Insper

01 - Ciclo de vida de um bug

Desenvolvimento aberto 2020/1

Igor Montagner

Discussão inicial e motivação

Suponha que você decidiu liberar um projeto que você desenvolveu nos últimos semestres. Um usuário de seu software teve um problema e te contactou pedindo ajuda. Como você lidaria com isto? Seu grupo deverá responder à seguinte pergunta:

Quem paga por esse novo desenvolvimento?

Workflow distribuídos usando `git`

Neste roteiro trabalharemos no workflow padrão para contribuir com projetos hospedados no Github (mas que também serve para projetos git em geral). Antes de começar cada aluno deverá localizar no github da disciplina a issue correspondente a criação de seu usuário.

Nosso fluxo de trabalho será baseado em três grandes partes. Na primeira será criada uma cópia do repositório insper/dev-aberto onde faremos todas as mudanças necessárias. Na segunda enviaremos nossas modificações para o repositório original usando um *Pull Request*, que é um pedido de aceite das mudançás de um *fork* no repositório original. Por último, atualizaremos nosso *fork* com modificações enviadas pelos colegas.

Alguns pontos a serem destacados no fluxo de trabalho acima:

1. Mesmo que um usuário não tenha acesso ao repositório original ele pode trabalhar em sua cópia e somente quando tudo estiver pronto enviar suas modificações ao repositório original.
2. É necessário que um desenvolvedor do projeto original "se responsabilize" pelas modificações externas aceitas.
3. A aba *Pull requests* permite que os desenvolvedores discutam as propostas de modificações e as melhorem. Todo commit feito após a criação do PR é incluido e pode ser testado por qualquer um.

Parte 1 - Criando uma cópia local

Iremos começar nosso fluxo de trabalho criando um *fork* do repositório insper/dev-aberto. Todas nossas modificações serão feitas no nosso *fork* em um branch separado (o correto é sempre usar um branch diferente para cada issue). Desta maneira nossas modificações ficam completamente isoladas do código original e podemos testá-las lado a lado com o código original.

Primeiro, crie o *fork* via interface do Github. Depois, clone seu fork e crie um novo branch chamado *issue-X*, onde *X* é o número da sua issue no projeto original.

```
$ git checkout -b issue-X
```

Para garantir que você está no diretório do seu fork, execute o comando:

```
$ git remote -vv
```

Os endereços mostrados devem ser os do seu *fork*, não os do projeto original.

Com o *fork* criado e estando no branch *issue-X* (você pode checar usando git branch e mudar usando git checkout issue-X), vamos começar a realizar modificações.

Interagindo com o repositório da disciplina

A criação de usuários e adição de skills é feita usando o comando dev-aberto.py. Para utilizá-lo é necessário instalar os pacotes listados no arquivo requirements.txt. Sua execução no terminal deverá listar os comandos disponíveis.

```
$ python3 dev-aberto.py
```

Para checar se tudo está funcionando direito, liste todos os usuários cadastrados.

Criando um usuário

A criação de usuários é feita com o comando:

```
$ python3 dev-aberto.py new-user
```

Isto criará 3 arquivos na pasta 'students':

- `login`: informações básicas do usuário em formato JSON.
- `login-achievements`: arquivo criptografado contendo as entregas de cada aluno em formato JSON.
- `login.key`: chave criptográfica do arquivo acima.

Verifique que seu usuário foi criado corretamente listando novamente os usuários existentes. Seu usuário deverá apresentar um * ao lado do nome, o que significa que o arquivo login.key está presente no sistema.

Você não deverá incluir em seu PR o arquivo `*.key`. Ele deverá ser enviado por email para o professor. Faça isto agora antes que esqueça!

Verifique também que você consegue usar o comando compute-grade. Se estiver tudo ok, passe para o próximo item.

Adicionando uma skill

Com o usuário criado podemos adicionar a skill *Primeiros passos*. Você já deve ter notado que a chave do professor está disponível (arquivo `students/igorsm1.key`). Isto foi feito para que vocês possam ter ao menos um exemplo de como cada skill deverá ser adicionada. Veja abaixo um exemplo de como a skill deverá ser incluida:

```
$ python3 dev-aberto.py edit-achievements igorsm1
```

Isto abrirá um arquivo para edição no *Vi*. Veja o formato usado para incluir a skill e faça o mesmo para o seu usuário.

Se você quiser usar outro editor de texto pode setar a variável de ambiente `EDITOR` logo antes de chamar o `dev-aberto.py`.

Confira que sua skill foi corretamente adicionada usando o comando compute-grade.

Se seu repositório estiver *OK* ajude os colegas do seu grupo.

Parte 2 - enviando as modificações para o projeto original

Vamos agora criar um commit e enviá-lo como *Pull Request* para o repositório da disciplina. Adicione os arquivos criados (menor o arquivo `*.key`!) e faça um commit com a seguinte mensagem (substituindo o *X* pelo número da sua issue no repositório):

```
Fixes issue #X
```

Isto faz com que a issue seja automaticamente fechada quando (e se) este commit for integrado ao repositório. Execute um `git push` e continue.

Com as suas modificações já presentes no seu *fork* é hora de enviá-las para o repositório original. Isto é feito na interface do Github. Primeiro, acesse seu *fork* no navegador, localize seu branch *issue-X* e clique no botão "Pull request".

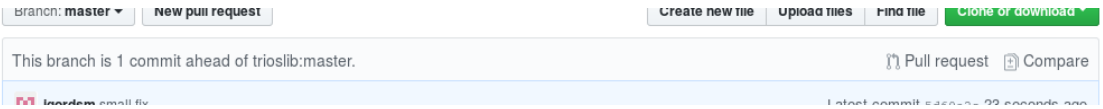

Figure 1: Esta mensagem aparece quando seu *fork* tem commits que não estão presentes no repositório original.

O título de seu Pull Request deverá ser *Cria usuário login*. Seu PR deverá conter somente um commit e deverá ter como origem o branch `issue-X` criado acima. Ele só será aceito se estiver tudo em ordem **E** se você tiver enviado o arquivo `*.key` para o professor.

Não serão aceitos PRs feitos a partir do `master` nem que tenham mais de um commit.

Assim que seu PR for aceito você pode remover o branch issue-X.

Parte 3 - atualizando seu *fork*

Ao ter seu PR aceito você deve ter notado que seu commit aparece no `master` do projeto original mas não aparece no seu fork. Isto ocorre pois um fork não é automaticamente atualizado quando seu repositório original correspondente receber novos commits. Para que isto ocorra é necessário realizar a sincronização *manualmente*. Isto envolve duas etapas. Na primeira, que só precisa ser feita uma vez, é adicionado um novo repositório remoto que aponta para o repositório original. Na segunda baixamos os arquivos deste repositório remoto e os incorporamos aos nossos.

O Github tem uma excelente documentação explicando como fazer o primeiro passo (https://help.github.com/articles/configuring-a-remote-for-a-fork/) e depois como sincronizar seu fork com o repositório original (https://help.github.com/articles/syncing-a-fork/).

Ao executar este passos agora você deverá estar vendo o seu commit aparecer no branch `master` de seu repositório local. Publique as modificações importadas no seu fork dando um `git push`.