**FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS**

**FATEC PROFESSOR JESSEN VIDAL**

**THIAGO LUIS SILVA FORTUNATO**

**PLANEJAMENTO E IMPLANTAÇÃO DO SOFTWARE**

**AVALIADOR DE COMPETÊNCIAS MAPSKILLS**

**São José dos Campos**

**2016**

**THIAGO LUIS SILVA FORTUNATO**

**PLANEJAMENTO E IMPLANTAÇÃO DO SOFTWARE**

**AVALIADOR DE COMPETÊNCIAS MAPSKILLS.**

Trabalho de Graduação apresentado à

Faculdade de Tecnologia São José dos

Campos, como parte dos requisitos

necessários para a obtenção do título de

Tecnólogo em Banco de Dados.

**Orientador: Me. Eduardo Sakaue**

São José dos Campos

2015

**Sumário**

**- CAPITULO I**

**Problema (Tema):**

Planejamento e Implantação do Software Avaliador de Competências Mapskills.

**- Justificativa (Motivação): (Projeto Permanência e Desenvolvimento de Talentos Profissionais)**

Com o intuíto 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 Carreitas 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 apartir das respostas fornecidas (Inácio, 2017).

Para suportar esta plataforma é necessário prover toda uma infraestrutura adequada ao formato que a plataforma foi desenvolvida. A plataforma foi desenvolvida para que tenha acesso simultâneo de aproximadamente 80 usuários, porém, é de interesse que todas Fatecs e Etecs tenham acesso a plataforma.

**- PROPOSTA E SOLUÇÃO (Metodologia):**

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

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

**- CAPITULO II (Levantamento de Requisitos)**

Neste capítulo serão citadas metodologias utilizadas para o planejamento da arquitetura da plataforma de Jogos de Competência.

**Metodologia**

Foi utilizado a metodologia de entrevista para coleta de requisitos não funcionais, produzindo bons resultados na fase incial do projeto.

**DevOps**

O termo DevOps surgiu num evento organizado por Andrew 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 dispobibilidade da aplicação.

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. A arquitetura definida como requisito esta descrita na Figura X

Plataforma



Os requisitos deste projeto são:

Os requisitos deste projeto são:

**Máquina Virtual**

Software em que um Sistema Operacional é instalado e executado, fornecendo os serviços necessários para que a plataforma estaja em produção.

**Servidor Web front-End**

Servidor Web com a finalidade de disponibilizar a aplicação que realizará será a interface de comunincação com o usuário. 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 grafica) e Back End (regra de negoócio) que seja responsável por disponibilizar a aplicação front-end compilada no formato .war com java 8.

**Servidor Web Back-End**

Servidor responsável por disponibilizar a aplicação Back-End da plataforma. 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 compilada no formato .war.

**Banco de Dados**

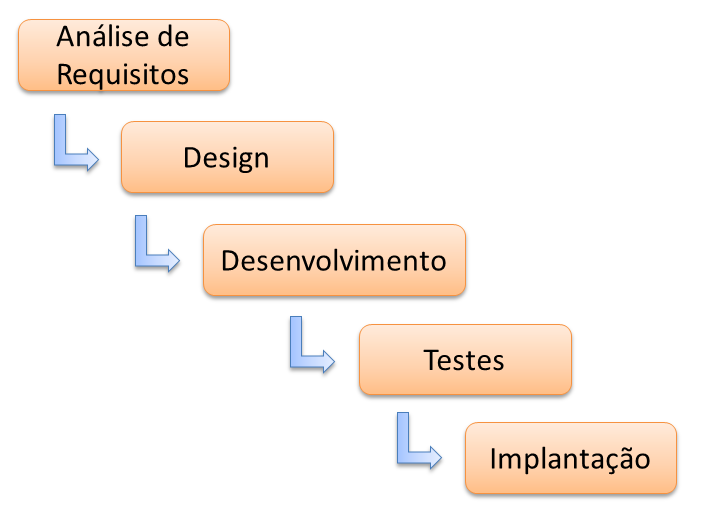
É necessário armazenar todas informações pertinentes ao jogos, como características do jogo a ser aplicado, imagens, perguntas e alterntivas, bem como informações das instituições, usuários, afim de gerar relatórios para que os responsáveis possam visualizar os resultados do jogo.

**Balanceador de Carga**

É fundamental controlar a quantidade de acesso ao Servidor de Back-End, para que isso ocorra é necessario que este serviço seja escalável. Este serviço será responsável por direcionar as requisições para o Servido Back-End com menos carga.

**Integração Contínua**

É necessário garantir que um novo código esteja apto à ser disponibilizado frequentemente. Controlando as contruções, versionamento, validações e testes. Para que o processo de frequentes alterações parciais entejam disponíveis de maneira automática e com garantia de que a plataforma funcione como esperado.



**Interface de Monitoramento**

Serviço de monitoramento em tempo real para que informe ao Técnico resposável 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.

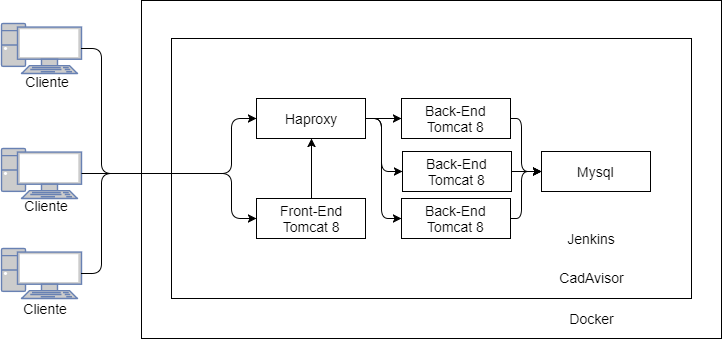
**Quantidade e Escalabilidade**

É primordial que a plataforma seja escalável, garantindo que o software estará sempre disponível sendo disponibilizado aos usuários com qualidade.

Como o software será utilizado por alunos do Centro Paula Souza, é fundamental que este requisito seja preenchido, pois de qualquer lugar e a qualquer momento o software deverá estar disponível.

**- CAPITULO III (Desenvolvimento)**

**Pegar o diagrama e colocar as tecnologias**



**Explicar como utilizou cada uma delas**

**Docker**

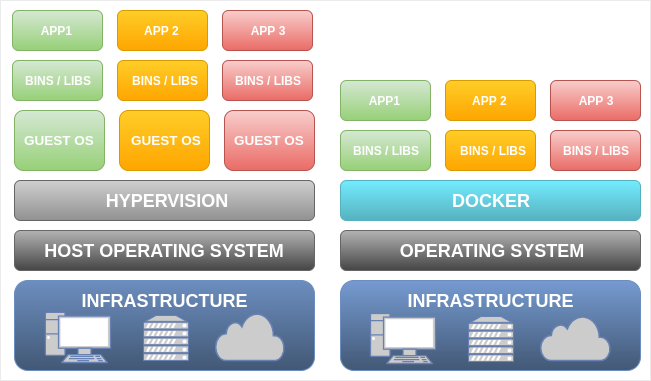
Plataforma instalada na Máquina Virtual responsável por administrar e controlar todos serviços necessários para que a aplicação esteja em operação.

O tutorial seguido para instalação do Docker pode ser encontrado neste [link](https://docs.docker.com/engine/installation/).

Com o Docker realiza o empacotamento do ambiente inteiro dentro de um contêiner, compartilhando com outras aplicações o que irão utilizar. Foi utilizado esta aplicação pois diminuirá drasticamente o tempo de necessário para disponibiizar a aplicação, pois não será necessário configurar o ambiente toda vez que for instalado .

A escolha do Docker também baseada na redução de custo do projeto, pois temos somente uma máquina virtual, hospeda no Azure com o Docker instalado, e todos recursos que serão necessários instalados dentro dele, desta forma, não é necessário ter computadores robustos ou vários serviços específicos para execução de tarefas diferentes.

Cada container criado no Docker é isolado a virtualização a nível do sistema operacional, um método de virtualização onde o kernel do sistema operacional permite que múltiplos processos sejam executados isoladamente no mesmo host.



**Containers**

**Mapskills-Cadivisor**

Container que auxilia no gerenciamento dos containers e recursos consumidos pelo host. Com esta imagem e possivel monitorar todos processos e saber em tempo real todo funcionamento do host alocado.

A Imagem Docker utilizada neste container pode ser encontrada no Repositório oficial Docker Hub thiagolsfortunato[/mapskills-cadvisor](https://hub.docker.com/r/google/cadvisor/).

A aplicação cAdivor pode ser acessada pela url: http://ip\_do\_host:8888.

**Mapskills-Jenkins**

Container Jenkins realiza o controle dos deploys realizados durante a implementação do software. Após configurado, tem o trabalho de realizar construções de forma instantânea, com testes sendo executados e falhas detectadas caso encontre-as.

Para configurar o Jenkins é necessário primeiramente configurar o Java para versão 1.8, a mesma utilizada no desenvolvimento do projeto, além de configurar o Maven 3.5, ferramenta de integração de projeto responsável com armazenar todas dependencias externas à aplicação, além de conter comandos para validação, contrução , execução de testes, instalação e entre outras.

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



A Interface Web do Jenkins pode ser acessada através do http://ip\_do\_host:8585.

**Build-Mapksills-App**

Este trabalho 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

**Build Mapskills-Web**

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

mvn clean install

**Copy Artifact Mapskills App**

Este trabalho depende de que o trabalho Build-Mapskills-App 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

**Copy Artifact Mapskills App**

Assim como o Trabalho Copy Artifact Mapskills Back Este trabalho depende também de que o trabalho 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.

**Deploy Mapskills**

Como produto final, o trabalho Deploy Mapskills é resposável por disponiblizar os projetos de front e back-end em produção, ou seja, esteja disponível a ultima versão estável do projeto. Para que seja esteja em produção, é executado o arquivo “docker-compose.yml”.

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

**Tomcat-Back**

Container responsável por conter o projeto Java de Back-End, mapskills.war. Neste container está instalado o Java na versão 1.8 e o Tomcat na versão 8.5, para criação deste container foi utilizada uma Image Alpine, pois reduziu significavamente 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

**Tomcat-Front**

Container responsável por conter o projeto de Interface Front-Web, mapskills-web.war. 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 Image Alpine, pois reduziu significavamente o tamanho da imagem, rodando apenas um processo Java.

Para se encaminhar uma requisição ao front-end é necessário acessar aplicação através da url: http://ip\_do\_host:80/mapskills-web

**Mapskills-mysql**

Container que está contido o Banco de Dados Relacional Mysql 5.6, tem a finalidade de armezar todas informações sobre os alunos, instituições, jogos e configurações pertinente a plataforma.

Ao configurar o container, foi necessário permitir acesso remoto ao banco de dados, pois a aplicação poderá não estar no mesmo servidor que a base de dados. Para isto, foi alterado o valor do parametro *bind-address* para *0.0.0.0* no arquivo *my.cnf* permitindo acesso remoto.

Foi configurado um usuário de aplicação que contém as permissões necessárias para poder consultar, inserir, atualizar ou deletar informações na Base de Dados Mapskills, este é o único usuário que tem acesso remoto as informações do Banco de Dados.

Somente o Container Tomcat-BackEnd pode se comunicar com a Base de Dados, desta forma, foi isolado o aesso aos dados nassim a aplicação pode realizar todas operações para que a aplicação realize todas possa ter acesso.

O volume Docker *manager\_mysql* tem a finalidade de armazenar todos os dados de maneira que mesmo que os containers caiam, os dados estão salvos em um diretório a parte.

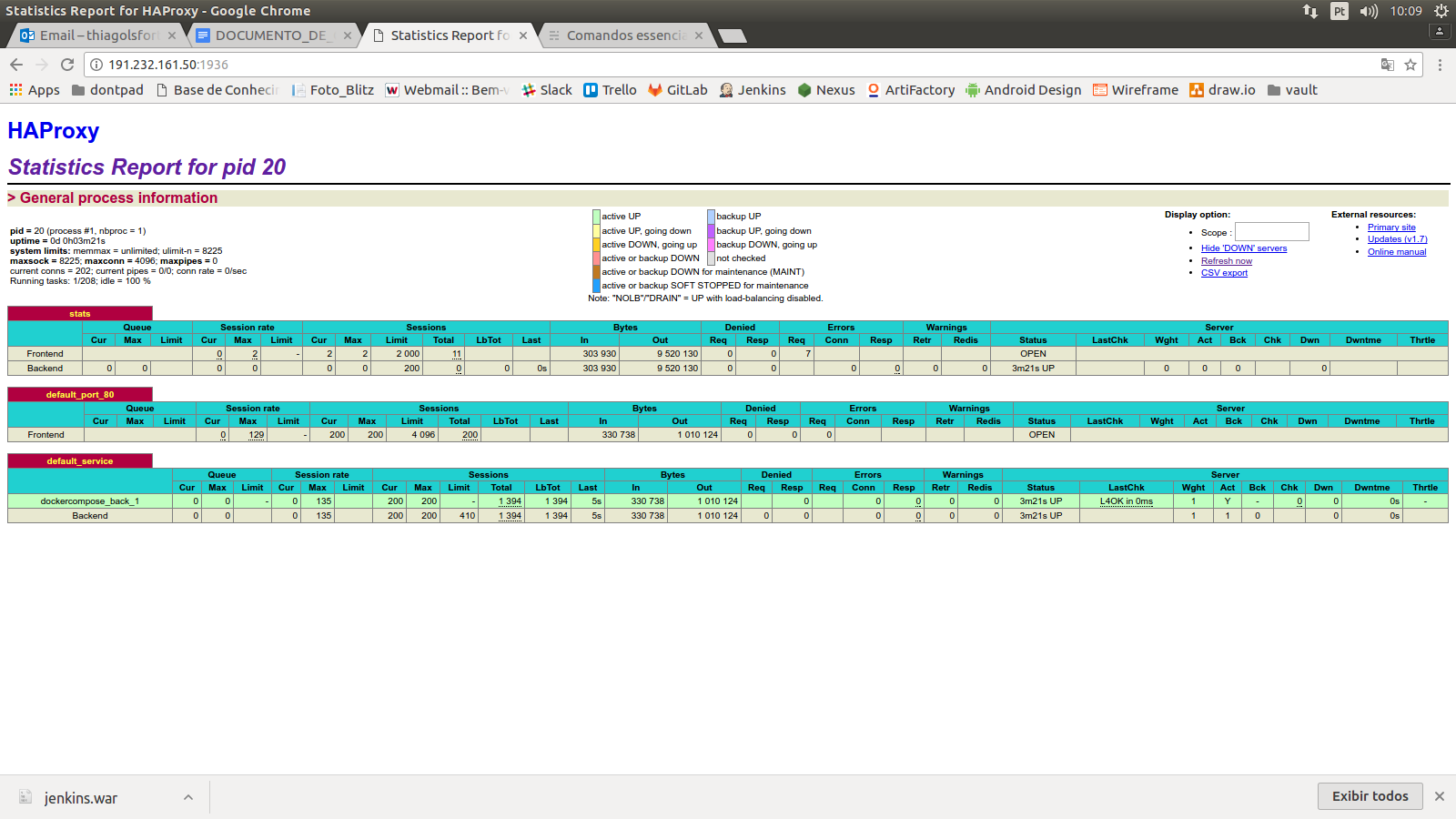
**Mapskills-Haproxy**

Imagem que tem a finalidade gerar o container que irá gerenciar o Load Balance dos Containers de Mapskills-app. Toda requisicao destinada ao Software Mapskills quem controlorá a carga sera o Haproxy por meio do um algoritmo de balanceamento de carga.

O Haproxy trabalhara como um Proxy Reverso, recebendo todas requisicoes atraves da porta 80 e redirecionando internamente a porta 8080 destinada ao Tomcat que contem o mapskills-war. O balanceamento de carga utilizado e de camada 4 (camada de transporte) da tabela OSI, desta forma o Haproxy encaminhara o trafego do usuario com base no alcance e na porta do IP, no caso definida como 80.

Aplicacao tambem fornece um Dashboard para visualizacao dos servers, numeros de requisicoes com sucesso e falha, alem de poder monitorar a quantidade que foi trafegada pela rede, conforme figura abaixo.

Alem da liberacao da porta 80 no host, a porta 1936 tambem foi configurada no Haproxy, pois e atraves dela que pode-ser visualizar o seu Dashboard e monitorar todo ambiente utilizando o usuario Mapskills.



**Round Robin**

Algoritmo utilizado para load balance que tratar os servers como iguais, independente do numero de conexoes solicitadas, sempre redirecionando a proxima requisicao ao server seguinte, desta forma, todos servers terao o mesmo numero de conexoes.

**Docker Compose**

Este arquivo de configuraçao tem a finalidade de automatizar o deploy de todo ambiente de produçao necessario para que o Mapskills funcione. O arquivo docker-compose.yml criara e iniciara todos serviços definidos.

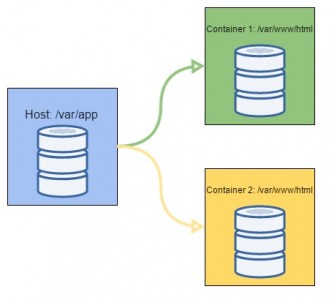
O arquivo inicia os seguintes servicos: Container Nexus dando start ao Repositrio, Container Tomcat para o Front-End, composto pelo arquivo mapskills-web.war, Container Tomcat para o Back-End, composto pelo arquivo mapskill.war, Container Haproxy composto pelo software que ira realizar o Load Balancer, e Container cAdvisor que ira monitorar os recursos utilizado por todos containers.

No cabeçalho deste documento foi definido que utilizara a versao 2 do docker-compose e apos isso os servicos que serao configurados e inicializados.

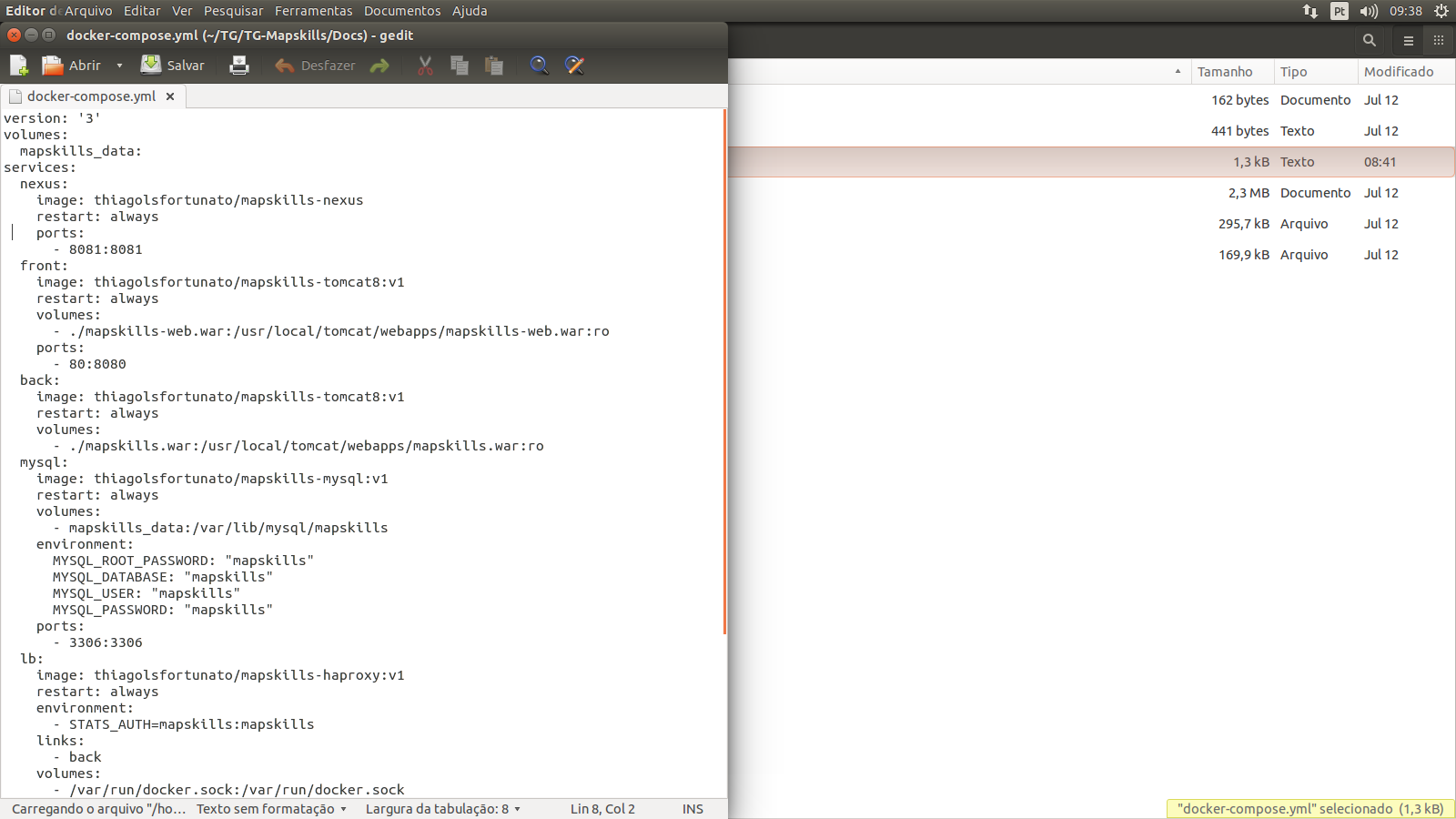


**Volume**

Um volume pode ser tanto um mapeamento de uma pasta local para um container, quanto um container de volume compartilhado entre outros containers.



Foi criado um container de volume para mapeamento da base de dados mapskills, desta forma todos arquivos gerados pelo container mysql serao compartilhados no container, e caso o container venha a sair de operacao, todos os dados foram mantidos no volume.



**Container Haproxy**

Este container e proveniente da Imagem thiagolsfortunato/mapskills-haproxy:v1, sendo restartado toda vez que o docker sair de operacao. Configurada o usuario e senha para acesso a Pagina de Estatisticas do Haproxy, configurado um link entre o Container Haproxy, para que o aconteça o Load Balace no Container Tomcat Back End, criado um volume e compartilhado entre o container e o host o arquivo docker.sock. As portas definidas para acessar a aplicacao remotamente sao 8080:80 e 1936:1936, onde a porta 8080 sera responsavel por redirecionar todas requisicoes que tiverem como destino os Containers Tomcat Back End e mapear internamente para a porta 80, acontecendo de forma parecida com a porta 1936 sendo acessada remotamente e redirecionada para a porta 1936 internamente.

**Capitulo 4**

**Referências**

Lyman,Jay.  https://blogs.the451group.com/opensource/2010/03/03/devops-mixing-dev-ops-agile-cloud-open-source-and-business/ 451 CAOS Theory.

  Nasrat, Paul. <https://www.infoq.com/presentations/agile-infrastructure. InfoQ. 05 March 2010>.

http://www.devmedia.com.br/gestao-de-projetos-e-integracao-continua-com-devops/34180

http://www.ibm.com/developerworks/br/rational/library/defining-deployment-deliverable-devops/

http://blog.justdigital.com.br/devops-qual-a-diferencas-entre-continuous-delivery-continuous-integration-e-continuous-deployment/

<http://www.infowester.com/cloudcomputing.php>

Catteddu, Daniele. "Cloud Computing: benefits, risks and recommendations for information security." Web Application Security. Springer Berlin Heidelberg, 2010. 17-17.

Humble, Jez, and David Farley. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.

http://www.dell.com/learn/br/pt/brbsdt1/sb360/what-is-a-server