UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Departamento de Ciência da Computação

Criação de aplicações para Cloud Computing utilizando a plataforma Windows Azure

André Luiz Campos Esqueff Abdalla

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Eduardo Berrére

Juiz de Fora, MG

Dezembro de 2009

André Luiz Campos Esqueff Abdalla

Criação de aplicações para Cloud Computing utilizando a plataforma Windows Azure

Monografia submetida ao corpo docente do Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora, como parte integrante dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Eduardo Barrére

JUIZ DE FORA - MG
DEZEMBRO, 2009

Criação de aplicações para Cloud Computing utilizando Windows Azure

André Luiz Campos Esqueff Abdalla

Aprovado em _____ de _____ de 2009.

Comissão Examinadora:

_____

Eduardo Barrére - orientador.
DSc. em Engenharia de Sistemas e Computação / COPPE-UFRJ

_____

Eduardo Pagani Julio
MSc em Computação / UFF

_____

Daves Márcio Silva Martins
MSc. em Computação de Alto Desempenho / UFRJ

JUIZ DE FORA
DEZEMBRO, 2009

Resumo

Cloud Computing é um termo utilizado para caracterizar um ambiente de computação baseado em um conjunto de servidores virtuais ou físicos, que agrega na nuvem todo o armazenamento e processamento de uma aplicação. Este trabalho apresenta o processo de criação, publicação e gerenciamento de dados disponíveis no *Windows Azure* para implementação de aplicações para *Cloud Computing*.

Palavras-chave: Cloud Computing, Windows Azure, Virtualização

Abstract

Cloud Computing is a term used to describe a computing environment on a set of physical or virtual servers, which adds in the cloud all the storage and processing of an application. This paper presents the creation, deploying and data management process available in Windows Azure for the implementation of applications for Cloud Computing.

Keywords: Cloud Computing, Windows Azure, Virtualization

Agradecimentos

Primeiramente a toda minha família, com certeza a base para que eu conseguisse chegar onde estou. Em especial minha mãe Creuza, minhas irmãs Marcela e Mariana, Alexandre, Joana, Laura, meu avó José e a memória da minha querida avó Maria José.

À minha namorada Maria Fernanda, pelo apoio e companheirismo em todos os momentos.

Aos meus amigos, sempre comigo nos bons e maus momentos.

Ao meu orientador Eduardo Barrére pela orientação e apoio no desenvolvimento deste trabalho.

Sumário

Lista de Figuras

Lista de Siglas

API: Application Programming Interface

BD: Banco de Dados

DaaS: Database as a Service

GDT: Gerenciador de Tarefas

IaaS: Interface as a Service

PaaS: Platform as a Service

QoS: Quality of Service

SaaS: Software as a Service

SMS: SEL Manegement Studio

VDC: Virtual Data Center

VM: Virtual Machine

VS: Visual Studio

WAT: Windows Azure Tools

WCF: Windows Communication Foundation

Lista de Tabelas

1. Introdução

O surgimento de novos termos e tecnologias é constante na área da computação, sendo que em muitas situações há pouca informação disponível acerca deste novo termo. Isto tem ocorrido com o *Cloud Computing*, que vem sendo cada vez mais utilizado e é crescente a procura por informações nos últimos anos, segundo Barbosa e Charão (2009).

O objetivo do trabalho é entender e aplicar o processo de desenvolvimento de uma aplicação, utilizando ferramentas do *Windows Azure,* plataforma da *Microsoft* para *Cloud Computing*, que seja capaz de ser totalmente processada e armazenada em um servidor acessível pela internet, não utilizando o *hardware* da máquina do usuário.

São objetivos específicos deste trabalho ampliar o conhecimento sobre o desenvolvimento de aplicações para *Cloud Computing*, estudar a ferramenta *Windows Azure*, no que se refere aos padrões por ela adotados, seus potenciais e limitações, e desenvolver uma aplicação para *Cloud Computing* comparando o processo a um de desenvolvimento de uma aplicação *web* tradicional.

Inicialmente é apresentada uma contextualização tecnológica sobre *Cloud Computing*, onde são apresentadas as categorias de desenvolvimento de aplicações, assim como projetos e plataformas existentes. No fim do capítulo são discutidos alguns obstáculos e possíveis soluções para o desenvolvimento de aplicações utilizando este paradigma.

A seguir, a plataforma da *Microsoft*, conhecida por *Windows Azure* é apresentada. Seus serviços são explicados, assim como padrões de computação utilizados e taxas que são cobradas pelo uso da plataforma.

Já no capítulo seguinte, são apresentados os passos para a implementação de uma aplicação que foi hospedada e gerenciada na nuvem da *Microsoft*, utilizando características de *Cloud Computing* e a plataforma *Windows Azure*.

Finalizando, é apresentada uma comparação entre o desenvolvimento de aplicações *on-premise*, instaladas na máquina do usuário, e *off-premise*, baseando-se nas informações adquiridas durante o desenvolvimento da aplicação do capítulo anterior.

2. Contextualização tecnológica de *Cloud Computing*

Embora a construção e operação de *datacenters*[1] extremamente poderosos tenha sido a chave necessária para o desenvolvimento do modelo, tendências tecnológicas adicionais e novos modelos de negócio também foram responsáveis em torná-lo real. Assim que *Cloud Computing* surgiu, novas oportunidades de aplicações foram descobertas, já que não faziam sentido antes. [ARMBRUST, 2009]

De acordo com Dos Santos e De Menezes (2009), atualmente existe uma crescente demanda de acesso dinâmico e flexível, onde os usuários não estão presos a uma máquina específica, sendo necessário que softwares e serviços estejam disponibilizados em qualquer lugar, através da internet.

Cloud Computing surgiu da necessidade do compartilhamento de ferramentas computacionais, com a utilização da Internet como meio de comunicação entre os sistemas. Neste contexto, os computadores pessoais estarão conectados à rede mundial, a fim de colher dados na nuvem e trazê-los aos usuários, com o processamento ocorrendo no servidor virtual. [DOS SANTOS e DE MENEZES, 2009]

Cloud Computing é definido como: "*Um paradigma de computação em larga escala que possui foco em proporcionar economia de escala, em que um conjunto abstrato, virtualizado, dinamicamente escalável de poder de processamento, armazenamento, plataformas e serviços são disponibilizados sob demanda para clientes externos através da Internet.*" [FOSTER et al., 2009]

Encontram-se uma variedade de produtos utilizando este paradigma, como softwares com código aberto, sendo testados na computação em nuvem. Alguns exemplos são o projeto *Eucalyptus*[2], desenvolvido pela universidade da Califórnia e o *Enomaly*[3], e recentemente a *NASA* anunciou que está investindo em uma iniciativa de *Cloud Computing*, também em código aberto, que poderia ser utilizada como suporte em suas missões espaciais. Esta

[1] Conjunto de computadores responsáveis pelo processamento de dados de uma empresa.

[2] Eucalyptus - http://open.eucalyptus.com/

[3] Enomaly: Elastic / Cloud Computing Platform - http://www.enomaly.com/

iniciativa recebeu o nome de *Nebula*[4]. Outros exemplos, porém de código fechado, merecem destaque: sistemas do *Google*, como *Google Docs* e *Google Pages*. [DOS SANTOS e DE MENEZES, 2009]

Com a evolução deste modelo, o usuário não precisará possuir máquinas muito robustas, e sim uma placa-mãe, processador, RAM e pouca quantidade de espaço na memória persistente, rodando apenas um sistema operacional e um browser conectado à internet. Com esta considerável redução dos componentes, ocorrerão quedas significativas no preço dos mesmos, facilitando o acesso à tecnologia. [DOS SANTOS e DE MENEZES, 2009]

Cloud Computing se diferencia de outros paradigmas importantes da computação, como *Distributed Computing*[5], *Grid Computing*[6] e *Cluster Computing*[7], pelos seguintes aspectos [WANG e VON LASZEWSKI, 2008]:

- Interfaces pensadas no usuário, para que possam acessá-las com simplicidade, da mesma forma que usuários tem acesso à energia elétrica, água e telefone em suas casas.

- Serviços sob demanda, onde o usuário pode customizar o ambiente, instalar *softwares* e configurar o sistema à medida que for necessário, pois possuem privilégios de administrador.

- Qualidade de Serviço (QoS) garantida, como por exemplo, velocidade dos processadores, banda garantida de entrada e saída de dados e memória mínima disponível, de acordo com o contratado.

- Sistema Autônomo, onde *hardware*, *software* e dados que estão na nuvem, podem ser automaticamente reconfigurados e consolidados para finalmente ser entregue aos usuários.

- Escalabilidade e flexibilidade, que são os aspectos mais importantes para a evolução

[4] NASA Nebula – Cloud Computing - http://nebula.nasa.gov/

[5] Combinação de vários computadores realizando a mesma tarefa.

[6] Caso particular da *Distributed Computing*, onde são orientados para aplicações que precisem de uma grande capacidade de cálculos.

[7] Semelhante ao Grid Computing, só que são utilizadas máquinas homogêneas.

do *Cloud Computing*. Serviços e plataformas da nuvem podem ser escaláveis por várias preocupações, como por exemplo, localização geográfica, desempenho de hardware e configurações de software. Uma plataforma baseada neste conceito deve ser flexível para se adaptar a vários requerimentos de um potencial elevado número de usuários.

Neste contexto, a implementação e o uso de aplicações em *Cloud Computing* é uma tendência para várias soluções na área de computação.

Um dos fatores que dificultam os usuários domésticos ou empresas a utilizarem soluções computacionais mais robustas é a necessidade de recursos de *hardwares* constantemente atualizados. Outro ponto que dificulta a adoção de soluções computacionais é o alto custo para se manter um parque informatizado, ou seja, pessoal técnico especializado, recursos de *hardware* e de segurança. Neste cenário, a adoção de soluções computacionais que demandam pouco recurso de *hardware* para o cliente e a diminuição significativa de investimentos em pessoal e segurança, passa a ser um fator importante para a implementação de soluções computacionais.

Hoje em dia, com o acesso em banda larga à Internet mais difundido, o acesso a aplicações que se encontram em locais remotos, normalmente chamados de *datacenters*, se torna uma solução mais plausível. Este tipo de solução pode e deve ser questionada somente se, o local onde ela é remotamente executada, não apresenta os critérios básicos de segurança, disponibilidade e *backup*. O fato dos dados manipulados pela aplicação estarem armazenados em um *datacenter* remoto é um fator que pode ser criticado, porém facilmente resolvido através da manipulação de contratos mais rigorosos sobre o sigilo da informação.

Quando se pensa em aplicações armazenadas e executadas remotamente, ou seja, *off-premise*, alguns obstáculos podem ser encontrados [ARMBRUST, 2009]:

- Disponibilidade do serviço;
- Segurança dos dados;
- Confidencialidade dos dados;
- Congestionamento na transferência de dados;

- Desempenho imprevisível;
- *Bugs* nos sistemas distribuídos.

Além de acessar aplicações e utilizar serviços na nuvem, outro aspecto importante é a possibilidade de armazenar documentos, planilhas, vídeos, fotos ou músicas nestas nuvens. *Cloud Computing* centraliza em *datacenters* estes arquivos, facilitando o acesso através de qualquer lugar, e utilizando dispositivos diferentes.

3. Processo de desenvolvimento e execução de aplicações em *Cloud Computing*

Cloud Computing oferece uma variedade de melhorias aos desenvolvedores de software, como agilidade, eficiência e custos compensadores, acelerando a entrega de novas aplicações. Neste sentido, grandes empresas estão direcionando recursos significativos para a criação de ferramentas de desenvolvimento de serviços para a nuvem. [MILLER, 2009]

Este capítulo está dividido de forma a apresentar tipos de desenvolvimento de serviços na nuvem, plataformas de serviços, projetos e também obstáculos para o uso de *Cloud Computing*.

3.1. Tipos de desenvolvimento de serviços na nuvem

O conceito de desenvolvimento de serviços na nuvem compreende diversos tipos diferentes de desenvolvimento. A seguir são apresentadas diferentes maneiras que uma empresa pode utilizar *Cloud Computing* para desenvolver suas próprias aplicações:

- ***Software as a Service* (SaaS)**, é provavelmente o tipo mais comum. Com ele, uma simples aplicação é disponibilizada a milhares de usuários. Eles não pagam para possuir o software, e sim para usá-lo. O acesso a essas aplicações é feito através de uma API acessível pela internet. Para os clientes, SaaS não requer investimentos iniciais em servidores ou softwares. Para os desenvolvedores, a manutenção é realizada apenas em uma aplicação, que é disponibilizada para muitos usuários. Um exemplo disso são as aplicações para escritório do *Google*, o *Google Docs* [MILLER, 2009];

- ***Platform as a Service* (PaaS)**, é uma variação do SaaS, onde o ambiente de desenvolvimento é oferecido como um serviço. O desenvolvedor utiliza facilidades do ambiente, inclusive para distribuição das aplicações. Por exemplo, *Microsoft Windows Azure*, *Google AppEngine* e *Salesforce* [MILLER, 2009];

- ***Infrastructure as a Service* (IaaS)**, onde são disponibilizados recursos de hardware, como espaço em disco e capacidade de processamento, como o *Amazon S3* [BARBOSA e CHARÃO, 2009];

- ***Database as a Service* (DaaS)**, um tipo especializado de armazenamento que envolve serviços de banco de dados, por exemplo *Amazon SimpleDB*, *Google Big Table* e *Azure SQL Services* [BARBOSA e CHARÃO, 2009];

3.2. Plataformas e projetos baseados em *Cloud Computing*

Com a crescente importância deste paradigma, empresas e universidades investem em projetos e plataformas voltadas a este conceito. Com isso, uma gama de serviços está surgindo, e algumas delas estão apresentadas a seguir.

3.2.1. *Amazon Web Services*

A *Amazon Web Services* [AWS, 2009] é uma plataforma de serviços que implementa uma IaaS, onde se tem acesso a uma suíte de serviços, como processamento, armazenamento, entre outros. Os serviços disponíveis são:

- *Amazon Elastic Compute Cloud* (Amazon EC2), que disponibiliza capacidade computacional redimensionável. O usuário deverá configurar uma máquina virtual específica, *Amazon Machine Instance*, e carregá-la. Escalabilidade de fácil configuração, de acordo com os recursos necessários;

- *Amazon Simple Storage Service* (Amazon S3), interface simples que é utilizada para armazenar e acessar grandes quantidades de dados. Provê uma infraestrutura de armazenamento confiável, rápida e barata, fazendo com que a *Amazon* possua sua própria rede de *sites*;

- *Amazon CloudFront*, o ponto de entrada para a nuvem. Integrada a outros serviços da AWS, disponibilizando uma maneira de distribuir conteúdo aos usuários finais, com baixa latência e altas taxas de transferências de dados;

- *Amazon SimpleDB*, realiza consultas em dados estruturados em tempo real. Integra-se principalmente aos serviços *Amazon S3* e *Amazon EC2*. Funciona de maneira semelhante aos bancos de dados comuns;

- *Amazon Simple Queue Service* (Amazon SQS), serviço confiável e altamente escalável que armazena mensagens, que são transmitidas entre computadores. Ao utilizar esse serviço, desenvolvedores podem mover facilmente dados entre componentes

distribuídos de suas aplicações, realizando diferentes tarefas, sem perder mensagens ou exigir a disponibilidade de cada componente.

3.2.2. *Eucalytpus*

Eucalytpus [NURMI et al., 2009] utiliza a infraestrutura da *Amazon Web Services*, e está disponível para grupos de pesquisa acadêmica, provendo uma plataforma modular e aberta para instrumentação experimental e estudos, esperando com isso criar uma comunidade de desenvolvimento. *Eucalyptus* é composto por diversos componentes que interagem através de uma interface bem definida e desenvolvedores substituem as implementações originais pelas próprias, modificando os módulos existentes. Os componentes são divididos em três níveis, e cada nível foi implementado como um serviço autônomo. São eles:

- *Instance Manager*: controla a execução, inspeção e conclusão das instâncias de VM no servidor;
- *Group Manager*: reúne informações sobre VM e agenda execuções em gerenciadores de instâncias específicos;
- *Cloud Manager*: é o ponto de entrada para esta nuvem, tanto para usuários quanto para administradores. Disponibiliza informações sobre recursos utilizados e disponíveis, e ainda controla decisões sobre agendamentos e faz requisições aos *Group Managers*.

3.2.3. *Sun Open Cloud*

A *Sun Open Cloud* [SUN, 2009] é potencializada pelas tecnologias de software líderes de mercado da empresa, incluindo *Java*, *MySQL*, *OpenSolaris* e *Open Storage*. A idéia da *Sun* é oferecer nuvens públicas, destinadas a desenvolvedores, estudantes e novas empresas. Os serviços oferecidos pela *Sun Cloud* serão *Sun Cloud Storage Service* e *Sun Cloud Compute Service*.

Graças a várias tecnologias, o modo de utilização da *Sun Cloud* é considerado facilitado - desde a implementação de aplicações ao aprovisionamento de recursos. Na base do *Sun Cloud Compute Service* encontram-se as capacidades de *Virtual Data Center* (VDC) obtidas quando a *Sun* adquiriu a *Q-layer*[8]. Estas capacidades oferecem tudo o que um

[8] Empresa da Bélgica especialista em *Cloud Computing*.

indivíduo ou equipe de desenvolvedores necessita para construir e operar um centro de dados numa nuvem. VDC proporciona uma interface unificada e integrada para disponibilizar uma aplicação a ser executada em qualquer sistema operacional, incluindo *OpenSolaris*, *Linux* e *Windows*.

O *Sun Cloud Storage Service* suporta protocolos *WebDAV* para maior facilidade de acesso aos arquivos e *Object Store* compatíveis com a API *S3* da *Amazon*.

3.2.4. *OnLive*

OnLive [ONLIVE, 2009] é uma plataforma de vídeo sob demanda de jogos, onde são sincronizados, renderizados e armazenados em um servidor remoto e então entregue via coexão com a Internet. A plataforma foi anunciada como sendo compatível com qualquer computador utilizando *Windows XP*, *Vista* ou *Mac OS*.

Um problema já surge, onde a banda de Internet do usuário será um limitador da qualidade de resolução dos jogos. Por exemplo, uma conexão de 1.5 Mbps permitirá imagens com qualidade equivalente ao *Nintendo Wii*, enquanto que para obter uma resolução de alta definição, será necessário uma conexão de 4 a 5 Mbps.

A *OnLive* disponibilizará o "MicroConsole", que poderá ser conectado em um televisor e diretamente com os servidores, para que usuários sem computador também possam utilizar o serviço.

3.2.5. *Windows Azure*

O *Windows Azure* [AZURE, 2009] é um sistema operacional da *Microsoft* que funciona como uma plataforma de serviços para desenvolvedores, porém não está disponível para compra em lojas ou *sites*, apenas seus serviços estão sendo comercializados. Esta plataforma será melhor explicada no próximo capítulo, e será utilizada no desenvolvimento de uma aplicação utilizando os conceitos de *Cloud Computing*.

Os serviços do *Azure* atualmente em funcionamento são três, porém já estão em desenvolvimento novos serviços, como *Microsoft SharePoint* e *Microsoft Dynamics CRM*. A seguir uma breve descrição dos serviços em funcionamento:

- *.NET Services*: oferece serviços distribuídos de infraestrutura para aplicativos na nuvem ou locais. Oferece as principais funcionalidades para criação de um *Service Bus* (barramento de serviços) na nuvem, que pode ser utilizado em aplicações desde simples cenários até os mais complexos;

- *Live Services*: possibilita aos desenvolvedores construírem aplicações baseadas em redes sociais, utilizando o *Windows Live*, que possui mais de 450 milhões de usuários. Disponibiliza os serviços *Client Authentication*, *Delegated Authentication* e *Web Authentication*, cada um com suas particularidades, permitindo que usuários se conectem a áreas restritas das aplicações desenvolvidas;

- *SQL Services*: serviço de armazenamento de dados e processamento de consultas *TRANSACT-SQL*, construído baseado nas tecnologias do *SQL Server* e *Windows Server*. Provê serviços com altos níveis de disponibilidade, escalabilidade e segurança. Auxilia no provisionamento e implantação de bancos de dados múltiplos. Os desenvolvedores não necessitam instalar, configurar ou gerenciar qualquer *software*.

3.3. Obstáculos e possíveis soluções na criação de aplicações em *Cloud Computing*

A seguir são apresentados alguns obstáculos que vem crescendo proporcionalmente com o uso desse paradigma, e também algumas oportunidades para superá-los. Dentre estes obstáculos, existem alguns acerca da adoção de aplicações em *Cloud Computing*, outros em relação ao crescimento destas aplicações, após ser adotado o uso das mesmas, e ainda alguns no que se referem às políticas de negócios. São eles [ARMBRUST, 2009]:

- Disponibilidade de serviço: empresas se preocupam quando precisam utilizar um serviço através da Internet, principalmente a respeito da disponibilidade deste serviço. Ironicamente, os principais produtos criados sob este paradigma possuem altos padrões em relação a esta consideração. Então, usuários já acostumados com serviços como o *Google*, que está disponível 99,99% do tempo [GOOGLE, 2009], já esperam uma disponibilidade semelhante de novos serviços, o que é muito difícil. Também em relação à disponibilidade, pode-se citar o ataque de negação de serviços, podendo prejudicar o funcionamentos dos servidores. Para evitar situações em que um serviço fique indisponível, uma solução plausível seria a existência de uma grande quantidade de servidores. Assim, mesmo com falhas em alguns pontos, o servidor continuará funcionando;

- Aprisionamento de dados: os softwares de hoje em dia tem melhorado em relação a interoperabilidade entre diferentes plataformas, porém as API para *Cloud Computing* ainda são essencialmente proprietárias, e ainda não estão sujeitas e uma padronização. Assim, usuários não podem migrar de uma nuvem para outra com facilidade. A preocupação acerca disso está fazendo com que algumas empresas não adotem *Cloud Computing*. Provedores destes serviços que mantém usuários "presos" a eles podem achar isto interessante, porém usuários ficarão vulneráveis ao aumento de taxas de utilização, problemas de segurança e até mesmo à possível falência do servidor, perdendo assim todos seus dados. A solução óbvia, porém não tão fácil, seria padronizar as API, assim os desenvolvedores poderiam disponibilizar seus serviços em vários provedores, fazendo com que qualquer falha em uma empresa não afete o serviço como um todo, pois outras instâncias dele estarão em pleno funcionamento;

- Confidencialidade dos dados: as nuvens atuais são essencialmente redes públicas, expondo o sistema a mais ataques. Porém isso não é um novo problema, já que não há diferença em relação à segurança em servidores comuns e servidor em *Cloud Computing*. Desta forma, podem-se usar os mesmo dispositivos de segurança já existentes, como criptografia dos dados, filtros de pacotes, *firewalls*[9] entre outros. Criptografar dados antes de serem postados na nuvem é uma prática aceitável;

- Gargalo na transferência de dados: aplicações realizam cada vez mais trocas de informações entre si. Levando em consideração aquelas aplicações que estão separadas por *modems* e roteadores através da Internet, a locação e o transporte de dados poderão ser bastante complicados. Usuários e provedores devem pensar nas implicações de transportar e locar dados em todos os níveis do sistema se desejam minimizar custos. Uma possível solução seria enviar fisicamente os discos de armazenamento, através de serviços de transportadoras. Porém não há garantias dos fabricantes dos discos que permite o transporte de dados desta maneira. Segundo ARMBRUST (2009), em 400 (quatrocentas) tentativas apenas 1 (uma) falhou, porém isto pode ser resolvido enviando discos extras com dados duplicados;

- Desempenho imprevisível: ao utilizar *Cloud Computing*, várias maquinas virtuais compartilham processadores e memórias de forma correta, porém compartilhamento

[9] Dispositivo de rede que aplica políticas de segurança a um determinado ponto de controle da rede.

de entrada e saída é mais problemático. Melhorar arquiteturas e sistemas operacionais para eficientemente virtualizar interrupções seria uma solução. Um exemplo de sucesso em uma situação parecida foi realizada pela IBM, que nos anos 80 superou esses problemas através dos seus *mainframes* e sistemas operacionais. Outra possibilidade é o uso de memória *flash*, que sustenta muito mais trocas de informações por segundo do que os discos rígidos tradicionais;

- Armazenamento escalável: foi mostrado anteriormente que *Cloud Computing* possui como atrativo algumas importantes características, e uma das principais é a capacidade infinita de armazenamento sob demanda. Enquanto isto possa parecer trivial aplicado ao poder de processamento e memória, não é tão óbvio quando se refere a armazenamento persistente. Para superar este obstáculo, pode-se criar um sistema de armazenamento que, além de resolver este problema, seja capaz de alterar arbitrariamente a escala do sistema sob demanda, aumentando ou diminuindo-a. Desta forma é possível alcançar as expectativas dos programadores em relação ao gerenciamento de recursos para escalabilidade, durabilidade dos dados e alta disponibilidade;

- Problemas em sistemas distribuídos de larga escala: um dos desafios mais difíceis no *Cloud Computing* é a tentativa de remover erros nos sistemas distribuídos de larga escala. Uma ocorrência comum são os erros não detectáveis em configurações de pequena escala, então a depuração deve ocorrer em situações compatíveis com cenários reais, ou seja, em larga escala. Uma possível solução pode ser a utilização de máquinas virtuais simulando estes cenários. Desta forma, com o nível de virtualização proporcionado por elas, é possível capturar valiosas informações sobre o sistema;

- Rápida escalabilidade: algumas empresas cobram seus serviços de acordo com os bytes transferidos. Então se o sistema for pouco usado em um determinado mês, a taxa paga será baixa. Porém algumas cobram pelo número de horas que o sistema permanece *on-line*, mesmo sem transferência de dados. Para resolver esta situação, poderá ser utilizado um esquema automático de ampliar ou reduzir o sistema, podendo assim economizar dinheiro e recursos em momentos de baixa utilização da aplicação. Este tipo de escalabilidade já é aplicado à maioria das plataformas de *Cloud Computing* existentes, e sempre destacam como uma das principais vantagens de utilizar este modelo.

4. *Windows Azure*

Em um alto nível de abstração, pode-se dizer que *Windows Azure* é uma plataforma para execução e armazenamento de dados na nuvem. É executada em um grande número de servidores, locados em *datacenters* da *Microsoft* e acessíveis via Internet.

Para avaliação da primeira versão lançada no mercado em janeiro de 2009, a Microsoft disponibilizou a plataforma em um projeto chamado *Community Technology Preview*, e permitiu a execução somente de aplicações desenvolvidas em *.NET Framework,* porém com planos de abrir a plataforma alguns meses depois.

Na atual versão de avaliação, disponível até dezembro de 2009, o software desenvolvido pode ser baseado em linguagens como *ASP.NET* e *C#*, e ainda utilizar ferramentas tradicionais de desenvolvimento, como *Visual Studio 2008*. Tanto aplicações *off-premise* quanto aplicações *on-premise* podem acessar os serviços do *Azure*, utilizando protocolos de comunicação, como *REST* e *SOAP*.

No entanto, ao disponibilizar serviços na nuvem, é necessário um gerenciamento bastante efetivo. No *Azure*, cada aplicação possui um arquivo de configuração. Ao alterar informações neste arquivo, seja manualmente ou automaticamente, o desenvolver consegue controlar vários aspectos do comportamento da aplicação, assim como escolher o número de instâncias que devem ser executadas naquele instante. A plataforma monitora a aplicação para mantê-la no estado desejado.

Um portal acessível via *web* é disponibilizado para que o cliente possa criar, gerenciar e monitorar suas aplicações. O cliente deve se cadastrar e possuir uma *Windows Live ID*. Na versão de avaliação, é necessário esperar que o sistema lhe envie uma chave de entrada, que pode demorar até duas semanas para chegar. Porém na versão oficial de lançamento, não será necessário, e o uso da plataforma será taxado.

A seguir alguns possíveis cenários para o uso da plataforma [CHAPPEL, 2008]:

- Uma nova empresa criando um serviço, como por exemplo, um novo *Orkut*. A plataforma suporta serviços *web* e também a execução de processos em *background* de forma assíncrona. Outro motivo seria não precisar fazer um grande investimento em infraestrutura, fazendo com que a nova empresa foque seus esforços no serviço a ser

desenvolvido. Quando for necessário aumentar a estrutura e o número de usuários ativos no serviço, o espaço em disco e número de instâncias pode ser aumentado através de m arquivo de configuração, que será mostrado no capítulo 4;

- Uma empresa que possui softwares em linguagem *.NET*, pode transferi-los para a nuvem sem maiores problemas, pois *Azure* já é compatível com as aplicações *on-premise* desenvolvidas em *.NET*;

- Uma loja virtual poderá utilizar os serviços do *Windows Azure*. Em épocas de grande número de acessos, por exemplo o Natal, os servidores precisam suportar a alta demanda acessos, e no *Azure* basta solicitar esta alteração, sem precisar se preocupar com gastos em infraestrutura e funcionários especializados.

Além de recursos nativos do *Azure*, esta plataforma oferece ainda uma gama de serviços que capacitam as aplicações com bancos de dados, barramentos de serviços, mecanismos para controle de acesso, suporte ao modelo de *SaaS*, entre outros. [CAMBIUCCI, 2009]

4.1. Serviços disponíveis no *Azure*

Os serviços da plataforma podem ser utilizados por aplicações locais rodando em uma variedade de sistemas, incluindo versões do *Windows*, dispositivos móveis e outros. A seguir são descritos os serviços disponíveis.

4.1.1. *.NET Services*

De acordo com Chappel (2008), executar aplicativos na nuvem é um aspecto importante, porém está distante de ser o principal. Outro aspecto importante é a possibilidade de prover serviços que podem ser utilizados tanto por aplicações em *Cloud Computing* quanto por aplicações *on-premise*. Promover esta integração é o principal objetivo deste serviço, que está dividido em três componentes:

- Controle de Acesso *(Access Control)***:** auxilia na construção de aplicações e serviços seguindo padrões de segurança, de forma que a implementação seja facilitada. Com suporte a um simples modelo de regras e acordos, as regras do *Access Control* podem ser facilmente configuradas para cobrir uma variedade de necessidades relacionadas à

segurança. Um dos principais benefícios é seu modelo de programação simples, baseado no *Framework .NET* e *Windows Communication Foundation (WCF);*

- Barramento de Serviço *(Service Bus)***:** provê conectividade segura entre serviços ou aplicações, e lhes permite navegar através de *firewalls* e redes utilizando uma variedade de padrões de comunicação. Serviços registrados no *Service Bus* podem ser facilmente descobertos e acessados, através de qualquer topologia de rede. Também tem a função de ajudar no bloqueio de tráfego malicioso e proteger os serviços de invasões e de ataques de negação de serviço.

Ainda de acordo com Chappel (2008), um exemplo de utilização do *.NET Services seria* uma empresa que desenvolve uma aplicação utilizada por clientes em diferentes organizações, pode utilizar o componente *Access Control* para simplificar o desenvolvimento. Por exemplo, este componente é capaz de traduzir todas as regras de acesso existentes nas diferentes organizações, cada uma das quais com suas próprias regras internas de acesso, em um conjunto consistente de regras que poderá ser utilizado por todas estas organizações.

4.1.2. *SQL Services*

Chappel (2008) afirma que uma das principais motivações para utilizar servidores acessíveis pela Internet é a possibilidade de armazenar dados nestes servidores, provendo um núcleo de banco de dados. Porém o objetivo do *SQL Services* é prover um conjunto de serviços para a nuvem para armazenamento de variados tipos de dados, desde os não estruturados até os relacionais. Este serviço incrementa várias facilidades em serviços orientados a dados, como por exemplo, relatórios e estatísticas.

Este componente disponibiliza um banco de dados na nuvem. Esta tecnologia permite que aplicativos tradicionais e para a nuvem armazenem e acessem dados nos servidores da *Microsoft*. Assim como em outras tecnologias para *Cloud Computing*, o usuário só paga pelo que usa efetivamente, aumentando ou diminuindo os gastos de acordo com a necessidade atual.

Seu principal objetivo é ser amplamente acessível, e por isto é acessível através de *SOAP* e *REST*, permitindo que dados sejam acessados em diferentes maneiras. Logo este serviço pode ser utilizado por aplicações baseadas em qualquer sistema operacional, não somente *Windows*.

Diferente do serviço de armazenamento do *Azure*, este serviço é baseado no *Microsoft SQL Server*. No entanto, o serviço não exibe uma interface relacional tradicional. Ao invés disto provê um modelo de dados hierárquico que não requer um esquema pré-definido. Cada item de dado armazenado é mantido como uma propriedade com o próprio nome, tipo e valor. Para realizar consultas, uma aplicação pode utilizar acesso através de *REST*, ou utilizar uma linguagem baseada na sintaxe *C#*, definida por *LINQ* (*Microsoft's Language Integrated Query)*.

A versão desenvolvida especificamente para a nuvem oferece algumas vantagens em relação a versão tradicional do *SQL Server*, como melhor escalabilidade, disponibilidade e confiabilidade. A maneira que dados são organizados e retornados faz com que a replicação dos dados seja facilitada e agilizada, pois o modelo de replicação é tratado internamente pelo *Azure*.

A utilização deste serviço é indicada nos seguintes cenários:

- Uma aplicação deseja arquivar dados antigos que não são muito acessados, porém precisam estar disponíveis. Ao utilizar o *SQL Data Services* o cliente terá uma boa alternativa com baixo custo e confiável;

- Supondo um fabricante que precise disponibilizar informações sobre seus produtos para clientes e distribuidores. Ao utilizar este serviço, é possível que essas informações sejam acessadas por aplicativos executados nos distribuidores e também por aplicativos Web voltados para o cliente. Estes aplicativos podem ser escritos com qualquer tecnologia e executados em qualquer plataforma, já que estes dados podem ser acessados utilizando protocolos *REST* e *SOAP*.

Seja para arquivar dados, ou para armazenar dados acessíveis por qualquer locação, um banco de dados na nuvem pode ser uma boa idéia. A *Microsoft* também anunciou que melhorias significativas neste serviço estão para serem lançadas, inclusive em relação a aumentar a compatibilidade com outros tipos de bancos de dados.

4.2. Padrões de aplicações

Existe um número de arquiteturas e padrões de *design* que ajudam o desenvolvedor a escolher uma *PaaS*. Em geral, esses padrões se enquadram em quatro categorias: computação,

armazenamento, comunicação e gerenciamento [JOSEPH, 2009], conforme apresentado na tabela 4.1.

Tabela 4.1 – Padrões de aplicações

Computação	Assim que o tipo de aplicação a ser desenvolvida está definido, o padrão de computação deve ser escolhido. Ao construir uma aplicação interativa, deverá ser utilizado o *Web role*, enquanto que para aplicações em *background* deverá ser utilizado o *Worker role*. Existem casos em que ambos recursos serão utilizados.
Armazenamento	O serviço de armazenamento provê armazenamento remoto e abstrai o *middleware* aos usuários. O *design* é suficientemente flexível para suportar vários tipos de aplicações. *Azure* utiliza dois diferentes padrões de armazenamento**:** • **Table storage:** permite que aplicações armazenem pares chave/valor de uma estrutura. • **Blob storage:** permite armazenamento de qualquer tipo de dados.
Comunicação	Viabiliza a troca de mensagens. A tecnologia da plataforma aproveita a *WCF* e protocolos como *REST* para realizar a comunicação no serviço.
Gerenciamento	Distingue dois aspectos principais do gerenciamento de serviços: desenvolvimento e gerenciamento. • **Desenvolvimento:** definição, configuração e monitoramento do serviço. • **Gerenciamento:** manutenção operacional regular.

Nem todas as aplicações são adequadas para serem executadas em uma plataforma em *Cloud Computing*. Entre fatores limitantes estão segurança e aprisionamento de dados, eficiência ao mover dados de uma nuvem para outra, integração com serviços já existentes e preocupações legais e de privacidade.

4.3. Taxas para utilização

Durante a fase de avaliação, a *Microsoft* liberou o uso da plataforma sem cobranças de taxas [PRICING, 2009], porém os serviços possuem certos limites. Armazenamento de 50GB, transferência de dados de 20GB por dia, e ainda limite de uso de 2000 horas de computação através de máquinas virtuais.

Com o lançamento oficial da plataforma, previsto para janeiro de 2010, a cobrança será feita através de pacotes baseados nos modelos de consumo apresentados na tabela 4.2, e a unidade monetária utilizada é o dólar.

Tabela 4.2 – Taxas e medição de consumo do *Windows Azure* [PRICING, 2009].

	Taxas	**Medição do consumo**
Windows Azure	• Computação**:** $0,12 / hora • Armazenamento**:** $0,15/GB armaz./mês • Transferência**:** $0,10 entrada / $0,15 saída /GB	• Computação**:** medido em horas de uso de máquina. A plataforma só cobra pelas horas em que a aplicação está em estado *deploy*, ou seja, implementada. Enquanto desenvolve e testa a aplicação, o cliente não precisa deixá-la em *deploy*, não precisando pagar pelo uso de máquina. • Armazenamento**:** cliente para pela média diária de dados armazenados durante o mês. • Transferência**:** transmissão de dados recebidos e enviados pela plataforma. Transmissões internas a plataforma não serão cobradas.
.NET Services	• Mensagens**:** $0,15/100.000 operações de mensagens, incluindo mensagens do *Service Bus* e do *Access Control*. • Transferência**:** $0,10 entrada / $0,15 saída /GB	Cliente paga somente pelo número de mensagens operacionais que sua aplicação utilize. As mensagens são taxadas em blocos discretos de 100.000 a cada mês utilizado. Por exemplo: • Um cliente que consumiu 95.000 mensagens pagará o equivalente a 100.000 mensagens, ou seja $0,15 (mais a transferência de dados utilizada). • Um cliente que consumiu 150.000 mensagens pagará o equivalente a 200.000 mensagens, ou seja, $0,30 (mais a transferência de dados utilizada).
SQL services	• Edição para *web* (*Web Edition*): banco de dados relacional com até 1GB de tamanho, $9,99/mês • Edição para Negócios (*Business Edition*): banco de dados relacional com até 10GB de tamanho, $99,99/mês • Transferência**:** $0,10 entrada / $0,15 saída /GB	• Edição para *web*: estão incluídos até 1GB de BD relacionais, BD auto-gerenciáveis e alta disponibilidade automática. • Edição para Negócios: estão incluídos até 10GB de BD relacionais, BD auto-gerenciáveis e alta disponibilidade automática. Também estão em desenvolvimento novos recursos, como auto-particionamento e uso de *CLR*[10], aplicação que providencia a execução das aplicações escritas para .NET.

[10] Common Language Runtime.

5. Implementação de uma aplicação em *Cloud Computing* utilizando *Windows Azure*

Neste capítulo é descrita a implementação de uma aplicação baseada em alguns conceitos de *Cloud Computing*. Para isso é necessário a utilização da plataforma *Windows Azure*, apresentada anteriormente.

5.1. Visão Geral

A aplicação desenvolvido durante este estudo foi um Gerenciador de Tarefas (GDT), onde os usuários que possuem uma *Windows Live ID* podem acessar e cadastrar tarefas a serem realizadas. A linguagem utilizada durante a implementação foi *ASP.NET*. Todos os arquivos utilizados, assim como os dados utilizados estão armazenados em *datacenters* da *Microsoft*.

O GDT está basicamente divida em três módulos, para seguir a linha de serviços do Azure. São eles:

- Cadastro e gerenciamento de tarefas em banco de dados: utilizando o serviço *SQL Azure* para armazenar os dados na nuvem da *Microsoft*;
- Controle de cadastro e autenticação de usuários: utilizando o serviço *Live Services*, que após autenticar um usuário utilizando uma *Windows Live ID*, redireciona-o a uma página restrita;
- Envio automático de e-mails: utilizando um serviço que funciona em *background* para enviar relatórios e estatísticas de acesso ao administrador do sistema. Este módulo funciona de maneira independente do resto da aplicação, não sendo necessário um usuário interagindo com o sistema para realizar as funções.

GDT foi baseado em uma série de exemplos e kits de treinamentos disponibilizados pela *Microsoft* [CAMBIUCCI, 2009], e ainda dicas e exemplos dos autores Carlos (2009) e Torquato (2009). O *download* do projeto completo pode ser realizado através do endereço http://www.diplomaster.com/gdt.

Para realizar o desenvolvimento, foi utilizado *Windows 7* como sistema operacional, e *Visual Studio 2008 SP1* (VS) juntamente com *Windows Azure Tools* (WAT) como

ferramentas de desenvolvimento. Alguns outros softwares também foram necessários, e todos estão explicitados no Apêndice A. A ferramenta WAT, instalada juntamente ao VS permite que projetos do tipo *Cloud Services* sejam criados. A Figura 5.1 mostra a tela de escolha de tipo de projeto no VS. A seguir os passos para o desenvolvimento desta aplicação.

5.2. Criando um *Windows Azure Cloud Service* no *Visual Studio*

Para criar um projeto para a nuvem clique em *File* > *New* > *Project*. A seguir selecione *Cloud Service* na coluna *Project Types*, e na coluna *Templates* selecione *Windows Azure Cloud Service* (Figura 5.1).

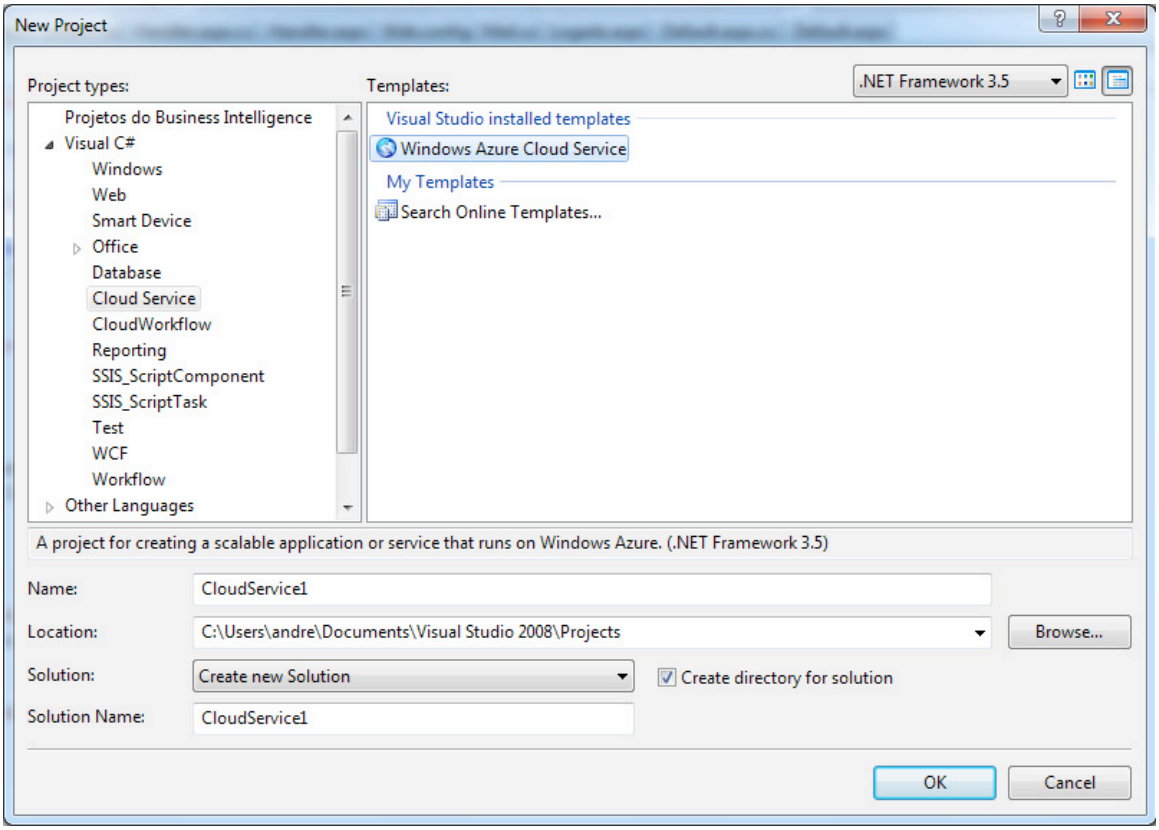

Figura 5.1: Tela de escolha do projeto no VS

A próxima tela apresentada é para a seleção de *roles* (Figura 5.2), que são os tipos de funções que a aplicação possui, disponíveis aos desenvolveres. Segundo NAGY, *web role* é similar a uma aplicação web, e é conduzido através da integração com o usuário, que faz uma requisição no sistema, e recebe uma resposta. Já *worker role* é similar a um serviço do *Windows*, que assim que é inicializado continua sendo executado em *background*, até que atinja o objetivo e pare a execução. No caso desta aplicação foi utilizado *Asp.Net Web Role*.

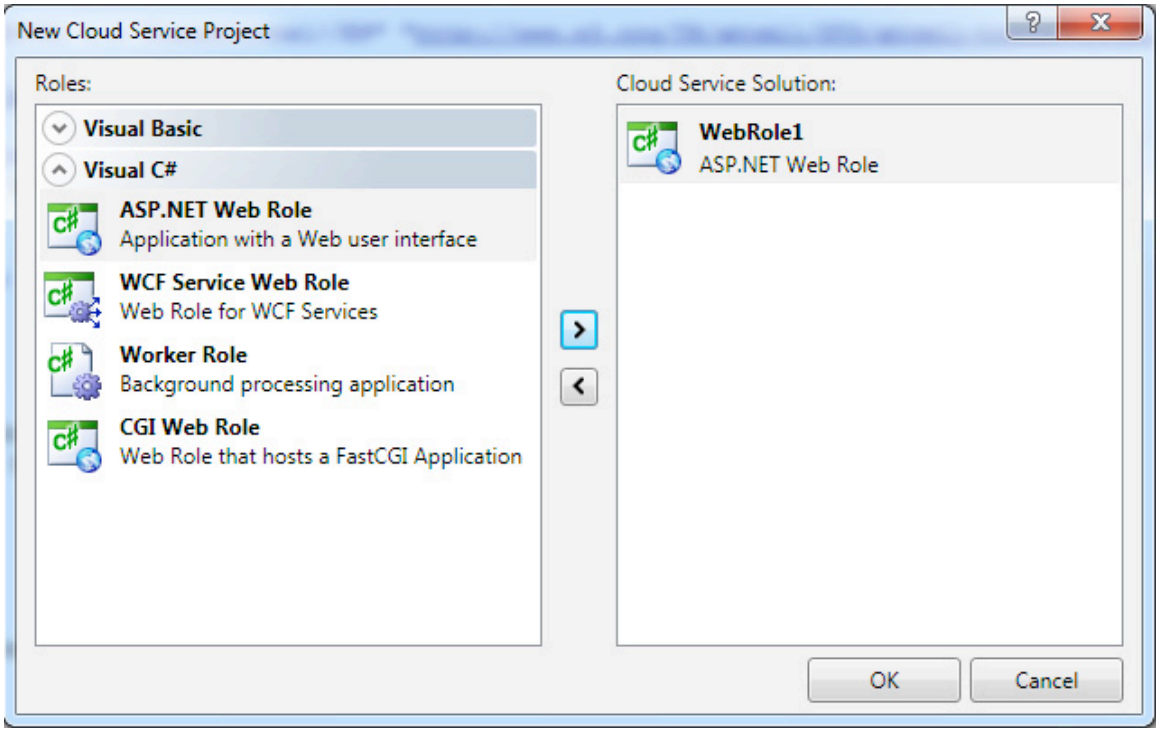

Figura 5.2: Tela de escolha de *roles*

A partir deste ponto o projeto desenvolvido segue os mesmos moldes de um projeto para um software *on-premise*. As principais diferenças estão na publicação e no gerenciamento da aplicação, que são explicados no final deste capítulo.

A seguir os serviços utilizados e a implementação realizada nos módulos da aplicação.

5.2.1. *SQL Azure*

Assim como a maioria das aplicações, a GDT também depende de um banco de dados (BD) para armazenar todas as informações relacionadas ao sistema, inclusive para armazenar informações que podem ser utilizadas na geração de relatórios estatísticos.

Para ter acesso ao *SQL Azure*, o usuário deverá acessar o portal de criação e gerenciamento de bases de dados na nuvem da *Microsoft* (Figura 5.3) através do endereço http://sql.azure.com.

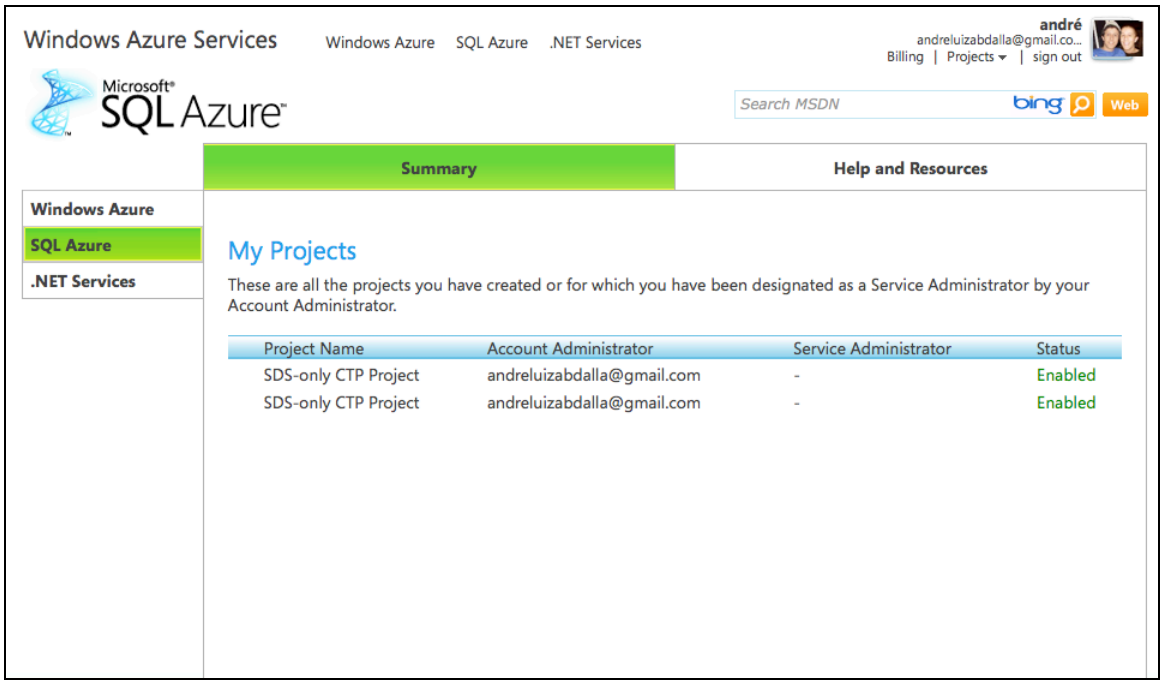

Figura 5.3: Página inicial do serviço SQL Azure

Para criar uma nova base de dados, é necessário preencher os seguintes dados:

- *(Administrator Username*: nome de usuário do administrador do BD;
- *Administrator Password/Retype Password*: senha do administrador do BD;
- *Location*: seleciona um servidor disponível em várias partes do mundo, espalhados pelos *datacenters* da *Microsoft*. No momento do desenvolvimento deste trabalho, apenas uma localização estava disponível, Estados Unidos.

Ao completar o cadastro, o usuário é redirecionado a uma nova página (Figura 5.4), onde encontram-se opções de gerenciamento de dados de conexão e criação de bancos. Lembrando que o nome do servidor de BD será criado automaticamente pelo *Azure*, e é mostrado na próxima página, após a criação desta base.

Um importante recurso disponível neste serviço é o *Firewall*. Este permite que seja especificada uma lista de endereços de IP que podem acessar este servidor de BD. Todos os IP estão bloqueados por padrão, tornando o preenchimento desta lista obrigatória.

Após a criação do BD, é necessário um software de gerenciamento, para a criação de tabelas e outros objetos necessários. Neste estudo foi utilizado o *SQL Management Studio 2008* (SMS).

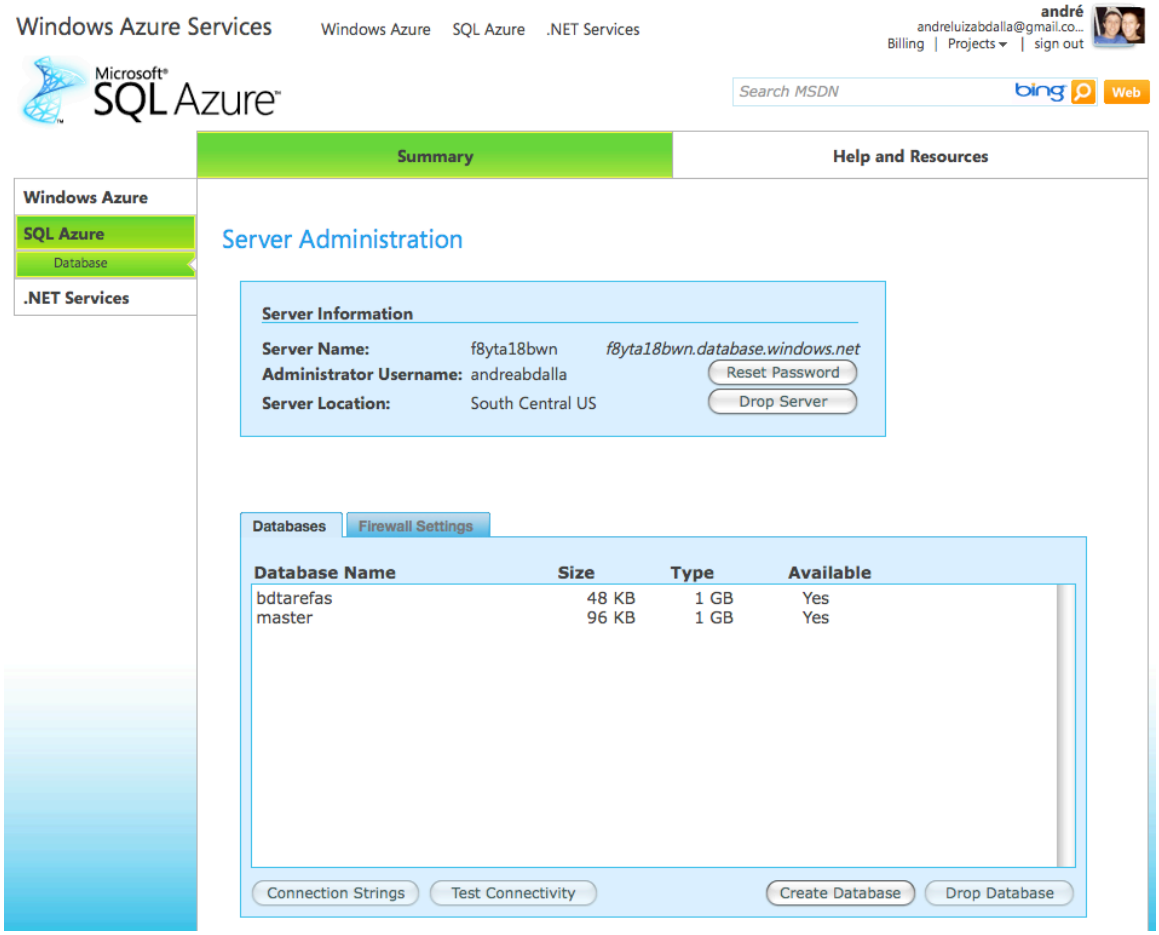

Figura 5.4: Administração do servidor

Ao executar o SMS, uma tela inicial é apresentada. Segundo Carlos (2009) é necessário fechar esta janela, e abrí-la novamente clicando em Nova Consulta, pois esta versão do SMS ainda não está 100% compatível com *Azure*. Nesta janela alguns dados são necessários para a conexão com o BD na nuvem:

- *Server Name*: nome do servidor fornecido pelo *SQL Azure* após o cadastro;
- *Authentication*: selecionar *SQL Server Authentication*, pois está conectando a uma base remota;
- *Login*: equivalente ao *Administrator Username;*
- *Password*: equivalente ao *Admistrator Password.*

Agora clique em *Options*, para editar opções avançadas, na aba *Connection Properties*:

- *Connect to database*: digite o nome da base de dados criada no *SQL Azure*

Após preenchimento correto das informações, a conexão estará estabilizada. A partir deste ponto, o BD pode ser gerenciado da mesma forma que uma base local.

O GDT utiliza as seguintes tabelas de BD:

- Tarefas: armazena as informações relativas às tarefas de cada usuário.
- Estatisticas: armazena as ações realizadas no sistema, que são utilizadas na criação de logs e relatórios.

Segundo Torquato (2009), o *Azure* requer que todas as tabelas do banco tenham pelo menos um índice clusterizado (*clustered*), aumentando o desempenho dos serviços. Caso não insira um índice neste formato, o SMS acusa um erro.

5.2.1.1. Implementação utilizando *SQL Azure*

Para realizar alterações a consultas à base de dados utilizada, foi implementada uma classe chamada *Connection.cs*. Esta contém os dados de acesso fornecidos pelo *Azure*, e encontra-se no Apêndice B.

Para realizar operações na base de dados, foram utilizados métodos contidos na classe *Tarefas.cs*. A classe encontra-se no Apêndice C.

5.2.2. *Live Services*

Neste módulo do GDT, foi utilizado o serviço *Live Services* disponibilizado no *Azure*. Este funciona de modo a autenticar usuários no sistema e permitir o acesso a páginas restritas.

Para realizar a implementação deste serviço, primeiramente o usuário deve registrar-se junto à *Microsoft*, obtendo uma identificação chamada *Application ID*. Para realizar o registro, deve-se acessar o endereço http://live.azure.com (Figura 5.5).

Para criar um novo serviço deste tipo, deverão ser preenchidos os seguintes dados:

- *Service Component Label*: um rótulo único e de fácil entendimento para diferenciar este serviço de outros cadastrados;
- *Service Component Description*: descrição do serviço;

- *Domain*: nome do *site* ou serviço que utilizará *Live Services*. Se estiver utilizando somente *Live Services Web Autorization* este campo pode ser ignorado;

- *Return URL*: endereço para qual o sistema redirecionará o usuário que se autenticou corretamente.

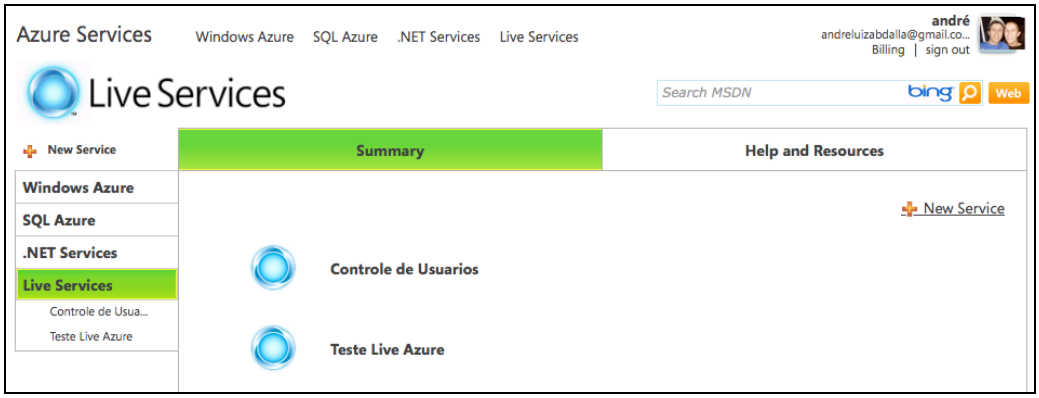

Figura 5.5: Página inicial do *Live Services*

Ao término do cadastro, o usuário então será redirecionado a uma nova página (Figura 5.6), que conterá as seguintes informações:

- *Application ID*: identificador da aplicação. Será utilizado na implementação do serviço;

- *Domains*: endereço do *site* que utilizará o serviço. O uso da *Application ID* estará restrito a este *site*;

- *Return URL*: endereço de retorno cadastrado na criação de um novo serviço;

- *Secret Key*: chave secreta. Será utilizada na implementação do GDT, para permitir acessos de usuários utilizando *Windows Live ID* ao sistema.

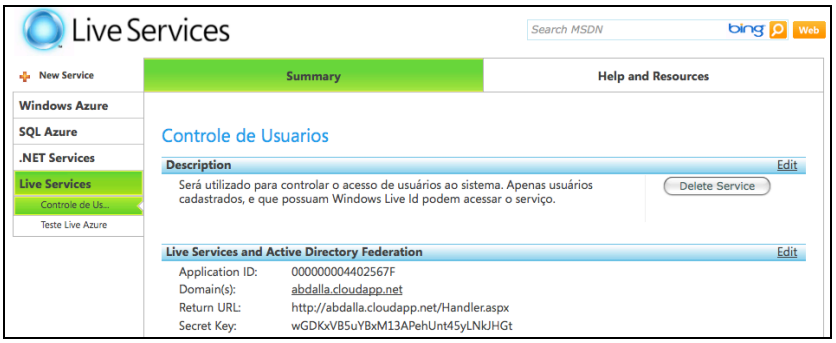

Figura 5.6: Página com informações da aplicação cadastrada

Para inserir o *link* de acesso ao serviço, o seguinte código é utilizado (Figura 5.7) [SIGN-IN, 2009]:

```
<iframe id="WebAuthControl" name="WebAuthControl"
      src="http://login.live.com/controls/WebAuthButton.htm?
appid=<%=AppId%>&context=myContext&style=font-size%3A+10pt%3B+
font-family%3A+verdana%3B+background%3A+white%3B"
    width="80px"
    height="20px"
    marginwidth="0"
    marginheight="0"
    align="middle"
    frameborder="0"
    scrolling="no"
    style="border-style: hidden; border-width: 0"></iframe>
```

Figura 5.7: Código que mostra o *link* de acesso para usuários

No lugar de <%=AppId%> deverá estar a *Application ID* fornecida no cadastro do serviço.

5.2.2.1. Implementação utilizando *Live Services*

Para utilização deste serviço, uma classe chamada *Handler.cs* é implementada. Esta classe controla *login*, *logout* e ainda os *cookies*[11] da aplicação.

Esta página é utilizada como *Return URL* do GDT. Quando acessada, verifica se o parâmetro de retorno *action* está designado e realiza ações de acordo. A seguir as ações de acordo com o valor de *action*:

- *logout*: limpa os *cookies* de *login* e redireciona o usuário para a página de *logout*;
- *clearcookies*: limpa os *cookies* de *login*;
- *login*: por padrão, tenta realizar *login* do usuário. Caso seja bem sucedido, armazena o *token* do usuário em *cookies* e redireciona o usuário para a página restrita. Caso seja mal sucedido, realiza as mesmas ações do *logout*.

[11] Dados trocados entre o navegador e o servidor

5.3. Publicação da aplicação na plataforma *Azure*

Após implementação do projeto, segue-se para a parte de publicação. Ainda no VS, na coluna *Solution Explorer* clicar com o botão direito em cima do projeto criado, e após clicar em *Publish* (Figura 5.8). Inicia-se então o processo de *deploy* no *Azure*.

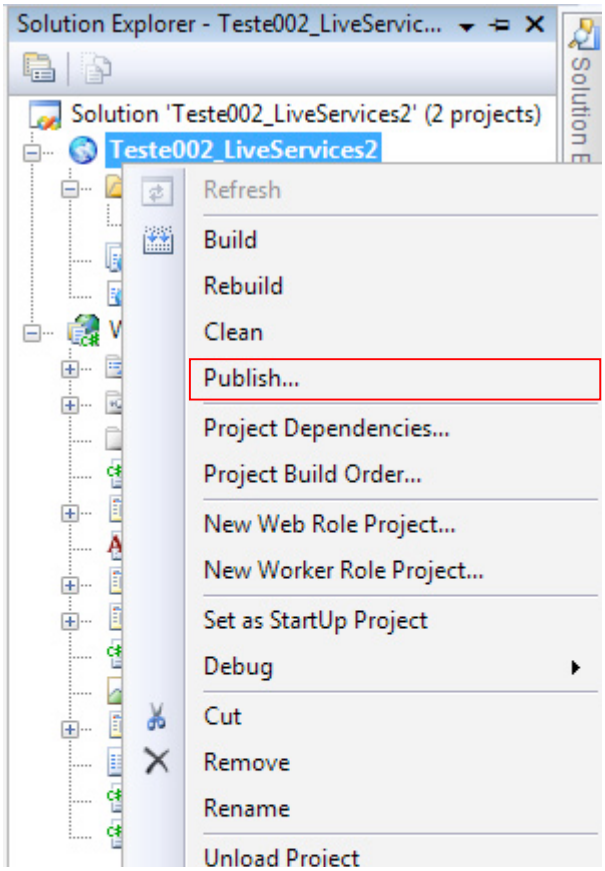

Figura 5.8: Publicação do serviço pelo VS.

Se a publicação ocorrer corretamente, o projeto será compilado, o portal do *Azure* será aberto no navegador e os arquivos compilados serão mostrados no *Windows Explorer*. Estes arquivos são:

- ServiceConfiguration.cscfg[12]: contém configurações do projeto;
- NomeDoProjeto.cspkg[13]: contém todos os arquivos do projeto compilado.

[12] A extensão cscfg significa ser um arquivo do tipo Cloud Service Configuration

[13] A extensão cspkg significa ser um arquivo do tipo Cloud Service Package

No portal *Azure*, a primeira página mostrada é para efetuar *login*, logo após o usuário é redirecionado para a página de seleção de projeto (Figura 5.9).

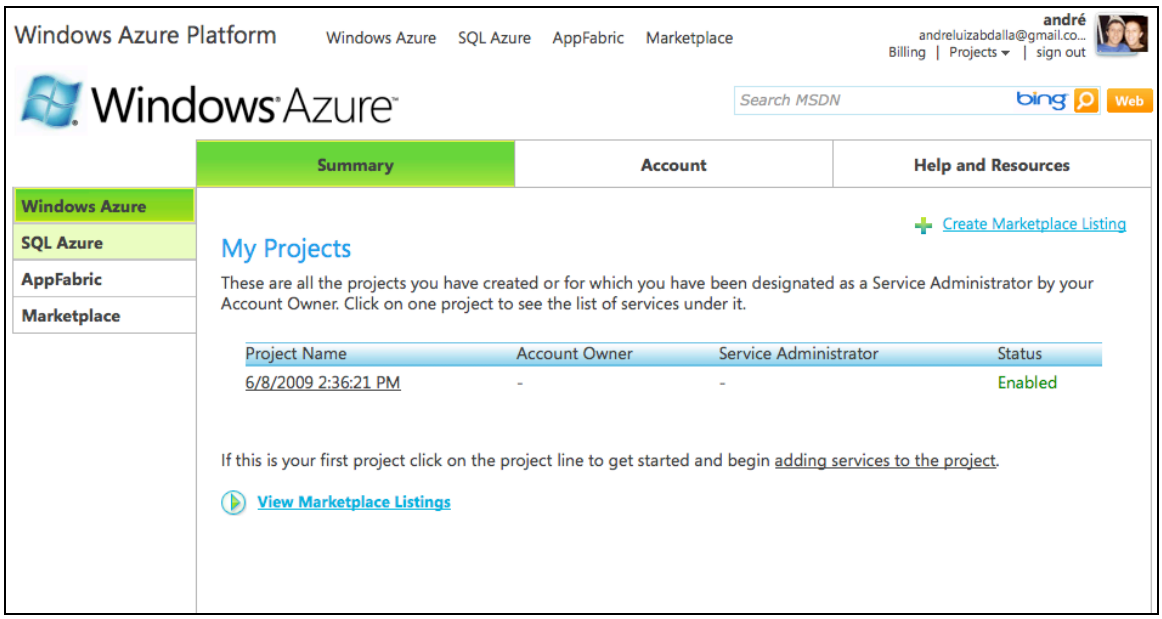

Figura 5.9: Página de seleção do projeto para publicação.

Ao selecionar o projeto, o usuário deve criar ou selecionar entre os já criados o tipo de serviço que está utilizando. Existem duas opções (Figura 5.10):

- Serviços Hospedados *(Hosted Services)*: armazena os arquivos responsáveis pela interface com o usuário;
- Conta de Armazenamento (*Storage Account)*: armazena grandes volumes de dados estruturados ou não-estruturados.

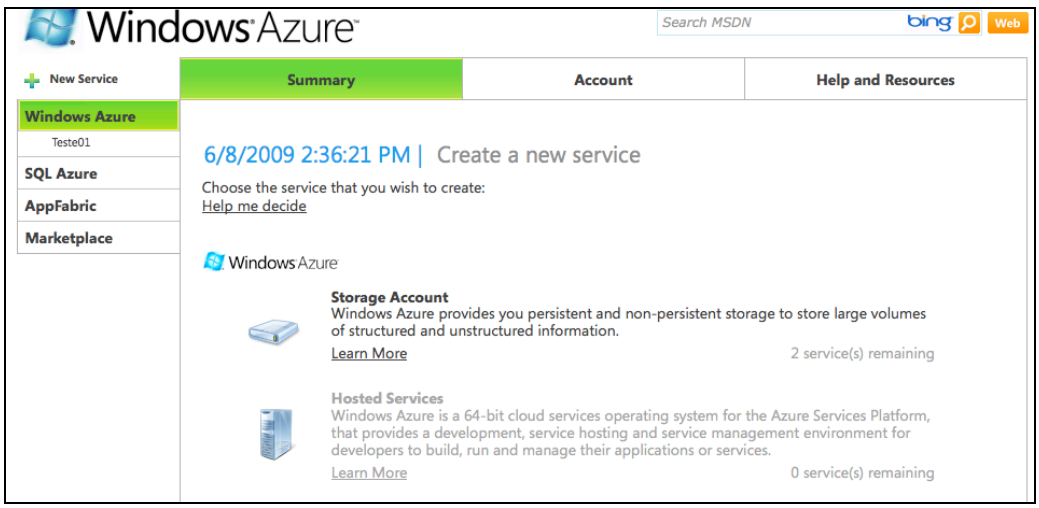

Figura 5.10: Página de seleção do tipo de serviço para publicação.

Ao clicar em *Hosted Services*, são apresentados duas opções em forma de cubos, *Production* e *Staging* (Figura 5.11). Ao utilizar *Production*, o serviço estará funcionando devidamente no endereço da aplicação, e utilizando *Staging*, o serviço estará em um ambiente de testes. Ao clicar em *Deploy* e aguardar a próxima página, o usuário é redirecionado a uma nova página, onde seleciona-se os dois arquivos compilados citados anteriormente.

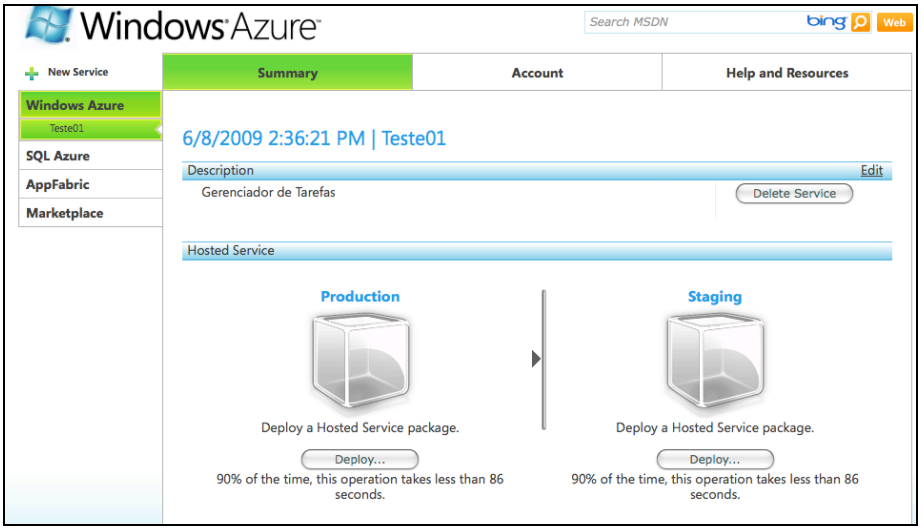

Figura 5.11: Página de *deploy*.

Após a seleção dos arquivos, o usuário é redirecionado à página de *deploy*, agora com algumas diferenças, como alguns novos botões e informações da aplicação enviada (Figura 5.12). O serviço está com *status Stopped*.

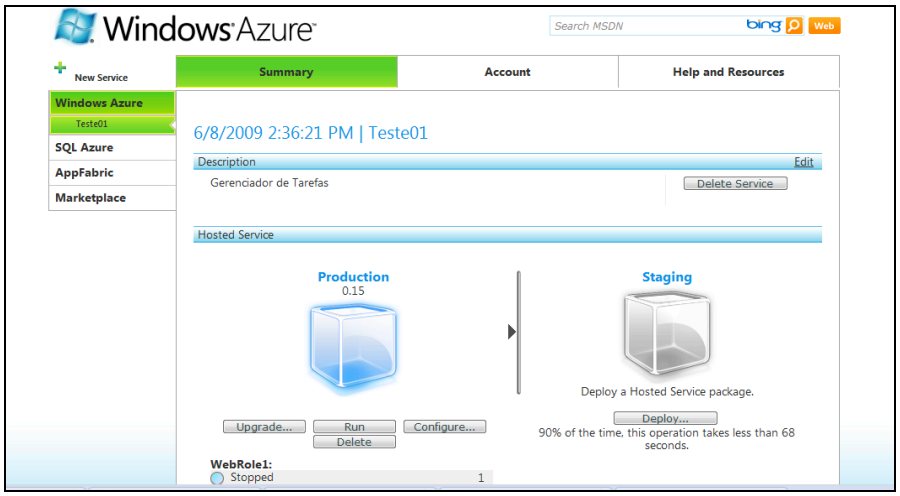

Figura 5.12: Página de *deploy* com *Production* ativo, e *status* como *Stopped*.

Para mudar o *status* do serviço, clica-se em *Run*. Agora o serviço está com *status Ready*, isso quer dizer que está em pleno funcionamento, e pode ser acessado pelo endereço mostrado na Figura 5.13, logo abaixo da informação sobre o *status*.

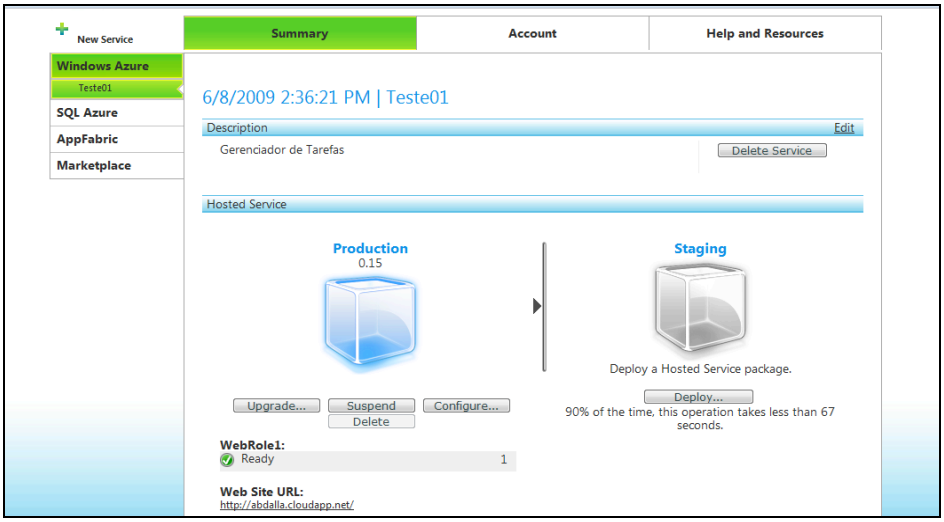

Figura 5.13: Página mostrando o serviço em pleno funcionamento, após ser enviado.

Para atualizar o serviço, basta clica-se no botão *Upgrade*, e seguir os mesmos passos citados anteriormente. Ao clicar em *Suspend*, o serviço é temporariamente suspenso, podendo ser excluído ou retornado ao *status* ativo.

Para configurar o número de instâncias executadas ao mesmo tempo, clica-se no botão *Configure*. Duas opções são apresentadas, em uma delas um novo arquivo de configuração pode ser enviado ao portal, na outra é possível alterar manualmente o arquivo já enviado, através da opção *Instances count*, alterando para o número de instâncias desejadas. Cada instância pode ser considerada uma máquina virtual de *Windows* executando a aplicação desenvolvida.

5.4 Aplicação em execução na plataforma *Azure*

Para acessar a aplicação basta acessar o endereço http://abdalla.cloudapp.net. A Figura 5.14 apresenta a página inicial da aplicação, onde estão os *links* para o usuário realizar *login*, se cadastrar e obter maiores informações sobre o projeto.

Figura 5.14 Página inicial da aplicação em execução

Ao se conectar ao serviço, utilizando o *Windows Live ID*, o usuário é então redirecionado à página de gerenciamento das tarefas (Figura 5.15), onde é possível cadastrar, editar ou excluir uma tarefa.

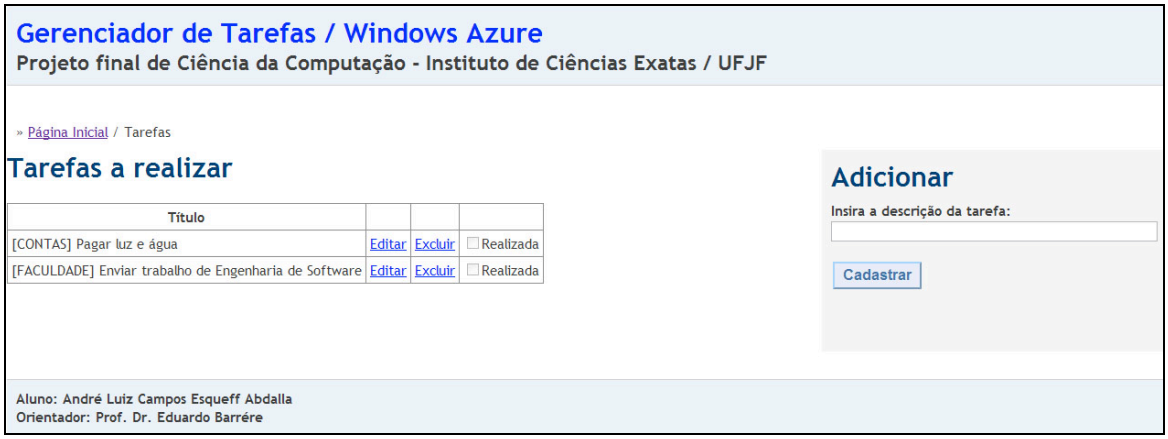

Figura 5.15 Página de gerenciamento das tarefas

Já na seção Contato (Figura 5.16) o usuário pode enviar comentários, críticas ou sugestões sobre a aplicação desenvolvida.

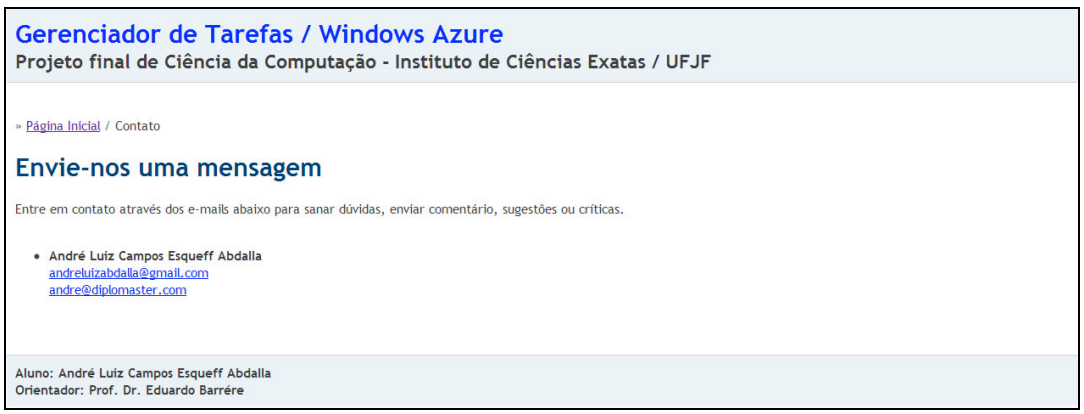

Figura 5.16 Seção Contato

E na seção Saiba Mais (Figura 5.17) o usuário encontra informações mais detalhadas sobre o projeto.

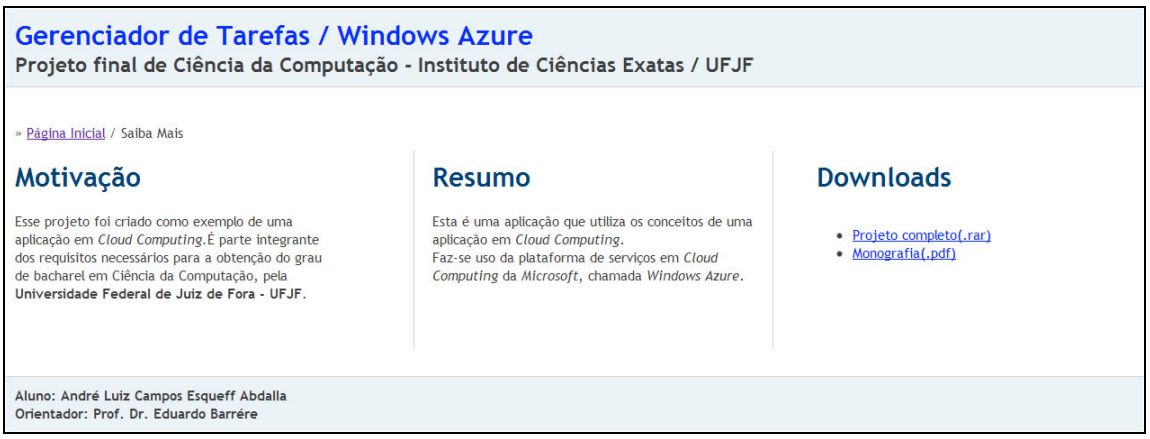

Figura 5.17 Seção Saiba Mais

5.5. Diferenças e semelhanças nos tipos de desenvolvimento

Durante o desenvolvimento da aplicação GDT, foi possível constatar algumas diferenças e semelhanças em relação ao desenvolvimento de aplicações *on-premise*, descritas a seguir.

5.5.1. Implementação

Em relação ao desenvolvimento e programação da aplicação, ambas as categorias de aplicações podem utilizar o mesmo modelo, como se o programador estivesse desenvolvendo uma aplicação web tradicional. As mesmas linguagens, como *.NET*, *PHP* e *JAVA*, e ferramentas, como *Visual Studio*, podem ser utilizadas. Porém serviços disponibilizados pela ferramenta ou plataforma utilizada, como *Windows Azure*, são capazes de facilitar o desenvolvimento ao programador, pois faz uso de classes e bibliotecas pré-definidas próprias da plataforma.

Outro fato importante é em relação à quantidade de material para estudos disponibilizado pela *Microsoft*. Não somente tutoriais e referências sobre as ferramentas, mas detalhes de como a própria plataforma *Azure* é construída, por exemplo, podem ajudar de maneira mais detalhada os desenvolvedores. Existem também os arquitetos e desenvolvedores da *Microsoft*, que estão sempre postando em blogs e sites especializados artigos, kits de treinamentos e *hands-on labs*(que são exemplos ensinando passo-a-passo a utilização das ferramentas).

5.5.2. Gerenciamento

Ao levarmos em conta o gerenciamento da aplicação, as diferenças se destacam. Vale ressaltar que a escalabilidade e a flexibilidade são algumas das principais vantagens deste modelo de computação, foi possível perceber que realmente existem facilidades.

Por exemplo, ao precisar aumentar o número de instâncias sendo executadas, para épocas que exigem maior armazenamento e computação, basta o desenvolvedor alterar um arquivo de configuração que cada aplicação possui. Em outra época, quando o sistema não estiver sendo muito requisitado, o usuário pode diminuir, evitando que recursos fiquem ociosos, e pagando apenas pelo que de fato estiver sendo utilizado.

Ao levar em conta o serviço *SQL Azure*, também em relação ao gerenciamento, o usuário se preocupa apenas com a parte lógica, como criar tabelas, e a *Microsoft* administra a parte física, como discos rígidos, servidores e replicação. Desta forma, *SQL Azure* provê serviços de bancos de dados que oferecem altos níveis de disponibilidade, escalabilidade e segurança.

5.5.3. Custos

A *Microsoft* lançou no próprio *site* da plataforma *Azure*, uma calculadora de custos de utilização para comparação entre soluções *Azure* e *on-premise*. A ferramenta, que pode ser acessada pelo endereço http://www.microsoft.com/windowsazure/tco/, ainda oferece uma série de cenários de possíveis utilizações.

De acordo com Cambiucci (2009), em uma simulação de custos, foi possível verificar que teria gasto de $161.324/ano utilizando um servidor *on-premise*, contra $4.115/ano utilizando a plataforma *Azure*.

6. Considerações finais

Realmente este modelo chega com a filosofia de facilitar a vida de desenvolvedores e empresas. Através do estudo sobre os conceitos da *Cloud Computing* e da plataforma *Windows Azure* foi possível o desenvolvimento de uma aplicação, processando e armazenando-a nos *datacenters* da *Microsoft*. Com isso, foi possível conhecer e utilizar algumas ferramentas e serviços disponibilizados no *Windows Azure*, e ainda estudar padrões, potenciais e limitações da plataforma.

Ainda foi realizada uma comparação em relação ao desenvolvimento de aplicações em *off-premise* e aplicações *on-premise*, indicando principais diferenças e semelhanças.

A utilização desta plataforma é indicada em cenários onde há um grande compartilhamento de dados e arquivos, devido à facilidade de utilizar todo seu potencial através da elasticidade de processamento e armazenamento. Outro cenário favorável seria um aplicativo que necessite de um alto grau de processamento paralelo, para análise de grandes quantidades de dados, por exemplo. O *Azure* pode ser utilizado para a infraestrutura deste aplicativo, podendo alocar os servidores e recursos somente quando realmente precisar.

Porém o *Azure* não é indicado para hospedar, por exemplo, sites em HTML puros, pois o custo poderá ser desvantajoso, comparado a outros servidores de hospedagem. E outro problema detectado durante o desenvolvimento do GDT, é que foi possível perceber alguns momentos em que o portal de entrada da plataforma *Azure* ficava indisponível, mostrando que a ferramenta possui algumas falhas em relação à disponibilidade.

Como trabalhos futuros, seria indicado implementar maiores funcionalidade no GDT, utilizando mais serviços e recursos da plataforma *Azure*. O desenvolvimento de uma *IDE*[14], nos mesmos moldes de algumas existentes, fazendo com que o processamento da aplicação fique todo na nuvem.

[14] Integrated Development Environment

Referências Bibliográficas

ARMBRUST, M. - **Above the Clouds: A Berkeley View of Cloud Computing** – 10 de Fevereiro de 2009

AWS. **What is AWS?** - http://aws.amazon.com/what-is-aws/ - Acessado em 28 de setembro de 2009, às 00:13.

AZURE. **Centro de Desenvolvimento da Plataforma Azure** - http://msdn.microsoft.com/pt-br/azure/default.aspx - Acessado em 28 de setembro de 2009, às 21:45.

BARBOSA, F. P., CHARÃO, A. S. - **Grid Computing e Cloud Computing – Uma Relação de Diferenças, Semelhanças, Aplicabilidade e Possibilidades de Cooperação entre os dois Paradigmas.** Programa de Pós-Graduação em Informática – Universidade Federal de Santa Maria (UFSM). Centro de Tecnologia, Campus Universitário – Camobi – Santa Maria – RS – Brasil - Julho de 2009.

CAMBIUCCI, W. – **Artigos, training kits, samples e demos para o Windows Azure** - http://blogs.msdn.com/wcamb/archive/2009/08/20/artigos-training-kits-samples-e-demos-para-o-windows-azure.aspx , 20 de agosto de 2009.

CAMBIUCCI, W. - **Cloud computing e algumas questões sobre custos de TI** - http://www.profissionaisti.com.br/2009/11/cloud-compuding-e-algumas-questoes-sobre-custos-de-ti/ , acessado em 04 de novembro de 2009, às 21:05 horas.

CAMBIUCCI, W. - **Uma Introdução sobre Cloud Computing e Windows Azure** - 23 de outubro de 2009.

CARLOS, A. – **Criando serviços na nuvem do Windows Azure** - http://www.desenvolvendoparaweb.net/profiles/blogs/criando-servicos-na-nuvem-do , 2 de novembro de 2009.

CHAPPEL, D. - **Introducing the Azure Services Platform:** An early look at Windows Azure, .NET Services, SQL Services and Live Services. Outubro de 2008.

DANTAS, C. – **Windows Live – Introdução ao Windows Live** - http://www.linhadecodigo.com.br/ArtigoImpressao.aspx?id=2035 , 30 de setembro de 2008.

DOS SANTOS, B. C., DE MENESES, F. G. A. - **Cloud Computing: conceitos, oportunidades e desafios da nova computação.** Instituto Federal de Educação Ciência e Tecnologia - Campus Parnaíba. II Encontro Unificado de Computação em Parnaíba-PI, dia 06 de junho de 2009

FOSTER I., YONG ZHAO, RAICU I, LU S. **Cloud Computing and Grid Computing 360-Degree Compared** – Department of Computer Science, University of Chicago – 16 de dezembro de 2008.

GOOGLE. **Indisponibilidade do Google no Brasil é a menor do mundo** - http://www.itweb.com.br/noticias/index.asp?cod=42549 - Acessado em 15 de setembro de

2009, às 15:30.

JOSEPH, J. - **Patterns For High Availability, Scalability, And Computing Power With Windows Azure** - http://msdn.microsoft.com/en-us/magazine/dd727504.aspx, acesso em 6 setembro de 2009, às 10:30.

MILLER, M. – **CLOUD COMPUTING – Web-Based Applications That Change the Way You Work and Collaborate Online**. 2009.

NAGY, S. – **Above the Cloud - Web Role vs Worker Role** - http://azure.snagy.name/blog/?p=26 , acessado em 16 de dezembro de 2009, às 23:12.

NURMI, D., WOLSKI, R., GRZEGORCZYK, C., OBERTELLI, G., SOMAN, S., YOUSEFF, L., ZAGORODNOV, D. - **The Eucalyptus Open-source Cloud-computing System.** Computer Science Department - University of California, Santa Barbara, California – 2009.

ONLIVE. **OnLive: The Future of Video Games** - http://www.online.com - Acessado em 27 de setembro de 2009, às 14:25.

PRICING. **Windows Azure Platform Pricing - Windows Azure Platform** - http://www.microsoft.com/windowsazure/pricing/ , acesso em 06 de novembro de 2009, às 00:30.

PRODUCTS. **.NET Services -- Products -- Windows Azure Platform** - http://www.microsoft.com/windowsazure/dotnetservices/ , acessado em 15 de outubro de 2009, às 15:30

SIGN-IN. **Displaying the sign-in link** - http://msdn.microsoft.com/en-us/library/bb676638.aspx , acessado em 22 de novembro de 2009, às 23:30.

SUN. **Sun Microsystems anuncia plataforma aberta de Cloud Computing** - http://br.sun.com/sunnews/press/2009/20090324.jsp - Acessado em 20 de setembro de 2009, às 15:45.

TORQUATO, T. - **ASP.NET + SQL Azure Database (SAD)** - http://www.infoconsultor.info/?p=507 , 17 de outubro de 2009.

WANG, L., VON LASZEWSKI, G. - **Cloud Computing: a Perspective Study**. Service Oriented Cyberinfrastruture Lab, Rochester Inst. of Tech - Dezembro de 2008.

Windows Azure Platform Developer Center - http://msdn.microsoft.com/en-us/azure/default.aspx , acessado em 21 de outubro de 2009, às 18:20.

Apêndice A

Softwares utilizados no desenvolvimento da aplicação

Foi utilizando o sistema operacional Windows 7, podendo também ser utilizado Windows Vista. Também foi necessário ativar o IIS *(Internet Information Service)* no Windows.

A ferramenta de desenvolvimento utilizada foi Visual Studio 2008 SP1, juntamente com o *add-in Windows Azure Tools for Microsoft Visual Studio*.

Alguns SDK *(Software Development Kit)* foram necessários para a implementação dos serviços da plataforma *Azure*:

- *Windows Azure SDK*
- *Microsoft .NET Services SDK*
- *Microsoft Live SDK*
- *Windows Live ID Web Authentication SDK*

No gerenciamento do *SQL Azure*, foi utilizado *SQL Management Studio 2008*, para a criação de tabelas, realização de consultas entre outros.

Apêndice B

Implementação da classe Connections.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.SqlClient;

namespace WebRole1
{
    public class Connection
    {
        private string userName = "andreabdalla";
        private string password = "**************";
        private string dataSource = "tcp:f8yta18bwn.database.windows.net";
        private string databaseName = "bdtarefas";
        private SqlConnectionStringBuilder connStringBuilder = null;

        public Connection()
        {
            connStringBuilder = new SqlConnectionStringBuilder();
            connStringBuilder.DataSource = dataSource;
            connStringBuilder.InitialCatalog = databaseName;
            connStringBuilder.Encrypt = true;
            connStringBuilder.TrustServerCertificate = true;
            connStringBuilder.UserID = userName;
            connStringBuilder.Password = password;
        }

        public SqlConnection getConnection()
        {
            return new SqlConnection(connStringBuilder.ToString());
        }
    }
}
```

Apêndice C

Implementação da classe Tarefas.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.SqlClient;
using WindowsLive;

namespace WebRole1
{
    public class Tarefas
    {
        const string LoginCookie = "webauthtoken";

        // Initialize the WindowsLiveLogin module.
        static WindowsLiveLogin wll = new WindowsLiveLogin(true);

        protected static string AppId = wll.AppId;
        protected string UserId;
        protected string UserToken;

        protected void Page_Load(object sender, EventArgs e)
        {
                /* If the user token has been cached in a site cookie,
attempt
               to process it and extract the user ID. */

            HttpRequest req = HttpContext.Current.Request;
            HttpCookie loginCookie = req.Cookies[LoginCookie];


            if (loginCookie != null)
            {
                string token = loginCookie.Value;

                if (!string.IsNullOrEmpty(token))
                {
                    WindowsLiveLogin.User user = wll.ProcessToken(token);


                    if (user != null)
                    {
                        UserId = user.Id;
                        UserToken = user.Id;

                    }

                }
            }
        }



        public int CodTarefa { get; set; }
        public string TituloTarefa { get; set; }
```

```
        public bool Realizado { get; set; }
        public string TokenUsuario { get; set; }

        public string Hoje;
        public string Tipo;
        public int Quant;

        public Boolean Enviar()
        {
            Hoje = "2009-11-28";
            Tipo = "CadastroDeTarefa";
            Quant = 1;

            int rowsAdded = -1;
            using (var conn = new Connection().getConnection())
            {
                using (SqlCommand command = conn.CreateCommand())
                {
                    conn.Open();
                    command.CommandText = string.Format("INSERT INTO
tarefas(TituloTarefa,Realizado,TokenUsuario) VALUES('{0}','{1}','{2}');" ,
                                                        TituloTarefa,
Realizado, TokenUsuario );
                    rowsAdded = command.ExecuteNonQuery();

                    command.CommandText = string.Format("INSERT INTO
estatisticas(Data,Tipo,Quantidade) VALUES('{0}','{1}','{2}');",
                                                        Hoje, Tipo, Quant);
                    rowsAdded = command.ExecuteNonQuery();

                }
            }
            if (rowsAdded != -1) { return true; } else { return false; }
        }

        public List<Tarefas> getTarefas()
        {
            using (var conn = new Connection().getConnection())
            {
                using (SqlCommand command = conn.CreateCommand())
                {
                    conn.Open();
                    command.CommandText = string.Format("INSERT INTO
estatisticas (Tipo) VALUES ('{0}') ;", UserToken);
                    command.ExecuteNonQuery();

                    command.CommandText = string.Format("SELECT * FROM
tarefas WHERE TokenUsuario = '{0}' ORDER BY realizado ASC, titulotarefa ASC
;", UserToken);
                    using (SqlDataReader dataReader =
command.ExecuteReader())
                    {
                        List<Tarefas> tarefasList = new List<Tarefas>();
                        while (dataReader.Read())
                        {
                            Tarefas tar = new Tarefas();
                            tar.CodTarefa =
Convert.ToInt32(dataReader["codtarefa"].ToString());
                            tar.TituloTarefa =
dataReader["titulotarefa"].ToString();
```

```
                            tar.Realizado =
Convert.ToBoolean(dataReader["realizado"].ToString());
                            tar.TokenUsuario = UserToken;
                            tarefasList.Add(tar);
                        }
                        return tarefasList.ToList<Tarefas>();
                    }
                }
            }
        }


        public void updateTarefa(int CodTarefa,string TituloTarefa,bool
Realizado)
        {
            using (var conn = new Connection().getConnection())
            {
                using (SqlCommand command = conn.CreateCommand())
                {
                    conn.Open();
                    command.CommandText = string.Format("UPDATE tarefas SET
realizado = '{0}', titulotarefa = '{1}' WHERE CodTarefa =
'{2}'",Realizado,TituloTarefa,CodTarefa);
                    command.ExecuteNonQuery();
                }
            }
        }


        public void deleteTarefa(int CodTarefa)
        {
            using (var conn = new Connection().getConnection())
            {
                using (SqlCommand command = conn.CreateCommand())
                {
                    conn.Open();
                    command.CommandText = string.Format("DELETE FROM
tarefas WHERE CodTarefa = '{0}'", CodTarefa);
                    command.ExecuteNonQuery();
                }
            }
        }



    }
}
```

Apêndice D

Implementação da classe Handler.cs

```
using System;
using System.Web;
using System.IO;
using WindowsLive;
using System.Data.SqlClient;

/// <summary>
/// This page handles the login, logout and clearcookie Web Auth
/// actions.  When you create a Windows Live application, you must
/// specify the URL of this handler page.
/// </summary>

namespace WebRole1
{
    public partial class HandlerPage : System.Web.UI.Page
    {
        const string LoginPage = "Logado.aspx";
        const string LogoutPage = "Default.aspx";
        const string LoginCookie = "webauthtoken";
        static DateTime ExpireCookie = DateTime.Now.AddYears(-10);
        static DateTime PersistCookie = DateTime.Now.AddYears(10);

        public string Hoje;
        public string Tipo;
        public int Quant;

        // Initialize the WindowsLiveLogin module.
        static WindowsLiveLogin wll = new WindowsLiveLogin(true);

        protected void Page_Load(object sender, EventArgs e)
        {
            HttpRequest req = HttpContext.Current.Request;
            HttpResponse res = HttpContext.Current.Response;

            // Extract the 'action' parameter from the request, if any.
            string action = req["action"];

            /*
              If action is 'logout', clear the login cookie and redirect
              to the logout page.

              If action is 'clearcookie', clear the login cookie and
              return a GIF as response to signify success.

              By default, try to process a login. If login was
              successful, cache the user token in a cookie and redirect
              to the site's main page.  If login failed, clear the cookie
              and redirect to the main page.
            */

            if (action == "logout")
            {
                HttpCookie loginCookie = new HttpCookie(LoginCookie);
                loginCookie.Expires = ExpireCookie;
```

```
                res.Cookies.Add(loginCookie);
                res.Redirect(LogoutPage);
                res.End();
            }
            else if (action == "clearcookie")
            {
                HttpCookie loginCookie = new HttpCookie(LoginCookie);
                loginCookie.Expires = ExpireCookie;
                res.Cookies.Add(loginCookie);

                string type;
                byte[] content;
                wll.GetClearCookieResponse(out type, out content);
                res.ContentType = type;
                res.OutputStream.Write(content, 0, content.Length);

                res.End();
            }
            else
            {
                WindowsLiveLogin.User user = wll.ProcessLogin(req.Form);

                HttpCookie loginCookie = new HttpCookie(LoginCookie);
                if (user != null)
                {
                    loginCookie.Value = user.Token;

                    if (user.UsePersistentCookie)
                    {
                        loginCookie.Expires = PersistCookie;
                    }
                }
                else
                {
                    loginCookie.Expires = ExpireCookie;
                }

                res.Cookies.Add(loginCookie);

                Hoje = "2009-12-03";
                Tipo = "LoginDeUsuario";
                Quant = 1;

                //Aqui serve para inserir na tabela estatísticas que um
usuário entrou no sistema
                using (var conn = new Connection().getConnection())
                {
                    using (SqlCommand command = conn.CreateCommand())
                    {
                        conn.Open();
                        command.CommandText = string.Format("INSERT INTO
estatisticas(Data,Tipo,Quantidade) VALUES('{0}','{1}','{2}');",
                                                            Hoje, Tipo,
Quant);
                        command.ExecuteNonQuery();
                    }
                }

                res.Redirect(LoginPage);
                res.End();
}}}}
```

Apêndice E

Implementação da classe Default.cs

```
using System;
using System.Web;
using System.IO;
using WindowsLive;

namespace WebRole1
{
    public partial class DefaultPage : System.Web.UI.Page
    {
        const string LoginCookie = "webauthtoken";

        // Initialize the WindowsLiveLogin module.
        static WindowsLiveLogin wll = new WindowsLiveLogin(true);

        protected static string AppId = wll.AppId;
        protected string UserId;
        protected string UserToken;

        protected void Page_Load(object sender, EventArgs e)
        {
            /* If the user token has been cached in a site cookie, attempt
               to process it and extract the user ID. */

            HttpRequest req = HttpContext.Current.Request;
            HttpCookie loginCookie = req.Cookies[LoginCookie];


            if (loginCookie != null)
            {
                string token = loginCookie.Value;

                if (!string.IsNullOrEmpty(token))
                {
                    WindowsLiveLogin.User user = wll.ProcessToken(token);


                    if (user != null)
                    {
                        UserId = user.Id;
                        UserToken = user.Id;
                    }
                }
            }
        }


        protected void BotaoCadastrar_Click(object sender, EventArgs e)
        {
            Tarefas novaTarefa = new Tarefas();
            novaTarefa.TituloTarefa = TextBoxTitulo.Text;
            novaTarefa.Realizado = false;
            novaTarefa.TokenUsuario = UserToken;

            if (novaTarefa.Enviar())
```

```
            {
                TextBoxTitulo.Text = string.Empty;
                LabelMsg.Visible = true;
                LabelMsg.Text = "Nova tarefa cadastrada com sucesso!";
                GridView.DataBind();
            }
        }
    }
}
```

Apêndice F

Implementação do arquivo de configuração Web.config

```
<?xml version="1.0"?>

<configuration>


  <appSettings>
    <add key="wll_appid" value="000000004402567F"/>
    <add key="wll_secret" value="wGDKxVB5uYBxM13APehUnt45yLNkJHGt"/>
    <add key="wll_securityalgorithm" value="wsignin1.0"/>
  </appSettings>


    <configSections>
      <sectionGroup name="system.web.extensions"
type="System.Web.Configuration.SystemWebExtensionsSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
        <sectionGroup name="scripting"
type="System.Web.Configuration.ScriptingSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
          <section name="scriptResourceHandler"
type="System.Web.Configuration.ScriptingScriptResourceHandlerSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication"/>
          <sectionGroup name="webServices"
type="System.Web.Configuration.ScriptingWebServicesSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
            <section name="jsonSerialization"
type="System.Web.Configuration.ScriptingJsonSerializationSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="Everywhere" />
            <section name="profileService"
type="System.Web.Configuration.ScriptingProfileServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication" />
            <section name="authenticationService"
type="System.Web.Configuration.ScriptingAuthenticationServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication" />
            <section name="roleService"
type="System.Web.Configuration.ScriptingRoleServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="MachineToApplication" />
          </sectionGroup>
        </sectionGroup>
      </sectionGroup>
    </configSections>
```

```
    <system.diagnostics>
        <trace>
            <listeners>
                <add
type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener,
Microsoft.WindowsAzure.Diagnostics, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
                    name="AzureDiagnostics">
                    <filter type="" />
                </add>
            </listeners>
        </trace>
    </system.diagnostics>



  <system.web>
    <customErrors mode="Off"/>
  </system.web>



  <connectionStrings/>

    <system.web>
        <!--
            Set compilation debug="true" to insert debugging
            symbols into the compiled page. Because this
            affects performance, set this value to true only
            during development.
        -->
        <compilation debug="false">

          <assemblies>
            <add assembly="System.Core, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089"/>
            <add assembly="System.Data.DataSetExtensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
            <add assembly="System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
            <add assembly="System.Xml.Linq, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
          </assemblies>

        </compilation>
        <!--
            The <authentication> section enables configuration
            of the security authentication mode used by
            ASP.NET to identify an incoming user.
        -->
        <authentication mode="Windows" />
        <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.
```

```
        <customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm" />
        </customErrors>
        -->

      <pages>
        <controls>
          <add tagPrefix="asp" namespace="System.Web.UI"
assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
          <add tagPrefix="asp" namespace="System.Web.UI.WebControls"
assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
        </controls>
      </pages>

      <httpHandlers>
        <remove verb="*" path="*.asmx"/>
        <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
        <add verb="*" path="*_AppService.axd" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
        <add verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"
validate="false"/>
      </httpHandlers>
      <httpModules>
        <add name="ScriptModule" type="System.Web.Handlers.ScriptModule,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
      </httpModules>

    </system.web>

    <system.codedom>
      <compilers>
        <compiler language="c#;cs;csharp" extension=".cs" warningLevel="4"
                  type="Microsoft.CSharp.CSharpCodeProvider, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
          <providerOption name="CompilerVersion" value="v3.5"/>
          <providerOption name="WarnAsError" value="false"/>
        </compiler>
     </compilers>
    </system.codedom>

    <!--
        The system.webServer section is required for running ASP.NET AJAX
under Internet
        Information Services 7.0.  It is not necessary for previous version
of IIS.
    -->
    <system.webServer>
      <validation validateIntegratedModeConfiguration="false"/>
      <modules>
```

```
        <remove name="ScriptModule" />
        <add name="ScriptModule" preCondition="managedHandler"
type="System.Web.Handlers.ScriptModule, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
      </modules>
      <handlers>
        <remove name="WebServiceHandlerFactory-Integrated"/>
        <remove name="ScriptHandlerFactory" />
        <remove name="ScriptHandlerFactoryAppServices" />
        <remove name="ScriptResource" />
        <add name="ScriptHandlerFactory" verb="*" path="*.asmx"
preCondition="integratedMode"
             type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
        <add name="ScriptHandlerFactoryAppServices" verb="*"
path="*_AppService.axd" preCondition="integratedMode"
             type="System.Web.Script.Services.ScriptHandlerFactory,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
        <add name="ScriptResource" preCondition="integratedMode"
verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
      </handlers>
    </system.webServer>

    <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
        <dependentAssembly>
          <assemblyIdentity name="System.Web.Extensions"
publicKeyToken="31bf3856ad364e35"/>
          <bindingRedirect oldVersion="1.0.0.0-1.1.0.0"
newVersion="3.5.0.0"/>
        </dependentAssembly>
        <dependentAssembly>
          <assemblyIdentity name="System.Web.Extensions.Design"
publicKeyToken="31bf3856ad364e35"/>
          <bindingRedirect oldVersion="1.0.0.0-1.1.0.0"
newVersion="3.5.0.0"/>
        </dependentAssembly>
      </assemblyBinding>
    </runtime>

</configuration>
```