FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS
FATEC PROFESSOR JESSEN VIDAL

THIAGO LUIS SILVA FORTUNATO

IMPLANTAÇÃO DA PLATAFORMA DE MAPEAMENTO DE COMPETÊNCIAS

São José dos Campos
2017

THIAGO LUIS SILVA FORTUNATO

IMPLANTAÇÃO DA PLATAFORMA DE MAPEAMENTO DE COMPETÊNCIAS

Trabalho de Graduação apresentado à Faculdade de Tecnologia São José dos Campos, como parte dos requisitos necessários para a obtenção da graduação de Tecnólogo em Banco de Dados.

Orientador: Me. Eduardo Sakaue

São José dos Campos

2017

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

FORTUNATO, Thiago Luis Silva Fortunato
Implantação do Plataforma para Mapeamento de Competências.
São José dos Campos, 2017.
41fs.

Trabalho de Graduação – Curso de Tecnologia em Banco de Dados,
FATEC de São José dos Campos: Professor Jessen Vidal, 2017.
Orientador: Me. Eduardo Sakaue.

1. Banco de dados, Sistemas de informação. I. Faculdade de Tecnologia. FATEC de São José dos Campos: Professor Jessen Vidal. Divisão de Informação e Documentação. II. Plataforma para Mapeamento de Competências

REFERÊNCIA BIBLIOGRÁFICA –

FORTUNATO, Thiago Luis Silva. **Implantação da Plataforma para Mapeamento de Competências.**
2017. 41f. Trabalho de Graduação - FATEC de São José dos Campos: Professor Jessen Vidal.

CESSÃO DE DIREITOS –

NOME DO AUTOR: Thiago Luis Silva Fortunato
TÍTULO DO TRABALHO: Implantação da Plataforma para Mapeamento de Competências.
TIPO DO TRABALHO/ANO: Trabalho de Graduação / 2017.

_____
Thiago Luis Silva Fortunato

Thiago Luis Silva Fortunato

IMPLANTAÇÃO DA PLATAFORMA DE MAPEAMENTO DE COMPETÊNCIAS

Trabalho de Graduação apresentado à Faculdade de Tecnologia São José dos Campos, como parte dos requisitos necessários para a obtenção da graduação de Tecnólogo em Banco de Dados.

Composição da Banca

Fabiano Sabha Walczak, Mestre, FATEC Professor Jessen Vidal

Jean Carlos Lourenço Costa, Pós-Graduado, FATEC Professor Jessen Vidal

Eduardo Sakaue, Mestre, FATEC Professor Jessen Vidal

_____/_____/_____
DATA DA APROVAÇÃO

DEDICATÓRIA

Dedico este trabalho à Deus e minha família, em especial a meus pais Paulo Fortunato e Rosângela Fortunato pelo apoio e compreensão demonstrados nesses anos de faculdade.

AGRADECIMENTOS

Agradeço em primeiro lugar ao Autor da Existência, Aquele que permite que todas as coisas se concretizem, nosso Deus.

Este trabalho de conclusão de curso não seria possível sem a colaboração de várias pessoas. Em especial, gostaria de agradecer:

Ao orientador Prof. Me. Eduardo Sakaue, por todo apoio e incentivo durante a elaboração deste trabalho e a todo período da faculdade.

A todos professores da FATEC, que são responsáveis pela minha formação técnica.

A todos meus colegas e amigos que formei durante os anos de estudo, principalmente Eduardo Di Nizo, Marcelo Inácio, Daniel Barbosa e William Penna que fizeram maior parte desta jornada.

RESUMO

O projeto Permanência e Desenvolvimento de Talentos Profissionais foi idealizado e mantido pelo Centro Paula Souza, onde visa reduzir a evasão dos alunos nas ETECs e FATECs. O projeto é formado por etapas, que vão desde a matrícula até a contratação dos alunos pelo mercado. Nessas etapas informatizar o processo de mapeamento de perfil do aluno traz grandes benefícios. É de interesse da gestão do Centro Paula Souza que todos alunos ingressantes tenham acesso a plataforma, desta forma, é fundamental que a plataforma esteja disponibilizada de forma centralizada, onde todas instituições (FATECs e ETECs) possam acessa-las. Implantar uma arquitetura que garante a disponibilidade com fácil manutenção é fundamental para O Centro Paula Souza possui uma conta na Plataforma Azure da Microsoft, porém existe uma dificuldade no gerenciamento e suporte das aplicações já disponíveis, por falta de pessoal. Ao centralizar a implantação, garante-se o alcance a todos alunos além de facilitar o gerenciamento dos serviços necessários para mantê-la sempre disponível.

Palavras Chave: *Arquitetura, Disponibilidade, Integração Contínua, DevOps.*

ABSTRACT

The Permanence and Development of Professional Talent Project ~~The Permanence and Professional Talent Development project for~~ the Paula Souza International Market ~~and Market~~, where it aims to reduce student evasion in ETECs and FATECs. The project is made up of stages, ranging from enrollment to recruitment of students through the market. In these steps computerizing the process of mapping the profile of the student brings great gains. It is in the interest of the management of the Centro Paula Souza ~~Paula Souza Center~~ that all incoming students have access to the platform, so it is fundamental that the platform is centrally available, where all institutions (FATECs and ETECs) access them. Deploy an architecture that guarantees availability with ease of maintenance and fundamental for the Center Paula Souza has an account in the Azure Platform of Microsoft, there is a difficulty without management and support of the applications already available, due to lack of personnel. By centralizing the implementation, it guarantees the reach of all the students besides facilitating the management of the necessary services to keep it always available

Keyword: *Architecture, Availability, Continuous Integration, DevOps.*

Formatado: Inglês (Estados Unidos)

Formatado: Recuo: Primeira linha: 1,25 cm

Formatado: Fonte: (Padrão) Times New Roman, 12 pt, Cor da fonte: Cor Personalizada(RGB(0;0;10)), Inglês (Estados Unidos), Padrão: Transparente

Comentado [ES1]: ?????

Comentado [TLSF2R1]: Resolvido

Comentado [ES3]: Em português

Comentado [TLSF4R3]: RESOLVIDO

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Com o intuito aumentar a permanência dos alunos nas instituições de ensino para mitigar a evasão dos estudantes, de forma a garantir a conclusão no prazo previsto, o Projeto Permanência e Desenvolvimento de Talentos Profissionais do Centro Paula Souza deseja reduzir em 50% o índice de evasão nos cursos das FATECs e ETECs selecionadas, visando desenvolver metodologia, ferramentas, processos, parâmetros, indicadores e recursos (PROJETO PERMANÊNCIA, 2017).

Como parte do Projeto de Desenvolvimento o Escritório de Carreiras da Fatec de São José dos Campos foi moldado para ser um mecanismo direcionado a ajudar na preparação dos alunos para o mercado de trabalho.

Uma das etapas do projeto Escritório de Carreiras é o mapeamento de competências. Nesta etapa foi desenvolvida uma plataforma que tem o objetivo de hospedar jogos do tipo perguntas e respostas, que possibilite gerar relatórios dos alunos que finalizaram o jogo a partir das respostas fornecidas (Silva, 2017).

Para suportar esta plataforma é necessário prover toda uma infraestrutura adequada ao formato que a plataforma foi desenvolvida. O Centro Paula Souza detém uma conta na Plataforma *Cloud* Azure da Microsoft, que tem a finalidade de centralizar o gerenciamento e controle de todos serviços necessários para que a plataforma esteja disponível, mas não existe uma quantidade de pessoas disponíveis para dar suporte à todas aplicações já implantadas no Azure. Neste contexto, é necessário prover uma arquitetura em que todos serviços sejam gerenciados de forma contínua, prática, de fácil manutenção e centralizada.

1.1 Objetivo

Prover uma arquitetura para dar suporte a plataforma, fornecendo os recursos necessários ao acesso em larga escala da aplicação, garantindo a agilidade, qualidade e estabilidade com escalabilidade, além de integrar de forma contínua.

Disponibilizar uma arquitetura para alta demanda de requisições em ambiente com pouco recurso computacional e pessoal.

1.2 Cronograma

Para este projeto foi definido um planejamento dividido em duas fases. A primeira fase elaborada no período do segundo semestre de 2016. Foi buscado ter todo conhecimento dos requisitos a serem atendidos e iniciado desenvolvimento.

Tabela 1 - Cronograma Fase 1

Etapa / 2016	Setembro	Outubro	Novembro	Dezembro
Levantamento de requisitos				
Desenvolvimento				

A segunda fase contemplada no primeiro semestre de 2017. Foi continuado o desenvolvimento da aplicação com previsão de uma primeira versão para a segunda semana de junho. Com o objetivo de testar a primeira versão da aplicação, entre os meses de junho e agosto de 2017, teve o lançamento da publicação parcial e em simultâneo a realização de testes com alunos e realização de correções de problemas encontrados na utilização. Por fim previsão da publicação final no mês de julho, mas devido a problemas de hospedagem só pode ser possível a publicação no mês seguinte de agosto de 2017.

Tabela 2 - Cronograma Fase 2

Etapa / 2017	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto
Desenvolvimento								
Testes com alunos								
Publicação parcial								
Correções								
Publicação da versão final								

2. LEVANTAMENTO DE REQUISITOS

Neste Capítulo serão apresentadas as metodologias de levantamento de requisitos não funcionais e casos de uso.

2.1 DevOps

Comentado [ES5]: início do cap

Comentado [TLSF6R5]: RESOLVIDO

O termo DevOps surgiu num evento organizado por Andrew [1]Shaffer e o engenheiro de sistemas John Allspaw, para discutir especificamente os desafios da integração das áreas de desenvolvimento e operações existentes nas empresas.

Modelo utilizado quando se trata de metodologia ágeis, afim de realizar entregas rápidas com qualidade. Tem finalidade de integrar os setores de desenvolvimento e operações, diminuindo a dificuldade que encontravam quando se lançava uma nova funcionalidade do software, pois os setores operacionais criam um ambiente propício para execução de determinadas ferramentas pré-definidas no escopo do projeto e caso algo seja alterado, pode-se perder pontos no quesito qualidade e disponibilidade da aplicação. (DevOps, 2017).

2.2 Metodologia de levantamento de requisitos

[1] Andrew Shaffer e John Allspaw – Idealizadores da 1 º Evento que tratou do assunto DevOps.

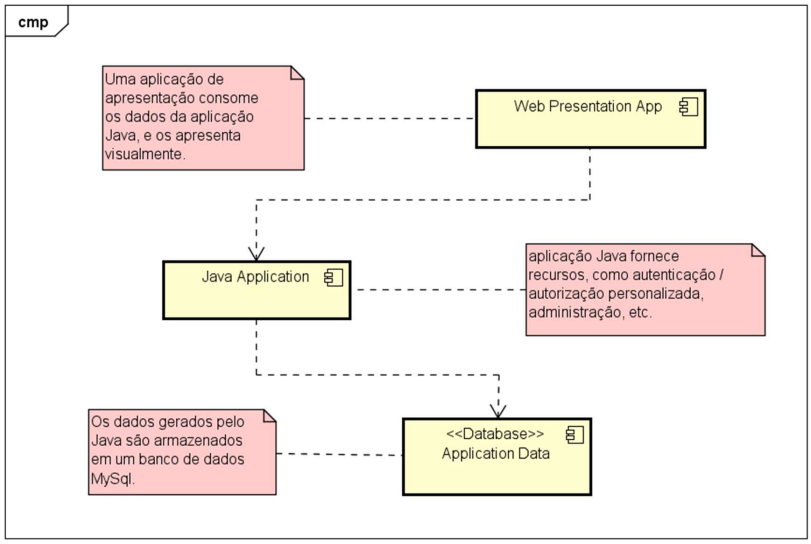
Para este projeto foi adotado a clássica metodologia de entrevista, pois se trata de uma metodologia simples e que produz bons resultados na fase inicial do projeto (PRESSMAN, 2016).

A partir desta técnica foi possível refinar os requisitos e reduzir o tempo de desenvolvimento da aplicação, trabalhando em cima das necessidades que à aplicação deve atender.

A organização da aplicação é distribuída conforma a Figura 1. Com base na nesta Figura, pode-se obter uma visão geral de como e com quem os serviços se relacionam e além de exibir demonstrar suas dependências (Silva, 2017).

Formatado: Justificado

Figura 1- Arquitetura da Aplicação Mapskills.



Formatado: Recuo: Primeira linha: 0 cm

Fonte: Silva, 2017.

2.3 Requisitos do Projeto

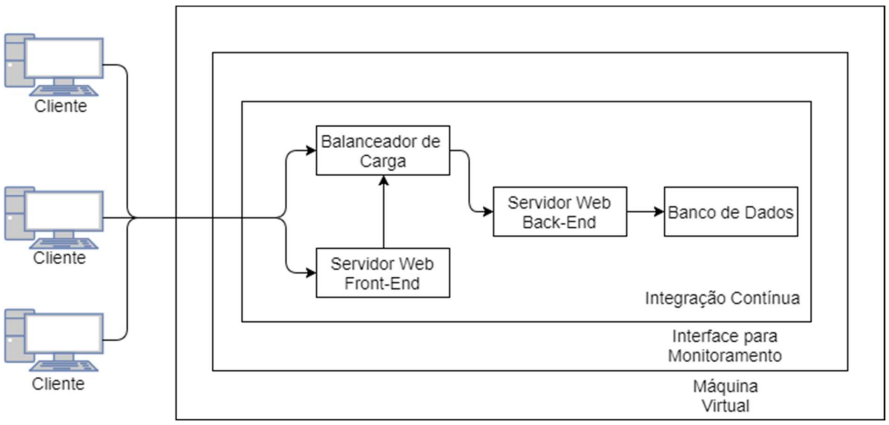
Para garantir a disponibilidade e qualidade da plataforma é necessário que contenha serviços específicos para que a plataforma consiga realizar seu objetivo proposto.

Com base na estruturação da aplicação Mapskills ~~demonstrada~~ conforme ~~na~~ Figura 1 (Silva, 2017), a arquitetura definida como requisito é evidenciada na Figura 2. A arquitetura proposta demonstra todos serviços que são necessários para atender as necessidades da aplicação.

Comentado [ES7]: Já cita o trabalho do Inácio e fala como ele esta estruturado

Comentado [TLSF8R7]: RESOLVIDO

Figura 2 - Arquitetura Mapskills com base nos Requisitos



2.3.1 PaaS – Plataforma como Serviço

Microsoft Azure é uma plataforma de execução de serviços e aplicações baseada em um conceito de *Clouding Computing*. Este termo está associado a utilização da rede mundial de computadores, a internet, para alocação de um ambiente computacional.

Um dos conceitos vinculados à *Cloud* é a Plataforma como um Serviço (PaaS), que é definida como disponibilização de plataformas de desenvolvimento que facilitam a implantação de aplicações assim como o gerenciamento do hardware e das camadas de

software. Fornece todas facilidades necessárias para suportar todo o ciclo de vida das aplicações Web, sem a necessidade de download ou instalação de aplicativos para os desenvolvedores ou usuários finais. (Nogueira, 2010).

Baseado no conceito do que é PaaS, é necessário ter um computador disponível na Plataforma Cloud Azure com Sistema Operacional Ubuntu 14.10 instalado, fornecendo todos recursos necessários para que a plataforma *Mapskills* esteja em produção, sendo escalável, com acesso remoto.

2.3.2 Servidor Web Front-End

Servidor Web com a finalidade de disponibilizar a aplicação que realizará será a interface de comunicação com o usuário. Conforme demonstrado na arquitetura da Aplicação, Figura 1 a aplicação de apresentação consome os dados da aplicação Java e os apresenta visualmente (Silva, 2017). A aplicação contida neste servidor trabalha no modelo Cliente-Servidor, onde o HTTP é o protocolo de comunicação entre as aplicações de front-end (interface gráfica) e back-end (regra de negócio) que seja responsável por disponibilizar a aplicação front-end compilada no formato Recurso de Aplicação Web (war) com Java 8.

2.3.3 Servidor Web Back-End

Servidor responsável por disponibilizar a aplicação *back-end* da plataforma evidenciado na Figura 1, fornecendo suporte aos serviços de autenticação/apresentação, administração e etc (Silva, 2017). A aplicação contida neste servidor trabalha no modelo Cliente-Servidor, realizando conexão com o *front-end* por meio do protocolo HTTP, esta aplicação também foi desenvolvida com Java 8 e compilada no formato Recurso de Aplicação Web (war).

2.3.4 Servidor Banco de Dados

É necessário armazenar todas informações pertinentes ao jogo, como características do jogo a ser aplicado, imagens, perguntas e alternativas, bem como informações das

instituições, usuários, afim de gerar relatórios para que os responsáveis possam visualizar os resultados do jogo, conforme demonstrado na Figura 1.

Comentado [ES9]: Front, Back e BD precisa de justificativa... novamente o trabalho do Inácio

Comentado [TLSF10R9]: RESOLVIDO

2.3.5 Balanceador de Carga

É primordial que a plataforma seja escalável, garantindo que o software estará sempre disponível sendo disponibilizado aos usuários com qualidade. Este serviço será responsável por direcionar as requisições para o Servido *back-end* com menos carga.

Este software será utilizado por 200 ETECs, 70 FATECs, cada uma oferecendo seus cursos, sendo estes, com em média 40 alunos matriculados.

Pode-se ter como exemplo a FATEC de São José dos Campos, que contém 7 cursos, cada um com 40 alunos ingressantes. Na primeira quinzena do semestre haverá um acesso de 360 alunos simultâneos, onde os mesmos devem acessar a qualquer momento e de qualquer lugar.

Comentado [ES11]: acima

Comentado [TLSF12R11]: RESOLVIDO

Comentado [ES13]: Explicar a quantidade de acessos previstos

2.3.6 Integração Contínua

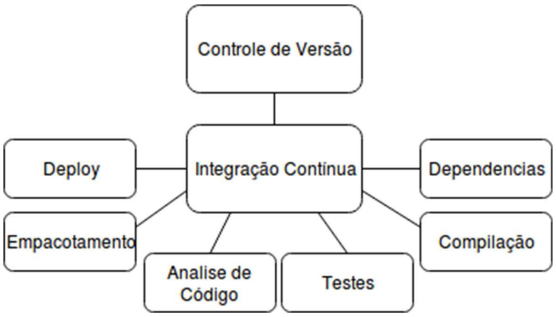
É necessário garantir que um novo código esteja apto a ser disponibilizado frequentemente. Controlando as construções, versionamento, validações e testes. Para que o processo de frequentes alterações parciais enteja disponível de maneira automática e com garantia de que a plataforma funcione como esperado. A Figura 3 ~~demonstra~~ exibe os processos realizados na Integração Contínua.

Comentado [ES14]: Explicar como a IC vai facilitar o gerenciamento e disponibilização em relação à administração

Comentado [TLSF15R14]: RESOLVIDO

E processo assiste o gerenciamento e disponibilização em relação à administração, pois os processos envolvidos ocorrem de maneira automática e não demanda a designação de recurso pessoal.~~Este processo assiste o gerenciamento e disponibilização em relação à administração, pois os processos envolvidos ocorrem de maneira automática e não demanda a designação de recurso pessoal.~~

Figura 3 - Processos realizados na Integração Contínua



2.3.7 Interface de Monitoramento

Serviço de monitoramento em tempo real para que informe ao Técnico responsável pela disponibilização da aplicação o estado da Máquina Virtual, demonstrando graficamente quanto é consumido de recurso de memória, processador, entrada e saída dados, bem como métricas referente a consumo de internet do host, além do estado dos containers.

2.4 Tecnologias Utilizadas

Neste capítulo está descrito cada tecnologia utilizada na arquitetura e o porquê ela foi escolhida.

Tabela 3 - Tecnologias Utilizadas

Nome	Versão	Funcionalidade
Java	8	Desenvolvimento *back-end*, regras de negócio e domínio.
Tomcat	8.5	Servidor Web para hospedagem da aplicação *back-end* e *front-end.*
MySQL	5.7	Persistência dos dados da aplicação.
Jenkins	2.60.2	Plataforma para Gerenciamento da Integração Contínua.
Cadivisor	0.25.0	Aplicação para Monitoramento.
Docker	17.06-0 ce	Plataforma para Gerenciamento dos Containers.
Docker Compose	1.14.0	Plataforma para Orquestração dos Containers.
GitHub		Repositório para código Java.

2.4.1 Docker

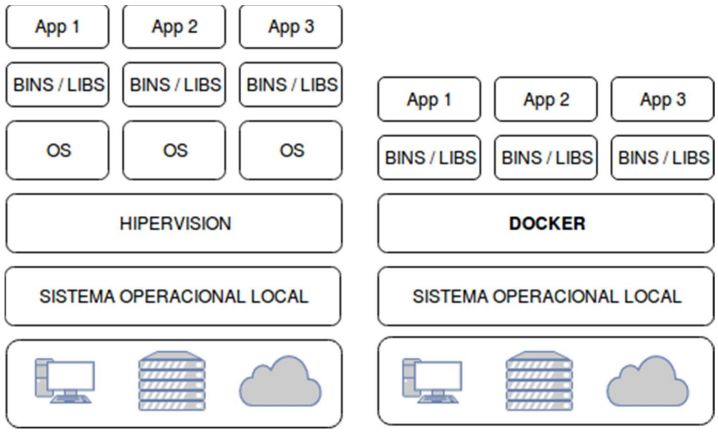
Plataforma instalada na PaaS Azure responsável por administrar e controlar todos serviços necessários para que a aplicação esteja em operação.

O tutorial seguido para instalação do Docker pode ser encontrado na url: *https://docs.docker.com/engine/installation/*.

NAa Figura 4 temos ~~demonstra~~ a comparação entre a arquitetura de virtualização e a arquitetura Docker. Como pode-se observar, o Docker realiza a virtualização de cada container a nível de sistema operacional, excluindo a necessidade do Hipervision[2]. Este método de virtualização faz com que o Kernel do sistema operacional permita que múltiplos processos sejam executados isoladamente no mesmo host. (Docker, 2107).

Comentado [ES16]: Quantas vezes tem que corrigir isso?

Figura 4 - Comparação entre a Arquitetura de Virtualização e o Docker



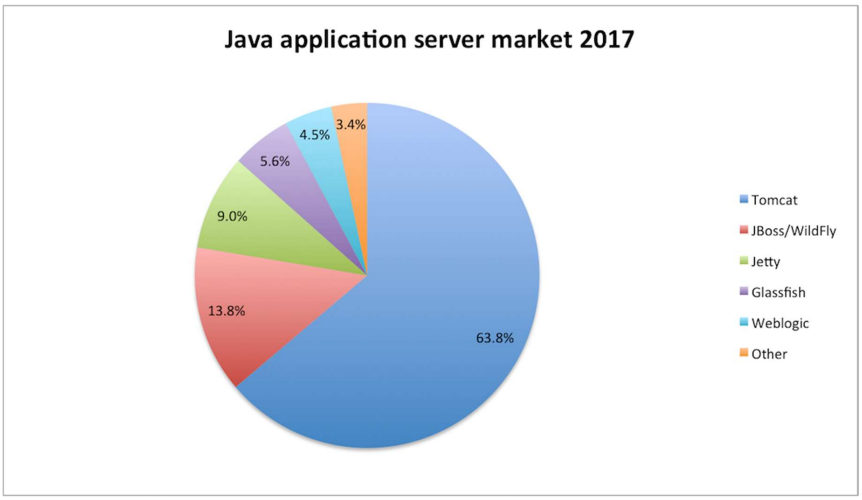
Fonte: Mundo Docker (2017)

[2] Hipervision – Software que controla o acesso dos SOs convidados aos dispositivos de hardware da máquina hospedeira.

2.4.2 Tomcat

Servidor de Aplicação Web desenvolvido e distribuído pela Apache como software livre, permite a execução de inúmeros aplicativos Java para web de grande escala, sendo utilizado por uma ampla gama de industrias e organizações (Tomcat, 2017). Sua escolha, se deu por ter maior integração com aplicações Java, por conter maior fórum e documentação disponível na internet, além de ter sido o maior servidor de aplicações Java Web utilizado em 2017, comparados aos outros servidores disponíveis, como pode-se observar na Figura 5.

Figura 5 - Mercado de Servidores de Aplicações Java 2017



Fonte: DZone / Java Zone (2017).

2.4.3 Haproxy

O HAProxy é um serviço Linux que garante um balanceamento e alta disponibilidade num conjunto de servidores, como também serviço de proxy, ao não expor diretamente estes mesmos servidores na rede externa (Haproxy, 2017).

2.4.4 Jenkins

Ferramenta de Integração Contínua, automatizada. A atividade de construir um projeto é composta por várias etapas, incluindo a compilação do código fonte, execução de testes, empacotamento, além de métricas referente a qualidade do código. O Jenkins trabalha para monitorar todas alterações realizadas no código e integrar com as atividades de construção, afim de dirimir as falhas no desenvolvimento (Jenkins, 2017).

Para que todas etapas da Integração Contínua sejam concluídas é necessário a criação e configuração de *Jobs* onde cada um, tem a finalidade de executar uma tarefa específica.

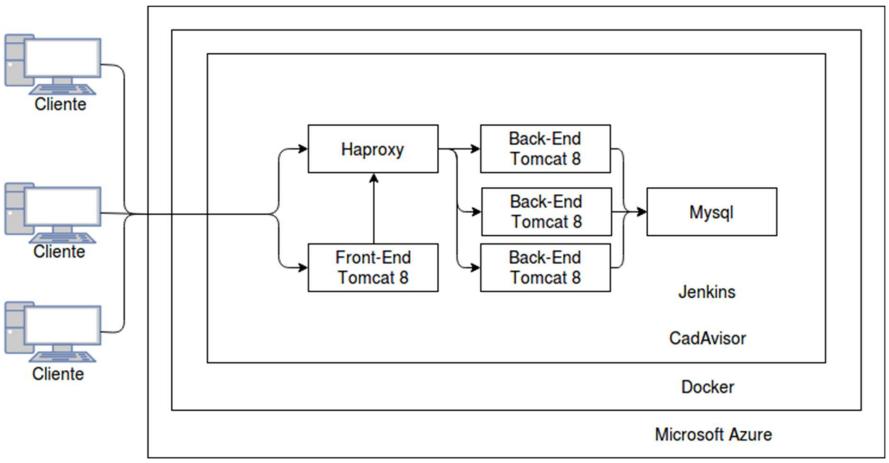
3. DESENVOLVIMENTO

Como pode ser observado na base na arquitetura descrita no item 2.3 ”Requisitos do Projeto”~~a Figura 2~~, são necessários diversos serviços para que a plataforma esteja em produção. Como solução para este problema foi utilizado o Docker para orquestrar e prover que todos serviços estejam sempre em funcionamento.

Todas imagens Docker utilizadas podem ser encontradas no repositório oficial Docker Hub conforme descrito no Anexo A, seção 3.5.2.~~Todas imagens Docker utilizadas podem ser encontradas no repositório oficial do Docker Hub: *https://hub.docker.com/u/thiagolsfortunato/[nome_da_imagem]:[versão]*~~.

A arquitetura foi separada em dois serviços independentes, onde cada um exerce uma funcionalidade e tem uma relação com outros serviços como pode ser observado na Figura 6~~5~~.

Figura 6 - Solução proposta com base na Arquitetura Docker



Comentado [ES17]: colocar a seção

Comentado [ES18]: fazer referência com anexo

Comentado [ES19R18]: ???? cade a ref?

Comentado [TLSF20R18]: A referência do repositório foi colocada no arquivo de configuração Jenkins.

3.1 Máquina Virtual Azure

Conforme requisito descrito na seção 2.2.1, a Máquina Virtual disponibilizada pelo Centro Paula Souza com acesso por meio de usuário e senha através do protocolo Secure Shell[3] (SSH).

O servidor contém instalado o Sistema Operacional Ubuntu 14.10 e seus recursos são apresentados conforme a Figura 7. A Máquina Virtual Azure será responsável por hospedar todos serviços contidos na arquitetura, como Docker e seus respectivos containers, cada qual com uma finalidade específica, que serão descritas nos itens abaixo.

Figura 7 - Descrição dos Recursos do Servidor

Tamanho	vCPU	Memória: GiB	Armazenamento temporário (SSD) GiB	Discos de dados máximos	Taxa de transferência máxima do disco em cache e armazenamento temporário: IOPS / MBps (tamanho do cache em GiB)	Taxa de transferência máxima do disco não armazenado em cache: IOPS / MBps	Máximo de NICs/Largura de banda de rede esperado (Mbps)
Standard_DS1_v2	1	3,5	7	4	4.000 / 32 (43)	3.200 / 48	2 / 750

3.2 Mapskills-Cadvisor

Container com a finalidade de solucionar o requisito descrito na seção 2.2.7, auxilia no gerenciamento dos recursos consumidos pelo host e containers, sabendo em tempo real o quanto é consumido. Por meio da interface gráfica disponibilizada pelo Cadvisor é possível monitorar todos os serviços que compõe a arquitetura.

[3] SSH – Protocolo de propósito geral que fornece conexão criptografada entre um cliente ~~utilizado para acesso remoto~~ e um ~~ao~~ servidor. (BEHROUZ; FIROUZ, 2013)~~, onde a transmissão de dados é criptrogafa~~.

Comentado [ES21]: seção

Comentado [TLSF22R21]: RESOLVIDO

Formatado: Espaço Depois de: 0 pt

Comentado [ES23]: Verificar a numeração

Comentado [TLSF24R23]: RESOLVIDO

Comentado [ES25]: Não colocou a explicação que pedipor whatz

Comentado [TLSF26R25]: A explicação está descrita no Roda pé da pagina

Comentado [TLSF27]:

Comentado [ES28]: Esta tentando explicar a finalidade 2x

Comentado [TLSF29R28]: RESOLVIDO

Formatado: Cor da fonte: Automática

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

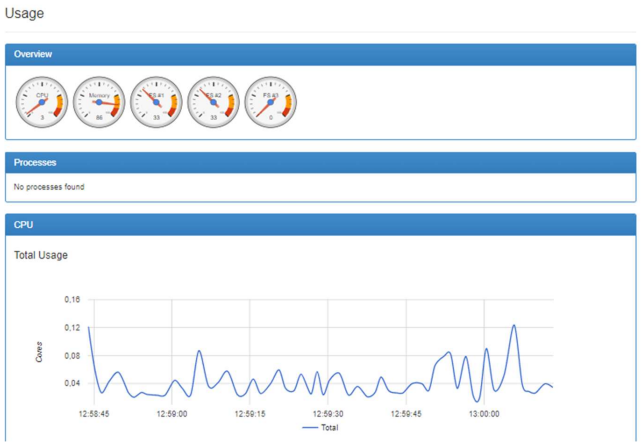
Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Este serviço auxiliará a equipe responsável por dar suporte à plataforma, de forma que não terão que acessar o servidor para se os serviços estão disponíveis ou se os recursos foram totalmente consumidos.

A Figurara~~Figura~~ ~~8~~8 exibe parte da interface gráfica oferecida pelo Cadvisor, a porta configurada para acesso a interface foi a 8888.

Figura 8 - Dashboard Mapskills-Cadvisor



Fonte: Elaborado pelo Autor.

Comentado [ES30]: Figura não mencionada

Comentado [TLSF31R30]: RESOLVIDO

3.3 Mapskills-Front

Container responsável por conter o projeto de Interface Front-Web, mapskills-web.war, fornecendo a interface de comunicação entre usuário e a plataforma. Por meio deste container é que os usuários poderão se conectar a aplicação, responder questões e consultar gráficos, cadastrar novos temas, instituições e cursos, além de prover todos recursos Web necessários para disponibilizá-la. Neste container também está instalado o Java na versão 1.8 e o Tomcat na versão 8.5, para criação deste container foi utilizada uma Imagem *Alpine*, pois reduziu significativamente o tamanho da imagem, rodando apenas um processo Java, atendendo desta forma os itens descritos no 2.2.2.

Para se encaminhar uma requisição ao front-end é necessário acessar aplicação através da url: *http://ip_do_host:80/mapskills-web*

Comentado [ES32]: No cap 2 vc já disse o "por quê?" ... OK

Thiago... vc está explicando apenas o "como" sem falar no "o que", por isso sai sucinto demais. Qual será o comportamento da aplicação nesta etapa de acordo com sua arquitetura??
Lembre-se que vc deu uma geral sobre a arq no início do cap, mas precisa detalhar mais agora

Verifique todo o cap 3 assim

3.4 Mapskills-Back

Container que atende os requisitos descritos no item 2.2.3, responsável por conter o projeto Java de *back-end*, *mapskills.war*, hospedando a aplicação responsável por conter toda regra de negócio, bem como a interface que acessa e valida os dados trafegados onde posteriormente são salvos no banco de dados~~.~~. Neste container está instalado o Java na versão 1.8 e o Tomcat na versão 8.5, para criação deste container foi utilizada uma Imagem *Alpine*, pois reduziu significativamente o tamanho da imagem, rodando apenas um processo Java.

Para se encaminhar uma requisição ao *back-end* é necessário acessar aplicação através da url: *http://ip_do_host:8080/mapskills.*

3.5 Mapskills-Mysql

Container responsável por armazenar informações referente a plataforma como: instituições, cursos, alunos e questões. Com a persistência dos dados é possível gerar relatórios para gerenciamento da aplicação, prover segurança por meio do controle de acesso, e métricas quanto as instituições, cursos e alunos cadastrados na plataforma.~~ todas informações referente a plataforma.~~

Comentado [ES33]: Tem que fazer isso no cap 3 inteiro

Foi configurado para permitir acesso remoto ao banco de dados, pois a aplicação pode não estar no mesmo servidor que a plataforma. Para isto, foi alterado o valor do parâmetro *bind-address* para *0.0.0.0* no arquivo *my.cnf* permitindo acesso remoto.

Um usuário com permissões para consultar, inserir, alterar ou deletar informações na Base de Dados Mapskills foi criado. Este é o único usuário que tem acesso remoto as informações do Banco de Dados.

Somente o Container Tomcat-Back-End pode se comunicar com a Base de Dados, desta forma, foi isolado o acesso aos dados.

Este container tem a finalidade de atender os requisitos descritos no item 2.2.4.

3.6 Mapskills-Haproxy

O Haproxy tem a finalidade de solucionar os requisitos descritos no item 2.2.5. Gerencia todas requisições HTTP destinadas ao Mapskills-app, realizando o balanceamento de carga entre os containers do *back-end*.

Para isto, o Haproxy utiliza do algoritmo Round Robin, este algoritmo tratar os servers como iguais, independentemente do número de conexões solicitadas, sempre redirecionando a próxima requisição ao server seguinte, desta forma, todos servers terão o mesmo número de conexões.

O Haproxy trabalha recebendo todas requisições através da porta 80 e redirecionando internamente a porta 8080 destinada ao Tomcat que contém o mapskills-war. O balanceamento de carga utilizado é de camada 4 (camada de transporte) da Tabela OSI, encaminhando o tráfego do usuário com base no alcance e na porta do IP, no caso definida como 80.

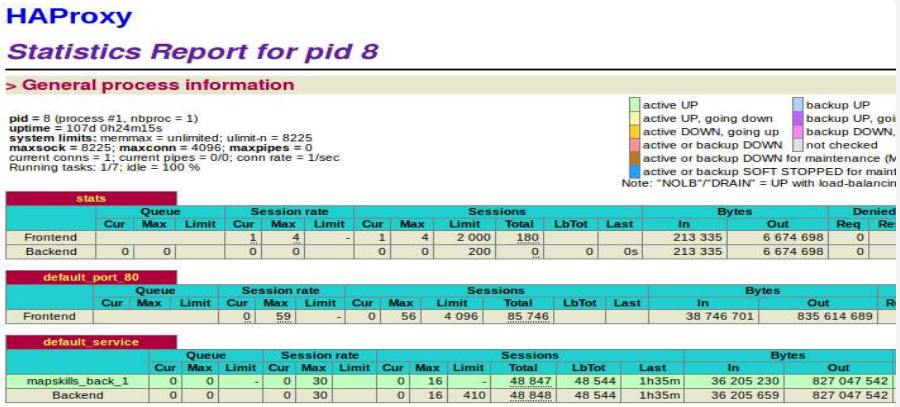
Além de controlar as requisições o Haproxy exibe também fornece um Dashboard para visualização dos servers, números de requisições com sucesso e falha, bem como saber em tempo real a quantidade de Kbps que foi trafegada pela rede, conforme figura abaixo.

A Figura 9 ~~demonstra~~ apresenta o Dashboard acessado por meio da porta 1936.

Comentado [ES34]: Figura não referenciada

Comentado [TLSF35R34]: RESOLVIDO

Figura 9 - Home Page HAProxy



3.7 Docker Compose

Comentado [ES36]: Pq adotou o docker?
Mover parte do 2 pra cá

Comentado [TLSF37R36]: RESOLVIDO

O Docker foi utilizado pois diminui drasticamente o tempo necessário para disponibilizar a aplicação, pois não é necessário configurar o ambiente toda vez que for instalado, sua escolha também foi baseada na redução de custo do projeto, pois o Docker é uma plataforma Open Source e foi disponibilizada apenas uma máquina virtual, hospeda no Azure, e todos recursos que serão necessários são instalados dentro dele, desta forma, não é necessário ter computadores robustos ou vários serviços específicos para execução de tarefas diferentes.

O docker-compose é um arquivo de configuração que tem a finalidade de automatizar a implantação de todo ambiente de produção necessário para que o Mapskills funcione. O arquivo docker-compose.yml criará e iniciará todos serviços definidos.

Para separar o software Mapskills dos aplicativos que realizam o gerenciamento da aplicação, foi necessário criar dois arquivos. Em um arquivo docker-compose será responsável por gerenciar as aplicações Jenkins, Cavisor e o Banco de Dados Mysql conforme Figura 10.

Formatado: Cor da fonte: Automática

Formatado: Cor da fonte: Vermelho

Este arquivo localizado no diretório /opt/mapskills/manager, é inicializado primeiro, pois desta forma se por algum motivo a aplicação deixar de estar disponível, as configurações

realizadas no container Mapskills-Jenkins, bem como os dados persistidos no container Mapskill-Mysql não serão perdidos.

Figura 10 - Arquivo docker-compose.yml de Gerenciamento

```
version: '3'
services:
  jenkins:
    image: thiagolsfortunato/mapskills-jenkins:2.60.2
    container_name: mapskills-jenkins
    restart: on-failure
    volumes:
      - /usr/bin/docker:/usr/bin/docker
      - /var/run/docker.sock:/var/run/docker.sock
      - /opt/mapskills/docker-compose.yml:/mapskills/docker-compose.yml
      - back:/mapskills/back:rw
      - front:/mapskills/front:rw
    environment:
      DOCKER_SOCKET: /var/run/docker.sock
      DOCKER_GROUP: dockerhost
      JENKINS_USER: jenkins
    ports:
      - 8585:8080
  cadvisor:
    image: thiagolsfortunato/mapskills-cadvisor
    container_name: mapskills-cadivisor
    volumes:
      - /:/roofs:ro
      - /var/run:/var/run:rw
      - /sys:/sys:ro
      - /var/lib/docker/:/var/lib/docker:ro
    ports:
      - 8888:8080
  mysql:
    image: thiagolsfortunato/mapskills-mysql:5.6
    container_name: mapskills-mysql
    restart: on-failure
    volumes:
      - mysql:/var/lib/mysql/mapskills
    environment:
      MYSQL_ROOT_PASSWORD: mapskills
      MYSQL_DATABASE: mapskills
      MYSQL_USER: mapskills
      MYSQL_PASSWORD: mapskills
    ports:
      - 3306:3306
volumes:
  mysql:
  back:
  front:
```

Para que a segunda parte seja inicializada, é necessário que a primeira esteja com todos serviços ativos. Neste arquivo docker-compose, serão inicializados os containers Mapkills-app, Mapskills-web e Mapkills-Haproxy conforme Figura 11. Todas configurações de acesso, além de como e com quem cada container se comunica é descrita neste arquivo.

Figura 11 - Arquivo docker-compose.yaml da Aplicação

```
version: '3'
services:
  front:
    image: thiagolsfortunato/mapskills-tomcat8:v1
    restart: on-failure
    volumes:
      - manager_front:/usr/local/tomcat/webapps:rw
    ports:
      - 80:8080
  back:
    image: thiagolsfortunato/mapskills-tomcat8:v1
    restart: on-failure
    external_links:
      - mapskills-mysql
    volumes:
      - manager_back:/usr/local/tomcat/webapps:rw
  lb:
    image: thiagolsfortunato/mapskills-haproxy:v1
    restart: on-failure
    container_name: mapskills-haproxy
    environment:
      - STATS_AUTH=mapskills:mapskills
    links:
      - back
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - 8080:80
      - 1936:1936
volumes:
  manager_back:
    external: true
  manager_front:
    external: true
```

3.7.1 Links

Além de inicializar os serviços, o arquivo docker-compose.yaml é responsável por configurar como e com quem cada um dos containers irão se comunicar, bem como os volumes necessários para cada container.

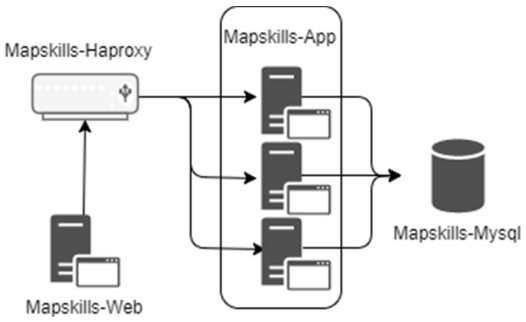
Foi criado um link entre os containers Mapskills-app e o Mapskills-Mysql, pois desta forma o *back-end* da aplicação poderá persistir, consultar deletar ou alterar qualquer informação no banco de dados.

O link entre o container Mapskills-app e Mapskills-web para que os dados inseridos na interface web seja trafegada para o *back-end*, e assim os dados sejam manipulados.

Link entre o Haproxy e Mapskills-app para que todas requisições destinadas ao Mapskills-app sejam controladas pelo balanceador de carga.

Os links criados no arquivo *docker-compose.yaml* formam a comunicação ilustrada ~~demonstrada~~ na Figura 12.

Figura 12 - Comunicação entre os Containers.



3.7.2 Volumes

Os Volumes Docker têm a finalidade de persistir os dados usados pelos Containers Docker. Cada container é responsável pelo seu volume e informação sensível ao serviço em que oferece.

O Mapskills-app e Mapskills-web tem os volumes mapeados para que o arquivo .war seja atualizado pelo Jenkins a qualquer momento, desta forma, é solucionado o problema de integração contínua, pois uma versão nova do sistema, atualizará automaticamente.

O volume Mapskills-Mysql tem a finalidade de armazenar todas informações salva na base de dados, podendo assim para a execução ou mesmo excluir o container Mapskills-Mysql que as informações não serão perdidas.

Os volumes utilizados no container Mapkills-Jenkins, tem funções importantes na solução para o requisito de Integração Contínua, pois eles formam uma comunicação entre os containers Mapkills-App e Mapskills-Web para que sejam atualizados a qualquer instante,

além de compartilhar os arquivos de execução do Docker e Docker-Compose, para que possam ser executados comando de dentro do container.

No container Mapskills-Cadvisor são compartilhados com o host os volumes necessários para o monitoramento de dados referente ao Host e Containers Docker.

3.8 Mapskills-Jenkins

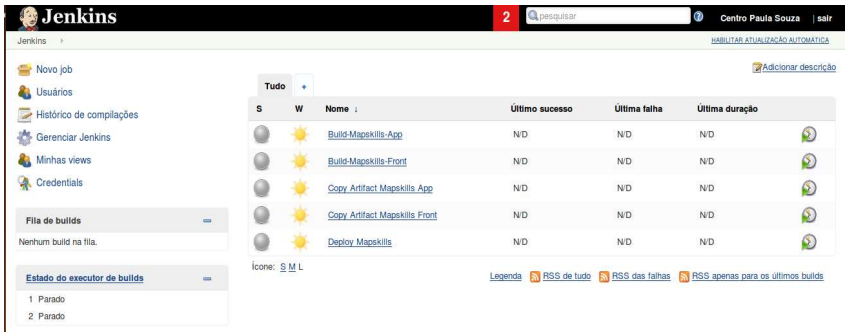
Container Jenkins realiza o controle das implantações realizadas durante a implementação do software conforme requisito descrito no item 2.2.6. Após configurado, tem o trabalho de realizar construções de forma instantânea, com testes sendo executados e falhas detectadas caso encontre-as. Esse processo auxilia na detecção antecipada de possíveis bugs ou que novas funcionalidades atrapalhem a execução de funções já implementadas.

Seu funcionamento baseia-se na criação de *Jobs* para execução de tarefas específicas, para que seja atendido todos requisitos propostos à Integração Contínua, foi necessário configurar quatro ~~Trabalhos~~*Jobs*: Build-Mapksills-App, Build-Mapskills-Front, Copy Artifact Mapskills App, Copy Artifact Mapskills Front, Deploy Mapskills.

Todas configurações realizadas nos *Jobs* estão descritas no Anexo A, item 3.

A Interface Web do Jenkins pode ser acessada através do http://ip_do_host:8585, conforme demonstrada na Figura 13.

Figura 13 - Dashboard Mapskills-Jenkins



Formatado: Fonte: Itálico

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt, Itálico

Formatado: À esquerda, Espaço Depois de: 8 pt, Espaçamento entre linhas: Múltiplos 1,05 lin.

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

3.8.1 Build-Mapksills-App

Este *Job* irá baixar o projeto Java do repositório GitHub e compilar o projeto no formato .war, para que seja disponibilizado posteriormente por outro *Job*.

O comando utilizado apaga a pasta Target, instala os pacotes nos respectivos repositórios e não rodar o script de criação do Banco de Dados, pois desta forma é garantido que os dados nunca serão apagados. Todos esses comandos são rodados pelo usuário Azure, configurado na aplicação:

```
mvn clean install -Dliquibase.should.run=false -Pazure
```

3.8.2 Build Mapskills-Web

Este job irá baixar o projeto de Interface Web do repositório GitHub e compilar o projeto no formato .war, para que seja disponibilizado posteriormente.

```
mvn clean install
```

Comentado [ES38]: anexo

3.8.3 Copy Artifact Mapskills App

Este job depende de que o Build-Mapskills-App (capítulo 3.8.2) tenha sido finalizado com sucesso, só assim, ele irá copiar o arquivo .war do diretório */var/jenkins_home/workspace/Build-Mapskills-Back/target/mapskills.war* para o volume Docker */mapskills/back, pois desta forma o arquivo .war é compartilhado com o container* responsável por conter a aplicação de *back-end*.

```
sudo cp /var/jenkins_home/workspace/Build-Mapskills-
Back/target/mapskills.war /mapskills/back
```

3.8.4 Copy Artifact Mapskills Front

Assim como o *Job* Copy Artifact Mapskills Back Este trabalho depende também de que o job Build-Mapskills-Web tenha sido finalizado com sucesso. Após isso ele copia o arquivo .war do diretório */var/jenkins_home/workspace/Build-Mapskills-Front/target/mapskills-web.war* para o volume Docker */mapskills/front,* pois desta forma o arquivo .war é compartilhado com o container responsável por conter a aplicação de *front-end*.

3.8.5 Deploy Mapskills

Como produto final, o trabalho Deploy Mapskills é responsável por disponibilizar os serviços de *front*, *back-end,* balanceador de carga e o banco de dados em produção.

Através da da execução do arquivo "docker-compose.yaml", localizado no diretório */mapskills,* o comando executado neste Job, cria os containers e volumes além definir como e com quem cada container se comunica, como foi ~~demonstrado~~ exposta na Figura 12. Após a criação dos containers é possível acessar através da web todos os recursos disponibilizados pela arquitetura.

```
sudo docker-compose -f /mapskills/docker-compose.yml up -d
```

4. RESULTADOS

Neste Capítulo serão apresentados os resultados obtidos com a implantação da aplicação. Foram realizados testes em ambiente de desenvolvimento entre os meses de junho e agosto, sendo acessada por mais de 300 alunos na Fatec São José dos Campos, além de testes de estresse por meio de software.

O Servidor Microsoft Cloud Azure contém as configurações descritas no item 3.1 "Máquina Virtual Azure" e Figura 7 e o servidor utilizado para Rede Interna continha as seguintes configurações:

- Notebook Dell Vostro 5470
- Processador: Core Intel i7 1.8Ghz
- Memória Ram: 8Gb,
- Sistema Operacional: Windows 10

Atualmente a aplicação encontra-se em produção e foi acessada por mais de 1400 alunos das instituições da FATEC São José dos Campos, FATEC Tatuapé, FATEC Araçatuba, FATEC Garça, FATEC Mogi Mirim, FATEC Pindamonhangaba e FATEC Jales.

4.1 Experimento 1

Comentado [ES39]: qual era o objetivo?
Resultados esperados x resultados atingidos
Medidas corretivas

Comentado [TLSF40R39]: RESOLVIDO

- Servidor: Rede Interna.
- Data Prevista: 13 de junho de 2017
- Horário: das 18h às 21h
- Público Alvo: Alunos da matéria de Gestão Produção Industrial
- Quantidade de Alunos: 120 alunos
- Local: Fatec São José dos Campos

O primeiro experimento foi utilizado a rede interna da Fatec com a finalidade de observar como a aplicação iria se comportar. Neste primeiro evento, muitos alunos não sabiam seu e-mail ou, e por diversas vezes foi necessário acessar o Banco de Dados, também foi observado o consumo de Memória e Processador do computador usado como servidor.

Durante o experimento pode-se observar um alto consumo de processamento pois eram realizadas diversas tarefas concorrentemente, além de operações de acesso ao banco de dados. Para solução deste consumo foi adicionado o balanceador de carga, dando a possibilidade de executar a aplicação em mais de um servidor concorrentemente.

4.2 Experimento 2

- Servidor: Rede Interna e Cloud Azure.
- Data Prevista: 01 de agosto de 2017
- Horário: das 18h às 21h
- Público Alvo: Alunos ingressantes
- Quantidade de Alunos: 150 alunos
- Local: Fatec São José dos Campos

O segundo experimento foi realizado na Fatec com o acesso de aproximadamente 150 alunos, com o objetivo de monitorar pelos containers Mapskills-Cadvisor e Mapskills-Haproxy o acesso a plataforma além dos recursos utilizados pelo servidor.

Dado momento, foi constatado uma inconsistência na execução da aplicação de *back-end* sendo necessária atualiza-la já em produção. Ao realizar o processo de atualização da versão aplicação de *back-end,* foi interrompido o acesso ao Servidor de Banco de Dados. Por conta da indisponibilidade de acesso ao banco dados a arquitetura do projeto foi fracionada em duas partes. Uma contendo a parte de gerenciamento da aplicação com os serviços: Mapskills-Jenkins, Mapskills-Cadivisor, Mapskills-Mysql, pois estes containers armazenam ou gerenciam dados importantes e críticos na aplicação. E a outra parte com os serviços Mapskills-Haproxy, Mapskills-Front, Mapskills-Back, com a finalidade automatizar os serviços de acesso a plataforma, facilitando também a integração, e controle na entrega de uma nova versão do software.

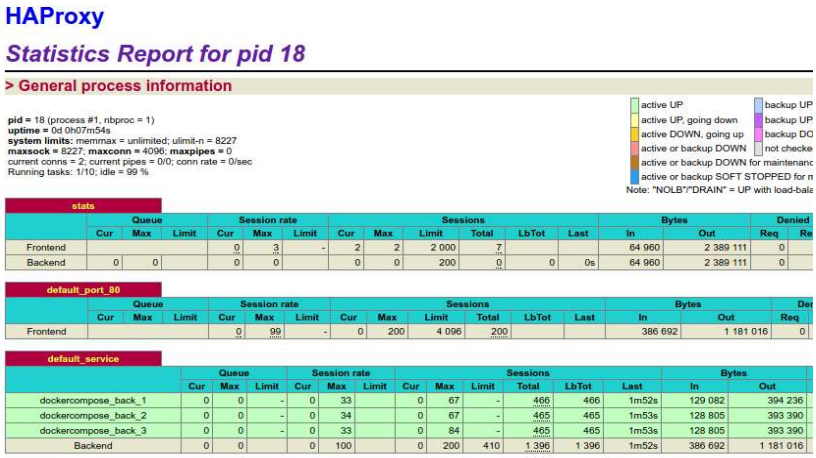
4.3 Experimento 3

- Servidor: Cloud Azure.
- Data Prevista: 15 de agosto de 2017
- Horário: das 13h às 13:30
- Quantidade de Requisição: 1.400
- Local: Fatec São José dos Campos

Este teste teve a finalidade de validar o serviço de Balanceamento de Carga entre os containers de back-end. Por meio do software Jmeter, foi configurado que 3 usuários diferentes realizassem o login na aplicação durante 30 min 400 vezes cada um, cada requisição tinha um intervalo de 4,5 segundo.

Como pode observar na Figura 14, o Haproxy distribuiu de forma igual todas requisições destinadas ao servidor de back-end, olhando de forma igual para todos servidores. Com isto, se um dia a aplicação for hospedada em servidores diferentes, o Haproxy será fundamental para redirecionar as requisições ao servidor com menos carga, melhorando o tempo de resposta, deixando de sobrecarregando o servidor garantindo a disponibilidade da aplicação.

Figura 14 - Teste Balancemento de Carga entre Containers



HAProxy

Statistics Report for pid 18

> General process information

pid = 18 (process #1, nbproc = 1)
uptime = 0d 0h07m54s
system limits: memmax = unlimited; ulimit-n = 8227
maxsock = 8227; maxconn = 4096; maxpipes = 0
current conns = 2; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/10; idle = 99 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenanc
active or backup SOFT STOPPED for n
backup UP
backup UP
backup DO
not checke
Note: "NOLB"/"DRAIN" = UP with load-bala

stats

	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Re
Frontend				0	3	-	2	2	2 000	7			64 960	2 389 111	0	
Backend	0	0		0	0		0	0	200	0	0	0s	64 960	2 389 111	0	

default_port_80

	Queue			Session rate			Sessions						Bytes		Der
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req
Frontend				0	99	-	0	200	4 096	200			386 692	1 181 016	0

default_service

	Queue			Session rate			Sessions						Bytes	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out
dockercompose_back_1	0	0	-	0	33		0	67	-	466	466	1m52s	129 082	394 236
dockercompose_back_2	0	0	-	0	34		0	67	-	465	465	1m53s	128 805	393 390
dockercompose_back_3	0	0	-	0	33		0	84	-	465	465	1m53s	128 805	393 390
Backend	0	0		0	100		0	200	410	1 396	1 396	1m52s	386 692	1 181 016

4.4 Experimento 4

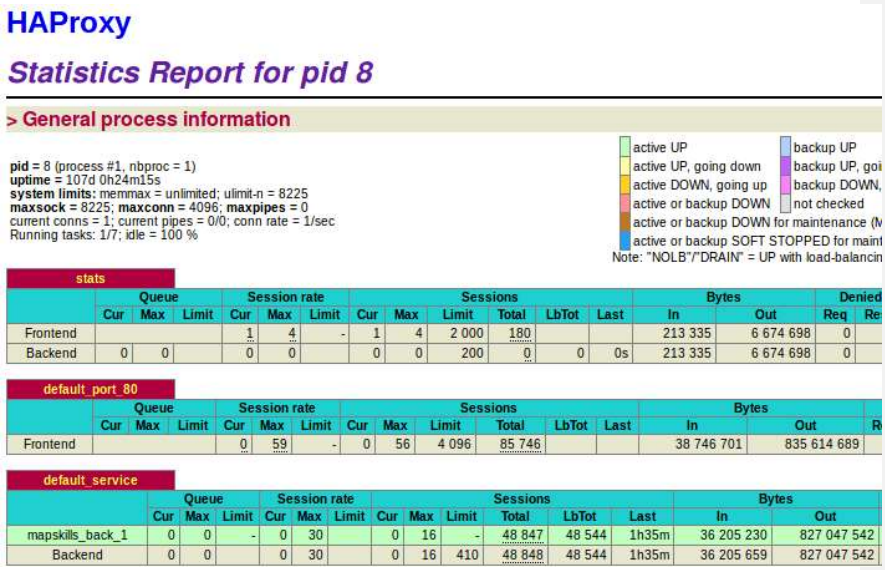
- Servidor: Cloud Azure.
- Data Prevista: Atualmente
- Quantidade de Requisição: 48.544
- Público Alvo: Alunos das Fatecs e Etecs.

A aplicação encontra-se em produção a mais de 5 meses, sendo utilizada por diversas instituições ligadas ao Centro Paula Souza no Estado de São Paulo. O servidor Cloud Azure tem respondido de forma satisfatória, mesmo com pouco recurso, atendendo todas requisições e direcionando ao servidor e respondendo de forma rápida, mesmo com uma alta demanda.

A Figura 15 apresenta as métricas referente aos dados trafegados no servidor durante os mais 5 meses em que está em produção.

Figura 15 - Resultados da Aplicação em Produção

Comentado [ES41]: Não foi chamado



HAProxy

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1)
uptime = 107d 0h24m15s
system limits: memmax = unlimited; ulimit-n = 8225
maxsock = 8225; **maxconn** = 4096; **maxpipes** = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/7; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (M
active or backup SOFT STOPPED for maint
backup UP
backup UP, goi
backup DOWN,
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancin

stats

	Queue			Session rate			Sessions						Bytes		Denied	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Re
Frontend				1	4	-	1	4	2 000	180			213 335	6 674 698	0	
Backend	0	0		0	0		0	0	200	0	0	0s	213 335	6 674 698	0	

default_port_80

	Queue			Session rate			Sessions						Bytes		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	R
Frontend				0	59	-	0	56	4 096	85 746			38 746 701	835 614 689	

default_service

	Queue			Session rate			Sessions						Bytes	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out
mapskills_back_1	0	0	-	0	30		0	16	-	48 847	48 544	1h35m	36 205 230	827 047 542
Backend	0	0		0	30		0	16	410	48 848	48 544	1h35m	36 205 659	827 047 542

5. TRABALHOS FUTUROS

Finalizado a implantação da plataforma e atendendo aos requisitos levantados, foram encontradas melhorias na arquitetura para que torne ainda mais eficiente e robusta.

- Alterar o Servidor de Gerenciador de Repositório para privado, como por exemplo o Git Lab, que permite a realização do processo de Integração Contínua.

- Excluir imagens não utilizadas das cenas quando são editadas, não gerando acumulo de arquivos não utilizados no servidor.

- Deixar flexível o arquivo *application.properties* da aplicação *back-end* para configuração do local onde ficarão as imagens das cenas dos jogos.

- Separar do Servidor de Banco de Dados em outro servidor para garantir a integridade e disponibilidade.

- Implementar uma arquitetura de clusters utilizando do Docker Swarm, onde vários computadores trabalham juntos afim de garantir desempenho e disponibilidade da plataforma.

REFERÊNCIAS

Apache Tomcat**, Tomcat,** Disponível em < https://tomcat.apache.org/>. Acesso em 27 de outubro de 2017.

Aulbach, Stefan; JACOBS, Dean; KEMPER, Alfons; et al. (2009) "**A Comparison of Flexible Schemas for Software as a Service**" 35th SIGMOD - International Conference on Management of Data, 2009.

BEHROUZ, A. Forouzan; FIROUZ Mosharraf. **Redes de Computadores: Uma Abordagem Top-Down**, Tradução Técnica: AHMG Editora Ltda, 2013. 896 p.

Docker Documentation, **Docker.** Disponível em <https://docs.docker.com>. Acesso em: 28 de outubro de 2017.

DevOps Conceitos, **DevOps**. Disponível em <https://aws.amazon.com/pt/devops/what-is-devops>. Acesso em: 28 de outubro de 2017.

Haproxy Description, **Haproxy,** Disponível em < http://www.haproxy.org/#desc>. Acesso em: 29 de outubro de 2017.

Jenkins Documentation, **Jenkin**, Disponível em <https://jenkins.io/doc>. Acesso em: 28 de outubro de 2017.

~~O que é Docker, **Mundo Docker,** Disponível em, < https://www.mundodocker.com.br/o-que-e-docker/ >. Acesso em 30 de outubro de 2017.~~Most Popular Java Application Servers (2017 Edition), **DZone / Java Zone**. Disponível em <https://dzone.com/articles/most-popular-java-application-servers-2017-edition>. Acesso em 27 de outubro de 2017.

Nogueira, Matheus Cadori1; PEZZI, Daniel da Cunha (2010) "**A Computação Agora é nas Nuvens**" Universidade de Cruz Alta (UNICRUZ) – Cruz Alta, RS – Brasil.

O que é Docker, **Mundo Docker**, Disponível em, < https://www.mundodocker.com.br/o-que-e-docker/ >. Acesso em 30 de outubro de 2017.

PROJETO PERMANÊNCIA. **Projeto Permanência e Desenvolvimento de Talentos Profissionais**. Disponível em: <http://www.projpermanencia.com.br>. Acesso em: 24 de junho de 2017.

PRESSMAN, Roger; MAXIM**, Bruce. Engenharia de Software-8ª Edição**. McGraw Hill Brasil, 2016.

SILVA, Marcelo Inácio da. **Plataforma para Jogos de Mapeamento de Competências, 2017.**

Formatado: Português (Brasil)

Apache Tomcat**, Tomcat,** Disponível em <https://tomcat.apache.org/>. Acesso em 27 de outubro de 2017.

Most Popular Java Application Servers (2017 Edition), **DZone / Java Zone**. Disponível em <https://dzone.com/articles/most-popular-java-application-servers-2017-edition>. Acesso em 27 de outubro de 2017.

ANEXOS

[A1] Configuração do Jenkins e Docker.

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt

Formatado: Fonte: (Padrão) Times New Roman, 12 pt