**Git GUI e GitHub**

**Hospedando código no GitHub**

Em 2008, foi criado o GitHub, uma aplicação Web que possibilita a hospedagem de repositórios Git, além de servir como uma rede social para programadores que disponibiliza repositórios Git acessíveis remotamente.

Vimos que a utilização de um repositório remoto nos ajuda a ter uma melhor organização para trabalhar em equipe em um projeto, além de também funcionar como backup. Entretanto, como o repositório remoto está localizado em um servidor da empresa, ficamos limitados a efetuar a sincronização apenas dentro dela, pois precisaremos estar conectados à rede local para conseguirmos acessar o servidor.

Se algum dia um desenvolvedor precisar sincronizar as alterações com o repositório remoto de fora da empresa, por exemplo de casa, isso não vai ser possível, pois ele não vai estar conectado à rede local. Até é possível acessar a rede local da empresa externamente, mas para isso será necessário configurar uma VPN (Virtual Private Network), ou liberar acesso externo ao servidor via SSH, o que pode ser algo complicado de se fazer, além de também poder ser proibido por algumas empresas, devido a políticas de segurança.

O ideal seria que o repositório remoto estivesse acessível pela internet, pois assim poderíamos acessá-lo de qualquer lugar do mundo, bastando apenas estarmos conectados à internet. O GitHub é muito utilizado para projetos open source que possuem vários colaboradores do mundo todo.

**Serviços de hospedagem de projetos**

Existem alguns serviços para hospedagem de projetos na internet, que permitem a utilização de ferramentas de controle de versão. A ideia é que você possa hospedar seus projetos, juntamente com os repositórios, na internet, e assim será possível acessá-los de qualquer lugar do mundo.

Dentre os principais serviços que suportam o Git, estão:

* GitHub
* Bitbucket
* Google Code

Vários projetos importantes estão hospedados no GitHub, dentre eles:

* jQuery
* Ruby on Rails
* Node.js
* Django
* Bootstrap
* Jenkins
* Spring
* JUnit

O GitHub é também uma ferramenta colaborativa, pois nos permite interagir nos repositórios, por meio de algumas funcionalidades como:

* **Issue Tracker:** para criação e gestão de bugs e milestones do projeto;
* **Pull Requests:** para que outros usuários possam enviar seus commits com alterações no projeto, ou commits com correções de bugs;
* **Commit Comments:** para que os usuários possam comentar e discutir sobre as modificações no código, de um determinado commit.

Além disso, no GitHub é possível criar organizações, e dentro delas criar equipes, para que então seja possível vincular determinados usuários a determinadas equipes, bem como vincular as equipes aos repositórios em que elas vão trabalhar. Isso é muito útil para empresas que possuem muitos projetos e muitos colaboradores, pois permite uma melhor organização.

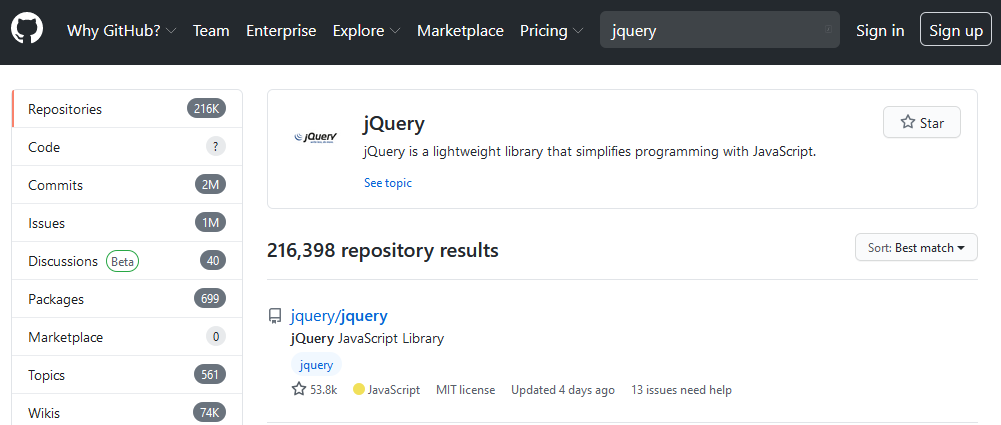
Por conta dessas funcionalidades e por permitir uma maior colaboração nos repositórios, o GitHub é considerado por muitos como uma espécie de rede social para desenvolvedores.

**Encontrando projetos e visualizando o código-fonte**

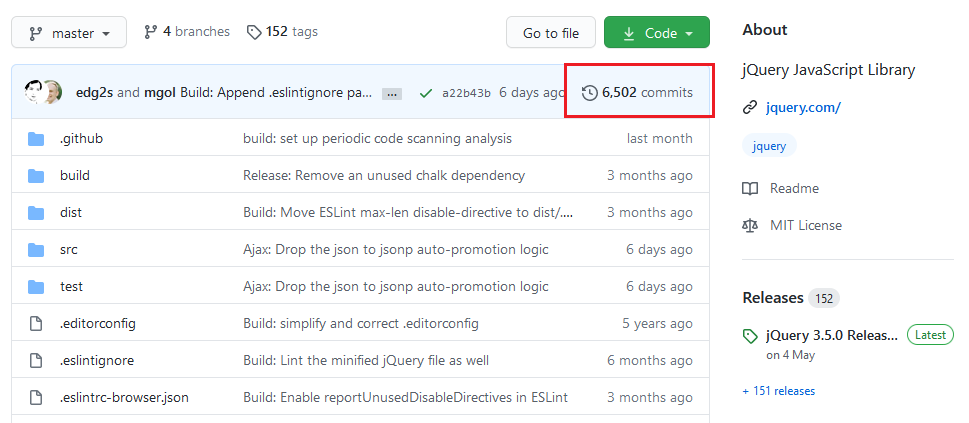
Vamos acessar o site do GitHub e visualizar o código-fonte de algum projeto open source. Para isso, basta abrir um navegador e acessar o endereço:

**http://github.com**

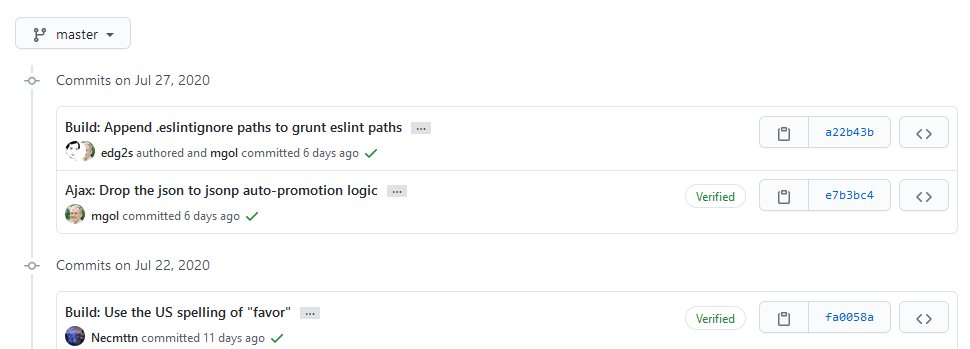
Na página inicial do GitHub, nela podemos efetuar login, cadastrar um novo usuário, fazer buscas e acessar alguns links interessantes. No topo da página existe um campo de texto, usado para pesquisar por repositórios ou usuários. Vamos pesquisar pelo repositório do jQuery, uma famosa biblioteca de JavaScript, bastando para isso digitar a palavra jquery no campo de texto e apertar a tecla Enter. Após realizar a pesquisa, será apresentada uma página com o resultado, listando todos os repositórios que possuamno nomeou na descrição a palavra jquery.



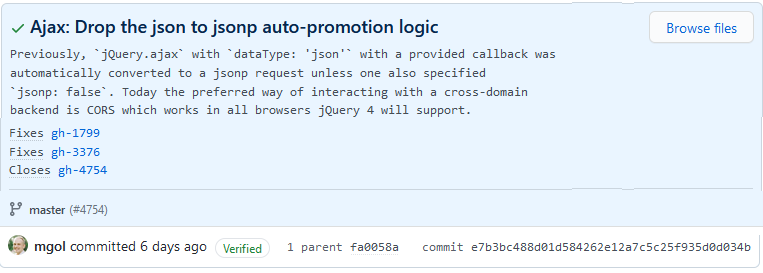
Vamos selecionar o primeiro resultado, que no caso é o projeto jQuery que estamos buscando. Na página do repositório do jQuery é possível navegar pelo código-fonte, visualizar a descrição do projeto, os commits, os usuários que contribuem para o projeto, e até mesmo baixar o repositório compactado em um arquivo zip. Vamos agora visualizar os últimos commits realizados no projeto. Para isso, basta clicar no link commits:



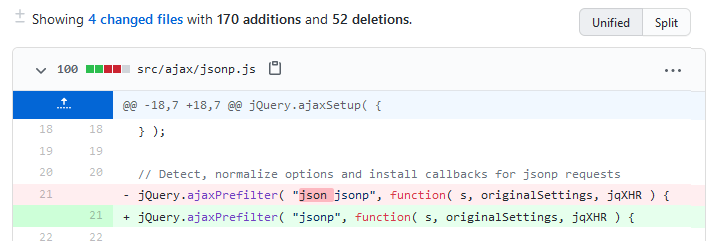
Será apresentada uma página listando os últimos commits efetuados no projeto, onde também é possível navegar por commits mais antigos. Visualizar esta página seria como executar o comando git log no repositório.



Ao clicar em algum dos commits, será apresentada uma página de detalhamento:

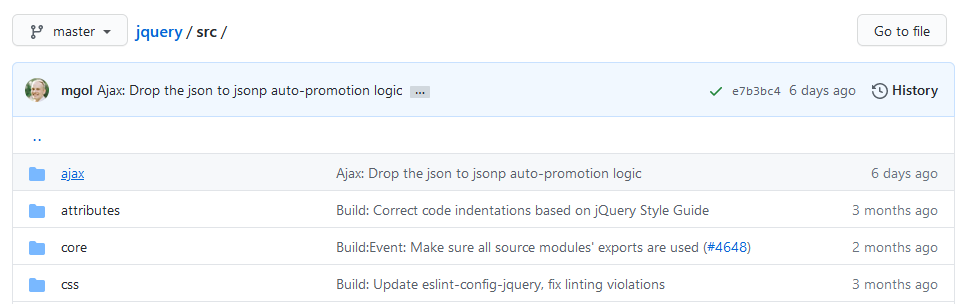


E onde podemos visualizar as alterações efetuadas, similar a quando executamos o comando git diff:

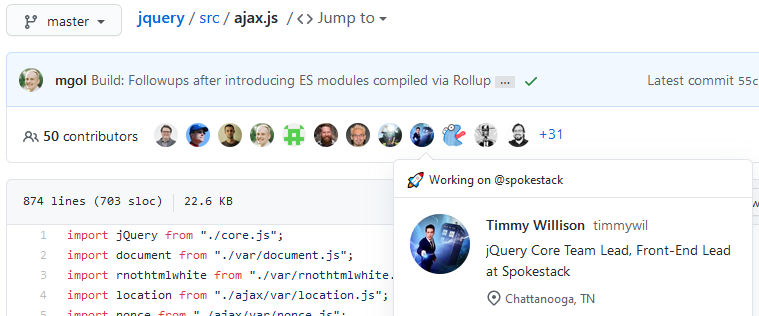


**Visualizando o código-fonte do projeto**

De volta à página principal do repositório, agora vamos navegar pelo código-fonte. Já na página principal do repositório é possível visualizar os arquivos e diretórios do projeto. Vamos navegar pelo diretório src:



Serão exibidos todos os arquivos e subdiretórios existentes, e podemos detalhar algum deles, por exemplo o arquivo ajax.js:



Além de ser exibido todo o código-fonte do arquivo, nessa página também é possível visualizar todos os usuários que já efetuaram commits com modificações neste arquivo.

**Planos de contratação oferecidos pelo GitHub**

O GitHub possui alguns planos que podemos contratar, sendo um gratuito, e os outros com custos mensais. No plano gratuito do GitHub, o usuário não tem nenhum custo, entretanto ele somente poderá criar repositórios públicos, ou seja, qualquer pessoa poderá encontrar seus repositórios e inclusive ter acesso ao código-fonte.

Este plano é ideal para usuários que querem criar e/ou colaborar com projetos open source. Mas muitas empresas possuem projetos que são restritos, e não podem ter o código-fonte acessível a qualquer pessoa. Neste caso, a empresa deverá contratar algum dos planos pagos oferecidos pelo GitHub. A diferença entre os planos pagos está na quantidade de repositórios privados que poderão ser criados. No plano mais barato, que atualmente custa $7 por mês, é possível criar até 5 repositórios privados, enquanto que no plano mais caro, que atualmente custa $50 por mês, essa quantidade de repositórios privados sobe para 50.

Após escolher o plano que mais se adeque às nossas necessidades, devemos finalizar o cadastro clicando no botão Finish sign up, e seremos redirecionados para a página que mostra os repositórios do nosso usuário e outras informações. Essa página é conhecida como Dashboard.

**Colaborando com projetos open source**

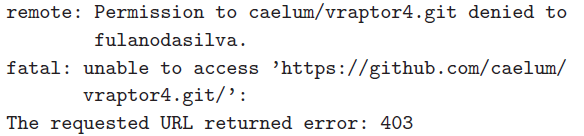
Conforme citado anteriormente, existem milhares de projetos hospedados no GitHub, sendo que muitos destes são open source, ou seja, significa que qualquer desenvolvedor pode colaborar com melhorias e correções de bugs. Vamos supor que queremos colaborar com algum projeto open source, por exemplo, o VRaptor, um framework MVC para desenvolvedores Java. Primeiramente vamos precisar clonar o repositório, para termos acesso ao código-fonte, e então efetuar as melhorias desejadas:

**https://github.com/caelum/vraptor4.git**

Já temos uma cópia do repositório com todos os arquivos do projeto. Agora basta efetuar as melhorias desejadas, registrando as alterações realizadas com commits. Mas lembre-se de que os commits realizados existem apenas no nosso repositório local, precisamos agora enviá-los para o repositório remoto do projeto, no GitHub, com o comando git push:

**git push origin master**

Entretanto, a execução do comando anterior falha, gerando a seguinte mensagem de erro:



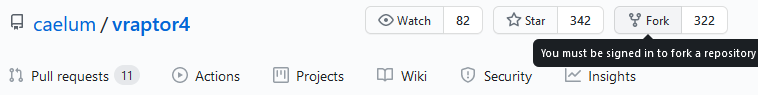
O problema é que o usuário fulanodasilva não tem permissão para enviar commits para o repositório do VRaptor, pois esse repositório não pertence a ele, mas sim ao usuário caelum. Para resolver esse problema, o usuário caelum poderia acessar a página de configurações do repositório VRaptor no GitHub, e adicionar o usuário fulanodasilva como colaborador.

Mas essa abordagem deve ser utilizada apenas para adicionar os usuários que fazem parte da equipe de desenvolvimento do projeto, ou seja, apenas os desenvolvedores que trabalham efetivamente no projeto. Dessa forma, acaba sendo inviável ter que adicionar um novo usuário como colaborador toda vez que um novo desenvolvedor quiser colaborar com o projeto.

Além disso, a partir do momento em que adicionamos um usuário como colaborador, ele passa a ter permissão de enviar commits para o repositório sem restrições, o que pode ser arriscado, pois não há uma avaliação prévia das alterações que foram realizadas.

**Pull requests**

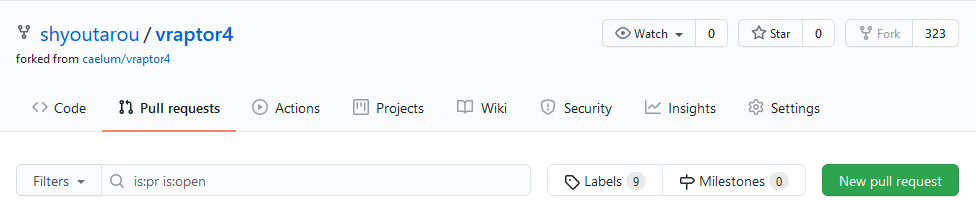
A solução do GitHub para a colaboração com projetos open source, foi uma funcionalidade chamada de pull requests. Primeiramente, devemos acessar a página do projeto que queremos colaborar no GitHub, no nosso caso http://github.com/caelum/vraptor4, e clicar no botão Fork:



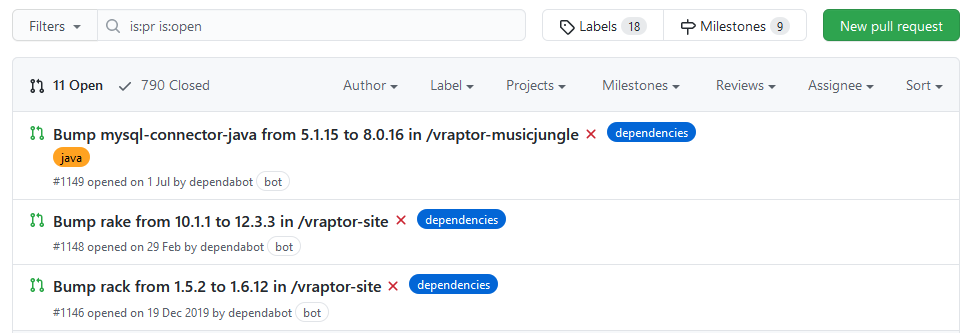
Repare que agora o nosso usuário do GitHub possui um repositório chamado vraptor4, ou seja, esse repositório é a nossa versão do projeto. O próximo passo é clonar o repositório a partir do nosso usuário do GitHub, e não do usuário caelum, pois assim conseguiremos realizar o push dos commits locais para o nosso repositório no GitHub:

**git clone** [**https://github.com/fulanodasilva/vraptor4.git**](https://github.com/fulanodasilva/vraptor4.git)

Agora, após realizar as alterações e commits no projeto, conseguiremos realizar o push normalmente, pois o repositório remoto no GitHub pertence ao nosso usuário. Entretanto, os commits foram enviados apenas para o nosso repositório no GitHub. Para que os commits sejam integrados ao repositório original do projeto, no caso o repositório vraptor4 do usuário caelum, devemos enviar um Pull Request para o usuário caelum, solicitando que seja feito um pull dos nossos commits.



Para criar um Pull Request devemos acessar a página do nosso repositório no GitHub, clicar no link Pull Requests, e em seguida clicar no botão New pull request. Seremos redirecionados para a página de criação do pull request, na qual serão listados os commits e alterações realizados. Para confirmar a criação do pull request, basta clicar no botão Create pull request:



Após criarmos o pull request, o usuário caelum receberá uma notificação do GitHub, e poderá efetuar o pull dos nossos commits, integrando os ao repositório original do projeto. Claro, isso após uma análise das alterações efetuadas em nossos commits. Esse é o processo utilizado pelos desenvolvedores que colaboram com projetos open source hospedados no GitHub.

**Protocolos suportados pelo Git**

Quando adicionamos ou clonamos um repositório Git, devemos informar a url do repositório, que utiliza algum protocolo para comunicação e transferência de dados. O Git suporta quatro protocolos para comunicação e transferência de dados:

* **Local**
* **SSH**
* **Git**
* **HTTP/HTTPS**

**Protocolo local**

O protocolo local foi o utilizado neste capítulo. Ele pode ser utilizado quando o repositório remoto estiver localizado no mesmo computador em que se encontra o repositório local, ou em outro computador que esteja conectado na mesma rede. A utilização do protocolo local é feita com o uso do prefixo file:// na URL do repositório a ser clonado:

**git clone file:///opt/repositorios/moveis-ecologicos.git**

É possível omitir o prefixo file:// na URL, bastando apenas informar o caminho do repositório:

**git clone /opt/repositorios/moveis-ecologicos.git**

As vantagens da utilização do protocolo local são a sua simplicidade, e o uso das permissões existentes de acesso a arquivos e diretórios. Já uma das desvantagens é a dificuldade de configuração para acesso externo à rede.

**Protocolo SSH**

O protocolo SSH é, provavelmente, o mais utilizado, por ser rápido, seguro, simples de configurar e por suportar tanto a leitura quanto a escrita de dados. O uso do protocolo SSH é feito com a URL seguindo o padrão usuario@servidor:/caminho/repositorio.git. Por exemplo:

**git clone root@192.168.1.1:/opt/repositorios/moveis-ecologicos.git**

**Protocolo Git**

O Git possui um protocolo próprio, que é similar ao SSH, mas sem o mecanismo de autenticação. Por conta disso, ele acaba sendo mais rápido; entretanto não é seguro, e seu uso é apenas para leitura. Para clonar um repositório utilizando o protocolo Git, a URL deve possuir o prefixo git://:

**git clone git://192.168.1.1/opt/repositorios/moveis-ecologicos.git**

**Protocolo HTTP/HTTPS**

O Git também suporta o protocolo HTTP, que é bastante utilizado quando estamos trabalhando em empresas que possuem um controle rígido de segurança, e a porta 22, utilizada pelo protocolo SSH, é bloqueada. Para clonar um repositório utilizando o protocolo HTTP, a URL deve possuir o prefixo http://:

**git clone http://192.168.1.1/opt/repositorios/moveis-ecologicos.git**

Também é possível utilizar o protocolo HTTPS, que adiciona uma camada de segurança sobre o HTTP, com a utilização do protocolo SSL/TLS:

**git clone https://192.168.1.1/opt/repositorios/moveis-ecologicos.git**