Trabalhando com Branches

Depois de liberarmos o código de uma aplicação para o cliente, continuamos melhorando funcionalidades existentes e criando novas funcionalidades, sempre comitando essas alterações. Mas, e se houver um bug que precisa ser corrigido imediatamente? O código com a correção do bug seria liberado junto com funcionalidades pela metade, que ainda estavam em desenvolvimento.

Para resolver esse problema, a maioria dos sistemas de controle de versão tem **branches**: linhas independentes de desenvolvimento nas quais podemos trabalhar livremente, versionando quando quisermos, sem atrapalhar outras mudanças no código.

Em projetos que usam Git, podemos ter tanto branches locais, presentes apenas na máquina do programador, quanto branches remotas, que apontam para outras máquinas. Por padrão, a branch principal é chamada **master**, tanto no repositório local quanto no remoto. Idealmente, a master será uma branch estável, isto é, o código nessa branch estará testado e pronto para ser entregue.

Para listar as branches existentes em seu repositório Git, basta executar:

```
git branch
```

Para saber mais: Branches e o HEAD

No Git, uma branch é bem leve: trata-se apenas de um ponteiro para um determinado commit. Podemos mostrar os commits para os quais as branches estão apontando com o comando

```
git branch -v.
```

Há também um ponteiro especial, chamado HEAD, que aponta para a branch atual.

Criando uma branch

Uma prática comum é ter no repositório branches novas para o desenvolvimento de funcionalidades que ainda estão em andamento, contendo os commits do que já foi feito até então. Para criar uma branch nova de nome work a partir do último commit da master, faça:

```
git branch work
```

Ao criar uma nova branch, ainda não estamos automaticamente nela. Para selecioná-la:

```
git checkout work
```

Criando e selecionando uma branch

É possível criar e selecionar uma branch com apenas um comando:

```
git checkout -b work
```

Para visualizar o histórico de commits de todas as branches, podemos fazer:

```
git log --all
```

Para uma representação gráfica baseada em texto do histórico, há a opção:

```
git log --graph
```

Bibliografia

Caelum - **Trabalhando com Branches**
Disponivel em:
https://www.caelum.com.br/apostila-java-testes-spring-design-patterns/primeiras-interacoes-com-o-novo-projeto/