Universidade Estadual de Campinas - UNICAMP

Faculdade de Tecnologia - FT

SI304/A - Engenharia de Software II - Atividade A12

Gestão de Configuração e Versão

GRUPO C

LÍDER:

Gabriel Felipe Kugel 234782

GRUPO:

Bruno Alexander Klimowitsch Lins 167460,

Gabriel Santos Bueno 182500,

Guilherme Henrique Trevisan Conceicao 170959,

Jean Marcos De Andrade 175710,

Lucca Gonçalves Ferreira Santos 220996,

Marx Maciel Xavier 184878,

Victor Anthony Mozer Cazotti 194637,

Limeira - São Paulo

2021

Sumário

Introdução

Nesta atividade utilizamos o GitHub para realizar todas as tarefas solicitadas, alguns membros do grupo utilizam a ferramenta no cotidiano e com isso ficou mais fácil a realização do exercício.

Escolha do Workflow

A partir das opções de workflows dadas em aula, o grupo decidiu utilizar o Gitflow. A justificativa para tal escolha se deu pela popularidade e familiaridade entre os integrantes do grupo com o workflow, além disso podemos citar algumas outras características que influenciaram a escolha:

- Praticidade;
- Facilidade;
- Acessibilidade;

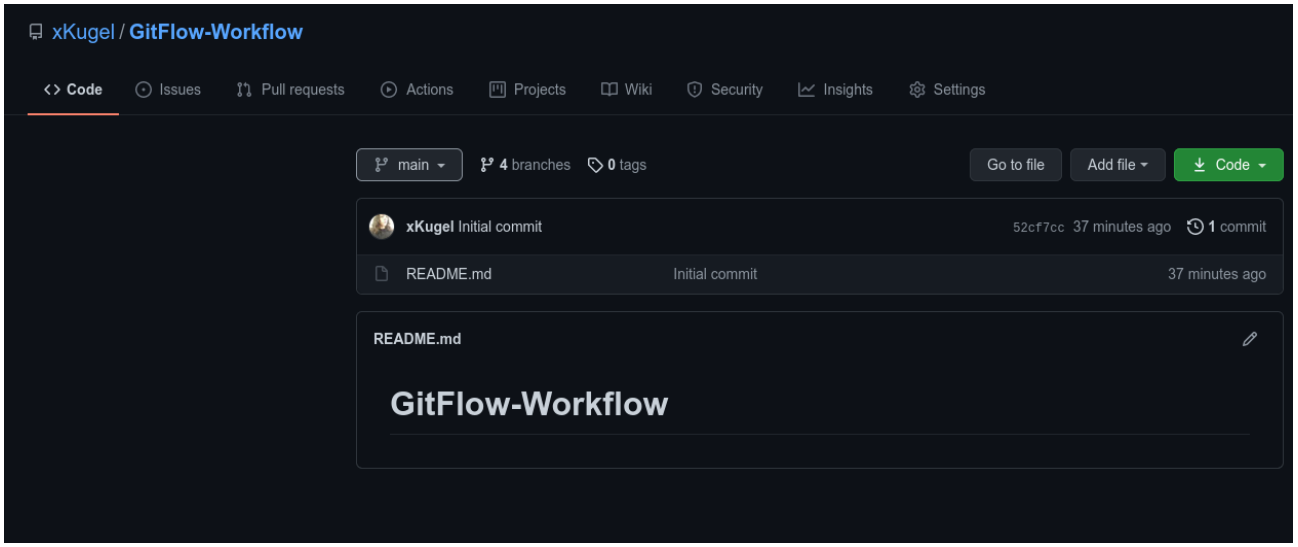
Aplicação do Workflow proposto

Após a escolha do nosso workflow, foi decidido que utilizariamos o GitHub como nossa plataforma de hospedagem de código-fonte. Dessa forma, elaboramos um pequeno fluxo de ações conforme o padrão Workflow.

No Exemplo abaixo fica evidente o uso do Workflow escolhido para exemplificar a criação de um projeto no GitHub, e o desenvolvimento do processo de versionamento para uma pequena alteração (feature):

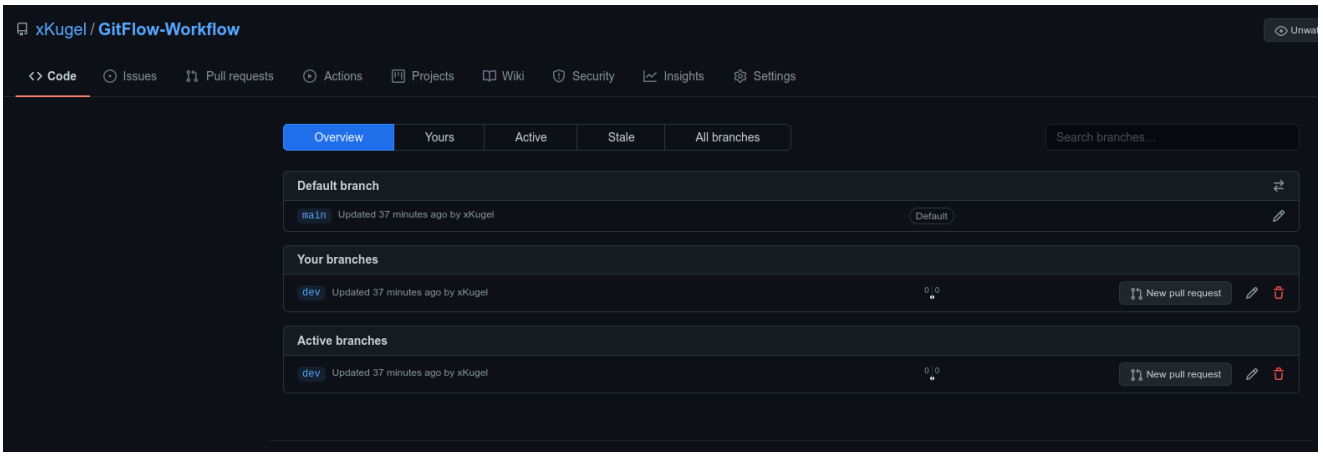
- **Criando o Projeto**

Criado o projeto, foram dadas as devidas permissões para os colaboradores conseguirem acessar, e editar o projeto público!



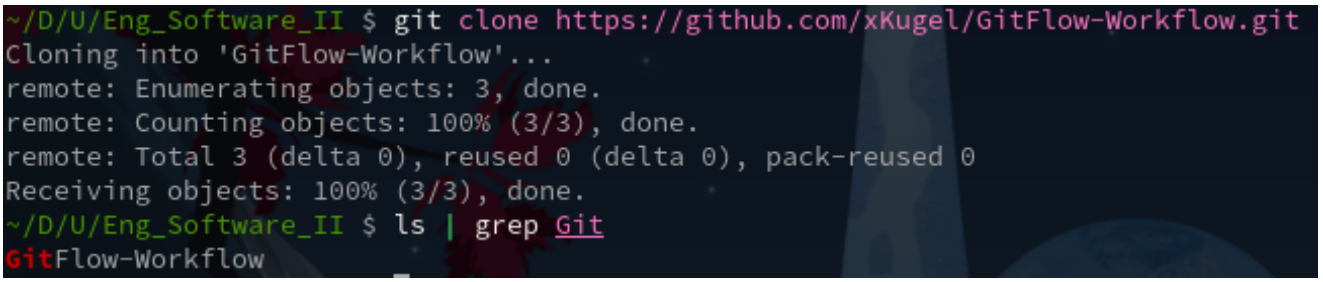
- **Ramo de Desenvolvimento**

Depois foi criado o ramo de Desenvolvimento, como sugere o Gitflow, que deve ser uma ramo sempre paralelo ao main/master, onde as features serão primeiramente margeadas e testadas.



- **Clone do repositório remoto para a máquina**

O comando é responsável pelo download do repositório remoto, para o ambiente local do desenvolvedor (git clone <link do repositório>).

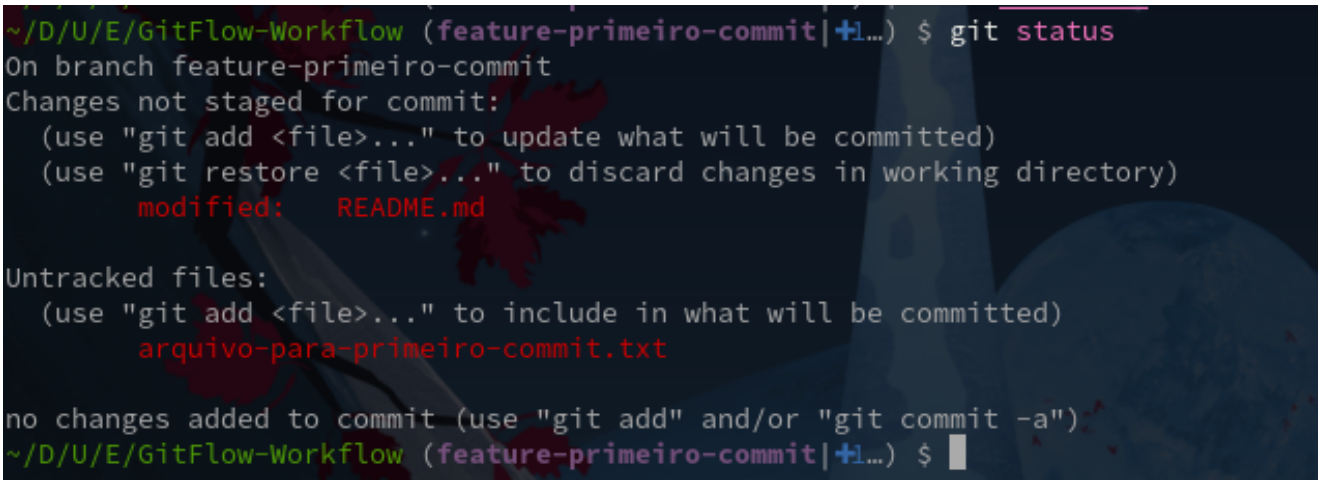


- **Criando um nova branch**

Os comandos abaixo fazem atualização do repositório local, de acordo com o repositório remoto (comando “git pull”), e em seguida cria um ramo novo chamado “feature-primeiro-commit” (comando “git checkout -b feature-primeiro-commit”).

- **Modificações**

São realizadas alterações no projeto, em suma uma edição do arquivo README.MD e a criação de um arquivo em branco para observar a comportação do comando “git status”.



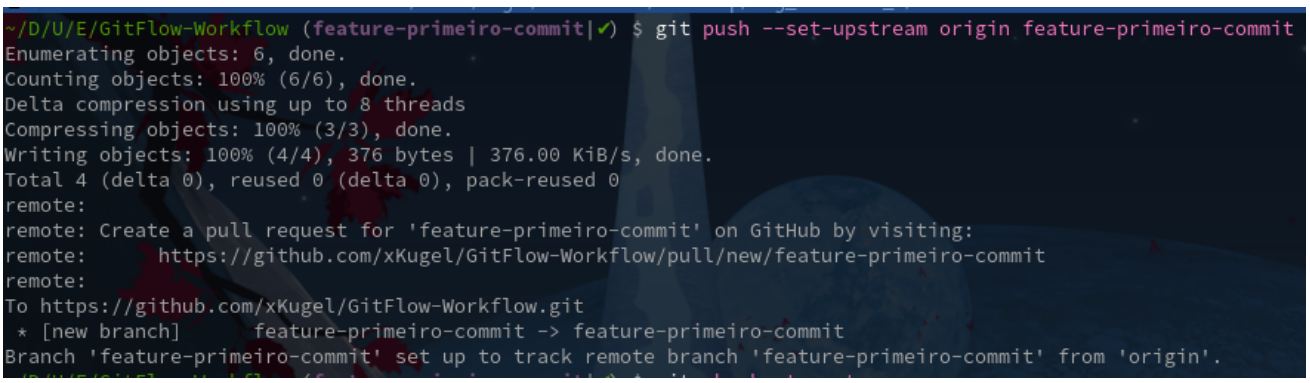
- **Fazendo commit das alterações na branch local**

Uma vez que terminada as alterações no repositório local, é hora de mandá-las para o repositório remoto, os comandos abaixo exemplificam isso, adicionando as alterações ao pipe de trabalho com “git add <arquivo específico>" ou “git add .” para adicionar tudo.

Em seguida é feito o commit, no exemplo abaixo, apenas com uma mensagem simples.

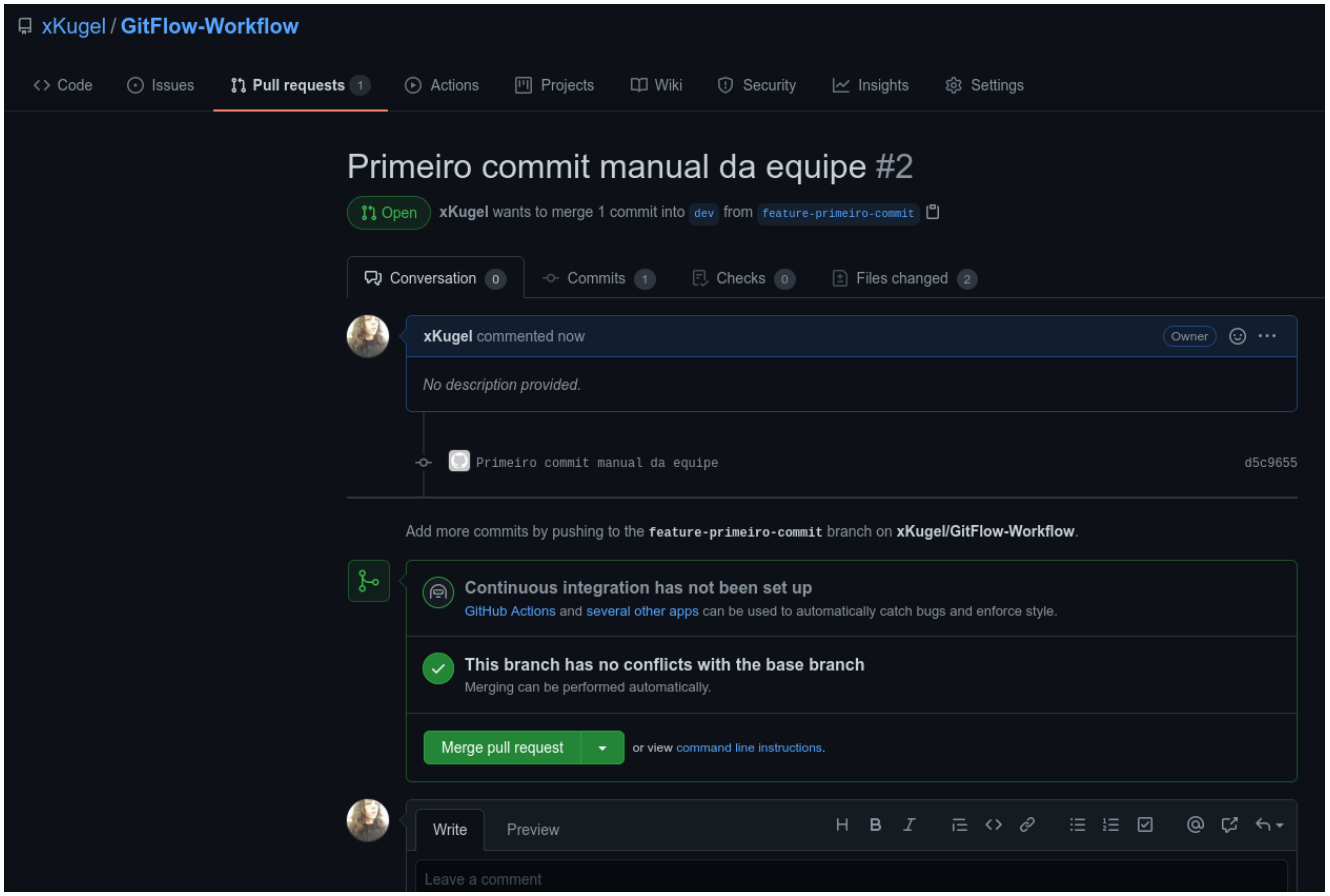
- **Enviando as modificações para a branch remota**

Depois de criado o commit, a última coisa que falta para disponibilizar isso no repositório remoto, é atualizar o repositório remoto com o local, e para isso utilizamos o “git push”, os demais parâmetros utilizados são para definir a criação de um ramo remoto novo, que até então só existia remotamente.



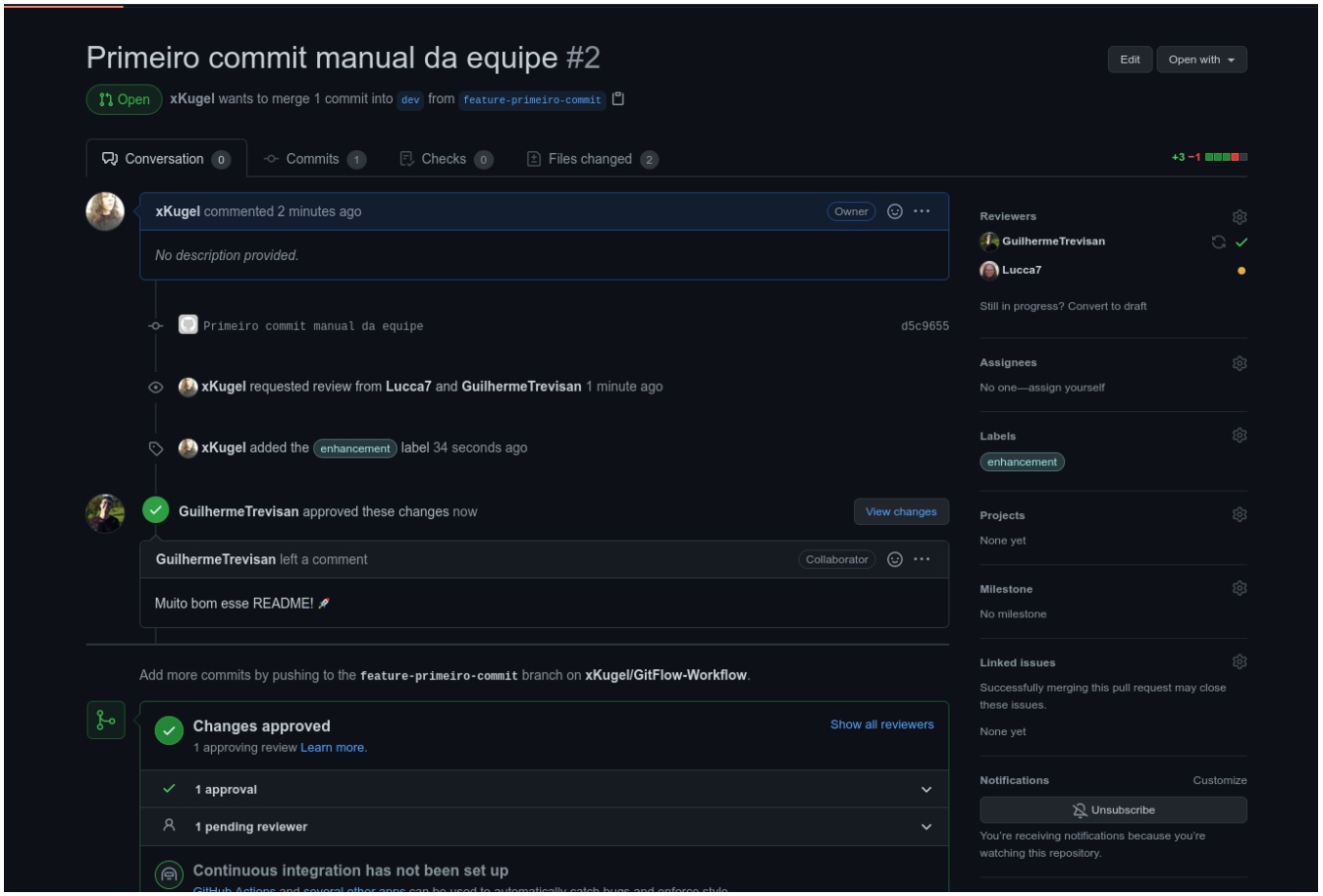
- **Abrindo Pull Request**

Para garantir que as alterações feitas não são um problema, as alterações não vão parar no ramo de desenvolvimento diretamente, é feita uma bateria de revisão por outros colaboradores em cima das alterações realizadas, o que sedia essas revisões são os “pull requests/merge requests”, que são os pedidos de aprovação/margeamento entre os ramos.



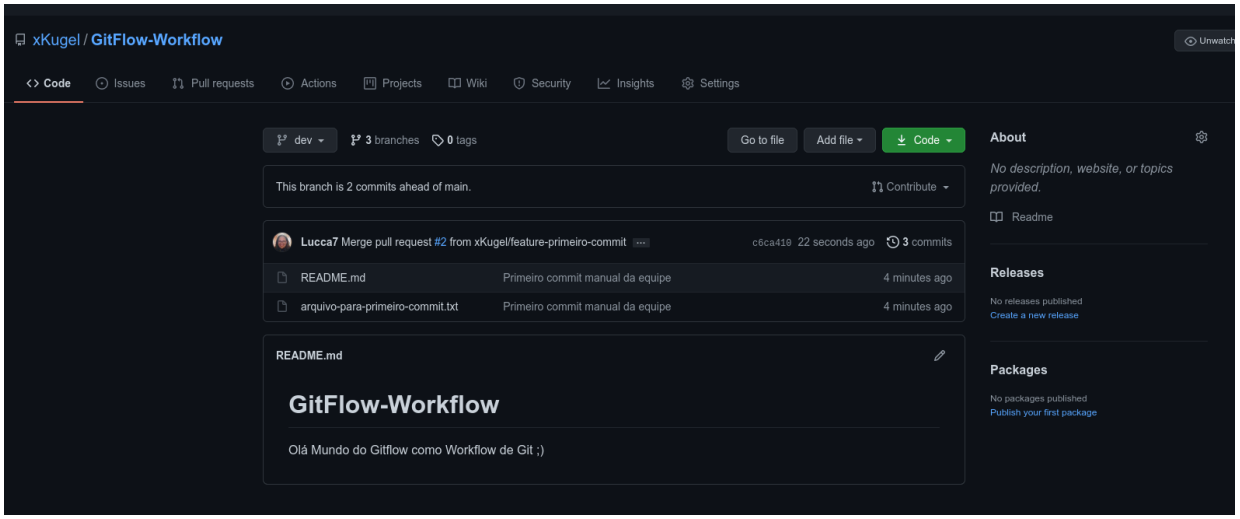
- **Discussões de Pull Request**

Nos pull requests são comuns discussões e elogios entre programadores, para garantir a excelência do código desenvolvido, e uma vez sanadas todas as discussões, é hora de mergear.



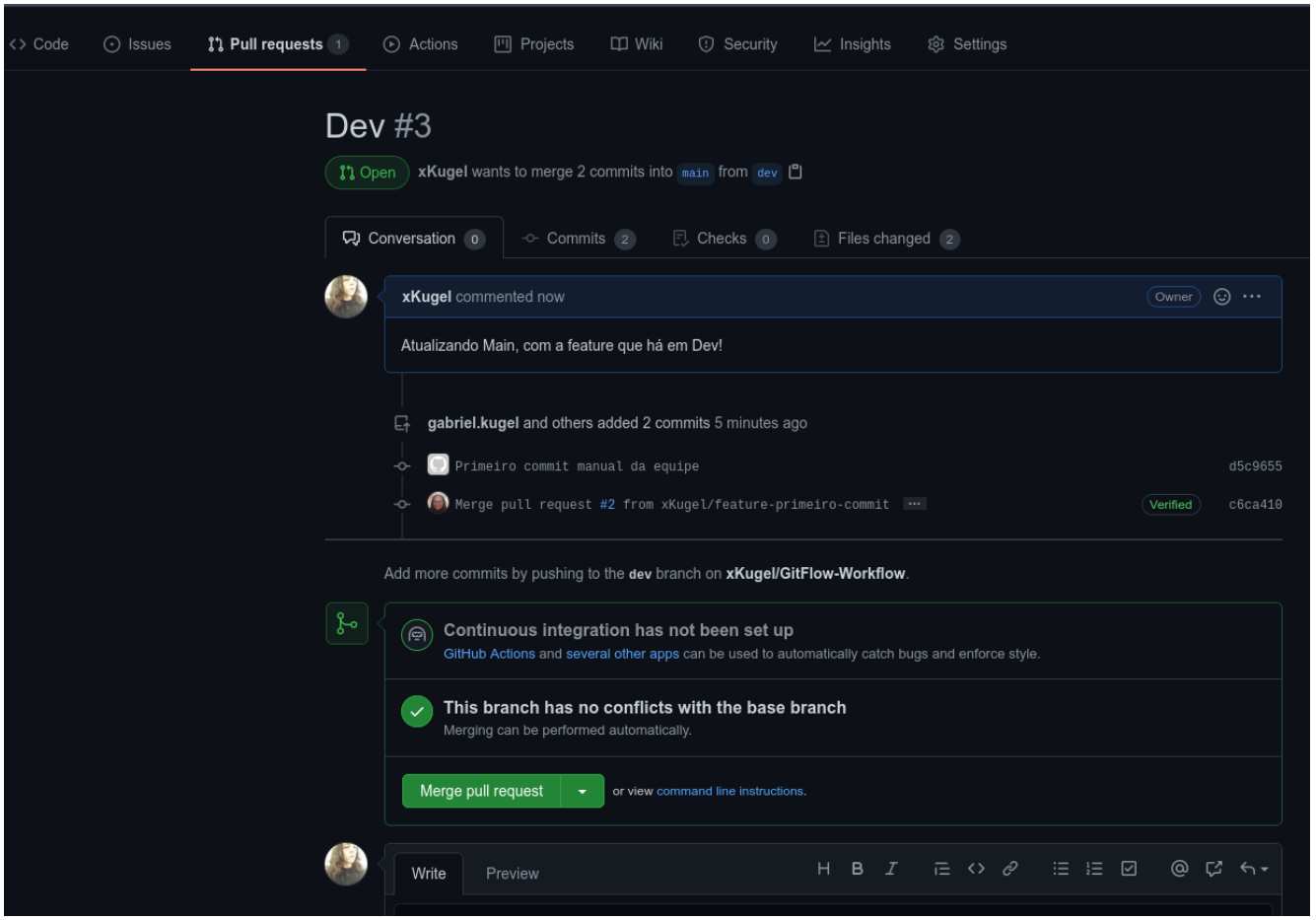
- **Ramo de Desenvolvimento Atualizado**

Uma vez que o ramo de desenvolvimento é atualizado, agora todos os programadores podem trabalhar com base no código desenvolvido anteriormente, testar e utilizá-lo em demais atualizações.



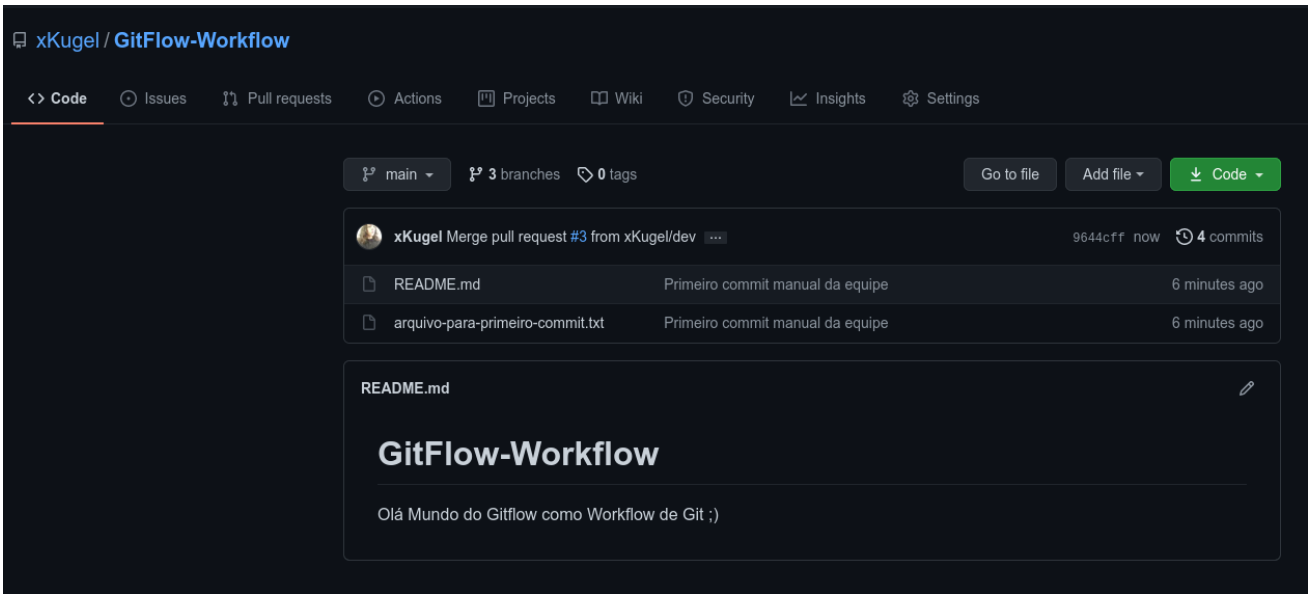
- **Release**

Quando um projeto se terminar, e todas suas funcionalidades foram aplicadas e testadas no ramo de Desenvolvimento, é hora de jogar isso para o ramo principal, para o cliente poder usar, então é criado o pull request de Desenvolvimento para “main/master”, também conhecido como Release



- **Alteração concluída e disponível**

Quando o ramo de desenvolvimento é mergeado no ramo principal, todo código desenvolvido previamente está disponível para uso na versão principal do programa, e o Gitflow para uma inovação (feature) está concluído!


xKugel / GitFlow-Workflow
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
3 branches
0 tags
Go to file
Add file
Code
xKugel Merge pull request #3 from xKugel/dev
9644cff now
4 commits
README.md
Primeiro commit manual da equipe
6 minutes ago
arquivo-para-primeiro-commit.txt
Primeiro commit manual da equipe
6 minutes ago
README.md
GitFlow-Workflow
Olá Mundo do Gitflow como Workflow de Git ;)

Avaliação do Desempenho do Grupo

No geral, os integrantes da equipe deram suas opiniões e participaram ativamente no desenvolvimento da atividade, através de uma reunião via Google Meet e por mensagens via WhatsApp, com exceção do Jean Andrade (RA: 175710) que não participou e avisou previamente no grupo do Whatsapp.