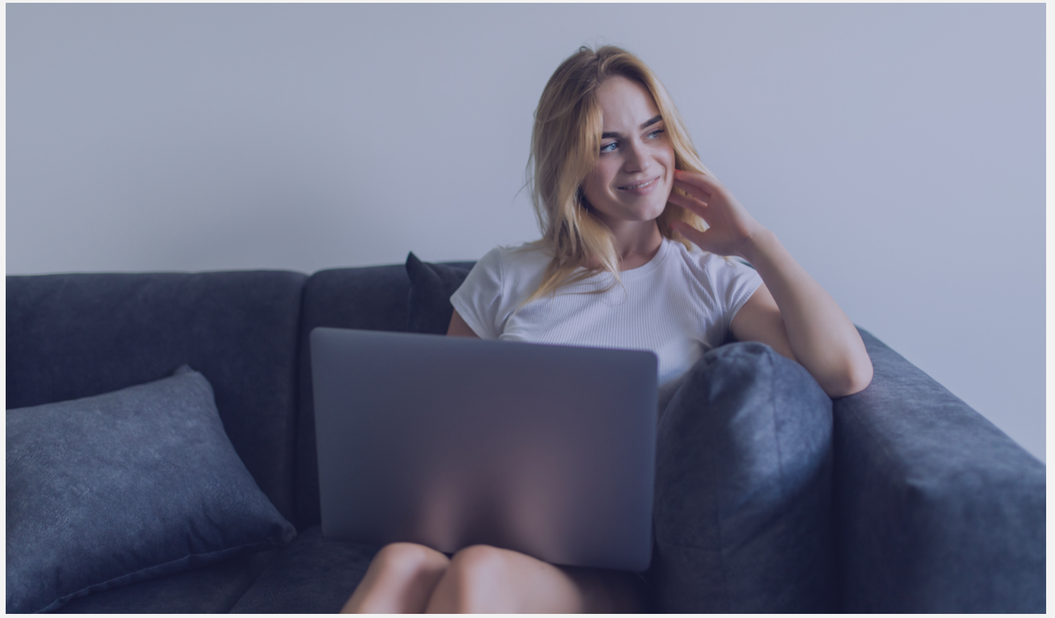
POTÊNCIA FEMININA

GLOSSÁRIO

Git & GitHub:

Entenda o que é Git e GitHub e dê os primeiros passos para se tornar uma desenvolvedora.



Consultas rápidas

Utilize este documento para consultar termos utilizados de forma recorrente no curso e facilitar o seu aprendizado.

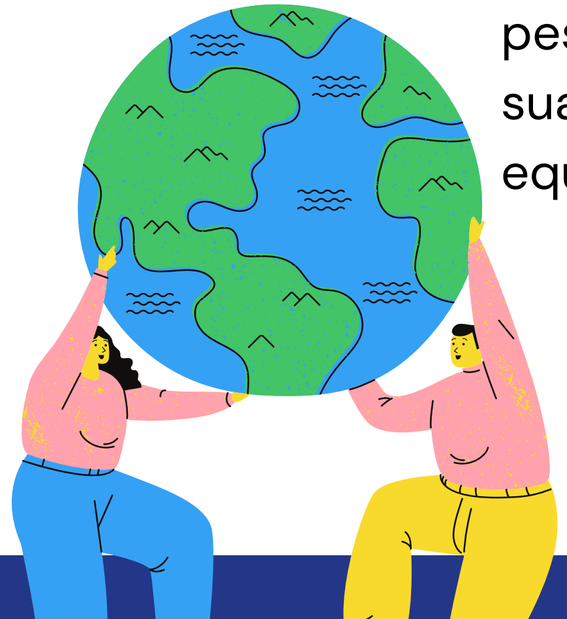
Lembre-se de conversar com suas colegas, vizinhas e familiares sobre o que está aprendendo no Potência Feminina.

Colabore

Ouviu algum termo no curso e gostaria de vê-lo aqui no glossário? Converse a pessoa que está facilitando este curso em sua ONG ou envie uma mensagem para equipe do Potência Feminina.

Agradecemos a sua ajuda 

Git: é um sistema de controle de versão de arquivos. Através deles podemos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.



GitHub: é um site que abriga um software de controle de versão de desenvolvimento através do sistema Git. Através desse sistema, duas ou mais pessoas podem editar arquivos existentes ou criar novos sem arriscar que suas alterações sejam apagadas pelas modificações de outros membros da equipe.

Overview: termo em inglês que significa "visão geral"

Default: termo em inglês que significa "padrão"

Open Source: termo em inglês que significa código aberto. Isso diz respeito ao código-fonte de um software, que pode ser adaptado para diferentes fins.

Debian e Ubuntu: são os dois sistemas operacionais Linux mais populares atualmente. O Ubuntu é um sistema operacional Linux completo e disponível livremente. O Projeto Debian é uma associação de indivíduos que têm como causa comum criar um sistema operacional livre.

Mobile: toda tecnologia que permite a uma pessoa se movimentar sem abdicar dela pode ser considerada mobile.

Terminal: é aquela famosa tela preta na qual você digita comandos para dar instruções para um computador e visualizar informações dessa máquina. Ou seja, ele serve para você executar tarefas no computador sem utilizar a interface gráfica, com pastinhas e ícones, ou o bom e velho mouse.

Versionamento: versionamento de software é um processo de controle de versões estabelecido por meio de numerações diferentes. Isso permite que os programadores saibam quando e quais alterações foram realizadas, acompanhando as mudanças aplicadas no software.

Repositório: é a pasta onde armazenamos o nosso projeto. Dentro do repositório podemos ter uma série de pastas e arquivos.

Issues: no github, issues é onde os usuários contribuem ou usuários finais da aplicação reportam os problemas/bugs encontrados. Ficando assim mais fácil a correção do problema e a obtenção das informações para simulação dos problemas. Issues é traduzido por *problemas*.

Commit: é uma alteração individual em um arquivo (ou conjunto de arquivos). Quando você faz um commit para salvar seu trabalho, o Git cria um ID exclusivo (também conhecido como "SHA" ou "hash") que permite que você mantenha um registro das alterações, dos autores e de quando as alterações foram feitas. Em geral, os commits têm uma mensagem que mostra uma breve descrição das alterações.

Diff: é a diferença nas alterações entre dois commits ou o registro das alterações salvas. O diff descreverá visualmente o que foi adicionado ou removido de um arquivo desde o commit mais recente.

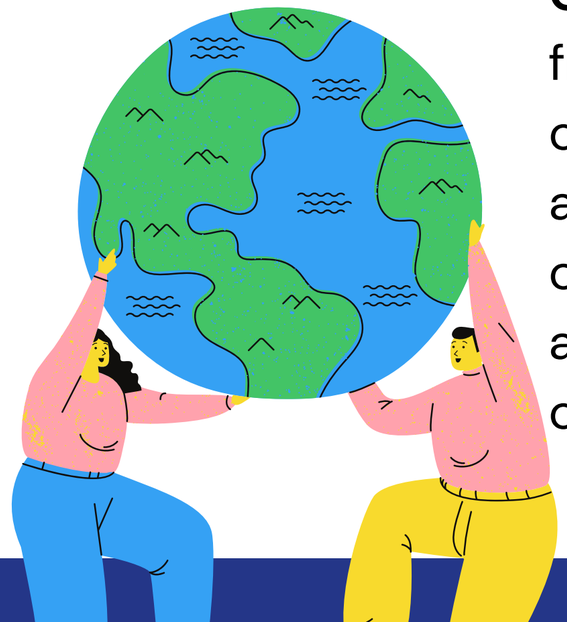
Alterações: o github entende por alterações no projeto, qualquer criação, renomeação ou exclusão de arquivo, inserção ou exclusão de uma linha em um arquivo (uma linha modificada é uma inserção ou exclusão de trecho de código ou até mesmo pontuação).

Branch: é uma versão paralela de um repositório. O branch está contido no repositório, mas não afeta o branch principal ou mestre, o que permite trabalhar livremente sem interromper a versão ativa. Após concluir as alterações necessárias, você poderá fazer merge entre o branch alterado e o branch mestre para publicar as alterações.

Branch de recurso (ou teste/homologação): branch usado para testar novos recursos ou corrigir problemas que não estejam em produção. Também é conhecido como branch de tópico.

Branch de produção (ou master): branch com alterações finais prontas para uso ou implementação em um aplicativo ou site.

Merge: a operação de merge aplica as alterações de um branch (no mesmo repositório ou em uma bifurcação) a outro. Em geral, isso acontece como uma "pull request" (que pode ser considerada como uma solicitação de merge) ou pela linha de comando. É possível fazer um merge via pull request pela interface web do GitHub.com, se não houver alterações conflitantes, ou pela linha de comando.



Clone: é a cópia de um repositório que fica no seu computador, em vez de ficar em algum lugar do servidor de um site. Criar um clone permite alterar os arquivos no editor de sua escolha e usar o Git para monitorar suas alterações sem precisar estar online. O repositório clonado continua conectado à versão remota, de modo que você possa fazer push das alterações locais para manter as duas versões sincronizadas quando estiver online.

Pull: tem relação com os processos de fetch e merge das alterações. Por exemplo, se alguém editou o arquivo remoto no qual você e outra pessoa estão trabalhando, você vai querer fazer pull dessas alterações em sua cópia local para atualizar o arquivo.

Push: fazer push significa enviar as alterações que passaram por commit para um repositório remoto no GitHub.com. Por exemplo, se fizer alguma alteração local, você poderá fazer push para que outras pessoas tenham acesso a ela.

Fork: copia um repositório de outra conta no github para a sua.

Rebase: reaplicar uma série de alterações de um branch para uma base diferente e redefinir o HEAD daquele branch para o resultado.

Pull Request: são alterações propostas em um repositório enviadas por um usuário e que são aceitas ou rejeitadas pelos colaboradores do repositório. Assim como os problemas, as pull requests têm um fórum próprio de discussão.

Fetch: ao usar *git fetch*, você inclui alterações do repositório remote no seu branch local sem comprometê-las. Diferentemente do que ocorre com git pull, fazer fetch permite revisar as alterações antes de fazer commit delas no branch local.

HEAD: um commit definido de um branch, geralmente o commit mais recente na ponta do branch.

Features: funcionalidade ou característica de um projeto ou programa.

Kanban: é um método visual para gerenciar e conduzir o trabalho. Divide as tarefas em três grupos: A Fazer, Fazendo e Feito.

Markdown: é uma sintaxe usada para padronizar e facilitar formatação de texto na web, utilizada em aplicativos como Slack e GitHub.

Git Bash: é o aplicativo para ambientes do Microsoft Windows que oferece a camada de emulação para a experiência de linha de comando Git.

Upstream: Quando mencionamos um branch ou bifurcação, o branch primário no repositório original é frequentemente denominado "upstream" (ou branch de origem), por ser o local principal de onde outras alterações virão. Por sua vez, o branch/bifurcação em que você está trabalhando é chamado de "downstream".

SSH: chaves SSH são uma forma de identificar-se com um servidor on-line, usando uma mensagem criptografada. É como se o seu computador tivesse sua própria senha exclusiva para outro serviço. GitHub usa chaves SSH para transferir informações de forma segura para o seu computador.

Comandos Básicos e Como Usá-los

git init: inicializando o seu repositório.

git status: informa os arquivos e pastas que não estão em conformidade, comparando seu repositório local e remoto.

git clone: fazendo o clone do repositório desejado.

git add: podemos realizar o comando git add . para incluir todos os arquivos na fila ou git add seguido do nome do arquivo específico que você quer incluir. Ex: *git add notasdeaula.md*

git commit: o comando deve ser seguido de -m e a mensagem do commit deve estar entre aspas simples ou duplas. O commit deve ser uma frase curta mas de fácil compreensão de quais arquivos/funções foram alteradas. Ex: *git commit -m "notas da aula 3 do modulo 2"*

git push: enviando de fato suas alterações para a branch. Caso a branch local não tenha ligação com a branch remota, o comando deve ser executado na primeira vez como *git push --set-upstream origin master*.

git pull: trazendo as alterações realizadas na branch remota para a sua branch local. Após realizar o *git pull*, você pode executar o comando *git status* para ter certeza de que a branch local e a remota estão com as mesmas informações.

Trabalhando com Branchs

Criar branch pelo terminal: utilize o comando *git checkout -b nomedabranch*. Dessa forma, você cria uma nova branch e já pode executar comando dentro dela. Você pode usar o comando *git checkout master* para voltar para a branch principal.

Criar branch pelo github: também é possível criar branch pelo github, seguindo o passo a passo da aula. No site talvez fique mais "fácil" pelo fato de terem botões com os nomes, mas não deixe de exercitar o uso dos comandos.

Enviando sua branch local para o seu repositório remoto: você deve fazer o procedimento padrão de git add, git commit e git push. No entanto, no primeiro commit, aparecerá a mensagem de que a banch ainda não possui upstream. Para isso, você utiliza o comando já citado no material para fazer a configuração necessária da branch com a master. No github, o pull request fica disponível para análise antes de realmente modificar a branch.

Fazendo um merge local: lembre sempre que, caso você precise pegar arquivos da branch remota para a sua branch local, deve usar o comando *git pull*. Para unir a branch local com a branch vazia que você criou anteriormente direto no github, você se certifica que está tudo ok, e utiliza o comando *git merge nomedapasta* (nesse caso, é o nome da branch que você quer unir com a branch em que você está executando o comando). Verifique com o git status a situação da branch, e realize o pull request para a branch master.

Colaborando com projetos de outras pessoas

Fazendo um fork: você pode fazer um fork de um projeto Open Source para contribuir e colaborar. Com acesso ao código fonte do projeto, você pode faezr pull requests com melhorias. Além disso, você pode contribuir em um projeto Open Source gerando documentação, atualizando read.me, pois essas infos também são importantes. No caso do curso, fizemos o fork e clonamos a branch para trabalhar nela e enviar pull requests para a Kamila.

Mantendo seu projeto atualizado: você pode atualizar seu fork pelo github, que é mais simples. Ou então você pode atualizar utilizando comandos, utilizando o comando *git remote add upstream* e a url do repositório no github. Depois o comando git fetchupstream, que funciona semelhante a um git pull. Em seguida, utilize o comando *git rebase upstream/master*, que vai fazer o balanço entre a upstream e a master.

Criando uma branch para enviar as suas alterações: você pode criar uma nova branch para enviar as pull requests para o repositório da Kamila. Siga o passo a passo da aula, ele é bem semelhante aos passos já executados por você antes (criação de branch, configuração de upstreame etc.)

Criando sua primeira pull request: seguindo as instruções da aula, você verá como é fácil enviar a pull request para um projeto no github. Apenas fique atenta para manter a sua branch atualizada, para evitar problemas de merge, e selecionar a branch do projeto e não o seu fork para fazer o pull.

GitHub Project

Ferramenta de gerenciamento de tarefas do próprio github, em que você utiliza da mesma lógica de issues e pull request para organizar o seu próprio trabalho ou de um time. Você pode linkar pull requests com issues e dar uma maior fluidez ao desenvolvimento de um projeto, por exemplo. Todo o acompanhamento do projeto pode ser feito utilizando o Github Project. Você pode verificar todas as features do GitHub Project no link abaixo:

Gerenciar seu trabalho com problemas e pull requests - GitHub Docs