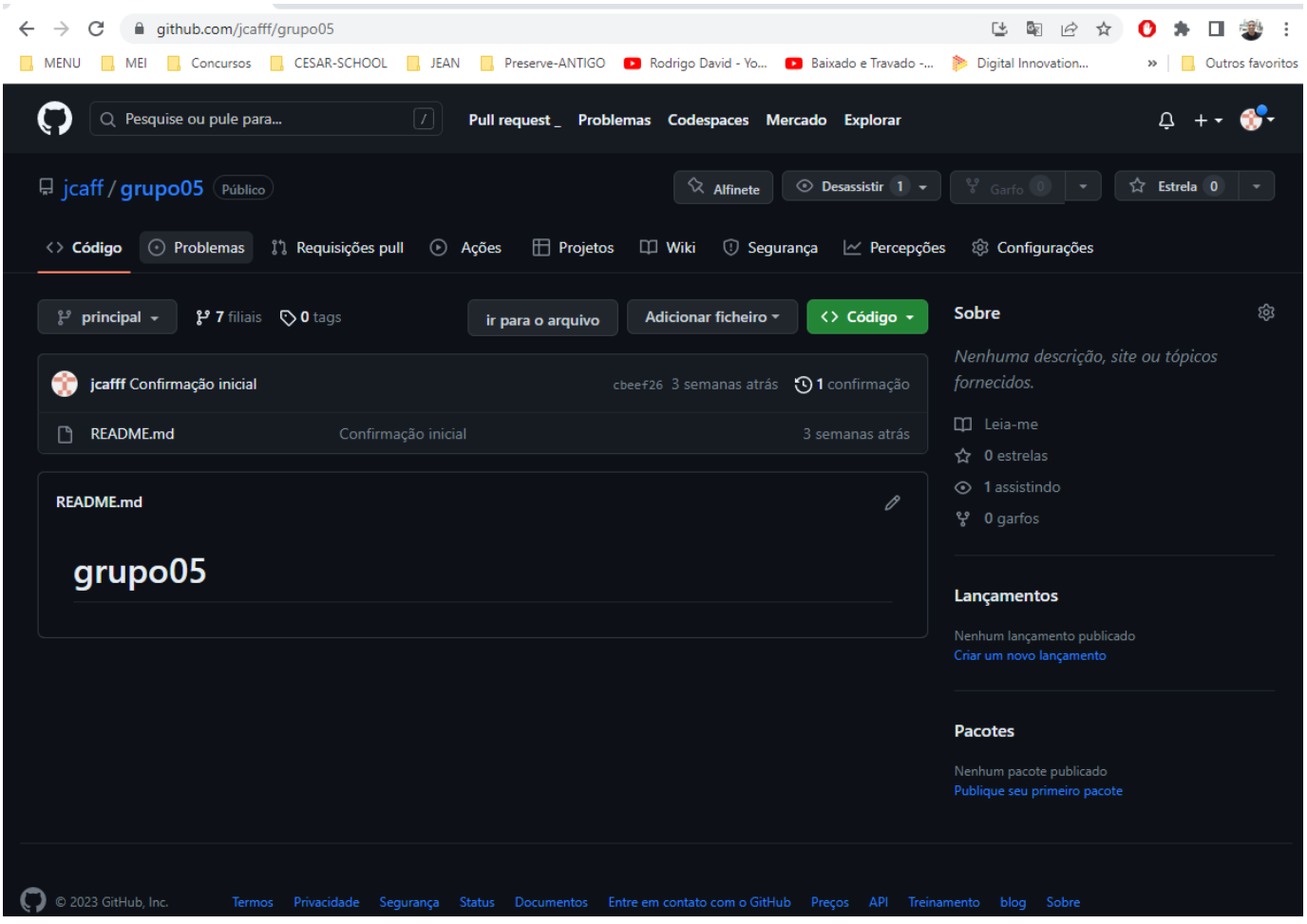
1. Grupo 5Tech S/A

Somos uma empresa criativa para o desenvolvimento de software mobile focada no segmento de pet, visando fornecer soluções tecnológicas avançadas, melhorando a vida dos proprietários de animais de estimação e seus companheiros peludos.

2. Repositório



https://github.com/jcafff/grupo05

3. Descrevendo a empresa e o (app)

Nós desenvolvemos um aplicativo (**PetHungry**) que atende especificamente proprietários de animais de estimação, oferecendo soluções de alimentação e nutrição.
Nosso aplicativo é fácil de usar, intuitivo e altamente eficaz, fornecendo informações precisas e atualizadas sobre a alimentação ideal para seu animal de estimação, levando em consideração sua idade, raça e necessidades nutricionais. Atualmente, nosso aplicativo está disponível exclusivamente para dispositivos Android, mas estamos trabalhando duro para expandir para outras plataformas no futuro.

Missão

Nossa missão é ajudar os proprietários de animais de estimação a cuidar melhor dos seus companheiros, oferecendo soluções inovadoras de software móvel para o segmento de pet food. Seja através de nossa tecnologia de ponta ou nosso atendimento ao cliente excepcional, estamos comprometidos em proporcionar aos nossos clientes uma experiência única e satisfatória em todos os aspectos.

Visão

Ser reconhecido como o principal fornecedor de soluções tecnológicas avançadas para o segmento de pet food, criando valor para os proprietários de animais de estimação e seus companheiros.

Valores

Compromisso com a excelência: Buscamos a excelência em tudo o que fazemos e nos esforçamos para oferecer soluções de software mobile inovadoras e de alta qualidade aos nossos clientes.

Foco no cliente: Nosso foco principal é entender as necessidades de nossos clientes e seus animais de estimação, e fornecer soluções personalizadas e orientadas para o cliente.

Inovação: Estamos constantemente em busca de inovação e novas tecnologias para melhorar nossas soluções e serviços.

4. ISSUES, Resumo de testes e Estratégia de deploy

a. Nomenclatura das ISSUES

Nome	Descrição
[DEV-ANDROID]	para questões relacionadas ao desenvolvimento para o sistema operacional Android;
[BUGFIX-ANDROID]	para correção de bugs específicos para o sistema operacional Android;
[FEATURE-ANDROID]	para novas funcionalidades específicas para o sistema operacional Android;
[HOTFIX-ANDROID]	para correções emergenciais específicas para o sistema operacional Android;
[IMPROVEMENT-ANDROID]	para melhorias no software específicas para o sistema operacional Android;
[DOCS-ANDROID]	para questões relacionadas à documentação específicas para o sistema operacional Android;
[PERFORMANCE-ANDROID]	para mudanças de código focadas em melhorar o desempenho para o sistema operacional Android;
[TEST-ANDROID]	para adicionar ou corrigir testes específicos para o sistema operacional Android.

Assim, é possível identificar de forma clara e rápida o tipo de questão e o sistema operacional específico em que ela se aplica. Além disso, é importante manter essa nomenclatura padronizada e documentada para garantir a clareza e a organização do processo de desenvolvimento e correção de problemas no software.

b. Testes

O processo de teste é uma etapa fundamental no desenvolvimento de software, pois é através dele que é possível garantir que o produto final esteja funcionando conforme a expectativa do cliente final. Nesse sentido, adotaremos uma abordagem de testes manuais para garantir que o software esteja conforme as especificações e requisitos definidos.

A equipe de qualidade será responsável por executar os testes manuais em cada funcionalidade entregue, verificando se tudo está funcionando corretamente. Isso inclui a execução de testes funcionais, como testes de aceitação, exploratórios e regressão, para garantir que o software atenda aos requisitos e critérios de aceite especificados.

O software será testado em diversos ambientes, com finalidades específicas:

1. **Ambiente de Desenvolvimento:** normalmente onde o software é desenvolvido, ou seja, na máquina local do desenvolvedor. Neste ambiente, serão realizados testes unitários e de integração para garantir que a funcionalidade implementada por ele esteja funcionando corretamente e integrada com as demais partes do software.

2. **Ambiente de QA:** ambiente onde serão realizados os testes funcionais para verificar se o software está atendendo aos requisitos e critérios de aceite especificados. Aqui serão realizados testes manuais, incluindo testes de aceitação, exploratórios e regressão, para garantir que o software esteja funcionando corretamente.

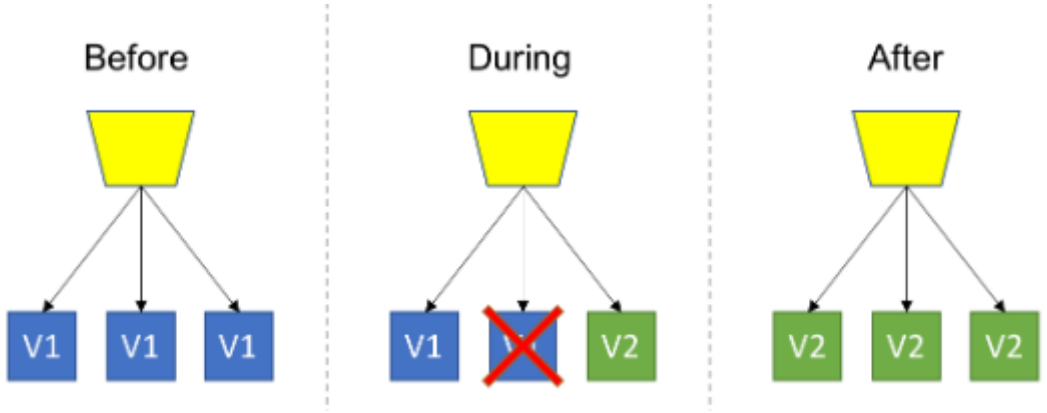
3. **Ambiente de Homologação:** neste ambiente, os testes serão realizados pelo cliente e/ou usuário final antes da disponibilização do software em ambiente de produção. Aqui serão realizados testes manuais para garantir que o software esteja conforme as especificações e requisitos definidos.

4. **Ambiente de Produção:** este é o ambiente no qual o software é disponibilizado para uso real pelos usuários. Neste ambiente, será possível realizar o monitoramento contínuo do software, verificando a disponibilidade, desempenho e segurança para garantir que tudo esteja funcionando segundo as expectativas dos usuários finais.

A abordagem de testes adotada garante que o software estará em condições testáveis com as especificações e requisitos definidos, reduzindo o risco de erros e problemas no uso real pelos usuários.

c. Estratégias de deploy

Nossa equipe optou por usar o modelo de **Rolling Upgrade** ou entrega contínua. É uma estratégia na qual consiste em atualizar o código de forma gradual até que todas as instâncias possuam a nova versão, sendo assim, possível mensurar os impactos de forma mais sutil e avançar conforme as possibilidades.



A equipe de desenvolvimento adotará um modelo de entrega contínua, o que significa que o software será entregue em pequenas e frequentes atualizações. Esse modelo garante que novas funcionalidades e correções de bugs sejam entregues aos usuários finais com mais frequência e de forma mais consistente.

Para suportar a entrega contínua, a equipe utilizará quatro ambientes distintos: desenvolvimento, QA (garantia de qualidade), homologação e produção. Inicialmente, as atualizações serão enviadas para o ambiente de QA, onde serão testadas e validadas pela equipe de qualidade antes de serem liberadas para ambiente de homologação. Finalmente, as atualizações serão implantadas no ambiente de produção, onde estarão disponíveis para os usuários finais.

Para garantir que o processo de deploy seja rápido e confiável, será utilizado um processo automatizado. A automação reduz a possibilidade de erros humanos, tais como esquecimento de seguir roteiros ou ordem de execução dos scripts, o que pode levar a falhas e atrasos no

processo de entrega. A equipe de desenvolvimento priorizará a confiabilidade e segurança do processo de deploy, a fim de garantir que os usuários finais tenham uma experiência de uso de alta qualidade.

Fluxo de Deploy Automatizado

1. A equipe de desenvolvimento fez um commit do código para o repositório Git.
2. O Git dispara um webhook para o serviço de CI/CD (Continuous Integration/Continuous Deployment), como o AWS CodePipeline ou o Jenkins, para iniciar o pipeline de build e deploy.
3. O pipeline de CI/CD compila o código e executa testes automatizados para garantir que a nova versão do software esteja funcionando corretamente.
4. Se os testes passarem, o pipeline de CI/CD cria um pacote de deploy e o envia para o serviço de distribuição de aplicativos móveis para o sistema operacional Android, como o Google Play Console.
5. O serviço de distribuição de aplicativos móveis realiza testes de integração e aceitação em diferentes dispositivos Android.
6. Se os testes de distribuição passarem, o serviço de distribuição de aplicativos móveis libera a nova versão do aplicativo para os usuários finais que utilizam dispositivos Android.

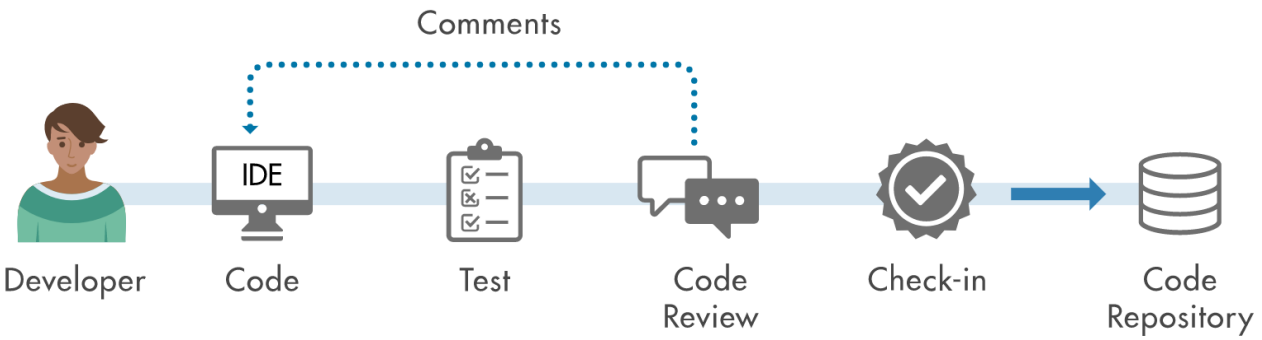
5. Importância do Code Review

O processo de Code Review é importante, pois permite que os desenvolvedores possam se ajudar mutuamente a melhorar a qualidade do código produzido. Durante essa revisão, é possível compartilhar conhecimentos e experiências entre os membros da equipe, o que contribui para o desenvolvimento profissional de todos.

Desta forma, obtemos a redução de tempo e custos de manutenção de código, visto que possíveis erros podem ser identificados e corrigidos nesse processo, evitando problemas críticos e complexos no futuro. Além disso, a revisão do código também ajuda a manter a consistência e a padronização do software, garantindo uma melhor compreensão e colaboração entre os desenvolvedores.

Por fim, é importante destacar que a estratégia de Code Review deve ser documentada e adotada como uma prática padrão. Isso porque, além de garantir a qualidade do código, essa

prática também ajuda a aumentar a eficiência e a produtividade da equipe de desenvolvimento, resultando em um software mais robusto e confiável.



6. Cronograma

Grupo 5Tech — Powered by monday.com

Projeto Pert Food

Planejamento

Nome	Pessoa	Status	Cronograma - Start	Cronograma - End
Definição dos requisitos		Feito	2023-02-23	2023-04-24
Definição do escopo do projeto		Feito	2023-01-17	2023-04-07
Definição da equipe e recursos necessários		Feito	2023-02-01	2023-05-14
			2023-01-17	**2023-05-14**

Desenvolvimento

Nome	Pessoa	Status	Cronograma - Start	Cronograma - End
Desenvolvimento da primeira funcionalidade		Feito	2023-02-27	2023-05-27
Desenvolvimento da segunda funcionalidade		Feito	2023-03-13	2023-06-21
Desenvolvimento da terceira funcionalidade		Feito	2023-03-27	2023-07-04
			2023-02-27	**2023-07-04**

Geração de build

Nome	Pessoa	Status	Cronograma - Start	Cronograma - End
Build da versão 1.0.0		Feito	2023-04-17	2023-04-22
			2023-04-17	**2023-04-22**

Depoly em ambiente de QA

Nome	Pessoa	Status	Cronograma - Start	Cronograma - End
Testes manuais		Feito	2023-04-24	2023-06-04
			2023-04-24	**2023-06-04**

Deploy em ambiente de homologação

Nome	Pessoa	Status	Cronograma - Start	Cronograma - End
Depoly da versão 1.0.0 no ambiente de homlogação		Feito	2023-05-12	2023-09-08
Teste de aceitação e validação de clientes		Não iniciado	2023-05-15	2023-08-22
			2023-05-12	**2023-09-08**

Deploy em ambiente de produção

Nome	Pessoa	Status	Cronograma - Start	Cronograma - End
Deploy da versão 1.0.0 no ambiente de produção		Não iniciado	2023-05-29	2023-09-18
Monitoramento e ajustes		Não iniciado	2023-05-29	2023-06-30
			2023-05-29	**2023-09-18**

7. Teste e ambiente de teste do software

Nossos analistas de teste, irão testar o aplicativo utilizando dispositivos físicos e emuladores para garantir sua qualidade.

Os testes serão realizados manualmente e automatizados. Para os testes manuais, serão usados diversos dispositivos físicos e emuladores com diferentes resoluções de tela para garantir que o aplicativo seja testado em várias plataformas.

Já para os testes automatizados, o Robot Framework, uma ferramenta de automação de testes de código aberto, será utilizado em conjunto com o Selenium WebDriver para simular ações do usuário e garantir que todas as funcionalidades do aplicativo sejam testadas adequadamente. A linguagem de programação Java será usada para desenvolver scripts de teste.

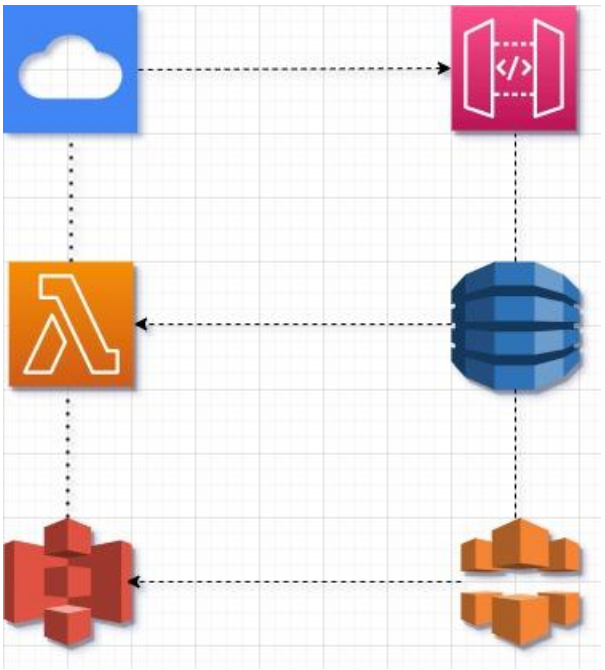
8. Estratégias de CI/CD

CI (Integração Contínua) e CD (Entrega Contínua) são práticas essenciais para garantir a qualidade e a entrega rápida de software. Vamos seguir alguns passos para implementar essa estratégia, como:

1. **Versionamento de Código:** Primeiro, é importante garantir que o código da aplicação seja versionado corretamente usando um sistema de controle de versão, como o Git. O Git permite que vários desenvolvedores trabalhem no mesmo código, sem afetar o trabalho um do outro.
2. **Build Automatizada:** O próximo passo é configurar um sistema de build automatizado que compila o código-fonte e criará o pacote para a aplicação móvel. O sistema de build deve incluir todas as dependências necessárias para compilar o código e deve ser configurado para executar em cada check-in de código.
3. **Testes Automatizados:** Para garantir que a aplicação móvel esteja funcionando corretamente, é necessário realizar testes automatizados (Selenium, Robot Framework ou Java), incluindo testes de unidade, testes de integração e testes de aceitação. Os testes automatizados são executados automaticamente como parte do pipeline de CI/CD e fornecem feedback instantâneo sobre a qualidade da aplicação.

4. **Implantação Contínua:** Após a compilação e os testes automatizados serem bem-sucedidos, a próxima etapa é a implantação da aplicação em um ambiente de teste. Isso permite que os desenvolvedores testem a aplicação em um ambiente controlado antes de implantá-la no ambiente de produção.
5. **Monitoramento Contínuo:** É importante monitorar constantemente a aplicação móvel para detectar quaisquer problemas que possam surgir. Isso pode ser feito usando ferramentas de monitoramento, como o Nagios ou o Zabbix.
6. **Implantação em Produção:** Após a validação final, a aplicação móvel pode ser implantada em produção usando um sistema de implantação automatizado. A implantação em produção é realizada após os testes de aceitação serem executados e a aplicação ter sido aprovada para uso.
7. **Atualizações Contínuas:** A implementação de uma estratégia de CI/CD permite que as atualizações sejam feitas continuamente, o que significa que os desenvolvedores podem implantar correções de bugs e novos recursos rapidamente, sem afetar a estabilidade da aplicação.

9. Diagrama de Componentes



Neste diagrama, o aplicativo móvel se comunica com a Api Gateway da AWS, que atua como um gateway para as funções do AWS Lambda. Essas funções do Lambda são usadas para processar dados, executar lógica de negócios e interagir com o banco de dados DynamoDB. O Amazon S3 é usado para armazenar e servir conteúdo estático, enquanto o CloudFront é usado para fornecer conteúdo em cache de forma eficiente. Todos esses componentes são executados na infraestrutura de nuvem altamente estável e segura da AWS.

10. Disaster Recovery

Disaster recovery é o processo de restaurar a infraestrutura de tecnologia da informação (TI) e os sistemas críticos de uma organização após um desastre natural ou humano, falha de hardware, erro humano ou ataque cibernético. O objetivo do disaster recovery é minimizar o tempo de inatividade e recuperar os dados perdidos para garantir a continuidade das operações da organização.

1. **Desastre físico**

Identificação: A perda de infraestrutura física como resultado de um desastre natural, como um terremoto ou incêndio.
Recuperação: A organização deve ter backups off-site de todos os dados e sistemas críticos para garantir a continuidade das operações. Além disso, é importante ter um plano de recuperação de desastres que envolvem a reconstrução da infraestrutura física necessária.

2. **Falha de hardware**

Identificação: A falha de hardware pode ser identificada através de monitoramento contínuo dos sistemas, incluindo testes de integridade e diagnósticos regulares.
Recuperação: A organização deve ter um plano de contingência para a substituição rápida do hardware danificado, bem como backups de todos os dados e sistemas críticos.

3. **Ataque cibernético**

Identificação: A detecção de um ataque cibernético pode ser realizada por meio de monitoramento de segurança contínuo, incluindo análise de tráfego e detecção de anomalias.
Recuperação: A organização deve ter um plano de resposta a incidentes para lidar com o ataque e minimizar os danos. Isso pode incluir a restauração de backups, a aplicação de patches de segurança e a implementação de medidas de segurança adicionais.

4. **Erro humano**

Identificação: A maioria dos erros humanos pode ser detectada através de monitoramento contínuo dos sistemas, incluindo auditorias regulares.
Recuperação: A organização deve ter um plano de contingência para lidar com erros humanos, incluindo backups de todos os dados e sistemas críticos, bem como treinamento e conscientização dos funcionários.

5. **Interrupção de serviços**

Identificação: A identificação de interrupções de serviços pode ser realizada através de monitoramento contínuo dos sistemas, incluindo testes de integridade e diagnósticos regulares.
Recuperação: A organização deve ter um plano de recuperação de desastres que envolva a restauração dos serviços interrompidos. Isso pode incluir a aplicação de patches de segurança, a restauração de backups e a implementação de medidas de segurança adicionais.

11. Pipeline com Jenkins

1. A primeira linha define que a pipeline pode ser executada em qualquer agente disponível (definido por "any").
2. A seção "stages" contém uma lista de etapas que a pipeline executará. Há duas etapas: "Clonar repositório" e "Executar operação".
3. Cada etapa é composta por um "step" que define uma ação específica que deve ser executada. O primeiro step "git" clona o repositório do GitHub especificado na URL. O segundo step "sh" executa um comando de shell para executar a operação desejada.

```
pipeline {
    agent any
    stages {
        stage('Clonar repositório') {
            steps {
                git 'https://github.com/jcafff/grupo05.git'
            }
        }
        stage('Construir o projeto') {
            steps {
                sh 'npm install'
                sh 'npm run build'
            }
        }
    }
}
```