

O GITHUB COMO REPOSITÓRIO E VERSIONAMENTO

Uma empresa que recruta acadêmicos para ingressarem em atividade de estágio me trouxe uma inquietação: os alunos que estavam em período possível para estágio não sabiam trabalhar de maneira colaborativa.

Isso me trouxe uma certa angústia, pois realmente os alunos não viam este conteúdo em nosso curso de Ciência da Computação. Certamente, em um curso de tecnologia este conteúdo, em forma de ferramenta, poderia ser trabalhado, mas em CC, por ser generalista, uma visão maior, menos especializada, não havia espaço para este conteúdo específico, mas era algo necessário.

Decidi então trazer para uma disciplina específica que ministro e o resultado foi imensamente positivo. Os alunos gostaram muito e aos poucos foram desbravando e descobrindo os pontos positivos do desenvolvimento colaborativo.

Com este pequeno laboratório e com o estudo em Flutter e o projeto deste livro, pensei: vou anexar ao material do livro este

conteúdo, mesmo que de maneira básica e introdutória. E aqui estamos! Vamos torcer para que você goste e possa se aprofundar após a introdução que trabalharemos aqui.

3.1 MAS, O QUE É O GITHUB?

Muitos portais, livros, artigos e diversos tipos de conteúdo trazem o GitHub como sendo um sistema focado em gerenciamento de projetos e versões de códigos. Alguns se arriscam em dizer que é uma rede social para desenvolvedores, e eu concordo.

O GitHub nos propicia o desenvolvimento colaborativo de projetos, o que significa que podemos ter uma equipe, pequena ou grande, trabalhando em um único projeto, com código centralizado. Em prática, podemos dizer que quando um desenvolvedor for implementar algo neste projeto, ele traz para sua máquina (pull) o projeto, implementa o que é necessário e devolve (push) para o repositório (GitHub no caso).

Quando um desenvolvedor realiza implementações no projeto, em sua máquina local e então submete novamente ao GitHub, ele realiza um processo chamado de `commit`, onde registrará tudo que foi feito desde o `pull` até o `push`. Com isso, fica fácil a equipe identificar o que foi feito, de maneira textual. Lembrando que após o `commit`, um `push` é necessário.

Se a equipe precisar, ainda, identificar no código as alterações realizadas, isso também se torna muito simples, pois o GitHub oferece mecanismos para esta finalidade.

Não entraremos em um aprofundamento em versionamento

ou no próprio GitHub. Nosso objetivo aqui é despertar em você uma centelha de curiosidade, para que você possa perceber a importância de uma tecnologia como o GitHub em seu processo de desenvolvimento de softwares.

A título de curiosidade, é comum também a utilização do GitHub como repositório de documentos colaborativos, ou seja, um uso além do desenvolvimento de software. Entretanto, caso você queira comparar apenas as mudanças entre duas versões de arquivos, eu posso recomendar o `DiffChecker`, em <https://www.diffchecker.com/diff>. É bem prático e útil.

3.2 E O GIT, O QUE É?

Já sabemos que o GitHub é focado em projetos, repositórios, uma rede, por onde é possível uma equipe trabalhar de maneira colaborativa. Mas o GitHub não é só. Ele precisa de um cara chamado `Git`. Algumas referências o chamam de `Kernel`, área central e até de coração do GitHub. O Git é o sistema de controle de versão que está por trás do GitHub.

Como dito anteriormente, talvez em outras palavras, sempre que desenvolvemos, mesmo após concluirmos o projeto, ele pode sofrer alterações, sejam elas evolutivas ou corretivas. Com o Git, estas atualizações não sobrescrevem o código que já temos armazenados, elas serão gravadas "em separado", permitindo que, em caso de problemas, as anteriores possam ser revertidas como atual.

Para que possamos trabalhar com o uso do Git e GitHub em nossos projetos do livro, precisaremos ter o Git instalado em

nossos equipamentos. Para isso, acesse <https://git-scm.com/downloads> e siga as orientações do portal para o download e instalação da ferramenta. São passos bem tranquilos.

Caso você esteja em uma rede com um proxy, pode ser que haja a necessidade de realizar algumas configurações para que o Git funcione perfeitamente. Se for este o seu caso, veja as orientações em <https://gist.github.com/evantoli/f8c23a37eb3558ab8765>.

3.3 REPOSITÓRIO, BRANCH, PULL, FORK, COMMIT E PUSH

Quando trabalhamos com o Git e GitHub, ou qualquer outra ferramenta com a mesma finalidade, nos deparamos com termos específicos às tecnologias adotadas e, nesta seção, buscaremos descrevê-los.

Repositório (ou repo) é o local (diretório) onde os arquivos de nossos projetos ficam armazenados. Este local pode ser remoto, não necessariamente outro lugar geográfico, mas sim um local diferente de sua máquina de desenvolvimento, pode ser um servidor disponibilizado por seu local de trabalho. Há também o repositório local, aí sim, sua máquina. Normalmente, nossa aplicação possui os dois repositórios, o remoto, onde o projeto fica centralizado e o local, onde nós clonaremos o projeto para nossas alterações, implementações, correções, ou qualquer tipo de atividade em nossos arquivos.

Branch (ou ramificação) é o termo utilizado para nos referirmos a uma cópia de nosso diretório referente ao repositório

remoto. Podemos ter um branch para desenvolvermos isoladamente, não de maneira colaborativa. Podemos pensar nessa situação quando queremos realizar testes no código já disponibilizado e não nos preocupar com um commit errado. Entretanto, é possível, ao finalizarmos nossas alterações, combinarmos o nosso branch com o branch (ou os branches), por meio de uma operação de merge. Normalmente isso pode ser feito por um Pull Request .

Realizar um Pull Request é o processo de informar a equipe que você implementará as mudanças criadas no seu branch ao repositório master. Os colaboradores do projeto poderão ou não aceitar esta solicitação.

Fork é uma operação onde criamos um novo projeto, tendo em base um repositório já existente. Pode-se dizer que a operação de forking é uma cópia de um repositório em seu atual estado. Isso permite que você adapte o projeto, melhore-o, corrija-o e então o distribua como um outro projeto, em outro repositório, respeitando aqui todos os direitos legais do copiado, é claro.

Temos ainda os termos Commit e Push . Vamos a eles? Sempre que precisarmos atualizar um projeto/repositório, precisamos realizar um Pull , onde traremos o repositório remoto para o local. Implementamos o que for preciso, e realizamos um Commit , dando um título para a manutenção realizada e podendo gerar uma descrição do que foi feito. Podemos ter vários commits locais. Após concluirmos, precisamos enviar estas alterações para nosso repositório remoto e realizamos isso em uma operação de Push .

Todos os termos e operações que vimos nesta seção, podem ser

realizados via linha de comando de nosso terminal, ou por meio de ferramentas, como o GitHub Desktop, ou nosso ambiente de desenvolvimento, IDE, como o Android Studio, que utilizaremos no livro, ou o Visual Studio Code, ferramenta também muito utilizada.

Não vamos nos estender muito na teoria do Git e GitHub, pois não é o foco principal de nosso trabalho, mas tem dois links bem legais que recomendo que você leia, caso queira, para ter uma noção de tudo isso que vimos aqui: https://rogerdudler.github.io/git-guide/index.pt_BR.html e <https://blog.dankicode.com/introducao-ao-git-e-github/>.

3.4 NOSSO PROJETO E REPOSITÓRIO NO GITHUB

Muito bem! Já temos uma base mínima de conhecimento sobre o Git, GitHub e o que oferecem. Vamos pôr a mão na massa agora. Acesse <https://github.com/>, crie um usuário, caso ainda não tenha e, após ter acessado a plataforma, localize uma área da tela, que aponta os repositórios que já temos em nosso perfil e, ao lado dele, possivelmente verá um botão com o texto `New`. As interfaces podem mudar com o tempo, mas isso é mais ou menos o que temos na figura a seguir.

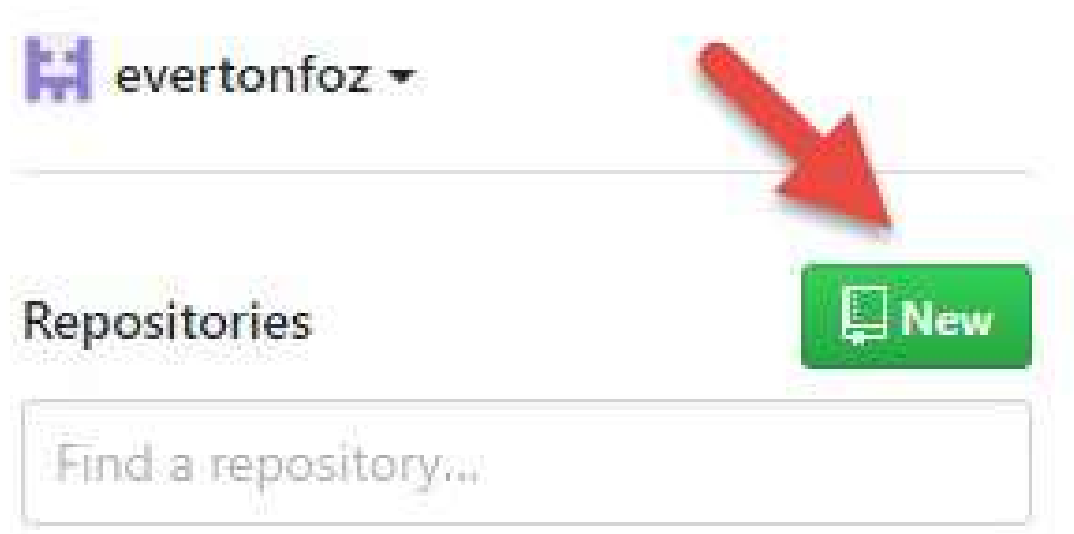


Figura 3.1: Criação de um novo repositório

Ao clicar no botão apontado anteriormente, seremos redirecionados a uma página para informação do nome para o repositório e mais algumas informações, que têm objetivos semânticos ao que é solicitado. Ao informar todos os dados, clique no botão responsável pela criação do repositório.

Quando o repositório for criado, você será direcionado a ele. Veja a figura a seguir, que ilustra essa situação.

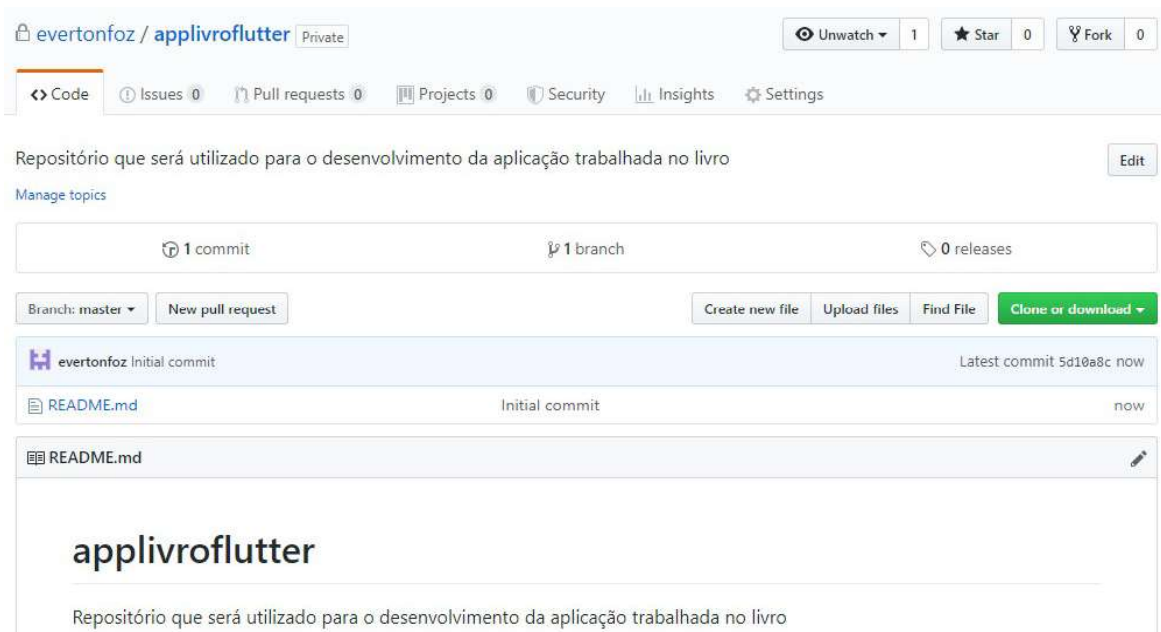


Figura 3.2: Página do repositório criado

O GitHub pode trabalhar tranquilamente apenas com repositórios. Poderíamos ter um repositório pura e simplesmente para hospedar nosso projeto que criaremos no livro. Entretanto, ele oferece a possibilidade de termos projetos criados nele e faremos isso. Veja em sua página, que deve ser semelhante à figura anterior, que temos um acesso à `Projects`. Vamos clicar nesta opção. Você certamente verá uma página semelhante à apresentada na figura a seguir.

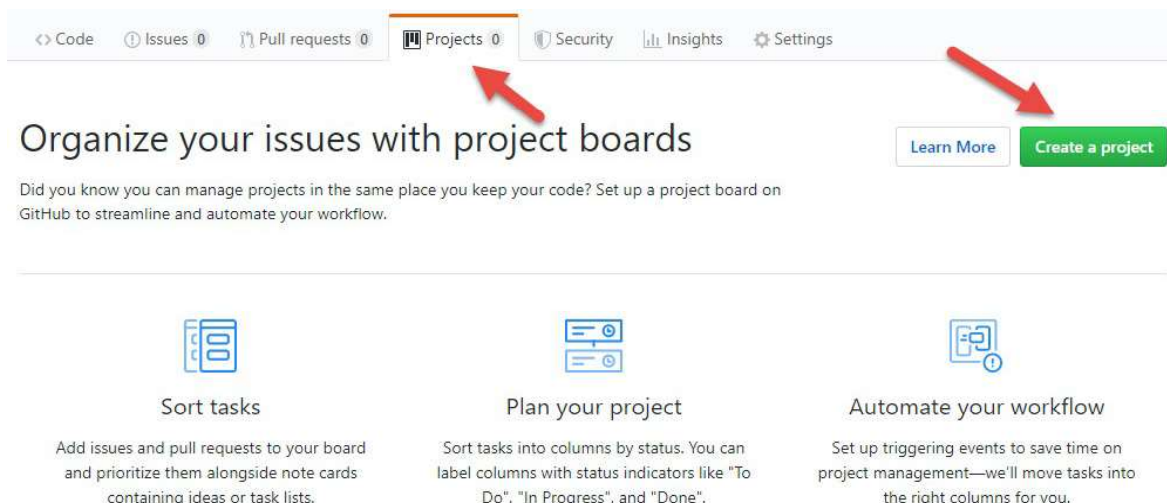


Figura 3.3: Página de projetos relacionados ao repositório

Nessa nova interface, precisamos informar o nome do projeto, sua descrição e qual template será utilizado em sua criação. Recomendo que verifique as opções e dê uma estudada sobre elas. No projeto do livro não utilizaremos nenhum template, vamos criar tudo que julgarmos necessário. Veja a figura desta interface a seguir.

Create a new project

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

Project board name

jogo_da_forca

Description (optional)

Projeto relacionado ao Jogo da Forca, aplicativo que será desenvolvido no livro sobre flutter

Project template

Save yourself time with a pre-configured project board template.

Template: None ▾

Create project

Figura 3.4: Criação do projeto para o Jogo da Forca

Ao confirmarmos a criação do projeto, com o template que escolhemos, uma nova página será exibida, informando a inexistência de colunas e nos oferecendo a possibilidade de criarmos uma neste momento. Vamos confirmar isso. Clique no botão para a criação da coluna.

As colunas para projetos do GitHub costumam ter nomes predefinidos e alguns deles são exibidos no campo, como um placeholder, mas, novamente, recomendo um estudo sobre isso, caso um aprofundamento neste tema seja um de seus objetivos.

Para nós, neste momento, daremos o nome de `To Do` e em `Preset`, escolheremos `To do`, marcando as opções de automação tal qual é apresentado pela figura a seguir. Após a configuração e informação do nome, clique no botão responsável

pela criação da coluna.

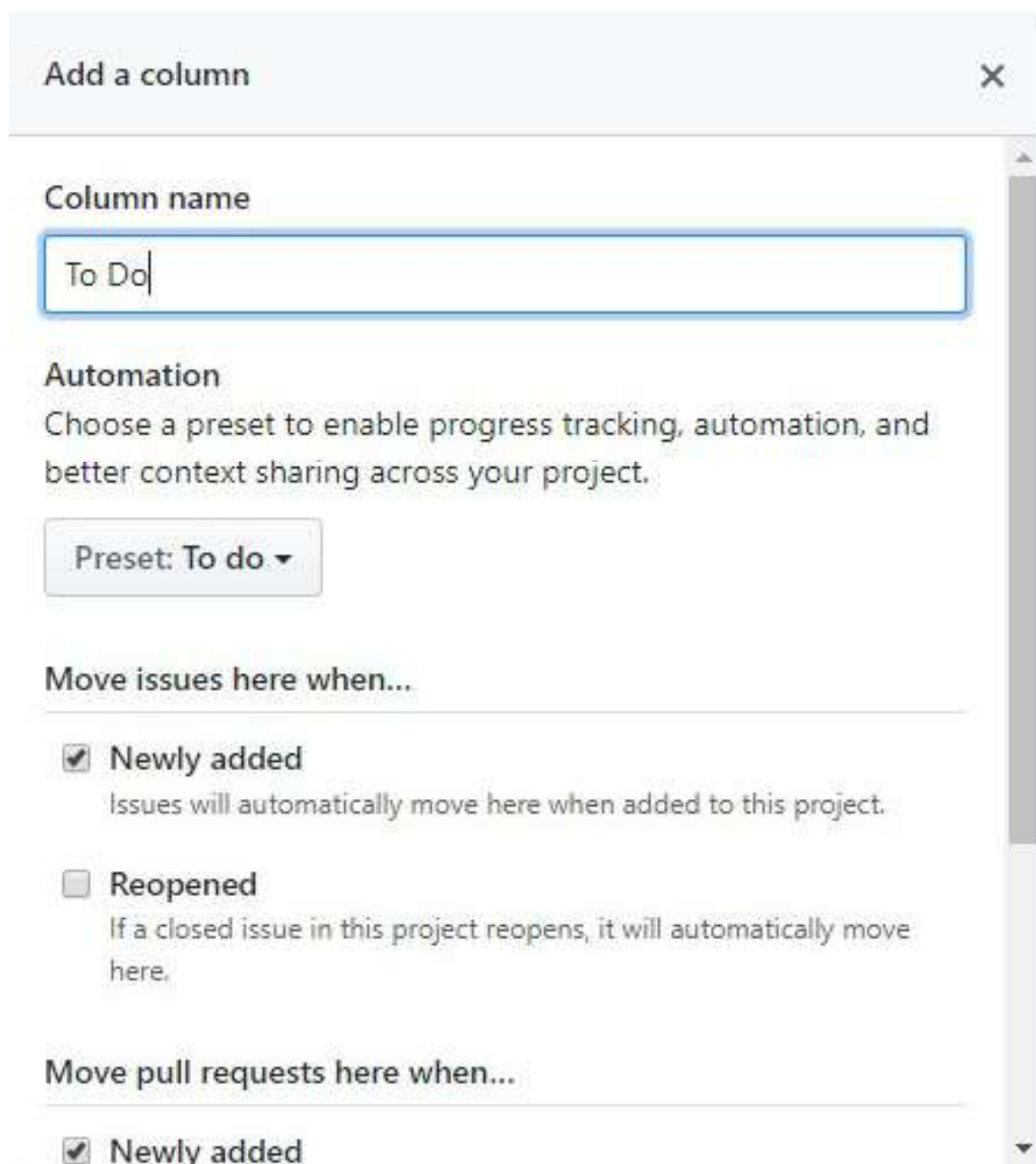


Figura 3.5: Criação da coluna To Do para o projeto

Com nossa coluna criada, teremos a visão de nosso projeto semelhante ao que temos apresentado na figura a seguir.

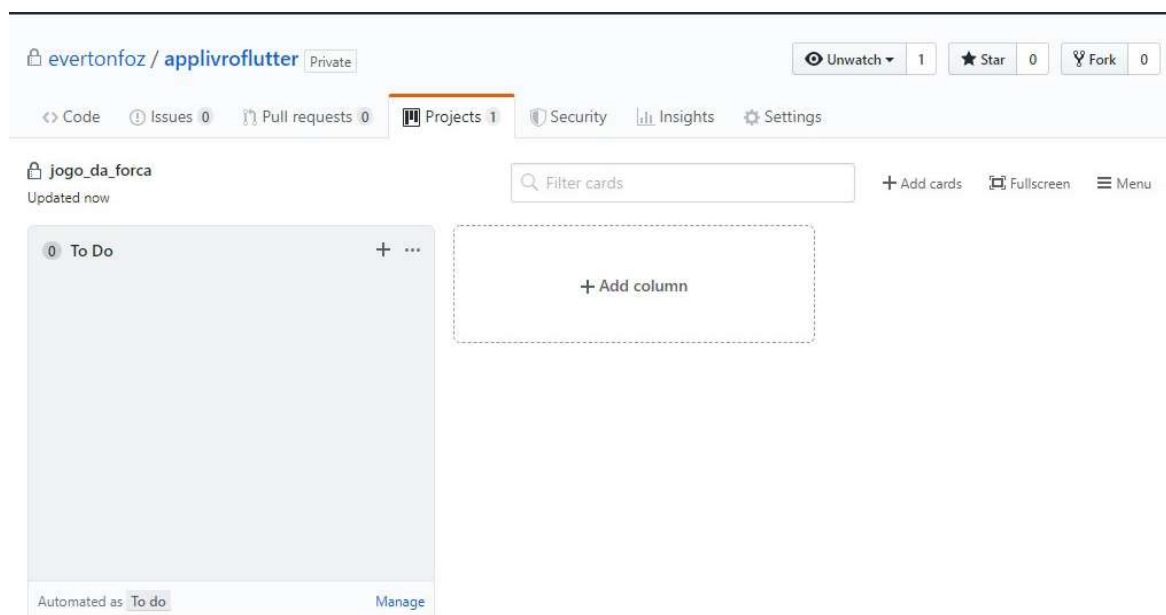


Figura 3.6: Projeto com a coluna To Do criada

Muito bem! Chegamos até aqui. Mas o que são estas colunas? Vamos a uma breve explicação? Bem, quando temos um projeto, temos nele, necessariamente, o registro de atividades, de questões (issues) que devem ser atendidas pela equipe de desenvolvimento. Estas issues, precisam ser registradas a uma área específica do processo de desenvolvimento de nosso projeto e, a cada área, no GitHub, temos uma coluna ligada a ela. Em nosso caso, a To do , que é a responsável por registrar funcionalidades que devem ser implementadas, corrigidas, melhoradas. É a relação do que deve ser feito pela equipe. Ficou um pouco mais claro?

3.5 O REGISTRO DE ISSUES AO PROJETO

Outra tradução possível para Issue pode ser "Problema". Normalmente, uma issue é algo reportado por usuários e que a equipe de desenvolvimento recebe, classifica e delega. Em nosso caso, no GitHub, as issues são classificadas e registradas em cards

específicos. No momento, temos apenas o `To do` .

Poderíamos, nós desenvolvedores, registrar nossos problemas diretamente neste card, pois estamos em fase de desenvolvimento e somos nós que temos os problemas levantados. Para isso, no topo do card, temos um botão de adição, que possibilita a criação de uma observação, chamado pelo GitHub de `Note` .

Podemos deixar esta observação assim, ou transformá-la, após a criação, em uma issue. Este processo é tranquilo. Teste criar uma observação e após ela criada, veja nas opções do menu da observação a possibilidade de convertê-la para uma issue.

Mas vamos criar direto uma issue para aprendermos? Nas opções oferecidas pela página, temos uma aba chamada `Issues` , clique nela. A interface é simples, não vou inserir imagem, ok? Nela, você verá facilmente um botão identificado como `New Issue` . Vamos clicar nele?

Na página que se abre, precisamos, além de informar o título para a issue, fornecer a descrição dela. É importante aqui ser o mais detalhista possível. Também se faz necessário algumas configurações, que estão sinalizadas na figura a seguir. Ao selecionar o repositório, ou projeto que temos, e, pela configuração que fizemos ao criar o card `To do` , a nova issue será automaticamente ligada a ele. Insira a issue e abra a guia do projeto e veja o card como estará.

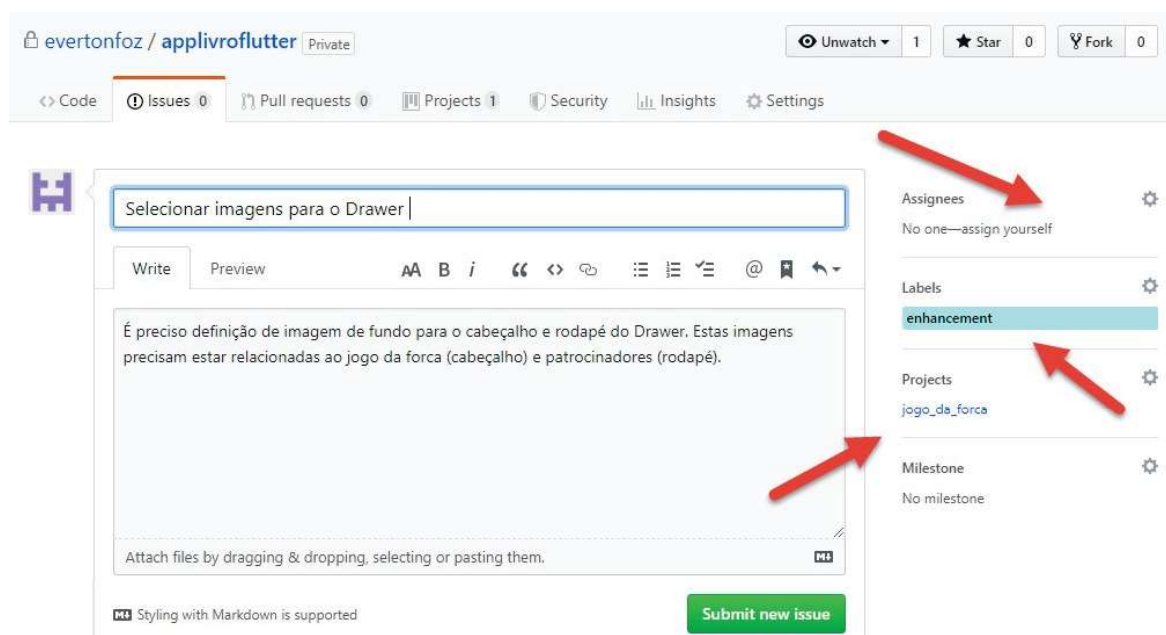


Figura 3.7: Registro de Issues

Isso tudo é básico. Eu realmente aconselho, caso seja seu interesse é claro, um aprofundamento nisso. Lembre-se de que nosso foco no livro não é o GitHub, mas vamos utilizá-lo no livro, então quis trazer um pouco para você.

3.6 REGISTRO DE ATIVIDADES EM UMA ISSUE

Em nosso Card To do temos lá uma tarefa a ser realizada, que é a responsável por realizar a busca e identificação das imagens que utilizaremos no desenho de nosso Drawer, o Widget escolhido, que nos propiciará a navegação entre as páginas disponibilizadas pela aplicação. Logo veremos este componente.

Vamos então à issue para registrar as imagens que comentamos. Vá ao seu repositório no GitHub e então, após encontrar o Card To Do, clique sobre o nome da Issue em uma

nova área, à direita, será ela exibida para você. Vamos aos detalhes da issue, clicando em `Go to issue for full details`.

Abaixo da Issue, existe campo para novos registros. Aqui, a equipe ou usuários da App podem realizar comentários, sugestões e críticas em relação à issue. É importante compreender este processo e metodologia. Veja a figura a seguir, que traz a resolução para nossa issue.

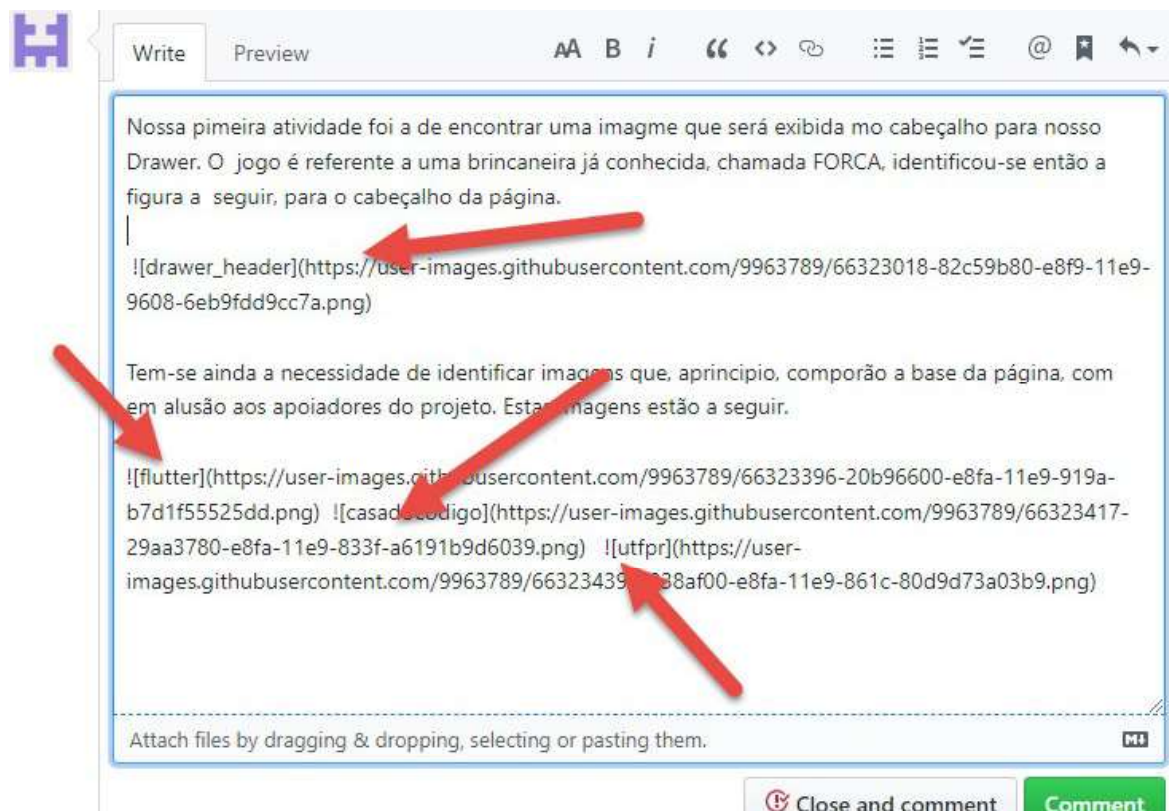


Figura 3.8: Registro do comentário em relação a execução da Issue

Caso você queira visualizar a Issue de uma maneira mais apresentável, selecione, ao lado esquerdo da Issue a opção `Preview` e então poderá visualizar as imagens, as quais você arrastou anteriormente.

Chegamos à finalização de uma issue, mas, se lembrarmos, não temos um bloco em nosso projeto para ter nele as issues fechadas.

Vamos então criá-los.

Vá a guia de Projetos. Nela você verá apenas o card `To do`, precisamos agora um que seja responsável em abrigar as issues que foram concluídas. Para isso, em projetos, adicione um novo Card. Dê a ele o nome `Done`, de realizados. Em `Preset`, escolha `Done` e então o grave.

Agora, de volta ao fechamento de nossa issue. Com nossa atividade realizada e já aprovada, precisamos então fechar a Issue. No campo abaixo de sua issue, existe outra, com opções para comentar e finalizar a demanda. Coloque um comentário que seja compreensível, em relação ao que foi realizado e então feche a issue. Lembre-se que esta documentação é sempre pública a toda a equipe e, dependendo do escopo de seu projeto, a todo o mundo.

Volte ao seu projeto e verifique seus Cards. O `To do` teve retirado sua Issue, que foi para o `Done`. Parece simples, mas é algo muito importante para uma equipe de desenvolvimento.

A partir de agora, o registro e atualização de Issues fica a seu critério, ok? Vamos focar em nosso App.

3.7 CLONANDO O REPOSITÓRIO REMOTO DO GITHUB PARA UM LOCAL EM NOSSO EQUIPAMENTO

Lembra que quando falamos sobre repositório, comentamos a existência de repositórios remotos e repositórios locais? Pois bem. Nós já temos nosso repositório remoto, que criamos no GitHub. Precisamos agora trazer este repositório para nossa máquina e então termos ele também localmente. O Git traz instruções de

console para auxiliar em todo este processo. Entretanto, aqui faremos uso de uma interface visual, que automatizará todo este processo para nós. Estou falando do `GitHub Desktop`. Acesse <https://desktop.github.com/>, realize o download do aplicativo e o instale em sua máquina.

Com o `GitHub Desktop` instalado, podemos trabalhar para trazermos à nossa máquina o repositório remoto. Acesse a guia `Code` de seu repositório. Nela, ao lado direito, verá um botão identificado como `Clone or download`, clique nele. Será exibida uma interface dando a você a possibilidade de realizar o download do projeto em formato de um arquivo compactado, ou, de abrir o projeto no seu desktop. Vamos optar por essa situação, abrir no desktop. Ao confirmar esta opção, uma janelinha pedindo autorização para abrir o `GitHub Desktop` será exibida. Confirme clicando no botão identificado como `Abrir GitHub Desktop`. O `GitHub Desktop` será aberto com uma janela que deve ser utilizada para configurar este repositório remoto como repositório local. Veja isso na figura a seguir.

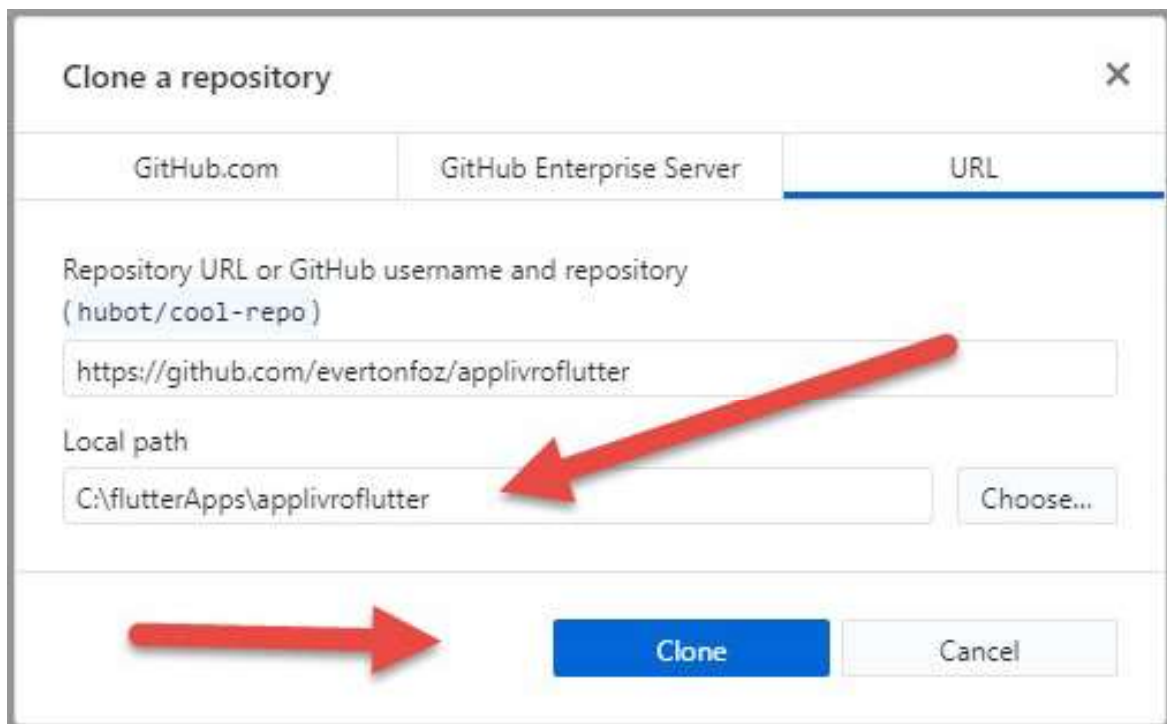


Figura 3.9: Realizando o Clone do repositório remoto para um repositório local

Configure corretamente o campo `Local path`, que é o local em que seu repositório local será criado. Ao final, pressione o botão `Clone`. A título de curiosidade, vá até a pasta indicada como repositório local. Veja nela a existência de uma pasta oculta, chamada `.git` e de um arquivo chamado `README.MD`, que é o arquivo que criamos junto com o repositório. A pasta `.git` é uma pasta que contém configurações que serão utilizadas pelo `Git` para manter os repositórios atualizados entre os `pushs` e `pulls`. Caso tenha interesse, pode entender um pouco sobre esta pasta em https://githowto.com/pt-BR/git_internals_git_directory.

Vamos agora criar nosso projeto Flutter referente ao jogo da Força no Android Studio. O que faremos pode parecer uma gambiarra, mas é para termos nosso projeto criado dentro de nosso repositório local, mas vamos lá, apenas leia toda esta subseção. Depois, leia a próxima, que é mais simples, para depois

então definir como prefere realizar este processo. Agora siga os passos já conhecidos para a criação do projeto. Apenas lembre-se que este é um dos caminhos.

Após a criação de nosso projeto, copie todo o conteúdo da pasta onde ele foi criado para a pasta de nosso repositório local e então retorne ao GitHub Desktop. Veja que uma série de arquivos apareceram como modificados desde o último pull que realizamos, que na realidade foi quando clonamos nosso repositório remoto para local. Veja estas informações na figura a seguir, apontados pelas setas em vermelho.

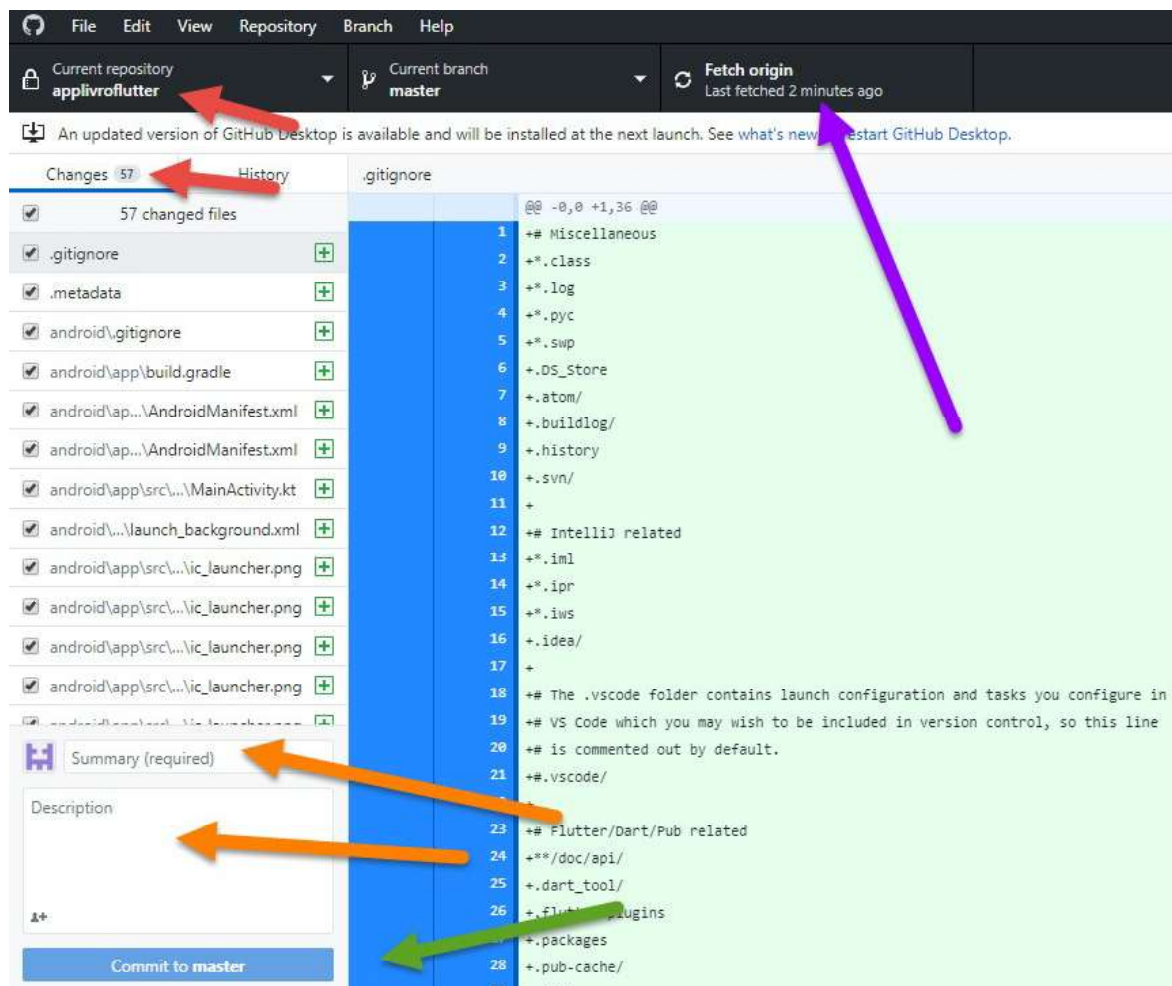


Figura 3.10: Verificando alterações realizadas no repositório local

Na figura anterior, nos pontos sinalizados pelas setas laranjadas, precisamos documentar o que foi feito em nosso projeto, localmente, para então submetermos para o repositório remoto. Em `Summary`, vamos colocar Criação do projeto e em `Description`, vamos inserir O projeto foi criado no Android Studio. É muito importante descrevermos sempre bem nosso commit, pois isso poderá ser usado como documentação orientativa para quem visitar nosso repositório remoto, ou seja, a equipe envolvida no projeto. Muito bem! Agora precisamos clicar no botão `Commit to master`.

Ao registrar o commit, note que a informação da área apontada pela seta em roxo é alterada. Temos algo modificado em nosso repositório local. Temos agora o `Push origin`. Precisamos submeter estas alterações para o repositório remoto e faremos isso clicando nessa área.

Uma situação inversa também pode ocorrer. Ou seja, termos a informação de que algo no repositório remoto mudou em relação à versão que temos em nosso repositório local e essa informação estaria no mesmo local comentado. Neste caso, precisaríamos realizar um pull para atualizar nosso repositório local. A título de curiosidade, veja no GitHub como está nosso repositório.

Precisamos agora garantir que o Android Studio esteja devidamente configurado para usar o Git e GitHub. Sendo assim, no Android Studio, acesse `File->Settings->Version Control->Git` e configure o caminho para o Git, previamente instalado. Teste para garantir que esteja tudo certo. Veja a figura a seguir.

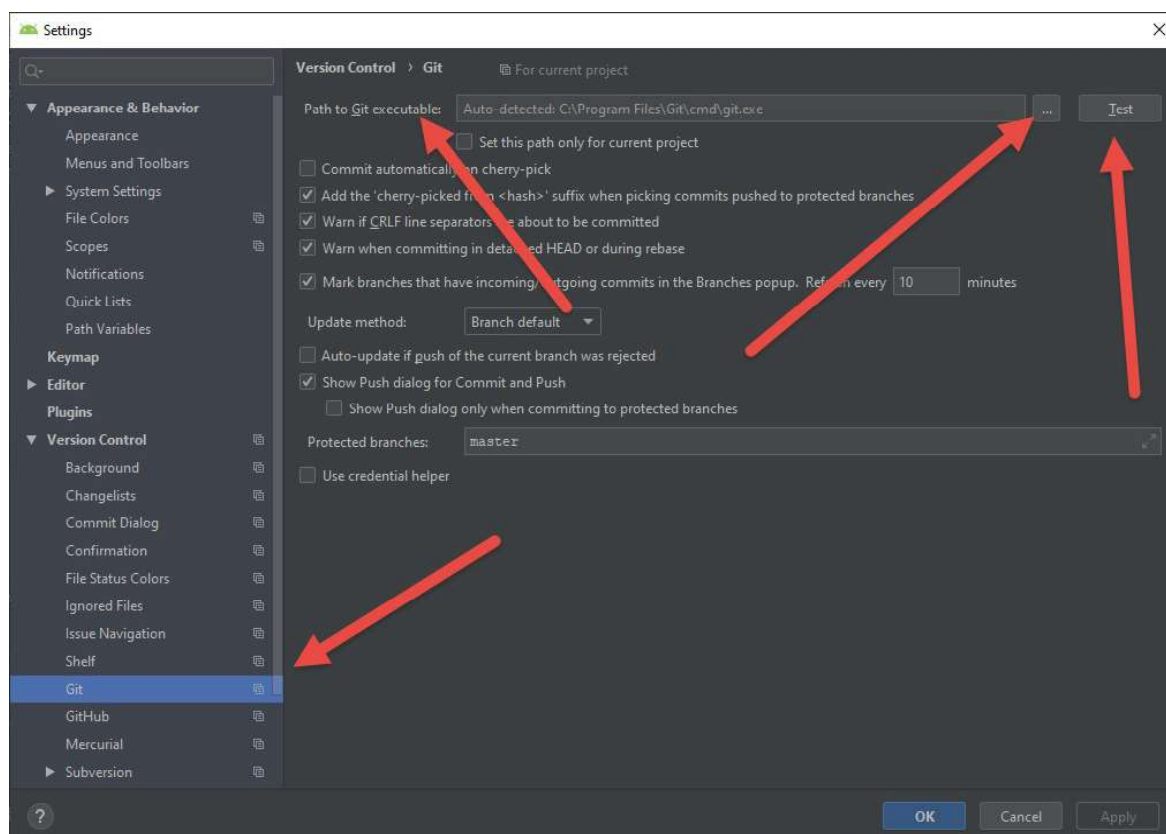


Figura 3.11: Configuração do Git no Android Studio

Na sequência, precisamos configurar nosso acesso ao GitHub , na mesma janela anterior, mudando apenas a última opção para o GitHub . Você precisará adicionar suas credenciais. Para auxiliar, veja também a figura a seguir.

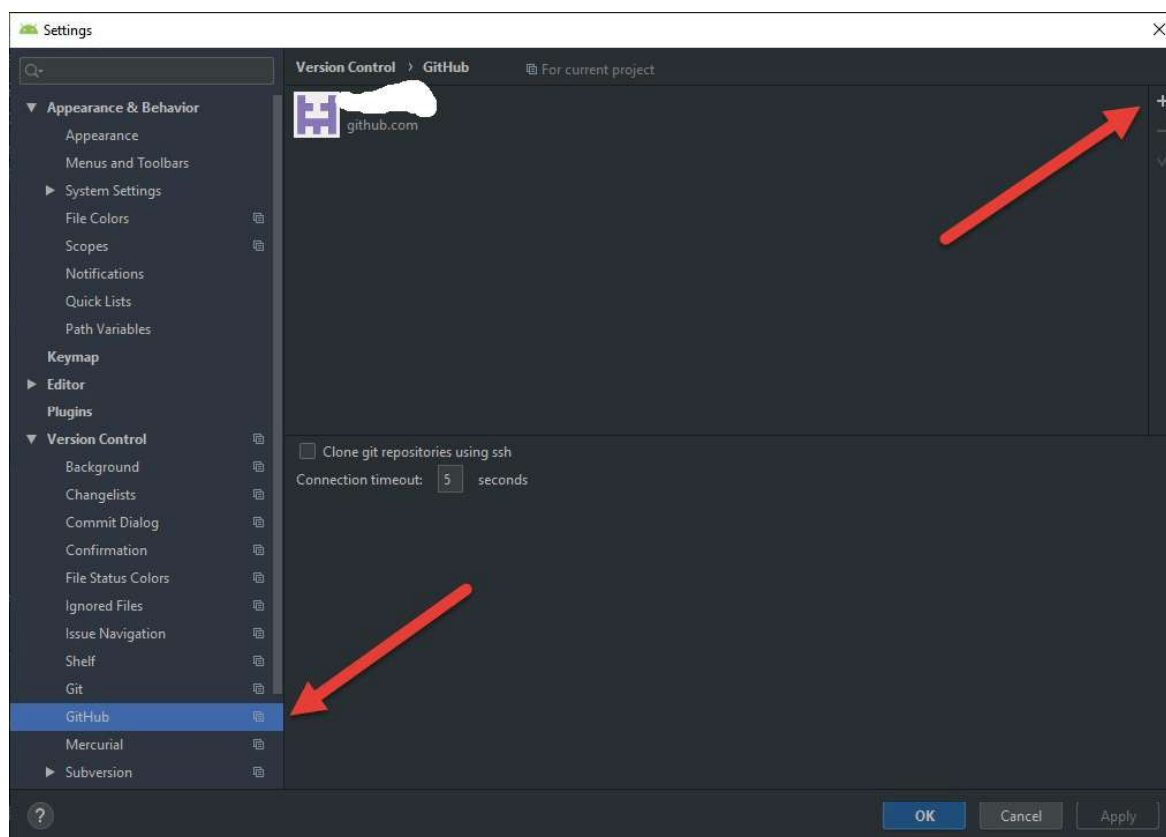


Figura 3.12: Configuração do GitHub no Android Studio

Pode fechar agora o GitHub Desktop e abrir nosso projeto no Android Studio. Veja que, ao abrirmos nosso projeto, temos na barra de tarefas do IDE uma nova região de botões, que é relacionada a operações com nosso repositório. Poderemos realizar nossos commits, pushes e pulls diretamente do IDE. Isso é bem prático. Veja esta área na figura a seguir.

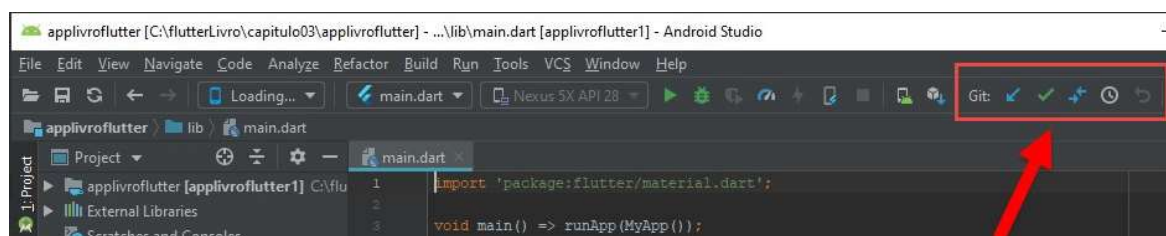


Figura 3.13: Acesso a ferramentas Git no Android Studio

A maneira apresentada anteriormente foi para a situação de

termos criado um projeto no GitHub para hospedar nosso projeto Flutter, criado em momentos diferentes. Veremos agora uma maneira em que podemos, diretamente do Android Studio, criar nosso repositório no GitHub, é bem mais prático, e, após isso, poderíamos realizar toda a configuração vista anteriormente.

3.8 CRIAÇÃO DO REPOSITÓRIO DIRETAMENTE DO ANDROID STUDIO

Tendo a certeza de que temos o Git e GitHub devidamente configurados em nosso Android Studio e que você tenha seu projeto Flutter aberto no IDE, siga os passos a seguir. Eles possibilitarão a criação de um repositório no GitHub e então a integração de nosso projeto com ele.

No Android Studio, acesse `VCS->Import into Version Control->Share Project on GitHub` e então suas credenciais serão solicitadas. Veja a figura a seguir.

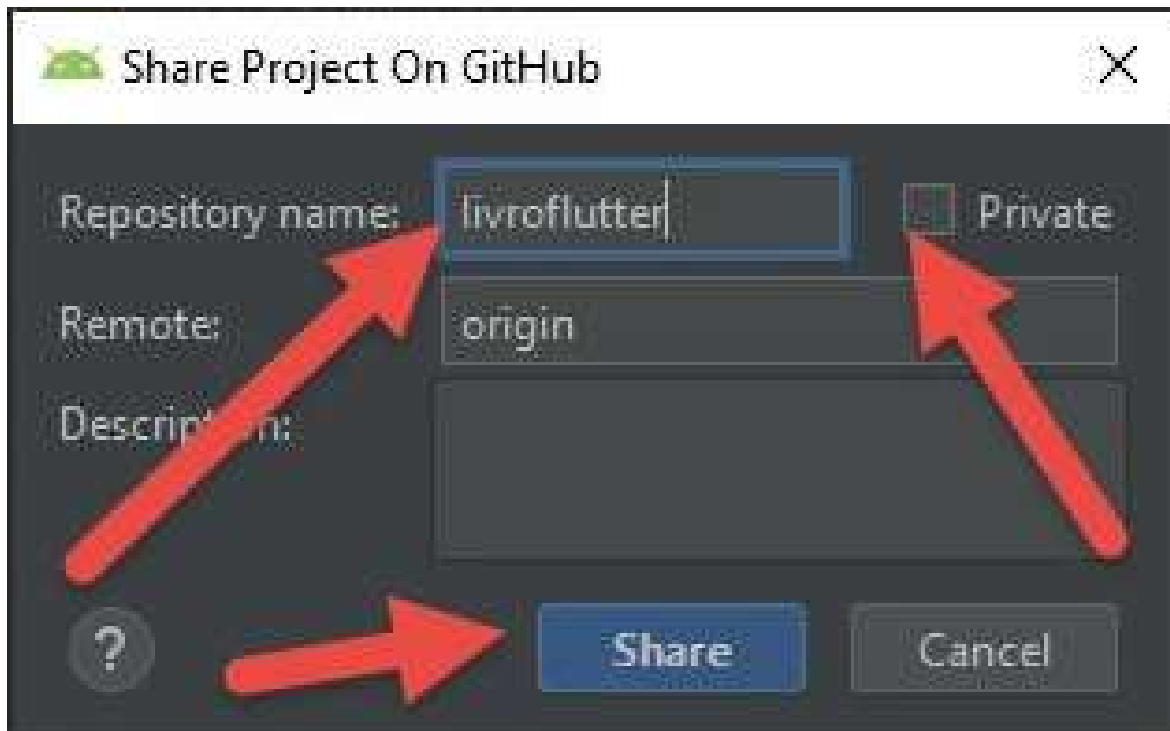


Figura 3.14: Compartilhando projeto ao GitHub

Pronto, já temos um novo repositório criado no GitHub. Caso você opte por esta maneira, pode realizar neste repositório toda configuração que vimos anteriormente. Verifique que a barra de tarefas do IDE já mostra os botões específicos para interação com o GitHub, tal qual vimos na subseção anterior.

Existe, outras maneiras de criarmos ou configurarmos nossos projetos para o uso do Git, mas preferi utilizar estas duas, por serem mais simples e de fácil assimilação. Poderemos agora, enfim, começarmos nosso projeto do jogo da Força e, nos momentos certos, faremos uso dos recursos do Git e GitHub.

3.9 CONCLUSÃO

Esta foi uma simples introdução ao GitHub e nos permitiu preparar nosso projeto Flutter para ser compartilhado no GitHub.

Aprendemos como, no GitHub, criarmos nosso repositório, ligar projetos a eles, criarmos `Cards` para organizar nossas atividades durante o processo de desenvolvimento e também ao registro de `Issues` .

Isso propiciará a você o registro de acompanhamento de toda a etapa de desenvolvimento de nosso projeto.

No próximo capítulo, começaremos a implementar nossa aplicação que utilizaremos aqui no livro, criando nossa página de inicialização, a `Splashscreen` .