FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS

FATEC PROFESSOR Jessen Vidal

MARCELO AUGUSTO DOS SANTOS

LABORATÓRIO DE PROJETO DE BANCOS DE DADOS - DEVOPS

Orientador: Me. Eduardo Sakaue

São José dos Campos

2022

SUMÁRIO

[1 Introdução 3](#_Toc102424188)

[1.1 Definição do problema 3](#_Toc102424189)

[1.2 Objetivo 3](#_Toc102424190)

[2 Desenvolvimento 5](#_Toc102424191)

[2.1 Arquitetura 5](#_Toc102424192)

[2.2 Qualidade e testes 8](#_Toc102424193)

[2.3 Detalhes 10](#_Toc102424194)

[3 Resultados e Discussão 11](#_Toc102424195)

# Introdução

Este Capítulo apresenta o problema a ser resolvido e o objetivo deste trabalho.

## Definição do problema

O desenvolvimento “tradicional” de um *software* ou solução computacional é geralmente composto por uma ou mais equipes que possuem desenvolvedores. Estes, irão trabalhar individualmente em tarefas – geradas a partir de requisitos – do início ao fim do projeto. O projeto pode passar por uma fase de *design* que é geralmente executada por uma pessoa ou equipe e depois de concluído o trabalho, inicia-se uma nova fase.

O maior problema que pode ser enfrentado nesse modelo é que não há fluidez perfeita de um estágio de desenvolvimento para o outro. Muitas dessas etapas provavelmente deveriam ser desenvolvidas ao mesmo tempo ou podem passar por revisões e quando isso acontece, pode haver falhas ou atrasos de comunicação de uma equipe para outra.

Trabalhando isoladamente desta forma, há demasiada perda de tempo, eficiência e geração de trabalho extra. Desta forma também, é muito mais complicado e oneroso fazer alterações ou corrigir erros, já que não é possível obter *feedbacks* antes de estar num ciclo mais avançado do desenvolvimento.

Este formato de desenvolvimento agride de forma direta a qualidade e garantia da entrega do *software*, pois é muito mais difícil de testar e corrigir de forma rápida.

O problema é...

Este Trabalho de Graduação atende os critérios estabelecidos na disciplina Laboratório de Projeto de Banco de Dados.

## Objetivo

O objetivo deste trabalho consiste em modernizar a forma como um software é desenvolvido, testado e entregue, visando a qualidade do produto e da entrega de forma contínua.

Para que isso seja possível, será utilizada a metodologia *DevOps*.

A utilização dessa metodologia e da cultura de *DevOps* oferece diversos benefícios como o ganho de integração com as equipes, agilidade no desenvolvimento, testes e sobretudo, o alinhamento das entregas.

*DevOps* vem da combinação de “desenvolvimento” e “operações” e a partir daí já é possível observar a integração da equipe de desenvolvimento em todas as suas fases com a de equipe de operações. No entanto, essa metodologia representa um acervo de ideias e boas práticas que vão além do significado dessas duas áreas. Quando falamos de *DevOps*, falamos também de segurança, trabalho colaborativo, análise de dados e diversos outros conceitos.

A metodologia fornece processos e abordagens para ajudar a acelerar o processo necessários para levar uma ideia de desenvolvimento de um software a implantação, onde há entregas de valor para o usuário.

# Desenvolvimento

Apresente neste Capítulo detalhes sobre o produto desenvolvido para cumprir com o objetivo proposto.

## Arquitetura

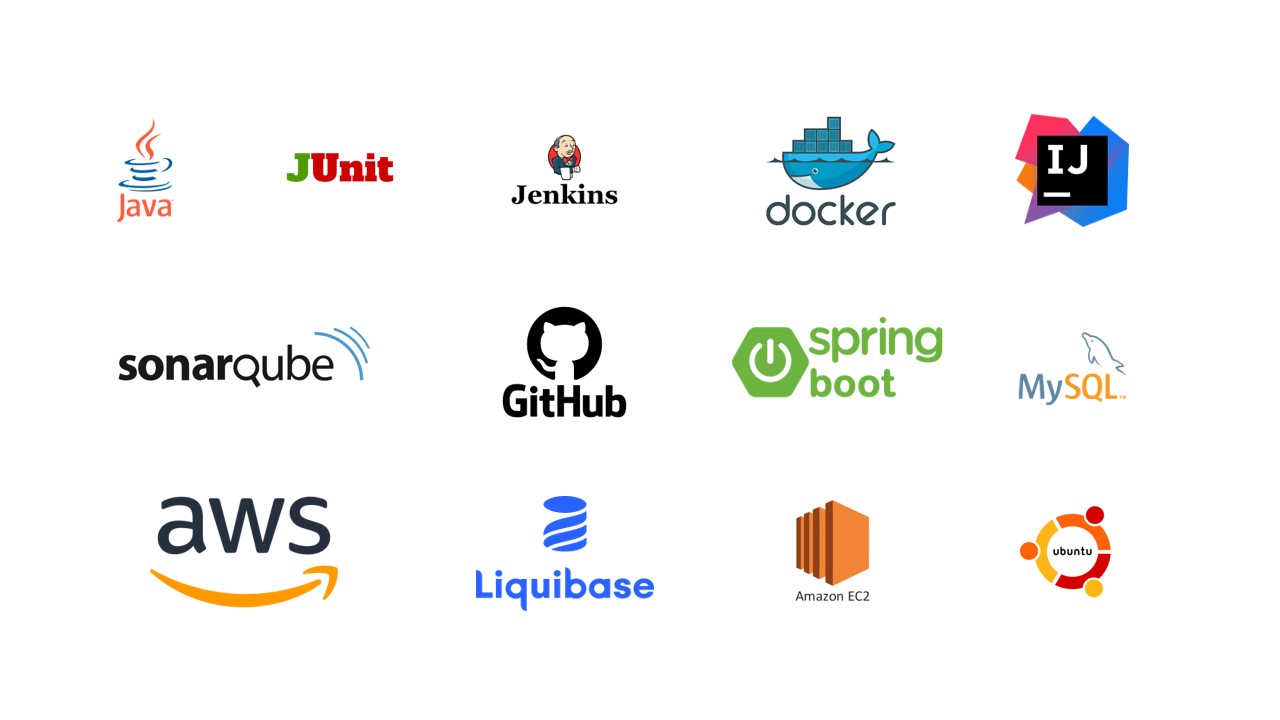
Antes de qualquer coisa, para este projeto, o ambiente de produção e onde ficará a estrutura necessária para executar todos os passos do *DevOps*, serão todos hospedados na *Amazon Web Services*.

Pela quantidade de processos que serão executados e tecnologias utilizadas, será utilizada uma instância do tipo t2.medium com 4 Gb de RAM e 2 CPUs. O custo médio por hora é de 0.0744 dólares - um valor irrisório para execução em poucas horas, mas que pode escalar conforme a utilização aumentar.

Nessa instância serão instalados todos os *softwares* necessários para o ambiente de execução dos passos do DevOps e também para que a aplicação seja executada em produção. Alguns *softwares* serão instalados manualmente para os demais possam ser instalados depois via Docker, como o Jenkins e MySQL.

Para que seja possível automatizar o máximo de tarefas e etapas possíveis do ciclo de desenvolvimento de um *software*, serão utilizadas diversas ferramentas diferentes que possuem também objetivos diferentes.

**Figura 1** – Aplicações utilizadas



No caso desse projeto, será criada uma API (*Application Programming Interface*) para interação de uma interface de usuário com o banco de dados. A API será desenvolvida em Java utilizando Springboot. O banco de dados que armazenará estes dados será o MySQL. Para efetuar os testes unitários e de integração da aplicação, será utilizado o plugin JUnit.

É importante também que o banco de dados possua versões para que caso ocorra algum problema na arquitetura do banco, seja possível reverter para uma versão anterior. Para que isso seja possível, a ferramenta utilizada será o Liquibase, um plugin para Java.

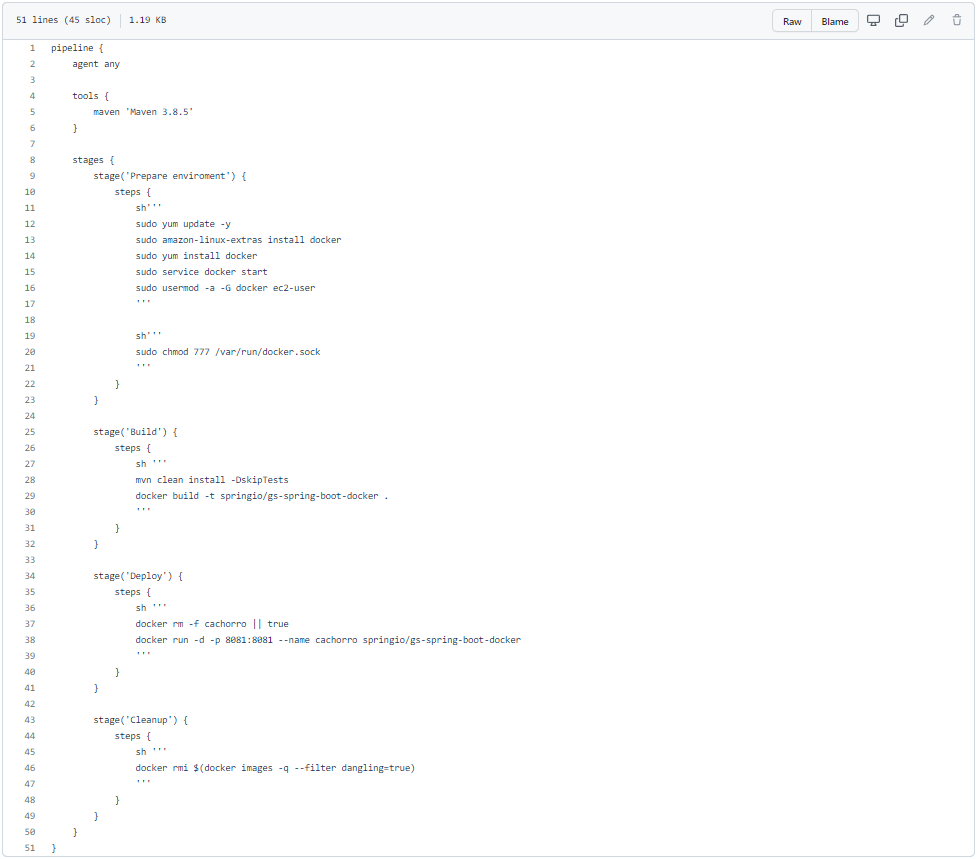
As ferramentas citadas acima irão auxiliar na automatização de diversas etapas do desenvolvimento e teste de qualidade do software. Há ainda duas partes muito importantes neste ciclo que são a integração e a entrega contínuas. Existem diversas formas e ferramentas possíveis para estes casos e para este específico, a ferramenta será o Jenkins.

Com o Jenkins, é possível criar *pipelines* e *jobs* para sequenciar etapas consecutivas do desenvolvimento, teste, integração e entrega desse *software*.

No projeto em desenvolvimento, será criada uma *pipeline* para fazer *download* das dependências, configuração do ambiente e disponibilização da API e um *job* para realizar a verificação de qualidade de código e realização de testes de integração e unitários.

A configuração da *pipeline* será feita através de um arquivo de configuração chamado Jenkinsfile, que possui todas as instruções necessárias para executar os passos necessários para integração contínua e entrega do *software*.

Figura 2 – Jenkinsfile

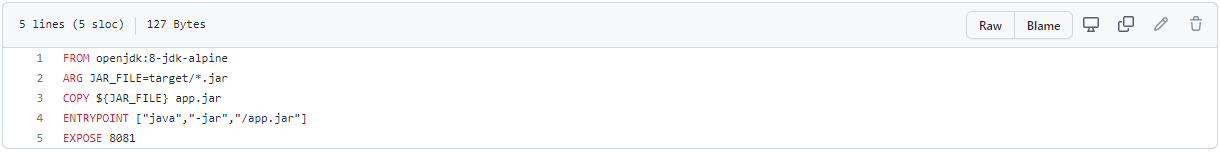


Para que não ocorram problemas de configurações e aplicações funcionarem em uma máquina e não em outra, será utilizado o sistema de *containers*, com o Docker.

Com a utilização do Docker, podemos ter aplicações e ambientes previamente configurados e encapsulados que facilitam o processo de instalação de ferramentas do ambiente de produção além do funcionamento do próprio *software*.

Para que uma aplicação possa ser configurada e acessada de forma externa ao *container*, precisamos configurar as portas abertas e aplicar configurações iniciais através de parâmetros e isso pode ser feito utilizando um arquivo Dockerfile.

**Figura 3 –** Dockerfile



## Qualidade e testes

Sendo praticamente todo conteúdo possível automatizado, é possível também validar a qualidade de código e realizar testes automatizados.

Para realizar testes unitários, será utilizado o plugin para Java, o JUnit. Com ele podemos criar diversos casos de teste e usando o Maven podemos, através da linha de comando, executar estes testes. Esse comando pode ser incorporado ao job de teste do Jenkins, onde serão executados também os scanners de qualidade de código.

**Figura 4 –** Casos de teste do JUnit



Utilizando o SonarQube, podemos realizar verificações de possíveis bugs, vulnerabilidades e melhorias de código como remoção de *imports* não utilizados, código duplicado, cobertura de código etc.

O SonarQube é uma aplicação que podemos configurar localmente e possui uma interface onde é possível visualizar toda análise feita por ele e as sugestões de como realizar as mudanças para melhorar a qualidade e performance da aplicação em nível de codificação.

**Figura 5 –** Dashboard do SonarQube

(IMAGEM DO DASHBOARD DO SONARQUBE)

O DockerHub possui uma imagem Docker para o SonarQube e isso facilita para que possamos configurar a maior parte de seu funcionamento inicial usando um comando com parâmetros ou um arquivo de configuração executável (.sh ou .bat).

**Figura 6 –** Script de inicialização do *container* do SonarQube

(TRECHO DE CÓDIGO DO CONTAINER SONARQUBE)

## Detalhes

Apresente nesta Seção diagramas e trechos de código importantes para o entendimento do produto desenvolvido. Apresente pelo menos um trecho de código exemplificando cada tecnologia utilizada.

Todo trecho de código deve ser comentado (use marcações ou numeração de linhas para facilitar a explicação). É altamente recomendada a utilização de listas nas explicações.

# Resultados e Discussão

Apresente neste Capítulo uma discussão sobre cada uma das tecnologias utilizadas no desenvolvimento do produto. Indique a motivação para utilização da tecnologia e como ela auxiliou no desenvolvimento. Apresente também que problemas foram encontrados e como foram solucionados.

Se há um cliente, indique se a solução foi implantada e qual a opinião dos usuários.