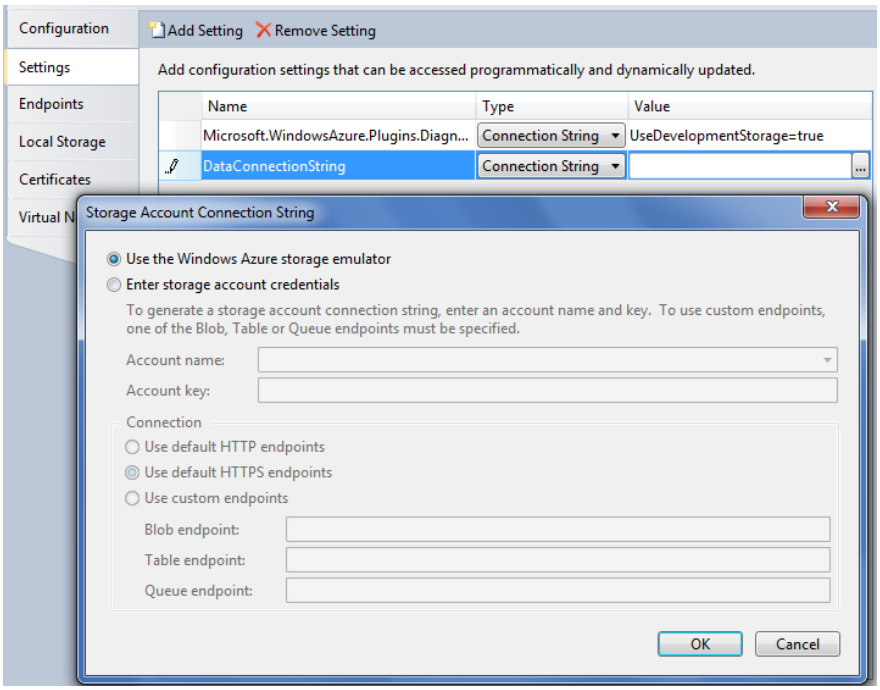

Figura 29 - Configuração do *Web Role* (*Settings*)

Para isto, basta adicionar um novo detalhe a esta tabela (*Add Setting*), atribuir-lhe um nome e um valor. Este valor poderá inicialmente ser o emulador local, que simula o ambiente *Cloud* localmente, sendo posteriormente alterado para a conta real mediante a introdução do nome e chave da conta. A este detalhe foi atribuído o nome *DataConnectionString*, nome do atributo da *connection string* que por predefinição está configurado no projeto *AspProviders* (pertencente ao *Windows Azure Platform Training Kit*) [40], componente que irá ser utilizado para a gestão de sessões. Por predefinição os *roles* já possuem um detalhe (*Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString*) que define o local utilizado para iniciar o diagnóstico da instância.

As configurações aplicadas traduzem-se nos ficheiros de configuração da aplicação referidos no tópico 3.1 (*ServiceDefinition.csdef* e *ServiceConfiguration.csc*).

4.4 Sessões em Windows Azure

4.4.1 Contexto

As aplicações *web* baseiam-se no protocolo de comunicação HTTP, o que implica especial cuidado com a gestão de sessões, uma vez que este constitui um protocolo *stateless* [48].

Uma das formas utilizadas para a gestão das sessões por parte do servidor é o armazenamento do estado destas em memória. Tal como nas aplicações ASP.NET comuns, também em Azure instruções de atribuição de um objeto a uma sessão e retorno desse objeto são possíveis:

```
Session("BProfile") = New BProfile(..)
Dim profile As BProfile = CType(Session("BProfile"), BProfile)
```

Porém, em Azure, tal como noutra qualquer infraestrutura em que haja mais que uma instância da aplicação *web*, o armazenamento de sessões em memória não é viável.

Como os pedidos do utilizador são processados por um *load balancer* antes de serem reencaminhados para um determinado *Web Role*, caso se utilize mais que uma instância deste tipo, o Azure não garante o reencaminhamento dos pedidos para a mesma instância [48]. Isto porque a principal função do *load balancer* é precisamente essa, reencaminhar uniformemente o tráfego mediante a carga verificada.

Como cada instância *Web Role* possui as suas próprias sessões, não existe nenhum mecanismo de partilha entre elas. Para além deste problema causado pelo *load balancer*, em caso de falha de um *Web Role*, o Azure estabelece uma nova instância noutro servidor, perdendo-se desta forma as informações de sessão armazenadas em memória na instância [48].

Assim sendo, a solução passa por armazenar a informação de sessão na *Storage*, mais concretamente em *Tables* e *Blobs*. Desta forma, a informação relativa às sessões está constantemente disponível para qualquer *Role*, garantindo-se a conservação da informação mesmo em caso de falha, devido à política de replicação de dados da *Storage* do Azure [50].

4.4.2 Solução

Para se obter um mecanismo fiável de gestão de sessões com recurso à *Storage*, recorreu-se ao projeto *AspProviders* (*Windows Azure Platform Training Kit*) [40].

O *AspProviders* foi acrescentado à solução e adicionada uma referência para este no projeto *BContestWeb*. Para a utilização deste componente, também o *webconfig* do *BContestWeb* foi alterado, acrescentando-se a configuração demonstrada na Figura 30.

```
<system.web>
  ...
  <sessionState mode="Custom" customProvider="TableStorageSessionStateProvider">
    <providers>
      <clear />
      <add name="TableStorageSessionStateProvider"
           type="Microsoft.Samples.ServiceHosting.AspProviders.TableStorageSessionStateProvider"
           applicationName="BContestWeb" />
    </providers>
  </sessionState>
 ...
</system.web>
```

Figura 30 – Configuração para utilização do componente *AspProviders* (*webconfig*)

Desta forma definiu-se que o *session provider* que irá ser utilizado será o *TableStorageSessionStateProvider*, classe presente no projeto *AspProviders*.

Utilizando este componente, as sessões passam a ser armazenadas nas *Windows Azure Tables* e *Windows Azure Blobs*. As *Blobs* são utilizadas para o armazenado do estado das sessões, enquanto que as *Tables* são utilizadas para armazenamento de informação de configuração acerca das sessões (data, nome da aplicação, nome da *Blob* que armazena a sessão, data de criação, data de expiração,..) [49].

O projeto *AspProviders* possui uma classe de configuração onde é possível definir, entre outros, o nome do atributo da *connection string* da aplicação (predefinição: *DataConnectionString*), o nome do *container* da *Blob* onde serão armazenadas as sessões (predefinição: *sessionprovidercontainer*) e o nome da tabela onde serão armazenadas as informações de configuração (predefinição: *Sessions*).

Com a alteração efetuada a nível do *Global.asax* (tópico 4.3.1), o *AspProviders* consegue aceder ao valor da *connection string* que se definiu a nível do *Web Role*, de forma a obter toda a informação necessária para aceder à *Storage* e gerir escrita e leitura de sessões a esse nível:

```
CloudStorageAccount account =
Configuration.GetStorageAccount(Configuration.DefaultStorageConfigurationSt
ring)
```

Na Figura 31 encontram-se detalhes acerca das sessões na tabela *Sessions* e na Figura 32 as sessões efetivamente armazenadas no *container sessionprovidercontainer*.

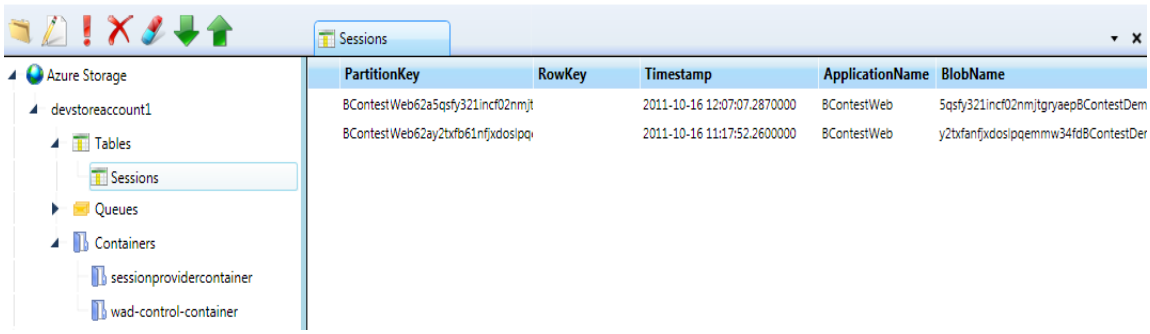

Figura 31 – Visualização de conteúdo: tabela *Sessions* (ferramenta Celebrata Cloud Storage Studio)

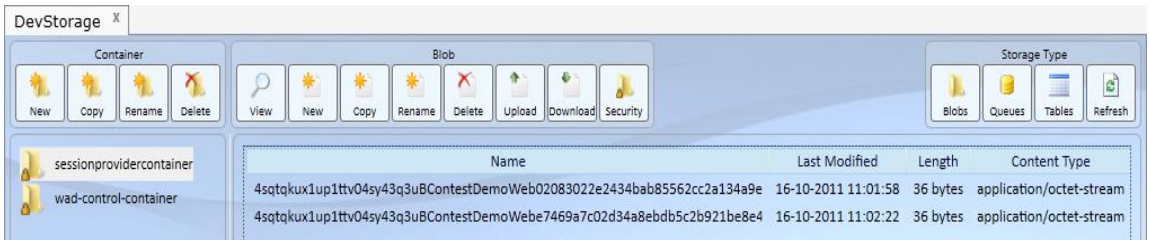

Figura 32 - Visualização de conteúdo: *container sessionprovidercontainer* (ferramenta Azure Storage Explorer)

Para validar a correta implementação deste mecanismo de armazenamento de sessões dever-se-á, por exemplo, simular uma reinicialização do *Web Role* quando se visualiza uma página que exija a existência de uma sessão ativa (ex: página com o nome do utilizador autenticado).

Basta para isso no emulador de computação (Figura 24) suspender o *Web Role* e na consola de comandos do Windows e invocar o comando *iisreset*. Estes passos simulam o que aconteceria na *Cloud* caso uma instância fosse reiniciada.

Iniciando a instância e forçando a atualização da página, será carregada a sessão armazenada na *Storage* (ex: indicando o nome do utilizador autenticado). Se as sessões apenas se encontrassem armazenadas em memória, ao ser parada a instância, todos os estados relativos a estas seriam perdidos.

Com este mecanismo funcional, foi utilizada uma classe (*BProfile*) que representa o perfil do utilizador (Figura 33).

```
<Serializable()>
Public Class BProfile

    Property AccessKey As Guid = Guid.NewGuid
    Property UserID As Decimal = Nothing
    Property Name As String = String.Empty
    Property Email As String = String.Empty
    Property IdentifierMaster As String = Nothing
    Property IdentifierFollower As String = Nothing
    Property NameMaster As String = Nothing
    Property IsMaster As Boolean = False
    Property IsFollower As Boolean = False
    Property IsAuthenticated As Boolean = False

    Public Property ConnectionString As String = System.Configuration.ConfigurationManager.ConnectionStrings("BContest").ConnectionString

    Private Sub New()
    End Sub

    Public Shared ReadOnly Property GetProfileFromSession() As BProfile
        Get
            If (HttpContext.Current.Session("BProfile") Is Nothing) Then
                HttpContext.Current.Session("BProfile") = New BProfile()
            End If
            Return CType(HttpContext.Current.Session("BProfile"), BProfile)
        End Get
    End Property
```

Figura 33 – Propriedades da classe *BProfile*

Deste modo, é possível em qualquer ponto da aplicação aceder às variáveis de sessão e caso esta não exista, é inicializada com valores predefinidos:

```
If BProfile.GetProfileFromSession.IsAuthenticated Then
…
```

Esta classe incorpora também operações de *login*, *logout* e registo de utilizador, atualizando neste caso o seu perfil.

4.5 Autenticação e Registo na aplicação - integração com Facebook

O processo de autenticação / registo no *site* do concurso baseia-se em dados provenientes do sistema Facebook, com o qual a aplicação interage.

4.5.1 Contexto

O Facebook utiliza o protocolo *OAuth 2.0* para autenticação e autorização de acesso [53], recorrendo-se à *Graph API* [54] para interação com este sistema. A sua implementação compreende três passos essenciais: autenticação do utilizador, autorização da aplicação e autenticação da aplicação:

- *Autenticação do utilizador* – permite que o utilizador se autentique no Facebook;
- *Autorização da aplicação* – permite que o utilizador saiba exatamente que tipo de dados e que capacidade fornece à aplicação;
- *Autenticação da aplicação* – garante que o utilizador está a dar a sua informação à aplicação específica e não a outra.

O esquema presente na Figura 34 representa graficamente o processo de autenticação.

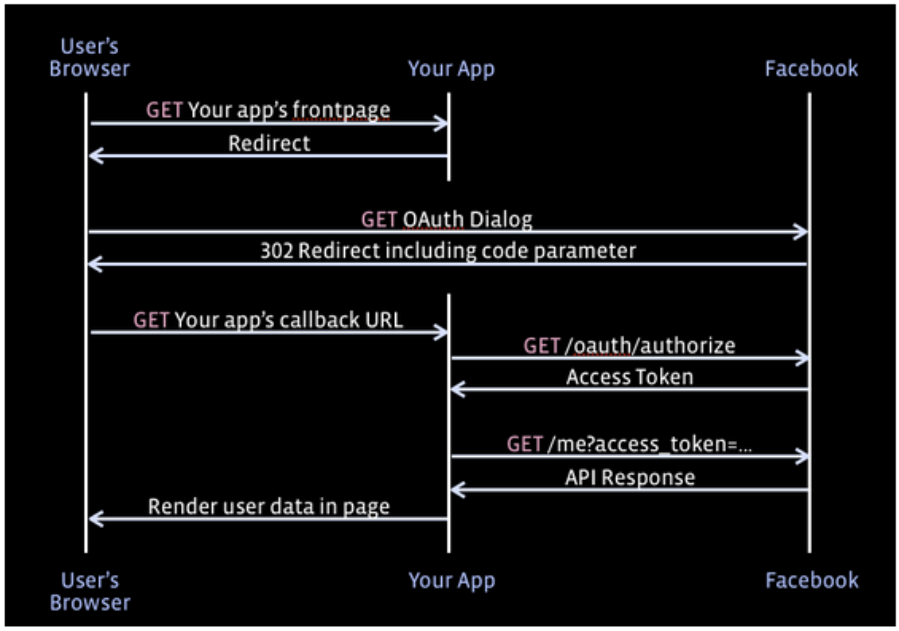

Figura 34 - Processo de autenticação no Facebook [53]

Quando se concluem estes três passos, é emitido um *token* de acesso, que permite à aplicação aceder às informações do utilizador do Facebook e efectuar operações em seu nome, se este assim o autorizar [53].

4.5.2 Solução

Pretendia-se com a autenticação/registo, que os utilizadores se autenticassem com a sua conta do Facebook, obtendo-se alguns dados necessários para o registo no concurso (primeiro/último nome e *e-mail*).

Os utilizadores teriam a possibilidade de se autenticarem diretamente clicando em "Ver os meus contactos", ou através da página de participação através do botão "*Login*" (Figura 35). Se o utilizador seguisse um *link* de um *Master*, seria diretamente reencaminhado para a página de participação, contendo o regulamento do concurso.

Figura 35 – Página de participação (Register.aspx)

Para o desenvolvimento de uma aplicação que interaja com o Facebook, deve-se previamente registar a mesma [55]. Uma vez registada a aplicação, são gerados códigos de acesso, que permitem a interação entre esta e o sistema Facebook: *App Key* e *App Secret*. Estes códigos foram acrescentados ao *webconfig* da aplicação:

```
<appSettings>
  <add key="fb_appid" value="184925684[redacted]9" />
  <add key="fb_secret" value="aeb615010becc38f20480724[redacted]6" />
  <add key="url" value="http://www.voanaweb.com" />
</appSettings>
```

Figura 36 – *App Key* e *App Secret* para interação com Facebook (*webconfig*)

Com base na documentação do Facebook [53] e [54], do protocolo *OAuth 2.0* e diferentes abordagens à interação com a API [56], [57] e [59], desenvolveu-se uma classe para gerir o processo de comunicação com este sistema, para autenticação dos utilizadores (Figura 37).

```
Public Class Facebook

    Public Const AUTHORIZE As String = "https://graph.facebook.com/oauth/authorize"
    Public Const ACCESS_TOKEN As String = "https://graph.facebook.com/oauth/access_token"
    Public Const CALLBACK_URL As String = "/Authentication/Facebook.ashx"

#Region "Properties"

    Property Server As String = "{0}"
    Public Property ConsumerKey() As String = String.Empty
    Public Property ConsumerSecret() As String = String.Empty
    Public Property Token() As String = String.Empty

#End Region

    Public Function GetAuthorizationLink() As String
        Return String.Format("{0}?client_id={1}&redirect_uri={2}&scope=email", _
                                    AUTHORIZE, _
                                    Me.ConsumerKey, _
                                    String.Format(Server, CALLBACK_URL))
    End Function

    Public Sub GetAccessToken(ByVal authToken As String)
        Me.Token = authToken
        Dim accessTokenUrl As String = String.Format("{0}?client_id={1}&redirect_uri={2}&client_secret={3}&code={4}", _
                                                     ACCESS_TOKEN, _
                                                     Me.ConsumerKey, _
                                                     String.Format(Server, CALLBACK_URL), _
                                                     Me.ConsumerSecret, authToken)

        Dim response As String = WebRequest(accessTokenUrl)

        If response.Length > 0 Then
            Dim qs As NameValueCollection = HttpUtility.ParseQueryString(response)

            If qs("access_token") IsNot Nothing Then
                Me.Token = qs("access_token")
            End If
        End If
    End Sub
```

Figura 37 – Classe *Facebook* (parcial) – métodos necessários para obter *token* de acesso

Quando um utilizador clica no botão para se autenticar, o primeiro método *GetAuthorizationLink* é invocado, sendo previamente preenchidas as propriedades da classe *Facebook* (*ConsumerKey*, *ConsumerSecret* e *Server*) a partir dos dados presentes no *webconfig* (Figura 38).

```
Dim oAuth As New BAuthentication.Facebook
oAuth.ConsumerKey = System.Configuration.ConfigurationManager.AppSettings("fb_appid")
oAuth.ConsumerSecret = System.Configuration.ConfigurationManager.AppSettings("fb_secret")
If Not Request.QueryString("v") Is Nothing Then
    oAuth.Server = String.Format("{0}{{0}}?v={1}", System.Configuration.ConfigurationManager.AppSettings("url"), Request.QueryString("v"))
Else
    oAuth.Server = String.Format("{0}{{0}}", System.Configuration.ConfigurationManager.AppSettings("url"))
End If

Response.Redirect(oAuth.GetAuthorizationLink())
```

Figura 38 – Evento botão de *login*

O método *GetAuthorizationLink* devolve um *link* que reencaminha o utilizador para uma nova página, de forma a permitir a autenticação no sistema Facebook. No URL são enviados os seguintes parâmetros:

- *Cliente_id* – código *App Key* fornecido pelo Facebook quando a aplicação foi registada;
- *Redirect_uri* – URL para o qual o utilizador será reencaminhado após autenticação no Facebook;
- *Scope* – para se obter informação extra acerca do perfil do utilizador (neste caso para se obter o *e-mail*). A informação básica a que se tem acesso sem a utilização do *scope* é constituída por: identificador do utilizador, nome, foto de perfil, sexo, faixa etária, local, lista de amigos e informação que o utilizador tenha considerado pública [58].

Exemplo do *link* obtido para um *Master*:

```
https://graph.facebook.com/oauth/authorize?
        client_id=184925XXXXXXX29&
        redirect_uri=http://www.voanaweb.com/Authentication/Facebook.ashx&
        scope=email
```

A página do Facebook (*OAuth Dialog*) surge assim para autenticação do utilizador e, caso se trate do primeiro acesso, para este autorizar a aplicação a aceder às suas informações e efetuar determinadas operações (Figura 39).

Figura 39 – Autenticação no Facebook

Se o utilizador introduzir os dados de acesso e autorizar a aplicação a aceder aos seus dados, os primeiros dois passos da autenticação estão concluídos. O utilizador é reencaminhado para a página especificada no *Redirect_uri* com um parâmetro *code* devolvido pelo Facebook. Neste caso é redirecionado para a *callback* "/Authentication/Facebook.ashx", que constitui um *http handler* [59] que vai gerir o restante processo de autenticação e registo no concurso (Figura 40).

```
If context.Request("code") Is Nothing Then
    context.Response.Redirect(oAuth.GetAuthorizationLink())
Else
    oAuth.GetAccessToken(context.Request("code"))
    Dim fb As BAuthentication.Facebook.FacebookUser = oAuth.GetUser()

    If oAuth.Token.Length > 0 Then
        If Not BContestLibraries.BProfile.GetProfileFromSession.Login(fb.id, fb.name, fb.email) Then
            context.Response.Redirect("~/")
        End If

        Dim bcontest As New BContestLibraries.BContest
        If Not bcontest.Register(context.Request.QueryString("v")) Then
            context.Response.Redirect("~/")
        End If

        If Not context.Request.QueryString("v") Is Nothing Then
            context.Response.Redirect(String.Format("~/Contest/Follower.aspx?v={0}", _
                                                    BContestLibraries.BProfile.GetProfileFromSession.IdentifierFollower), _
                                                    True)

        ElseIf context.Request.QueryString("v") Is Nothing _
                AndAlso BContestLibraries.BProfile.GetProfileFromSession.IsFollower = True _
                AndAlso BContestLibraries.BProfile.GetProfileFromSession.IsMaster = False Then
            context.Response.Redirect(String.Format("~/Contest/Follower.aspx?v={0}", _
                                                    BContestLibraries.BProfile.GetProfileFromSession.IdentifierFollower), _
                                                    True)

        ElseIf context.Request.QueryString("v") Is Nothing _
                AndAlso BContestLibraries.BProfile.GetProfileFromSession.IsMaster = True Then
            context.Response.Redirect("~/Contest/Status.aspx")
        Else
            context.Response.Redirect("~/")
        End If
    End If
End If
```

Figura 40 – Extrato do método *ProcessRequest* (*Http Handler Facebook.ashx*)

Caso o parâmetro *code*, proveniente do pedido do Facebook exista, pode ser processado o terceiro passo (autenticação da aplicação) de forma a obter-se o *token* de acesso que permite efectuar chamadas à API do Facebook.

É invocado desta forma o segundo método da classe Facebook, *GetAccessToken* que recebe como parâmetro o *code* retornado pelo Facebook. Através de um pedido do tipo GET [60] [61], este método permite efectuar a autenticação da aplicação, colocando no URL do pedido os seguintes parâmetros:

- *Cliente_id e Redirect_uri* – tal como na primeira chamada;
- *Cliente_secret* – código *App Secret* fornecido pelo Facebook quando a aplicação foi registada;
- *Code* – parâmetro obtido como resposta do Facebook ao primeiro método invocado.

Exemplo de pedido:

```
https://graph.facebook.com/oauth/access_token?
        client_id=184925XXXXXXX29&
        redirect_uri=http://www.voanaweb.com/Authentication/Facebook.ashx&
        client_secret=aeb615010becc38f2XXXXXXXXXXXXXX46&
        code=AQAII4AQ71CXZ57FzwYFebdnYfT_DKsJJqtnm01LZ1R6YmrbDfeomWRLuqvdNFCkf-XY-
        ZLXvQrPcoJMUBbbXXXXXXXXXX-9hpHN9yji3sbJOH5BqK0-
        GGHBTdAfFMnBlFWlCLG32e8Nhkek0vSXXXXXXXXXXXXXXpfmwCU1M5pPm-
        WxUkUJATXOPGlOTf1r_tVCrUQjj4LtZx0K4lNhCZ4l
```

Se a aplicação estiver corretamente autenticada e o código de autorização do utilizador for válido, é devolvido o *token* de acesso. Com este *token*, a aplicação consegue interagir com a *Graph API*, acedendo à informação pretendida (à qual é autorizada a aceder) [58]. É através do método *GetUser* da classe Facebook conjuntamente com este *token*, que se obtém a informação necessária para o registo do utilizador no concurso (identificador do utilizador, nome e *e-mail*), conforme o ilustrado na Figura 41.

```
Class FacebookUser
    Property id As String
    Property name As String
    Property email As String
End Class


Public Function GetUser() As FacebookUser
    Dim userInformationUrl As String = String.Format("https://graph.facebook.com/me?fields=id,name,email&access_token={0}", Token)

    Dim wc As New WebClient()
    Dim ms As New System.IO.MemoryStream(wc.DownloadData(userInformationUrl))
    Dim ser As New System.Runtime.Serialization.Json.DataContractJsonSerializer(GetType(FacebookUser))
    ms.Position = 0
    Dim p2 As FacebookUser = DirectCast(ser.ReadObject(ms), FacebookUser)
    Return p2
End Function
```

Figura 41 - Classe *Facebook* (parcial) – método para obter dados do utilizador

Exemplo de pedido:

```
https://graph.facebook.com/me?
        fields=id,name,email&
        access_token=AAACoMF3B0KkBALkTjeXXXXXXXXXXXXXXXXflGZAUOasUb6KDXZBZCedA0YNqdeCHMY
        MnYoprm1xhaVoBZBn5oVnLWXXXXXXXXXXXXXXXXXXXXXXXXKjdniR
```

A Graph API responde com um objeto do tipo JSON [62] que é desserializado para um objeto próprio (*FacebookUser* - Figura 41).

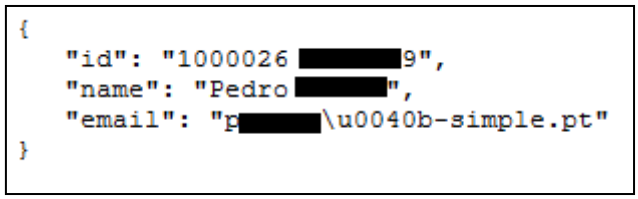

Figura 42 – Objeto obtido através da *Graph API*

Com estes dados é invocado o método *login* do *BProfile* através da seguinte instrução do *Facebook.ashx*:

```
BContestLibraries.BProfile.GetProfileFromSession.Login(fb.id, fb.name,
fb.email)
```

Este método é o responsável por preencher a informação de perfil do utilizador, registando-o, ou não, na aplicação. Para se obter os dados relativos ao utilizador armazenados na BD, recorre-se ao *script* ilustrado na Figura 43.

```
select
    *
    ,isnull((select top 1 convert(bit,1)
             from   Register
             where  UserID=[User].ID
                    and Type=@TypeMaster),convert(bit,0))    as IsMaster

    ,isnull((select top 1 convert(bit,1)
             from   Register
             where  UserID=[User].ID
                    and Type=@TypeFollower),convert(bit,0)) as IsFollower

    ,(select top 1 Identifier
      from   Register
      where  UserID=[User].ID
             and Type=@TypeMaster)                          as MasterIdentifier

    ,(select top 1 Identifier
      from   Register
      where  UserID=[User].ID
             and Type=@TypeFollower)                        as FollowerIdentifier

    ,(select (select Name from [User] where ID=UserID)
      from   Register
      where  Identifier=(select top 1 Identifier
                         from   Register
                         where  UserID=[User].ID
                                and Type=@TypeFollower)
             and Type=@TypeMaster)                          as MasterName
from
    [User]
where
    Token=@Token
    and Token<>''
```

Figura 43 – *Script* para obter perfil de utilizador

Para além das informações básicas do utilizador (*Token* ou identificador do utilizador, *Name* e *Email*), com recurso à tabela *Register* são obtidos todos os campos necessários para preencher o perfil do concorrente, nomeadamente:

- *IsMaster* – se o utilizador é do tipo *Master*, ou seja se existe algum registo na tabela *Register* em que o utilizador surja como tipo 1;
- *IsFollower* – se o utilizador é do tipo *Follower*, ou seja se existe algum registo na tabela *Register* em que o utilizador surja como tipo 2;
- *MasterIdentifier* – identificador externo que utiliza para divulgar o seu *link* aos *Followers* (preenchido caso seja *Master*);
- *FollowerIdentifier* – identificador externo que utilizou para seguir um *Master* (preenchido caso seja *Follower*);
- *MasterName* – nome do *Master* que seguiu.

Exemplo de *output* do perfil do utilizador na Figura 44.

ID	Token	Name	Email	IsMaster	IsFollower	MasterIdentifier	FollowerIdentifier	MasterName
136	1000026[redacted]9	Pedro[redacted]	p[redacted]@b-simple.pt	1	0	7S58[redacted]a[redacted]	NULL	NULL

Figura 44 – Perfil de utilizador

A classe que representa o perfil do utilizador (*BProfile*) é então preenchida caso o utilizador já esteja registado no concurso. Caso não se obtenha qualquer registo nesta consulta, o utilizador ainda não se encontra registado, pelo que se procede ao seu registo (tabela *User*), antes de serem preenchidos os seus dados de perfil.

Concluído este passo, é registado o tipo de utilizador na aplicação, através da seguinte instrução do *Facebook.ashx*:

```
bcontest.Register(context.Request.QueryString("v"))
```

Este método é o responsável pelo preenchimento da tabela *Register*, que regista as relações entre os utilizadores, definindo as relações entre *Masters* e *Followers*. Antes de algum registo ser efetuado, é verificado se nenhuma das regras do concurso já referidas é violada, conforme ilustra a Figura 45:

```
Public Function Register(ByVal identifier As String) As Boolean
    If identifier = Nothing AndAlso BProfile.GetProfileFromSession.IsMaster Then
        'Se já é master não se volta a inscrever
    ElseIf identifier = Nothing = False AndAlso identifier = BProfile.GetProfileFromSession.IdentifierMaster _
                                        AndAlso BProfile.GetProfileFromSession.IsMaster Then
        'Não pode ser seguidor de si próprio
    ElseIf identifier = Nothing = False AndAlso BProfile.GetProfileFromSession.IsFollower Then
        'Não pode ser seguidor duas vezes
    Else
```

Figura 45 – Validação das regras do concurso

Se a participação for válida, analisa-se o modo de entrada do utilizador na aplicação. Se entrou através de um *link* fornecido por um *Master* (contendo **v** na *query string*) e passou a validação das regras, está a ser seguidor pela primeira vez e assim sendo é registado como seguidor na tabela *Register*, de um *Master* representado pelo identificador externo *identifier*.

Caso não exista *identifier* o utilizador é *Master*, sendo registado na tabela como tal, conjuntamente com um código externo único, gerado para o utilizador divulgar o seu *link*.

Mediante o perfil do utilizador e a forma como entrou na aplicação, o utilizador é por fim reencaminhado pelo *Facebook.ashx* para uma página pessoal específica para utilizadores do tipo *Master* (*Status.aspx* - Figura 46) ou do tipo *Follower* (*Follower.aspx* - Figura 47).

4.6 Página Pessoal

Terminado o processo de autenticação/registo, o utilizador é reencaminhado para a sua página pessoal, mediante o seu perfil (*Master* ou *Follower*).

Figura 46 – Página de um *Master* - *Status.aspx*

Figura 47 - Página de um *Follower* - *Follower*.aspx

As páginas dos dois tipos de utilizadores são semelhantes. A página de um *Follower* contém um agradecimento em nome do *Master*, contendo o seu nome (recorrendo à propriedade *NameMaster* do *BProfile*), *link* e incentivo à divulgação da sua página.

Já a página de um *Master*, contém o número total de seguidores que possui, bem como o seu *link* pessoal para divulgação.

Para obterem o *link*, ambas as páginas acedem à função *GetSharedURL* que mediante o tipo de utilizador, *Master* ou *Follower*, retorna o *link* para divulgação recorrendo às propriedades *IdentifierMaster* ou *IdentifierFollower* do *BProfile*.

A página pessoal do *Master* recorre também a uma outra função que lhe permite calcular o total de seguidores que possui, contando (via *SQL COUNT*) o número de concorrentes *Follower* da tabela *Register* que possuem o identificador do *Master* em questão.

4.7 Partilha nas redes sociais

Um dos principais requisitos da aplicação era a divulgação do concurso via redes sociais, ficando estabelecido que teria que permitir a divulgação dos *links* (presentes nas páginas pessoais dos utilizadores) através do Facebook e do Twitter.

4.7.1 Facebook

Através da página do Facebook relativa a este assunto [63] foi possível criar um botão de partilha conforme o pretendido, adaptando-o à página do concurso.

Para a colocação do botão de partilha na página foi adicionado um extrato de código HTML, adaptado à página em questão. Para a página do *Master* recorreu-se novamente à função *GetSharedURL* para se obter o *link* a partilhar (Figura 48).

```
<a name="fb_share" type="button" share_url='<%=GetShareURL(BContestLibraries.BContest.enumType.Master)%>'
    href="http://www.facebook.com/sharer.php">Facebook</a>
<script src="http://static.ak.fbcdn.net/connect.php/js/FB.Share" type="text/javascript"> </script>
```

Figura 48 – Botão *Share* do Facebook

Quando o utilizador clica neste botão, o Facebook analisa o código HTML gerado pela página partilhada e procura certas *tags* de forma a fornecer uma pré-visualização da página,

nomeadamente o seu título, descrição e imagem. Para o correto funcionamento desta funcionalidade a página terá que ser obrigatoriamente pública, de forma a permitir esta análise.

O Facebook guarda a informação relativa a cada *site* em cache, o que em termos de desenvolvimento/atualização é de certa forma problemático uma vez que, alterando-se qualquer um destes três elementos, as alterações não têm efeito imediato.

O melhor método para limpar a cache do Facebook relativamente à página, resume-se à utilização de uma ferramenta fornecida por este, o *URL Linter* [64] que permite, para uma determinada página, saber qual a informação (atualizada) que o Facebook possui acerca desta e possíveis problemas na sua implementação.

Neste caso, o título da página manteve o slogan do concurso *VoaNaWeb*, foi colocada uma *meta description* com o *link* devolvido pelo método *GetSharedURL* e uma imagem na página, sendo o resultado da partilha o ilustrado na Figura 49.

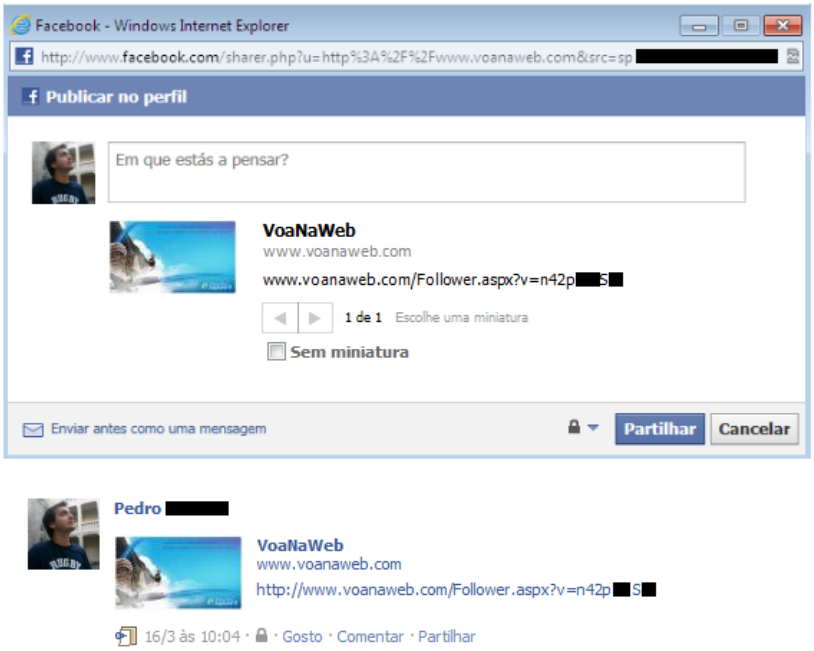

Figura 49 – Partilha de *link* de *Master* no Facebook (*preview* e *post*)

4.7.2 Twitter

Tal como o Facebook, o Twitter também possui um botão para partilha de conteúdos, embora bastante mais simples de implementar.

Na página do Twitter relativa ao botão de partilha [65] este pode ser personalizado através da indicação do *link* que se pretende partilhar, do texto que se pretende colocar antes do *link* de partilha, do estilo de botão e do idioma. Terminado o processo de personalização do botão é gerado o código HTML, para incluir na página pretendida (Figura 50).

```
<a href="http://twitter.com/share" class="twitter-share-button"
    data-url='<%=GetShareURL(BContestLibraries.BContest.enumType.Master)%>'
    data-text="Voa Na Web" data-count="none">Twitter</a>
<script type="text/javascript" src="http://platform.twitter.com/widgets.js"></script>
```

Figura 50 - Botão *Tweet* do Twitter

Ao ser pressionado o botão do Twitter, o texto presente na propriedade *data-text* é partilhado bem como o URL presente no *data-URL*. Uma vez que existe a limitação de 140 caracteres por *post* no Twitter, este utiliza uma funcionalidade que automaticamente encurta os URL para 19 caracteres, tornando a partilha mais compacta, conforme ilustrado na Figura 51.

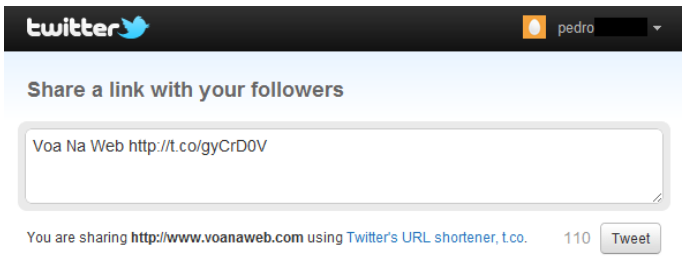

Figura 51 - Partilha de *link* de *Master* no Twitter

4.8 Partilha via e-mail

O envio de *e-mails*, foi planeado de forma a ser um processo assíncrono, envolvendo uma troca de mensagens entre os *Web Roles* e os *Worker Roles* via *Queues*.

O desenvolvimento começou pela definição de uma classe que representasse genericamente um *e-mail*, *BContestNotifMail* (Figura 52). Esta estrutura continha os destinatários da

mensagem (*To*,*CC* e *BCC*) o assunto (*Subject*), a mensagem em si (*Message*) e possíveis anexos (*Attachments*). O *BContestNotifMail* possuia também uma função *GetMessageBody* que recebendo o *link* a partilhar, construia o HTML necessário para criar o conteúdo do *e-mail*.

```
<Serializable()> _
Public Class BContestNotifMail

    Public Sub New()
    End Sub

    Public Property Message As String
    Public Property Subject As String
    Public Property [TO] As List(Of String)
    Public Property CC As List(Of String)
    Public Property BCC As List(Of String)
    Public Property Attachments As Dictionary(Of String, Byte())
```

Figura 52 – *BContestNotifEmail* – classe que representa mensagem de *e-mail*

Foi também utilizada uma classe genérica, a nível da biblioteca da aplicação, contendo funcionalidades para o processamento de notificações (*BNotif*). Nesta classe existe um método *Send* que recebe todos os elementos que compõem um objeto do tipo *BContestNotifMail* (*Message*,*Subject*,*TO*,*CC*,..) e toda a informação relativa à conta SMTP. Com esta informação é criado um objeto do tipo mensagem de *e-mail* (*MailMessage* [67] – sendo este preenchido com os elementos que constituem o *BContestNotifMail*) bem como um cliente SMTP (*SmtpClient* [68]) que, inicializado com as informações relativas à conta, envia efetivamente a mensagem.

A funcionalidade base seria então a geração e envio de *e-mails* personalizados com o *link* a divulgar. Para isso bastaria que o utilizador (na sua página pessoal) clicasse no botão de envio de *e-mail* e digitasse os endereços de *e-mail* dos amigos para os quais o pretendia divulgar.

Para a introdução dos *e-mails* por parte do utilizador recorreu-se a uma *dialog JQueryUI* [69][70][71], que era fechada caso os endereços de *e-mail* introduzidos fossem válidos, iniciando-se o seu processamento. Caso os endereços de *e-mail* fossem inválidos, a janela mantinha-se aberta sendo o utilizador notificado para o problema (Figura 53).

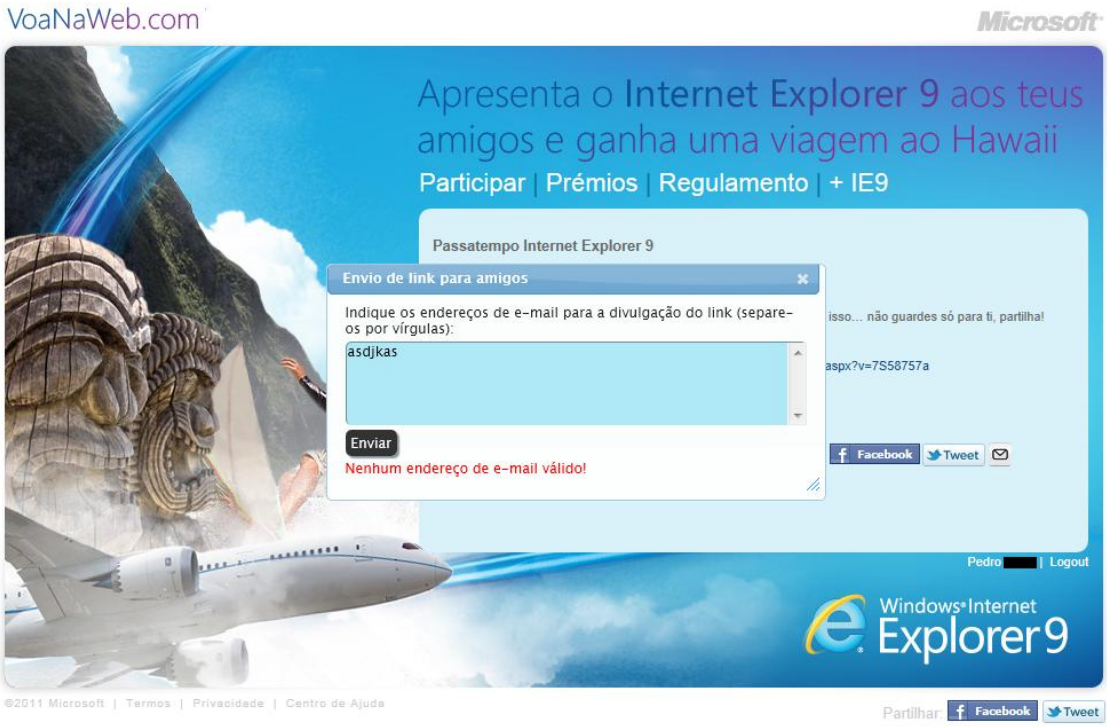

Figura 53 – *Popup* para introdução de endereços de *e-mail*

4.8.1 Colocar objetos na Queue (através de Web Role)

O processo de envio de notificação iniciava quando o utilizador carregava no botão "Enviar", presente na janela de introdução de endereços. Os *e-mails* introduzidos eram submetidos a uma expressão regular para validação [72], sendo posteriormente adicionados a uma lista de *e-mails* (Figura 54). O assunto do *e-mail* era personalizado com o nome do *Master* e o conteúdo, com o seu *link pessoal*:

- *BContestLibraries.BProfile.GetProfileFromSession.Name* e *GetShareURL(BContestLibraries.BContest.enumType.Master)* no caso da página *Status.aspx*;
- *BContestLibraries.BProfile.GetProfileFromSession.NameMaster* e *GetShareURL(BContestLibraries.BContest.enumType.Follower)* no caso da página *Follower.aspx*.

Com estes dados era construído um objeto do tipo *BContestNotifMail* que iria ser colocado numa *Queue* através do método *sendToQueue* (Figura 54).

```
Private Sub btnSend_Click(sender As Object, e As System.EventArgs) Handles btnSend.Click
    Dim EmailTOList As New List(Of String)
    '...
    Dim reg As New System.Text.RegularExpressions.Regex("\b[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\b")

    For Each it As System.Text.RegularExpressions.Match In reg.Matches(txtEmails.Text)
        EmailTOList.Add(it.Value.Trim())
    Next
    '...
    Dim Subject As New StringBuilder()
    Subject.AppendFormat("Voa Na Web: Participa e ajuda o teu amigo {0} a ir ao Hawaii!", _
                                                        BContestLibraries.BProfile.GetProfileFromSession.Name)

    Dim Message As String = BContestNotif.BContestNotifMail.getMessageBody(GetShareURL(BContestLibraries.BContest.enumType.Master))

    Dim emailNotif As New BContestNotif.BContestNotifMail With {.BCC = EmailBCCList, _
                                                                .CC = EmailCCList, _
                                                                .TO = EmailTOList, _
                                                                .Subject = Subject.ToString, _
                                                                .Message = Message, _
                                                                .Attachments = Attachs}

    sendToQueue(emailNotif)
End Sub

Private Sub sendToQueue(ByVal emailNotif As BContestNotif.BContestNotifMail)
    Dim account = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")

    Dim queueClient = account.CreateCloudQueueClient()
    Dim queue As CloudQueue = queueClient.GetQueueReference("emailqueue")
    queue.CreateIfNotExist()

    Dim objStream As New MemoryStream
    Dim objFormatter As New Formatters.Binary.BinaryFormatter
    objFormatter.Serialize(objStream, emailNotif)

    Dim msg As New CloudQueueMessage(objStream.ToArray)
    queue.AddMessage(msg)
End Sub
```

Figura 54 – Envio de mensagens para uma *Queue* (*Status.aspx*)

Após se adicionarem os *namespaces* necessários para a interação com a *Storage*, o método *sendToQueue* (através dos dados da conta definidos a nível do *Web Role*) obtém uma referência para a *Queue*, criando uma nova caso a *emailqueue* não exista [40]. O objecto do tipo *BContestNotifMail (emailNotif)* é então serializado e adicionado à dita *Queue*.

4.8.2 Obter objetos da Queue (através de Worker Role)

A nível do *Worker Role*, após a configuração da conta *storage* tal como no *Web Role*, adicionaram-se os *namespaces* necessários para a interação com a *Storage* e colocou-se na função *OnStart* a mesma instrução introduzida na *aplication_start* do *Global.asax,* conforme o detalhado no tópico 4.3.1 (Desenvolvimento Azure – particularidades):

```
CloudStorageAccount.SetConfigurationSettingPublisher(Function(configName,
configSetter)
configSetter(RoleEnvironment.GetConfigurationSettingValue(configName)))
```

No método *Run,* método que o *Worker Role* irá processar após o arranque, obtém-se a referência para a *Queue emailqueue* e verifica-se a existência de novas mensagens. Caso exista alguma mensagem para processar, esta é desserealizada e com o objecto obtido invocado o método *sendEmail*. Se o envio do *e-mail* for processado correctamente a

mensagem é eliminada da *Queue*, não ficando disponivel para novo processamento (Figura 55).

```
Public Overrides Sub Run()
    '..
    initializeNotif(NotifMaxBackoff, NotifBackoffStep, CurrentBackoff)
    Dim storageAccount As CloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString")
    Dim queueClient As CloudQueueClient = storageAccount.CreateCloudQueueClient()
    Dim queue As CloudQueue = queueClient.GetQueueReference("emailqueue")
    While (True)
        If queue.Exists() Then
            Dim msg As CloudQueueMessage = queue.GetMessage()
            If (msg IsNot Nothing) Then
                initializeNotif(NotifMaxBackoff, NotifBackoffStep, CurrentBackoff)
                Try
                    Dim objStream As New MemoryStream(msg.AsBytes)
                    Dim objFormatter As New Formatters.Binary.BinaryFormatter
                    Dim emailNotif As New BContestNotif.BContestNotifMail
                    emailNotif = CType(objFormatter.Deserialize(objStream), BContestNotif.BContestNotifMail)

                    For Each Email As String In emailNotif.TO
                        Trace.TraceInformation(String.Format("Mensagem recebida com o titulo '{0}', para '{1}'.", emailNotif.Subject, Email))
                    Next

                    Dim sendError As String = Nothing
                    sendError = sendEmail(emailNotif)

                    If sendError Is Nothing Then
                        queue.DeleteMessage(msg)
                        Trace.TraceInformation(String.Format("Mensagem enviada com sucesso"))
                    Else
                        Trace.TraceInformation(String.Format("Problema ao enviar mensagem. '{0}'.", sendError))
                    End If
                Catch ex As Exception
                    Trace.TraceInformation(String.Format("Problema ao enviar mensagem: '{0}' '{1}'.", ex.Message, ex.InnerException))
                End Try
            Else
                If CurrentBackoff < NotifMaxBackoff Then
                    CurrentBackoff += NotifBackoffStep
                End If
                Thread.Sleep(CurrentBackoff)
            End If
        End If
    End While
End Sub
```

Figura 55 – Leitura de mensagem de uma *Queue* (rotina *Run* do *Worker Role*)

Conforme já referido, existe uma tabela de configuração a nível da BD, com os dados de acesso à conta SMTP para o envio de notificações via *e-mail*. O método *sendEmail* do *Worker Role* é o responsável pela leitura desses dados e pela invocação do método *send* da classe *BNotif* com toda a informação necessária para o envio da mensagem de *e-mail*.

Implementou-se também um algoritmo de *backoff* de forma a adaptar a frequência de consulta de novas mensagens na *queue*. Com este mecanismo, sempre que o *Worker Role* consulta a *queue* e não encontra mensagens, é incrementada uma unidade (*NotifBackoffStep*) ao *CurrentBackoff*, diminuindo a frequência de verificação da *queue* até um máximo predefinido (*NotifMaxBackoff*) (Figura 55). Por outro lado, quando uma mensagem é processada, o valor do *CurrentBackoff* é colocado a zero e são repostos os restantes valores definidos na tabela *WNotif*, através da função *initializeNotif*.

Ao serem adicionados os comandos *Trace.TraceInformation* ao código, consegue-se adicionalmente gerar *outputs* para o *log* de diagnóstico do *Worker Role*. Localmente, no

emulador do Windows Azure, o sucesso ou insucesso no envio dos *e-mails* pode ser observado conforme ilustrado na Figura 56. No primeiro caso deste cenário os dados de acesso à conta SMTP encontravam-se incorretos, tendo sido retificados no segundo.

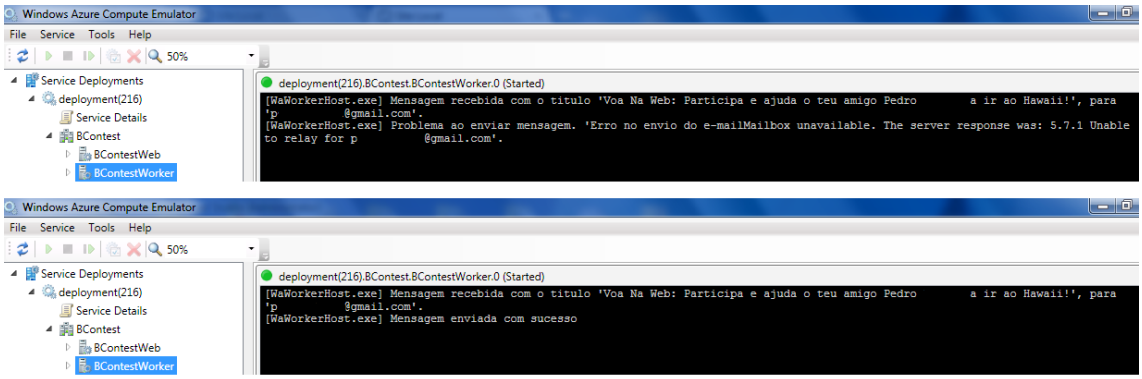

Figura 56 – Diagnóstico - envio de *e-mails*

Sendo corretamente processado, o(s) destinatário(s) recebia(m) assim um *e-mail* com o assunto indicando o remetente e com o corpo do *e-mail* constituído por uma imagem do concurso com a sua explicação geral (Figura 57). Ao clicar na imagem o utilizador era reencaminhado para o *site* do concurso através do *link* do *Master* e caso participasse desta forma, ficaria registado como *Follower* deste.

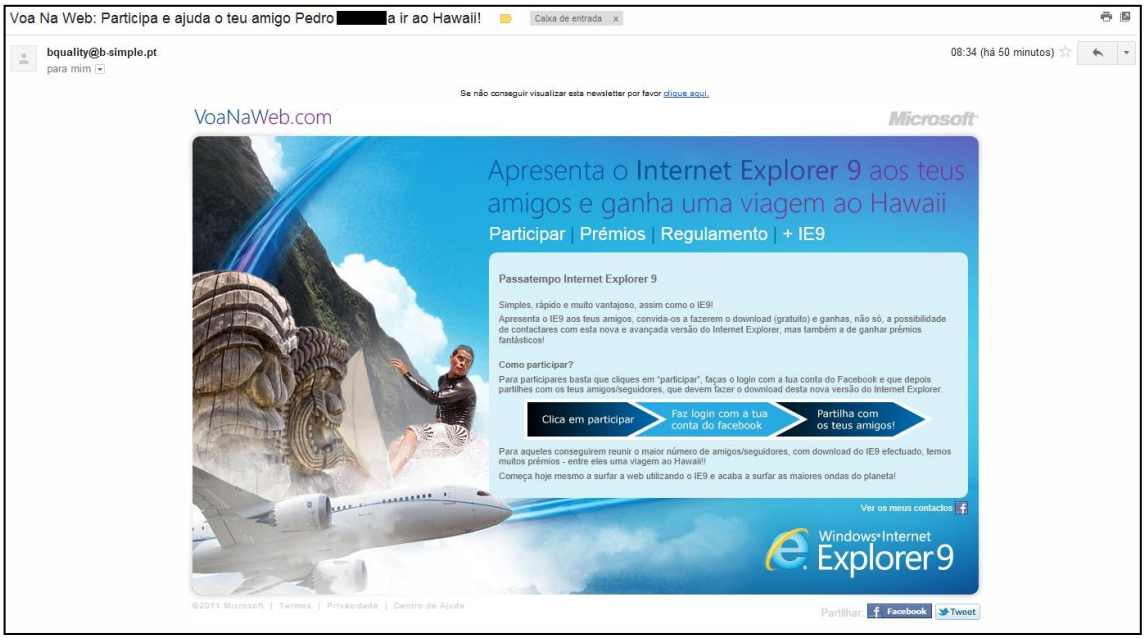

Figura 57 – *E-mail* recebido por destinatário

4.9 Deploy para a Cloud

Com o desenvolvimento terminado teria que se colocar a solução em produção na *Cloud*.

Na fase final do desenvolvimento, devido a políticas de divulgação do produto, o responsável pelo projeto estabeleceu que preferia que a divulgação do *site* fosse efetuada exclusivamente via redes sociais, abandonando-se definitivamente o conceito de divulgação via *e-mail*, considerado mais pessoal. Desta forma, o *Worker Role* não passaria para produção, nem as tabelas *WNotif* e *WNotifAccount*, tendo sido removidos do projeto todos os elementos que com estes interagiam**.**

A Microsoft forneceu uma conta no portal do Windows Azure, permitindo a utilização de forma livre uma máquina virtual de tamanho pequeno (Tabela 5), bem como os dados de acesso a uma conta SQL Azure do tipo *Web Edition* (3.4), de forma a colocar-se aplicação e BD a correr na *Cloud*.

4.9.1 Migração da Base de Dados para SQL Azure

A nível de base de dados, durante o desenvolvimento da aplicação, utilizou-se uma BD local a correr sobre uma instância SQL Server 2008 R2.

A migração da BD para SQL Azure poderia ser efetuada de diversas formas [73], tendo-se optado pela utilização da ferramenta *SQL Azure Migration Wizard*.

Esta ferramenta permite a migração de bases de dados provenientes de SQL Server 2005 ou 2008 para SQL Azure de uma forma integrada. Permite analisar a base de dados de origem, detetando potenciais incompatibilidades com o SQL Azure, permitindo efectuar parcial ou integralmente, a migração do esquema da base de dados e/ou dos respetivos dados.

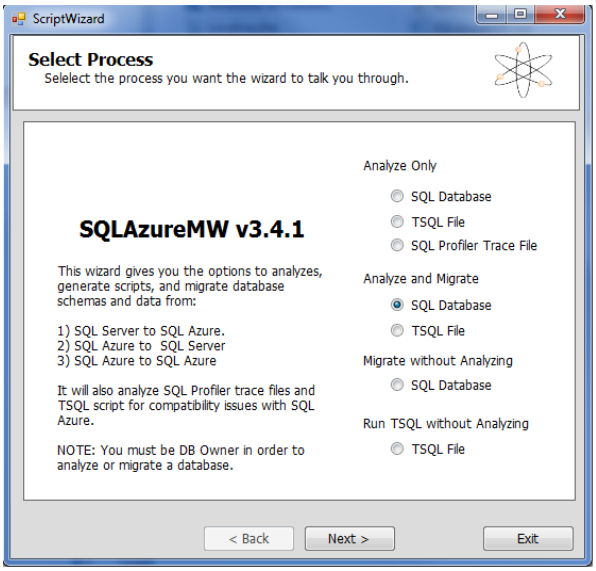

Figura 58 - SQL Azure Migration Wizard v3.4.1

Selecionando o processo de análise e migração da base de dados local (Figura 58), indica-se qual a base de dados de origem e definem-se as opções de migração. É gerado automaticamente um *script*, contendo as instruções necessárias para a construção de todos os objetos selecionados da BD de origem e a respetiva análise de compatibilidade. No passo seguinte, o *script* é executado pela aplicação sobre a base de dados SQL Azure, mediante a introdução dos dados de acesso à conta. O *SQL Azure Migration Wizard* permite também a transferência de dados de uma forma eficiente, uma vez que utiliza o mecanismo de BCP [74], que efetua a cópia da informação em massa entre a instância de SQL e ficheiros de dados (*.dat*), criados localmente no momento da construção do *script*. Esta funcionalidade não foi utilizada, uma vez que apenas se pretendia migrar o esquema da base de dados, tendo sido retirada essa opção nas opções de migração.

Uma vez na *Cloud*, através de SQL Server Management Studio 2008 R2, o acesso é similar a uma qualquer BD, sendo possível uma navegação e manipulação básica dos seus objetos.

Com a BD em produção alterou-se a *connection string* presente no *webconfig* da aplicação para a seguinte:

```
connectionString="Server=tcp:XXXXXXXXXX.database.windows.net;Database=bcontest;UserID=XXXXXXXX;Password=XXXXXXXX;Encrypt=True;TrustServerCertificate=False;"
```

Desta forma, explicita-se a obrigatoriedade de cifragem da conexão (*Encrypt=True*) e a não confiança de certificados devolvidos pelo servidor (*TrustServerCertificate=False*). Através

deste mecanismo, quando a conexão à BD SQL Azure ocorre, estes certificados são validados de forma a prevenir ataques do tipo *man in the middle* [75].

4.9.2 Publicação da aplicação

Para uma aplicação ser efetivamente enviada para produção em Azure, existem alguns passos que terão que ser seguidos após o desenvolvimento da mesma.

Inicialmente terá que ser criado no portal de gestão do Windows Azure um *hosted service* e uma *storage account* [76]. O *hosted service* constitui um *container* para a aplicação, sobre o qual vão correr os *roles* em máquinas virtuais. Quando a aplicação é publicada via Visual Studio, é gerado um *service package* que vai ser enviado para aquele *hosted service*. A *storage account*, conforme já referido, é a conta de armazenamento na *Cloud*, permitindo o armazenamento da informação em *Blobs*, *Tables* e *Queues*.

A criação destes dois elementos no portal Windows Azure pressupõe a escolha da conta sobre a qual vão ser criados, de um nome/url para acesso externo e da escolha da região dos *datacenters* onde os recursos irão ser alocados (Figura 59). A escolha do *datacenter* deverá recair sobre aquele que se encontra mais próximo dos utilizadores do serviço (*North* ou *West Europe*, Dublin ou Amsterdão neste caso), de forma a assegurar a maior largura de banda possível, reduzindo a latência no acesso aos dados e melhorando a performance geral da aplicação.

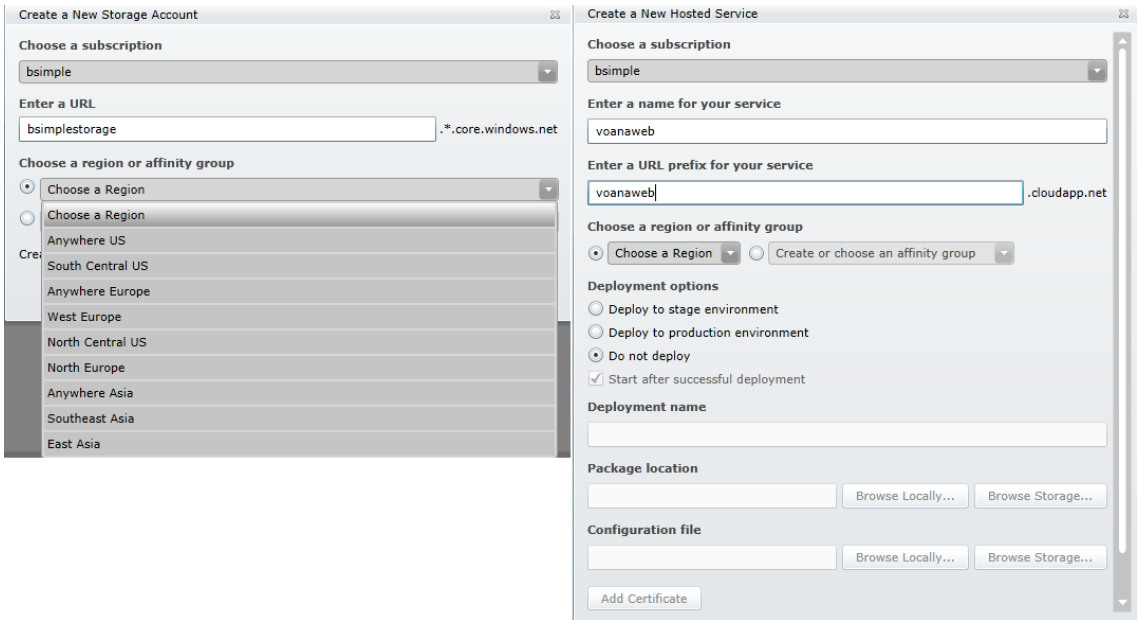

Figura 59 – Configuração da *Storage Account* e do *Hosted Service* - portal Windows Azure

Após a criação da *storage account* são geradas duas chaves para acesso da aplicação à conta de armazenamento.

Nas configurações dos *roles* na aplicação (Figura 28 e Figura 29) é então revisto o número e tamanho de máquinas necessárias e nos detalhes da conta de armazenamento, colocada a nova conta criada, indicando o seu nome (*bsimplestorage* - Figura 59) e uma das chaves geradas. A aplicação encontra-se assim pronta para a publicação na *Cloud*.

Utilizando-se o Visual Studio para publicar a aplicação, seleciona-se o projeto e clicando em *publish*, a opção de *deploy* para o Windows Azure. De forma a garantir que a máquina tem permissões para efectuar o *deploy*, existe adicionalmente um mecanismo de segurança que envolve a criação de um certificado, que terá que ser previamente definido no portal do Windows Azure [76].

Para realizar este processo, caso ainda não se possua um certificado válido, escolhe-se a opção de criação de uma nova credencial no Visual Studio (Figura 60), sendo criado um certificado na pasta do utilizador atual (passo 1).

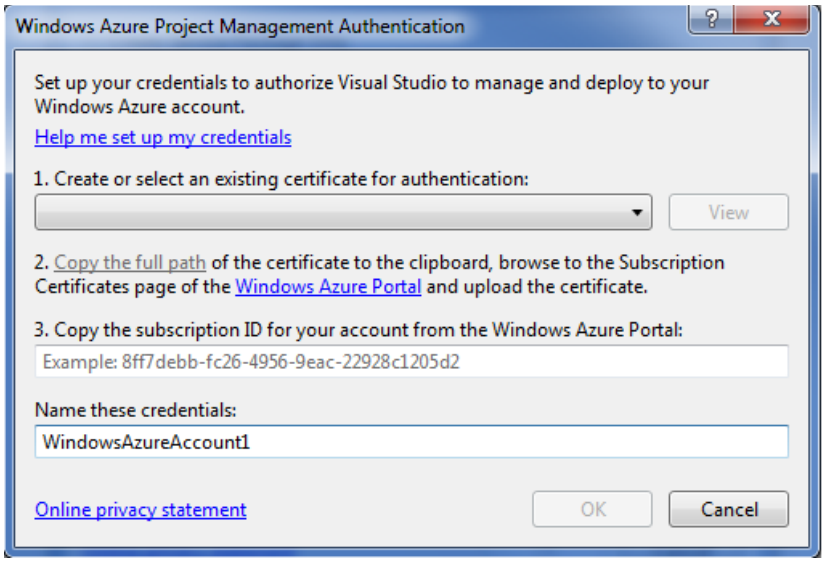

Figura 60 – Criação de credencial (Visual Studio)

Copia-se o *link* do certificado gerado (passo 2) e no portal do Windows Azure adiciona-se o novo certificado à conta. Obtém-se um código (*Subscription ID*) no portal e coloca-se na caixa de texto na janela da aplicação (passo 3).

Com as credenciais introduzidas, são carregados automaticamente os dados associados à conta Azure. Terá então que se selecionar o *hosted service* ao qual se pretende associar o projeto, a *storage account* na qual vai ser armazenado o *package* durante o *deploy* e a máquina para a qual se pretende efectuar o *deploy*: *staging* ou produção (Figura 61).

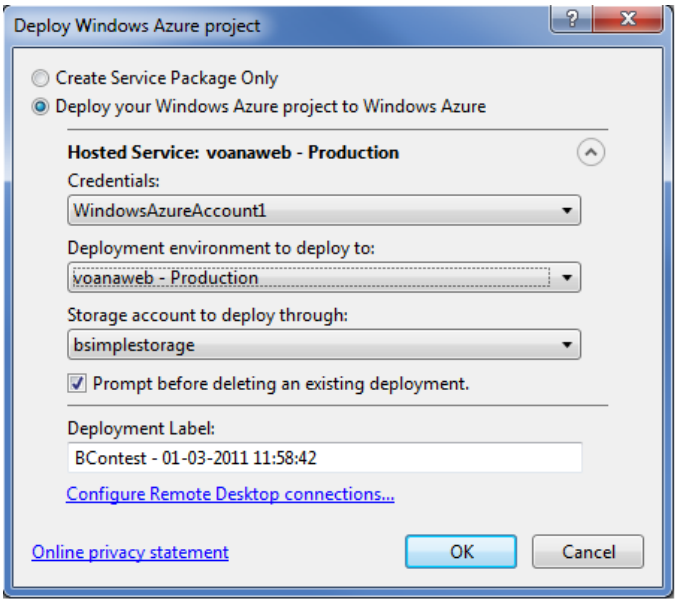

Figura 61 – *Deploy* da aplicação para Windows Azure (Visual Studio)

O Visual Studio começa o processo de *deploy* da aplicação, sendo mostrada uma janela de progresso, que possui um *log* detalhado das operações que decorrem. Neste processo que pode demorar entre 15 minutos a uma hora, é feito o *upload* do *package* com a aplicação para o Windows Azure e as instâncias configuradas na aplicação são alocadas na região selecionada [76]. No final do processo o estado do serviço pode ser verificado no portal do Windows Azure (Figura 62).

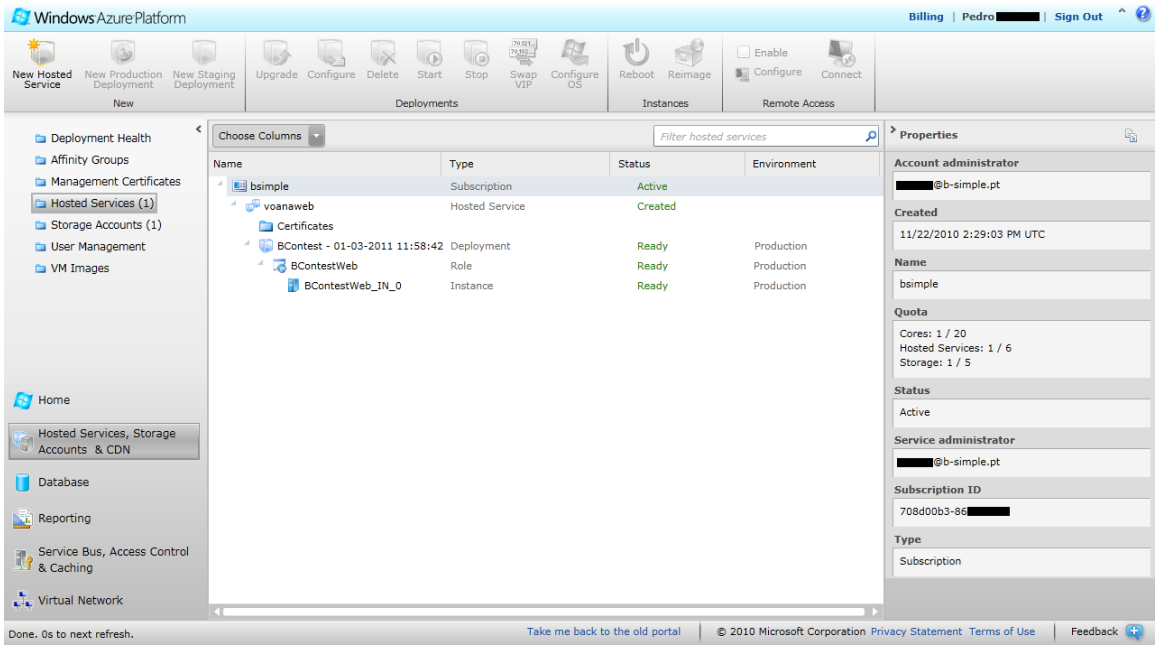

Figura 62 – *Hosted Service* - portal do Windows Azure

5 Análise no contexto da empresa

5.1 Introdução

Após o estudo sobre o paradigma de *Cloud Computing*, sobre a plataforma Azure e terminado o desenvolvimento do protótipo prático sobre esta plataforma, pretende-se perceber, face à situação atual da empresa, de que forma é que a utilização do Azure para o desenvolvimento aplicacional a afetaria.

Numa fase inicial é desenvolvido um estudo sobre o portefólio aplicacional da organização, sendo posteriormente analisadas as alterações que a migração dessas aplicações para Azure trariam, nomeadamente a nível de soluções, recursos humanos e clientes.

5.2 Portefólio aplicacional da empresa

A empresa possui atualmente diversas soluções de *software*, para diferentes áreas de negócio.

Embora variados, a maior parte dos seus produtos baseiam-se numa arquitetura cliente-servidor, em que tipicamente a aplicação se encontra instalada localmente e a BD / possíveis serviços, centralizados no servidor. São de seguida analisados os principais produtos que a empresa atualmente comercializa.

5.2.1 Labway LIMS[6]

Constitui uma solução integrada de gestão para laboratórios de ensaios, que permite a automatização de todos os processos laboratoriais neste âmbito. Algumas das características do *software*:

- Totalmente configurável pelo cliente;
- Rastreabilidade das operações realizadas;
- Solução adaptada para apoiar os laboratórios no seu processo de acreditação;
- Automatização dos processos laboratoriais de análise – Calendarização de colheitas; gestão de amostras; distribuição de trabalho; introdução de resultados; emissão de boletins e publicação de resultados;
- Envio de resultados por SMS, *e-mail* e publicação na *web*;
- Integração (uni/bi) direcional com equipamentos e aplicações externas;
- Certificação digital de boletins;
- Gestão de *stocks*;
- Controlo de qualidade;
- Consulta de mapas e estatísticas.

Esta solução é destinada a um nicho do mercado de análises laboratoriais, podendo atingir três áreas distintas: Indústria (de combustíveis, farmacêutica, carvão, componentes elétricos); Alimentícia (alimentação humana e ração animal) e Águas (entidades gestoras).

5.2.2 ERP

Trata-se de um *software* de gestão comercial que permite gerir empresas de forma integrada. Permite a automatização de processos e a organização e disponibilização de toda a

[6] LIMS - *Laboratory Information Management System*

informação necessária à gestão de empresas. Encontra-se dividida em duas grandes áreas *Business* e *Accounting*, possuindo um conjunto de funcionalidades descritas de seguida.

Para a área *Business*:

- *Gestão de documentos de clientes e fornecedores* – Encomendas e orçamentos; faturas, notas de débito ou crédito; integração com contas correntes e contabilidade;
- *Gestão de stocks* – Gestão de armazéns; controlo de existências;
- *Gestão de contas correntes* – Documentos diretos de conta corrente; gestão de pagamentos e recebimentos; integração com tesouraria e contabilidade;
- *Bancos e tesouraria* – Gestão de caixa e contas bancárias; movimentos bancários; planeamento de tesouraria; integração com contabilidade.

Para a área *Accounting*:

- *Contabilidade* – Geral; centros de custo; contabilidade analítica; apuramentos automáticos; declarações fiscais; extratos, balanço e balancetes;
- *Recursos Humanos* – Ficha de colaboradores; processamento de salários; controlo de pagamentos; declarações obrigatórias;
- *Imobilizado* – Ficha de bens, cálculo de amortizações, declarações obrigatórias.

Sendo composto por duas áreas que são básicas na maioria das empresas e dada a possibilidade de encaixar em soluções específicas com o desenvolvimento de módulos que complementam a aplicação, o ERP pode ser adaptado a qualquer tipo de organização.

5.2.3 Quality

Constitui um *software* de gestão orientado para Sistemas de Gestão de Qualidade (SGQ) que permite a implementação e/ou digitalização de sistemas de gestão segundo diferentes referenciais (normativos, legislativos ou guias de boas práticas) com vista (ou não) à certificação de processos. As funcionalidades macro do *Quality* contam com:

- *Documentos Inteligentes* – "Documentos digitais vivos" pesquisáveis, rastreáveis, configuráveis e integrados num sistema de notificações. Permitem desenhar e implementar processos da organização, tornando os documentos como a principal fonte de comunicação interna da organização;
- Envio de notificações e gestão de processos pendentes;
- Gestão de colaboradores e responsabilidades;
- Gestão documental;
- Configuração de *workflows*;
- Gestão de equipamentos;
- Planeamento, monitorização e indicadores de performance.

Com a aplicação *Quality* já se conseguiu dar resposta às especificações de vários referenciais, nomeadamente: ISO 9001; NP EN ISO/IEC 17025; NP 4457; ISO 18001, ISO 22000. Pode ser desta forma implementado em qualquer organização que pretenda implementar um sistema de gestão de qualidade, cujos processos possam vir a ser certificados.

5.2.4 Portals

Constitui um conjunto de soluções *web* dividido em duas vertentes de desenvolvimento: a vertente de *websites* (páginas *web*, portais,..) e a aplicacional, que no fundo se interliga com o ERP e com o *Quality*. As soluções *Portals* possuem as seguintes funcionalidades:

- Possibilidade de customização do aspeto de acordo com o cliente (temas);
- Integração com a *framework* da aplicação *windows-based*;

- Gestão de documentos inteligentes;
- Gestão documental;
- Visualização de *dashboards*;
- Gestão de questionários.

Atualmente este tipo de solução é destinada a clientes que possuam um elevado número de utilizadores, clientes em que a aplicação *windows-based* se demonstre desajustada ou a clientes que necessitem de fornecer informação aos próprios clientes.

5.2.5 Art-Center e Donor

Constituem soluções de *software* para a área da Procriação Medicamente Assistida (PMA).

A solução *ART-Center* (laboratorial) é destinada a laboratórios que efetuem técnicas de PMA, enquanto que a solução *DONOR* (médica e laboratorial) é destinado a centros de doação de gâmetas masculinos e femininos. As funcionalidades macro destas soluções são descritas de seguida.

Para a solução *ART-Center*:

- Gestão de utentes e requisições;
- *Gestão de técnicas de PMA* – Inseminação artificial; fertilização *in vitro*; injeção intracitoplasmática de espermatozoides; desvitrificação e transferência embrionária;
- Gestão de pedidos de gâmetas a bancos externos;
- Gestão de pedidos e receções;
- Estatísticas de requisições/técnicas:
- Rastreabilidade do processo;
- Integração de requisições com sistemas externos.

Para a solução *DONOR*:

- Gestão de dadores e doações;
- *Gestão de pedidos de amostras* – Seleção de amostra de acordo com o cruzamento de características entre dador e o recetor;
- *Subset* de área laboratorial – Espermogramas e criopreservação de doações.

Estas soluções de *software* podem ser implementadas em laboratórios de procriação medicamente assistida e centros de doação de gâmetas masculinos e femininos.

5.2.6 Logistics

Constitui uma solução vertical que fornece suporte aos principais processos de logística e distribuição de mercadorias, nos mercados nacional e internacional. As funcionalidades gerais deste *software* resumem-se a:

- Controlo de *stocks*, encomendas, pedidos de entregas/recolhas e devoluções desde a sua origem até à sua conclusão;
- *Tracking* de pedidos;
- Registo de entregas, recolhas e reservas;
- Planeamento de viaturas;
- Emissão de documentos de transporte nacional/internacional e alfandegários;
- Registo de saída da carga/retorno;
- Anexação de documentos de transporte assinados pelos clientes finais;
- Utilização dos códigos de barras originais para identificar e movimentar carga;
- Emissão e receção de ficheiros EDI[7];

[7] EDI - *Electronic Data Interchange* – Para transmissão estruturada de dados entre empresas

- Registo de reservas e *tracking* das entregas através de portal *web*;
- Registo e etiquetagem no expedidor.

Para este sistema integrado é de extrema importância o *software* adicional presente nos *Pocket PCs* utilizados no processo. Para a logística existem os seguintes *softwares* auxiliares:

- *Pocket Logistics* – Movimentação de unidades de logística (caixas, paletes,..); criação/manipulação dessas unidades através da emissão de etiquetas;
- *In-Out* – Apoio à receção, expedição e inventariação de produtos através do recurso a códigos de barras; controlo de armazéns, mercadoria e movimentos.

Para a distribuição:

- *Deliver* – Carregamento de viaturas; modificação do local de carga e leitura de códigos de barras externos;
- *Deliver Anyware* – Leitura de documentos de entrega; registo de entrega de carga no cliente final.

Esta solução integrada pode ser implementada em qualquer organização nas áreas de logística e distribuição.

5.2.7 POS (Point Of Sale)

Com o *software* POS a empresa dá resposta à área de vendas, registando e gerindo todos os processos que decorrem em lojas. É uma solução completa contando com as seguintes funcionalidades:

- Gestão de vendas a dinheiro; devoluções e cheques-prenda;
- Gestão de *stocks*;
- Controlo de caixa e diário de vendas;
- Gestão de cartões de cliente; *mailing* e descontos específicos;
- Controlo de vendedores e clientes;

- *Inventariação global ou parcial* – Inventários tipicamente realizados com recurso a *Pocket PC's* com o auxílio do *software In-Out*, referido no processo da logística (5.2.6);

- Transferência de mercadoria entre lojas;

- Encomendas à central, manualmente ou através das vendas efetuadas;

- Análise de vendas e estatísticas variadas.

O *software* POS é destinado a cadeias de lojas que necessitam de ter um sistema integrado de gestão de vendas. Toda a informação processada nas lojas é comunicada para a "central" permitindo uma análise global do estado de vendas de toda a cadeia.

5.3 Oportunidades Azure no âmbito da empresa

5.3.1 Portefólio de produtos

Os produtos que a empresa possui são variados e a possibilidade de "estender" e adaptar os *softwares* base fazem com que esta abranja diversas áreas de negócio. Atualmente a empresa possui um número elevado de clientes e embora esteja a dar os primeiros passos numa política de expansão para novos mercados, grande parte dos clientes atuais são nacionais.

Analisando os produtos e clientes que a empresa possui e o estado atual do mercado nacional, o Azure poderia vir a ser uma solução benéfica podendo a migração de grande parte do *software* atual trazer algumas vantagens, conforme justificado de seguida:

- *LabWay LIMS* – Atualmente o *software* com maior quota de mercado. É líder nacional no nicho de mercado de análises laboratoriais. A disponibilização de *software* do tipo SaaS abriria o mercado dos pequenos laboratórios que atualmente não dispõem de condições financeiras para a aquisição do produto. Atingiria também novos mercados com necessidades similares, independentemente do tamanho do laboratório em causa;

- *ERP* – Constitui o produto com maior concorrência no mercado. Passar esta solução para uma solução *web-based* migrando-a para a Azure permitiria atingir empresas geograficamente distantes, independentemente da sua dimensão e das necessidades

de escalabilidade e elasticidade que possuíssem, passando a empresa a disponibilizar um produto com mecanismos de balanceamento de carga e tolerância a falhas, garantidos pela própria infraestrutura;

- *Quality* – Um pouco como o ERP, o objetivo seria expandir esta solução para novos mercados através da utilização da plataforma Azure que não só dotaria a empresa de uma certa independência relativamente ao *hardware*, como também garantiria uma infraestrutura fiável sem elevados investimentos iniciais. Disponibilizava-se assim uma solução robusta com custo variável, sendo completamente adaptável ao cliente e configurável por este, através da disponibilização do *software* como serviço;

- *Portals* – As soluções do tipo *Portals* são as únicas do tipo *web*. A migração para Azure deste tipo de soluções seria a menos trabalhosa comparativamente com as restantes e teria como principal objetivo permitir que a empresa concretizasse projetos de curta duração e/ou com número de utilizadores incerto (não sendo necessário a compra/instalação/configuração de máquinas servidoras). Permitiria também fornecer aplicações com elevado grau de elasticidade e escalabilidade para clientes que assim o necessitassem;

- *Logistics* – O desafio a nível do *Logistics* seria colocar todo o sistema a funcionar *online*. Com a banalização das redes sem fios e da banda larga móvel, para além do *software Logistics*, todo o *software* presente nos *Pocket PCs* poderia ser migrado. Isto reduziria o tempo necessário com a instalação do *software* não só nos PCs, como nos próprios *Pockets*, condição essencial para a expansão para redes de logística e distribuição com elevada dimensão e geograficamente dispersas;

- *POS* – O *software* POS exige atualmente um investimento inicial que retira mercado à empresa, uma vez que pressupõe a existência de uma arquitetura rígida para o seu funcionamento (obrigatoriamente um servidor e os postos necessários que com este comunicam). Com a migração para Azure, esta solução poderia ser disponibilizada como serviço sendo o investimento inicial reduzido. Com este serviço pequenos clientes passariam a fazer parte do universo abrangido pela solução POS. À medida que fossem surgindo novos clientes, apenas seria necessário disponibilizar um novo posto, quer para lojas singulares como para grandes cadeias lojistas. Eliminavam-se

problemas relacionados com a instalação e manutenção dos terminais, sendo apenas necessária uma ligação à internet ativa.

Por outro lado, a migração para Azure poderia trazer algumas *desvantagens* para a empresa, sendo que em alguns casos a sua migração poderia não ser justificável, conforme fundamentado nos seguintes pontos:

- *Art-Center e DONOR* – Ao contrário das soluções referidas anteriormente, estas seriam as que menos vantagens trariam à empresa numa eventual migração para Azure. Nas anteriores pressupõe-se uma transição para soluções do tipo SaaS, de forma a abordarem-se novos mercados e clientes com necessidades aplicacionais (funcionalidades) semelhantes àquelas que os clientes atuais possuem. O *Art-Center* e o *DONOR*, por sua vez, constituem soluções extremamente específicas, direcionadas para um conjunto de clientes muito restrito e cuja solução exige uma forte componente de consultadoria. Só com este acompanhamento constante é que as necessidades reais e específicas de cada processo podem ser atingidas, não constituindo soluções que facilmente possam ser parametrizáveis de uma forma genérica;

- *LabWay LIMS*, *ERP*, *Quality, Logistics* e *POS* – A disponibilização deste tipo de *software* como serviço poderia trazer algumas desvantagens à empresa. A falta de experiência relativamente a SaaS seria a primeira delas, o que levaria a que a migração das soluções para a nova plataforma e a adaptação à nova metodologia de desenvolvimento fosse um processo moroso.

  A nível financeiro, o principal problema na aposta deste tipo de soluções seria a perda da componente de consultadoria, que atualmente atinge cerca de metade da faturação da empresa. A ideia por trás da implementação de soluções do tipo SaaS seria vender um produto com um pacote base de funcionalidades, completamente configurável pelo cliente e autoexplicativo (através de vídeos, ajuda inteligente na instalação e configuração do produto,..). Tendo em conta a legislação de cada mercado (documentos emitidos, formatos,..) os produtos teriam um conjunto de funcionalidades principais que lhes permitiria uma simples adaptação à maioria do mercado de cada uma das soluções. Se os clientes necessitassem de funcionalidades avançadas ou algum auxílio extra (formação remota, consultadoria de processos,..)

ser-lhes-ia cobrado para além do serviço base. Optando apenas pelo "produto base" perder-se-ia um pouco a identidade da empresa uma vez que o *software* nunca seria tão adaptado ao cliente como atualmente é, sendo esta uma característica muito valorizada pelos atuais clientes da empresa.

Outro ponto negativo é a existência no mercado de várias ofertas SaaS no âmbito de alguns destes produtos, nomeadamente de faturação e logística contando-se atualmente com soluções de várias empresas (SAP [91], PHC [92], Primavera [93], IBS [94],..). Algumas soluções SaaS possuem um determinado período de experiência sendo outras gratuitas até determinado nível de utilização (ex: transações/mês), o que faria com que os novos produtos já tivessem alguma concorrência.

Outra desvantagem decorrente da migração de soluções para Azure seria a dependência que a empresa passaria a contrair face ao fornecedor do serviço, neste caso a Microsoft. Vendendo produtos do tipo SaaS, parte dos custos imputados ao cliente seriam utilizados para o pagamento da fatura ao fornecedor do serviço. Esta dependência faria com que qualquer subida de custos a nível do Azure se refletisse na fatura final para o cliente, o que em certos casos poderia não ser facilmente suportado. A dependência referida é acentuada pelo fato de qualquer solução desenhada especificamente para Azure, não ser facilmente portável para uma outra plataforma.

5.3.2 Mercados e áreas de negócio

A migração das soluções atuais para Azure e o desenvolvimento de novas soluções para esta plataforma permitiria uma disponibilização de *software* como serviço, conforme já referido. O tipo de clientes e áreas de negócio da empresa teriam uma mudança significativa, permitindo-lhe:

- Atingir mercados que atualmente não dispõem de condições financeiras para a aquisição das soluções atuais;
- Atingir pequenas empresas que no futuro poderão crescer e sobrescrever um maior número de soluções da empresa (*startups*,..);

- Atingir mercados em zonas geográficas dispersas com necessidades aplicacionais similares;

- Atingir grandes empresas com elevadas necessidades de escalonamento, mecanismos de balanceamento de carga e/ou tolerância a falhas;

- Ter a possibilidade de entrar em concursos e projetos de curta duração (e/ou com número de utilizadores incerto) de forma a angariar novos clientes para as soluções atuais da empresa.

Por outro lado, a reação por parte dos clientes seria incerta, uma vez que existem certos pontos que lhes poderiam criar algum ceticismo quanto à mudança, nomeadamente [89]:

- *Segurança e privacidade* – Com os seus dados na *cloud*, os clientes poderiam questionar a segurança da *cloud*: Poderão pessoas sem autorização aceder a dados confidenciais? Os dados poderão ser de alguma forma perdidos? Os fornecedores de *cloud computing* garantem a segurança dos dados e a sua preservação através de replicação de dados por várias máquinas, porém poderão ter 100% de certeza relativamente à sua segurança [90]?

- *Conexão* – Soluções na *cloud* exigem sempre uma conexão à internet. Sem conexão ativa não se pode efectuar nenhum tipo de operação nem aceder aos dados da aplicação (a não ser que sejam efetuados *backups* de dados da *cloud*, o que traz custos acrescidos). Para além disto, a conexão não poderá ser de baixa velocidade, uma vez que os serviços *web* requerem alguma largura de banda para gerar *outputs* "ricos";

- *Velocidade* – Mesmo com uma conexão de alta velocidade, as aplicações *web* poderão ser por vezes mais lentas do que uma aplicação local semelhante. Tudo tem que ser enviado e recebido entre o computador local e a *cloud*, pelo que qualquer pequeno problema na rede ou nos servidores *cloud* poderão refletir-se na velocidade da aplicação;

- *Custos Variáveis* – Contrastando com os modelos tradicionais, o regime de pagamento de soluções baseadas em *cloud computing* é variável, pagando-se os recursos efetivamente consumidos. Esta questão é por um lado positiva, uma vez que permite

a adaptação dos custos face aos recursos efetivamente necessários, porém conta sempre com componentes variáveis (ex: tráfego de/para *cloud*) o que faz com que não seja possível prever um valor fixo de gastos mensais.

5.3.3 Recursos Humanos

Em termos de recursos humanos, a transição aplicacional para Azure traria um grande desafio aos *developers* que atualmente fazem parte da organização.

Desde a criação da empresa, a maior parte destes estão afetos ao desenvolvimento e manutenção de aplicações *windows-based,* maioritariamente com uma topologia cliente-servidor.

A transição para o desenvolvimento de aplicações *web-based* seria o primeiro desafio a superar, que envolveria um período de adaptação considerável. O segundo desafio seria a mudança de paradigma no desenvolvimento aplicacional e na manutenção das plataformas dos clientes.

Atualmente, para além do desenvolvimento e manutenção das aplicações, os *developers* são responsáveis pela instalação, parametrização e manutenção das máquinas servidoras e das máquinas locais dos clientes, o que faz disparar o número de horas utilizadas a resolver problemas técnicos, que seriam mitigados com a utilização de tecnologias *web-based*. Com a migração para Azure, existiriam apenas serviços disponibilizados via *web*, bastando parametrizar as diversas aplicações, para as adaptar aos processos dos diferentes clientes. Não existiriam instalações nem manutenções de máquinas, libertando os recursos humanos para o desenvolvimento efetivo de aplicações e sua manutenção.

A transição da mentalidade dos recursos humanos para um pensamento *web-based* e mais concretamente *cloud-based*, seria um processo gradual que envolveria um espirito inovador e empreendedor por parte destes, mas que a médio-longo prazo traria claros benefícios não só para a empresa como para os próprios trabalhadores.

6 Conclusões

6.1 Resposta ao problema

A inovação é indissociável do mercado e da sociedade atual. Diariamente surgem novos produtos e tecnologias que motivam pessoas e empresas a evoluírem, estudando novos paradigmas e explorando novos mecanismos, que poderão vir a revelar-se como mais-valias pessoais/organizacionais.

O *Cloud Computing* é um destes paradigmas. Surge para dar resposta às aplicações que cada vez mais exigem poder de computação, flexibilidade, escalabilidade, redundância, mecanismos de balanceamento de carga e tolerância a falhas. Mitiga os problemas relacionados com a capacidade da infraestrutura aplicacional, garantindo fiabilidade e permitindo um ajuste dinâmico da resposta aplicacional face à carga verificada.

Paradigmas com este grau de importância são incontornáveis na área de desenvolvimento de *software*. Foi esta a motivação que suscitou o interesse pessoal e organizacional no aprofundamento de conhecimento nesta área.

A proposta de uma empresa de topo (Microsoft) para o desenvolvimento de uma aplicação sobre Azure, constituiu uma motivação extra para o acompanhamento desta tecnologia emergente. O objetivo era não só aprender o conceito de *cloud computing* (e de Azure mais concretamente), mas também dar os primeiros passos no que toca ao desenvolvimento segundo esta tecnologia.

Com este *know-how* a empresa pretende entrar no desenvolvimento de soluções para a *Cloud,* alargando a sua oferta relativamente a soluções SaaS. Desta forma poderá passar a:

- Disponibilizar soluções de *Cloud Computing* a clientes de grande dimensão, que possuem elevadas necessidades de escalabilidade e elasticidade nas suas aplicações;
- Fornecer serviços que não envolvam elevados investimentos iniciais por parte dos clientes;
- Desenvolver projetos para períodos de tempo curtos e/ou cujo número de utilizadores é incerto (como é o caso deste *case study*);
- Tirar partido deste tipo de soluções para reestruturar algumas das aplicações que atualmente possui.

6.2 Avaliação de objetivos

O presente trabalho trouxe diversos benefícios e explorou algumas oportunidades em linha com os objetivos da empresa.

Iniciando-se um estudo acerca do conceito de *cloud computing*, estudou-se a génese deste paradigma e os avanços históricos por parte de múltiplas empresas que contribuíram para a formação de alguns dos conceitos que hoje se associam ao paradigma *cloud computing*. Estudaram-se os diferentes tipos de implementações de *cloud computing*, bem como as principais soluções que atualmente existem no mercado. A plataforma Azure teve especial enfoque, devido à ligação entre a empresa e a Microsoft. A este nível, estudaram-se os diferentes conceitos que o Azure possui e que o identificam como solução *cloud computing* concluindo-se que:

- O Azure encontra-se em linha com o conceito de *cloud computing*, aplicando corretamente o conceito de *utility computing*, vendendo recursos computacionais de acordo com as necessidades aplicacionais requeridas, sendo apenas taxados os recursos utilizados (a partir do momento que é feito o *deploy* da aplicação);
- Tal como as ofertas típicas de *cloud computing*, é uma solução flexível, escalável e redundante, que possui mecanismos de balanceamento de carga e tolerância a falhas;

- Tal como a maior parte dos fornecedores de serviços *cloud computing*, garante a fiabilidade do serviço através de SLAs, com rácios de *uptime* que tipicamente rondam os 100%;

- É através da virtualização de recursos computacionais, que possui a capacidade de oferecer computação elástica, permitindo um ajuste dinâmico dos recursos de computação e de armazenamento de forma simplificada;

- A sua topologia garante que todos os pedidos são processados por um *load balancer* antes de serem reencaminhados para as instâncias de computação, instancias estas independentes e classificadas segundo a sua funcionalidade (*Web Role* para suporte do *front-end* das aplicações e *Worker Role* para o processamento de tarefas em *background*). As *queues* são o mecanismo para troca de mensagens entre os diferentes componentes, permitindo desacoplar as diferentes partes da aplicação, obtendo-se uma maior flexibilizar e escalabilidade na utilização de recursos;

- A *Storage* garante um armazenamento dinâmico na *Cloud*, permitindo o armazenamento de dados em múltiplos formatos e adereçando os problemas de escalabilidade, concorrência, distribuição e descentralização dos dados:

    - Dados armazenados na *storage* replicadas três vezes no mesmo *datacenter* de forma a prevenir falhas de *hardware* e geograficamente entre dois *datacenters* (no caso das *Blobs* e *Tables*) de forma a salvaguardar os dados em casos de desastres naturais;

    - *Escalabilidade* – particionamento da informação pelos múltiplos servidores através da monitorização de padrões de utilização e balanceamento de carga pelos vários nós servidores;

    - *Consistência e concorrência* – mecanismos como *snapshot isolation* para obtenção de dados coerentes no início de processos transacionais e *optimistic concurrency* para submissão de transações, caso os dados a alterar não tenham sofrido alterações no armazenamento;

- O SQL Azure é um serviço de base de dados de alta disponibilidade na *Cloud*, escalável e com mecanismos de tolerância a falhas que replica múltiplas cópias de dados para

diversos servidores físicos, de forma a preservar a disponibilidade destes e a continuação do negócio em caso de falha. O *SQL Azure Fabric* (componente responsável pela gestão das múltiplas instancias SQL Server) é o responsável pelas questões de *load balancing* e *failover*.

A proposta da Microsoft para o desenvolvimento de uma aplicação para promoção de um novo produto, conjugada com o estudo aprofundado da plataforma Azure, levou ao desenvolvimento de um protótipo prático sobre esta plataforma, dado que o número de utilizadores que a solução teria que suportar era incerto e que o período de tempo que iria estar *online* era bastante reduzido (cerca de um mês).

O protótipo prático permitiu desenvolver competências no âmbito do desenho de soluções para a plataforma Azure. Desenvolveu-se uma aplicação adaptada à sua topologia, utilizando-se *web roles* para processamento de todos os pedidos do cliente no *site* e *worker roles* para o processamento e envio de *e-mails*, com a comunicação entre estes a ser realizada através de *queues*. Com recurso a *tables* e *blobs*, mitigou-se o problema de sessões em aplicações distribuídas e para armazenamento de dados aplicacionais utilizou-se o SQL Azure. A par das particularidades do Azure, estudaram-se as redes sociais Facebook e Twitter, com maior incidência sobre o Facebook e respetiva API, devido ao requisito de autenticação e obtenção de informações do utilizador a partir deste sistema.

No final foi feito o *deploy* da aplicação e BD para a *Cloud* da Microsoft, cumprindo-se três dos quatro objetivos propostos para o trabalho (Estudar a génese do *Cloud Computing* e os tipos de soluções existentes; Analisar a plataforma Azure e a forma como mitiga os problemas típicos das aplicações atuais; Criar um protótipo prático e colocá-lo a correr na *cloud*). O último objetivo (Analisar, mediante o contexto atual da empresa, o impacto que a adoção do Azure traria) foi concretizado no capítulo 5 (Análise no contexto da empresa), em que se estudou o contexto atual da empresa e as vantagens e desvantagens que o Azure traria a nível de portefólio aplicacional, mercados/áreas de negócio e recursos humanos.

Concluiu-se com esta análise que o Azure poderia trazer múltiplas vantagens para a empresa uma vez que, passando a disponibilizar soluções do tipo SaaS, alargaria o seu mercado alvo atingindo não só pequenas empresas com recursos financeiros limitados, como clientes de dimensão considerável com necessidades de escalabilidade e elasticidade elevadas. Passaria a contar com soluções robustas, com custos variáveis e com mecanismos de balanceamento de

carga e tolerância a falhas, permitindo a concretização de projetos de curta duração e/ou projetos com um número de utilizadores incerto ou extremamente variável.

Por outro lado, alertou-se para a necessidade de adaptação dos recursos humanos à nova metodologia de desenvolvimento; para a possível perda financeira da componente da consultadoria e da identidade da empresa (software mais genérico e menos adaptado ao cliente); para a reação deste face à adoção de uma solução *cloud* (segurança e privacidade da informação, conexão e velocidade, custos variáveis,..); para a concorrência que se teria que enfrentar no mercado dos SaaS e para a dependência que a empresa passaria a ter face à Microsoft: financeira (custos *cloud*) e aplicacional (portabilidade da aplicação).

Para além do desenvolvimento pessoal pode então concluir-se que o estudo, no seu todo, trouxe benefícios e oportunidades para a empresa explorar, nomeadamente:

- Passou a existir um *know-how* no desenvolvimento de soluções para a *Cloud* e na integração com redes sociais;
- Alertou a empresa para novas possibilidades no desenvolvimento aplicacional e o impacto que este desenvolvimento traria a nível de produtos, mercados, áreas de negócio e recursos humanos;
- Possibilitou uma aproximação aos objetivos estratégicos da empresa uma vez que, através do desenvolvimento de SaaS, poderá aumentar a sua capacidade competitiva (aplicações com grande capacidade e com um custo moderado) e alargar a sua oferta a novos mercados (grandes cliente, clientes geograficamente distantes, *startups* que possam vir a crescer,..).

6.3 Limitações e trabalho futuro

Este trabalho teve como principal objetivo o estudo do paradigma *cloud computing* e mais concretamente da solução fornecida pela Microsoft, a plataforma Azure.

O estudo realizado e a necessidade de criação de uma primeira aplicação prática em Azure possibilitaram, respondendo ao problema em concreto, desenvolver competências para a futura criação de aplicações sobre esta plataforma. É assim natural, até pelo curto período de

desenvolvimento do mesmo, que algumas das funcionalidades que o Azure possui não tenham sido exploradas no seu desenvolvimento. Pretendeu-se abranger o máximo de *know-how* possível com este estudo, tendo o protótipo prático servido como primeira experiência no desenvolvimento de aplicações para esta plataforma.

Conforme o estudo de benefícios revelou, existem várias oportunidades que a empresa pode explorar com o desenvolvimento de soluções em Azure. Assim sendo, este trabalho poderá ser o ponto de partida na migração de parte das soluções da empresa para o paradigma de *cloud computing*. Pretende também contribuir para uma mudança de mentalidade no desenho de novos produtos, de forma a simplificar a concretização dos objetivos estratégicos da empresa de expansão e procura de novos mercados.

Referências

[1] Hilley, David. "Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings". Abril 2009.

[2] Polze, Prof. Dr. Andreas. "A Comparative Analysis of Cloud Computing Environments". 2009.

[3] Armbrust, Michael et all. "Above the Clouds: A Berkeley View of Cloud Computing". Fevereiro 2009.

[4] Calder, Brad; Tony Wang, Shane Mainali, and Jason Wu. "Windows Azure Blobs". Fevereiro 2010.

[5] Haridas, Jai; Niranjan Nilakantan, and Brad Calder. "Windows Azure Tables".

[6] Schwegler, Beat. "Head in the Cloud, Feet on the Ground".

[7] "Windows Azure Queue - Programming Queue Storage". Dezembro 2008. http://www.microsoft.com/windowsazure/whitepapers

[8] Krishnaswamy, Jayaram. "Microsoft SQL Azure: Enterprise Application Development". Dezembro 2010.

[9] "Azure Support". http://www.azuresupport.com (Acedido a 28-10-2010)

[10] "Windows Azure". http://www.microsoft.com/windowsazure (Acedido a 28-10-2010)

[11] "SQL Azure Architecture". http://msdn.microsoft.com/en-us/library/windowsazure/ee336271.aspx (Acedido a 19-02-2012)

[12] "SQL Azure Data Access". http://msdn.microsoft.com/en-us/library/windowsazure/ee336239.aspx (Acedido a 19-02-2012)

[13] "Transact-SQL Support (SQL Azure Database)". http://msdn.microsoft.com/en-us/library/windowsazure/ee336250.aspx (Acedido a 19-02-2011)

[14] "Accounts and Billing in SQL Azure". http://msdn.microsoft.com/en-us/library/windowsazure/ee621788.aspx (Acedido a 19-02-2011)

[15] "Table Service REST API". http://msdn.microsoft.com/en-us/library/dd179423.aspx

[16] "Blob Service REST API". http://msdn.microsoft.com/en-us/library/dd135733.aspx

[17]“Windows Azure”. http://www.microsoft.com/windowsazure/windowsazure

[18]“Querying Tables and Entities”. http://msdn.microsoft.com/en-us/library/dd894031.aspx

[19]H. Krawczyk, M. Bellare, R. Canetti. “HMAC: Keyed-Hashing for Message Authentication”. IETF RFC 2104. Fevereiro 1997. http://www.ietf.org/rfc/rfc2104.txt

[20]R. Fielding et al. “Hypertext Transfer Protocol -- HTTP/1.1”, World Wide Web Consortium (W3C) note. Junho 1999. http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.19

[21]“Amazon Elastic Compute Cloud (Amazon EC2)”. http://aws.amazon.com/ec2

[22] Exforsys Inc. “Mainframes Computers”. Maio 2007. http://www.exforsys.com/tutorials/mainframe/mainframes-computers.html

[23]“Can autonomic computing implemented successfully?”. http://www.computeruser.com/articles/can-autonomic-computing-implemented-successfully.html (Acedido a 26-02-2011)

[24]“The Web And Web Hosting: A Beginner’s Guide!”. http://www.alreadyhosting.com/articles/beginners-guide-to-web-hosting.php

[25]“Cloud Computing Shakes Up Traditional IT Outsourcing2. Overby, Stephanie. 2010. CIO. http://www.cio.com/article/590115/Cloud_Computing_Shakes_Up_Traditional_IT_Outsourcing

[26]“Cloud Computing Expo: Introducing the Cloud Pyramid”. Sheehan, Michael. Agosto 2008. Cloud Computing Journal. http://cloudcomputing.sys-con.com/node/609938

[27]“Understanding Public Clouds: IaaS, PaaS, & SaaS”. Pijanowski, Keith . Maio 2009. http://www.keithpij.com/Home/tabid/36/EntryID/27/Default.aspx

[28]“Amazon Web Services”. http://aws.amazon.com/ (Acedido a 14-01-2012)

[29]“Go Grid”. http://www.gogrid.com/ (Acedido a 14-01-2012)

[30]“The Rackspace Cloud”. http://www.rackspace.com/cloud/ (Acedido a 14-01-2012)

[31]“What Is Google App Engine?”. http://code.google.com/appengine/docs/whatisgoogleappengine.html (Acedido a 14-01-2012)

[32]"Windows Azure". http://www.windowsazure.com/en-us/ (Acedido a 07-01-2012)

[33]"The Social Enterprise Platform". http://www.salesforce.com/platform/ (Acedido a 21-01-2012)

[34]"Microsoft Online Services". http://www.microsoft.com/online/en-gb/default.aspx (Acedido a 21-01-2012)

[35]"Trusted cloud apps and platform for the Social Enterprise™". http://www.salesforce.com/products/ (Acedido a 21-01-2012)

[36]"Lotus Live". https://www.lotuslive.com/ (Acedido 21-01-2012)

[37]"Integrated Development Environment – IDE". http://www.explainstuff.com/classes/java/tools-to-be-used-for-programming/integrated-development-environment-ide (Acedido a 26-02-2011)

[38]"Work Breakdown Structure: Purpose, Process and Pitfalls". http://www.projectsmart.co.uk/work-breakdown-structure-purpose-process-pitfalls.html (Acedido a 26-02-2011)

[39]"IIS Server Variables". http://msdn.microsoft.com/pt-pt/library/ms524602(v=vs.90).aspx (Acedido a 08-10-2011)

[40]"Windows Azure Training Kit". http://www.microsoft.com/download/en/details.aspx?id=8396 (Acedido a 09-10-2011)

[41] "CloudStorageAccount e o método SetConfigurationSettingPublisher". Meriat, Vitor. Julho 2011. TechNet Articles.

http://social.technet.microsoft.com/wiki/contents/articles/cloudstorageaccount-e-o-metodo-setconfigurationsettingpublisher.aspx

[42]"Advanced scenarios with Windows Azure Queues". Balliauw, Maarten. Junho 2011.

http://www.developerfusion.com/article/120619/advanced-scenarios-with-windows-azure-queues/

[43]"Google Analytics". http://code.google.com/intl/pt-PT/apis/analytics (Acedido a 24-02-2011)

[44]"Google Analytics - Tracking Basics (Asynchronous Syntax)". http://code.google.com/intl/pt-PT/apis/analytics/docs/tracking/asyncUsageGuide.html (Acedido a 24-02-2011)

[45]"How to Configure Virtual Machine Sizes". http://msdn.microsoft.com/en-us/library/ee814754.aspx (Acedido a 17-02-2011)

[46]"Windows Azure Compute". http://www.microsoft.com/windowsazure/features/compute (Acedido a 17-02-2011)

[47]"Windows Azure Compute". http://www.windowsazure.com/en-us/home/tour/compute (Acedido a 11-02-2012)

[48]"Session State in Windows Azure". White, Jim. Agosto 2010. http://www.intertech.com/Blog/post/Session-State-in-Windows-Azure.aspx (Acedido a 21-02-2011)

[49]"Windows Azure Blobs". http://social.msdn.microsoft.com/Forums/en/windowsazure/thread/7ddc0ca8-0cc5-4549-b44e-5b8c39570896 (Acedido a 21-02-2011)

[50]"Windows Azure Storage". http://www.microsoft.com/windowsazure/features/storage (Acedido a 17-02-2011)

[51]"Windows Azure Storage". http://www.windowsazure.com/en-us/home/tour/storage (Acedido a 12-02-2012)

[52]"Singleton". http://msdn.microsoft.com/en-us/library/ff650849.aspx (Acedido a 21-02-2011)

[53]"Autentication". Facebook Developpers. http://developers.facebook.com/docs/authentication (Acedido a 23-02-2011)

[54]"Graph API". Facebook Developpers. http://developers.facebook.com/docs/reference/api (Acedido a 23-02-2011)

[55]"Aplications". http://developers.facebook.com/setup (Acedido a 23-02-2011)

[56] E. Hammer-Lahav,D. Recordon, D. Hardt. "The OAuth 2.0 Protocol draft-ietf-oauth-v2-05". Internet-Draft. Maio 2010. http://tools.ietf.org/html/draft-ietf-oauth-v2-05#section-3.5.2

[57]"Using ASP.Net with Facebook's Graph API and OAuth 2.0 Authentication". Abril 2010. http://osnapz.wordpress.com/2010/04/23/using-asp-net-with-facebooks-graph-api-and-oauth-2-0-authentication (Acedido a 23-02-2011)

[58]“Permissions”. Facebook Developpers. http://developers.facebook.com/docs/reference/api/permissions (Acedido a 23-02-2011)

[59]“HttpHandlers”. http://msdn.microsoft.com/en-us/library/5c67a8bd(v=VS.71).aspx (Acedido a 23-02-2011)

[60]R. Fielding et al. “Hypertext Transfer Protocol -- HTTP/1.1”, World Wide Web Consortium (W3C) note. Junho 1999. http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3 (Acedido a 23-02-2011)

[61]“HowTo-Rest”. http://developer.yahoo.com/dotnet/howto-rest_vb.html (Acedido a 23-02-2011)

[62]D. Crockford. “The application/json Media Type for JavaScript Object Notation (JSON)”. RFC 4627. Julho 2006. http://tools.ietf.org/html/rfc4627

[63]“Facebook Share”. Facebook Developpers. http://developers.facebook.com/docs/share/ (Acedido a 24-02-2011)

[64]“Debugger”. http://developers.facebook.com/tools/debug (Acedido a 24-02-2011)

[65]“Twitter Buttons”. http://twitter.com/about/resources/tweetbutton (Acedido a 24-02-2011)

[66]“How to Post Shortened Links (URLs)”. https://support.twitter.com/articles/332912-como-encurtar-links-urls (Acedido a 24-02-2011)

[67]“MailMessage Class”. http://msdn.microsoft.com/en-us/library/system.net.mail.mailmessage.aspx (Acedido a 24-02-2011)

[68]“SmtpClient Class”. http://msdn.microsoft.com/en-us/library/system.net.mail.smtpclient.aspx (Acedido a 24-02-2011)

[69]“Dialog”. http://jqueryui.com/demos/dialog (Acedido a 24-02-2011)

[70]“How Do I Display a Dialog Box?”. Chambers, James. http://www.asp.net/ajaxlibrary/jquery_ui_mvc_dialog.ashx (Acedido a 24-02-2011)

[71]“JQuery UI Dialog with ASP.NET button postback”. http://stackoverflow.com/questions/757232/jquery-ui-dialog-with-asp-net-button-postback (Acedido a 24-02-2011)

[72]"Email Address". http://www.regular-expressions.info/regexbuddy/email.html (Acedido a 24-02-2011)

[73]"Migrating Databases to SQL Azure". http://msdn.microsoft.com/en-us/library/windowsazure/ee730904.aspx (Acedido a 25-02-2011)

[74]"Bcp Utility". http://msdn.microsoft.com/en-us/library/ms162802(SQL.105).aspx (Acedido a 25-02-2011)

[75]"Security Guidelines and Limitations (SQL Azure Database)". http://msdn.microsoft.com/en-us/library/windowsazure/ff394108.aspx#encryption (Acedido a 25-02-2011)

[76]"Publishing a Windows Azure Application using the Windows Azure Tools". http://msdn.microsoft.com/en-us/library/ff683672.aspx (Acedido a 25-02-2011)

[77]"How to Use the Blob Storage Service". http://www.windowsazure.com/en-us/develop/net/how-to-guides/blob-storage/ (Acedido a 29-01-2012)

[78]"Understanding Block Blobs and Page Blobs". http://msdn.microsoft.com/en-us/library/windowsazure/ee691964.aspx (Acedido a 29-01-2012)

[79]"Understanding Block Blobs and Page Blobs". http://msdn.microsoft.com/fr-fr/library/ee691964.aspx (Acedido a 04-01-2012)

[80]"Save your job with Windows Azure". Koval, Alexander. Fevereiro 2010. http://www.codemastersintl.com/Blogs/Alexander-Koval/2010/2/27/save-your-job-with-windows-azure (Acedido a 20-05-2012)

[81]"Compare SQL Server with SQL Azure". Nethi, Dinakar. Junho 2010. TechNet Articles. http://social.technet.microsoft.com/wiki/contents/articles/996.compare-sql-server-with-sql-azure.aspx

[82]"Accounts and Billing in SQL Azure". http://msdn.microsoft.com/en-us/library/ee621788.aspx (Acedido a 22-05-2012)

[83]"SQL Azure". http://www.windowsazure.com/en-us/home/features/sql-azure/ (Acedido a 22-05-2012)

[84]"Management Portal for SQL Azure". http://msdn.microsoft.com/en-us/library/gg442309.aspx (Acedido a 22-05-2012)

[85]"Supported Transact-SQL Statements (SQL Azure Database)". http://msdn.microsoft.com/en-us/library/ee336270.aspx (Acedido a 22-05-2012)

[86]"Partially Supported Transact-SQL Statements (SQL Azure Database)". http://msdn.microsoft.com/en-us/library/ee336267.aspx (Acedido a 22-05-2012)

[87]"Unsupported Transact-SQL Statements (SQL Azure Database)". http://msdn.microsoft.com/en-us/library/ee336253.aspx (Acedido a 22-05-2012)

[88]"Understanding the Cloud Computing Stack: PaaS, SaaS, IaaS". Kepes, Ben. 2011. http://broadcast.rackspace.com/hosting_knowledge/whitepapers/Understanding-the-Cloud-Computing-Stack.pdf

[89]"Cloud Computing Pros and Cons for End Users". Miller, Michael. Fevereiro 2009. http://www.quepublishing.com/articles/article.aspx?p=1324280&seqNum=2

[90]"Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data". Blodget, Henry. Abril 2011. http://articles.businessinsider.com/2011-04-28/tech/29958976_1_amazon-customer-customers-data-data-loss

[91]"Overview | Cloud Computing | SAP". http://www.sap.com/solutions/technology/cloud/overview/index.epx (Acedido a 07-06-2012)

[92]"SaaS e Cloud Computing com PHC FX". http://www.phc.pt/portal/programs/ewpview.aspx?codigo=saas (Acedido a 07-06-2012)

[93]"Oferta Primavera na Cloud". http://www.primaverabss.com/pt/Solu%C3%A7%C3%B5es-Solu%C3%A7%C3%B5es%20na%20Cloud-Oferta%20PRIMAVERA%20na%20Cloud.aspx (Acedido a 07-06-2012)

[94]"Software as a Service from IBS". http://www.ibsplc.com/Software-as-a-Service-from-IBS.html (Acedido a 07-06-2012)

Anexo 1 – Google Analytics