APRESENTANDO A PLATAFORMA DE SERVIÇOS AZURE

UMA BREVE ANÁLISE DO WINDOWS AZURE, .NET SERVICES, SQL SERVICES E LIVE SERVICES

DAVID CHAPPELL

OUTUBRO DE

2008

PATROCINADO PELA MICROSOFT CORPORATION

CONTEÚDO

VISÃO GERAL DA PLATAFORMA DE SERVIÇOS AZURE

Usar computadores na nuvem pode fazer muito sentido. Ao invés de comprar e manter suas próprias máquinas, por que não explorar os servidores acessíveis na Internet oferecidos hoje? Para alguns aplicativos, os códigos e dados podem viver na nuvem, onde outra pessoa gerencia e mantém os sistemas usados. Alternativamente, aplicativos que são executados dentro da organização —aplicativos locais (ou on-premises) — podem armazenar dados na nuvem ou contar com outros serviços de infra-estrutura da nuvem. Aplicativos executados em estações de trabalho e dispositivos móveis podem usar serviços na nuvem para sincronizar informações com muitos sistemas ou de outras formas. Não importa como seja feito, explorar as possibilidades da nuvem pode melhorar nosso mundo.

Mas, se um aplicativo roda na nuvem, usa os serviços fornecidos por ela, ou ambos, algum tipo de plataforma de aplicativos é necessário. De maneira geral, pode-se considerar uma plataforma de aplicativos como algo que forneça serviços acessíveis ao desenvolvedor para a criação de aplicativos. No mundo do Windows local, por exemplo, isso inclui tecnologias como o .NET Framework, o SQL Server e outros softwares e serviços. Para permitir que aplicativos explorem a nuvem, plataformas de aplicação na nuvem também devem existir. E, como existem várias formas dos aplicativos usarem os serviços da nuvem, diferentes tipos de plataformas da nuvem são úteis em situações diferentes.

A Plataforma de Serviços Azure da Microsoft é um grupo de tecnologias da nuvem, que fornece um conjunto específico de serviços para desenvolvedores de aplicativos. Como a Figura 1 mostra, a Plataforma de Serviços Azure pode ser usada tanto por aplicativos em execução na nuvem quanto por aqueles executados em sistemas locais.

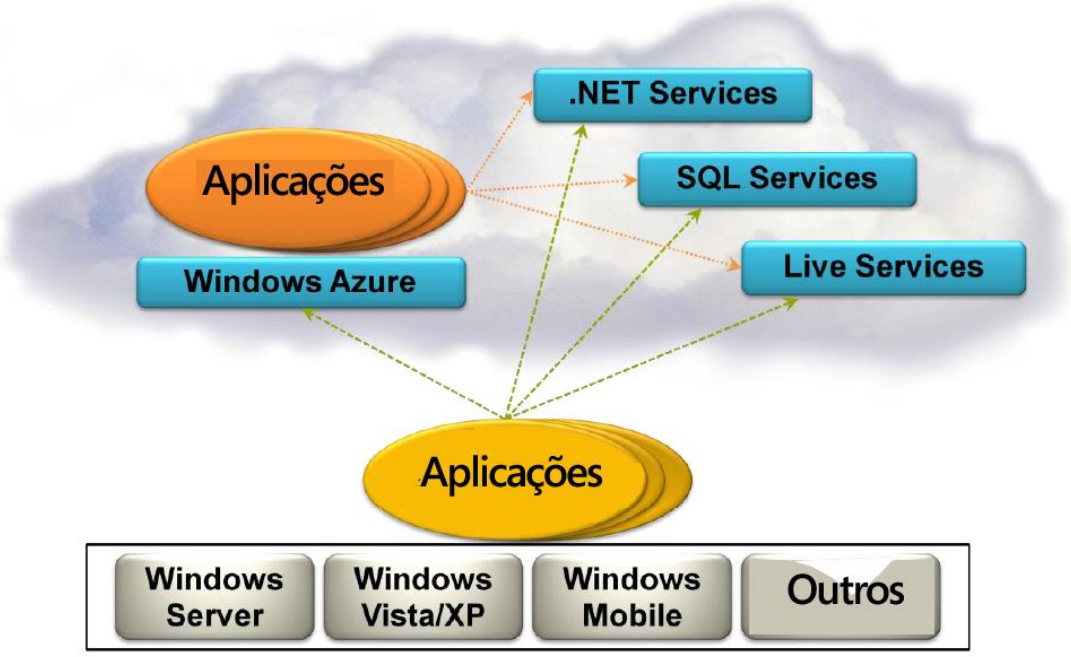

Figura 1: A Plataforma de Serviços Azure suporta aplicativos executados na nuvem e em sistemas locais.

Os componentes da Plataforma de Serviços Azure podem ser usados por aplicativos locais executados em uma variedade de sistemas, inclusive vários tipos de sistemas operacionais Windows, dispositivos móveis ou outros. Esses componentes incluem:

- *Windows Azure*: Fornece um ambiente baseado no Windows para executar aplicativos e armazenar dados nos servidores dos centros de dados da Microsoft.
- *Microsoft .NET Services*: Oferece serviços de infra-estrutura distribuídos para aplicativos baseados em nuvem e locais.
- *Microsoft SQL Services*: Fornece serviços de dados na nuvem baseados no SQL Server.
- *Live Services*: Através do Live Framework, fornece acesso aos dados a partir de aplicativos Live da Microsoft e outros. O Live Framework também permite sincronizar esses dados em estações de trabalho e dispositivos, encontrando e fazendo o download de aplicativos, e muito mais.

Cada componente da Plataforma de Serviços Azure tem seu próprio papel a desempenhar. Este documento descreve os quatro componentes, primeiro com uma visão geral e depois de uma forma um pouco mais detalhada. Embora nenhum deles ainda seja definitivo — detalhes ou outros podem mudar antes da versão inicial — não é cedo demais para começar a entender esse novo conjunto de tecnologias de plataforma.

WINDOWS AZURE

Em um alto nível, o Windows Azure é simples de entender: É uma plataforma para executar aplicativos Windows e armazenar seus dados na nuvem. A Figura 2 mostra seus principais componentes.

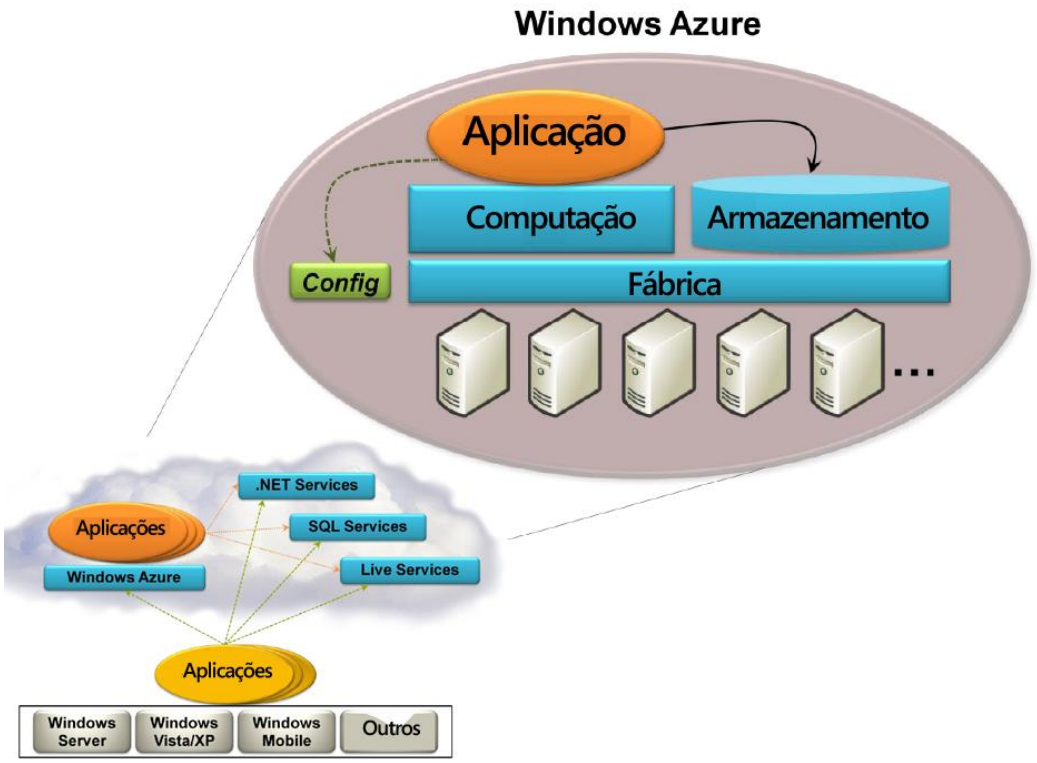

Figura 2: O Windows Azure fornece serviços de computação e armazenamento baseados no Windows para aplicativos de nuvem.

Como a figura sugere, o Windows Azure é executado em um grande número de máquinas, todas localizadas em centros de dados da Microsoft, acessíveis através da Internet. Uma malha comum do Windows Azure costura essa abundância de poder de processamento em uma totalidade unificada. Os serviços de computação e armazenamento do Windows Azure são construídos sobre essa malha.

O serviço de computação do Windows Azure se baseia no Windows, é claro. Para a disponibilidade inicial desse serviço, uma CTP (Community Technology Preview) tornada pública no segundo semestre de 2008 permitiu que o Windows Azure executasse apenas aplicativos construídos no .NET Framework. A empresa anunciou planos para dar suporte também a código não gerenciado, isto é, a aplicativos não construídos no .NET Framework, no Windows Azure em 2009.

Na versão CTP do Windows Azure, os desenvolvedores podem criar software baseado em .NET tais como aplicativos ASP.NET e serviços do WCF (Windows Communication Foundation). Para isso, eles podem usar C# e outras linguagens .NET, junto com ferramentas de desenvolvimento tradicionais como o Visual Studio 2008. E, embora muitos desenvolvedores provavelmente usem essa versão inicial do Windows Azure para criar aplicações Web, a plataforma também dá suporte a processos em segundo plano que são executados independentemente — não é unicamente uma plataforma de Web.

Tanto aplicativos Windows Azure como locais (on-premises) podem acessar o serviço de armazenamento do Windows Azure, e o fazem da mesma forma: usando uma abordagem RESTfull. O armazenamento de dados subjacente, porém, não é feito no Microsoft SQL Server. De fato, o armazenamento do Windows Azure não é um sistema relacional e sua linguagem de consulta não é o SQL. Como foi primariamente projetado para dar suporte a aplicativos baseados no Windows Azure, ele fornece tipos de armazenamento mais simples e escalonáveis. Dessa forma, permite armazenar blobs (binary large objects – grandes objetos

binários), fornece filas para comunicação entre componentes de aplicativos do Windows Azure e até mesmo oferece uma forma de tabelas com linguagem de consulta direta.

Executar aplicativos e armazenar seus dados na nuvem pode ter benefícios claros. Em vez de comprar, instalar e operar seus próprios sistemas, por exemplo, uma organização pode contar com um provedor na nuvem para fazer isso para ela. Além disso, os clientes pagam apenas pela computação e armazenamento que eles usam, em vez de manter um grande conjunto de servidores apenas para cargas de pico. E, se forem escritos corretamente, os aplicativos podem escalar facilmente, tirando proveito dos enormes centros de dados que os provedores da nuvem oferecem.

Entretanto, atingir esses benefícios exige um gerenciamento eficiente. No Windows Azure, cada aplicativo tem um arquivo de configuração, como mostrado na Figura 2. Ao alterar manual ou programaticamente as informações nesse arquivo, o proprietário de um aplicativo pode controlar vários aspectos de seu comportamento, tais como definir o número de instâncias que o Windows Azure deve executar. A malha do Windows Azure monitora o aplicativo para manter seu estado desejado.

Para permitir que seus clientes criem, configurem e monitorem aplicativos, o Windows Azure fornece um portal acessível por navegador. O cliente fornece um ID do Windows Live, depois escolhe criar uma conta de hospedagem para executar aplicativos, uma conta de armazenamento para guardar dados, ou ambos. Um aplicativo pode tarifar seus clientes como quiser: assinaturas, taxas por uso ou outro método.

O Windows Azure é uma plataforma geral que pode ser usada em vários cenários. Aqui estão alguns exemplos, todos baseados naquilo que a versão CTP permite:

- Uma empresa iniciante criando um novo site da Web — digamos, o próximo Facebook — poderia construir seu aplicativo sobre o Windows Azure. Como essa plataforma suporta tanto serviços voltados para a Web como processos em segundo plano, o aplicativo pode fornecer uma interface de usuário interativa assim como executar trabalho para usuários de maneira assíncrona. Em vez de gastar dinheiro e tempo preocupando-se com infra-estrutura, a empresa iniciante pode, em vez disso, concentrar-se unicamente em criar código que forneça valor a seus usuários e investidores. A empresa também pode começar pequena, incorrendo em baixos custos enquanto o aplicativo tiver apenas alguns usuários. Se o aplicativo se popularizar e o uso aumentar, o Windows Azure pode escalar o aplicativo como necessário.

- Um ISV criando uma versão SaaS (software-as-a-service – software como serviço) de um aplicativo .NET local existente pode escolher construí-lo no Windows Azure. Como o Windows Azure fornece principalmente um ambiente .NET padrão, mover a lógica de negócios .NET do aplicativo para essa plataforma de nuvem tipicamente não apresentará muitos problemas. E, uma vez mais, construir sobre uma plataforma existente permite ao ISV concentrar-se em sua lógica de negócios — aquilo que gera dinheiro — em vez de gastar tempo com a infra-estrutura.

- Uma empresa que crie um aplicativo para seus clientes pode escolher construí-lo no Windows Azure. Como o Windows Azure é baseado em .NET, desenvolvedores com as habilidades certas não são difíceis de encontrar e nem proibitivamente caros. Executar o aplicativo em centros de dados da Microsoft libera a empresa da responsabilidade e da despesa de gerenciar seus próprios servidores, transformando gastos de capital em despesas operacionais. E, especialmente se o aplicativo tiver períodos de pico de uso — talvez uma floricultura online que deve lidar com a explosão de vendas no Dia das Mães — deixar que a Microsoft mantenha a grande base de servidores necessária para isso pode ser economicamente benéfico.

Executar aplicativos na nuvem é um dos muitos aspectos importantes da computação na nuvem. Com o Windows Azure, a Microsoft fornece uma plataforma para fazer isso, junto com uma forma de armazenar dados de aplicativos.

À medida que o interesse na computação na nuvem continua a crescer, espere ver mais aplicativos Microsoft criados para esse novo mundo.

.NET SERVICES

Executar aplicativos na nuvem é um aspecto importante da computação na nuvem, mas é apenas parte da história. Também é possível fornecer serviços baseados em nuvem que podem ser usados por aplicativos locais ou aplicativos na nuvem. Preencher essa lacuna é a meta do .NET Services.

Originalmente conhecido como BizTalk Services, as funções fornecidas pelo .NET Services lidam com desafios de infra-estrutura comuns na criação de aplicativos distribuídos. A Figura 3 mostra seus componentes.

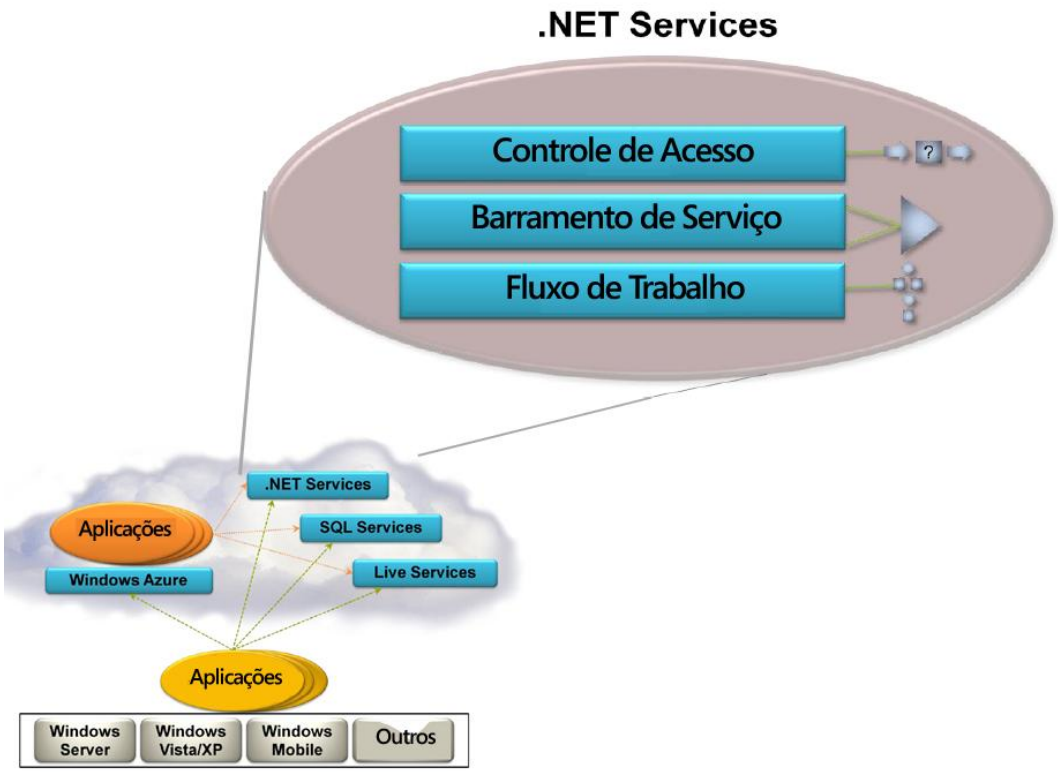

Figura 3: O .NET Services fornece infra-estrutura baseada em nuvem que pode ser usada por aplicativos na nuvem ou locais (on-premises).

Os componentes do .NET Services são:

- Controle de Acesso: Uma abordagem cada vez mais comum à identidade é fazer cada usuário fornecer ao aplicativo um *token* contendo algum conjunto de *declarações.* O aplicativo pode, então, decidir o que esse usuário tem permissão de fazer com base nessas declarações. Fazer isso efetivamente em empresas requer *federação de identidade,* que permite que declarações criadas em um escopo de identidade sejam aceitas em outro escopo. Ele também pode exigir *transformação de declarações,* modificando declarações quando são transmitidas entre escopos de identidade. O serviço de Controle de Acesso fornece uma implementação de ambos, baseada em nuvem.

- Barramento de Serviço: Expor os serviços de um aplicativo na Internet é mais difícil do que a maioria das pessoas pensa. O objetivo do Barramento de Serviço é simplificar isso permitindo que um aplicativo exponha end-points (pontos de extremidade) de serviços Web que possam ser acessados por outros aplicativos, sejam locais ou da nuvem. A cada end-point exposto é designado um URI que

os clientes podem usar para localizar e acessar o serviço. O Barramento de Serviço também trata dos desafios de lidar com tradução de endereços de rede e passar por firewalls semabrir novas portas para aplicativos expostos.

- Fluxo de Trabalho: Criar aplicativos compostos, como na integração de aplicativos corporativos, requer lógica que coordena a interação entre as várias partes. Essa lógica às vezes é mais bem implementada usando um fluxo de trabalho capaz de suportar processos de execução longa. Construído sobre o Windows WF (Workflow Foundation), o serviço de Fluxo de Trabalho permite a execução desse tipo de lógica na nuvem.

Eis alguns exemplos de como .NET Services podem ser usados:

- Um ISV que forneça um aplicativo usado por clientes em várias organizações diferentes pode usar o serviço de Controle de Acesso para simplificar o desenvolvimento e operação do aplicativo. Por exemplo, esse componente do .NET Services poderia traduzir as diversas declarações usadas nas várias organizações dos clientes, cada uma das quais podendo usar internamente uma tecnologia de identidade diferente, em um conjunto uniforme que o aplicativo do ISV pudesse usar. Fazer isso também permite o descarregamento da mecânica da federação de identidades no serviço de Controle de Acesso baseado na nuvem, liberando o ISV da execução de seu próprio software de federação no local.

- Imagine que uma empresa desejasse permitir que software de seus clientes comerciais acessasse um de seus aplicativos. Ela poderia expor as funções daquele aplicativo através de SOAP ou serviços de Web REST, depois registrar seus pontos de extremidade com o Barramento de Serviço. Seus parceiros comerciais poderiam, então, usar o Barramento de Serviço para encontrar aqueles pontos de extremidade e acessar os serviços. Como fazer isso não exige que se abram novas portas no firewall da organização, isto reduz o risco de expor o aplicativo. A organização também poderia usar o serviço de Controle de Acesso, que é projetado para trabalhar com o Barramento de Serviço, para racionalizar informações de identidade enviadas ao aplicativo por aqueles parceiros.

- Talvez a organização do exemplo anterior precise certificar-se de que um processo de negócios envolvendo seus parceiros comerciais deva ser executado de maneira consistente. Para isso, pode usar o serviço de Fluxo de Trabalho para implementar um aplicativo baseado no WF que execute esse processo. O aplicativo pode comunicar-se com parceiros usando o Barramento de Serviço e contar com o serviço de Controle de Acesso para atenuar as diferenças nas informações de identidade.

Como com o Windows Azure, um portal acessível por navegador é fornecido para permitir que clientes assinem .NET Services usando uma ID do Windows Live. O objetivo da Microsoft com o .NET Services é claro: fornecer infra-estrutura útil baseada em nuvem para aplicativos distribuídos.

SQL SERVICES

Uma das formas mais atraentes de se usar servidores acessíveis pela Internet é manipular dados. Certamente isso significa fornecer um banco de dados básico, mas também pode-se fazer mais. O objetivo do SQL Services é fornecer um conjunto de serviços baseados em nuvem para armazenar e trabalhar com muitos tipos de dados, de não-estruturados a relacionais.

A Microsoft diz que o SQL Services incluirá uma variedade de recursos orientados a dados, como relatórios, analítica de dados e outros. O primeiro componente do SQL Services a aparecer, contudo, é o SQL Data Services. A Figura 4 ilustra essa idéia.

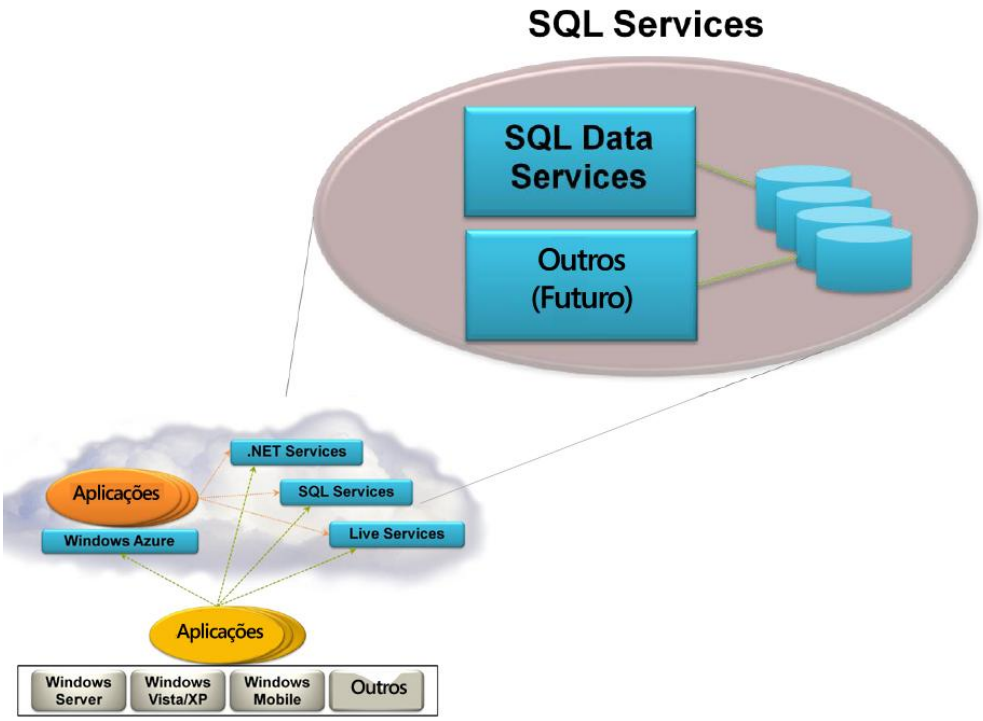

Figura 4: O SQL Services fornece recursos orientados a dados na nuvem.

O SQL Data Services, anteriormente conhecido como SQL Server Data Services, fornece um banco de dados na nuvem. Como a figura sugere, essa tecnologia permite que aplicativos locais e na nuvem armazenem e acessem dados em servidores Microsoft em centros de dados da Microsoft. Como ocorre com outras tecnologias de nuvem, uma organização pagará apenas por aquilo que usar, aumentando e reduzindo o uso (e custo) conforme as necessidades da organização mudam. Usar um banco de dados de nuvem também permite converter o que seriam gastos de capital, como investimentos em discos e em sistemas de gerenciamento de bancos de dados, em despesas operacionais.

Um objetivo primário do SQL Data Services é ser amplamente acessível. Para isso, ele expõe tanto interfaces SOAP como REST, permitindo que seus dados sejam acessados de várias maneiras. E, como esses dados são expostos através de protocolos padrão, o SQL Data Services pode ser usado por aplicativos em qualquer tipo de sistema — não é uma tecnologia exclusivamente para o Windows.

Diferentemente do serviço de armazenamento do Windows Azure, o SQL Data Services foi construído sobre o Microsoft SQL Server. Contudo, o serviço não expõe uma interface relacional tradicional. Em vez disso, o SQL Data Services fornece um modelo de dados hierárquico que não exige um esquema pré-definido. Cada item de dados armazenado nesse serviço é mantido como uma propriedade com seu próprio nome, tipo e valor. Para consultar esses dados, os aplicativos podem usar acesso REST direto ou uma linguagem baseada na sintaxe C# definida pela LINQ (Language Integrated Query) da Microsoft.

Existe uma questão óbvia aqui: Por que não apenas oferecer o SQL Server na nuvem? Por que, em vez disso, fornecer um serviço de banco de dados de nuvem que usa uma abordagem diferente daquela que a maioria de nós já conhece? Uma resposta é que fornecer esse conjunto de serviços ligeiramente diferentes oferece algumas vantagens. O SQL Data Services pode fornecer melhor escalabilidade, disponibilidade e confiabilidade do que é possível pela simples execução de um DBMS na nuvem. A maneira como ele organiza e recupera dados torna a replicação e o balanceamento de carga mais fáceis e rápidos do que com uma abordagem relacional tradicional. Outra vantagem é que o SQL Data Services

não requer que os clientes gerenciem seus próprios DBMS. Em vez de preocupar-se com a mecânica, como monitorar o uso do disco, fazer a manutenção de logs e determinar quantas instâncias são necessárias, um cliente do SQL Data Services pode concentrar-se no que é importante: os dados. E, finalmente, a Microsoft anunciou planos para adicionar mais recursos relacionais ao SQL Data Services. Espere um crescimento na funcionalidade deste serviço.

O SQL Data Services pode ser usado de várias maneiras. Aqui estão alguns exemplos:

- Um aplicativo pode arquivar dados mais antigos no SQL Data Services. Por exemplo, pense em um aplicativo que forneça feeds RSS atualizadas freqüentemente. Informações nessas feeds que tenham, digamos, mais de trinta dias provavelmente não são acessadas frequentemente, mas ainda precisam ficar disponíveis. Movê-las para o SQL Data Services pode fornecer uma alternativa de baixo custo e confiável.

- Suponha que um fabricante deseje disponibilizar informações de produto para sua rede de distribuidores e diretamente para os clientes. Colocar esses dados no SQL Data Services permitiria que eles fossem acessados por aplicativos executados nos distribuidores e por um aplicativo Web voltado para o cliente executado pelo próprio fabricante. Como os dados podem ser acessados por interfaces REST e SOAP, os aplicativos que usam estes dados podem ser escritos com qualquer tecnologia e executados em qualquer plataforma.

Como outros componentes da Plataforma de Serviços Azure, o SQL Data Services torna simples o uso de seus serviços. Vá a um portal da Web e forneça as informações necessárias. Quer seja para arquivar dados de forma barata, tornar dados acessíveis a aplicativos em locais diversos ou outros motivos, um banco de dados na nuvem pode ser uma idéia atraente. Conforme novas tecnologias se tornam disponíveis à sombra do SQL Services, as organizações terão a opção de usar a nuvem para mais e mais tarefas orientadas a dados.

LIVE SERVICES

Embora a idéia de plataformas na nuvem seja relativamente nova, a Internet não é. Centenas de milhões de pessoas em todo o mundo a utilizam todos os dias. Para ajudá-las a fazer isso, a Microsoft fornece um grupo crescente de aplicativos na Internet, inclusive a família Windows Live e outros. Esses aplicativos permitem que as pessoas enviem mensagens instantâneas, armazenem suas informações de contato, façam buscas, obtenham orientações e façam outras coisas úteis.

Todos esses aplicativos armazenam dados. Alguns desses dados, como contatos, variam com cada usuário. Outros, como o mapeamento e busca de informações, não variam — todos nós usamos as mesmas informações subjacentes. Em qualquer dos casos, por que não tornar esses dados disponíveis a outros aplicativos? Embora sejam necessários controles — expor livremente as informações pessoais de todos não é uma boa idéia — deixar que aplicativos usem essas informações faz sentido.

Para permitir isso, a Microsoft reuniu esse conjunto abrangente de recursos em um grupo chamado Live Services. Aplicativos da Microsoft existentes, como a família Windows Live, contam com Live Services para armazenar e gerenciar suas informações. Para permitir que aplicativos novos acessem essas informações, a Microsoft fornece o Live Framework.

A Figura 5 ilustra alguns dos aspectos mais importantes do Framework.

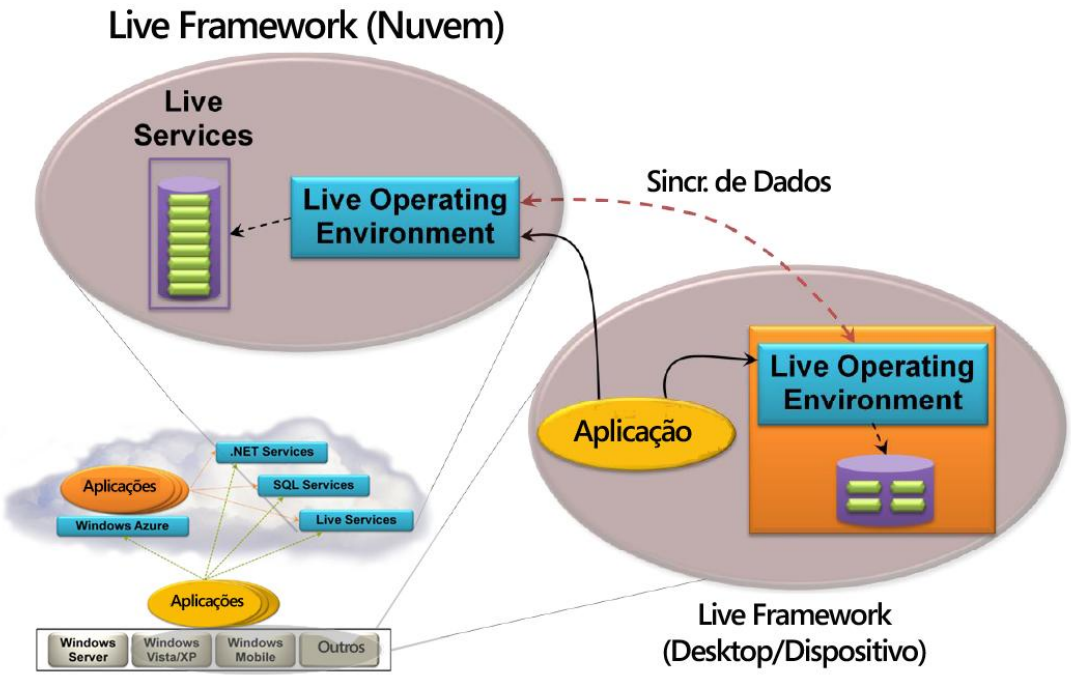

Figura 5: O Live Framework permite que aplicativos acessem dados do Live Services, opcionalmente sincronizando aqueles dados em estações de trabalho e dispositivos.

O componente fundamental no Live Framework é o Live Operational Environment (Ambiente Operacional do Live). Como a figura mostra, esse componente é executado na nuvem, e aplicativos o usam para acessar dados do Live Services. O Acesso a Dados através do Ambiente Operacional do Live conta com HTTP, o que significa que aplicativos escritos usando o .NET Framework, JavaScript, Java ou qualquer outra linguagem podem usar dados do Live Services. Informações no Live Services também podem ser acessadas como uma feed Atom ou RSS, permitindo que um aplicativo saiba sobre mudanças nesses dados. E, para configurar e gerenciar os Live Services de que seu aplicativo precisa, um desenvolvedor pode usar o Portal de Desenvolvedor de Live Services.

A Figura 5 mostra outro aspecto do Live Framework: O Ambiente Operacional do Live também pode residir em sistemas de estação de trabalho executando o Windows Vista, Windows XP ou Macintosh OS X e em dispositivos Windows Mobile 6. Para usar essa opção, um usuário agrupa seus sistemas naquele que é conhecido como malha (mesh). Por exemplo, você pode criar uma malha que contenha seu computador pessoal, seu notebook e seu telefone móvel. Cada um desses sistemas executa uma instância do Ambiente Operacional do Live.

Uma característica fundamental de toda malha é que o Ambiente Operacional do Live pode sincronizar dados em todos os sistemas da malha. Usuários e aplicativos podem indicar que tipos de dados devem ser mantidos em sincronia e o Ambiente Operacional do Live atualizará automaticamente todas as estações de trabalho, notebooks e dispositivos na malha com alterações feitas nos dados em qualquer um deles. E, como a nuvem é parte da malha de todos os usuários — ela atua como um dispositivo especial — isso inclui dados do Live Services. Por exemplo, se um usuário tiver entradas mantidas no banco de dados de contatos usado pelo Windows Live Hotmail, Windows Live Messenger, Contatos do Windows Live e outros aplicativos, esses dados são automaticamente mantidos em sincronismo em todos os dispositivos na malha dele (entretanto, esta habilidade ainda não é suportada no CTP de Novembro de 2008 do Live Framework). O Ambiente Operacional do Live

também permite que o usuário exponha dados a partir de sua malha a outros, assim compartilhando seletivamente essas informações.

Como a Figura 5 mostra, um aplicativo pode acessar dados da malha através da instância local do Ambiente Operacional do Live ou da instância da nuvem. Em ambos os casos, o acesso é conseguido da mesma forma: através de solicitações HTTP. Essa simetria permite que um aplicativo trabalhe identicamente, esteja ele conectado à nuvem ou não — os mesmos dados ficam disponíveis e são acessados da mesma maneira.

Qualquer aplicativo, em execução no Windows ou em outro sistema operacional, pode acessar dados do Live Service na nuvem via Ambiente Operacional do Live. Se o aplicativo estiver em execução em um sistema que seja parte de uma malha, ele também tem a opção de usar o Ambiente Operacional do Live para acessar uma cópia local daqueles dados do Live Services, como descrito acima. Existe também uma terceira possibilidade, entretanto: um desenvolvedor pode criar o que é chamado de aplicativo de Web habilitado para malha. Esse estilo de aplicativo é construído usando-se uma tecnologia multiplataforma como o Microsoft Silverlight e acessa dados através do Ambiente Operacional do Live. Por causa dessas restrições, um aplicativo habilitado para a malha pode ser executado em qualquer máquina na malha de um usuário — uma máquina Windows, Macintosh, ou dispositivo Windows Mobile — e sempre tem acesso aos mesmos dados (sincronizados). Para ajudar o usuário a encontrar esses aplicativos, o ambiente do Live Framework fornece um catálogo de aplicativos baseados em nuvem para aplicativos de Web compatíveis com malha. Um usuário pode examinar esse catálogo, escolher um aplicativo e instalá-lo. E, para ajudar os criadores a construir um negócio a partir de seu trabalho, a Microsoft planeja fornecer suporte embutido para exibir publicidade nesses aplicativos.

O Live Framework oferece um conjunto diversificado de funções que podem ser usadas em uma variedade de maneiras diferentes. Aqui estão alguns exemplos:

- Um aplicativo Java executado em Linux pode contar com o Live Framework para acessar as informações de contato de um usuário. O aplicativo não tem conhecimento de que a tecnologia usada para expor essas informações é o Live Framework; tudo que ele vê é uma interface HTTP consistente com os dados do usuário.

- Um aplicativo criado com base no .NET Framework pode exigir que seu usuário crie uma malha, depois usar o Live Framework como um serviço de cache e sincronização de dados. Quando a máquina em que esse aplicativo é executado é conectada à Internet, o aplicativo acessa uma cópia de seus dados na nuvem. Quando a máquina é desconectada — talvez esteja sendo executada em um laptop em um avião — o aplicativo acessa uma cópia local dos mesmos dados. Alterações feitas em qualquer cópia dos dados são propagadas pelo Ambiente Operacional do Live.

- Um ISV pode criar um aplicativo de Web habilitado para malha que ajude as pessoas a acompanhar o que seus amigos estão fazendo. Esse aplicativo, que pode ser executado inalterado em todos os sistemas de seu usuário, explora vários aspectos do Live Framework que suportam aplicativos sociais. Como o Live Framework pode expor informações na malha de um usuário como uma feed, por exemplo, o aplicativo pode rastrear atualizações de qualquer dos amigos do usuário. Como o Live Framework fornece um mecanismo de fornecimento para aplicativos de Web compatíveis com malha, a distribuição viral é possível, com cada usuário convidando amigos a usar o aplicativo. E, como a malha inclui automaticamente os contatos do Live Services do usuário, este pode pedir ao aplicativo que convide amigos por nome, permitindo que o aplicativo os contate diretamente.

O Live Framework fornece uma maneira direta para acessar dados do Live Services (e não se deixe enganar pelo exemplo simples de contatos usado aqui — há muito mais no Live Services). Suas funções de

sincronização de dados também podem ser empregadas em uma variedade de aplicativos. Para aplicativos que precisam daquilo que ele fornece, essa plataforma oferece um conjunto único de funções que o suportam.

MAIS DETALHES SOBRE AS TECNOLOGIAS

Uma compreensão ampla da Plataforma de Serviços Azure é um primeiro passo importante. Conseguir uma compreensão mais profunda de cada tecnologia também é útil, entretanto. Essa seção examina um pouco mais profundamente cada membro da família.

WINDOWS AZURE

O Windows Azure faz duas coisas principais: Executa aplicativos e armazena seus dados. Por conseguinte, essa seção é dividida em duas partes, uma para cada uma dessas áreas. Como essas coisas são gerenciadas também é importante, assim esta descrição também examina essa parte da história.

Execução de Aplicativos

No Windows Azure, um aplicativo tipicamente possui múltiplas *instâncias*, cada qual executando uma cópia do todo ou de parte do código dele. Cada uma dessas instâncias é executada em sua própria máquina virtual (VM). Essas VMs são executadas em Windows Server 2008 64 bits e são fornecidas por um hipervisor que é especificamente projetado pelo uso na nuvem.

Contudo, um aplicativo Windows Azure não pode realmente ver a VM em que está sendo executado. Um desenvolvedor não pode fornecer sua própria imagem de VM para o Windows Azure ser executado, nem precisa preocupar-se em manter essa cópia do sistema operacional Windows. Em vez disso, a versão da CTP permite ao desenvolvedor criar aplicativos .NET 3.5 usando instâncias de função de Web e/ou de Função de trabalho. A Figura 6 mostra a aparência disso.

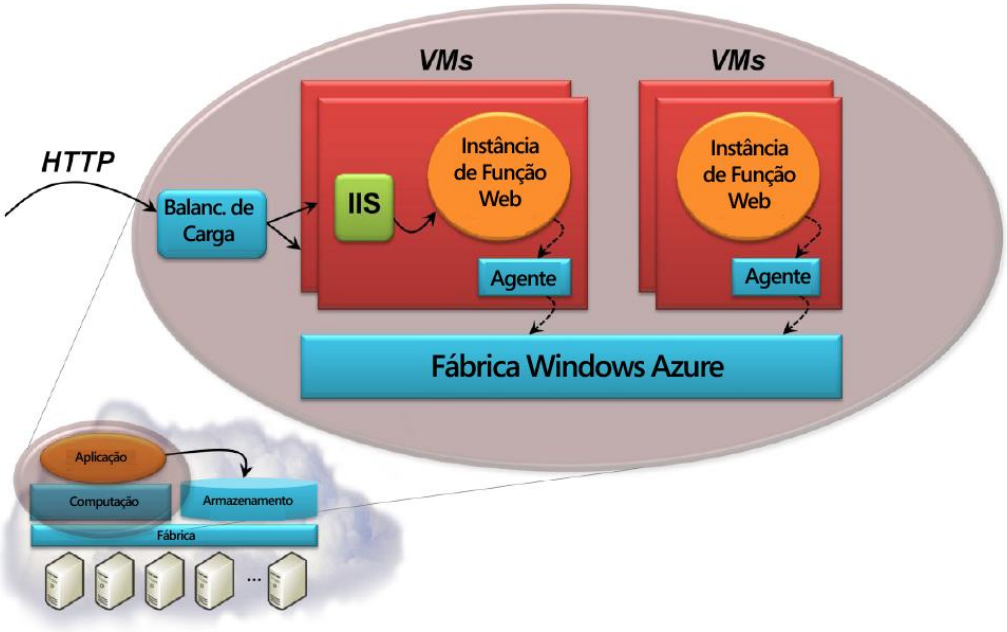

Figura 6: Na versão de CTP inicial, aplicativos Windows Azure podem consistir em instâncias de função de Web e de Trabalho, com cada uma delas sendo executada em sua própria máquina virtual.

Como o nome sugere cada instância de Função de Web aceita solicitações HTTP (ou HTTPS) via IIS (Internet Information Services) 7.0. Uma função de Web pode ser implementada usando ASP.NET, WCF ou outra tecnologia do .NET Framework que funcione com o IIS. Como a Figura 6 mostra, o Windows Azure fornece balanceamento de carga embutido para disseminar solicitações em instâncias de função de Web que sejam parte do mesmo aplicativo.

Uma instância de Função de Trabalho, em comparação, não pode aceitar solicitações diretamente do mundo exterior — não tem permissão de receber nenhuma conexão de entrada, e o IIS não está executando sua VM (Máquina Virtual). Em vez disso, recebe sua entrada de uma instância de função de Web, tipicamente via uma fila no armazenamento do Windows Azure. O resultado desse trabalho pode ser gravado no armazenamento do Windows Azure ou enviado para o mundo exterior — conexões de rede de saída são permitidas. Diferentemente de uma instância de função de Web, que é criada para lidar com uma solicitação HTTP de entrada e desligada depois que aquela solicitação é processada, uma instância de função de Trabalho pode ser executada indefinidamente — é um trabalho em lotes. Condizente com esse princípio, uma função de Trabalho pode ser implementada usando-se qualquer tecnologia .NET com um método principal (sujeita aos limites de relações de confiança do Windows Azure, como descrito abaixo).

Quer execute uma instância de função de Web ou de Trabalho, cada VM também contém um agente do Windows Azure que permite que o aplicativo interaja com a malha do Windows Azure, como mostrado na Figura 6. O agente expõe uma API definida pelo Windows Azure que permite que a instância grave em um log mantido pelo Windows Azure, envie alertas ao seu proprietário via malha do Windows Azure e mais.

Embora isso possa mudar com o passar do tempo, o lançamento inicial do Windows Azure mantém uma relação de um para um entre uma VM e um núcleo de processador físico. Por isso, o desempenho de cada aplicativo pode ser garantido — cada instância de função da Web e de Trabalho possui seu próprio núcleo de processador dedicado. Para aumentar o desempenho de um aplicativo, seu proprietário pode aumentar o número de instâncias em execução especificado no arquivo de configuração do aplicativo. A malha do Windows Azure então gera novas VMs, atribuem-lhes núcleos e começa a executar mais instâncias desse aplicativo. A malha também detecta quando uma função de Web ou de Trabalho falha e inicia uma nova.

Note o que isso implica: Para ser escalonáveis, funções de Web do Windows Azure não devem ter monitoramento de estado. Qualquer estado específico de cliente deve ser gravado no armazenamento do Windows Azure ou devolvido ao cliente em um cookie. A condição de não monitoramento de estado da função de Web também é quase obrigatória pelo balanceador de carga embutido do Windows Azure. Como ele não permite a criação de uma afinidade com uma determinada instância de função de Web, não há como garantir que várias solicitações do mesmo usuário sejam enviadas à mesma instância.

Tanto funções de Web e de Trabalho são implementadas usando tecnologias .NET padrão. Contudo, mover aplicativos do .NET Framework existentes para o Windows Azure sem alterações geralmente não funciona. Primeiro, a maneira como um aplicativo acessa o armazenamento é diferente. O acesso ao armazenamento do Windows Azure usa Web Services ADO.NET, uma tecnologia relativamente nova que ainda não está presente em todos os aplicativos locais. Similarmente, instâncias de função de Trabalho tipicamente contam com filas no armazenamento do Windows Azure para suas entradas, uma abstração que não está disponível em ambientes Windows locais. Outra limitação é que aplicativos Windows Azure não são executados em um ambiente de relação de confiança completa. Em vez disso, são limitados ao que a Microsoft chama de relação de confiança do Windows Azure, que é similar à relação de confiança média permitida por muitos hosters atualmente.

Para desenvolvedores, construir um aplicativo Windows Azure na versão CTP parece muito com construir um aplicativo .NET tradicional. A Microsoft fornece modelos de projeto do Visual Studio 2008 para criação de funções de Web, de Trabalho e a combinação das duas no Windows Azure. Os desenvolvedores podem usar qualquer linguagem .NET (embora o foco inicial da Microsoft para o Windows Azure tenha sido em C#). Além disso, o SDK (kit de desenvolvimento de software) do Windows Azure inclui uma versão do ambiente dele que é executado na máquina do desenvolvedor. Esse Windows Azure pronto inclui armazenamento, um agente e tudo mais visível por um aplicativo executado na nuvem. Um desenvolvedor pode criar e depurar esse aplicativo usando esse simulacro local, depois implantá-lo no Windows Azure na nuvem quando estiver pronto. Ainda assim, algumas coisas são realmente diferentes na nuvem. Não é possível vincular um depurador a um aplicativo baseado na nuvem, por exemplo, e, portanto, a depuração de aplicativos na nuvem conta primariamente com a gravação em um log mantido pelo Windows Azure via agente.

O Windows Azure também fornece outros serviços para desenvolvedores. Por exemplo, um aplicativo do Windows Azure pode enviar uma string num alerta através de seu agente e ele encaminha o alerta via e-mail, mensagem instantânea ou outro mecanismo para seu destinatário. Se desejado, a malha do Windows Azure pode ela mesma detectar uma falha de aplicativo e enviar um alerta. A plataforma Windows Azure também fornece informações detalhadas sobre o consumo de recursos pelo aplicativo, inclusive tempo de processador, largura de banda de entrada e saída e armazenamento.

Acesso a Dados

Aplicativos trabalham com dados de várias maneiras diferentes. Às vezes, tudo que é necessário são blobs simples, enquanto outras situações pedem uma maneira mais estruturada para armazenar informações. E, em alguns casos, tudo que é realmente necessário é uma forma de trocar dados entre diferentes partes de um aplicativo. O armazenamento do Windows Azure lida com esses três requisitos, como demonstra a Figura 7.

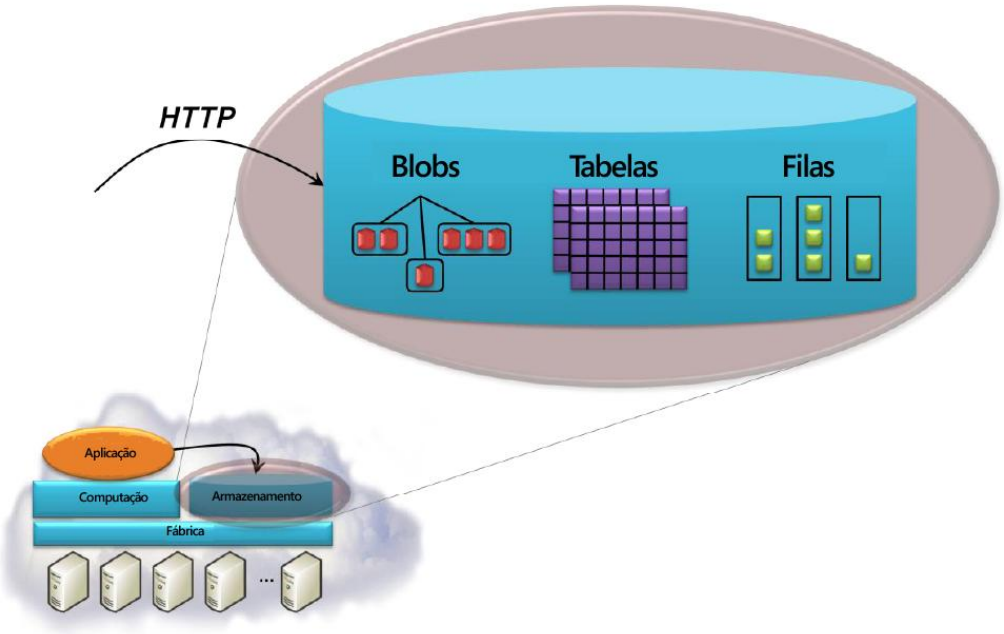

Figura 7: O Windows Azure permite o armazenamento de dados em blobs, tabelas e filas, todos acessados segundo o estilo REST via HTTP.

A maneira mais simples de guardar dados no armazenamento do Windows Azure é usar blobs. Como a Figura 7 sugere, existe uma hierarquia simples: uma conta de armazenamento pode ter um ou mais contêineres, cada qual com um ou mais blobs. Blobs podem ser grandes — com até 50 gigabytes — e para tornar mais eficiente a transferência de blobs grandes, cada um pode ser subdividido em blocos. Se ocorrer uma falha, a retransmissão pode recomeçar com o bloco mais recente em vez de enviar o blob inteiro novamente. Blobs também podem ter metadados associados, como informações sobre onde uma fotografia JPEG foi tirada ou quem é o compositor de um arquivo MP3.

Blobs são certos para alguns tipos de dados, mas não são estruturados o suficiente para muitas situações. Para permitir que aplicativos trabalhem com dados de uma forma mais granulada, o armazenamento do Windows Azure fornece tabelas. Não se deixe enganar pelo nome: não são tabelas relacionais. De fato, embora sejam chamadas de “tabelas”, os dados que contêm na verdade são armazenados em uma hierarquia simples de entidades com propriedades. Uma tabela não tem esquema definido; em vez disso, propriedades podem ter vários tipos, tais como String, Boolean ou DateTime. E, em vez de usar SQL, um aplicativo acessa os dados de uma tabela usando uma linguagem de consulta com sintaxe Linq. Uma única tabela pode ser bastante grande, com bilhões de entidades contendo terabytes de dados, e o armazenamento do Windows Azure pode particioná-la por muitos servidores, se necessário, para melhorar o desempenho.

Tanto blobs como tabelas focam no armazenamento de dados. A terceira opção no armazenamento do Windows Azure, as filas, tem uma finalidade diferente. A função primária das filas é fornecer uma forma para que instâncias de função de Web se comuniquem com instâncias de função de Trabalho. Por exemplo, um usuário pode enviar uma solicitação para realizar uma tarefa de computação intensa via uma página da Web implementada por uma função de Web do Windows Azure. A instância de função de Web que recebe essa solicitação pode gravar uma mensagem em uma fila descrevendo o trabalho a ser feito. Uma instância de função de Trabalho que está esperando nessa fila pode, então, ler a mensagem e executar a tarefa especificada. Quaisquer resultados podem ser retornados via outra fila ou tratados de alguma outra forma.

Independentemente de como são armazenados — em blobs, tabelas ou filas — todos os dados mantidos no armazenamento do Windows Azure são replicados três vezes. Essa replicação permite tolerância a falhas, já que perder uma cópia não é fatal. O sistema garante consistência, entretanto. Assim, um aplicativo que leia dados que tenham acabado de ter sido gravado recebe o que espera.

O armazenamento do Windows Azure pode ser acessado por um aplicativo do Windows Azure ou por outro sendo executado em outro local. Nos dois casos, os três estilos do armazenamento do Windows Azure usam as convenções de REST para identificar e expor dados. Tudo é nomeado usando URIs e acessado com operações HTTP padrão. Um cliente .NET também pode usar Data Services ADO.NET e LINQ, mas o acesso ao armazenamento do Windows Azure a partir de, digamos, um aplicativo Java pode usar apenas REST padrão. Por exemplo, um blob pode ser lido com um GET HTTP em um URI formatado assim:

http://<StorageAccount>.blob.core.windows.net/<Container>/<BlobName>

<StorageAccount> é um identificador atribuído quando uma nova conta de armazenamento é criada e identifica exclusivamente os blobs, tabelas e filas criadas usando essa conta. <Container> e <BlobName> são apenas os nomes do contêiner e do blob que essa solicitação está acessando.

Similarmente, uma consulta a uma determinada tabela é expressa como um GET HTTP em um URI formatado assim: http://<StorageAccount>.table.core.windows.net/<TableName>?$filter=<Query>

Aqui, <TableName> especifica a tabela sendo consultada, enquanto <Query> contém a consulta a ser executada nessa tabela.

Até mesmo filas podem ser acessadas por aplicativos do Windows Azure e externas emitindo um GET HTTP em um URI formatado assim: http://<StorageAccount>.queue.core.windows.net/<QueueName>

A plataforma Windows Azure cobra de forma independente pelos recursos de computação e armazenamento. Isso significa que um aplicativo local pode usar o armazenamento do Windows Azure apenas, acessando seus dados no formato REST que acabamos de descrever. Ainda, é justo dizer que a finalidade primária do armazenamento do Windows Azure é manter dados usados por aplicativos do Windows Azure. E, como aqueles dados podem ser acessados diretamente a partir de aplicativos diferentes do Windows Azure, ficam disponíveis se o aplicativo do Windows Azure que o usar não estiver em execução.

O objetivo das plataformas de aplicações, tanto local como na nuvem, é suportar aplicativos e dados. O Windows Azure oferece um ambiente para essas duas coisas. Num futuro, espere ver a partilha do que teriam sido aplicativos para o Windows local sendo construídos para essa nova plataforma na nuvem.

.NET SERVICES

Executar aplicativos na nuvem é útil, mas fornecer serviços de infra-estrutura baseados em nuvem também é. Esses serviços podem ser usados por aplicativos locais ou baseados na nuvem, e podem lidar com problemas que não podem ser resolvidos tão bem de qualquer outra maneira. Esta seção examina em mais detalhes as ofertas da Microsoft nessa área: o Serviço de Controle de Acesso .NET, Barramento de Serviço .NET e o Serviço de Fluxo de Trabalho .NET, conhecidos coletivamente como .NET Services.

Serviço de Controle de Acesso

Trabalhar com identidades é uma parte fundamental da maioria dos aplicativos distribuídos. Baseado nas informações de identidade de um usuário, um aplicativo toma decisões sobre o que aquele usuário pode fazer. Para transmitir essas informações, os aplicativos podem contar com tokens definidos usando SAML (Security Assertion Markup Language). Um token SAML contém declarações, cada uma das quais levando alguma informação sobre um usuário. Uma declaração pode conter o nome, outra pode indicar o cargo, como gerente, enquanto uma terceira contém o endereço de e-mail. Tokens são criados por softwares chamados STS (security token service - serviço de token de segurança), que assina digitalmente cada um deles para confirmar sua origem.

Uma vez que um cliente (como um navegador da Web) tenha um token para seu usuário, ele pode apresentá-lo a um aplicativo. O aplicativo então usa as declarações do token para decidir o que aquele usuário tem permissão de fazer. Contudo, há alguns problemas possíveis:

- E se o token não contiver as declarações de que aquele aplicativo precisa? Com identidade baseada em declarações, cada aplicativo tem a liberdade de definir o conjunto de declarações que seus usuários devem apresentar. Entretanto, o STS que criou aquele token pode não ter colocado nele exatamente aquilo que o aplicativo requer.

- E se o aplicativo não confiar no STS que emitiu aquele token? Um aplicativo não pode aceitar tokens emitidos por qualquer STS. Em vez disso, o aplicativo tipicamente tem acesso a uma lista de certificados para STSs confiáveis, permitindo que ele valide as assinaturas de tokens que aqueles

criam. Apenas tokens daqueles STSs confiáveis são aceitos.

Incluir outro STS no processo pode resolver os dois problemas. Para assegurar que os tokens contenham as declarações certas, esse STS extra realiza a transformação de declaração. O STS pode conter regras que definem como declarações de entrada e saída devem ser relacionadas, depois usa aquelas regras para gerar um novo token contendo as declarações exatas que um aplicativo exige. Tratar do segundo problema, geralmente chamado de federação de identidade, requer que o aplicativo confie no novo STS. Também requer estabelecer uma relação de confiança entre esse novo STS e aquele que gerou o token que o STS recebeu.

Acrescentar outro STS permite transformação de declarações e federação de identidades, ambas úteis. Mas, onde esse STS deve ser executado? É possível usar um STS que seja executado dentro de uma organização, uma opção oferecida por vários fornecedores atualmente. Mas, por que não executar um STS na nuvem? Isso o tornaria acessível a usuários e aplicativos em qualquer organização. E também transfere o fardo de executar e gerenciar o STS em um provedor de serviços.

Isso é exatamente o que o Serviço de Controle de Acesso oferece: é um STS na nuvem. Para ver como esse STS pode ser usado, suponha que um ISV forneça um aplicativo acessível via Internet que pode ser usado por pessoas em várias organizações diferentes. Embora todas aquelas organizações possam fornecer tokens SAML para seus usuários, é improvável que esses tokens contenham o conjunto exato de declarações de que esse aplicativo precisa. A Figura 8 ilustra como o Serviço de Controle de Acesso pode lidar com esses desafios.

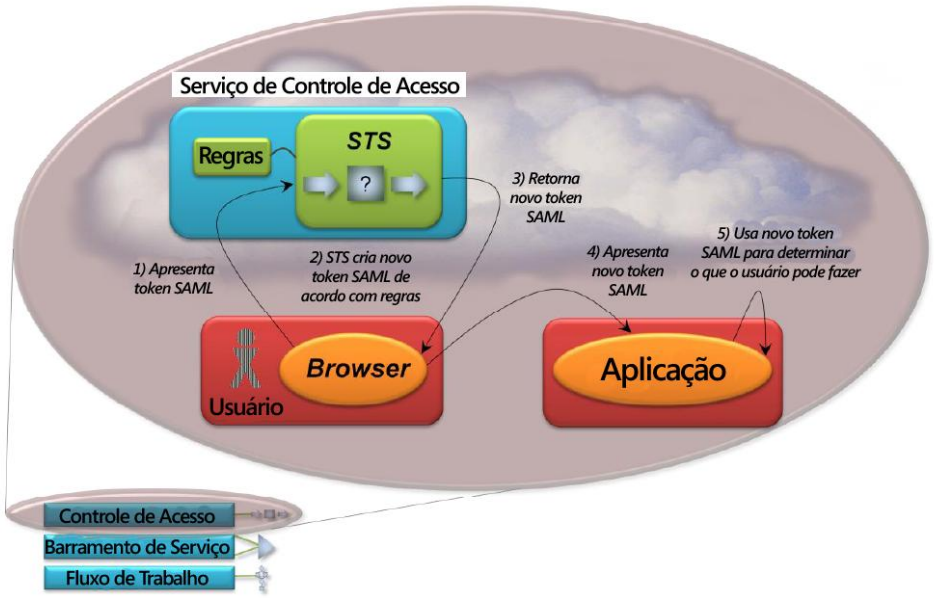

Figura 8: O Serviço de Controle de Acesso fornece transformação de declarações e federação de identidades baseadas em regras.

Primeiro, o aplicativo do usuário (que, neste exemplo é um navegador da Web, mas poderia ser um cliente WCF ou outro) envia o token SAML do usuário para o Serviço de Controle de Acesso (etapa 1). Esse serviço valida a assinatura no token, confirmando que ele foi criado por um STS em que o serviço confia. O serviço então cria e assina um novo token SAML contendo exatamente as declarações que esse aplicativo exige (etapa 2).

Para isso, o STS do Serviço de Controle de Acesso conta com regras definidas pelo proprietário do aplicativo que o usuário está tentando acessar. Por exemplo, suponha que o aplicativo conceda direitos de acesso específicos a qualquer usuário que seja gerente na empresa. Embora cada empresa possa incluir uma declaração em seu token indicando que um usuário é um gerente, provavelmente todos eles seriam diferentes. Uma empresa poderia usar a cadeia de caracteres “Gerente”, outra “Supervisor” e uma terceira um código de número inteiro. Para ajudar o aplicativo a lidar com essa diversidade, seu proprietário pode definir regas no Controle de Acesso que convertam essas três declarações na cadeia de caracteres comum "Responsável por Decisões". A vida do aplicativo foi simplificada, já que o trabalho de transformação de declarações é feito para ele.

Uma vez criado, o STS no Serviço de Controle de Acesso retorna esse novo token para o cliente (etapa 3), que o repassa para o aplicativo (etapa 4). O aplicativo valida a assinatura no token, certificando-se de que ele foi realmente emitido pelo STS do Serviço de Controle de Acesso. Note que, embora o STS do serviço deva manter uma relação de confiança com o STS da organização de cada cliente, o próprio aplicativo precisa confiar apenas no STS do Serviço de Controle de Acesso. Uma vez que se assegure da procedência do token, o aplicativo pode usar as declarações que ele contém para decidir o que o usuário tem permissão de fazer.

Outra maneira de usar o Serviço de Controle de Acesso está implícita no nome: Um aplicativo pode efetivamente descarregar no serviço as decisões sobre que tipo de permissão de acesso cada usuário recebe. Por exemplo, suponha que o acesso a certa função de um aplicativo exija que o usuário apresente uma determinada declaração. As regras do Serviço de Controle de Acesso para o aplicativo podem ser definidas para dar essa declaração apenas a usuários que apresentem outras declarações exigidas, como uma das de gerente descritas anteriormente. Quando o aplicativo recebe um token de usuário, pode conceder ou negar acesso com base na presença dessa declaração — a decisão foi efetivamente tomada para ele pelo Serviço de Controle de Acesso. Fazer isso permite que um administrador defina regras de controle de acesso em um local comum e também pode ajudar no compartilhamento de controle de acesso em vários aplicativos.

Toda a comunicação com o Serviço de Controle de Acesso se baseia em protocolos padrão como WS-Trust e WS- Federation. Isso torna o serviço acessível a partir de qualquer tipo de aplicativo em qualquer plataforma. E, para definir regras, o serviço fornece tanto uma interface gráfica baseada em navegador como uma API cliente para acesso programático.

Identidade baseada em declarações caminha para se tornar a abordagem padrão para ambientes distribuídos. Ao fornecer um STS na nuvem, completo com transformação de declarações baseada em regras, o Serviço de Controle de Acesso torna mais atraente essa abordagem moderna à identidade.

Barramento de Serviço

Suponha que você tem um aplicativo em execução dentro de sua organização que você gostaria de expor a softwares em outras organizações através da Internet. À primeira vista, isso pode parecer um problema simples. Presumindo que seu aplicativo forneça essa funcionalidade como serviços da Web (como REST ou baseados em SOAP), você pode simplesmente deixar esses serviços visíveis para o mundo exterior. Quando você realmente tenta fazer isso, entretanto, surgem alguns problemas.

Primeiro, como aplicativos em outras organizações (ou até mesmo em outras partes da sua própria) encontram pontos de extremidade aos quais possam se conectar para os seus serviços? Seria bom haver algum tipo de registro de onde outros poderiam localizar o seu aplicativo. E, uma vez que o encontrem, como as solicitações de softwares de outras organizações podem chegar ao seu aplicativo? A NAT (network

address translation - conversão de endereço de rede) é muito comum, assim um aplicativo freqüentemente não possui um endereço IP fixo para expor externamente. E, mesmo que a NAT não estiver sendo usada, como as solicitações podem atravessar o seu firewall? É possível abrir portas de firewall para permitir acesso a seu aplicativo, mas a maioria dos administradores de rede desaprova isso.

O Barramento de Serviço trata desses desafios. A Figura 9 mostra como.

Figura 9: O Barramento de Serviço permite a um aplicativo registrar pontos de extremidade e depois fazer outros aplicativos descobrir e usá-los para acessar seus serviços.

Para começar, seu aplicativo registra um ou mais pontos de extremidade com o Barramento de Serviço (passo 1), que os expõe em seu nome. O Barramento de Serviço atribui uma raiz de URI à sua organização, abaixo da qual você tem a liberdade de criar qualquer hierarquia de nomenclatura que quiser. Isso permite que seus pontos de extremidade sejam atribuídos a URIs específicos e detectáveis. Seu aplicativo também deve abrir uma conexão com o Barramento de Serviço para cada ponto de extremidade que expuser. O Barramento de Serviço mantém essa conexão aberta, o que resolve dois problemas. Primeiro a NAT não é mais um problema, já que o tráfego na conexão aberta com o Barramento de Serviço será sempre roteado para o seu aplicativo. Segundo, como a conexão foi iniciada atrás do firewall, não há problema em passar informações de volta para o aplicativo — o firewall não bloqueará esse tráfego.

Quando um aplicativo em outra organização (ou até mesmo em outra parte da sua própria) desejar acessar seu aplicativo, entra em contato com o registro do Barramento de Serviço (etapa 2). Essa solicitação usa o Atom Publishing Protocol e retorna um documento de serviço AtomPub com referências aos pontos de extremidade de seu aplicativo. Uma vez que ele os tenha, pode invocar serviços oferecidos através desses pontos de extremidade (etapa 3). Cada solicitação é recebida pelo Barramento de Serviço, depois repassada para o seu aplicativo, com as respostas fazendo o caminho inverso. E, embora não seja mostrado na figura, o Barramento de Serviço estabelece uma conexão direta entre um aplicativo e seu cliente sempre que possível, tornando a comunicação deles mais eficiente.

Além de facilitar a comunicação, o Barramento de Serviço também pode melhorar a segurança. Como os clientes agora vêem apenas um endereço IP fornecido pelo Barramento de Serviço, não há a necessidade de expor nenhum endereço IP dentro de sua organização. Isso torna seu aplicativo efetivamente anônimo, já que o mundo exterior não pode ver o endereço IP dele. O Barramento de Serviço atua como uma DMZ externa, fornecendo uma camada de dissimulação para conter ataques. E, finalmente, o Barramento de Serviço é projetado para ser usado com o Serviço de Controle de Acesso, permitindo transformação de declarações baseada em regras. De fato, o Barramento de Serviço aceita apenas tokens emitidos pelo STS do Serviço de Controle de Acesso.

Um aplicativo que deseje expor seus serviços via Barramento de Serviço é tipicamente implementado usando o WCF. Clientes podem ser montados com o WCF ou outras tecnologias, como Java, e podem fazer solicitações via SOAP ou HTTP. Aplicativos e seus clientes também têm a liberdade de usar seus próprios mecanismos de segurança, como criptografia, para proteger suas comunicações de ataques e do próprio Barramento de Serviço.

Expor aplicativos ao mundo exterior não é tão simples quanto pode parecer. O objetivo do Barramento de Serviço é tornar a implementação desse comportamento útil o mais direto possível.

Serviço de Fluxo de Trabalho

O Windows Workflow Foundation é uma tecnologia geral para criar aplicativos baseados em fluxo de trabalho. Um cenário clássico para fluxo de trabalho é controlar um processo prolongado, como é freqüentemente feito em integração de aplicativos corporativos. Mais geralmente, um aplicativo baseado no WF pode ser uma boa escolha para coordenar muitos tipos de trabalho. Especialmente quando o trabalho sendo coordenado se localiza em diferentes organizações, executar a lógica de controle na nuvem pode ser interessante.

O Serviço de Fluxo de Trabalho permite isso. Ao fornecer um processo de host para aplicativos baseados no WF 3.5, ele permite que desenvolvedores criem fluxos de trabalho que sejam executados na nuvem. A Figura 10 mostra a aparência disso.

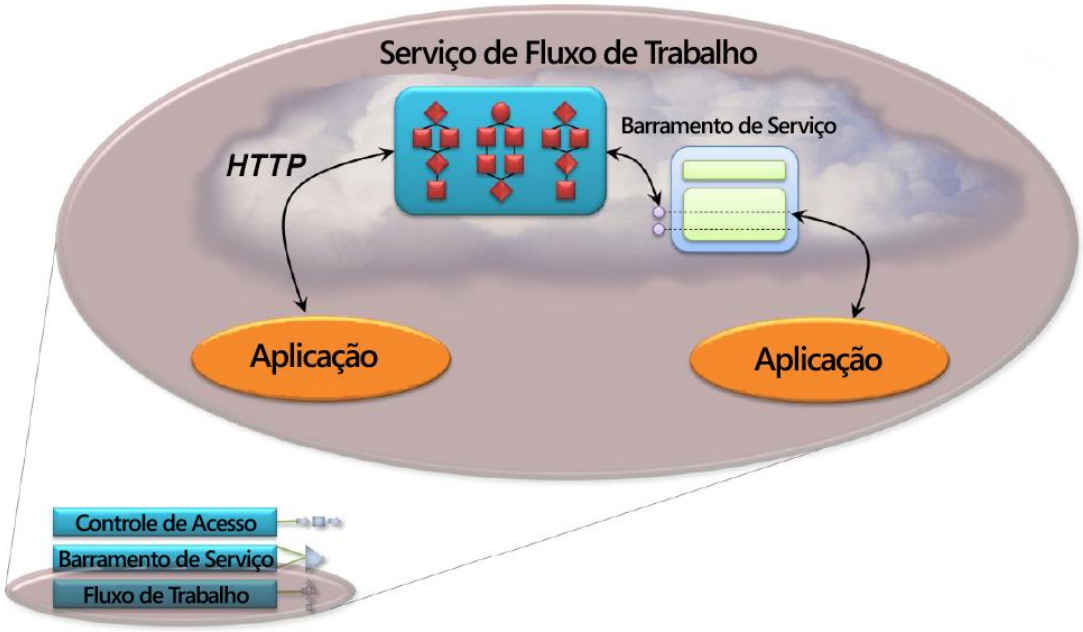

Figura 10: O Serviço de Fluxo de Trabalho permite a criação de aplicativos baseados no WF que podem comunicar-se usando HTTP ou o Barramento de Serviço.

Todo fluxo de trabalho do WF é implementado usando algumas atividades, mostradas em vermelho na figura. Cada atividade executa uma ação definida, como enviar ou receber uma mensagem, implementar uma instrução IF ou controlar um loop While. O WF fornece um conjunto padrão de atividades conhecido como

BAL (Base Activity Library) e o Serviço de Fluxo de Trabalho permite que os aplicativos que executa usem um subconjunto da BAL. O serviço também fornece diversas de suas próprias atividades. Por exemplo, os aplicativos que ele hospeda podem se comunicar com outros softwares usando HTTP ou o Barramento de Serviço, como a Figura 10 mostra, e assim o Serviço de Fluxo de Trabalho fornece atividades embutidas para fazer ambos. O Serviço de Fluxo de Trabalho também fornece atividades para trabalhar com mensagens XML, um requisito comum para integração de aplicativos.

A execução na nuvem traz algumas limitações, entretanto. Aplicativos baseados no WF executados no Serviço de Fluxo de trabalho podem usar apenas um modelo de fluxo de trabalho seqüencial do WF, por exemplo. Além disso, executar código arbitrário não é permitido, e assim nem a atividade de Código da BAL nem atividades personalizadas podem ser usadas.

Para criar aplicativos para o Serviço de Fluxo de Trabalho, desenvolvedores podem usar o designer de fluxo de trabalho de WF padrão do Visual Studio. Uma vez escritos, aplicativos baseados no WF podem ser implantados na nuvem usando um portal de Fluxo de Trabalho baseado em navegador ou programaticamente usando APIs fornecidas pelo Fluxo de Trabalho. Fluxos de trabalho em execução também podem ser gerenciados usando o portal ou essas APIs. E, como o Barramento de Serviço, aplicativos que interagem com o Serviço de Fluxo de Trabalho primeiro devem obter um token do Serviço de Controle de Acesso — o único STS confiável.

Aplicativos baseados no WF não representam a abordagem certa para tudo. Quando esse tipo de solução é necessário, porém, usar um fluxo de trabalho pode facilitar muito a vida do desenvolvedor. Ao fornecer uma maneira gerenciável e escalonável de hospedar aplicativos do WF na nuvem, o Serviço de Fluxo de Trabalho estende o alcance dessa tecnologia útil.

SQL SERVICES

SQL Services é um nome abrangente para o que será um grupo de tecnologias baseadas em nuvem. Todos eles enfocam trabalhar com dados: armazenar, analisar, criar relatórios a partir deles e mais. Supondo que funções de banco de dados básicas talvez sejam as mais fundamentais, o primeiro membro dessa família a surgir é o SQL Data Services.

Um banco de dados na nuvem é atraente por muitas razões. Para algumas organizações, permitir que um provedor de serviços especializado assegure confiabilidade, gerencie backups e execute funções de gerenciamento é interessante. Dados na nuvem também podem ser disponibilizados a aplicativos executados em qualquer parte, até mesmo em dispositivos móveis. E, devido às economias de escala que um provedor de serviços desfruta usar um banco de dados de nuvem pode bem ser mais barato que fazê-lo você mesmo. O objetivo do SQL Data Services é fornecer todos esses benefícios.

Contudo, implementar um banco de dados confiável e de alto desempenho com escalabilidade de Internet não é simples; são necessárias compensações. Como descrito anteriormente, por exemplo, o SQL Data Services não fornece um banco de dados relacional padrão, nem suporta consultas em SQL. Em vez disso, os dados são organizados usando a estrutura mostrada na Figura 11.

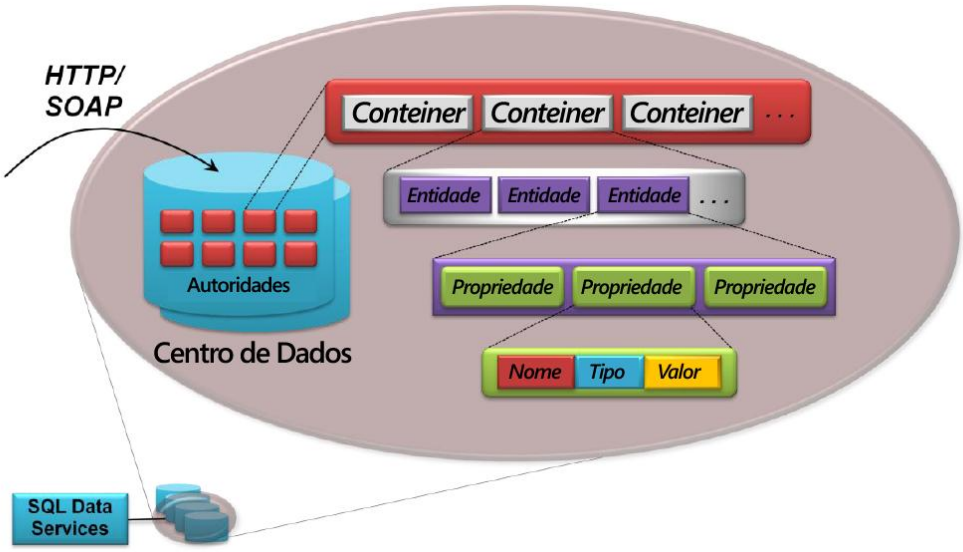

Figura 11: Um centro de dados do SQL Data Services é dividido em autoridades, cada uma das quais mantendo contêineres que, por sua vez, mantém entidades que contêm propriedades.

As informações no SQL Data Services são armazenadas em vários centros de dados. Cada centro de dados contém algumas autoridades, como mostra a Figura 11. Uma autoridade é a unidade de geolocalização — armazenada em um centro de dados Microsoft específico — e possui um nome DNS exclusivo. Uma autoridade mantém contêineres, cada um dos quais é replicado dentro de seu centro de dados. Contêineres são usados para balanceamento de carga e disponibilidade. Se ocorrer uma falha, o SQL Data Services começa a usar outra réplica do contêiner automaticamente. Toda consulta é emitida para um contêiner específico — consultas abrangendo a autoridade não são permitidas. Cada contêiner agrupa certo número de entidades, cada uma das quais, por sua vez, contendo propriedades. Cada propriedade tem um nome, tipo e um valor daquele tipo. Os tipos que o SQL Data Services suporta incluem String, Data/Hora, Base64Binary, Booleano e Decimal. Aplicativos também podem armazenar blobs com tipos MIME.

Para consultar esses dados, os aplicativos têm algumas opções. Uma é usar uma linguagem que seja muito similar à sintaxe C# de LINQ, com consultas enviadas via SOAP ou uma abordagem de REST. A outra é usar Data Services ADO.NET, uma maneira de REST alternativa para acessar dados. Em qualquer dos casos, aplicativos emitem consultas para contêineres usando operadores como ==, !=, <, >, AND, OR e NOT. Consultas também podem incluir algumas operações parecidas com SQL, como ORDER BY e JOIN.

Não importa como as consultas são emitidas, a unidade de recuperação e atualização são entidades, não propriedades. Uma consulta retorna certo número de entidades, por exemplo, inclusive todas as propriedades que contêm. Similarmente, não é possível atualizar apenas uma propriedade em uma entidade — a entidade inteira tem de ser substituída. E, como entidades não têm esquemas pré-definidos, as propriedades em uma única entidade podem ter tipos diferentes. As entidades em um contêiner também podem ser diferentes entre si, cada qual contendo diferentes tipos de propriedades.

Dados em SQL Data Services são nomeados com URIs, de forma muito parecida com o serviço de armazenamento do Windows Azure. O formato geral de um URI que identifica uma determinada entidade tem a aparência a seguir:

http://<*Authority*>.data.database.windows.net/v1/<*Container*>/<*Entity*>

Vale reiterar que nada a respeito do SQL Data Services exige um cliente baseado em .NET executado em Windows. Em vez disso, os dados que ele contém podem ser acessados via REST — ou seja, HTTP padrão — ou SOAP a partir de qualquer aplicativo executado em qualquer plataforma. Qualquer que seja a plataforma em que esteja sendo executado, um aplicativo que acesse dados deve identificar seu usuário com um nome de usuário e senha definidos pelo SQL Data Services ou com um token criado pelo STS do Serviço de Controle de Acesso.

Avançando, a Microsoft anunciou planos para evoluir o SQL Data Services em uma tecnologia mais relacional. Lembre-se de que, diferentemente do armazenamento do Windows Azure, o SQL Data Services se baseia no SQL Server, o que torna essa evolução mais natural. Entretanto, qualquer que seja o modelo que ele fornecer, o objetivo da tecnologia permanece o mesmo: fornecer um banco de dados de nuvem escalável, confiável e de baixo custo para todos os tipos de aplicativos. Conforme o SQL Services se expande para incluir mais serviços de dados baseados em nuvem, espere ver aqueles serviços contarem com esse primeiro membro da família.

LIVE SERVICES

O que motiva a criação de novas plataformas de aplicação? A resposta é mudança: mudanças em hardware, software e em como usamos aplicativos e dados. Telefones móveis transformaram-se em computadores, por exemplo, e servidores na nuvem se tornaram grandes partes de nossas vidas. Os aplicativos se tornaram mais pessoais, como os dados que armazenamos naqueles aplicativos. Combine essas mudanças e o palco está montado para um novo tipo de plataforma de aplicação.

O Live Services e o Live Framework exemplificam isso. Aplicativos podem usar o Live Framework para acessar dados do Live Services, e também podem contar com o Live Framework para sincronizar esses dados em estações de trabalho, laptops e dispositivos. A Figura 12 mostra como o Live Services e o Live Framework combinam.

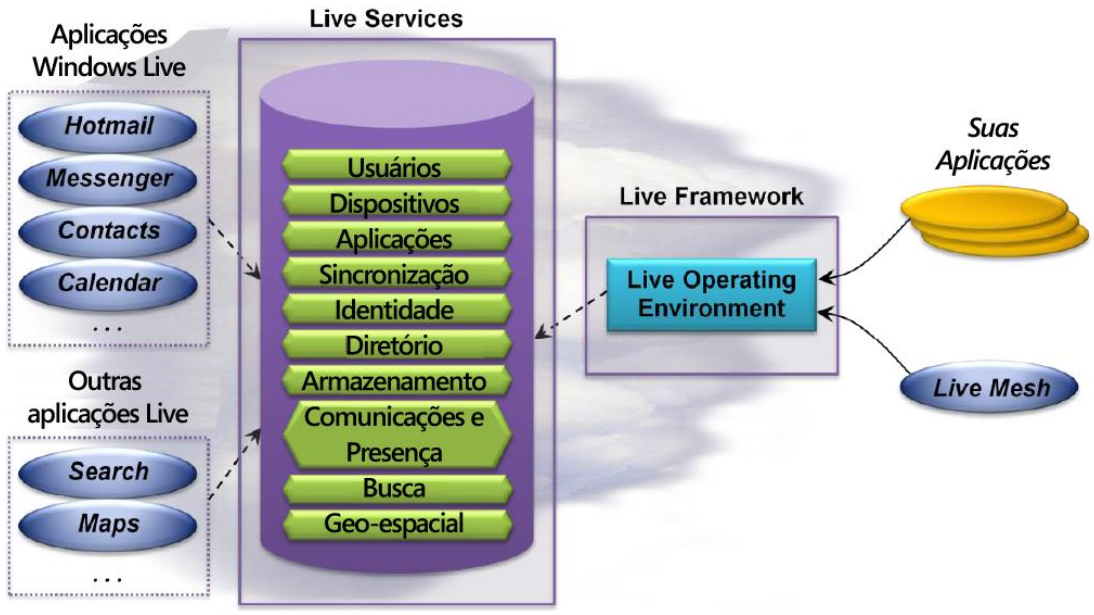

Figura 12: O Live Framework permite que aplicativos acessem dados do Live Services e mais.

O Live Services é desmembrado em diversas categorias, como a figura mostra. Cada serviço permite acesso a um conjunto particular de recursos, que podem ser específicos do usuário ou compartilhados. Por exemplo, a lista de contatos de um usuário é um recurso fornecido pelo serviço de Diretório, enquanto seu perfil é um recurso fornecido pelo serviço de Armazenamento. Ambos são serviços específicos de usuário, já que expõem dados associados com um determinado usuário. O serviço Geoespacial fornece recursos que contêm dados compartilhados, porém — mapas e outras informações geográficas — como faz o serviço de Busca.

Os dados no Live Services são usados por aplicativos Microsoft existentes de várias maneiras, como a figura indica. Um objetivo primário do Live Framework é facilitar a criação de novos aplicativos que usem esses dados. O Live Mesh da Microsoft é um exemplo disso, e ISVs e usuários finais têm a liberdade de criar outros. Todos esses aplicativos acessam dados através do componente primário do Live Framework: o Ambiente Operacional do Live. Sua aparência é descrita a seguir.

Acesso a Dados

A maneira mais simples de acessar dados do Live Services é diretamente através do Ambiente Operacional do Live. A Figura 13 mostra a aparência disso.

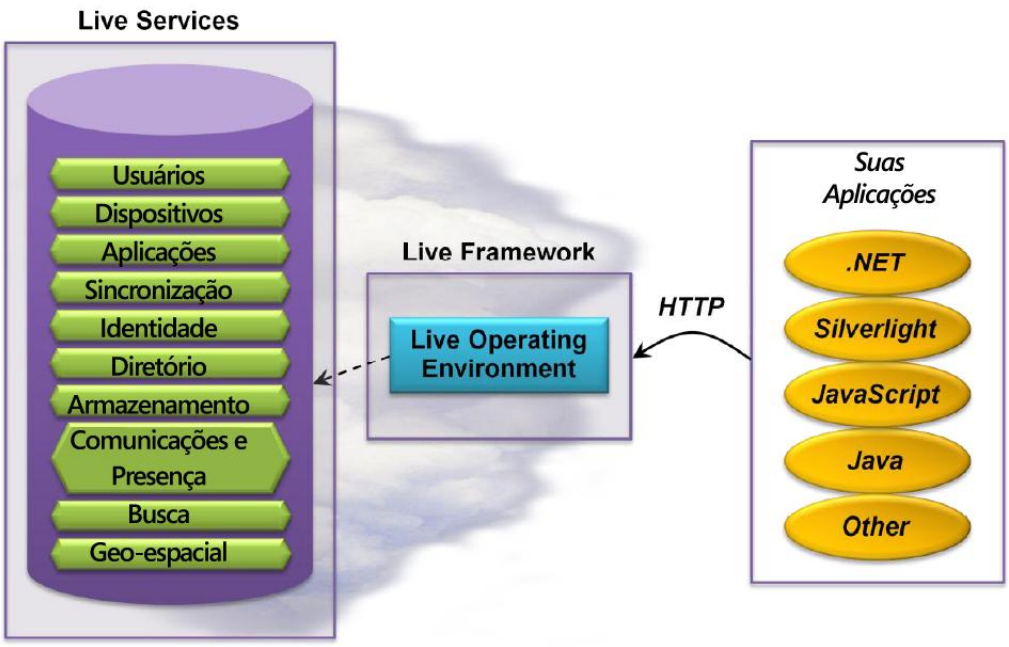

Figura 13: Como o Live Framework expõe dados do Live Services via HTTP, aplicativos escritos usando muitas tecnologias podem acessá-los.

Todos os recursos fornecidos pelo Live Services — ambos centrados no usuário e compartilhados — são nomeados com URIs. Para acessar essas informações, um aplicativo pode fazer solicitações de REST usando HTTP. Recursos também podem ser acessados via AtomPub ou outras maneiras baseadas em HTTP. Não importa como seja feito, as informações podem ser transferidas usando XML ou JSON, com dados de agregação transmitidos usando RSS ou Atom.

Para permitir uma abordagem consistente com a descrição e nomenclatura de dados do Live Services, o Live Framework define um modelo de recurso. Esse modelo especifica tipos e as relações permitidas entre instâncias daqueles tipos, junto com um esquema de nomenclatura de URI consistente. Aplicativos também podem criar tipos personalizados para personalizar seus próprios tipos de informações. O objetivo é fornecer semelhança suficiente para permitir que os aplicativos descubram e naveguem em dados do Live Services ao mesmo tempo em que dá aos desenvolvedores de aplicativos a flexibilidade de que precisam para armazenar informações diversificadas. E, como cada usuário possui controle detalhado sobre exatamente quais de seus recursos são expostos a que aplicativos e por quanto tempo, os dados pessoais de ninguém ficam livremente disponíveis.

Vale salientar que os dados usados por aplicativos do Live Microsoft são expostos hoje através de APIs do Live Services existentes (às vezes chamadas de Plataforma Windows Live). Essas APIs variam significativamente entre aplicativos, entretanto. Ao fornecer acesso comum e baseado em HTTP a todas essas informações, o Live Framework substituirá essa abordagem mais antiga por uma interface mais simples e consistente.

Para criar um aplicativo que acesse dados do Live Services através do Live Framework, um desenvolvedor pode escrever código usando uma interface HTTP bruta. Para facilitar isso, entretanto, o Live Framework também inclui Kits de Ferramenta do Live Framework. Essas bibliotecas fornecem uma abordagem mais simples e natural para os desenvolvedores construírem aplicativos que acessam o Live Services via Live Framework. A Microsoft fornece kits de ferramentas para o .NET Framework, Silverlight e JavaScript e outros provavelmente surgirão a partir da comunidade de programação. Mais uma vez, nada sobre a maneira como os dados são expostos pelo Live Framework os vincula a tecnologias Microsoft — Kits de Ferramentas do Live Framework podem ser criados para qualquer linguagem ou plataforma.

Uso do Mesh (Malha)

Contanto que tenha as permissões certas, qualquer aplicativo pode acessar dados do Live Services através do Live Framework. Opcionalmente, porém, um aplicativo pode estar sendo executado em um sistema que foi transformado em parte de uma malha. Se estiver, o aplicativo tem mais algumas opções.

Como descrito anteriormente, cada usuário pode ter sua própria malha contendo os sistemas que usa. Por exemplo, talvez ele tenha uma estação de trabalho Windows XP no trabalho, um Macintosh em casa, um laptop executando o Windows Vista e um telefone que executa o Windows Mobile. Todos esses sistemas podem ser agrupados em uma malha, como mostrado na Figura 14.

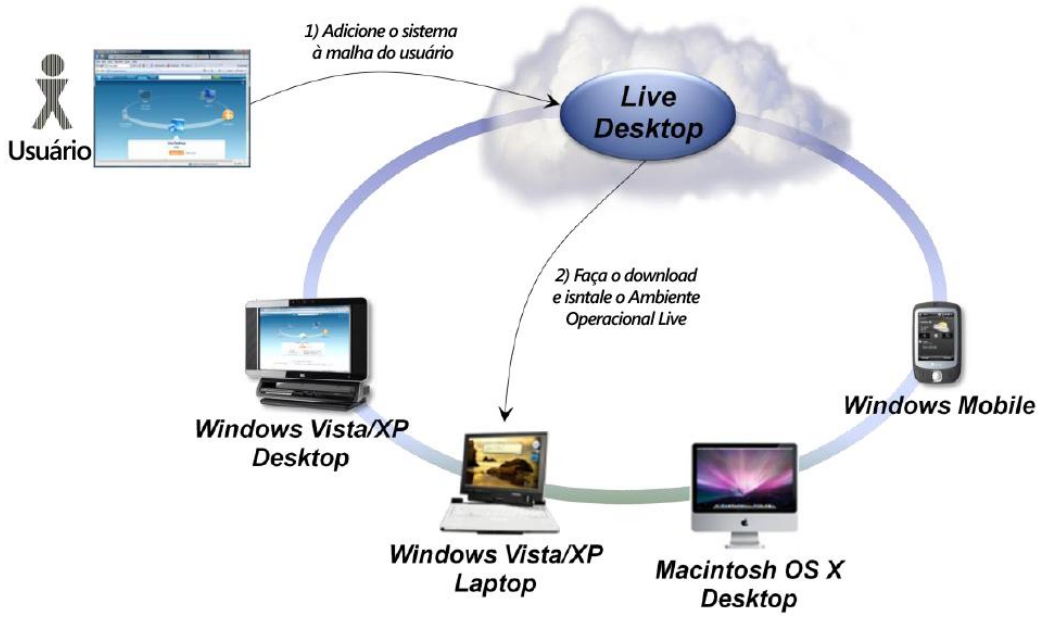

Figura 14: Incluir um sistema em uma rede instala o Ambiente Operacional do Live naquele sistema.

Para criar uma malha, o usuário pode assinar sua Live ID e depois acessar seu próprio Live Desktop através do navegador. Ele usa esse aplicativo baseado em nuvem para incluir sistemas em sua malha. Como ilustrado na Figura 14, o usuário especifica um sistema para incluir, que neste exemplo é seu laptop (etapa 1), e o Live Desktop o adiciona à malha. Para fazer isso, o Live Desktop na nuvem faz o download e instala uma cópia do Ambiente Operacional do Live naquela máquina (etapa 2).

Como descrito anteriormente, o Ambiente Operacional do Live permite que aplicativos acessem dados do Live Services via HTTP. Quando usado em uma malha, porém, esse componente também faz mais: Ele sincroniza os dados do Live Services do usuário na nuvem e em todos os sistemas da malha. A Figura 15 ilustra essa idéia.

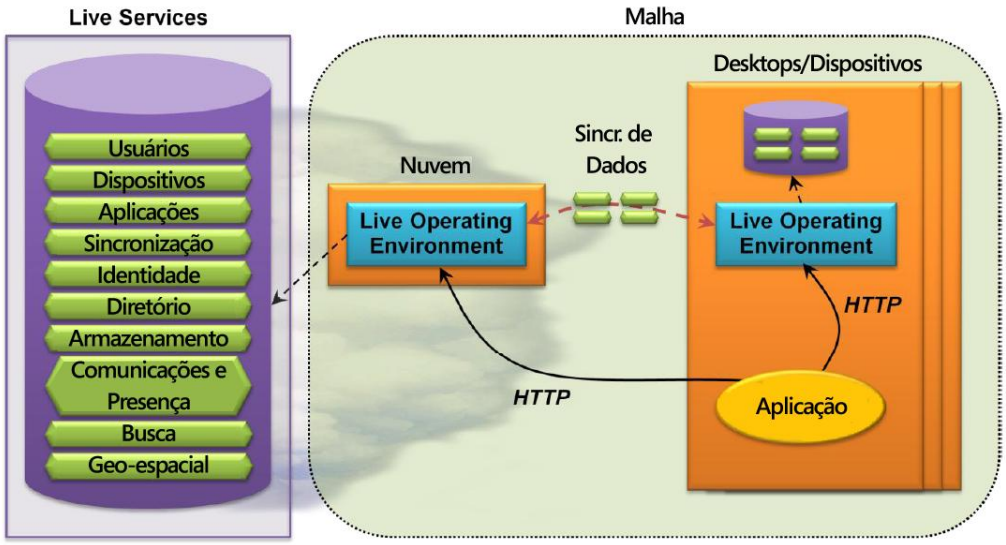

Figura 15: O Live Operating Environment (Ambiente Operacional do Live) mantém dados do Live Services sincronizados em estações de trabalho, dispositivos e na nuvem.

Usuários e aplicativos podem indicar quais dados devem ser incluídos na malha, e o Ambiente Operacional do Live cuida de manter aquelas informações sincronizadas. Por exemplo, o aplicativo Live Mesh da Microsoft permite ao usuário designar pastas específicas que devem fazer parte da malha. Uma vez feito isso, o Ambiente Operacional do Live propaga silenciosamente alterações feitas nos dados em qualquer daquelas pastas a todos os sistemas da malha. De forma similar dados de usuário do Live Services, como contatos e informações de perfil, podem ser mantidas em sincronia em toda a malha.

A sincronização da malha é multimestre, o que significa que um usuário pode alterar qualquer cópia das informações em qualquer dispositivo — não há apenas um mestre que deve ser atualizado. A tecnologia usada para fazer isso é FeedSync, um protocolo definido pela Microsoft e publicamente disponível que se baseia no HTTP. Sempre que possível os dados são sincronizados entre sistemas diretamente conectados — é ponto a ponto. Isso nem sempre é uma opção, entretanto, e, assim, um sistema também pode sincronizar-se com o Ambiente Operacional do Live na nuvem. Essa instância baseada na nuvem pode conectar-se diretamente a qualquer sistema na malha — é do mesmo nível de todos — e, portanto pode sincronizar-se com qualquer um deles.

Como sempre, um aplicativo em execução em um sistema habilitado para malha pode acessar dados fazendo solicitações HTTP para o Ambiente Operacional do Live na nuvem. E também tem acesso a uma cópia local de todos os dados do Live Services que tenham se tornado parte dessa malha. Em vez de interagir com a instância remota do Ambiente Operacional do Live, o aplicativo também pode emitir as mesmas solicitações HTTP para a instância em execução local, como mostra a Figura 15. Exceto pelo URI base, essas solicitações são idênticas para o Ambiente Operacional do Live local e na nuvem.

Essa simetria permite que um aplicativo trabalhe da mesma maneira com dados locais e com os armazenados na nuvem. Se um aplicativo estiver em execução em uma estação de trabalho ou um dispositivo desconectado no momento, por exemplo, ele pode acessar a cópia local, que atua como um cache para o último estado conhecido dos dados da nuvem. Quando o serviço é reconectado, o aplicativo pode acessar os dados da nuvem diretamente — só é necessário mudar o URI — ou esperar que a cópia local dos dados seja atualizada pela sincronização do Ambiente Operacional do Live.

Sistemas que não são executados no Ambiente Operacional do Live também podem participar de uma malha, embora de uma maneira mais limitada. Como o Live Desktop pode ser acessado usando qualquer navegador, um usuário executando, digamos, um sistema Linux pode usá-lo para criar uma malha com apenas um componente de nuvem — a malha não contém estações de trabalho ou dispositivos. Aplicativos executados no sistema Linux podem armazenar e acessar dados nessa malha simples como faz com outros dados do Live Services: usando HTTP. De fato, um aplicativo pode até mesmo implementar o protocolo FeedSync para sincronizar esses dados de nuvem com uma cópia local. Embora sistemas executados no Ambiente Operacional do Live tenham mais recursos, os que não têm também podem achar útil esse aspecto do Live Framework.

Aplicativos de Web Compatíveis com a Malha

Qualquer aplicativo, baseado em Windows ou não, pode acessar dados do Live Services — não precisa ser parte de uma malha. Se um desenvolvedor estiver construindo um aplicativo que será sempre executado em sistemas de malha, porém, há outra opção. Ele pode criar um aplicativo de Web habilitado para malha que pode ser distribuído e gerenciado pelo próprio Live Framework. A Figura 16 mostra o básico de como isso funciona.

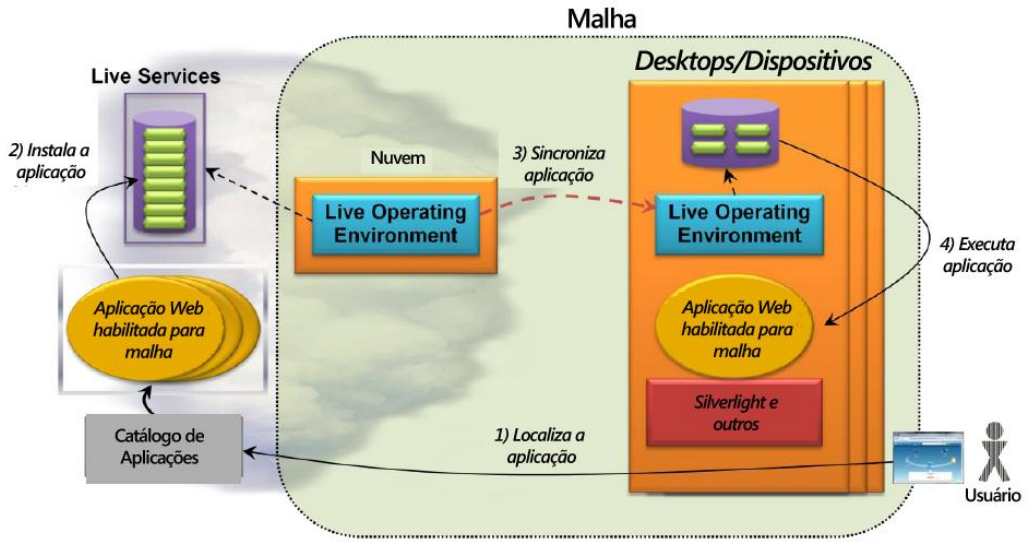

Figura 16: O usuário pode descobrir um aplicativo de Web habilitado para malha e depois instalá-lo na sua malha.

Como a figura mostra, um aplicativo de Web compatível com malha pode ser disponibilizado através de um catálogo de aplicativos fornecido pela Microsoft na nuvem. Um usuário pode acessar esse catálogo para descobrir aplicativos de Web compatíveis com malha (etapa 1). Uma vez escolhido o aplicativo, o usuário pode instalá-lo (etapa 2). Inicialmente, isso apenas copia o aplicativo para o armazenamento da nuvem do usuário no Live Services. O aplicativo será sincronizado com as estações de trabalho e dispositivos do usuário, porém, como outros dados de nuvem (etapa 3). O aplicativo agora pode ser executado a partir do armazenamento local em qualquer sistema na malha desse usuário (etapa 4). De fato, não é exato pensar em um aplicativo de Web habilitado para malha como sendo instalado em apenas um sistema. Em vez disso, o aplicativo é instalado em todos eles — é instalado na malha.

Um aplicativo de Web habilitado para malha deve ser implementado usando uma tecnologia multiplataforma, como o Microsoft Silverlight, DHTML ou Adobe Flash. Essas tecnologias têm suporte em todos os sistemas operacionais que podem executar o Live Framework: Windows Vista/XP, Macintosh OS X e Windows Mobile 6. Assim, qualquer aplicativo de Web habilitado para malha pode ser executado em qualquer sistema na malha (embora essas opções não tenham suporte na CTP de Novembro do Live Framework).

Como o Ambiente Operacional do Live mantém todos os dados de malha em sincronia, um aplicativo de Web habilitado para malha verá os mesmos dados, não importa onde estiver em execução. Isso dá um novo significado interessante à noção de gravar uma vez, executar em qualquer lugar: Um aplicativo de Web habilitado para malha pode ser executado sem alteração em qualquer sistema dentro de uma malha e também pode contar com os dados disponíveis, não importando onde estiver sendo executado.

Como com qualquer outro tipo de acesso aos dados do Live Framework, um aplicativo de Web habilitado para malha tem acesso somente aos dados com que um usuário o tenha autorizado especificamente para trabalhar. E, como outros aplicativos do Silverlight, DHTML e Flash, um aplicativo de Web habilitado para malha é executado em uma área de segurança. Salvo especificamente permitido pelo usuário, esses aplicativos não acessam diretamente o disco local ou os dados de outros aplicativos de Web habilitados para malha. Contudo, o usuário tem a liberdade de compartilhar um aplicativo de Web habilitado para malha com a malha de outro usuário. Por exemplo, um usuário pode dizer a um aplicativo de Web habilitado para malha para convidar todos em seu catálogo de endereços a usá-lo. Como suas informações de contatos estão diretamente disponíveis ao aplicativo — faz parte de sua malha — mandar o aplicativo fazer isso é simples.

Para ajudar os desenvolvedores a criar aplicativos de Web habilitados para malha, a Microsoft fornece modelos de projeto para Visual Studio 2008. Para facilitar a atualização desses aplicativos, um desenvolvedor pode carregar uma nova versão no catálogo de aplicativos e depois deixar o Live Framework cuidar automaticamente da atualização do aplicativo na malha de todos os usuários que o instalarem. E para ajudar os desenvolvedores a ganhar dinheiro com seus aplicativos, a Microsoft planeja permitir a conectar seu próprio adCenter ou um serviço de terceiros para permitir que um aplicativo de Web habilitado para malha mostre anúncios.

Dessa forma, o Live Framework é um tipo totalmente novo de plataforma de aplicação. Muitos aspectos do ambiente, como acesso a dados do Live Services e o foco em estações de trabalho e dispositivos, tornam claro que a meta principal dessa tecnologia é dar suporte a aplicativos orientados ao consumidor e socialmente compatíveis. Em um sentido bastante real, o Live Framework está no cruzamento de nova tecnologia e de novos tipos de interação humana.

CONCLUSÕES

A verdade é evidente: A computação na nuvem está aqui. Para desenvolvedores, tirar proveito da nuvem significa usar plataformas de nuvem de alguma forma. Com a Plataforma de Serviços Azure, a Microsoft apresenta uma variedade de estilos de plataforma tratando de uma variedade de necessidades:

- O Windows Azure fornece computação e ambiente de armazenamento baseados em Windows na nuvem.

- O .NET Services oferece infra-estrutura baseada em nuvem para aplicativos na nuvem e locais (on-

premises).

- O SQL Services fornece um banco de dados na nuvem hoje através do SQL Data Services, com mais serviços de dados baseados na nuvem planejados.
- O Live Services fornece o Live Framework, que permite que aplicativos acessem dados do Live Services, sincronizem dados em sistemas unidos em uma malha e muito mais.

Essas quatro abordagens lidam com uma variedade de requisitos, e a maioria dos desenvolvedores provavelmente não usará todos eles. Contudo, quer você trabalhe para um ISV ou uma empresa, alguns serviços de plataforma de nuvem provavelmente serão úteis para aplicativos que sua organização cria. Um novo mundo está despontando.

SOBRE O AUTOR

David Chappell é Diretor da Chappell & Associates (www.davidchappell.com) em São Francisco, Califórnia. Através de suas palestras, colunas e consultoria, ele ajuda pessoas ao redor do mundo a entender, usar e tomar decisões melhores a respeito de novas tecnologias.