GitHub Workflow

Relatório de Fluxo de Trabalho de Eletrônica Para o GitHub

INTRODUÇÃO

Considerando as recorrentes confusões associadas ao controle de versão do time de eletrônica, optou-se pela criação de um relatório com o detalhamento de como deve ser o fluxo de trabalho do time quando editando os arquivos do seu repositório. Esse relatório se baseia no fluxo encontrado na documentação oficial do GitHub.

SUMÁRIO

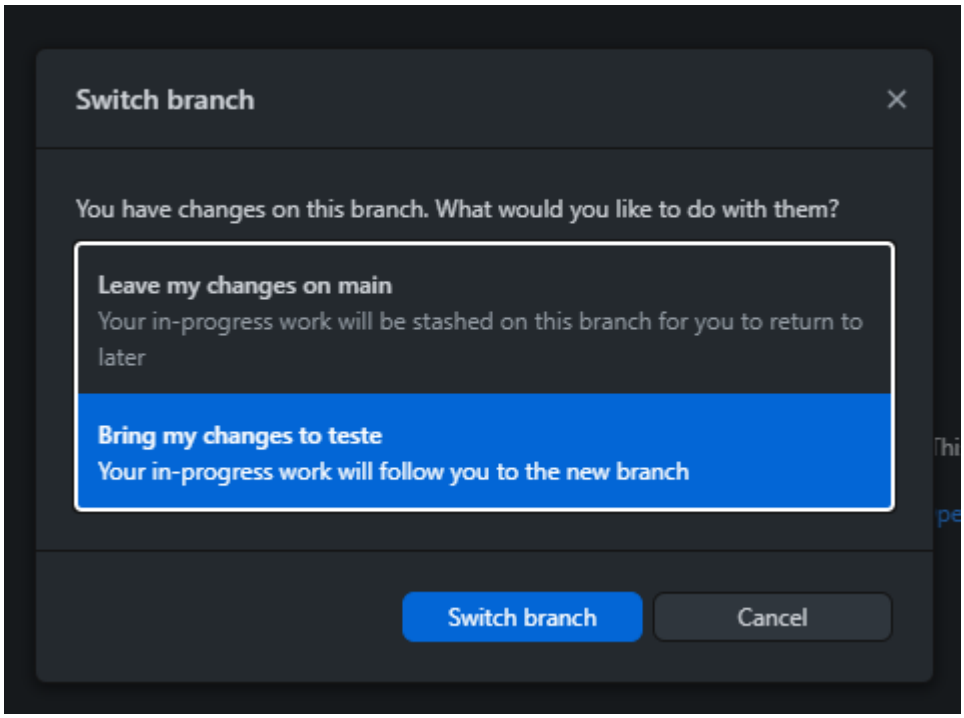
1. RESUMO

a. Sempre tenha o branch que quer trabalhar atualizado – considere a possibilidade do computador/notebook não estar ligado à internet. Antes de modificar o repositório, faça um *fetch* do branch para garantir que seu repo local esteja atualizado. Isso se aplica para o main branch, principalmente.
b. A partir do repositório atualizado, crie um branch. Nomeie-o de acordo com o que pretende fazer. É recomendado o uso de hífen, como por exemplo, “consertar-combustível”.
c. Para cada modificação completa e razoável, faça um commit. Por completa, entenda tanto as pequenas modificações que compõem uma modificação maior, quanto a própria modificação maior. Commits concisos e claros

facilitam na hora de encontrar erros, corrigir bugs e previnem perda de trabalho no caso de erros e falhas.

d. Faça um *merge* ou *pull request*. É **crucial** testar e conferir as modificações, para garantir que está tudo funcional. Caso queira dar *merge*, é importante que o usuário confira as modificações e teste adequadamente o código. O *pull request*, por outro lado, é um pedido para que outro contribuidor do repositório confira as modificações, as corrija/aprove e faça o merge. Recomenda-se o *pull request* no caso de dúvidas quanto às modificações ou de como realizar o merge – embora aprender a usar o merge seja encorajado.

OBS: Caso tenha iniciado modificações antes de criar um branch, crie um mesmo assim. Antes de fazer um commit no main branch, mesmo se testado, crie um branch e transfira as modificações feitas para esse branch. No tópico 3 será explicado melhor o porquê. Note que um pop-up similar ao da imagem abaixo irá aparecer. A opção certa é "*Bring my changes to [branch]*".



2. DETALHAMENTO DA METODOLOGIA

O fluxo de trabalho de eletrônica irá se basear na criação de branches para edição. É fluxo de trabalho usado pelo Github. Resumidamente, o colaborador que for modificar o repositório de eletrônica deve criar um Branch, modificar apenas esse Branch e solicitar um merge entre o criado e o principal. A solução dos conflitos do merge requer conhecimento técnico do assunto do que se trata o branch, além de atenção, mas pode ser realizada por qualquer colaborador com as devidas permissões, que podem ser concedidas pela conta do GitHub da MR.

2.1. CRIAR UM BRANCH

Crie um branch no seu repositório. Dê um nome de branch curto e descritivo, que deixe bem claro no que está trabalhando. O nome é importante para evitar que outros colaboradores realizem o mesmo trabalho simultaneamente. Um branch deve ser objetivo e tratar de um conjunto de modificações ou implementações. Por exemplo, correção-rpm ou implementação-telemetria. Caso não saiba criar branches, veja "Como criar e excluir branches no seu repositório".

Ao criar um branch, você cria um espaço para trabalhar sem afetar o branch principal. Além disso, você fornece aos colaboradores a oportunidade de revisar seu trabalho. Trataremos disso mais adiante.

2.2. FAZER ALTERAÇÕES

No seu branch, faça quaisquer alterações desejadas no repositório. Seu branch é um lugar seguro para fazer alterações. Se você cometer um erro, você poderá reverter suas alterações ou fazer push das alterações adicionais para corrigir o erro. As suas alterações não serão feitas no branch padrão até que você faça merge de seu branch, mesmo quando fizer pushes.

2.3. FAÇA COMMITS

Lembre-se de dar commit às suas alterações, com razoável frequência. Dê a cada commit uma mensagem descritiva para ajudar você e futuros contribuidores a entender quais alterações o commit contém. Lembre-se que commits são locais, ou seja, os colaboradores só podem visualizar as alterações do branch criado após o colaborador realizar um push em seu branch.

Idealmente, cada commit contém uma alteração isolada e completa. Isso facilita reverter as suas alterações se decidirmos adotar uma abordagem diferente. Por exemplo, se você deseja renomear uma variável, modificar uma função e adicionar um código de teste, faça cada alteração em commits completamente isolados. Posteriormente, se você quiser reverter apenas uma das modificações, ou múltiplas, você poderá reverter os commits específico com mais facilidade, economizando esforço e evitando retrabalho.

Ao realizar commits e pushes das suas alterações, você fará backups do seu trabalho nos armazenamentos locais e remotos. O aplicativo de Desktop do GitHub é suficiente para o nível de manipulação que usamos nesse fluxo de trabalho. Continue a criar, fazer commits e pushes das alterações no seu branch até que você esteja pronto para pedir feedback ou realizar um merge.

2.4 CRIAR UM PULL REQUEST (OPCIONAL)

Um pull request é utilizado para pedir a outros colaboradores feedback sobre suas alterações. Alguns fluxos de trabalho exigem que os branches sejam revisados e aprovados antes de serem mesclados. Se você busca feedback ou conselhos antes de concluir suas alterações, você poderá marcar seu pull request como um rascunho. Para obter mais informações, confira "Como criar uma solicitação de pull".

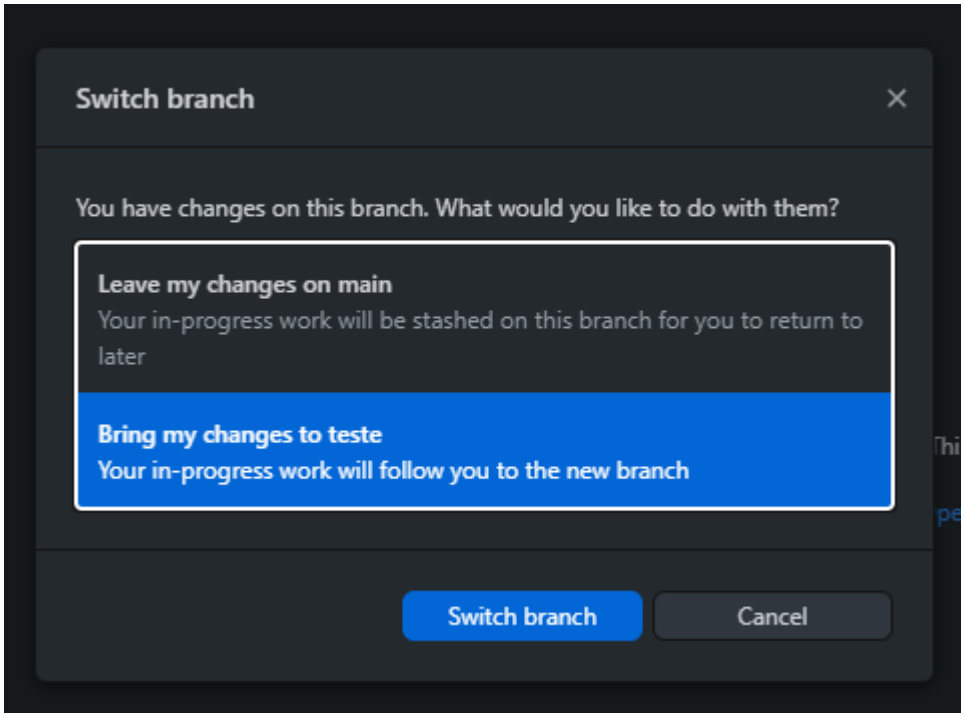
Em outra nota, é possível que você não consiga concluir a sua tarefa em um ciclo de trabalho. Nesse caso, o ideal não é mesclar o branch no final do ciclo, mas iniciar um pull request e informar neste que o trabalho está incompleto. Com

isto, outros colaboradores podem criar branches a partir do seu e continuar as implementações até criarem novos pull requests. Quando disponível, você pode revisar esse pull request e decidir se as modificações feitas pelo outro colaborador atendem ao objetivo inicial do branch e realizar a mesclagem das implementações.

3. ERROS E MODIFICAÇÕES INDEVIDAS NO MAIN BRANCH

É comum esquecer de criar um branch antes de iniciar as modificações, até porque ainda não é necessariamente parte da rotina do time de eletrônica. Nesses casos, – em que você modifica localmente o branch principal – crie um branch mesmo assim: quando lembrar, ou estiver prestes a dar um commit.

Transfira as modificações para esse novo branch. A opção correta é, normalmente, a segunda: “Bring my changes to branch”, onde “branch” será o nome do branch que estiver sendo criado no momento (teste, na imagem).



Isso é feito principalmente para permitir que vários colaboradores trabalhem simultaneamente no repositório. Caso alguém dê um commit no branch principal e

não der push, os outros colaboradores não conseguirão atualizar seus repositórios locais – porque commits são locais. Ainda que o colaborador inicial dê um push, os outros colaboradores terão que fazer um merge do seu trabalho com o push feito, tornando o fluxo confuso e propenso a erros. Não só isso, mas há a chance de inserir bugs ou modificações não intencionais no branch principal, que seriam mais fáceis de serem resolvidas no branch pessoal.

Idealmente, o branch principal deve ser modificado por meio de pull requests. Há vantagens e desvantagens em esperar que outro colaborador cheque ou não suas modificações, valendo sempre o bom senso. Os pull requests devem ser verificados por um dos responsáveis pelo repositório – qualquer colaborador do sub de eletrônica da MR – e resolvido. Com a devida aplicação desse fluxo de trabalho, as modificações devem ser isoladas o bastante a ponto de que a atualização do main através de vários pull requests pode ser feita rapidamente e com branches que interferem pouco – ou não interferem – entre si.