Proposta de uma plataforma de Cloud Computing para disponibilização de um sistema online para consultórios e clínicas por meio do modelo SaaS

Robertson Ebling dos Santos[1], **Alexandre de Oliveira Zamberlan**[1], **Sylvio André Garcia Vieira**[1]

[1]Sistemas de Informação – Centro Universitário Franciscano
Conjunto I – Rua dos Andradas, 1614 – 97.010-032 – Santa Maria – RS

robertson@ersistemas.info, alexz@unifra.br, sylvio@unifra.br

Abstract. *In an increasingly connected world it is common in corporate environments using applications installed on remote servers that centralize company data and allow user access via the Internet. In a doctor's office system high availability allows quick access to patient records which speeds diagnosis and initiation of treatment. The goal of this paper is to propose and install an architecture of Cloud Computing based on the characteristics proposed by NIST (National Institute of Standards and Technology) and using free software implements high availability and horizontal scalability needed for the provision of an online system for management's offices and clinics through the SaaS model.*

Resumo. *Em um mundo cada vez mais conectado, é comum em ambientes corporativos a utilização de aplicativos instalados em servidores remotos que centralizam os dados da empresa e permitem o acesso dos usuários através da Internet. Em um consultório médico, a alta disponibilidade do sistema permite o acesso rápido ao prontuário do paciente que agiliza o diagnóstico e o início do tratamento. Assim o objetivo deste trabalho é propor e instalar uma arquitetura de Cloud Computing baseada nas características propostas pelo NIST (National Institute of Standards and Technology) que utilizando software livres implemente a alta disponibilidade e escalabilidade horizontal necessários para a disponibilização de um sistema online para gestão de consultórios e clínicas por meio do modelo SaaS.*

1. Introdução

Segundo uma pesquisa realizada nos Estados Unidos [Informédica 1994], um médico gasta quase metade do seu tempo para obter e registrar informações acerca de seus pacientes. Fora isto, existe o tempo necessário para que o auxiliar/recepcionista controle a agenda, o cadastro de informações administrativas e financeiras sobre os pacientes. Dentro deste quadro, em um estudo realizado em 2014 (por meio de uma consulta realizada com 100 médicos do corpo clínico do Hospital de Caridade Dr. Astrogildo Azevedo de Santa Maria RS), foi identificada a alta taxa (cerca de 70%) de utilização de métodos tradicionais baseado na ficha e agenda - manual em papel. No mesmo estudo, foi percebido que os resultados apontados por [Informédica 1994] sobre a alocação do tempo deste profissional condizem com a atualidade.

Um sistema de gestão de consultório instalado localmente auxilia no controle destes aspectos, porém traz a preocupação com manutenção do *software* que fica por conta e risco do profissional de saúde. Em [Taurion 2009], cita-se que alguns estudos têm mostrado que empresas de pequeno porte - consultórios - gastam até 70% do seu tempo gerenciando os recursos de TI (algo que não gera valor agregado) e apenas 30% em atividades focadas no seu próprio negócio. Por outro lado, um *software* baseado em *Cloud Computing* permite centralizar os dados e disponibilizá-los em qualquer lugar com acesso à Internet. Também, libera o profissional das preocupações operacionais e técnicas que ficam, agora, de responsabilidade da empresa prestadora do SaaS (*Software as a Service*).

Assim, surge o objetivo deste trabalho, projetar, implementar e testar uma infraestrutura de *Cloud Computing* no contexto do modelo SaaS para disponibilização de um sistema Web de gestão de consultórios e clínicas.

Para atingir o objetivo geral deste trabalho, foi fundamental realizar algumas ações importantes, como: definir uma arquitetura para a infraestrutura de *Cloud Computing* utilizando ferramentas baseadas em *software* livre; instalar, configurar e conectar tecnologias; testar o escalonamento horizontal através do Balanceamento de Carga de uma aplicação Web desenvolvida com Python/Django utilizando o Servidor Web Nginx, o banco de dados Postgres em *Cluster* com Distribuição de carga e o escalonamento de *storage* com o GlusterFS.

O presente artigo é apresentado em seções. Na revisão teórica, apresenta-se a definição e características de *Cloud Computing*, bem como seus modelos de serviço e implantação da tecnologia. Além disso, alguns trabalhos relacionados são apresentados. Na seção de proposta, apresenta-se a metodologia de trabalho, o sistema ER Clinic que é objeto do estudo de caso e a definição, implantação e teste da estrutura e tecnologias empregadas na plataforma. Finalmente, seguem as conclusões e as referências bibliográficas.

2. Revisão teórica

Nesta seção, é discutido o conjunto de tecnologias que auxiliam na compreensão da implementação do modelo SaaS (*Software as a Service*) que disponibiliza aplicações baseadas na Web como serviço.

2.1. Cloud computing

Um estudo [CETIC 2014] aponta que 74% das empresas que utilizam computador já oferecem alguma forma de compartilhamento de recursos através da Internet. Conforme [Gartner 2013], é evidenciado que a adoção de sistemas baseados na Web vêm crescendo de maneira acelerada em todo mundo. A estimativa prevê ainda que dentro do nicho de sistemas de escritório baseados na Web a quantidade de usuários passará de 4 milhões em 2015 para 16,5 milhões até 2017. Estima-se também que este número cresça para cerca de 695 milhões em 2022, representando 60% de usuários de sistemas de escritório.

Percebe-se através desses estudos que existe a necessidade de uma infraestrutura flexível que proporcione uma forma indefinida de recursos de processamento, memória e armazenamento. Além disso, que responda ao aumento indeterminado na demanda por volume de acessos em sistemas baseado na Web. Portanto, acredita-se que *Cloud Computing* surge como um novo paradigma para esse cenário de impresivibilidade da demanda.

Empresas *startups*[1] que possuem como característica o baixo custo inicial de manutenção e que precisam escalar proporcionalmente ao aumento da demanda do mercado pelos seus serviços e produtos, encontram nesse modelo de computação uma excelente ferramenta para acompanhar a evolução de seus negócios.

De acordo com [Taurion 2009], *Cloud Computing* pode ser definido como um conjunto de recursos com capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na Internet.

Segundo [Cordeiro 2013], existem quatro problemas que (ainda) requerem constante inovação tecnológica e que a Computação em Nuvem serve como ferramenta integrante da solução, são eles: escalabilidade em Web, *data centers* distribuídos geograficamente, computação paralela e distribuída e aplicações Web interativas.

O Departamento Nacional de Comércio dos Estados Unidos por intermédio do *National Institute of Standards and Technology* (NIST) define *Cloud Computing* como um modelo projetado para permitir o acesso à rede ubíqua[2] sob demanda (*on-demand*) para um compartilhamento de *pool* de recursos computacionais[3] configuráveis como por exemplo: redes, servidores, armazenamento, aplicações e serviços [NIST 2011]. Ademais, esses recursos podem ser rapidamente fornecidos e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços.

Características essenciais de um Cloud Computing

Serviços sob demanda com capacidade de auto atendimento [4] possibilitam que o cliente final possa a qualquer momento requerer maior ou menor quantidade de recursos computacionais que são oferecidos de forma automática, sem necessidade de interação humana por parte do prestador de serviço. Acesso multi-dispositivo significa que a nuvem deve ser o ponto de acesso de recursos para seus usuários. Esses recursos disponibilizados por meio da Internet devem estar disponíveis para dispositivos computacionais padrão (PCs, laptops, smartphones, tablets e etc). *Pool* de recursos pode ser entendido como recursos computacionais físicos e virtuais que são organizados de maneira distribuída para atender de forma transparente os múltiplos usuários. Define-se como rápida elasticidade a capacidade em que o sistema tem de adicionar e/ou remover recursos computacionais em tempo de execução. Essa característica fornece a impressão para o usuário final de que os recursos parecem ser ilimitados. Por fim, monitoramento é o mecanismo de cobrança de utilização de recursos da nuvem de acordo com o consumo.

Modelos de serviço

Conforme [World 2010], existem 11 categorias principais de tecnologia para Computação em Nuvem: Armazenamento como Serviço (AaaS), Banco de Dados como Serviço

[1] Empresa nova, inovadora, que conta com projetos promissores e normalmente pouco recurso financeiro.

[2] Integra mobilidade a computação pervasiva onde o computador está embarcado no ambiente de forma invisível para o usuário [Araújo 2003].

[3] Conglomerado de computadores, *software* e rede de computadores que com a réplica de instâncias são capazes de distribuir o trabalho e assumir o trabalho de partes que estão momentaneamente indisponíveis

[4] *self-service*

(BaaS), Informação como Serviço, Processo como Serviço, *Software* como Serviço (SaaS), Plataforma como Serviço (PaaS), Integração como Serviço, Segurança como Serviço, Gestão/Governança como Serviço, Teste como Serviço e Infraestrutura como Serviço (IaaS).

IaaS é a camada que representa a base física que dá suporte de *hardware* para as demais camadas. Ademais, permite o fornecimento de recursos como alocação de *datacenter*, rede de energia e dados e servidores. Exemplos de fornecedores desse serviço: Amazon Elastic Cloud Computing (EC2)[5], Microsoft Azure[6], Digital Ocean[7] e Locaweb[8].

PaaS é a camada que provê aos desenvolvedores de aplicação o acesso de uma forma regular, estruturada e mais prática à camada de infraestrutura. Além disso, permite ações como compilar, desenvolver, monitorar, depurar e testar. Sistemas operacionais, sistemas gerenciadores de banco de dados, interfaces de programação de aplicações (APIs), sistemas de armazenamento em disco e demais aplicativos que dão suporte para a aplicação principal da arquitetura são características do que este modelo de serviço oferece. Enquadram-se como fornecedores desse serviço: Heroku[9] e Google Cloud Platform[10].

SaaS é o modelo que disponibiliza por meio da Internet sistemas com propósitos específicos para o usuário consumidor. O usuário que contrata esta camada envolve-se apenas com configurações da aplicação. Percebe-se que este modelo faz com que o cliente pague somente pelo uso, manutenção e suporte técnico prestados. Serviços como Google Docs[11], Gmail[12] e Dropbox[13] são oferecidos como SaaS.

Modelos de implantação

A implantação do *Cloud Computing* depende da necessidade da aplicação que executará sobre este meio. Aparentemente os serviços são disponibilizados de forma pública, mas foram desenvolvidos modelos de implantação que procuram garantir um nível adequado de controle da informação.

Os modelos de implantação apontados por NIST são: *Nuvem Privada, Nuvem Comunidade, Nuvem Pública e Nuvem Híbrida*.

2.2. Escalabilidade e alta disponibilidade em Nuvens

Garantir o retorno das requisições dos usuários ainda é um desafio para sistemas de alta demanda. Sistemas baseados em *Cloud Computing* devem ter a capacidade de adicionar e/ou remover recursos computacionais em tempo de execução deixando este procedimento totalmente transparente para o usuário final.

Em [Taurion 2009], destaca-se a imprevisibilidade da demanda como um complicador adicional ao ambiente de empresas cada vez mais interconectadas. Para o autor,

[5]https://aws.amazon.com/pt/ec2
[6]https://azure.microsoft.com/pt-br
[7]https://www.digitalocean.com
[8]http://www.locaweb.com.br/cloud
[9]https://www.heroku.com
[10]https://cloud.google.com/appengine
[11]https://docs.google.com
[12]https://www.gmail.com
[13]https://www.dropbox.com

esta demanda imprevisível exige que os sistemas tenham condições de adaptar-se instantaneamente a flutuações significativas.

Já em [Cipriani 2009], justifica-se a extrema importância da garantia do funcionamento contínuo de sistemas para que uma empresa informatizada opere em sua plenitude. Sua indisponibilidade, seja por falha do *hardware* ou do *software*, implica direta ou indiretamente em perda de dinheiro por parte da organização.

Três aspectos de alta disponibilidade relacionados ao sistema do estudo de caso deste trabalho são apresentados: balanceamento de carga, banco de dados clusterizado e escalonamento de armazenamento.

Balanceamento de carga, segundo [Bourke 2001], busca obter a distribuição de requisições de maneira equitativa sobre os nós de um ambiente distribuído. As requisições chegam no balanceador de carga, que por meio do algorítimo de balanceamento define a prioridade pelo servidor de destino ao qual a requisição é encaminhada. Para o usuário final, esta arquitetura é vista como apenas um servidor - dando a impressão de ser apenas um grande servidor virtual. Em [Roger 2012], é apresentado um balanceador de carga com alta disponibilidade utilizando Nginx como *proxy* reverso e o Apache[14] como servidor de aplicação HTTP nos nós secundários. Já em [de Moura Nóbrega 2013] o Nginx é destacado como um balanceador de carga largamente empregado pelo seu excelente desempenho e facilidade de configuração.

Banco de dados clusterizado entende-se como um recurso em que se o banco de dados principal tiver algum problema, outro banco de dados secundário assume automaticamente, sem prejudicar o serviço, deixando transparente para o usuário final. Bancos de dados como Oracle[15], SQLServer[16], MySQL[17] e Postgresql[18] são exemplos de soluções compatíveis com a clusterização.

Escalonamento de armazenamento (*storage*) , é a tecnologia capaz de aumentar o espaço em disco horizontalmente - sem a necessidade de interferir na execução do sistema. São exemplos de soluções de alocação dinâmica de espaço em disco, sem a necessidade de parada no funcionamento do sistema, Amazon EFS (Elastic File System)[19] e o GlusterFS[20] Este último possibilita criar soluções de armazenamento grandes e distribuído usando *hardware* comum - máquinas físicas ou virtuais.

Finalmente, as técnicas apresentadas na seção têm como principal objetivo garantir a disponibilidade contínua do sistema, evitando paradas para manutenção ou ampliação da capacidade de alocação de espaço e processamento da plataforma.

2.3. Trabalhos relacionados

No trabalho [Boufleur 2013], foram aplicadas técnicas de clusterização para a otimização da performance e promoção da alta disponibilidade do *software* SIGA-ECPT[21] (utilizado

[14]https://httpd.apache.org
[15]http://www.oracle.com
[16]http://www.microsoft.com/pt-br/server-cloud/products/sql-server/Overview.aspx
[17]https://www.mysql.com/products/enterprise/router.html
[18]http://www.postgresql.org
[19]https://aws.amazon.com/pt/efs
[20]http://www.gluster.org
[21]Sistema Integrado de Gestão Acadêmica da Educação. [Boufleur 2013]

na Universidade Federal de Santa Maria). A ferramenta GlassFish[22] foi utilizada como *cluster* de servidores de aplicação para garantir a escalabilidade horizontal da solução. Ao migrar o sistema SIGA-ECPT do modo sistema com único servidor (*standalone*) para o clusterizado, foram realizados alguns ajustes nos arquivos de configuração do sistema no GlassFish. A fim de corrigir erros que alertavam incompatibilidade com as demandas de compartilhamento de sessão do *software* clusterizado, mudanças no código-fonte do *software* SIGA-ECPT também foram necessárias. Com a estrutura proposta, conseguiu-se alta disponibilidade através do balanceamento de carga entre os nós. Ao contrário do esperado, percebeu-se através de testes que o tempo de execução do método clusterizado foi maior do que um servidor *standalone*.

Em relação ao trabalho apresentado em [de Moura Nóbrega 2013], realizou-se um estudo a fim de determinar qual a melhor configuração de um ambiente no *data center* da Universidade Estadual do Ceará[23], utilizando computação em nuvem para atender as demandas dos sistemas Web da instituição. Destaca-se a forma como os testes foram conduzidos, como por exemplo, a análise de diferentes soluções em nuvem com plataformas pública e privada, interessantes para este estudo. Ainda com relação aos testes de carga, foram comparados os resultados obtidos com o emprego dos servidores Web - Nginx e Apache - e os servidores de aplicação - GlassFish e Tomcat (tecnologias Java) - em cinco diferentes fornecedores de ambiente de nuvem. Nesse mesmo trabalho, definiu-se o uso do *software* OpenStack[24] para gerenciar os recursos de nuvem privada juntamente com o *cluster* do GlassFish e o balancemento de carga do Nginx.

Já no trabalho [Roger 2012], foi desenvolvido um ambiente de alta disponibilidade utilizando Nginx como balanceador de carga entre os nós, Apache como servidor Web, RSYNC (*software* que sincroniza remotamente os dados entre duas máquinas) como replicador de arquivos entre os nós do *cluster* e compartilhamento de sessão com PHP. Nessa proposta, destaca-se pela eliminação do Ponto único de Falha (*SPOF, Singe Point of Failure*) replicando também os balanceadores de carga através da utilização de *failover*[25] e *keepalive*[26].

Na Figura 1, são apresentadas as tecnologias empregadas na solução de problemas correlacionados a esta proposta.

É possível identificar nos trabalhos citados, que o foco principal está na garantia da escalabilidade e alta disponibilidade dos servidores de aplicação. Entretanto, nota-se uma lacuna quanto aos aspectos de escalabilidade e alta disponibilidade para armazenamento em disco e servidor de banco de dados.

Sendo assim, na próxima seção é apresentada o projeto que pretende gerar a escalabilidade e alta disponibilidade de servidores de aplicação, armazenamento em disco e banco de dados.

[22]https://glassfish.java.net

[23]http://www.uece.br

[24]https://www.openstack.org

[25]Técnica que evita o ponto único de falha. [Roger 2012].

[26]Permite detectar a inoperância de um servidor e atribuir virtualmente o IP do servidor inoperante ao substituto [Roger 2012].

Trabalhos relacionados	Sistema Operacional	Tecnologia (Linguagem de Programação)	Servidor Web	Método de Cache	Balanceamento de Carga de servidores de aplicação	Compartilhamento de sessão	Banco de Dados	Servidor de dados (storage)
[Boufleur 2013]	Linux / Ubuntu	Java	Apache	-	GlassFish / Oracle iPlanet Web Server 7	GlassFish	PostgreSQL não clusterizado	-
[de Moura Nóbrega 2013]	-	Java	Nginx	-	GlassFish / Nginx / OpenStack	GlassFish	-	-
[Roger 2012]	Linux / Debian	PHP	Apache	-	Nginx Proxy	Memcached / Repached	-	Duplicado entre os nós através de RSYNC
Proposta do trabalho	Linux / Ubuntu	Python/Django	Nginx	Varnish	Nginx Proxy	Django Redis Sessions	Clusterizado (Pgpool-II / PostgreSQL)	Clusterizado (GlusterFS)

Figura 1. Quadro comparativo entre a proposta deste trabalho e as soluções propostas nos trabalhos relacionados

3. Proposta

A seção tem como objetivo descrever a proposta de uma arquitetura para *Cloud Computing* baseada em *software* livre. Além disso, apresentar o estudo de caso do sistema de gestão para consultórios e clínicas e avaliar se o conjunto (sistema mais arquitetura) atenderão os aspectos de: auto atendimento sob-demanda (*self-service*), acesso multi-dispositivo, escalabilidade (rápida elasticidade) e monitoramento. Por fim, é apresentado o emprego das tecnologias como solução do seu referido aspecto no contexto do *Cloud Computing* e apresentado o resultado dos testes avaliativos aplicados na infraestrutura.

3.1. Metodologia de Trabalho

Com o objetivo de atender as diversas áreas nas quais este trabalho se correlaciona - balanceamento de carga de servidores de aplicação, escalabilidade em servidores de sistema de arquivos e alta disponibilidade em servidores de banco de dados - realizou-se pesquisa bibliográfica e projetou-se estudo de caso. Essas duas atividades definiram quatro etapas de investigação e implementação, que são mostradas no diagrama de atividades da Figura 2. Essa metodologia foi construída/pensada uma vez que não foi encontrado na literatura algum recurso que contemplasse os requisitos do presente trabalho.

3.1.1. Descrição das atividades

- ***Levantamento de dados:*** Identificação do problema - ofertar o sistema discutido na subseção 3.2, aplicando o modelo *Cloud Computing*/SaaS. Estudo bibliográfico de Computação em Nuvem identificando e classificando pontos importantes para a solução do problema e levantamento de trabalhos correlacionados também foram previstos nesta etapa;
- ***Estudo de soluções:*** Levantamento de soluções comerciais e ferramentas livres que atendem a demanda de *Cloud Computing*;
- ***Criação e implantação da proposta:*** Criação do projeto de uma arquitetura que objetiva atender os pré-requisitos de um ambiente de Computação em Nuvem. Nesta etapa está previsto a instalação e configuração dos itens pertencentes a arquitetura;
- ***Testes e ajustes finais:*** Os testes foram previstos com intuito de identificar possíveis falhas e pontos críticos da plataforma. Detalhes dos testes são mostrados na Subseção 3.5.

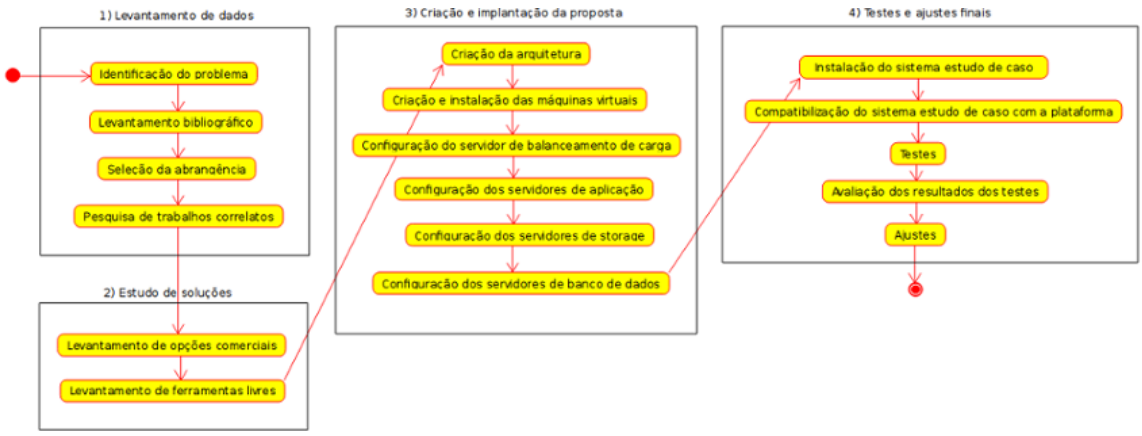

Figura 2. Diagrama de atividades da metodologia de trabalho proposta

3.2. Estudo de caso

A partir de um sistema Web para gestão de consultórios e clínicas, pretende-se avaliar a infraestrutura proposta na Subseção 3.3. O ER Clinic[27] é um *sofware* baseado na Web projetado para gestão ágil de consultórios e clínicas. O sistema foi desenvolvido pela ER Sistemas[28] como um serviço, visando o auto atendimento, em que o usuário monitora os recursos que estão sendo utilizados. O formato do sistema é *SaaS*.

Na Figura 3 é apresentada a tela principal do sistema ER Clinic e suas principais funcionalidades são:

- Cadastro de clientes;
- Agendamento de consultas com confirmação via SMS;
- Prontuário do paciente;
- Chat entre os profissionais da área de saúde e assistentes do consultório;
- Controle financeiro;
- Controle de estoque.

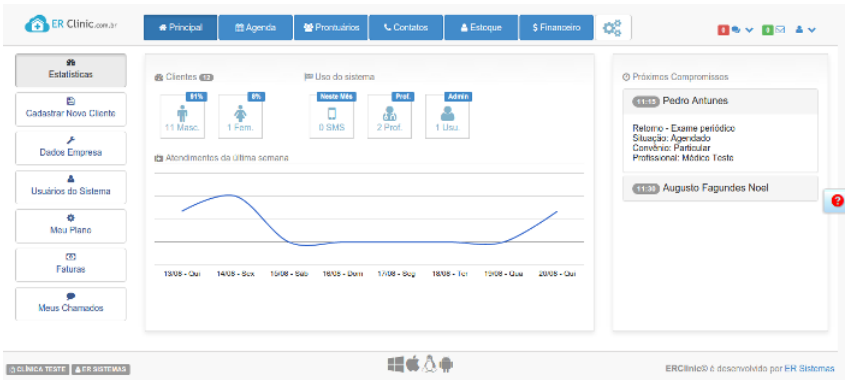

Figura 3. Estudo de caso: Sistema online para gestão de clínicas e consultórios - Dashboard

[27] https://erclinic.com.br
[28] http://www.ersistemas.info

Dentre as diversas tecnologias empregadas no desenvolvimento da ferramenta, destacam-se: Linguagem de programação Python[29] com o *framework* Django[30]; Node.js (https://nodejs.org) como Interpretador de código Javascript no lado do servidor; Sistema gerenciador de banco de dados Postgresql[31]; *Framework* de *frontend* Bootstrap[32].

3.3. Infraestrutura de Cloud Computing proposta

O desafio de projetar uma arquitetura que atenda as demandas de *Cloud Computing*, siga os critérios do SaaS e ainda não dependa de soluções prontas comercializadas, oportunizou o estudo de algumas ferramentas livres que podem garantir isoladamente os requisitos de alta disponibilidade e rápida elasticidade.

Para eleger as ferramentas utilizadas na arquitetura proposta foram seguidos os seguintes critérios: uso de *software* livre, disponibilidade de uma ampla documentação e fácil acesso na comunidade envolvida com a ferramenta.

Optou-se pelo programa Nginx como servidor *proxy* reverso[33] pela facilidade de configuração de rotas baseadas na URL (*Uniform Resource Locator*). Assim, as requisições são classificadas de acordo com o caminho da URL e encaminhada para o balanceamento de carga entre os servidores responsáveis pela requisição. Também o Nginx trabalha como ferramenta que disponibiliza o certificado digital SSL (*Secure Sockets Layer*) para ser homologado no órgão responsável, permitindo que o tráfego entre o cliente final e a plataforma seja utilizando conexão segura através do protocolo HTTPS (*Hypertext Transfer Protocol Secure*).

O Nginx também foi planejado para servir e fazer cacheamento de arquivos estáticos como imagens, CSS (*Cascading Style Sheets*) e Javascript além de servir o conteúdo dinâmico encaminhado pelo Gunicorn[34], como resultado do processamento realizado pelo Python/Django.

Como solução de armazenamento distribuído, foi planejada a implantação de um *cluster* de servidores utilizando GlusterFS conforme foi apresentado na Subseção 2.2.

Para atender a alta disponibilidade e escalabilidade do banco de dados, optou-se pela implantação do Postgresql com o *middleware* PgPoolII, cuja descrição e implantação foi apresentada no item da Subseção 2.2.

Decidiu-se por servidores virtuais privados para a instalação da plataforma proposta. Para a seleção de fornecedor, foram utilizados alguns critérios: qualidade do serviço, suporte e investimento mensal por servidor. Desta forma, escolheu-se a empresa americana Digital Ocean[35], onde serão instaladas instâncias virtuais com o Sistema Operacional Linux/Ubuntu 14.04 LTS de 64bits.

Alguns cuidados extras foram necessários ao implantar o *cluster* de servidores de aplicação no sistema do estudo de caso. Como haverá o balanceamento de carga, é

[29] https://www.python.org

[30] https://www.djangoproject.com/

[31] http://www.postgresql.org

[32] http://getbootstrap.com

[33] No contexto do trabalho, é um servidor que intermedia as requisições, fazendo o balanceamento de carga entre os servidores Web da plataforma. É a camada externa da aplicação.

[34] http://gunicorn.org

[35] https://www.digitalocean.com

possível que um mesmo usuário possa ser atendido por servidores de aplicação distintos. Neste caso, foi necessário a implantação de alguma tecnologia que compartilhe os dados de sessão entre os servidores de aplicação, para que não se perca nenhum dado do usuário logado no sistema.

A Figura 4 mostra de maneira ampla a infraestrutura proposta.

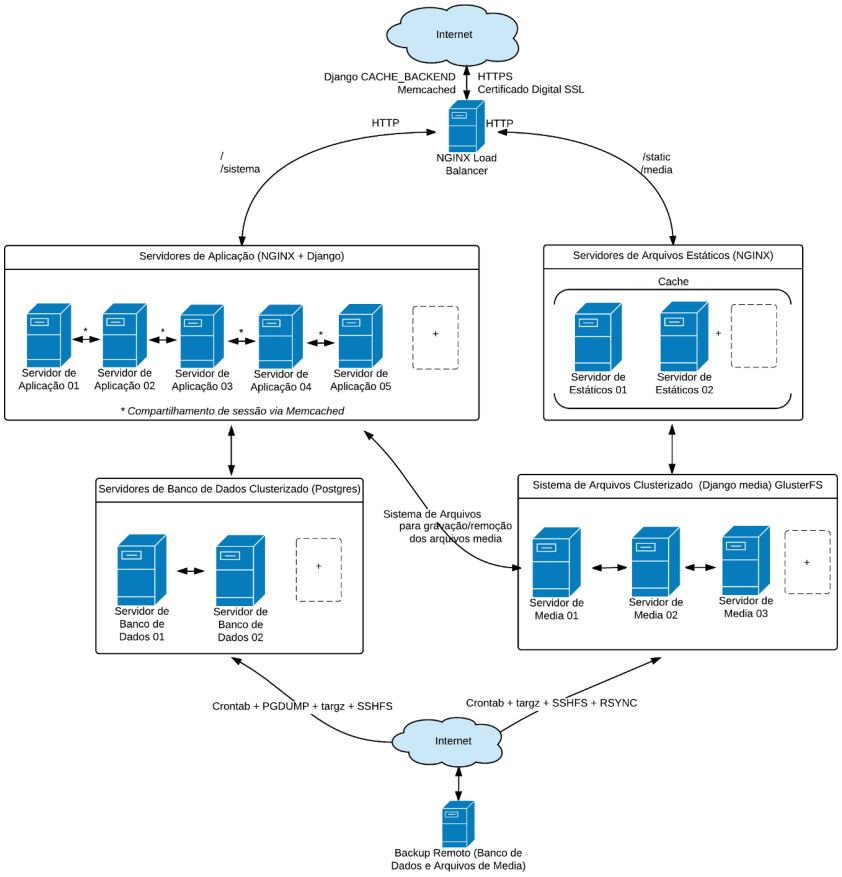

Figura 4. Diagrama estrutural

Um exemplo de atendimento de uma requisição HTTP/HTTPS atendida pela plataforma é mostrado no diagrama de atividades da Figura 5.

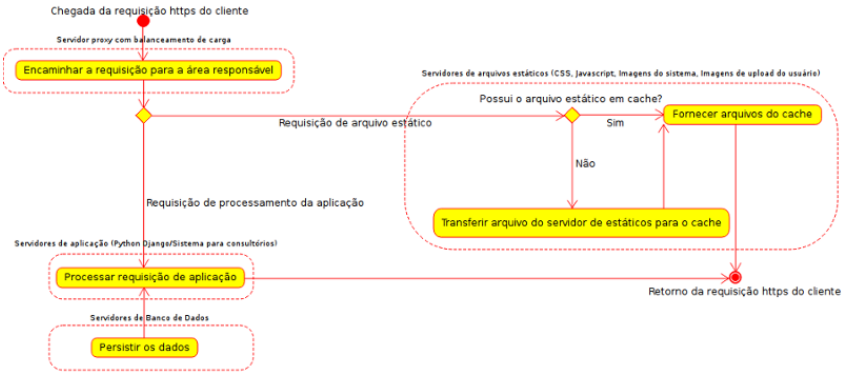

Figura 5. Diagrama de atividades - Requisições HTTPS/HTTP

3.4. Implantação

A arquitetura resultante deste estudo implanta uma nuvem do tipo comunidade, caracterizado pelo atendimento de um grupo de diferentes empresas - consultórios e clínicas - que têm preocupações comuns como requisitos de segurança, regras de negócio e política, dando suporte a alta disponibilidade e escalabilidade do sistema *online* para gestão de consultórios e clínicas.

3.4.1. Balanceador de carga e servidores de aplicação

Criou-se uma instância de máquina virtual com o sistema operacional Linux responsável pelo balanceamento de carga entre os servidores de aplicação, tunelamento HTTPS com certificado digital SSL e a disponibilização e cache de arquivos estáticos. A simples configuração do Nginx como proxy, proporcionou:

- Redirecionamento de requisições http para https, forçando que o dado trafegado entre o cliente e o *cluster* de servidores seja obrigatoriamente criptografado;
- Redirecionamento de requisições de arquivos estáticos para serem atendido pelos servidores específicos deste tipo de arquivo (CSS, JS e imagens);
- Disponibilização do certificado digital para que a entidade certificadora confirme a validade e confiança do servidor;
- Balanceamento entre os servidores de aplicação.

Instâncias de máquinas virtuais com o sistema operacional Linux foram criadas para operar como servidores de aplicação. Um servidor de aplicação roda o núcleo de processamento do sistema apresentado no estudo de caso - subseção 3.2 - que foi desenvolvido em Python utilizando o *framework* Django.

Outras máquinas virtuais de servidores de aplicação podem ser criadas a partir de uma imagem (réplica) de uma máquina virtual configurada para o mesmo fim. Desta forma, sempre que é aumentado a demanda por processamento na aplicação, rapidamente novos servidores podem ser criados e colocados para trabalhar recebendo o redirecionamento de requisições originadas do balanceador de carga. Tem-se desta forma uma escalabilidade horizontal, dando recurso suficiente para atender um número crescente de requisições.

Para cada novo nó de servidor de aplicação acrescentado na plataforma, torna-se necessário adicionar também na configuração do Nginx proxy. Esta parametrização não representa prejuízo, uma vez que o Nginx permite recarga da configuração sem que seja necessário a parada do serviço.

A utilização do balanceamento de carga em servidores Web, que rodam sistemas que dependem de dados armazenados em sessão, demanda de um mecanismo de compartilhamento desses dados. Recomenda-se a leitura da subseção 4 que contém o resultado de dois métodos usados na implantação da infraestrutura proposta neste trabalho.

3.4.2. Banco de dados clusterizado

Para a disponibilização do banco de dados em *cluster*, criou-se 3 novas instâncias virtuais de máquina Linux, sendo que uma delas é responsável pela execução do PGPoolII, que

recebe a conexão de banco oriunda dos servidores de aplicação e redireciona para as duas máquinas servidoras de banco de dados Postgres, promovendo o balanceamento e a replicação dos dados entre os servidores.

A aplicação se conecta sempre que necessário com o PGPoolII, nunca direto nos servidores de Postgres. No caso de indisponibilidade de um dos servidores o PGPoolII promove o desvio para apenas o servidor ativo, sem que a conexão entre o servidor de aplicação e o banco de dados seja perdida.

Percebeu-se durante os testes, que ao retornar um servidor Postgres de uma parada crítica, não houve a sincronização dos dados entre o servidor que estava ativo com o servidor que sofreu a parada. Para que houvesse a sincronia dos dados, um procedimento manual foi necessário, recomenda-se a leitura da subseção 4 como referência dos procedimentos executados para contornar esta situação.

3.4.3. Escalabilidade horizontal do armazenamento em disco

Adquiriu-se duas novas instâncias de máquinas virtuais Linux, com o objetivo de comportar os arquivos oriundos de *upload* realizados pelos usuários da aplicação. Estes computadores comportam-se como um grande disco virtual que é aumentado a cada novo computador adicionado ao *cluster*.

Utilizou-se a tecnologia GlusterFS para que este espaço seja identificado como um todo na rede. Os servidores de aplicação montam remotamente a unidade virtual do *cluster* de armazenamento em disco que é utilizada para que a cada adição ou remoção de arquivos seja percebida em tempo real por qualquer um dos servidores de aplicação disponíveis na plataforma.

3.5. Avaliação

Percebe-se que a proposta apresentada engloba diversos aspectos do *Cloud Computing* como: distribuição de carga e alta disponibilidade entre os servidores de aplicação, replicação e distribuição de carga entre servidores de banco de dados e a alta disponibilidade e escalabilidade horizontal de armazenamento em disco.

Optou-se pela aplicação de testes avaliativos para identificar se a plataforma atende os requisitos de *Cloud Computing*. O quadro da Figura 6 mostra os procedimentos de testes adotados durante o período de avaliação, com seus respectivos resultados alcançados.

Avaliação	Teste executado	Resultado obtido
Alta disponibilidade	Servidores de aplicação são desligados aleatoriamente, manualmente.	Outro servidor pertencente ao cluster de servidores de aplicação assume a requisição. Confirmando o funcionamento do balanceador de carga do servidor NGINX Proxy.
	Servidores de banco de dados são desligados aleatoriamente, manualmente.	O servidor PGPoolII identifica o timeout do servidor Postgres e redireciona o tráfego apenas para o servidor ativo. A sincronização dos dados entre o servidor que sofreu a parada crítica e o servidor ativo não ocorreu de forma automática, conforme o esperado. Detalhes na subseção 4.
Rápida elasticidade	Servidores de aplicação são criados a partir de imagens de outros servidores de aplicação e são adicionados ao cluster.	Rapidamente a plataforma aumenta a sua capacidade de processamento e memória recebendo o novo servidor como parte integrante do cluster.
	Aumento do espaço de armazenamento em disco ao adicionar um novo servidor responsável pelo armazenamento.	Os servidores de aplicação têm seu espaço para armazenamento em disco aumentado pronto para receber novos uploads de arquivos.
Teste de carga	Teste manual de operação com o sistema (20 usuários simultâneos).	Identificou-se o rápido esgotamento da memória RAM do servidor Balanceador de Carga que na configuração inicial mantinha o compartilhamento dos dados da sessão HTTP compartilhado entre os servidores de aplicação. Problema solucionado ao substituir o método de armazenamento em memória RAM para Banco de Dados. Leia na subseção 4.
	Teste automatizado simulando 20 usuários simultâneos executando 6,38 requisições por segundo. Tempo de teste: 15 minutos.	Obteve-se como resultado: 0% de respostas HTTP com erro; Média de tempo de resposta em 145,29 ms sendo que em 90% dos casos a média de tempo de resposta foi de 133 ms.

Figura 6. Avaliação da plataforma

4. Conclusões

Procurou-se, por meio deste trabalho, apresentar uma proposta de solução para promover a alta disponibilidade e rápida elasticidade em sistemas baseados na Web. Para isto, manteve-se a ideia de alinhar a pesquisa em torno das definições de *Cloud Computing* propostas pelo *National Institute of Standards and Technology*.

Percebe-se, durante a revisão bibliográfica, que soluções robustas de *Cloud Computing* comercializadas pela Amazon e Microsoft Azure resolvem o problema, sem a necessidade de configurações mais detalhadas, como as exigidas pela arquitetura proposta neste trabalho.

Entretanto, o foco da solução definida é evidenciar através de um trabalho acadêmico as preocupações que a Computação em Nuvem envolve e dar mais uma opção de solução. Além disso, a plataforma sugerida procura atender os requisitos de empresas *startups* que buscam o rápido e incremental amadurecimeto com um baixo investimento inicial.

Em [Taurion 2009], destaca-se a importância da utilização de *Open Source* em soluções de arquitetura em *Cloud Computing*. Dentre as principais vantagens apontadas pela utilização do *Open Source*, destaca-se a eliminação de dependências críticas que afetam a entrega de serviços. Também o autor acredita que quanto mais a Computação em Nuvem utilizar *Open Source*, mais maduros e evoluídos estes sistemas se tornarão, amplificando seu uso.

4.1. Resultados

A plataforma desenvolvida atendeu os requisitos da Computação em Nuvem, disponibilizando uma aplicação oferecida no modelo SaaS, com capacidade de ampliação de sua capacidade sem necessidade da parada no sistema. Contudo, alguns comportamentos inesperados ocorreram. O compartilhamento dos dados de sessão utilizada pelos servidores de aplicação deu-se inicialmente pela configuração do serviço de *memcached* em memória RAM instalado no servidor Nginx *proxy* - balanceador de carga. Identificou-se o rápido esgotamento da memória RAM do balanceador de carga a cada novo usuário conectado simultaneamente no sistema ao utilizar este método. Para resolver este problema, configurou-se o *framework* Django para que os dados de sessão fossem compartilhados via *memcached*, porém em banco de dados.

A sincronização de dados entre os servidores Postgres após uma parada crítica em um dos servidores não ocorreu automaticamente, como havia sido esperado. A configuração do PGPoolII deverá ser revista, entretanto, para contornar a situação, fez-se necessário executar o sincronismo manual entre os dados dos servidores de banco de dados por meio de exportação/importação. O procedimento manual não ocasionou nenhuma parada do sistema.

Como trabalhos futuros, recomenda-se soluções que removam os pontos únicos de falha. Na estrutura proposta neste trabalho, pode-se mencionar como ponto único de falha o servidor balanceador de carga Nginx *proxy* e o PGPoolII que faz o balanceamento de carga entre os servidores de banco de dados. Uma solução para estes "gargalos"é a duplicação destes dois servidores e a implantação de IPs flutuantes, disponibilizados pela própria Digital Ocean, que por meio da técnica de *failover* redireciona o tráfego de rede para um servidor secundário, caso o primário apresente alguma falha.

Referências

Araújo, R. B. (2003). Computação ubíqua: Princípios, tecnologias e desafios. In *Computação Ubíqua: Princípios, Tecnologias e Desafios*, volume 1, pages 1–71. SBC, 1 edition.

Barbosa, F. P. and ao, A. S. C. (2010). Grid computing e cloud computing - uma relação de diferenças, semelhanças, aplicabilidade e possibilidades de cooperação entre os dois paradigmas.

Boufleur, R. (2013). Otimização de performance de sistemas web através de técnicas de clusterização - monografia de curso - ctism - ufsm.

Bourke, T. (2001). *Server Load Balancing*. OReilly e Associates, inc.

CETIC (2014). Cetic.br divulga dados sobre provedores de internet e uso das tic por empresas brasileiras.

Cipriani, O. N. (2009). Replicação de bases de dados postgresql utilizando pgcluster.

Cordeiro, D. (2013). Introdução a computação em nuvem - conceitos teóricos e práticos, evolução e novas possibilidades.

de Moura Nóbrega, P. B. (2013). Proposta de um ambiente de alta disponibilidade para sistemas java web usando computação em nuvem.

Gartner (2013). Gartner says cloud office systems total 8 percent of the overall office market and will rise to 33 percent by 2017.

Informédica (1994). Revista de informática para médicos. *Informédica*, 8(10):12.

Melo, C. A., Arcoverde, D. F., Éfrem R. A. Moraes, ao H. C. Pimentel, J., and Freitas, R. Q. (2011). Software como serviço: Um modelo de negácio emergente. In *Centro de Informática*, volume 1, page 5. UFPE, 1 edition.

NIST (2011). The nist definition of cloud computing - recommendations of the national institute of standards and technology.

Roger, P. (2012). Implementando um sistema balanceador de carga http com alta disponibilidade utilizando nginx como proxy reverso, sincronização rsync, compartilhamento de sessões php e ssl.

Taurion, C. (2009). *Cloud Computing: Computação em nuvem*. Brasport.

World, C. (2010). 11 categorias de cloud computing. http://computerworld.com.br/tecnologia/2010/03/03/11-categorias-de-cloud-computing.