Implantação de DevOps e *pipelines* automatizadas em um projeto monolítico

Dennis Rojas Pereira

SUMÁRIO EXECUTIVO

O produto a ser analisado é um software de gestão de saúde onde é utilizado nos Postos de Saúde e Unidades de Pronto Atendimento.

Os usuários são funcionários administrativos, farmacêuticos e médicos, sendo alguns funcionários públicos e outros de uma empresa terceirizada.

As principais atividades do software são:

- Controle de agenda de consultas : onde o médico passa para o funcionário administrativo qual dia da semana ele vai atender.
- Marcação de exames e consultas : onde o munícipe vai até o balcão do Posto de Saúde e solicita ao funcionário administrativo.
- Lançamento de vacinas e campanhas : o funcionário administrativo realiza os lançamentos das vacinas aplicadas.
- Lançamento de teto de gastos em exames com laboratórios e clínicas conveniadas : o funcionário administrativo consegue parametrizar o sistema para que não seja possibilitado a marcação de exames quando atingir determinado valor.
- Prestação de contas ao governo por meio de um arquivo txt seguindo uma tabela chamada SIGTAP fornecida pelo DATASUS : O funcionário administrativo gestor, exporta em um arquivo e envia para o governo para prestação de contas.

PROBLEMATIZAÇÃO

O software é feito em Java e sua arquitetura é monolítica , onde existe uma única camada de aplicação no qual, a interface de usuário e código de acesso aos dados ficam em um único software. Caso haja mudança em alguma parte do software, causaria um tempo de indisponibilidade para que seja atualizado.

Existem dois times que trabalham no projeto são eles, suporte e desenvolvimento. Quando acontecem problemas ou requisições novas elas são direcionadas ao time de suporte que analisa-as e encaminha para o desenvolvimento, sem nenhuma plataforma e gestão de requisição.

O nível de maturidade exigida no software é bem grande, pois todas as áreas do software são totalmente acopladas, ou seja, a interdependência entre componentes no sistema é bem alta onde qualquer parte do código depende da outra. Para um novo desenvolvedor integrar a equipe deve fazer um treinamento e conhecer todas as partes do sistema para começar a interagir e colaborar com o grupo de desenvolvimento.

O software roda em um servidor dentro do próprio data center do cliente (on-premisses) e caso aumente a demanda será necessário trocar o hardware para atender.

Toda funcionalidade nova ou erro corrigido vai diretamente no servidor, sendo testado pelo próprio desenvolvedor em sua máquina, ocorrendo às vezes incompatibilidade de funções utilizadas por ser um sistema operacional diferente ou rodando em um ambiente diferente do de produção.

JUSTIFICATIVA E OBJETIVO GERAL

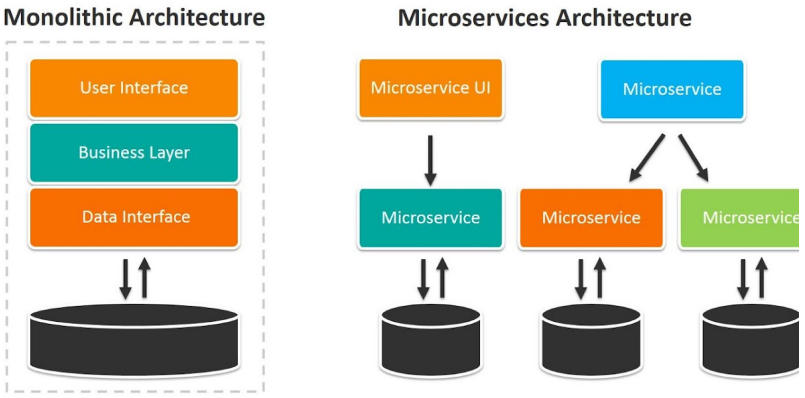
Este trabalho busca através de estudo de caso levantar os problemas existentes no projeto e encontrar uma solução, assim avaliando mudanças de cultura da organização e na interação entre equipes de suporte e desenvolvimento.

Alguns problemas encontrados como extensibilidade para novas requisições, disponibilidade e escalabilidade será necessária uma adequação do time para trabalhar com os desafios das práticas Devops e com microserviços para que a curva de aprendizado e a curva de início de integração com o time seja mais curta.

FUNDAMENTAÇÃO TEÓRICA

Segundo o Google em seu portal de Centro de Arquitetura, os microsserviços precisam ser projetados para recursos de negócios, não para camadas horizontais, como acesso a dados ou mensagens. Os microsserviços também precisam ter acoplamento flexível e coesão funcional alta. Os microserviços são acoplados com flexibilidade se você puder alterar um serviço sem exigir que outros serviços sejam atualizados ao mesmo tempo. Um micro serviço é coeso se tiver uma única finalidade bem definida, como gerenciar contas de usuário ou processar pagamentos.

Primeiramente seria uma adequação do time para trabalhar com micro serviços para que a curva de aprendizado e a curva de início de integração com o time seja mais curta. Sendo assim também o impacto em uma atualização, não será no software inteiro e sim em uma parte.



(DIFERENÇA..., 2021)

Após isto a cada alteração de sistema ou nova funcionalidade é gerido pelo Scrum. O método ágil Scrum tem como objetivo, segundo Schwaber, definir um processo para projeto e desenvolvimento de software orientado a objeto, que seja focado nas pessoas e que seja indicado para ambientes em que os requisitos surgem e mudam rapidamente. O Scrum também é considerado um método específico para o gerenciamento do processo de desenvolvimento de software. Com isto vamos ter um backlog que seria uma lista de requisitos e cada semana entregue chamada de Sprint.

Para o controle de requisições será feito que segundo o Site do Gitlab acessado em Julho de 2022 O GitLab é um gerenciador de repositório de software baseado em git, com suporte a Wiki, gerenciamento de tarefas e CI/CD. GitLab é similar ao GitHub, mas o GitLab permite que os desenvolvedores armazenem o código em seus próprios servidores, ao invés de servidores de terceiros. Será dividido em 5 projetos onde teremos o projeto de microservice-users,microservice-exams,microservice-appointments e front-end em React.

A criação de pipeline DevOps é uma forma automatizada de levar desde a requisição do pedido de alteração ou nova funcionalidade passando por um controle de versão até as mãos do usuário que necessita do recurso. Sendo necessário ter uma pipeline de CI/CD que inclui o monitoramento e automação de processos no desenvolvimento de aplicações, aplicando especialmente nos estágios de integração, teste, entrega e implantação.

Segundo o livro Jornada Devops 2 edição A automação de um teste consiste na utilização de um software para automatizar as tarefas presentes na execução de testes e com a automação onde a execução de um script codificado é realizada pela máquina garantindo a rapidez e a precisão necessárias para qualidade. Além disso, quanto maior o software ou a complexidade de regras, mais o esforço em executar os testes manualmente antes de cada necessidade de implantação.

Após isto inserir pilares de DevOps onde a cultura, automação, avaliação e compartilhamento serão essenciais para os times. Sendo assim teremos um modelo de serviço mais integrado, times multifuncionais e agilidade nas entregas sendo elas com qualidades com automação de tarefas manuais e reduzindo o down-time na atualização ou em problemas para que os usuários não sejam afetados.

Continuous Deployment é um processo sem intervenção humana, que faz todas as validações anteriores para disponibilizar a nova modificação em produção de forma automática. Apenas uma falha nos testes impede sua implementação na produção.

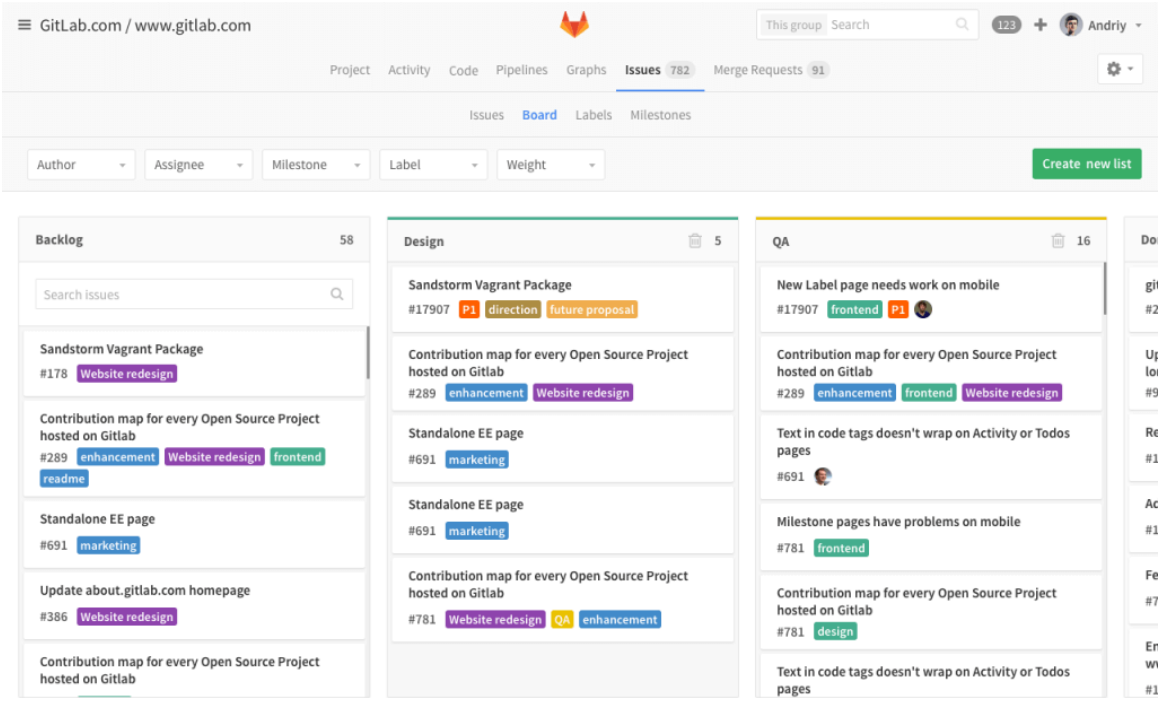
É fundamental validar com a área de segurança da informação e compliance da sua empresa a aplicabilidade deste processo. Operações críticas, que trabalham com deploys contínuos (dezenas, centenas ou milhares por dia), o Continuous Deployment é fundamental.

PERCURSO METODOLÓGICO DA INTERVENÇÃO

Serão escolhidas algumas ferramentas de CI Continuous Integration e CD Continuous Delivery para auxiliar nestas etapas atrelado a um software de gerenciamento de tickets e projetos para realização de conceitos ágeis como o SCRUM e KANBAN.

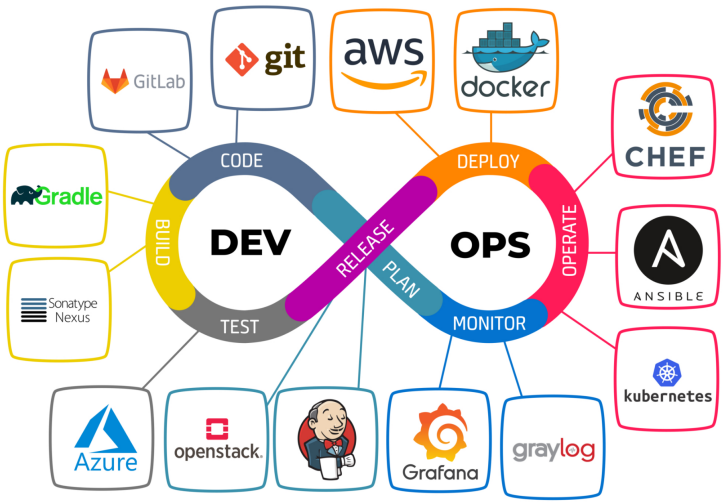
A cada solicitação será gerado um ticket por projeto onde será possível que de qualquer lugar do mundo, profissionais contribuam no projeto. Sendo possível ver todas as *issues* em um quadro Kanban, garantindo uma atualização visual sobre cada status de cada uma delas que pode ser em a fazer, em andamento, em revisão ou finalizada.

Quando o código for alterado em seu commit é necessário colocar qual ticket e projeto está fazendo parte da mudança, sendo possível rastrear quando foi feita a mudança e de qual solicitação ela partiu.



(GITLAB GERENCIAMENTO DE CÓDIGO E PROJETOS, 2021)

Estrutura proposta para o projeto seria conforme abaixo:



(COMO REFATORAR..., 2019)

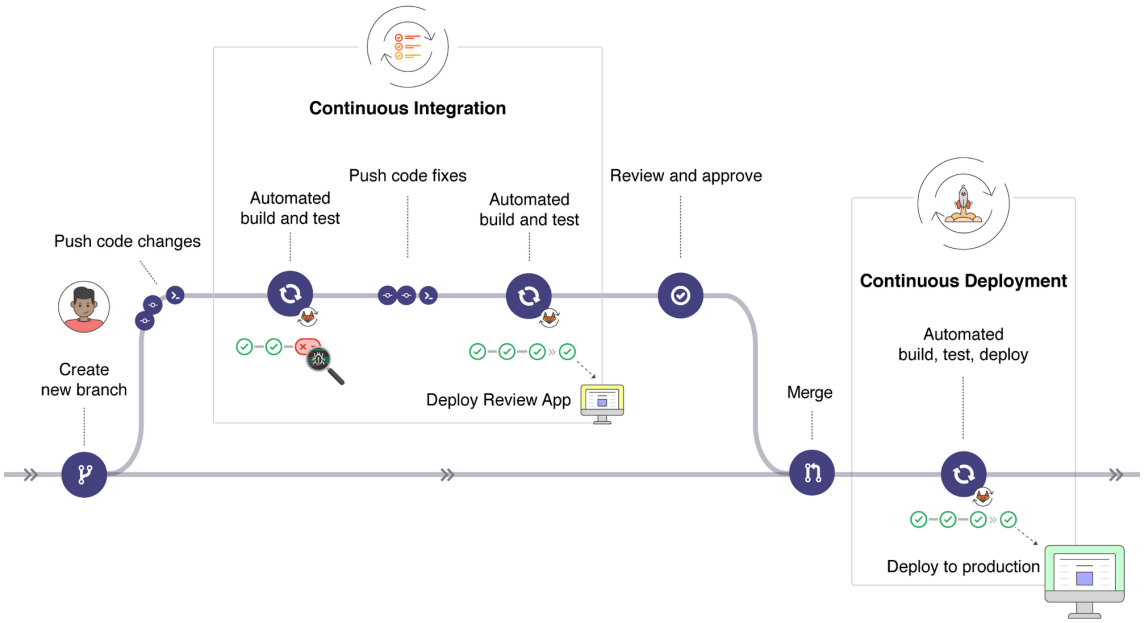
A figura acima indica que todo o código vai ser versionado por um protocolo chamado GIT onde todos os desenvolvedores conseguiram contribuir ao mesmo tempo, sem ficar esperando que alguém acabe para começar a alterar o código sendo armazenados no próprio GitLab, que além de ser uma ferramenta de armazenamento de código e versionamento ela também inclui processos de entrega contínua.

Com a automação todas as mudanças serão testadas e passarão por um processo de validação automática.A ferramenta escolhida para armazenar artefatos gerados a cada build este caso é o Nexus. Após gerar um artefato, ferramentas de entrega contínua após feitos testes manuais para ver se todo o sistema está correto com seus fluxos básicos será disponibilizado em produção.

Outra ferramenta que será colocada no processo de build e de versionamento de código será SonarQube, que garante a qualidade do código fonte em desenvolvimento realizando diversas análises durante o processo de compilação da aplicação detectando por exemplo:

- Trechos de código fonte que podem gerar bugs
- Duplicidade de linhas de comando, prevenindo a repetição de instruções desnecessárias.
- Segurança

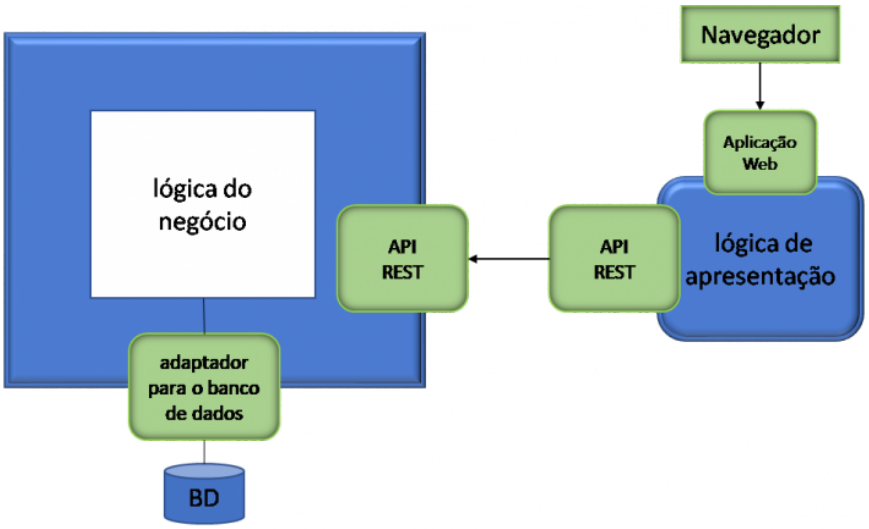
Estas análises atendem a métricas de qualidade configuradas na própria ferramenta seguindo alguns padrões definidos e facilmente customizáveis. A análise pode ocorrer em momento de build da aplicação ou a cada requisição de mudança de código chamado de Merge Request.



(CONHECENDO O GITLAB,2021)

Primeiramente será feita a separação da camada de apresentação da camada de negócio e dados. Isso ocorre através de uma API REST onde cada micro serviço vai responder via HTTP e encapsularam cada funcionalidade de negócio e dados.

A escalabilidade do sistema também será melhorada onde separando o monolítico em microserviços gradativamente teremos um sistema Stateless onde cada requisição que será feita pelo navegador é de forma independente, não existindo uma requisição que guarde um estado em uma sessão.



(AÇÕES DE MIGRAÇÃO DE UM MONOLITO,2020)

Após esta divisão cada micro serviço terá seu próprio banco e seu processo único tendo uma forte separação de responsabilidade sendo que cada solicitação passará antes por uma API Gateway que serve para direcionar as chamadas com uma tabela de roteamento. Com isto podemos colocar nossos serviços em *containers Dockers* facilitando a sua portabilidade pois cada imagem conta com suas respectivas dependências e pode ser orquestrada conforme a demanda a partir de um orquestrador.

Esta imagem pode ser utilizada tanto numa instância na nuvem ou em um servidor dedicado, um ganho bem importante é que a criação do container pode ajudar na recuperação de desastres onde uma atualização publicada deu algum problema basta voltar a imagem do ambiente anterior do Docker e facilidade de subir instâncias em alguma falha de hardware sendo possível restaurar o backup em outro local.

RESULTADOS ESPERADOS

Contribuir com a organização, através de um plano de Devops, uma vez que o mesmo terá impactos positivos.Sendo inicialmente percebidas melhorias que acarretarão no melhor fluxo de cada mudança do software.

Organização de células de desenvolvimento autônomas capazes de desenvolver incrementos de software e redução de tempo de integração e aprendizagem de novos integrados no time de desenvolvimento.

As melhorias apresentadas trarão uma redução nos tempos de entrega, aceleração da aprendizagem coletiva e entregas contínuas mais assertivas e com maior valor agregado. Sendo que alguns impactos negativos são a mudança organizacional e de processos sendo eles bem discutidos antes com a diretoria.

REFERÊNCIAS

DIFERENÇA ENTRE ARQUITETURA MONOLÍTICA E MICROSSERVIÇOS. [S. l.], 2021. Disponível em: https://blog.devnow.dev.br/2021/02/diferenca-entre-arquitetura-monolitica.html. Acesso em: 20 jul. 2022.

COMO REFATORAR UM MONOLÍTICO EM MICRO SERVIÇOS. [S. l.], 2022. Disponível em: https://cloud.google.com/architecture/microservices-architecture-refactoring-monoliths?hl=pt-br. Acesso em: 20 jul. 2022.

Antonio Muniz, ,Analia Irigoyen , Jornada DevOps 2a edição.

SCHWABER, K. The enterprise and Scrum. Ed. Microsoft, 2007

COMO REFATORAR um monolítico em microsserviços. [*S. l.*], 2019. Disponível em: https://blog.4linux.com.br/qual-o-nivel-de-maturidade-devops-da-sua-empresa/. Acesso em: 20 jul. 2022.

CONHECENDO o GitLab CI/CD.[S.l.]2021. Disponível em: https://blog.cubos.io/conhecendo-o-gitlab-ci-cd/. Acesso em 24/08/2022.

AÇÕES DE MIGRAÇÃO DE UM MONOLITO.[S. l.], 2020. Disponível em: https://www.opus-software.com.br/acoes-de-migracao-de-um-monolito-para-microsservicos/. Acesso em 24/08/2022.

GITLAB Gerenciamento de Código e Projetos. Disponível em https://spiritsec.zendesk.com/hc/pt-br/articles/360001865777.Acesso em 24/08/2022.