## 1. Objetivo

Ser uma instrução de trabalho para padronizar o versionamento de código com a utilização da ferramenta GIT e AzureDevops.

## 2. Pré-requisitos para Execução das Atividades

Projeto e Repositório criados na Azure DevOps.

Conhecimento de GIT. Esse conhecimento pode ser adquirido acessando o nosso GUIA.

## 3. Referências

Review and merge code with pull requests - Azure Repos | Microsoft Docs

Azure Repos Git Documentation | Microsoft Docs

Fork Azure DevOps

## 4. Definições

### 1. Versionamento 4 dígitos

O versionamento do código fonte será feito com quatro dígitos para melhor controlar quais funcionalidades estão presente naquela versão do projeto.

a) 0 - Mudanças negociais
b) 1 - Novos módulos
c) 2 - Novas Funcionalidades
d) 3 - Correções

### a. Mudança negocial

Mudança de versão comercial definida principalmente por quem está à frente do projeto, como Gerentes, Product Owner, etc.

### b. Novos módulos

Conjunto de novas funcionalidades que englobam regas de negócios ainda não desenvolvidas no sistema.

### c. Novas Funcionalidades

Criação ou alteração estrutural de funcionalidades dentro de um modulo, ou seja, nova ou alteração da regra de negócio de um modulo previamente desenvolvido.

### d. Correções

Correções de bugs e mudanças não estruturais, como botões e posições de elementos na tela.

## 2. HOTFIX

São branchs de correções pontuais como bugs e pequenos ajustes (mudança de uma label, um posicionamento do botão, uma margem, etc) que não alteram as estruturas da aplicação.

## 3. FEATURE

São branchs de desenvolvimento de regras de negócio dentro da aplicação.

## 4. MAIN

É a branch principal do projeto e nela que fica o código mais testado e confiável do projeto, é nesta branch que se encontra o versionamento tagueado do projeto.

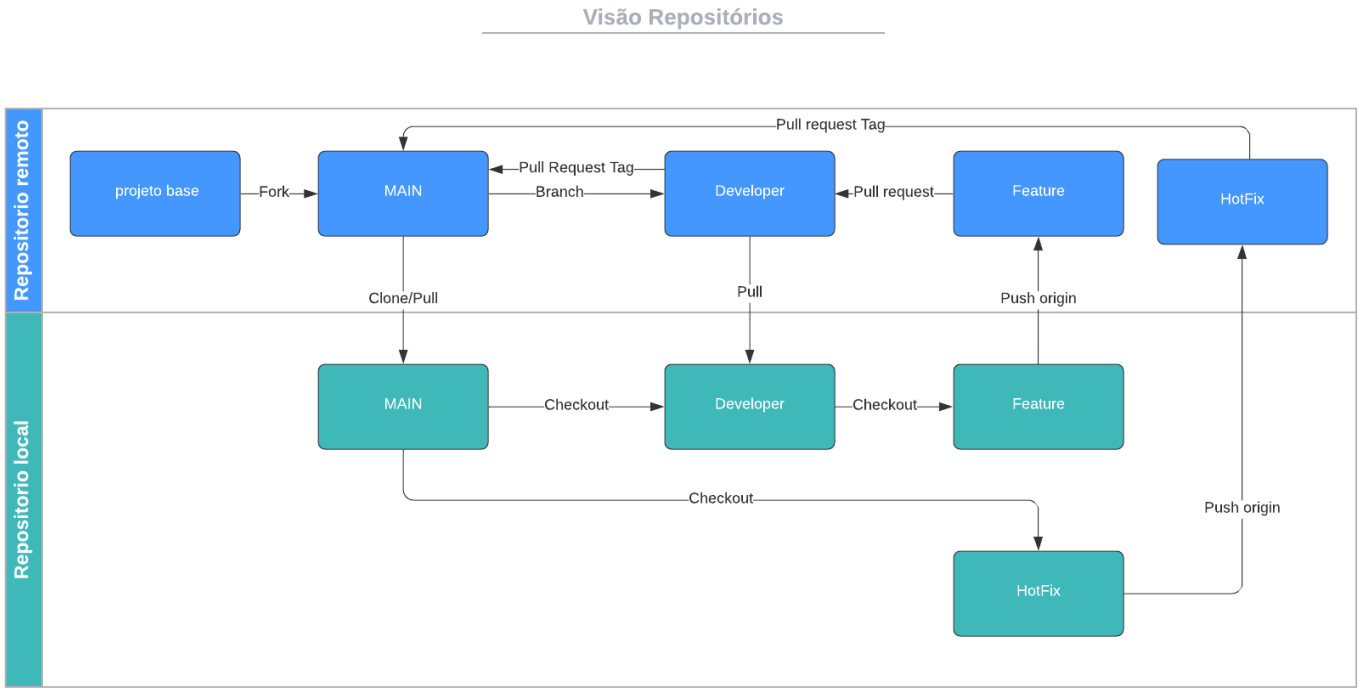
## 5. DEV

É a branch onde é feita o merge das features, também sendo a mais atualizada do projeto.

## 6. Pull request

É a solicitação para o merge para as branchs principais do projeto, geralmente estas estão bloqueadas para alteração direta.

**Figura 1** - Processo de versionamento de código

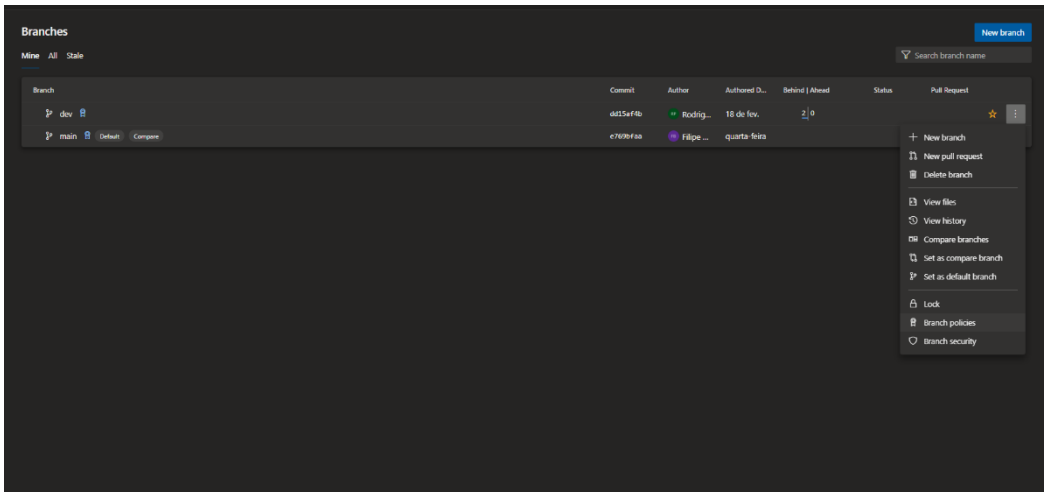


## 5. Descrição de Atividades

### 1. Iniciando o projeto

- Realizar um fork do projeto base. Caso tenha dúvidas, consultar a sessão de referências.
- Após o fork do projeto base ir em Repos > Branches > Branches polices
- Ativar a opção - Branch Policies Require a minimum number of reviewers – colocar um e marcar a opção when new changes are pushed Reset all code reviewer votes
- (Opcional) Ir em Automatically included reviewers e adicionar pelo menos um membro do projeto
- Criar a Branch Dev com base na Main e remarcar as mesmas opções em Branches polices

**Figura 2** – Tela Branches




### 2. Trabalhando com Feature

- Antes de criar a **feature**, deve ser feita a atualização da **dev local** de acordo com a **origin**.
- Criar a **branch feature** a partir da **dev local** atualizada.
- O nome desta **branch** deve ser **feature**/<nome-da-feature>

1. Fazer **clone** do projeto remoto para a máquina local
2. Criar uma **branch** a partir da **dev**
3. Todo dia durante o desenvolvimento fazer pelo menos um **pull** da **dev** na **feature** para manter o código atualizado
4. Após finalizar a **feature** fazer um **merge** da **dev** na **feature**
5. Enviar a **feature** para o repositório remoto
6. Solicitar um **pull request** para a **dev**

### 3. Solicitando um Pull Requests

- **Pull request** é a ação de mesclar o código desenvolvido com as **branchs** principais

- Após subir a **branch feature** ou **hotfix** o desenvolvedor deve solicitar um **pull request**
- Os **pulls requests** das **features** nunca devem ocorrer na **main**, qualquer **pull request** para **main** deve ser rejeitado pelo avaliador
- Todos os **pull request** dos **hotfixs** devem na **main**
- Ao solicitar um Pull Request é de **suma importância** que tenha um título e uma descrição coerente com o que foi desenvolvido e está sendo solicitado para o merge.
    1. Entrar no Repositório do projeto ir em **pull request**
    2. Apertar o botão new **pull request**
    3. Escolher a **branch source** e a **branch de destino** (lembrando da regra de **hotfix** e **feature**)
    4. Escolher pelo menos um revisor do código que também será responsável pelo merge
    5. Ao Abrir o **pull request**, escrever o título que corresponda ao que foi desenvolvido
    6. Na descrição dar preferência para o **commit** da **branch** de **origem**, isso não impede que a descrição seja aprimorada pelo desenvolvedor
    7. Se houver **work item** selecioná-lo

## 4. Trabalhando com Hotfix

- **hotfix** sempre são criadas a partir da **main**
- Toda branch hotfix será excluída após seu término, isso é feito no momento que o aprovador realizar o merge, na aprovação a própria Azure DevOps disponibilizará um checkbox.
    1. Fazer **clone** do projeto remoto para a máquina local
    2. Criar uma **branch** a partir da **main**
    3. O nome desta **branch** deve ser **hotfix**/<nome-da-hotfix>
    4. Após a correção fazer uma atualização da **main (Pull)**
    5. Efetuar o **merge** na **branch main** com a **hotfix**
    6. Resolver os possíveis conflitos
    7. Fazer um **push origin** hotfix/<nome-da-hotfix> para o servidor remoto
    8. Solicitar um **pull request**
    9. Após aprovado, gerar uma **tag**
    10. Efetuar o **merge** na **branch origin** de **dev** com a atualização da **branch main**.

## 5. Aprovando um Pull Request

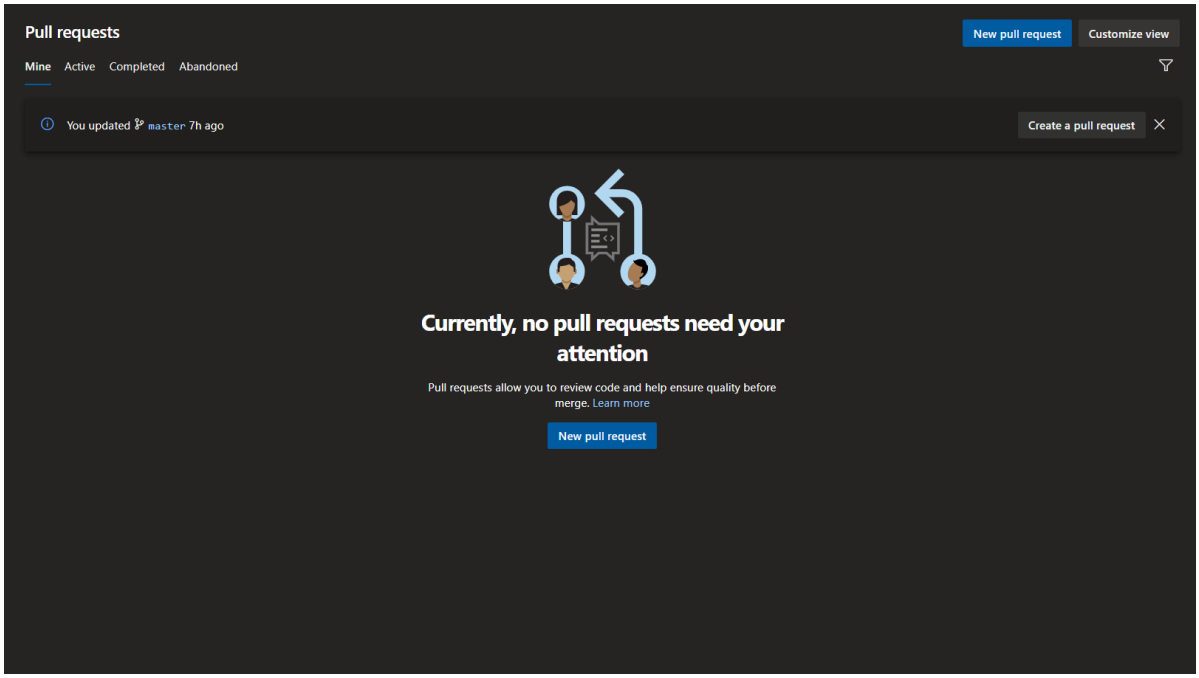
- O Aprovador do pull request não pode ser a mesma pessoa que solicitou
- Os aprovadores devem ser desenvolvedores do projeto
    1. Ao receber uma solicitação de análise com conflito recusar imediatamente
    2. Verificar a concordância do código, padrões de projetos e boas práticas
    3. O aprovador em caso de aprovação deve finalizar o merge.

## 6. Executando o Pull request da Dev para Main

- No final da sprint ou rodada de entrega o projeto que está na dev deve ser efetuado o merge para a **main**
- Somente neste caso será possível que a mesma pessoa que abriu o **pull request** aprove (mas é sempre recomendado que outra pessoa avalie o **pull request**)

- Ao fazer o **merge** a pessoa deve gerar uma **tag** conforme a entrega da sprint / versão
- **Pull request** da **dev** para a **main** deve ser feito por quem estiver mais inteirado do projeto

Figura 3 – Tela Pull requests




## 6. Utilização de Dados Pessoais (LGPD)

Não contempla a utilização de Dados Pessoais, conforme a Lei Geral de Proteção de Dados Pessoais (LGPD).

A utilização de dados pessoais é utilizada ao logar na ferramenta e na solicitação de aprovação do merge do código fonte.

| Dados de Entrada | Fonte de Dados | Tipo de Tratamento do Dado | Dados de Saída |
|---|---|---|---|
| Nome | Active Directory | Comunicação | Azure DevOps |
| E-mail | Active Directory | Comunicação | Azure DevOps |

O tratamento de dados é realizado exclusivamente para identificação dos envolvidos e envio de notificação para aprovação da solicitação de merge do código fonte.
Toda e qualquer informação a respeito de clientes e fornecedores (pessoa natural) somente serão repassadas mediante aprovação expressa destes ou por ordem judicial.

## 7. Anexos

N/A

## 8. Controle de Versão

Histórico (para uso da Gerência de Governança de TI):

| Versão | Data Publicação | Elaborador | Motivo da Versão |
|---|---|---|---|
| 1.0 | 19/03/2021 | Mikael do Nascimento Medeiros | Padronizar e dar conhecimento sobre o uso do versionamento de código com a utilização da ferramenta GIT e Azure DevOps na STI. |

Aprovação da Versão Atual:

| Nome | Cargo |
|---|---|
| | |