Opções de Estratégias para o Desenvolvimento de *Software as a Service*

Katyusco DE FARIAS SANTOS (1); Cícero Alan LEITE CRUZ (2); Giovanni FARIAS DA SILVA (3)

(1) Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco, Av. Prof. Luiz Freire, 500, Cidade Universitária, Recife - PE, Brasil, e-mail: katyusco@recife.ifpe.edu.br
(2), (3) Universidade Federal de Campina Grande, Rua Aprígio Veloso, 882, Bairro Universitário, Campina Grande – PB, CEP: 58.429-140, Brasil, e-mail: {*alanlcruz, giovanni.surubim*}@gmail.com

RESUMO

Software como serviço (*software as a service – SaaS)* ressurge hoje, de forma promissora dada a diversidade de Tecnologias da Informação atuais que compreendem a computação nas nuvens (*cloud computing*). *SaaS* é um novo paradigma de entrega e comercialização do produto software em que o cliente ao adquirir, não mais detém a sua posse, mas tem o direito de fazer uso das suas funcionalidades. A posse do software permanece com o fornecedor. Como conseqüência, as aplicações *SaaS* têm requisitos não-funcionais como escalabilidade, *deployment* contínuo e outros, que os faz possuir características inerentes e que os difere dos demais tipos de aplicação. Este trabalho reporta as implicações para a atividade de desenvolvimento trazidas com *SaaS*, as estratégias arquiteturais de maturidade que melhor se adéquem às necessidades dos fornecedores de *SaaS*, e as opções para executar o desenvolvimento: *ad-hoc* (auto-suficiente) ou utilizando a pilha arquitetural de *cloud computing* com *IaaS* (*Infrastructure as a Service)* e/ou *PaaS* (*Platform as a Service).*

Palavras-chave: *Software as a Service, SaaS, PaaS, cloud computing*

1 INTRODUÇÃO

O termo *cloud computing* passou a ser bastante explorado tanto pelas empresas ligadas à Tecnologia da Informação (TI) quanto pelos canais de mídia tradicionais. Os motivos para tanto são bastante diretos, pois os primeiros pretendem se manter competitivos no mercado e a mídia refletiu a ampla adoção de serviços on-line pelos usuários da internet. Ao mesmo tempo, o ambiente acadêmico têm realizado pesquisas na área e levantado desafios, proposto padronizações, taxonomias e ferramentas, bem como avaliado soluções e modelos. Um dos termos que está englobado em *cloud computing* é o de Software como Serviço, que também tem gerado alvoroço no ambiente coorporativo e acadêmico.

Embora a idéia de prover *software* como serviço, nos moldes do que se tem hoje, não ser algo novo, pois Bennet et. al. [5], já o discutia em 1999, e muito já se tenha discutido a respeito de *SaaS*, ainda há equívocos sobre o tema, principalmente no que se refere a definições e diferenciações entre *SaaS* e tecnologias, modelos e conceitos já existentes. Também, novos desafios surgem quando se observa que implicações para a atividade de desenvolvimento o *SaaS* traz consigo. Como reflexo, tanto empresas como academia têm sua própria visão de como devem oferecer suas respectivas soluções para o desenvolvimento de aplicações *SaaS*.

Nesse contexto, é objetivo deste trabalho identificar as implicações para a atividade de desenvolvimento de software que acompanham o paradigma de *SaaS*, e apresentar as opções para construção de *SaaS* em função dos modelos de maturidade de referência. A seção 2 discorre sobre os conceitos fundamentais de cloud computing, software as a service e suas características. Em seguida, na seção 3, são definidas as implicações sobre a forma de desenvolver software com a adoção de SaaS. Opções de estratégias para executar o desenvolvimento SaaS são aboradas na seção 4. Por fim, na seção 5, são apresentadas nossas conclusões.

2 Fundamentos

2.1 *Cloud Computing*

Cloud computing (computação nas nuvens) está relacionada com as capacidades a que são fornecidos, tal como um serviço, permitindo que usuários acessem serviços de tecnologia a partir da Internet, sem o conhecimento onde estão localizados, e sem controle sobre a infra-estrutura tecnológica que os suporta. De forma didática computação nas nuvens pode ser definida como um modelo no qual a computação (processamento, armazenamento e softwares) está disponível em algum lugar da rede de forma escalável, é acessada remotamente, via internet, e seu pagamento se dá em função das demandas de computação.

Dada a diversidade de tecnologias emergentes para computação nas nuvens, trabalhos [11] [13] indicam uma taxonomia e uma arquitetura de pilha de serviços padronizados (figura 1) que sirvam como referência para comparações, discussões e novas implementações. Sobre essa pilha é possível prover e consumir recursos físicos, infraestrutura virtualizada, plataformas de *middleware* e aplicações, como serviços.

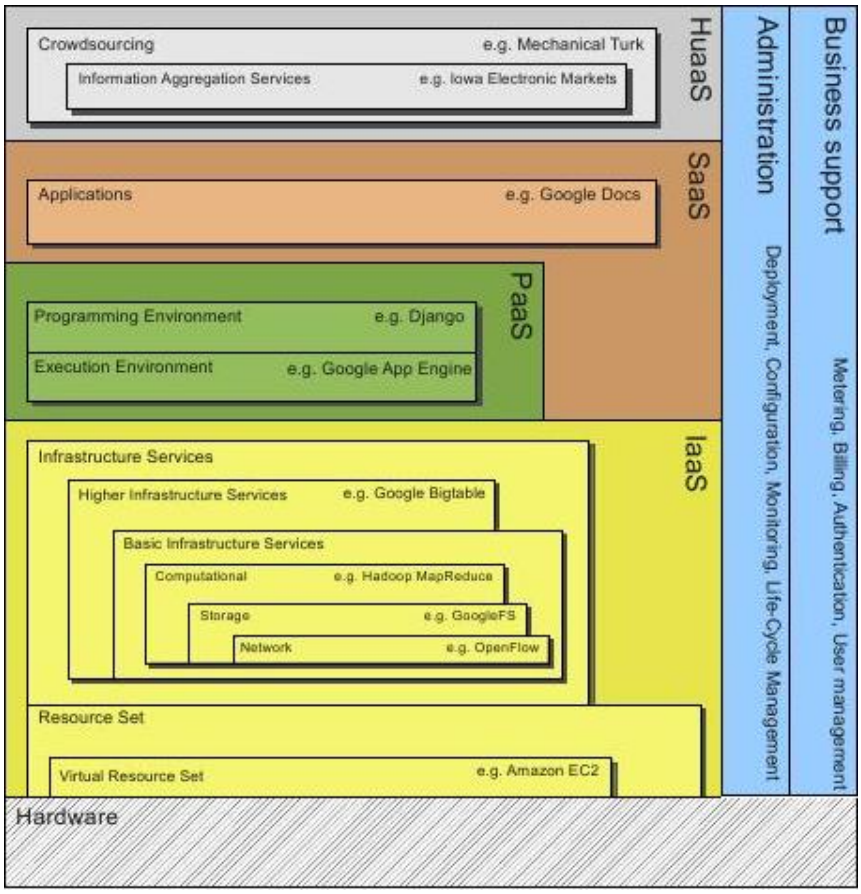

Figura 1 – Pilha Arquitetural para Computação nas Nuvens [12].

O nível *IaaS - Infrastructure as a Service* -, possui uma camada próxima ao *hardware*, a *Resource Set*, que engloba um conjunto de funcionalidades primárias (*e.g.* inicialização, configuração) e as disponibiliza através de uma API às camadas superiores da pilha. Divide-se em *Physical Resource Set (PRS)* cuja implementação é dependente do *hardware*, e *Virtual Resource Set (VRS)* implementado sob virtualização. Acima do *Resource Set*, são definidas a infraestrutura básica (*BIS - Basic Infrasctructure Services*) de processamento, armazenamento e rede. Em *IaaS*, encaixam-se: *datacenters* e grades computacionais.

Os ambientes de programação (*Programming Enviroments*) e de execução (*Execution Enviroments*) para aplicações na nuvem estão no nível de *Paas - Platform as a Service* -. Ambientes de execução tipicamente englobam os de programação, e podem utilizar diferentes ambientes de programação desde que haja compatibilidade entre os dois. *Google App Engine* suporta o *framerwork Django*, Java e Python [2].

Aplicações que executam sobre as nuvens, efetivamente, encontram-se na camada *SaaS - Software as a Service*. É divida em *Basic Application Services*, onde estão os serviços básicos (*e.g.* identificação), e em

Composite Application Services que são os serviços constituídos de outros, básicos inclusive (*e.g.* redes sociais).

No nível mais alto da pilha encontra-se o *HuaaS - Human as a Service* -. Nele estão os serviços agregados de informação (*IAS - Information Aggregation Services*) gerados a partir do uso coletivo ou em massa de uma aplicação, e que representam a opinião popular e promovem *marketing*. Tais aplicações se tornam ferramentas de trabalho para grupos de pessoas, e são denominadas de *CrowdSourcing* (*e.g: YouTube*).

Serviços de administração (*Administration*) e de suporte ao negócio *(Business support)*, *e.g.* medição, monitoramento e tarifação, devem oferecer, aos fornecedores e clientes, visão coordenada e completa de suas atribuições quando atuam sobre as aplicações. Logo, estes precisam ter acesso a todas as camadas da pilha.

É nesse acúmulo de tecnologias que surge um complexo ecosistema composto de clientes e fornecedores de serviços na nuvem, onde aplicações através da *Web* tornam-se viavelmente escaláveis, fazendo uso de uma poderosa infraestrutura computacional de forma transparente, que é racionalmente utilizada promovendo redução de custos para quem as provê. Junta-se, a necessidade de se ter acesso aos dados a partir de qualquer lugar através da Internet, e logo tem-se uma atmosfera propícia para disseminação do software como serviço.

2.2 *Software as a Service – SaaS*

Para um investidor, provedor de serviços ou até mesmo uma empresa de desenvolvimento, *SaaS* é um modelo de negócio. Para um cliente em potencial, é um modelo de entrega.

De forma unificada, *SaaS* deve ser visto como um modelo de entrega de software que permite a exploração de um novo modelo de negócio. Bennet et. al. (2000), sintetiza Software como Serviço: "Software as a Service (*SaaS*) é um modelo de distribuição de software em que o fornecedor armazena a aplicação na Internet, sob sua própria infra-estrutura, e a disponibiliza via browser aos usuários que executam e armazenam seus trabalhos de maneira online, e cujos serviços oferecidos são tarifados sob demanda." [5].

Como uma funcionalidade a ser fornecida ao cliente, *SaaS* é um modelo de entrega de software que modifica quem fica com a posse do produto, que agora passa a ser o provedor(fornecedor) do software, já que ele mantém o controle operacional do software. Enquanto que na maneira tradicional o software é tratado como qualquer outro tipo de mercadoria, um produto, *Software as a Product - SaaP*, e a posse é do cliente.

Na comercialização, *o SaaP* é vendido como um produto, empacotado com manual, e o cliente paga antecipadamente por todas as funcionalidades disponíveis, mesmo por aquelas que nunca serão utilizadas. Já o *SaaS* viabiliza um novo mecanismo de cobrança, o pagamento sob-demanda (pay-as-you-go), onde clientes pagam apenas pelo o que utilizam.

Permanecendo com a posse do software consigo os custos de mantê-lo em funcionando também é do provedor. Para oferecer garantias ao cliente é feito um Acordo de Nível de Serviço (SLA), um contrato em que ambas as partes formalizam a qualidade do serviço a ser prestado. SLAs refletem as preocupações dos clientes com segurança, privacidade, performance e disponibilidade, e selecionam um provedor de *SaaS*.

Diversas são as maneiras de cobrar um serviço sob-demanda. Pagamento de uma taxa periódica (e.g.: anual) fixa (assinatura), quantidade de transações ou de usuários que utilizam o serviço (*seats*) [12]. É possível oferecer diferentes opções para que o cliente escolha a que melhor se adéqua às suas necessidades. Assim, no *SaaS* os pequenos e médios negócios (*Small and Medium Business*), antes excluídos dos benefícios da informatização, dado os altos preços da TI, agora aumentam a base de clientes potenciais. A captação de receitas pode se dá ainda, trocando a gratuidade do serviço pela veiculação de propagandas, o que possibilita que usuários domésticos façam uso de *SaaS*.

2.3 Características de *SaaS*

Em função da maneira como são entregues pelos fornecedores aos clientes e de suas formas associadas de comercialização e negociação, as aplicações *SaaS* podem ser identificadas e diferenciadas de outros tipos de aplicações através das seguintes características prevalecentes:

- **Entrega e Acesso via *Web*** – O mundo *SaaS* inexiste sem a *Web*. As aplicações *SaaS* são projetadas para serem hospedadas em provedores, e globalmente acessíveis através dos *thin-clients* (*browsers*).
- **Pagamento sob-demanda** – É a característica que atende ao novo modelo de negócio. O fornecedor cobra do cliente por utilização dos serviços, e não pela licença do produto (modelo tradicional);

- **Gerência via Web** – Possibilita que o cliente de um *SaaS* possa ter acesso as estatísticas de uso dos serviços, bem como o potencial para customizar a aplicação para atender as suas necessidades;
- **Gerenciamento Centralizado** - Permite que o provedor de *SaaS* tenha uma única aplicação para gerenciar cada cliente. A customização é feita pelos próprios inquilinos, no paradigma *self-service*;
- **Medição** - Capacidade de rastrear e quantificar o uso de recursos e/ou de funcionalidades. Tarifa clientes no modelo *pay-as-you-go*, e monitorar a qualidade dos serviços;
- **Composabilidade** - É a possibilidade de criar novos serviços a partir de serviços já existentes, seja agregando ou integrando serviços;
- **Desenvolvimento e Atualização Contínuos** - Permite que continuamente os *SaaS* tenham disponível para uso, assim que concluídas, novas funcionalidades e correções;
- **Plataforma Multi-inquilino** – É a característica mais relevante e inovadora, que diferencia *SaaS* de aplicações Web [8]. Uma plataforma multi-inquilino utiliza recursos comuns e uma única instância, tanto da aplicação como do banco de dados, para suportar diversos clientes simultaneamente, de forma semelhante a clientes que teriam recursos dedicados a apenas a sua aplicação individual [9];
- **Escalabilidade** – Remete a capacidade de *SaaS* de incrementar a quantidade de clientes atendidos, sem comprometer o nível da qualidade de serviços prestados (quantidade de funcionalidades, tempo de resposta) a todos os demais, inclusive aos já em atendimento;
- **Online Marketing** - Capacidade de divulgação automática do serviço através das plataformas online, a exemplo de redes sociais, blogs ou catálogos de serviços.

A maioria dessas características, dependendo do porte da aplicação *SaaS* e dos requisitos adicionais, pode implicar na necessidade de outras características: Multi-inquilino implica em confiabilidade e segurança, já que plataformas multi-inquilino requerem que os dados dos clientes sejam mantidos de forma confiável (consistentes), e seguros contra acessos indevidos; Acesso via *Web* implica em disponibilidade, pois mesmo com acesso irrestrito a *Web*, caso a aplicação *SaaS* falhe nenhum dado do cliente poderá ser acessado.

3 Implicações de *SaaS* para o Desenvolvimento de Software

Para que as características de *SaaS* sejam atendidas, diversos interesses não funcionais precisam ser considerados. Requisitos não-funcionais como escalabilidade, customização através de configuração, internacionalização, segurança e confiabilidade, embora importantes, podem ser deixados em segundo plano no paradigm *SaaP*, pois, o *deployment* é feito na infra-estrutura do cliente e esse se responsabiliza pela operacionalização. Além disso, em caso de falha de serviço ou invasão no sistema, apenas um único cliente se prejudica, enquanto que na oferta de *SaaS* todos os Inquilinos são prejudicados.

Para provedores de *SaaS* existem diversas necessidades de natureza arquitetural e tecnológica para dar suporte ao provimento de software como serviço de maneira escalável. Nota-se que a característica multi-inquilino é a que mais gera implicações no desenvolvimento, pois é essa que leva a uma mudança de paradigma de desenvolvimento e a partir dela que se consegue viabilizar a venda de uma única instância de software para diversos usuários ao mesmo tempo e atingir a economia de escala, discutida na seção 3.3. Como o modelo de negócio do *SaaS* exige baixo custo de manutenção para o provedor e baixo custo de venda ao cliente, a arquitetura multi-inquilino é prevalecente e é com base nela que se avalia o nível de maturidade de uma solução *SaaS*.

Como o desenvolvimento de *SaaS* está envolto por interesses sistêmicos e comuns a todo tipo de aplicação oferecida como serviço, é natural o uso de componentes de software, frameworks e plataformas para acelerar o desenvolvimento. Ainda, como apresentado na figura 1, a infra-estrutura oferecida pela computação nas nuvens é um grande aliado para a oferta de software como serviço e muitos dos que oferecem *IaaS* também oferecem *PaaS* [11].

3.1.1 Modelos de Maturidade

Para classificar os serviços de *SaaS* quanto ao nível de conformidade para uma aplicação que siga fielmente o modelo de oferta de software como serviço, existem alguns modelos de maturidade, a exemplo de [1], [4] e [7]. Em geral, há consenso na existência de quatro níveis de maturidade quando se classificam as aplicações disponibilizadas como serviço. A figura 2 ilustra esse modelo de maturidade.

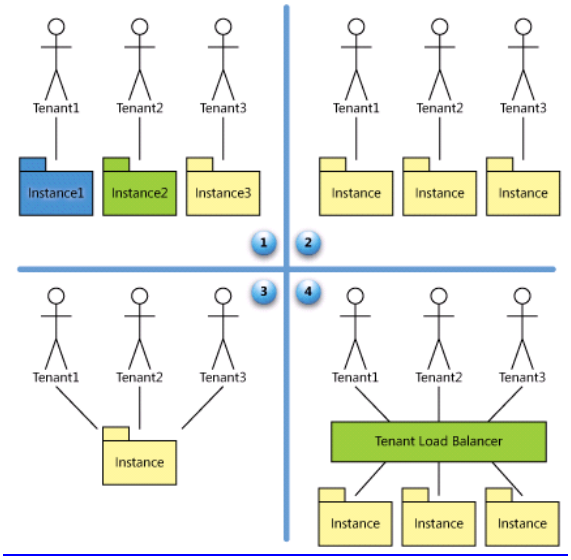

Figura 2. Modelo de maturidade *SaaS*, sobre a evolução do suporte multi-inquilino. [4]

Os pontos chave que diferenciam a arquitetura de um *SaaS* de um nível para outro são atributos de customização de *deployment* [13] que são: potencial de configuração, eficiência no atendimento a multi-inquilino e escalabilidade. A presença desses atributos diferencia um nível mais alto de um mais baixo.

No Nível 1, *Ad-Hoc*/*Custom*, existe uma versão customizada do software para cada cliente. Embora a aplicação esteja sendo mantida pelo fornecedor do software, há uma relação um-para-um entre cliente e versão. Obviamente, neste nível de maturidade o serviço não é escalável, pois a adição de mais um cliente resulta em um alto custo para o fornecedor, devido à ausência de compartilhamento de recursos e da necessidade de customização elevada. Como neste nível o esforço de desenvolvimento da aplicação *SaaS* é baixo, é muito comum estarem nele aplicações que foram migradas recentemente de um modelo tradicional mantido *in-house* e foi transferido para a infra-estrutura de terceiro para reduzir custos operacionais, isto é, com hardware e administração. De acordo com [8] é similar ao modelo de aplicações de software usadas pelos ASP (*Application Software Provider*); Isso garante um completo atendimento das demandas do cliente, mas com elevado custo de operação e manutenção devido à ausência de compartilhamento de recursos e customização elevada.

No Nível 2, *Configurable*, cada cliente continua possuindo sua própria instância rodando no servidor, no entanto, a customização é feita através de configuração em uma única aplicação. Embora não se ganhe muito com o compartilhamento de recursos, neste nível já se tem a vantagem de se possuir uma base de código (solução) comum que servirá para todos os clientes, diminuindo o custo de manutenção e atualização por parte do fornecedor, em relação ao nível 1;

No Nível 3, *Configurable*, *Multi-Tenant-Efficient*, todos os clientes dividem uma única instância de uma aplicação. Em relação ao nível 2, neste nível o uso de recursos no servidor se dá de forma mais eficiente, adicionando a capacidade de multi-inquilino (multi-tenancy) a aplicação. Para atender multi-inquilinos, tratamento de metadados, assim como manutenção e modelagem do banco de dados são necessários possibilitando a customização automática da instância da aplicação. Nesse nível, surgem diversas preocupações, pois clientes que podem ser rivais em seus negócios estariam compartilhando recursos e poderiam ter seus dados expostos para outros clientes. Em [8], é apresentado um *framework* utilizando Web Server, Application Server e o SGBD DB2, e uma implementação conceito de uma aplicação multi-inquilino sobre esse *framework*;

No Nível de maturidade 4, *Scalable, Configurable*, *Multi-Tenant-Efficient*, é adicionada a escalabilidade ao nível anterior, graças a uma arquitetura multi-camada para suportar balanceamento de carga entre diferentes instâncias de uma mesma aplicação, rodando em um número variável de servidores. Desta forma, a capacidade do sistema pode ser aumentada ou diminuída (elasticidade) sob demanda através da adição ou remoção de servidores sem haver a necessidade de alterações na arquitetura da aplicação. Este nível é o ponto chave para se obter vantagem da Cauda Longa em uma economia de escala, podendo atingir, potencialmente, centenas de milhares de clientes. Neste nível, há facilidade para prover qualidade de serviço,

pois se permite um atendimento diferenciado para inquilinos que exigem elevada demanda de recursos, havendo uma carga balanceada na infra-estrutura do provedor da solução *SaaS*.

Embora esse modelo de maturidade seja amplamente adotado, são omitidos fatores importantes quanto a arquitetura de um *SaaS*, a exemplo de particionamento de dados, versionamento de instâncias individuais e integração entre *SaaS*.

4 OPÇÕES PARA O DESENVOLVIMENTO DE *SAAS*

A maneira tradicional de se comercializar software é tratando-o como um produto (*Software as a Product - SaaP*), da mesma forma como se faz com qualquer outro tipo de mercadoria. Quando o software oferecido como produto não satisfaz os requisitos de customização de um cliente, seja por regras de negócio, integração, ou requisitos não funcionais, uma opção é o software como projeto, em que uma equipe de desenvolvimento ou uma *software-house* é contratada e desenvolve todo o projeto do software. Estas duas visões do software, a primeira como uma mercadoria bastante padronizada, e a segunda como um produto altamente específico, guiaram todo o desenvolvimento de software nos últimos 50 anos. Ambas se assemelham ao considerarmos que o cliente é quem fica com a posse do produto software.

O modelo de entrega e de negócio preconizado por *SaaS*, onde a posse do produto permanece com o provedor, gera implicações no desenvolvimento, principalmente nos aspectos relacionados a interesses sistêmicos, devido a monitoramento, segurança e configuração, e de *deployment*, devido aos ciclos de releases bastante curtos. Para se desenvolver uma solução *SaaS* pode-se optar por uma estratégia auto-suficiente [8] ou sobre a pilha *de cloud computing* [10]. Ainda, como pode ser visto na figura 1, pode-se optar por implementar a solução diretamente sobre *IaaS* ou utilizar as facilidades da *PaaS*.

4.1 *Ad-hoc* – Solução Auto-Suficiente

A solução auto-suficiente, isto é, independente de computação nas nuvens, foi a primeira forma de implementação de Software como Serviço, explorada desde o início dos anos 2000 e que tem na SalesForce o principal exemplo. Ainda hoje muitas soluções são feitas de maneira auto-suficiente, a exemplo de [3], [6], e [8], utilizando soluções de servidores *web*, SGBDs e *frameworks* proprietários. Embora seja uma estratégia válida principalmente para os grandes *players*, a exemplo de IBM, SalesForce, Microsoft e Oracle, pois os permite desenvolver a *expertise* para a provisão de *frameworks* e plataformas para desenvolvedores independentes, essa não se torna uma opção tão atrativa para pequenos *players* ou pequenos Desenvolvedores Independentes de Software (*Independent Software Vendors – ISV)*, visto a dificuldade de desenvolvimento, e alto-custo na provisão de recursos de infra-estrutura. A figura 3. mostra uma instância de um *SaaS*, x, atendendo a dois inquilinos distintos A e B onde suas Bases de Dados, A e B, estão em Servidores de Banco de Dados distintos, e outra instância do mesmo *SaaS*, x, atendendo os inquilinos C e D onde suas Bases compartilham o mesmo Servidor de Banco de Dados.

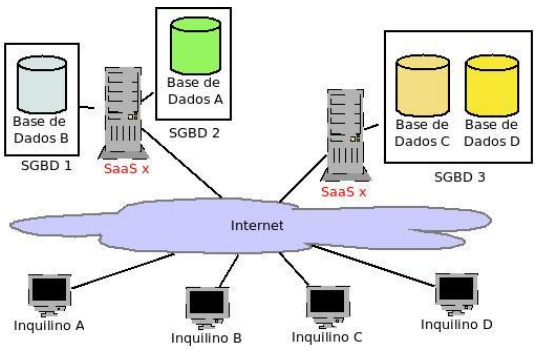

Figura 3. Solução *Ad-hoc*.

4.2 Na Nuvem (*Cloud*)

Uma solução auto-suficiente mantém um acoplamento fortíssimo entre a provisão do serviço e a provisão de infra-estrutura. A computação nas nuvens permite separar as duas provisões, oferecendo infra-estrutura como serviço por meio dos provedores de infra-estrutura. Indo um pouco além no nível de abstrações, alguns provedores fornecem toda uma plataforma de desenvolvimento ou execução, visando facilitar e agilizar o desenvolvimento do software como serviço.

Assim, ao desenvolver software como serviço na nuvem têm-se as opções de se apoiar diretamente na infra-estrutura como serviço (*IaaS*) ou na plataforma como serviço (*PaaS*), permitindo que o provedor do serviço ganhe em flexibilidade através da elasticidade da nuvem e reduza custos. Para alguns [9] [10], a computação nas nuvens é a mudança de paradigma na infra-estrutura que permite a ascensão do Software como Serviço. A figura 4. apresenta multi-inquilinos fazendo uso de um *SaaS* cujo processamento e armazenamento é dimensionado em função da demanda, e espalhado transparentemente na nuvem de recursos.

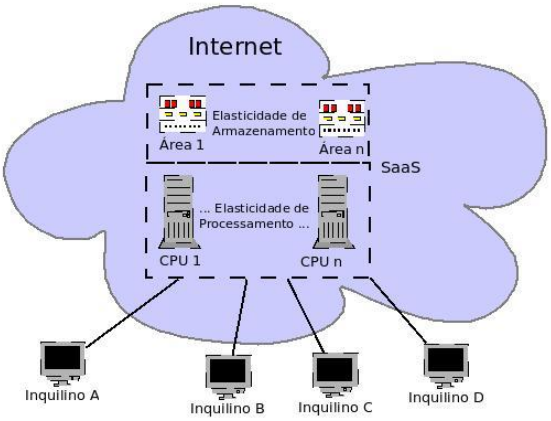

Figura 4. *SaaS* na nuvem.

4.2.1 Sobre *Infrastructure as a Service (IaaS)*

O desenvolvimento de *SaaS* sobre *IaaS* é a forma imediata de separar o serviço da infra-estrutura necessária para executá-lo. Nesse caso, o provedor do serviço terá a sua disposição uma quantidade elástica de máquinas virtuais e espaço de armazenamento para dar vazão às necessidades de sua aplicação, mas, terá de arcar com o custo de gerenciar essa infra-estrutura e terá que determinar o momento correto de prover mais ou menos recursos, alocar ou liberar máquinas virtuais. Trata-sede uma maneira menos invasiva de implementar *SaaS* sobre a nuvem, pois o provedor do serviço não fica preso ao provedor da infra-estrutura.

4.2.2 Sobre *Platform as a Service (PaaS)*

PaaS é uma das camadas da pilha arquitetural de computação nas nuvens, figura 1., que se constituí numa plataforma virtual de desenvolvimento para aplicações *Web*, hospedada num provedor, acessada geralmente via *browser*, fornecendo facilidades para editar código, depurar, implantar, executar e gerenciar *SaaS*. Diferentemente do desenvolvimento de *SaaS* sobre *IaaS*, *PaaS* abstrai para os desenvolvedores como e quais os serviços básicos (sistema operacional, linguagem de programação, API, rede) são utilizados através dos servidores da nuvem. Dessa forma, esconde toda a complexidade existente entre uma aplicação e um servidor fornecido por meio de uma infra-estrutura virtualizada. Para isso, *PaaS* oferece um conjunto de ferramentas e tecnologias: construtores de interfaces, de lógica de processos, de persistência, e de integração.

Alguns provedores de *PaaS* dão suporte para que desenvolvedores possam construir aplicações *SaaS*, não estando conectados ao provedor. Nesse caso, o ambiente virtual de desenvolvimento é instalado no computador local do desenvolvedor que poderá simular localmente o referido *PaaS*. A medida que desejar, o desenvolvedor poderá sincronizar seu desenvolvimento local com o do provedor.

PaaS devem oferecer ferramentas que possibilitem o rastreio preciso da utilização das aplicações *SaaS* sobre eles desenvolvidas, tais como: suporte para identificar quantas vezes as pessoas utilizaram a aplicação, por quanto tempo, quais funcionalidades foram utilizadas ou qual a performance obtida. Essa possibilidade de rastreio busca suportar características inerentes de *SaaS* outrora não existentes em *SaaP*.

Segundo [10], hospedar todo o ambiente de desenvolvimento aumenta a produtividade, reduz o tempo para liberação de novas releases e o custo total do software. O uso de *PaaS* elimina a necessidade dos desenvolvedores configurarem seus próprios servidores para o construção de *SaaS*, tornam os ambientes de execução escaláveis, e implementam e integram ferramentas de gerenciamento com o *SaaS* desenvolvido. Também não há preocupação com os mecanismos de armazenamento, de segurança, e sistema operacional. *PaaS* também torna transparente para os desenvolvedores o uso de interfaces de rede e de integração com outros *SaaS*. Assim, desenvolvedores se preocupam e focam apenas nas funcionalidades do *SaaS*.

5 CONCLUSÕES

O *software* oferecido como serviço representa uma importante mudança de paradigma na maneira de se entregar, negociar e desenvolver o *software*. Muitos desafios ainda precisam ser superados para *SaaS* se tornar uma unanimidade, embora pesquisas indiquem que sua expansão á algo certo e que representará um quarto do mercado de software nos próximos anos.

Devido às características do modelo de entrega e de negócio de *SaaS*, o seu desenvolvimento está repleto de preocupações sistêmicas cujos modelos de maturidade indicam o seu nível de conformidade com o modelo de referência de oferta de software como serviço.

Entre as opções de estratégias para executar o desenvolvimento de um *SaaS*, a *Ad-hoc* é adequada para grandes players da indústria que já possuem um conjunto de ferramentas desenvolvidas e que podem sofrer adequações para atender as novas demandas inerentes de *SaaS*, inclusive em todos os níveis da maturidade. A opção de utilizar a camada mais baixa da pilha arquitetural da nuvem, a *IaaS*, é adequada para aqueles desenvolvedores (forncedores) de *SaaS* que não querem ficar aprisionados aos fornecedores de infraestrutura, e querem ter a possibilidade de ou gerenciar seus próprios servidores e serviços, ou querem poder barganhar reduções financeiras na contração de serviços *IaaS*. Já o uso da camada *PaaS*, é direcionado para aqueles desenvolvedores que querem ter o maior foco possível no desenvolvimento das funcionalidade que *SaaS*, e querem fazer uso de ferramentas e de infraestrutura de provedores *de IaaS* especializados.

Uso de *PaaS* parece ser o caminho para tornar o desenvolvimento de *SaaS* uma atividade rotineira e comum quer seja para grandes *players* como para desenvolvedores individuais, no entanto, os serviços ofertados de *PaaS* ainda são muito incipientes, estão em constante evolução, e carecem de uma padronização para que serviços *PaaS* de um fornecedor possa se integrar aos serviços *PaaS* de outro.

REFERÊNCIAS

[1] _____.**Forrester's SaaS Maturity Model**, 2008, http://www.forrester.com/Research/Document/Excerpt

[2] _____.**Google AppEngine**. http://code.google.com/appengine/

[3] _____.**Oracle - A Practical Demonstration of SaaS using Oracle Application Express** http://www.oracle.com/technology/pub/articles/bobrowski-saas.html

[4] ___.**Uma introdução ao Software + Serviços, SaaS e SOA**. http://msdn.microsoft.com/pt-br/library/dd875466.aspx Acesso em: 17/05/2010.

[5] BENNETT, K.; LAYZELL, P.; BUDGEN, D.; BRERETON, P.; MACAULAY, L.; MUNRO, M. **Service-Based Software The Future for Flexible Software.** APSEC 2000.

[6] CARRARO, GIANPAOLO; CHONG, FRED. **Architecture Strategy team, LitwareHR - a fictitious HR application.** http://litwarehr.codeplex.com/. Microsoft Corp. 2010

[7] IOD**. Sotware as a Service**, Kogan Page Ltd, 2002.

[8] KWOK, T., NGUYEN, T.; LAM, L. **A Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application.** IEEE ICSEC 2008, Washington, USA.

[9] LAPLANTE, PHILLIP A.; ZHANG, J.; VOAS, J.. **"What's in a Name? Distinguishing between SaaS and SOA,**" IT Professional, May/June 2008, doi:10.1109/MITP.2008.60.

[10] LAWTON, GEORGE. **"Developing Software Online With Platform-as-a-Service Technology,"** Computer, vol. 41, no. 6, pp. 13-15, June 2008, doi:10.1109/MC.2008.185

[11] LENK, A.; KLEMS, M.; NIMIS, J.; TAI, S.; SANDHOLM, T.. **What's Inside the Cloud? An Architectural Map of the Cloud Landscape.** ICSE'09 CLOUD'09, May 2009, Vancouver, Canada.

[12] MOORE, B.; MAHMOUD, QUSAY H. **"A service broker and business model for saas applications,"** pp.322-329, 2009 IEEE/ACS Intern. Conf. on Computer Systems and Applications, 2009.

[13] YOUSEFF, L.; BUTRICO, M.; DA SILVA, D., "**Toward a Unified Ontology of Cloud Computing**," Grid Computing Environments Workshop, 2008. GCE '08 , vol., no., pp.1-10, 12-16 Nov. 2008.