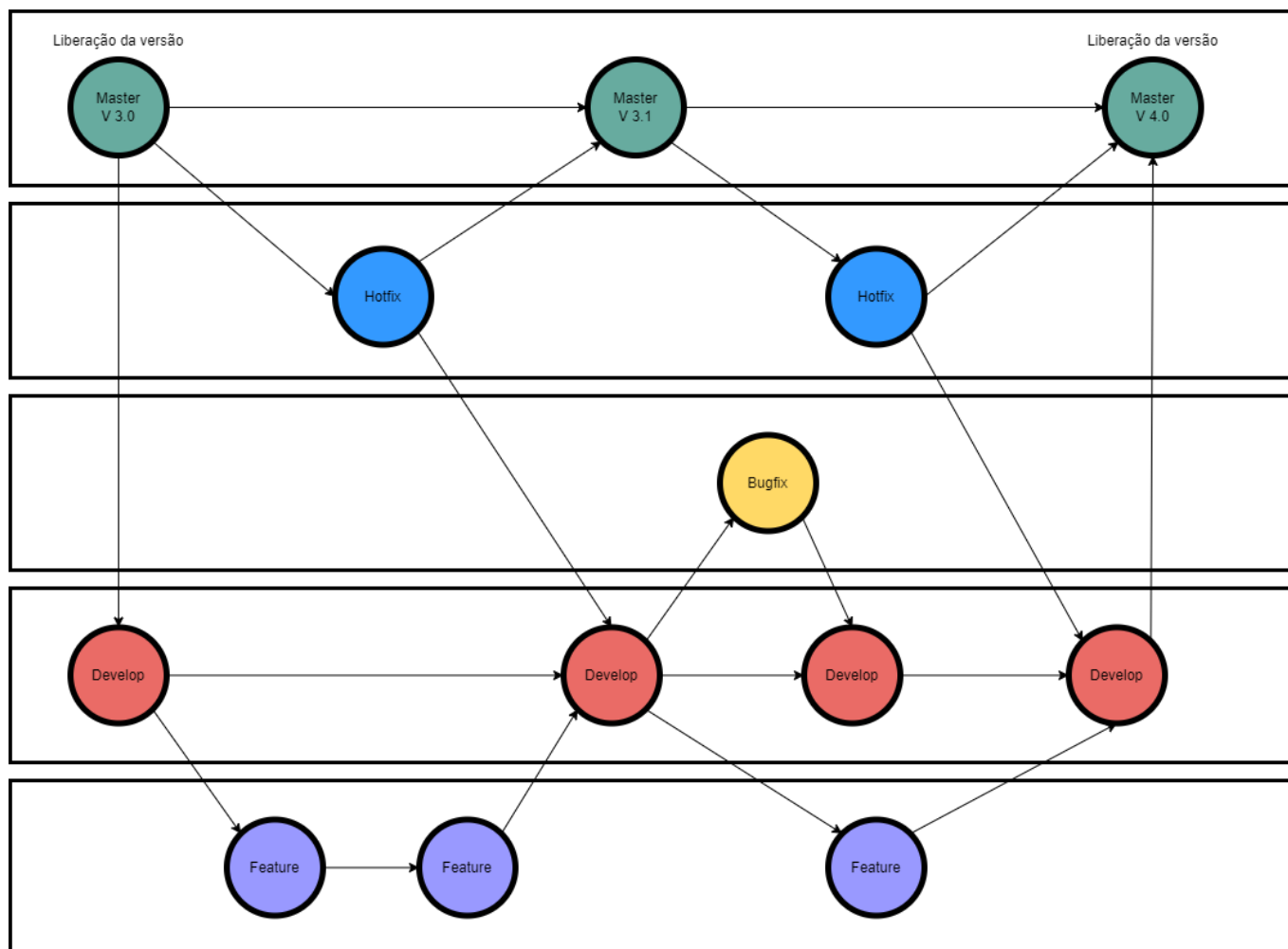


Gitflow

Fluxo



Master

Nesta branch está o código do produto que está em produção, liberado para os clientes, o qual ocorre uma atualização e liberação de nova versão a cada intervalo de tempo (quize dias, por exemplo). A master realiza merges os códigos testados da branch develop e pode receber alterações de hotfix.

Develop

Nesta branch está o código do produto em staging ou pré produção, onde ocorrem os testes antes que o produto seja liberado para utilização dos clientes, podendo ser realizadas alterações diariamente, geralmente por deploy. Recebe modificações vindas de features e bugfix.

Hotfix

Esta branch criada a partir da master é utilizada quando existe a necessidade de realizar uma correção diretamente dentro do ambiente de produção (master). Geralmente para aplicar correções críticas, que necessitam ser colocadas rapidamente em produção. Após ser fechada, a hotfix é finalizada e passa a fazer parte tanto do ambiente de staging quanto de produção (merge com develop e master)

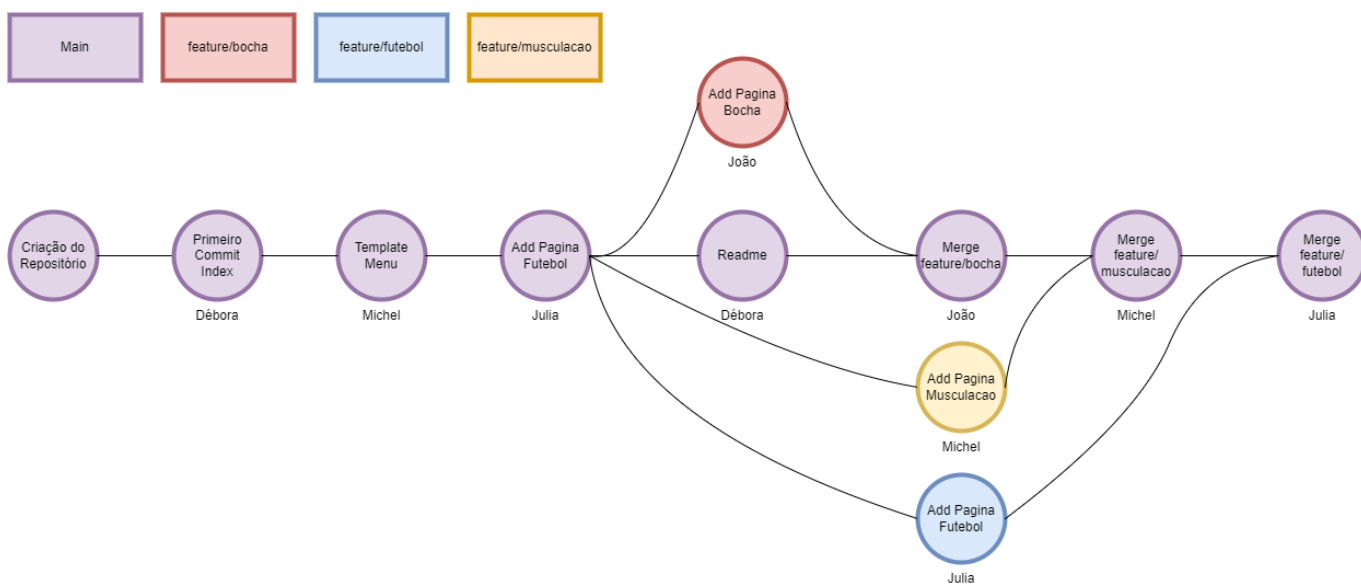
Bugfix

Esta branch deve ser criada quando um erro é detectado durante testes com a equipe de QA, realizados no ambiente de staging. Se aplica a features novas que ainda não fazem parte do ambiente de produção e portanto se iniciam e encerram a partir da branch develop.

Feature

Esta branch é criada ao iniciar uma nova funcionalidade no projeto, que receberá novos commits com o código específico para esta determinada funcionalidade, como a criação de uma nova tela de cadastro, por exemplo. Quando o desenvolvimento da funcionalidade for concluído, a branch é integrada ao código do ambiente de staging, ou seja, a develop.

Fluxo do tabalho realizado:



Orientações para a utilização do github como versionamento de software.

Primeiramente, se faz necessária a criação repositório no github.

Após isso, no computador, entrar no diretório em que deseja clonar o repositório.

O criador do repositório deve adicionar outros colaboradores ao projeto em Settings -> Collaborators.

Colaboradores devem aceitar se juntar ao projeto em Organizations -> Join -> Accept invite.

DA UTILIZAÇÃO DO GIT

Configurar usuario no seu git

```
git config --global user.email "email@email.com"
```

```
git config --global user.name "nome"
```

Clonar repositório remoto existente para o local de trabalho

```
git clone https://github.com/drsavi/gitflow.git
```

Exibir as condições (status) do diretório de trabalho e da área de staging

```
git status
```

Adicionar alterações para o próximo commit

```
git add {nome do arquivo} {ou . pra incluir tudo}
```

Adicionar alterações adicionadas na area de staging para o repositório

```
git commit -m "mensagem"
```

Enviar o conteúdo do repositório local para um repositório remoto

```
git push
```

ou quando for o primeiro commit do repositório:

```
git push -u origin HEAD
```

Exibir todas as branches

```
git branch -a
```

Criar nova branc

```
git checkout -b {nome da nova branch}
```

Mudar de branch

```
git checkout {nome da branch}
```

Apagar branch local

```
git branch -D {nome da branch}
```

Apagar branch remota

```
git push origin --delete {nome da branch}
```

Arquivar alterações não commitadas do seu local de trabalho

```
git stash
```

Aplicar as mudanças de um stash à área de trabalho

```
git stash pop
```

Unir duas branches (se faz necessário realizar checkout para a branch que deseja aplicar as modificações)

```
git merge {nome da outra branch}
```

Resolução de conflitos utilizando Git e VsCode

Para configurar o VS Code como editor padrão, basta rodar o comando no terminal:

```
git config --global core.editor "code --wait"
```

E quando precisarmos do editor, quem vai aparecer é o VS Code. Como nos casos de rebase, merge, commit, add -p, etc.

Para garantir que realmente houveram alterações, podemos rodar o comando que abre as configurações globais do Git no editor de textos e o VS Code irá abrir automaticamente:

```
git config --global -e
```

Agora, para adicionar o VS Code como nossa ferramenta padrão de merge, vamos rodar o comando `git config --global -e` e adicionar as linhas:

```
[merge]
  tool = vscode
[mergetool "vscode"]
  cmd = code --wait $MERGED
[diff]
  tool = vscode
[difftool "vscode"]
  cmd = code --wait --diff
```

Agora basta rodar o comando `git mergetool` em um merge com conflitos.

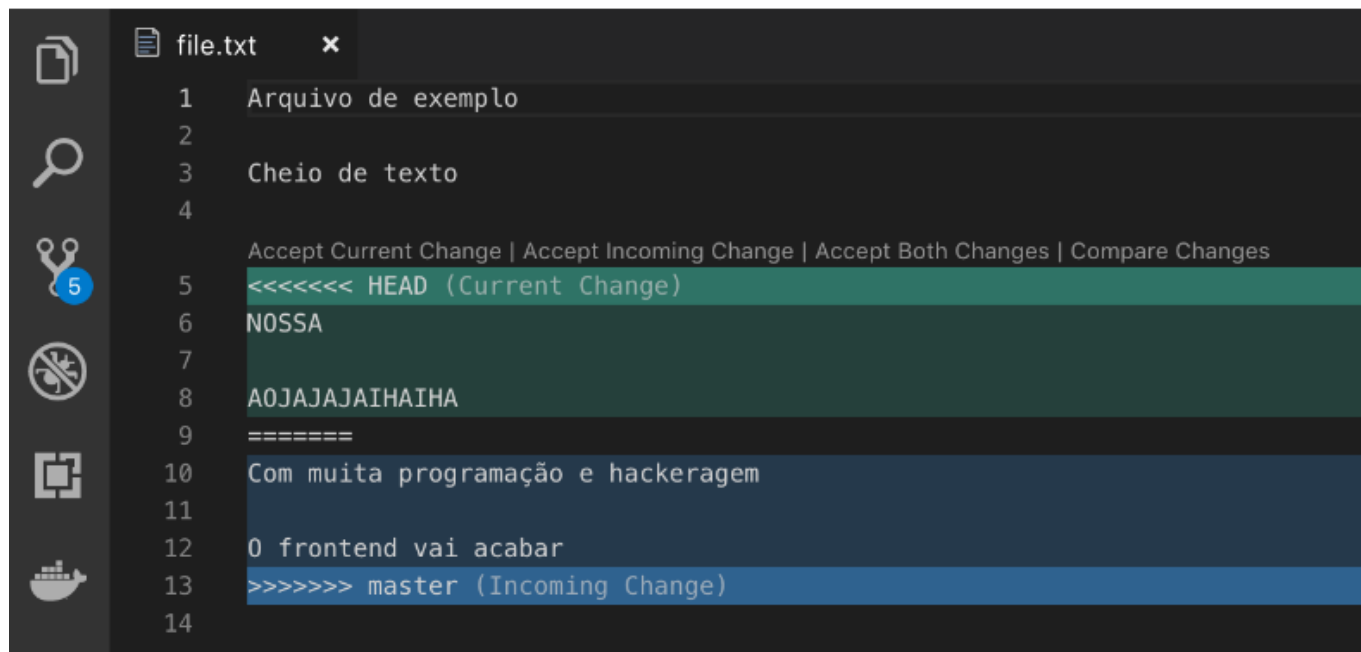
Exemplo:

Digamos que eu acabei de rodar o `git merge master` em uma branch que estava desatualizada.

Ao rodar o `git status` vemos que aconteceu um conflito.

Podemos então rodar o `git mergetool` e o Visual Studio Code irá abrir com a interface para correção dos conflitos.

Aparecerá uma tela como essa:



```
1 Arquivo de exemplo
2
3 Cheio de texto
4
5 <<<<<<< HEAD (Current Change)
6 NOSSA
7
8 A0JAJAJAIHAIHA
9 =====
10 Com muita programação e hackeragem
11
12 O frontend vai acabar
13 >>>>>> master (Incoming Change)
14
```

Onde o que está em verde é o que temos em nossa branch e o que está em azul são as alterações que devemos escolher se aceitamos ou não.

Logo acima das alterações temos os botões para aceitar ou recusar uma alteração:

Botões	Resultado
Accept Current Change	Aceitar a mudança que temos localmente/em nossa branch atual
Accept Incoming Change	Aceitar a mudança que estamos recebendo de outra branch/remoto
Accept Both Changes	Aceitar ambas as alterações
Compare Changes	Comparar as alterações