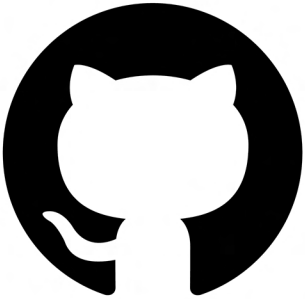
GUIA DO MOCHILEIRO TECH



GitHub NA PRÁTICA

Quer aprender a gerenciar seus projetos de código, colaborar com segurança com outras pessoas desenvolvedoras e fazer seu portfólio?

Dê os seus primeiros passos no GitHub e domine as funcionalidades essenciais para sua carreira.

alura

Quando trabalhamos com desenvolvimento, podemos nos deparar com projetos em que várias pessoas trabalham em colaboração.

Por isso, saber gerenciar e versionar seu código, além de colaborar e compartilhar projetos com outras pessoas desenvolvedoras é fundamental para sua carreira.

Esse material foi criado para explorar as principais funcionalidades do GitHub e te ajudar a dar os primeiros passos nessa ferramenta essencial para devs.

Mergulhe em todas as possibilidades que o GitHub pode te oferecer.

Sumário

O que é GitHub?

Imagine a seguinte situação: você precisa gerenciar um projeto de desenvolvimento de software com uma equipe global de devs.

O desafio de coordenar todas as contribuições de código é imenso. Na Alura, por exemplo, o time de desenvolvimento é segmentado em diferentes funções: há uma equipe que desenvolve as interfaces visuais da plataforma e outra que processa toda a operação da plataforma.

Nesse ponto, surge uma questão fundamental: como assegurar que todas as peças se encaixem perfeitamente e garantir um trabalho em conjunto harmonioso?

É aí que entram as ferramentas do GitHub, que, como você vai ver adiante, apresenta boas respostas para essas questões.

O GitHub é uma plataforma para gerenciar códigos e criar um ambiente de colaboração entre pessoas desenvolvedoras, usando o Git como sistema de controle. É lá que você, provavelmente, vai ter seu repositório e usar no dia a dia.

Inclusive, o sistema do GitHub permite que você altere arquivos lá mesmo – embora não seja uma ótima ferramenta, já que não tem um editor, nem um ambiente de desenvolvimento e de testes.

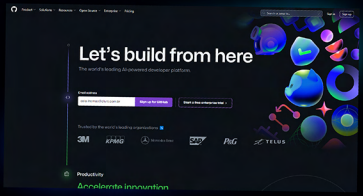
Como criar uma conta no GitHub?

Você já sabe que o GitHub é uma plataforma para hospedar repositórios de código e colaborar em projetos de software. Mas, para começar a explorar esses recursos, você precisa criar uma conta.

As etapas para criar uma conta podem variar de acordo com as atualizações da plataforma sem aviso prévio.

Mas, de forma geral, os passos que você deve fazer para criar uma conta no GitHub são os seguintes:

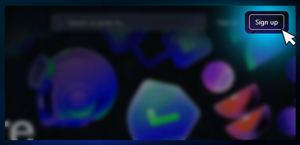
Passo 1: Acesse o site



Abra o seu navegador da web e acesse o site do GitHub em

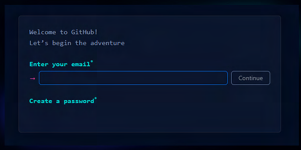
https://github.com

Passo 2: Comece a criar a conta



Na página inicial do GitHub, localize o botão **"Sign up"** (Inscrever-se), normalmente no canto superior direito da página. Clique nele para iniciar o processo de criação da conta.

Passo 3: Preencha suas informações

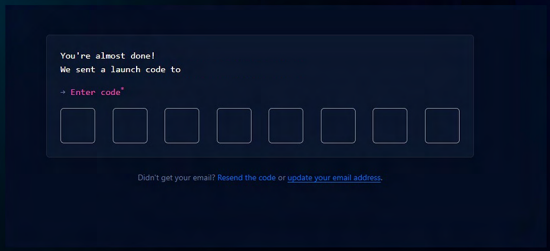


Depois disso, você deve preencher suas informações pessoais, incluindo o nome de usuário, endereço de e-mail e senha.

Passo 4: Verifique o captcha

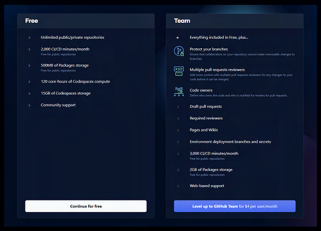
Para garantir que você não é um robô, o GitHub pode solicitar que você complete uma verificação de Captcha. Siga as instruções para provar que você é uma pessoa legítima.

Passo 5: Verificação de e-mail



O GitHub pode enviar um email de verificação para o endereço que você forneceu. Verifique sua caixa de entrada e siga as instruções para confirmar seu e-mail.

Passo 6: Escolha um plano



O GitHub oferece planos gratuitos e pagos. Selecione o plano que melhor atende às suas necessidades. Você pode começar com o plano gratuito e, se necessário, fazer upgrade posteriormente.

Pronto! Se você concluiu esses passos, já criou uma conta na plataforma com sucesso.

Agora, você pode explorar os recursos como criar repositórios, colaborar em projetos e compartilhar seu trabalho com outras pessoas desenvolvedoras.

Como criar um repositório no GitHub?

Criar um repositório no GitHub é um processo essencial para compartilhar seu código com outras pessoas desenvolvedoras.

Aqui estão os passos básicos para criar um repositório:

1. Acesse sua conta: certifique-se de que você fez login na conta do GitHub. Se você não tiver uma conta, siga as etapas para criar uma.

2. Página inicial: na página inicial do GitHub, clique no botão "New" (Novo), que pode estar localizado no canto superior esquerdo.

3. Nome e descrição: preencha o nome do seu repositório e uma breve descrição. Escolha se deseja que o repositório seja público (visível para todas as pessoas) ou privado (acessível apenas por convite).

4. Opções de inicialização: você pode optar por inicializar o repositório com um arquivo README, que é uma boa prática para fornecer informações sobre o projeto. Além disso, você pode escolher uma licença para o seu código, se desejar. Para este primeiro tutorial, recomendamos que você inicie um repositório vazio, então deixe a opção `Add a README file` desmarcada e `License: None`. Você poderá adicionar o README e um arquivo de licença depois.

5 `.gitignore`: você pode especificar tipos de arquivos que o Git deve ignorar ao rastrear alterações. Por exemplo, você pode selecionar uma linguagem de programação específica para gerar um arquivo `.gitignore` correspondente. Neste momento deixe essa opção marcada como `.gitignore template: None`. Você também poderá adicionar esse arquivo depois.

6 Create Repository: Após preencher todas as informações necessárias, clique no botão "Create repository" (Criar repositório) para criar o seu repositório.

Depois disso, seu repositório está pronto!

Você pode começar a adicionar arquivos, fazer commits e colaborar com outras pessoas.

Repositórios remotos e locais

No contexto do controle de versão com o Git, é importante entender a diferença entre repositórios locais e repositórios remotos.

Repositórios Locais: um repositório local é a cópia do seu projeto que reside no seu computador. É onde você faz as alterações, cria commits e mantém o histórico do projeto. Você pode trabalhar offline em um repositório local sem necessidade de conexão com a internet.

Repositórios Remotos: um repositório remoto é uma versão do seu projeto hospedada em um servidor na web, como o GitHub. Eles são usados para salvar e versionar as alterações de código, compartilhar seu projeto com a comunidade dev e colaborar em projetos de outras pessoas. Você pode enviar (push) as alterações do seu repositório local para o repositório remoto e também obter (pull) as alterações feitas por outras pessoas colaboradoras.

Ao criar um repositório no GitHub, você está criando um repositório remoto onde seu código será hospedado e compartilhado com outras pessoas.

Lembre-se de sincronizar regularmente seu repositório local com o repositório remoto para manter todas as pessoas colaboradoras atualizadas.

Como baixar e instalar o Git?

Antes de utilizarmos os recursos do GitHub localmente, você precisa instalar o Git em seu computador.

Mas qual é a diferença entre **Git** e **GitHub?**

- **Git** é atualmente o mais famoso e mais utilizado sistema de controle de versão (VCS ou Version Control System), embora existam outros. Estes sistemas são utilizados justamente para manter um controle sobre a versão do software que está sendo desenvolvida, testada e em produção, mantendo um histórico organizado que permita, por exemplo, que o sistema "volte" para uma versão anterior em caso de problemas.
  É o Git (e não o GitHub) que funciona localmente em seu computador, fazendo esse "rastreio" das alterações de código. Todos os comandos que utilizamos no terminal também são comandos do Git.

- Já o **GitHub** é uma das plataformas utilizadas pela comunidade e pelas empresas para hospedar códigos versionados com o Git. Existem outras, como o GitLab e o [BitBucket](https://bitbucket.org/), porém o GitHub se tornou a mais famosa delas e também a mais utilizada, devido à abordagem voltada para o compartilhamento e colaboração e a uma grande quantidade de recursos úteis no dia a dia do desenvolvimento de software.

Para utilizar o Git, você pode seguir esse passo a passo de como fazer o download e instalar, para cada sistema operacional:

Windows:

1. Acesse o site oficial do Git em "https://git-scm.com/download/win".
2. Clique no link para download do Git para Windows.
3. Após o download, execute o instalador.
4. Siga as instruções do instalador, aceitando as configurações padrão, se não for um usuário avançado.
5. Conclua a instalação.

Linux:

1. No Linux, você pode instalar o Git usando o gerenciador de pacotes da sua distribuição. Por exemplo, no Ubuntu, use o comando `sudo apt-get install git`.
2. Se estiver usando outra distribuição, substitua o comando de acordo.

macOS:

1. No macOS, o Git pode ser instalado de várias maneiras, incluindo o uso do Xcode Command Line Tools, que geralmente já está instalado no sistema.
2. Abra o Terminal e digite `git --version` para verificar se o Git está disponível. Se não estiver, o sistema solicitará a instalação.
3. Siga as instruções para instalar o Git. Com esses passos simples, você pode instalar o Git no seu sistema operacional e começar a usar essa poderosa ferramenta de controle de versão.

Configurando seu username e email no Git:

Após concluir a instalação, você precisa configurar o nome e email que será associado aos seus envios usando Git e GitHub.

Insira no terminal os comandos:

`git config --global user.name "Seu nome"`
`git config --global user.email "seu@email"` (insira o email utilizado no cadastro)

Como linkar os repositórios remoto e local?

Depois de criar um repositório no GitHub e instalar o Git, é essencial conectar seu repositório local a ele para que você possa enviar suas alterações para o repositório remoto. Aqui está um passo a passo, inspirado no tutorial fornecido pelo GitHub, assim que você clica em “Create Repository”:

- **Abra o terminal:** Se estiver usando um sistema Unix (Linux ou macOS), abra o terminal de comandos, já caso estiver utilizando Windows, abra o Git Bash no Windows. O GitBash terá sido instalado junto com o Git.

- **Navegue até o diretório do projeto:** Use o comando cd <caminho/do/seu/repositorio> para navegar até o diretório do seu projeto local. Ou, caso prefira, você pode acessar normalmente a pasta do projeto e clicar com o botão direito do mouse > “Abrir com GitBash” (Windows) / “Abrir no terminal” (Linux).

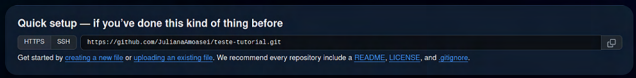
- **Inicie um repositório Git Local:** Se o seu projeto ainda não é um repositório Git, use o comando `git init` no GitBash/terminal para iniciá-lo. Você deve receber de volta a mensagem `Initialized empty Git repository in /<local da pasta do projeto>/.git/`, o que significa que a operação deu certo e agora você tem uma pasta “oculta” (porém ainda vazia) chamada `.git` dentro da pasta do seu projeto. É através dessa pasta que o Git fará o monitoramento local das alterações no seu código.

- **Adicione o remote:** Use o comando `git remote add origin <URL-do-Repositório>` para adicionar o repositório remoto como um "remote" chamado "origin".

Exemplo:

```
git remote add origin https://github.com/seu-usuario/seu-repositorio.git
```

Como descobrir qual é a URL do repositório no GitHub?
Se você já criou o seu repositório com as instruções que demos anteriormente, a página do seu repositório deverá ter uma seção parecida com essa:



Recorte de tela de uma seção de um repositório GitHub com instruções para configuração rápida. No topo, há um texto sugerindo que essas instruções são para quem já está familiarizado com o processo. Logo abaixo, há duas opções de protocolo para clonar o repositório: HTTPS e SSH. Ao lado, há uma URL de um repositório chamado "teste-tutorial".

Há também recomendações para criar ou fazer upload de um arquivo existente e um lembrete de incluir arquivos importantes como README, LICENSE e .gitignore no repositório.

- O endereço **HTTP** `https://github.com/<seuperfil>/<nome-do-repositorio>.git` é a URL do seu repositório! É através desse endereço que você vai conectar o repositório local (em seu computador) com o remoto (no GitHub).

- Se você executou o comando acima e não recebeu nenhuma mensagem como resposta no terminal, não se preocupe. É normal que vários comandos, quando executados com sucesso, não retornem nenhuma mensagem.

Como enviar código para o repositório remoto?

Depois que seus repositórios, local e remoto, estão vinculados, você pode começar a trabalhar em seu projeto e enviar seus arquivos para o repositório remoto:

- Crie um arquivo README (opcional): Se ainda não tiver um arquivo para capa do seu repositório, que explique o que é o projeto, suas funcionalidades, pré-requisitos etc., você pode criar um nesta etapa.

  Para criar o arquivo você pode iniciar com o seguinte comando no GitBash/Terminal (confira antes se você abriu o terminal dentro da pasta do projeto):

```
echo "# Meu Projeto" >> README.md
```

  E para saber mais informações sobre como escrevê-lo, confira nosso artigo Como escrever um README incrível no seu Github

- Adicione e faça o commit: No terminal, use os comandos `git add` e `git commit` conforme abaixo para adicionar e confirmar as alterações.

```
git add README.md
git commit -m "Adicionando arquivo README"
```

- No Git o código é separado por “branches” (ramos). Por exemplo, “main” (principal, onde está a versão mais atualizada do código), “develop” (ramo de desenvolvimento), etc. Nesse momento você precisa definir o nome da branch para onde vai enviar o código, e recomendamos que seja a branch padrão “main”. Use o comando `git branch -M main` no terminal para definir isso.

```
git branch -M main
```

Envie para o repositório remoto: Use o comando `git push -u origin main` para enviar o arquivo para o repositório remoto.

```
git push -u origin main
```

IMPORTANTE:

Da primeira vez que você executar o comando `push` é possível que o Git peça as suas credenciais de conta do GitHub para prosseguir:

```
Username for 'https://github.com': <insira aqui seu username>
Password for 'https://<seunome>@github.com': <insira sua senha>
```

Se você receber o seguinte aviso no terminal, ao tentar usar o comando `push`:

remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/<seunome>/<nome-repositorio>.git/'
```

Significa que precisaremos de um passo a mais para usar o GitHub, devido a mudanças nas políticas de autenticação. A senha (password) que você criou junto com a sua conta do GitHub não poderá mais ser usada neste processo de enviar e receber códigos para repositórios. No lugar dela, você precisa configurar um token de autenticação.

**Mas não se preocupe, siga esse TUTORIAL**
para entender o que é o token e fazer a configuração necessária!

Após concluir a configuração da chave, basta repetir a operação `git push -u origin main` e você deve receber de volta a mensagem de sucesso:
```
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 235 bytes | 235.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:<seunome>/<nome-repositorio>.git
 * [new branch]   main → main
```
```

Dica: Todos esses comandos estão disponíveis com os campos preenchidos na tela inicial do seu repositório no Github:

Recorte de tela do GitHub com instruções detalhadas para a configuração rápida de um repositório GitHub. A primeira seção mostra a URL do repositório. Abaixo, três seções explicam como: criar um novo repositório a partir da linha de comando, enviar um repositório já existente para o GitHub, e importar código de outros sistemas de controle de versão, como Subversion ou Mercurial. Cada seção traz comandos que guiam a pessoa a realizar o versionamento e publicação de código no repositório remoto.

Agora, os seus arquivos e códigos locais estão refletidos no repositório remoto no GitHub. Este processo é crucial para a colaboração em equipe e para manter um histórico centralizado do seu projeto. Lembre-se de adaptar as URLs e nomes do repositório conforme necessário.

Como baixar um repositório do GitHub?

Além de subir um projeto no Github, você também pode fazer o download de repositórios.

Baixar um repositório do GitHub permite que você tenha uma cópia local do código em seu próprio computador. Este processo é conhecido como "clonar" um repositório.

Confira um guia passo a passo para baixar um repositório do GitHub para o seu ambiente local aqui no nosso artigo Clonando um repositório com Git e GitHub.

Lembrando que esse processo é fundamental para contribuir com projetos de código aberto, colaborar em equipes e realizar desenvolvimento local.

Como baixar novos commits do repositório remoto?

Se você estiver trabalhando em uma equipe com o GitHub, diversas versões podem ter atualizações diferentes entre si a todo momento.

Nesse caso, é fundamental ficar por dentro de todas as alterações feitas por outras pessoas colaboradoras e, para ter esse controle de alterações, são feitos “commits”.

Mas, afinal de contas, o que é um “commit”?

O verbo “commit” pode significar várias coisas, entre “comprometer-se”, “despachar/enviar”, entre outros. No versionamento de código, chamamos de “commit” o conjunto das últimas alterações feitas no projeto e que foram previamente “adicionadas” (através do comando `git add`) para serem salvas.

É como se você criasse um “save point” em um jogo, para onde pode retornar depois caso precise. O commit salva o estado atual de seu projeto e dá a ele um identificador único. É dessa forma que podemos seguir a “história” de um projeto, entendendo quais alterações foram feitas, por quem e em que ponto do desenvolvimento. E também para que ponto devemos voltar, se for necessário.

Ao trabalharmos em projetos compartilhados, é normal passarmos pelo processo de atualizar nosso repositório local com alterações feitas por outras pessoas.

Por isso, aqui está um passo a passo para baixar novos commits de um repositório remoto para o seu repositório local, mantendo-o atualizado:

- Abra o terminal ou prompt de comando: Semelhante aos outros passos, vamos fazer os comandos no terminal (Linux, macOS) ou GitBash (Windows).
- Navegue até o diretório do repositório local: Use o comando cd para navegar até o diretório do seu repositório local.

```
cd <caminho/do/seu/repositorio>
```

- Atualize o repositório local com os novos commits: Utilize o comando git pull para buscar os novos commits do repositório remoto e atualizar sua branch local.

```
git pull origin nome-da-branch
```

- Se você estiver na branch principal (por exemplo, "main"), pode simplesmente usar:

```
git pull origin main
```

Isso trará as últimas alterações feitas por outros colaboradores para o seu repositório local.

- Resolva conflitos (se aplicável): Se ocorrerem conflitos durante o processo de atualização, o Git notificará você. Nesse caso, será necessário resolver os conflitos manualmente antes de continuar. As IDEs (Ambientes de desenvolvimento integrado) ou o próprio GitHub (recomendado) oferecem uma visualização otimizada de onde estão os conflitos para que você possa resolver.

- Verifique as alterações locais: Após o git pull, você pode verificar as alterações locais usando git log para visualizar os novos commits no histórico do seu repositório.

```
git log
```

Assim, você terá seu repositório local atualizado com os últimos commits do repositório remoto.

Esse processo é importante para manter a sincronização entre o seu ambiente de desenvolvimento e as contribuições feitas por outras pessoas da equipe.

Como separar o desenvolvimento de diferentes funcionalidades?

Em uma equipe de desenvolvimento, os integrantes trabalham em diferentes funcionalidades do projeto, por exemplo, você pode estar trabalhando na página Home, enquanto outra pessoa está desenvolvendo o rodapé — esse é, inclusive, um modelo de trabalho bastante usual.

Ou seja, ao trabalhar em projetos de software, é comum ter várias funcionalidades em desenvolvimento simultâneo.

O Git trabalha com as chamadas “branches” (ramos), onde é possível isolar diferentes cópias de um mesmo projeto e alterá-las sem que isso interfira com, por exemplo, a branch principal onde está hospedada a versão “final” do projeto.

Cada branch pode representar uma linha independente de desenvolvimento. Por exemplo, você pode ter uma branch para desenvolver uma nova funcionalidade, outra para corrigir um bug e assim por diante.

- Comando para criar uma nova branch para uma nova funcionalidade:

```
git branch nova-funcionalidade
```

- Mudar para a nova branch:

```
git checkout nova-funcionalidade
ou
git switch nova-funcionalidade
```

Importante: quando você inicia uma nova branch, o código da sua branch atual é transferido para a nova. Por exemplo, se você está na branch `main` e cria uma branch `nova-funcionalidade`, ela serácriada com uma cópia de todos os arquivos da branch `main`.

E qual a melhor forma de separar o trabalho de um time em branches? Existem algumas estratégias diferentes, cada uma com seus prós e contras:

Git Flow **GitHub Flow** **GitLab Flow**

Mesmo trabalhando em seus projetos solo, você ainda pode se beneficiar da separação em branches. Uma forma de organização simples que você pode praticar em seus projetos é a organização das branches por “feature” (funcionalidade).

É comum a utilização de branches específicas para cada funcionalidade que está sendo desenvolvida. Isso mantém as alterações referentes a uma funcionalidade isoladas de outras partes do código.

Por exemplo, use o comando abaixo para ao mesmo tempo criar uma nova branch e já mover seu código “ativo” para lá:

```
git checkout -b nova-funcionalidade
```

Git Merge: Após completar o desenvolvimento em uma branch de funcionalidade, você pode mesclar as alterações de volta para a branch principal (por exemplo, "main"). Isso integra a nova funcionalidade ao código principal.

- Mudar seu código ativo de volta para a branch principal

```
git checkout main
ou
git switch main
```

- Em seguida, mescle as alterações da feature de volta para a branch principal

```
git merge nova-funcionalidade
```

Com isso, o novo código desenvolvido na branch `nova-funcionalidade` será adicionado ao código que já estava na branch `main`, e ambas estarão atualizadas.

- Para exibir todas as branches existentes no seu projeto

```
git branch
```

O uso de branches é uma estratégia simples e muito útil para que você possa testar novas funcionalidades sem perder nem interferir com a última versão funcional de seu código, que normalmente está hospedada na branch `main`. Os códigos só serão incorporados se você escolher fazer isso através do comando `merge`.

Fork/Pull Request

O processo de fork e pull request é fundamental para contribuições em projetos de código aberto, permitindo que outras pessoas sugiram alterações sem afetar diretamente o repositório original.

Quando você quiser colaborar ou utilizar um projeto open source, a primeira coisa a fazer é copiar este projeto para a sua conta do GitHub, e em seguida trazer o conteúdo do repositório remoto no GitHub para o local em seu computador. Esse processo utiliza os comandos `fork` e `clone`.

Fork: Na página do projeto no GitHub, clique no botão "Fork" no canto superior direito da página. Isso criará uma cópia do repositório original na sua conta.

Clone: Utilize o comando abaixo para “clonar” o conteúdo do projeto que está no repositório remoto (GitHub) para o repositório local (seu computador):

```
git clone https://github.com/seu-usuario/nome-do-fork.git
```

Esse comando irá criar uma pasta em seu computador com o mesmo nome do repositório. Não esqueça de abrir o terminal no diretório correto (por exemplo, `Meus Documentos`) antes de executar o comando; o clone será feito dentro desse diretório.

Em seguida, é uma boa prática você criar uma branch para armazenar suas alterações, sem prejudicar o andamento do projeto principal, dessa forma:

```
git checkout -b feature/minha-contribuicao
```

Realize as alterações desejadas, faça commits com mensagens explicativas sobre as alterações e, se necessário, crie novas branches para diferentes funcionalidades ou correções.

Feito isso, para enviar as alterações para seu repositório com a versão do fork, faça:

```
git push origin feature/minha-contribuicao
```

Nesse momento, a sua cópia (fork) do projeto terá uma nova branch com as alterações. Porém, como enviar a sua contribuição para o repositório do projeto original?

A estratégia de “pull request” é muito utilizada em projetos open source para que possam receber sugestões de alteração, melhoramentos ou correções de bug, pois permite que o código enviado seja revisado e aprovado pela pessoa ou pessoas que mantêm o projeto antes de ser incorporado ao restante. Também é utilizado em empresas para organizar a revisão de código dentro dos times.

Como fazer um pull request?

No GitHub, vá até o seu fork e clique em “Pull requests” no menu superior, e em seguida no botão "New Pull Request" à direita da tela. Escolha a branch com suas alterações e sugira a incorporação ao repositório original.

Os colaboradores do projeto original vão revisar suas alterações por meio do pull request, onde será possível abrir campos de discussão para cada linha do código, tirar dúvidas e fazer novas sugestões. Se tudo estiver em ordem, eles poderão mesclar suas alterações no projeto principal.

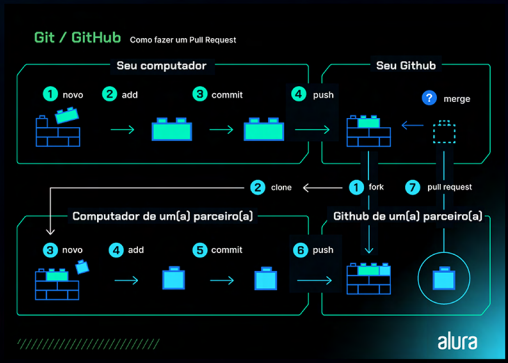


Diagrama explicativo de como fazer um Pull Request no Git e GitHub, ilustrando o fluxo de trabalho entre seu computador, o GitHub, e o computador de outra pessoa.

Utilizar forks e pull requests promove uma colaboração estruturada, permitindo contribuições externas enquanto mantém o controle sobre o repositório original.

Esse fluxo é fundamental em projetos de código aberto e ambientes colaborativos.

Integração com IDEs

Como já escrevemos código em Ambientes de Desenvolvimento Integrado (IDEs) como o Visual Studio Code, o Sublime Text, entre outros, integrar todos esses passos com Git e GitHub simplifica significativamente o ciclo de vida do desenvolvimento de software.

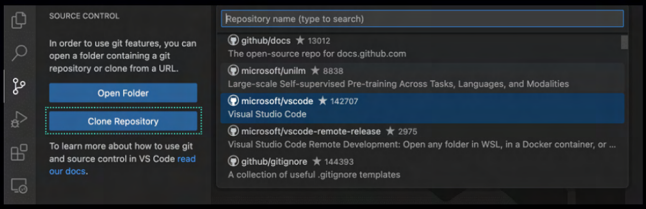
Vamos explorar como essa integração pode ser feita em IDEs populares.

Visual Studio Code (VSCode):

1. Uma vez que o Git já esteja instalado e configurado em seu computador, acesse a aba "Source Control" no menu lateral à esquerda da tela.

2. Caso seja a primeira vez que utiliza a ferramenta de controle de versão do Visual Studio Code, pode ser solicitado um processo para vincular a autenticação do GitHub e autorizar essa integração com o Visual Studio Code, pois esse processo não é automático. Siga os passos indicados na tela pela ferramenta.

3. Clonando um repositório:
Você pode usar a opção de clonagem no VSCode para copiar um repositório do GitHub para o seu ambiente de desenvolvimento.



Interface do Visual Studio Code exibindo a opção de clonar um repositório GitHub.

4. Controle de versão direto no editor:
Tendo o repositório Git no VSCode, você terá uma interface gráfica para operações do Git, como commit, push, pull e merge, facilitando o controle de versão diretamente no editor.

Outras IDEs:

- **Eclipse:**
  Para o Eclipse, você pode utilizar o plugin EGit para integrar o Git. Configure suas credenciais do GitHub e clone repositórios diretamente do ambiente de desenvolvimento.

- **IntelliJ IDEA:**
  Assim como o VSCode, o IntelliJ IDEA possui suporte nativo ao Git. Configure o GitHub nas configurações do Git e clone repositórios usando a interface gráfica.

- **Visual Studio (VS):**
  O Visual Studio também oferece integração direta com o Git. Portanto, da mesma forma como citado para o VSCode, configure suas credenciais e clone repositórios do GitHub sem sair do ambiente de desenvolvimento.

Além dessas interfaces gráficas, essas IDEs também tem um terminal de comandos incluso, tornando o controle de versão e a colaboração em equipe mais acessíveis diretamente no ambiente de codificação.

Github Desktop: usar Git sem a linha de comandos

Outra opção de utilização do Git e Github, sem utilizar o terminal de comandos ou extensões nas IDEs, é o Github Desktop, que possui uma interface gráfica específica como um "sincronizador de código".

Assim, é possível facilitar as visualizações, o envio e o recebimento das modificações, além dos famosos conflitos de merge, que você não precisa se preocupar nesse primeiro instante.

Nesse vídeo, o Felipe, da Alura, te explica como dar os passos no Github Desktop: GitHub sem linhas de comando | #AluraMais

Além do controle de versão: outras funcionalidades do GitHub

O Github é bastante popular, tanto que é apelidado de rede social da comunidade dev, pois além de oferecer os recursos de controle de versão, também é comumente utilizado para compartilhar código e, até mesmo, como um portifólio.

Além disso, na plataforma você consegue baixar projetos abertos pela comunidade por meio do git clone, em que você pode ver passo a passo no artigo Clonando um repositório com Git e GitHub.

Projetos de código aberto, ou open source, frequentemente envolvem colaboração de devs de todo o mundo. Para isso, existem eventos como o Hacktoberfest para incentivar a contribuição da comunidade.

Para garantir a contribuição harmoniosa de diversas pessoas colaboradoras, é comum usar o Github com práticas de ramificação e de colaboração, como diretrizes para criar forks (bifurcações) que é uma cópia independente do repositório de código-fonte.

Além da convenção de abrir issues (problemas) para discussão e fazer pull requests (sugestão de alteração) para propor alterações e revisões de código rigorosas para garantir a qualidade do código.

Mas o Github vai muito além disso: essa tecnologia possui diversas outras ferramentas que otimizam nosso trabalho. Confira na sequência:

GitHub Actions: Automatização de fluxos de trabalho

Com o Github Actions, é possível automatizar, personalizar e executar fluxos de trabalho no nosso repositório no próprio Github.

Então, por exemplo, qualquer teste que você precise executar a cada alteração no projeto, você consegue fazer de forma automatizada com o Github Actions.

GitHub Pages: Hospedagem de sites estáticos

É possível compartilhar um site estático feito com HTML, CSS e Javascript por meio do Github Pages, que usa os arquivos diretamente do seu repositório, executa os arquivos, publica um site e fornece um link automático, que pode inclusive ser vinculado a um domínio. Para saber mais como é feito esse processo, confira o artigo Como colocar seu projeto no ar com o Github Pages?.

GitHub Codespaces: Ambiente de desenvolvimento online

O GitHub Codespaces oferece um ambiente de desenvolvimento online e hospedado na nuvem.

Com essa ferramenta, pessoas desenvolvedoras podem acessar um ambiente de desenvolvimento funcional diretamente no navegador, eliminando a necessidade de configurações locais complexas.

Além disso, oferece integração perfeita com o GitHub, permitindo que devs acessem seus repositórios, issues e pull requests diretamente do ambiente de desenvolvimento.

E para acessar esse ambiente, basta você acessar um repositório no Github e apertar a tecla . (ponto final), para abrir automaticamente o ambiente do Codespaces.

GitHub Discussions: Comunicação da comunidade

O GitHub Discussions é uma plataforma de comunicação colaborativa destinada à comunidade que se reúne em torno de um projeto, seja ele de código aberto ou interno.

Dentro desse espaço, as pessoas da comunidade têm a oportunidade de fazer perguntas, fornecer respostas, compartilhar atualizações e realizar discussões abertas, permitindo o acompanhamento de decisões que impactam o funcionamento coletivo da comunidade.

GitHub Security: Ferramentas de segurança integradas

O GitHub Security engloba um conjunto de ferramentas integradas voltadas para a segurança dos projetos.

Essas ferramentas abrangem análise de código, verificações de segurança e testes para verificar se as dependências do código estão seguras, com o Dependabot.

Elas trabalham juntas para manter o código seguro e atualizado, proporcionando uma camada de proteção e detecção de vulnerabilidades diretamente na plataforma do GitHub.

GitHub Insights: Análise e estatísticas de repositórios

O GitHub Insights é uma funcionalidade que oferece uma variedade de análises e estatísticas relacionadas aos repositórios hospedados no GitHub.

Com ela, você pode obter informações valiosas sobre o desempenho do seu projeto, incluindo métricas de contribuição, atividade da comunidade e tráfego do repositório.

Essas estatísticas proporcionam uma visão detalhada sobre como o seu projeto está sendo utilizado e como a comunidade está interagindo com o seu código, auxiliando na avaliação e no aprimoramento do desenvolvimento e da colaboração.

Mais referências e conteúdos

Na websérie Git e GitHub para sobrevivência, Mário Souto, conhecido como DevSoutinho, traz importantes pontos do uso dessa plataforma:

- **Git e Github para Sobrevivência #01: Como o Git funciona?**

  O segundo episódio “Como funciona o merge?” vai te ajudar a entender os branches, a master e como juntamos o trabalho de diversas pessoas e equipes:

- **Git e Github para Sobrevivência #02: Como o merge funciona?**

  Se você quer ouvir um guia para iniciantes em Git e Github, vai gostar de ouvir o episódio

- **Guia do Iniciante em Github do Hipsters #184.**

  Ele é um episódio especial para quem está começando na área de tecnologia e precisa entender o que é, para que serve e como dar os seus primeiros passos com essa ferramenta.

Além disso, você também pode conferir o artigo tutorial sobre começar com Git, aprenda a versionar para você fazer seus primeiros commits. Sem contar nosso principal curso de Git e Github que é extremamente elogiado.

E a partir daí, você pode entrar em merges e branches. Enquanto isso, experimente fazer os primeiros pushes e pulls, sincronizando com o Github Desktop. É um excelente caminho.

Considerações finais

Sem dúvidas, o GitHub se solidificou em uma posição fundamental no universo do desenvolvimento de software e na colaboração entre equipes de pessoas programadoras. Hoje, essa tecnologia está presente em inúmeros projetos.

A flexibilidade e facilidade de uso têm sido essenciais para dar maior agilidade ao desenvolvimento de software.

Afinal de contas, melhora a colaboração entre pessoas desenvolvedoras e garante a organização e a segurança de versões de código.

alura

GUIA DO MOCHILEIRO TECH

Continue o seu mergulho

Agora que você já se familiarizou com os principais recursos do GitHub, que tal continuar desbravando o **Guia do Mochileiro Tech?**

Eu tenho certeza que há **muitos outros conteúdos** que vão potencializar, ainda mais, a sua jornada.

Me leve para o

Guia do Mochileiro Tech