FACULDADE DE TECNOLOGIA DE SÃO PAULO

Denis Motta Urbanavicius

Software as a Service

São Paulo

2013

FACULDADE DE TECNOLOGIA DE SÃO PAULO

Denis Motta Urbanavicius

Software as a Service

Monografia submetida como exigência parcial para a obtenção do Grau de Tecnólogo em Processamento de Dados
Orientador: Prof. Valter Yogui

São Paulo

2013

A meus pais que me ensinaram a não desistir de meus sonhos.

AGRADECIMENTOS

A minha família, que sempre acreditou em mim e me incentivou até o fim.

A todos os professores por terem me ensinado a ir atrás do conhecimento e não esperar por ele. Agradeço principalmente a meu orientador, Valter Yogui, pela paciência e compreensão.

Resumo

Este documento visa definir Software as a Service (SaaS), suas diferenças entre os softwares tradicionais, suas vantagens, desvantagens e seu modelo de negócio como um todo.

Hoje, a maioria das empresas usa algum tipo de software de gerenciamento, seja ele em modelo SaaS ou no modelo tradicional. Cada vez mais clientes de Softwares querem diminuir seus custos, sejam eles com o software em si, sejam eles com a infraestrutura necessária para manter esse Software. Aqui mostro o que aprendi com minha pesquisa e com a minha vivência desses dois modelos de software.

Abstract

The objective of this document is to define Software as a Service (SaaS), its differences among traditional softwares, advantages, disadvantages and its whole business model.

Today, most companies use some kind of management software, be it in SaaS model or traditional model. Increasingly, software customers want to reduce their costs, whether the software itself, whether the infrastructure needed to maintain that software. Here I show what I learned from my research and my experience of these two software models.

LISTA DE QUADROS E FIGURAS

SUMÁRIO

1. Histórico

A hospedagem centralizada de aplicações surgiu na década de 1960. No começo dessa década, a IBM e outros provedores de mainframes começaram a fornecer serviços pagos, referindo-os muitas vezes como Time-sharing[1]. Estes serviços ofereciam poder computacional e armazenamento de dados de seus data centers para bancos e outras grandes organizações.

A expansão da internet durante a década de 1990 trouxe consigo uma nova classe de computação centralizada, chamada Application Service Providers (ASP). O ASP servia negócios com o serviço de hospedagem e administração de aplicações especializadas, com o objetivo de reduzir custos via administração central e por meio da especialização do fornecedor de soluções em uma aplicação de negócio particular.

O modelo Software as a Service (que será tratado aqui por seu nome em tradução livre: Software como um Serviço) é essencialmente uma extensão da ideia do modelo ASP. O termo Software as a Service, porém, é comumente usado em mais situações específicas:

- Inicialmente, a maioria dos provedores de aplicações-serviço focava em administrar e servir softwares de terceiros (vendedores independentes). Atualmente, a maioria dos vendedores de Software como um Serviço desenvolve e administra seu próprio software;

[1]Time-sharing é o compartilhamento de recursos computacionais por vários usuários, via uso de multiprogramação e multi-tasking (processamento de várias tarefas em concorrência).

- Inicialmente, muitos provedores de aplicações-serviço ofereciam aplicações cliente-servidor mais tradicionais, os quais precisavam ser instalados nos computadores pessoais dos clientes (usuários). Atualmente, as soluções de Software como Serviço são predominantemente baseadas na web, precisando apenas de uma conexão com a internet para serem utilizados;

- Enquanto a arquitetura de software usada pela maioria das aplicações-serviço iniciais obrigava manter uma instância separada da aplicação para cada negócio, os Softwares como Serviço contemporâneos utilizam uma arquitetura multi-tenant, a qual irei comentar a frente.

O acrônimo SAAS é tido como aparecido pela primeira vez em um artigo chamado "Strategic Backgrounder: Software As A Service", publicado internamente em fevereiro de 2001 pela Software& Information Industry's (SIIA) eBusiness Division. Sua versão popular (SaaS) surgiu em março de 2005, em uma conferência do Fórum SD, por John Koenig, sendo adotado em seguida pela SalesForce.com, que usou por muitos anos o termo "OnDemand".

O Software como um Serviço, comumente referido como o modelo de Aplicação Provedora de Serviço (ASP – da sigla em inglês), é anunciado por muitos como a nova onda na distribuição de softwares aplicativos. Seguindo a máxima de que "a Internet muda tudo", muitos acreditam que os pacotes de aplicações empresariais e desktop vendidos tradicionalmente serão logo varridos pela onda do "Web-based", produtos e serviços terceirizados que removem a responsabilidade para instalação, manutenção e atualização de funcionários de gestão de sistemas de informação sobrecarregados. Alguns analistas e membros da indústria acreditam que os pacotes de software, como uma entidade separada, irão deixar de existir. Enquanto essas previsões drásticas ainda

não aconteceram, devido a questões técnicas e de negócios, o espirito dessa mudança – a entrega, gestão e pagamento de software como um serviço ao invés de como um produto – está afetando todos os participantes da indústria de software. (SIIA, 2001)

2. Modelos de cobrança

No modelo de Software como um Serviço, a aplicação, ou serviço, é disponibilizado de um *data center* centralizado sobre uma rede – Internet, Intranet, LAN ou VPN – fornecendo acesso e uso por uma taxa recorrente. Usuários "alugam", "assinam", "são consignados a" ou "recebem acesso a" as aplicações por um provedor central. Os modelos de negócios variam de acordo com o nível a qual o software é otimizado, para menor preço e eficiência maior, ou valor agregado via customização para aperfeiçoar ainda mais processos de negócios digitalizados. (SIIA, 2001)

Diferentemente dos softwares tradicionais, vendidos convencionalmente como uma licença perpétua associada a uma taxa única (e, tipicamente, pequenas taxas de suporte), os provedores de SaaS geralmente cobram pelo uso das aplicações usando taxas de assinatura, sendo, na maioria das vezes, uma taxa mensal ou anual. Consequentemente, o custo inicial de configuração para SaaS é tipicamente menor que o software empresarial equivalente. Os vendedores de SaaS costumam precificar suas aplicações baseando-se em alguns parâmetros de uso, como o número de usuários usando a aplicação. No entanto, como os dados dos clientes do ambiente SaaS ficam armazenados com o provedor, existe a oportunidade de cobrar também por transação, evento ou outra unidade de valor.

O custo relativamente baixo de provisionamento de usuário (por exemplo: configurar um novo usuário/cliente) em um ambiente multi-tenant possibilita alguns vendedores de SaaS de oferecer aplicações usando o modelo freemium. Neste modelo, um serviço grátis é liberado com funcionalidade ou escopo limitados e são cobradas taxas para melhores funcionalidades ou maior escopo. Algumas outras aplicações SaaS são completamente grátis para usuários, com a receita sendo derivada de fontes alternativas assim como propagandas.

Um ponto chave do crescimento do SaaS é a possibilidade dos vendedores de SaaS servirem um preço competitivo com softwares tradicionais. Isto é consistente com a lógica tradicional de terceirização de sistemas de TI, que envolve a aplicação de economias de escala para a operação da aplicação, ou seja, um provedor de serviços externos pode ser capaz de oferecer aplicações mais confiáveis, melhores e mais baratas.

Abaixo, segue um quadro com as principais diferenças de custos para o cliente, comparando-se os Softwares tradicionais e os Softwares como Serviço.

	Softwares tradicionais	Software como um Serviço
Custo de Instalação	Alto, sendo essa, muitas vezes, grande parte da receita da empresa desenvolvedora do software.	Baixo, sendo, em grande parte dos casos, zero.
Custo de infraestrutura	Variável. Pode ser extremamente alto em sistemas de missão crítica	Extremamente baixo. Em todos os casos é necessário ter uma conexão com a

	e/ou complexos, ou pode ser relativamente baixo em sistemas mais simples.	internet. Na maioria dos casos este é o único requisito.
Cobranças periódicas	Baixo. Aqui existe uma certa divisão entre os sistemas que cobram taxas mensais/anuais de manutenção e/ou suporte e os que não cobram por esses serviços.	Médio/Alto. Esta é, na maioria dos casos, a única fonte de renda da empresa desenvolvedora do Software como um Serviço. Aqui ficam inclusos tanto os custos de suporte e manutenção quanto os custos de desenvolvimento e infraestrutura.

Tabela 1: Principais diferenças de custos para o cliente.

Mesmo que o modelo de Software como um Serviço possa parecer mais caro a longo prazo–e em alguns casos realmente é – o cliente não tem a preocupação (e os gastos) de manter uma infraestrutura atualizada, além de não ter de investir uma quantia razoável adquirindo novas licenças do Software em caso de uma eventual expansão.

Abaixo, falarei sobre algumas das estratégias de cobrança mais populares, com alguns exemplos de empresas que os usam.

2.1. Grátis

Atualmente não existe um modelo grátis se tratando de aplicações SaaS, ao contrário de aplicações desktop ou distribuições baseadas na Web. O modelo Grátis é na verdade um modelo suportado por publicidade ou algo que ajude uma empresa

alavancar/popularizar outra marca ou produto. YouTube é um bom exemplo que cai neste modelo. Nenhum usuário paga para enviar ou consumir os vídeos. A receita gerada para o negócio é feita usando anúncios. Outros serviços que seguem modelos similares são sites populares de redes sociais como Facebook, Twitter e Orkut. (Shalin, 2010).

O modelo gratuito pode crescer rapidamente e é necessário que se atinja um grande número de usuários para que faça sentido usar um modelo baseado de publicidade. Poderá ser necessário um grande investimento em infraestrutura para que a mesma possa suportar um grande número de usuários antes mesmo da receita dos anúncios começar a aparecer. Esse modelo funciona bem para aplicações voltadas a consumidores, mas pode não funcionar para aplicações voltadas a negócios.

As Aplicações SaaS gratuitas mais bem-sucedidas tem sua própria rede de anúncios, para evitar que a receita seja dividida com um provedor de rede de anúncios, o que torna a sobrevivência do negócio bem mais difícil.

Figura 1: Facebook, serviço de rede social.

2.2. Freemium

Este é provavelmente o modelo de cobrança mais popular para aplicações SaaS. Existem várias aplicações tanto para consumidores quanto para negócios que usam esse modelo efetivamente. A ideia do Freemium é de oferecer um plano grátis junto com planos pagos. Os planos pagos normalmente oferecem alguns benefícios em relação aos planos gratuitos. Os planos gratuitos tornam-se um ótimo ponto de inicio para os usuários. (Shalin, 2010).

O Flickr é um bom exemplo que segue este modelo. O usuário pode enviar imagens e compartilhá-las usando um plano gratuito, com um limite de 1TB de armazenamento, com visualizações ilimitadas de suas imagens, porém com a possibilidade de serem colocados anúncios na página. Se o usuário quiser, pode optar por um plano Ad-Free que, por uma taxa anual de US$ 49,99, remove todos os anúncios de sua página. Se o

seu uso for maior que o limite inicial, o usuário pode mudar para o plano “Doublr”, que tem uma taxa anual de US$499,99, com limite de armazenamento de 2TB.

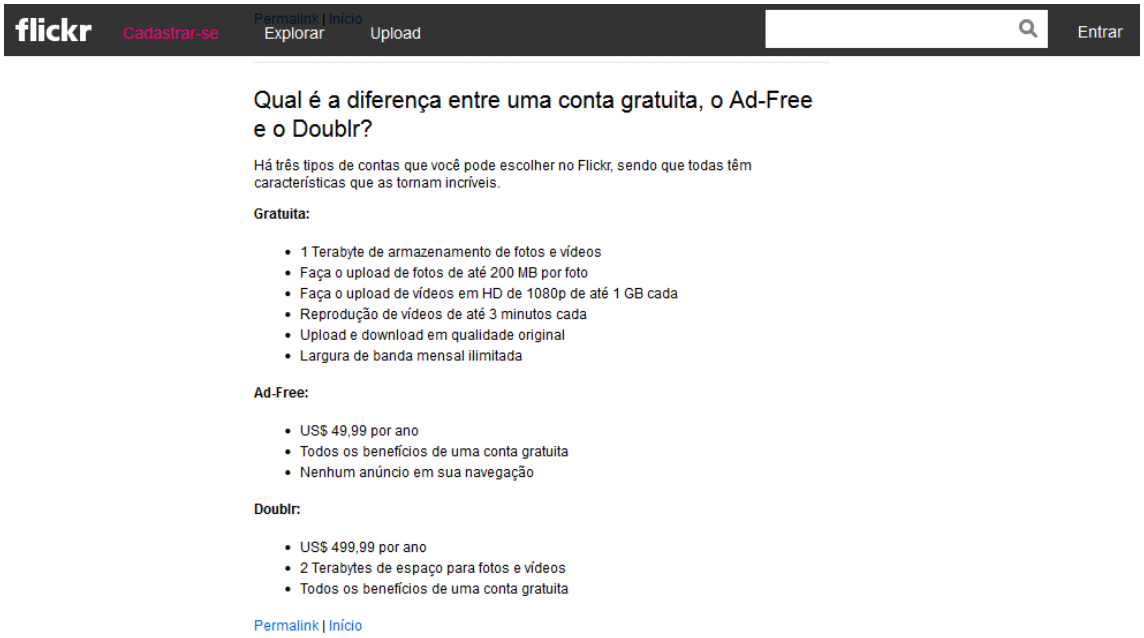


Figura 2: Os planos oferecidos pelo Flickr.

Planos “premium” normalmente oferecem mais funcionalidades, maiores limites, suporte prioritário e às vezes até mais seguranças dos dados, com cópias de segurança (backups). Geralmente os planos gratuitos são oferecidos de forma que atraia novos usuários, que podem futuramente optar por um plano pago.

2.3. Trial (Versão/Cópia de avaliação)

Teste antes de comprar. Um modelo de cobrança mais sério, feito para negócios. As aplicações são normalmente disponíveis livremente entre 7 e 60 dias para teste, porém com todas suas funcionalidades. O MobileMe e o Hubspot da Apple e são bons exemplos que usam esse modelo efetivamente (Shalin, 2010).

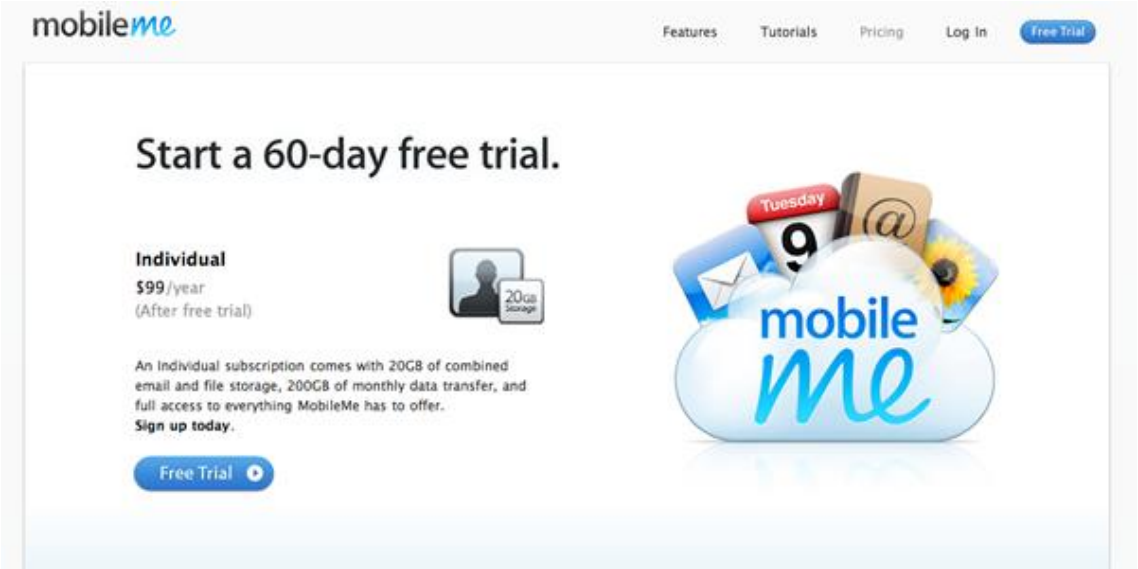


Figura 3: O serviço Mobile Me da Apple, que oferece Trial de 60 dias.

Pode ser necessário que o produto que irá usar esse modelo tenha concorrência limitada, ou seja, reconhecido como uma aplicação extremamente útil, para que essa versão de teste acabe atraindo o usuário para a compra do produto. Muitos serviços exigem que o usuário informe seus dados do cartão de crédito para o uso da versão de testes e começam a cobrar assim que o período de teste termina, mesmo que isso possa assustar novos clientes, de modo a diminuir o número de usuários gratuitos que não tem intenção de comprar o produto.

2.4. Quanto cobrar?

Depois de definido o modelo de cobrança a ser utilizado, vem a questão do valor a ser cobrado pelo serviço. Apesar de esse valor depender de muitas variáveis (público alvo, concorrência, valor que o cliente vê no serviço) e pesquisas de mercado, podemos ainda fazer uma escolha entre um plano único ou múltiplos planos.

2.5. Plano único

Evernote, Flickr e MobileMe usam uma estratégia de plano único “Premium”. Eles vendem essencialmente mais espaço de armazenamento e largura de banda, focado em atender consumidores ao invés de negócios. Seus produtos limitam um usuário por conta e fixam um preço único.

2.6. Planos múltiplos

É importante avaliar se o seu mercado pode ser segmentado. Por exemplo – pequenas, médias e grandes empresas, fornecedores de programas proprietários e de código aberto, freelancers iniciantes e Freelances já estabilizados. O objetivo dos planos múltiplos de cobrança é de cobrar mais dos clientes que usam mais seu serviço e vice-versa. Por exemplo, se você está vendendo uma solução de administração de projetos, você pode facilmente segmentar seu público alvo tanto no tamanho da empresa ou na quantidade de projetos. Negócios normalmente irão experimentar com o plano mais básico antes de pensar que a solução funciona para a empresa inteira. (Shalin, 2010).

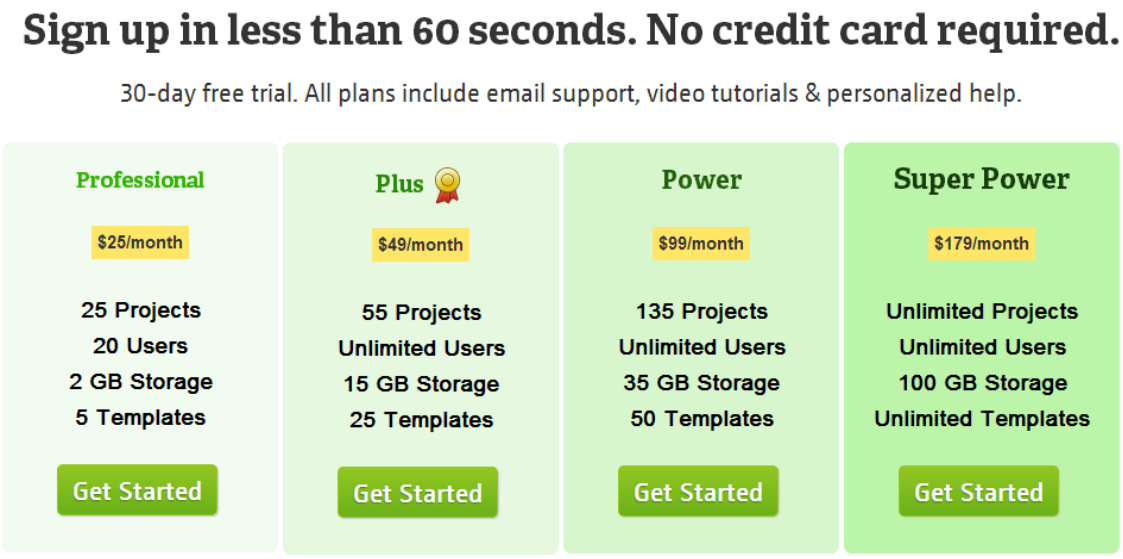


Figura 4: Modelo de cobrança da DeskAway, com múltiplos planos e opção de Trial.

3. Arquitetura

Como os dados dos clientes ficam armazenados com o fornecedor do software nas aplicações SaaS, existe a necessidade do cliente confiar no fornecedor de soluções SaaS para manter seus dados seguros. Para adquirir essa confiança, criar uma arquitetura de dados que seja robusta e segura o bastante se torna uma das mais altas prioridades.

A grande maioria das soluções de SaaS são baseadas na arquitetura multi-tenant. Neste modelo, uma versão única da aplicação, com uma única configuração (hardware, rede, sistema operacional) é usada por todos os clientes. Para suportar escalabilidade, a aplicação é instalada em várias máquinas (chamado escalonamento horizontal). Ao se decidir por fazer uma arquitetura multi-tenant, deve se definir qual será o modelo de dados a ser usado, usando uma arquitetura mais compartilhada ou mais isolada. Mesmo que vários clientes possam acessar o sistema ao mesmo tempo, é possível usar bancos de dados separados para cada cliente, tornando o acesso de dados mais isolado. Em contraste a isso, pode ser usado apenas um banco de dados para todos os clientes, separando os dados via esquemas ou campos de controle (neste último caso demandando mais cuidado ao desenvolver a aplicação, de modo a evitar que mesmo falhas graves não façam um cliente acessar dados que não o pertencem).

Em alguns casos, uma segunda versão da aplicação é configurada para atender um seleto grupo de clientes com acesso a versões pré-lançamento (por exemplo: uma versão beta) para testes. Isso é contrastado com software tradicional, onde várias cópias físicas do software – cada uma com uma versão potencialmente diferente, com uma configuração potencialmente diferente e comumente customizado – são instaladas por vários ambientes de clientes.

Mesmo sendo uma exceção, algumas soluções SaaS não usam multi-tenancy, ou usam outros mecanismos – como a virtualização – para gerenciar um grande número de

clientes a custos reduzidos. É um tema comumente discutido se o multi-tenancy é um componente necessário para o Software como um Serviço.

O uso de arquiteturas para múltiplos clientes é essencial para reduzir os custos de desenvolvimento, implantação e manutenção da aplicação SaaS e torná-la competitiva no mercado. Sem essas arquiteturas, seria impossível fazer economia de escalas em soluções SaaS.

É muito comum o uso de empresas terceirizadas para o aluguel de infraestrutura (servidores) de aplicações SaaS, de forma a diminuir os custos e eliminar a preocupação com a disponibilidade da infraestrutura do sistema.

Pelo fato de uma única aplicação atender vários clientes ao mesmo tempo, é exigido um cuidado maior no desenvolvimento do software, a fim de diminuir a probabilidade de erros causados por stress excessivo da aplicação. Dessa forma, a aplicação pode ser dividida em instâncias, de modo a dividir o stress causado pela quantidade de usuários acessando o sistema simultaneamente.

A otimização para uma maior eficiência num ambiente de compartilhamento não pode comprometer o nível de segurança e proteção no acesso aos dados. Os padrões de segurança listados abaixo demonstram como você pode criar uma aplicação com "isolamento virtual" através de mecanismos como permissões, views do SQL e criptografia. (Biblioteca MSDN, 2007)

3.1. Multi-Tenancy

A Confiança, ou a falta dela, é a principal razão para a não adoção do Software como um Serviço (Software as a Service - SaaS). (Biblioteca MSDN, 2006)

Para se adquirir essa confiança, uma das mais altas prioridades para o futuro da Arquitetura SaaS é a criação de uma arquitetura de dados que seja robusta e segura o bastante para satisfazer parceiros e clientes - preocupados com um eficiente controle dos dados empresariais (vitais para terceiros), ao mesmo tempo em que estes possuam uma boa relação de custo / benefício no gerenciamento e administração. (Biblioteca MSDN, 2006)

A biblioteca MSDN cita três abordagens distintas para a criação de uma aplicação multi-tenant (múltiplos inquilinos), onde os dados podem ser isolados ou compartilhados. Porém, esta distinção não é binária, podendo haver variações entre os dois extemos.

As abordagens estão divididas por nível de isolamento, sendo, do mais isolado para o mais compartilhado: banco de dados separado, esquema separado e esquema compartilhado.

3.2. Banco de dados separados

Este é o modo mais simples de se obter um modelo isolado. Os recursos da infraestrutura e da aplicação são geralmente compartilhados entre todos os clientes de um mesmo servidor, porém com cada cliente tendo acesso apenas a suas próprias informações, que ficam isoladas (de maneira lógica) dos dados dos outros clientes. A aplicação faz a associação de cada um dos bancos de dados com o cliente correto, evitando que um cliente acesse os dados de outro cliente.

Dessa forma, fica mais fácil fazer customizações para cada cliente, além de facilitar a restauração de dados em casos de falhas. Esta abordagem, porém, gera custos mais altos no gerenciamento de equipamentos e backup de dados, já que é necessário que se faça backup de cada banco de dados separadamente. Além disso, caso o número de clientes aumente muito, os custos de hardware podem aumentar, caso haja algum limite de bancos de dados suportados.

Esta abordagem é recomendada, pois há menor preocupação com a segurança dos dados e os clientes que precisam de um sistema mais seguro aceitam pagar mais caro por isso.

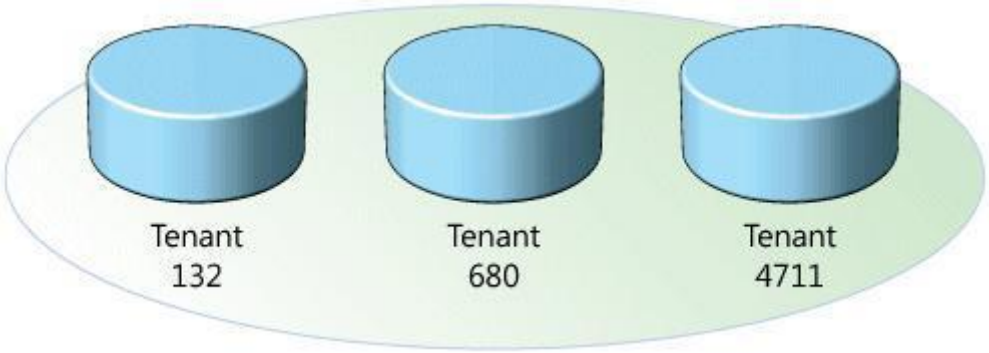


Figura 5: Esta abordagem usa um banco de dados distinto para cada cliente.

3.3. Banco de dados compartilhado, esquemas separados

Esta abordagem envolve o armazenamento das informações de vários clientes em um único banco de dados, com cada cliente possuindo seu próprio conjunto de tabelas em um esquema criado especificamente para cada cliente.

Da mesma forma que o método isolado citado anteriormente, essa separação de esquemas é relativamente fácil de ser implementada e os clientes ainda podem fazer customizações, desde que estendidas do modelo padrão. Essa abordagem oferece um isolamento moderado e pode suportar um grande número de clientes por base de dados.

Um ponto fraco desse modelo é a restauração de dados em caso de falha. Como os dados de vários clientes estão em uma única base de dados, restaurar os dados de um único cliente pode significar a restauração de todo o banco de dados a partir de um backup, sobrepondo dados de todos os clientes daquele banco de dados, independente de terem dados corrompidos ou não. Para contornar esse problema, o banco de dados de backup pode ser restaurado em um servidor temporário, para que as tabelas do único cliente que teve seus dados corrompidos possam ser sobrepostas – uma operação perigosa e demorada.

Essa abordagem é apropriada para aplicações que usem pequeno número de tabelas no banco de dados, podendo assim acomodar mais clientes por servidor – em relação ao modelo isolado – oferecendo assim uma aplicação a um custo menor.

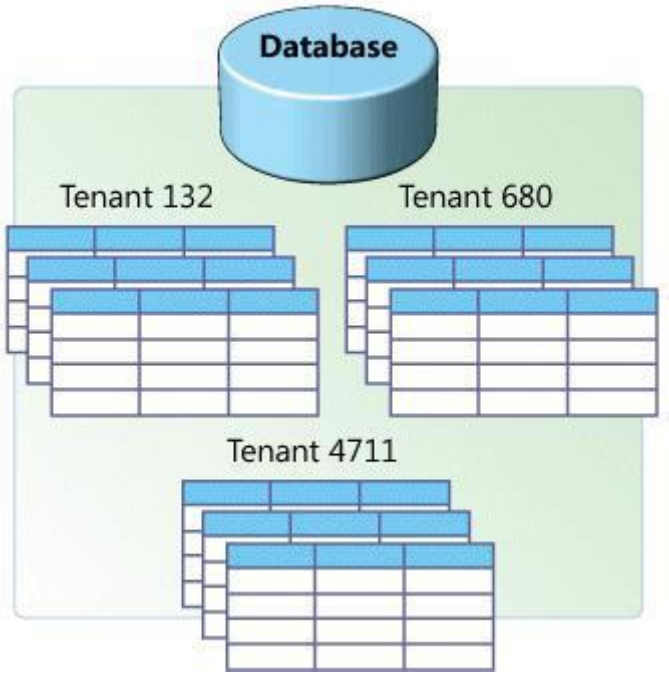


Figura 6: Nesta abordagem, cada cliente possui seu próprio conjunto de tabelas numa mesma base de dados.

3.4. Bando de dados compartilhado, esquema compartilhado

Uma terceira abordagem envolve o uso de um mesmo banco de dados e conjunto de tabelas para a hospedagem de múltiplos clientes. Uma única tabela pode ter registros vindos de vários clientes, armazenados em qualquer ordem, com uma única coluna para identificar a qual cliente o registro pertence.

Entre as abordagens apresentadas, esta é a que possibilita um número maior de clientes por servidor, gerando um custo de infraestrutura menor. Porém, essa abordagem pode propiciar um esforço maior para manter o banco de dados seguro, desenvolvendo o sistema de forma que garanta que um cliente não acesse dados de outros clientes, mesmo em caso de ataques ou bugs inesperados.

O procedimento para restaurar os dados de um cliente em caso de falha é similar o método de esquemas separados, com uma desvantagem a mais: as linhas individuais do cliente dentro do banco de dados de produção precisam ser apagadas e reinseridas em seguida. Caso o número de linhas e tabelas afetadas seja grande, essa operação pode gerar uma queda significativa de desempenho para todos os clientes.

Essa abordagem é apropriada quando é necessário que a aplicação suporte um grande numero de clientes com um numero de servidores reduzido, oferecendo um custo menor aos clientes.

Figura 7: Nesta abordagem, todos os clientes compartilham um mesmo conjunto de tabelas, e um ID para cada cliente associa-o aos seus próprios registros.

3.5. Escolhendo uma abordagem

Cada uma das abordagens definidas anteriormente oferece seus próprios conjuntos de benefícios, podendo ser escolhidos dependendo de algumas considerações técnicas.

Aplicações otimizadas por uma abordagem compartilhada necessitam de um grande esforço de desenvolvimento, devido à complexidade maior para desenvolver um sistema com arquitetura compartilhada, aumentando assim o custo inicial. Entretanto, devido à capacidade de suportar mais clientes por servidor, os custos operacionais tendem a ser mais baixos. Em casos em que é necessário que o software precise ir para o mercado o mais rápido possível ou a empresa não pode empregar um grande capital na fase de desenvolvimento, uma abordagem mais isolada é mais recomendável.

Conforme a aplicação necessitar armazenar dados sigilosos, os clientes irão exigir maior segurança. Apesar do uso da abordagem isolada ser recomendável, também é possível oferecer um forte esquema de segurança mesmo nas abordagens mais compartilhadas, usando padrões mais sofisticados de design.

Entretanto, o perfil dos clientes pode ter maior peso na hora de definir uma abordagem. Quanto à quantidade, quanto maior o número esperado de clientes, a expectativa de uso de uma arquitetura compartilhada é maior. Quanto ao espaço de armazenamento, se os clientes possuírem individualmente uma enorme quantidade de dados, poderá ser necessário uma abordagem mais isolada. Quanto ao fornecimento de serviços agregados (como backup e restauração de dados), o uso de uma arquitetura isolada facilitará esse tipo de serviço.

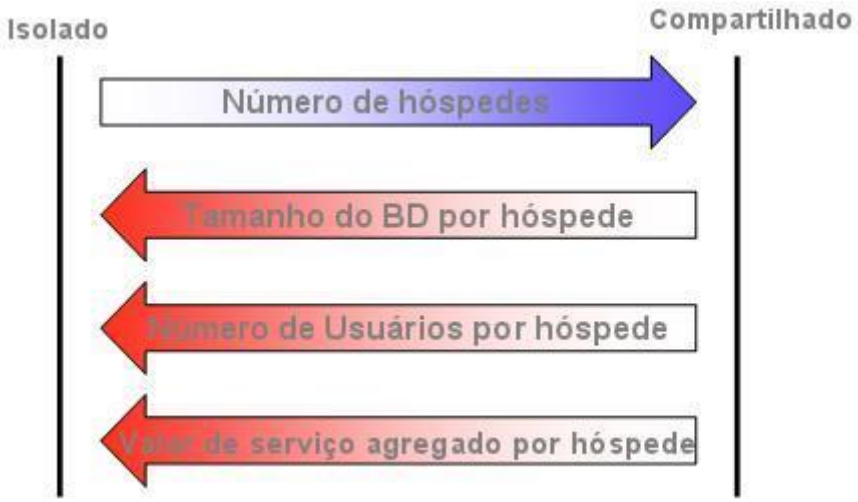


Figura 8: Fatores relacionados a clientes e como eles afetam as decisões de arquitetura de dados "isolados versus compartilhados".

3.6. Cloud Computing

Para o fornecimento de um Software como um Serviço, torna-se necessário o uso de computação nas nuvens (Cloud Computing) como estrutura da aplicação. A computação nas nuvens consiste em manter a aplicação e os dados dos clientes em servidores controlados pelo desenvolvedor da aplicação, sem que os clientes necessariamente saibam onde e como seus dados estão sendo guardados. Um conceito paralelo a isso seria a rede de eletricidade, onde os consumidores finais a usam sem necessariamente saber que componentes ou qual a infraestrutura necessária para fornecer o serviço.

O conceito de Cloud Computing é bastante abrangente e pode apresentar diversos cenários, mas sua principal característica é o fornecimento de recursos através de uma Nuvem Computacional, ou seja, um ambiente ubíquo no qual os usuários utilizam os serviços disponibilizados sem a necessidade de possuir informações sobre os elementos que os compõem" (Silva, 2009).

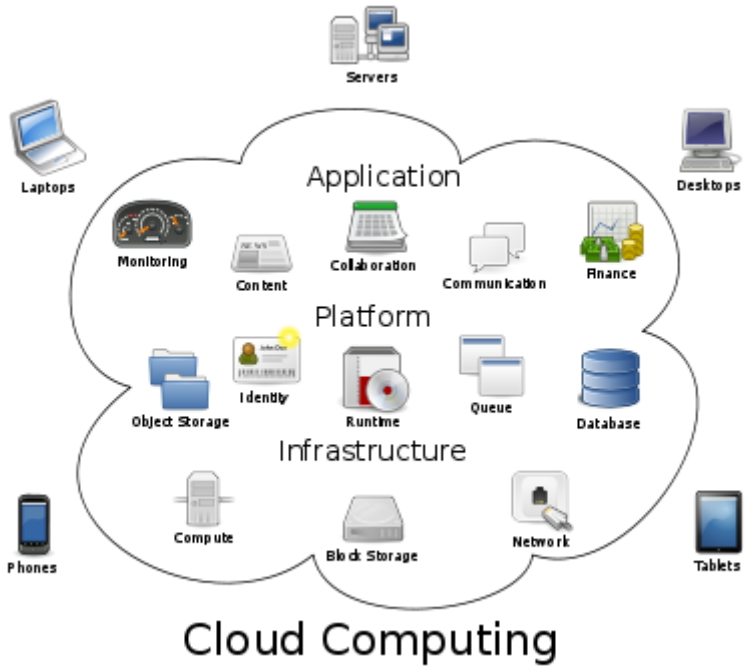


Figura 9: "Nuvem" da computação nas nuvens.

A computação nas nuvens possibilita que os usuários acessem a aplicação de qualquer lugar que estejam, não importando qual dispositivo que estejam usando (seja um computador, ou um celular, etc.), desde que tenha um navegador e acesso à internet. Além disso, pode-se aumentar a confiabilidade da aplicação com o uso de sites redundantes, tornando a aplicação apropriada para clientes que exijam disponibilidade extremamente alta e a possibilidade de se recuperar de um desastre (como a aplicação está na nuvem, um eventual desastre com a empresa não irá afetar a aplicação e seus dados).

Um dos principais benefícios da computação nas nuvens é a escalabilidade que o provedor disponibiliza para o usuário final. Esta capacidade de provisionamento automático de capacidade pode variar um pouco entre os provedores, podendo até usar sistemas totalmente automáticos, que podem variar a capacidade dedicada dos servidores de acordo com a demanda.

Para as empresas, as vantagens econômicas do modelo de provisionamento elástico são várias, principalmente quando se trata do custo inicial para aquisição de maquinário de grande porte como servidores. Outra vantagem é a não necessidade de contratação de funcionários dedicados a manter aqueles serviços funcionando. (GARTNER, 2008).

A empresa pode fornecer seu próprio equipamento para a nuvem (nuvem interna) ou uma empresa de terceiros pode fornecê-lo (nuvem hospedada), de modo que toda a infraestrutura da nuvem seja cuidada por esta empresa de terceiros. A nuvem pode ainda ser privada (restrita a uma única empresa ou grupo), pública (livre para o público geral via Internet) ou híbrida (compartilhada entre várias empresas ou grupos).

Uma nuvem engloba processamento, rede e elementos de armazenamento, e a arquitetura da nuvem consiste em três camadas abstratas. A camada de infraestrutura é a camada mais baixa e é um meio de fornecer armazenamento básico e capacidades de computação como serviços padronizados via rede. Servidores, sistemas de armazenamento, "switchs", roteadores e outros sistemas controlam tipos específicos de carga de trabalho, desde processamento batch até aumento de servidores ou armazenamento durante picos de carga. A camada intermediária da plataforma, fornece abstrações maiores e serviços para desenvolver, testar, implementar, hospedar e manter aplicações no mesmo ambiente de desenvolvimento integrado. A camada de aplicação é a camada mais alta e contém a aplicação completa oferecia como serviço. (Dikaiakos, 2009)

4. Características

Mesmo que as aplicações Software como um Serviço não partilhem todos os traços, as características abaixo são comuns entre muitas aplicações SaaS:

4.1. Customização e configuração

As aplicações SaaS suportam o que é conhecido tradicionalmente como customização da aplicação. Em outras palavras, da mesma forma que as aplicações empresariais tradicionais, um único cliente pode alterar um conjunto de opções de configuração (parâmetros) que afetam as funcionalidades, a aparência e a usabilidade da aplicação. Cada cliente pode ter sua própria configuração. A aplicação pode ser customizada ao nível em que ela foi desenvolvida, baseando-se em um conjunto de opções de configurações predefinidos.

Por exemplo: Para suprir a necessidade comum de mudar a aparência da aplicação para que ela pareça que tenha a marca do cliente, muitas aplicações SaaS possibilitam os clientes de fornecer um logo customizado e, em alguns casos, um conjunto de cores personalizado. Dessa forma, o cliente não pode alterar o layout da página a não ser que uma opção de configuração tenha sido desenvolvida.

4.2. Entrega acelerada de recursos

As aplicações SaaS são atualizadas com mais frequência que muitos softwares tradicionais, sendo, em muitos casos, atualizados semanalmente ou mensalmente. Isto é possível por vários fatores:

- A aplicação é hospedada centralmente, assim novas versões podem ser postas em prática sem precisar que os clientes instalem novas versões fisicamente.
- A aplicação tem apenas uma única configuração, tornando o desenvolvimento mais rápido.
- O fornecedor do aplicativo tem acesso a todos os dados do cliente, agilizando o projeto e testes de regressão.

- O provedor da solução tem acesso ao comportamento do usuário na aplicação, tornando mais fácil a identificação de áreas que valem a pena serem melhoradas.

Com a entrega acelerada de recursos é possível também o uso de metodologias de desenvolvimento ágil de software. Essas metodologias, que evoluíram em meados dos anos 1990, provisionam um conjunto de ferramentas de desenvolvimento de software e práticas para suportar lançamentos frequentes de versões.

Apesar das diferenças, um sistema SaaS pode ser desenvolvido no modelo tradicional, baseado no acesso direto a banco de dados, e ainda assim conseguir um desenvolvimento ágil. Nos sistemas tradicionais, perde-se um tempo considerável desde a finalização de uma atualização (ou recurso novo) até sua entrega ao cliente, devido à dificuldade de distribuir essa atualização a todos os clientes. Entre os processos de atualização de ambos modelos de sistemas, temos quatro etapas. Primeiro é feito o levantamento das mudanças, codificação, controle de qualidade e então a entrega. O tempo necessário para os três primeiros passos são parecidos entre o modelo tradicional e o modelo SaaS.

Onde eles de diferem é na última etapa: o ciclo de entrega. Aqui, sistemas tradicionais usam a metáfora do "grande ônibus": a entrega é contida até que todas as funcionalidades estejam a bordo, pois gerar uma versão de software é um pesadelo logístico. O fornecedor tem que gerar documentação e expedição para o canal, treinar tanto os clientes como o suporte pessoal para as mudanças, ajustar o preço e talvez a expedição, etc. (CREESE, 2010).

Já o SaaS, por outro lado, usa a metáfora do "transporte pequeno": uma nova versão sai quando uma ou duas funcionalidades são adicionadas, em grande parte por que a logística é muito mais simples. O fornecedor não precisa ajustar o preço pré-pago, e um

número pequeno de mudanças significa que o treinamento é ignorado ou é no máximo um pequeno texto de ajuda. (CREESE, 2010)

4.3. Protocolos de integração abertos

Como as aplicações SaaS não podem acessar um sistema interno de uma companhia (bancos de dados ou serviços internos), elas oferecem predominantemente integrações com protocolos e APIs que operam sobre uma rede de longa distância (WAN). Normalmente, estes protocolos são baseados em HTTP, REST, SOAP e JSON.

A onipresença das aplicações SaaS e outros serviços de internet e a padronização da sua tecnologia de API gerou o desenvolvimento de mashups, que são aplicativos leves que combinam dados, apresentação e funcionalidade de múltiplos serviços, criando um serviço composto. Esses mashups diferenciam ainda mais as aplicações SaaS das aplicações tradicionais já que estas não podem ser facilmente integradas fora do firewall da empresa.

4.4. Funcionalidade colaborativa

Inspirado pelo sucesso das redes sociais online e outras chamadas funcionalidades web 2.0, muitas aplicações SaaS oferecem recursos que permitem seus usuários colaborarem e compartilharem informação.

Por exemplo, muitas aplicações de gestão de projetos desenvolvidas no modelo SaaS oferecem – em adição ao recurso de planejamento de projetos – recursos de colaboração, possibilitando os usuários de comentar em tarefas e planos e compartilhar documentos dentro e fora da organização. Várias outras aplicações SaaS permitem os usuários a votarem e oferecerem novas ideias e recursos.

Enquanto algumas funcionalidades relacionadas à colaboração também podem ser integradas em aplicações tradicionais, a colaboração entre usuários de clientes diferentes é possível apenas com sistemas hospedados centralmente.

5. Metodologias

Apesar do modelo de Software como um Serviço ter muitas diferenças em relação aos Softwares tradicionais, para o desenvolvedor esses modelos são semelhantes. Apesar do modelo de negócio ser completamente diferente, as metodologias de desenvolvimento que são usadas nos Softwares tradicionais podem também ser usadas em Softwares como Serviço. A maior diferença, nesse caso, é o tipo de comunicação cliente-servidor, que deixa de existir no SaaS, para dar lugar a uma comunicação direta com o banco de dados dos clientes (normalmente ambos são hospedados no mesmo servidor) e o acesso do cliente a esse sistema que se torna remoto.

Dessa forma, fornecer suporte ao cliente fica mais fácil, já que não há problemas de compatibilidade de hardware e software, drivers, arquivos deletados indevidamente e ameaças de vírus específicos a um único cliente. Mesmo que esses problemas ainda possam existir, eles são praticamente impossíveis de serem causados por influência do cliente ou usuário, e sua incidência é extremamente baixa, já que o ambiente em que o sistema é instalado é controlado pelo desenvolvedor do Software.

Em muitos casos, há a necessidade de se comunicar com outras aplicações, sejam elas do próprio cliente ou de terceiros (para integrar recursos complexos, por exemplo). No caso de aplicações tradicionais, podem ser usados métodos simples, como o acesso direto ao banco de dados de outra aplicação, troca de dados por arquivos, troca de dados

via sockets e em alguns casos via internet (HTTP, Web Services, etc.). Já nas aplicações SaaS, devido à infraestrutura ser centralizada, o acesso a aplicações de terceiros se torna mais difícil, caso a mesma não possa ser implementada na infraestrutura utilizada pela aplicação SaaS, limitando a troca de dados apenas via internet.

6. Gestão

Devido ao modelo diferenciado de venda do software, a empresa acaba agregando custos que as empresas que vendem softwares tradicionais não têm. Da mesma maneira que o cliente não precisa mais se preocupar com infraestrutura e instalação, essas responsabilidades são passadas para o vendedor do software.

6.1. Infraestrutura

Os aplicativos de SaaS não exigem a implantação de grande infraestrutura no local do cliente e isso elimina ou reduz, drasticamente, o compromisso de recursos adiantados. Sem investimento inicial significativo para amortizar, a empresa que implanta um aplicativo SaaS, que resulte em produção de resultados desalentadores, poderá desistir e caminhar em outra direção, sem ter de abandonar a cara infraestrutura de suas instalações. (Biblioteca MSDN, 2007)

Como a responsabilidade da infraestrutura é passada para o vendedor do software, este precisa ter certeza de que ela é suficiente para suprir a atual quantidade de usuários, com uma margem de segurança para picos de acessos e outra margem ainda maior para suportar um aumento razoável de número de usuários, já que um cliente novo pode ter um volume de acessos muito maior que a média. Essa preocupação é constante, visto

que a cada cliente novo haverá uma carga maior – e uma necessidade maior – de infraestrutura.

6.2. Disponibilidade

Há uma preocupação grande em relação à disponibilidade do sistema, havendo a necessidade, em alguns casos, de infraestrutura redundante, para manter um alto nível de disponibilidade. É comum também, a adição de um limite mínimo de disponibilidade – dentro de um período – no contrato de venda, como forma de garantia, geralmente variando de 98 a 99,9%.

6.3. Instalação

Apesar de não haver gastos explícitos com a instalação do software em nenhuma das partes nesse modelo de negócio, existem alguns pontos a serem considerados pelo vendedor do software:

- No caso de um sistema WEB, existe a necessidade de o mesmo funcionar perfeitamente ao menos nos quatro navegadores mais utilizados atualmente – Internet Explorer, Firefox, Chrome e Safari correspondem juntos por aproximadamente 98% do mercado de navegadores desktop, segundo o site netmarketshare.com. Isso envolve uma demanda por desenvolvimento, manutenção e testes diferenciados, para que a aplicação continue servindo a esses navegadores.

- A adaptação do sistema para funcionar via aparelhos móveis (celular e tablets) pode se tornar financeiramente inviável, caso sejam necessárias muitas alterações.

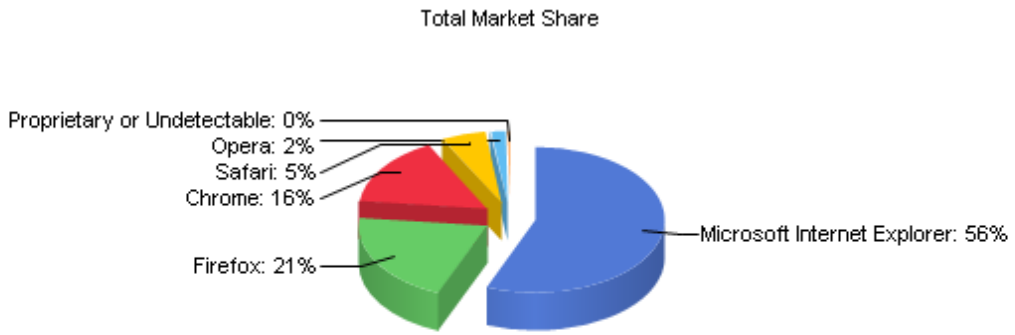
- A infraestrutura deve ser previamente preparada para novas instalações (novos clientes), pra que o volume de novos usuários não ultrapasse as limitações físicas.

- O Software tem que estar preparado para receber o volume extra de usuários, para que não haja perda de desempenho e/ou dados.

Segue abaixo gráfico traduzido retirado do site netmarketshare.com:

Fatia de mercado dos navegadores desktop

Maio de 2013



Browser	Total Market Share
Microsoft Internet Explorer	55.99%
Firefox	20.63%
Chrome	15.74%
Safari	5.46%
Opera	1.77%
ProprietaryorUndetectable	0.23%
Mozilla	0.03%
Flock	0.01%
Konqueror	0.01%

Figura 10: Fatia de mercado dos navegadores desktop.

7. Vantagens

Várias mudanças importantes no mercado de software e o cenário da tecnologia facilitaram a aceitação e o crescimento das soluções SaaS:

- O crescente uso de interfaces humanas baseadas na web por aplicações, junto com a proliferação de práticas associadas (web design, por exemplo), diminuíram continuamente a necessidade de aplicações cliente-servidor tradicionais. Consequentemente, o investimento dos vendedores de aplicações tradicionais em softwares baseados em clientes robustos[2] se tornou uma desvantagem (devido à necessidade de suporte), abrindo a porta para que os vendedores de novos softwares oferecessem uma experiência de uso percebida como mais "moderna".

- A padronização do "pacote" web (HTML, JavaScript, CSS, HTTP), o aumento da popularidade do desenvolvimento web como uma prática e a introdução e onipresença dos frameworks de aplicações web reduziram gradualmente o custo para desenvolver novas soluções SaaS e possibilitou os provedores de novas soluções a lançarem soluções com preços competitivos, desfiando os vendedores tradicionais.

- O alcance crescente do acesso de internet de banda larga possibilitou que as aplicações remotas hospedadas centralmente oferecessem velocidade comparável a sistemas tradicionais.

- A padronização do protocolo HTTPS como parte do pacote web ofereceu segurança leve disponível universalmente, que é suficiente para a maioria das aplicações diárias.

[2]Clientes robustos são programas cliente-servidor que mantêm na aplicação cliente a grande maioria dos recursos e das funcionalidades, tornando a aplicação cliente grande e complexa.

- A introdução e aceitação em larga escala de protocolos leves de integração, como o REST e o SOAP permitiram uma integração acessível entre aplicações SaaS (residindo em nuvem) com aplicações internas via redes de longo alcance e com outras aplicações SaaS.

8. Desvantagens

Algumas limitações retardam a aceitação do SaaS e impossibilita o seu uso em alguns casos:

- Como os dados são guardados nos servidores dos vendedores, a segurança desses dados começa a se tornar um problema.

- As aplicações SaaS são hospedadas na nuvem, longe dos usuários da aplicação. Isso causa um tempo de resposta maior ao ambiente, tornando inviável seu uso em uma aplicação que exige tempos de resposta extremamente curtos.

- As arquiteturas multi-tenant, que elevam o custo-benefício para os provedores de soluções SaaS, não permitem a real customização para grandes clientes, proibindo essas aplicações de serem usadas em cenários em que tamanha customização é necessária.

- Algumas aplicações comerciais precisam de acesso ou integração com os dados atuais do cliente. Quando estes dados são volumosos ou sensitivos (informações pessoais de usuários finais), integrá-los com um sistema hospedado remotamente torna-se muito caro e/ou arriscado.

9. Custódia de dados

Custódia de dados em um SaaS é o processo de manter uma cópia de dados críticos da aplicação SaaS com uma empresa independente terceirizada. Isso permite as companhias proteger e garantir todos os dados que residem nas aplicações SaaS, protegendo contra perda de dados.

Existem muitas razões para considerar a custódia de dados em SaaS, incluindo preocupações de falência do vendedor, interrupções de serviços não planejadas e potencial perda/corrupção de dados. Muitas empresas também estão ansiosas para assegurar que elas estão cumprindo com seus próprios padrões de governança de dados ou desejam melhorar os relatórios e análise de negócios em relação aos seus dados da aplicação SaaS. Uma pesquisa conduzida pela Clearpace Software Ltd. para o crescimento do Software como um Serviço mostrou que 85 por cento dos entrevistados gostariam de ter uma cópia de seus dados de suas aplicações SaaS. Um terço desses entrevistados gostaria de ter uma cópia em uma base diária.

A Associação Brasileira de Normas Técnicas (ABNT) possuiu uma norma relacionada à segurança da informação, visando à proteção da informação, seja ela eletrônica ou não.

A norma ABNT NBR ISO/IEC 27002:2005 define: “A informação é um ativo que, como qualquer outro ativo importante, é essencial para os negócios de uma organização e, consequentemente, necessita ser adequadamente protegida. [...] A informação pode existir em diversas formas. Ela pode ser impressa ou escrita em papel, armazenada eletronicamente, transmitida pelo correio ou por meios eletrônicos, apresentada em filmes ou falada em conversas. Seja qual for a forma de apresentação ou o meio através do qual a informação é compartilhada ou armazenada, é recomendado que ela seja sempre protegida adequadamente.”.

Ainda de acordo com a norma, “Segurança da informação é a proteção da informação de vários tipos de ameaças para garantir a continuidade do negócio, minimizar o risco ao negócio, maximizar o retorno sobre os investimentos e as oportunidades de negócio.”.

Portanto, o fornecedor do software tem a responsabilidade de manter seguros os dados de seus clientes, já que mantêm em seus servidores os dados confiados por seus clientes. Existe ainda a norma internacional ISO/IEC 27002, que é um conjunto de recomendações para práticas na gestão de segurança da informação, tendo como objetivo a confidencialidade, integridade e disponibilidade das informações.

Dessa forma, cabe ao fornecedor do software garantir que os dados de seus clientes não sejam acessados por pessoas não autorizadas pelo cliente dono da informação, sejam mantidos com todas as características originais estabelecidas pelo proprietário da informação e estejam sempre disponíveis para serem utilizados pelos usuários que estão autorizados pelo cliente proprietário.

10. Testes

Como o ambiente do usuário é controlado (dentro dos limites do provedor da aplicação), fazer testes no mesmo ambiente que o usuário se torna possível e sem a necessidade de se utilizar acesso remoto ou qualquer outra ferramenta. Isso diminui o tempo de reação e correção de bugs encontrados por usuários, diminuindo a chance do mesmo problema ser encontrado pelos demais usuários/clientes.

Há também a possibilidade de se criar testes beta com uma quantidade limitada de usuários, para demonstrar e testar funções novas com uma quantidade razoável de

testadores antes de liberar uma versão oficial, aumentando consideravelmente a confiabilidade das novas funcionalidades quando a versão oficial for lançada.

Referências Bibliográficas

Sites da internet:

- Diversos autores – Arquitetura de dados para múltiplos inquilinos, disponível em: http://msdn.microsoft.com/pt-br/library/aa479086.aspx (acesso: Janeiro/2013 a Junho/2013)

- Diversos autores – Software as a Service, disponível em: http://msdn.microsoft.com/en-us/library/bb245821.aspx (acesso: Janeiro/2013 a Junho/2013)

- LISSERMAN, Miroslaw -SaaSAnd The Everlasting Security Concerns, disponível em: http://community.forrester.com/message/10906 (acesso: Janeiro/2013 a Junho/2013)

WAINEWRIGHT, Phil – Workstream prefers virtualization to multi-tenancy, disponível em: http://www.zdnet.com/blog/saas/workstream-prefers-virtualization-to-multi-tenancy/400 (acesso: Janeiro/2013 a Junho/2013)

- CREESE, Guy – SaaS vs. Software: The Release Cycle for SaaS Is Usually (Not Always) Faster, disponívelem: http://blogs.gartner.com/guy-creese/2010/05/18/saas-vs-software-the-development-cycle-for-saas-is-usually-not-always-faster/ (acesso: Janeiro/2013 a Junho/2013)

- BARRETT, Larry – SaaS Market Growing by Leaps and Bounds: Gartner, disponívelem: http://itmanagement.earthweb.com/entdev/article.php/3895101/SaaS-Market-

Growing-by-Leaps-and-Bounds-Gartner.htm (acesso: Janeiro/2013 a Junho/2013)

- ANDERSON, Tim – Let the Cloud Developer Wars begin, disponívelem: http://www.theregister.co.uk/2011/05/05/cloud_vendor_platforms/ (acesso: Janeiro/2013 a Junho/2013)

- Gartner – Gartner Says Worldwide Software as a Service Revenue Is Forecast to Grow 21 Percent in 2011, disponívelem: http://www.gartner.com/it/page.jsp?id=1739214&M=6e0e6b7e-2439-4289-b697-863578323245 (acesso: Janeiro/2013 a Junho/2013)

- Software & Information Industry Association – Software as a Service:Strategic Backgrounder, disponível em: http://www.siia.net/estore/pubs/SSB-01.pdf (acesso: Janeiro/2013 a Junho/2013)

- Net Applications.com – Diagramas de fatia de mercado, disponível em:http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0 (acesso: Janeiro/2013 a Junho/2013)

- Diversos autores – Cloudcomputing não é tendência / Segurança em cloud começa no chão e não na nuvem, disponíveis em: http://saasbr.wordpress.com/ (acesso: Janeiro/2013 a Junho/2013)

- D. C. Silva. Grid Computing e CloudComputing- análise dos impactos sociais, econômicos e ambientais da colaboração por meio do compartilhamento de recursos. Pontifícia Universidade Católica de São Paulo/PUC-SP, 2009.

- Ten Miles – PricingyourSaaSApplication, disponível em: http://tenmiles.com/blog/2010/04/pricing-your-saas-application/(acesso: Janeiro/2013 a Junho/2013)

- Marios D. Dikaiakos, George Pallis, Dimitrios Katsaros, Pankaj Mehra e Athena Vakali - Cloud Computing: Distributed Internet Computing for IT and Scientific Research, disponível em: http://www.cs.ucy.ac.cy/~gpallis/publications/journals/editorial.pdf (acesso: Junho 2013)