Sobre as issues, ou problemas do GitHub, um ponto interessante é que se tivermos as permissões necessárias, ou seja, se formos os donos do repositório, ou fizermos parte da organização da empresa, poderemos fechar a issue ("Close issue"), indicando que o problema já foi resolvido. Se a issue estiver fechada, nestas condições, poderemos reabri-la para informar que o problema voltou a acontecer, por exemplo, e todo o histórico fica salvo no GitHub.

Vamos supor que outra pessoa da mesma equipe, Ana, já conhece o projeto e sabe como resolver o problema que publicamos anteriormente. Ela não poderá simplesmente editar o código criado pelo Vinicius. Isso evita que uma pessoa mal intencionada tenha acesso total ao código, criando bugs propositais, ou incluindo um código que não faz sentido. Então, mesmo que o projeto seja open source, existe sempre alguém **validando estas alterações**, garantindo que elas sejam de fato benéficas para o projeto.

Assim, o primeiro passo para que a Ana possa contribuir no projeto do Vinicius é criar uma cópia do mesmo para trabalhar nele e, no final, quando finalizada a solução da issue, ela enviará ao repositório do Vinicius apenas as alterações feitas. Vamos criar este repositório cópia!

Quando trabalhamos com controle de versões, principalmente com o Git e o GitHub, o nome deste repositório cópia é **fork**, que remete aos dentes de um "garfo", em inglês. Então, logados como Ana, clicaremos no botão "Fork" no topo direito da página. Ao atualizarmos a página, verificaremos a existência da cópia do projeto para analura, que é da Ana, e ela poderá editar este projeto da forma como preferir.

Temos todo este código na máquina local, no diretório "projeto", e garantiremos isto acessando o repositório pelo Terminal digitando cd ana/ seguido de cd projeto/. Executaremos git pull origin master, e teremos a mensagem de que este repositório não existe. Aqui, entra outro assunto: adicionaremos, com o origin, o repositório da Ana, o outro repositório, aquele que copiamos. Para isto, voltaremos ao GitHub e clicaremos em "Clone or download", um botão dropdown que exibe um link, que copiaremos.

No Terminal, digitaremos git remote add origin colar o link no comando e dar "Enter". E então traremos as alterações com git pull origin master. No VS Code, confirmaremos que o código está atualizado como esperávamos. Já temos o repositório local, como Ana, sincronizado com o nosso fork, no GitHub da Ana.

Feito isso, no código no VS Code, adicionaremos um título, acrescentando uma linha antes da abertura da tag <ul>: <h1>Cursos de DevOps</h1>. Criaremos um commit no Terminal começando por git status e git diff para verificarmos as modificações realizadas, e então git add index.html e git commit -m "Título adicionado". Repetiremos o mesmo processo após pularmos uma linha entre a linha recém implementada e <ul>, sendo portanto o último comando digitado git commit -m "Quebra de linha".

Feitas todas estas alterações, enviaremos os dados ao repositório com git push origin master. Porém, o envio será rejeitado, pois não temos permissão para isso. Para resolvermos isto, o primeiro passo será configurar o e-mail da Ana, com git config --local user.email "anaalura123321@gmail.com". Depois, usaremos git push origin master para verificarmos se, com isto, temos sucesso.

Não conseguiremos acesso, e o que acontece é que se fosse em um ambiente MacOX ou Linux, provavelmente você já teria recebido uma mensagem pedindo o login e senha do GitHub. No Windows, como já adicionamos um repositório do GitHub anteriormente, ele trabalha de forma diferente. A partir do momento em que adicionamos um repositório do GitHub, sempre que tentarmos enviar algo para lá, ele tentará utilizar estas credenciais.

Esta é a forma mais simples: apagaremos as credenciais do GitHub acessando "Painel de Controle > Contas de Usuário > Gerenciador de Credenciais > Credenciais do Windows". Dentre as Credenciais Genéricas, clicaremos na seta ao lado de git:https://github.com para que a informação seja expandida, e em "Remover". Assim, se tentarmos refazer o push, o GitHub nos solicitará os dados para login.

Faremos login como "analura", e agora, sim, conseguiremos enviar as alterações. Vamos atualizar a página com o projeto no GitHub, em que teremos mais commits adicionados — referentes ao título, e quebra de linha. De que forma enviaremos as modificações para o dono do repositório, ou seja, para o repositório do Vinicius?

Estas modificações são realizadas por meio de **pull requests**, um pedido de envio de modificação. Então, clicaremos no botão "New pull request", e o GitHub fará o trabalho de verificar se com o Git conseguimos fazer merges sem nenhum conflito, exibindo uma mensagem que diz que ambos os repositórios podem ser unificados sem nenhum problema, e mais abaixo, os commits enviados.

No fim, todas as alterações serão realizadas, neste caso, somente adicionamos duas linhas. Poderemos clicar em "Create pull request" para confirmar que queremos fazer esta ação, cujo título será "Adicionando título na página", sendo possível acrescentarmos um comentário, algo como "Adicionando um título para a página que antes não tinha.". Pressionaremos "Create pull resquest", e isto estará presente no repositório do Vinicius.

Poderemos confirmar isso logando como ele e acessando a aba "Pull requests" no GitHub. Analisaremos os commits e clicaremos em "Merge pull request" e "Confirm merge", já que não há nada a ser alterado. Com isto, teremos um pull request no status **merged**. É possível verificarmos o código, que estará com as alterações devidamente implementadas, e então fecharmos a issue, junto a um texto "Fechado pelo PR #2", clicando em "Close and comment".

A nossa issue estará fechada, e contendo um link para o último pull request junto a todas as alterações realizadas. Resumindo o que vimos, o Vinicius cria um repositório, a Ana enviou uma atualização, posteriormente revisado e autorizado pelo Vinicius, que enviou uma notificação informando que o pull request foi aceito.

Algo que não ficou tão legal, porém, é a lista de commits, pois o commit de quebra de linha não é relevante. Será que não seria interessante termos um único commit? Mas a Ana fez dois commits, e ela tem o direito de trabalhar com quantos quiser! Será que depois de realizar os commits, ela não conseguiria juntar vários em um só? Como será que conseguimos unir commits para ajudar quem for revisar o código?