**Praticando Git e GitHub**

**Diferença entre Git e GitHub**

**Git:**

Sistema de controle de versionamento de arquivos. Usado principalmente para desenvolvimento de softwares, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo.

Nos permite ter controle das alterações dos arquivos salvas em diferentes versões. Com isso, é possível retomar uma versão anterior (ou quantas necessário for), caso necessitemos corrigir alguma parte do código que não funcionou após alguma alteração ou até mesmo no caso de por algum motivo, não precisar mais dar continuidade àquela alteração.

Com o Git, é possível que diversas pessoas de um grupo de trabalho realizem alterações do mesmo arquivo, incrementando, removendo e alterando partes do código sem causar impacto sobre o outro integrante do grupo de trabalho. O arquivo alterado pode ser salvo de forma local na máquina, e posteriormente pode ser encaminhado para a um repositório remoto (com a validação de um aprovador), de forma que os outros membros do grupo tenham acesso àquela alteração.

Permite também a alteração ou criação de vários arquivos de um mesmo projeto de forma que cada participante deste grupo ao finalizar seu bloco de alterações (separados por branches), poderá testar seu código separadamente, e confirmar o funcionamento de sua parte. Com isso, ele poderá descrever as alterações realizadas em cada arquivo criado/alterado e enviar para o repositório remoto, que conforme informado anteriormente, após a validação do aprovador, este poderá compilar todas as partes do projeto em uma única “branch” (master/main), tendo então o projeto inteiro guardado.

Cito então, algumas ferramentas e funções básicas e essenciais para utilização:

Branch: São “*marcos*” nas ramificações contendo a “linha do tempo” de um projeto, onde é possível visualizar o marco do momento em que cada usuário “salvou” (deu commit) suas alterações. Com ela, é possível acompanhar e reunir várias alterações diferentes feitas simultaneamente no mesmo arquivo (contanto que a alteração de um código não afete a funcionalidade criada em outro) ou em diferentes arquivos do projeto.

Commit: Este é utilizado para *salvar* alterações realizadas em uma branch local. Os arquivos deverão conter uma breve descrição/comentário da alteração realizada separadamente, de forma que facilite a identificação da alteração realizada.

Push: Esta função é utilizada para *publicar* as modificações locais (que sofreram o commit) para um repositório central (remote) de forma a compartilhar com os membros da equipe (caso realizem o “pull”) o código alterado.

Fetch: *Baixa* o conteúdo do repositório remoto selecionado para o local.

Pull: *Busca e baixa* as alterações de códigos do repositório remoto selecionado e atualiza imediatamente o repositório local para que os conteúdos sejam iguais. O pull/merge, ele já tenta fazer o merge das alterações baixadas direto no seu código (caso não haja conflito).

Merge: *Mescla* as alterações realizadas, e publicadas (commit) de forma que todas as partes do código(arquivos/código/projeto) criadas/alteradas sejam reunidas, tornando-se então, uma nova versão do projeto.

Checkout: Basicamente permite a *navegação entre as branches*. Para iniciar as alterações em uma determinada branch, faz-se necessário primeiramente fazer checkout na mesma.

Remote: Repositório salvo em um local remoto (basicamente um serviço em nuvem) onde ficarão guardadas as alterações do projeto que poderão ser publicas ou não. Como exemplo de repositório remoto, temos o GitHub.

No site do Git, em “GUI Clients” existem algumas interfaces para melhor visualização e gerenciamento dos arquivos e códigos dos projetos. Porém, esta não é a proposta do documento. Sendo assim, utilizaremos o próprio prompt de comando do Git para realizar as funções da ferramenta, enquanto o controle de versões faremos pelo próprio GitHub.

**Alguns comandos Git**

**git init:** Cria o diretório “.git” dentro da pasta indicada tornando-a um repositório (master).

**git add Readme.md:** Envia um arquivo (especificado pelo nome) para a área de "staging" (encenação - uma espécie de espera). Local onde o arquivo fica aguardando o commit.

**git add .** **:** Envia todos os arquivos salvos (na área staging) para a área de commit.

**git status:** informa se existe algum arquivo aguardando commit na branch master. Caso haja, informa em "new file" o arquivo aguardando commit.

**git commit "comentário descrevendo as alterações realizadas":** realiza o commit do arquivo com o comentário contendo a descrição da alteração entre aspas.

**git branch -M "main":** altera o nome da branch principal de "master" para "main".

**git remote add origin** [**https://github.com/EnricoLaqua/BestTrip.git**](https://github.com/EnricoLaqua/BestTrip.git)**:** Faz a conexão do repositório local com o repositório remoto.

**git push -u origin main:** solicita o push das alterações locais para a remota. Abre a tela de login para acessar o github e permitir o envio. Envia um email confirmando o push e sobe a mensagem no prompt do Git informando o push.

**git push origin main:** (sem -u), quando realizar apenas fazer o push pois o repositório ja foi criado anteriormente, configurando a origin como upstream no Git.

**git checkout -b "nome-da-branch":** Cria a branch "nome-da-branch" e já da checkout nela.

**git checkout nome-da-branch:** realiza o checkout na branch (alternando entre as branches) para que possa realizar alterações nela e visualizar as alterações que foram feitas. Exemplo: “git checkout master” faz com que eu entre na branch master.

**GitHub:**

Plataforma para hospedagem de arquivos e código-fonte. Pode ser utilizado para guardar os arquivos com o controle de versionamento usando o Git.

O GitHub trabalha com repositórios (diretórios), usados para armazenar os projetos. Dentro destes repositórios (intitulados normalmente com o nome do projeto), ficam os respectivos arquivos do projeto.

A interface do GitHub permite que o usuário consiga visualizar o conteúdo do arquivo (script do código) dentro da própria página do arquivo apenas selecionando-o.

A ferramenta permite ainda o compartilhamento de arquivos e projetos, de forma a contribuir em projetos privados ou Open Source com qualquer usuário no mundo. Os usuários podem também mesclar repositórios remotos.

É muito usada também para a divulgação de trabalhos e como portifólio.

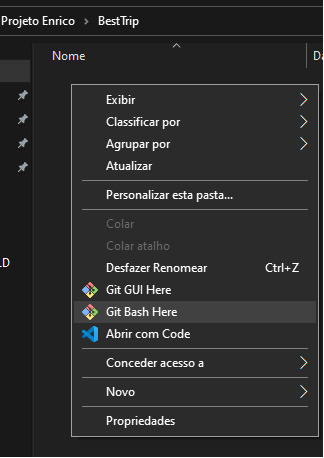
Assim como Git, o GitHub também trabalha com branches.

Em suma, ambos os sistemas são facilitadores do dia a dia de muitos desenvolvedores, uma vez que reduz a morosidade e o tempo de entrega dos projetos (visto que vários usuários podem trabalhar simultaneamente), além do controle de versões que reduz muito retrabalho.

**Projeto do Site BestTrip**

Inicialmente, precisaremos criar um diretório local na máquina, onde armazenaremos todos os arquivos do projeto. Nele, será gerado o arquivo “.git”, inicializando um novo repositório Git a partir deste repositório local. No caso do nosso site, utilizaremos o diretório “BestTrip”.

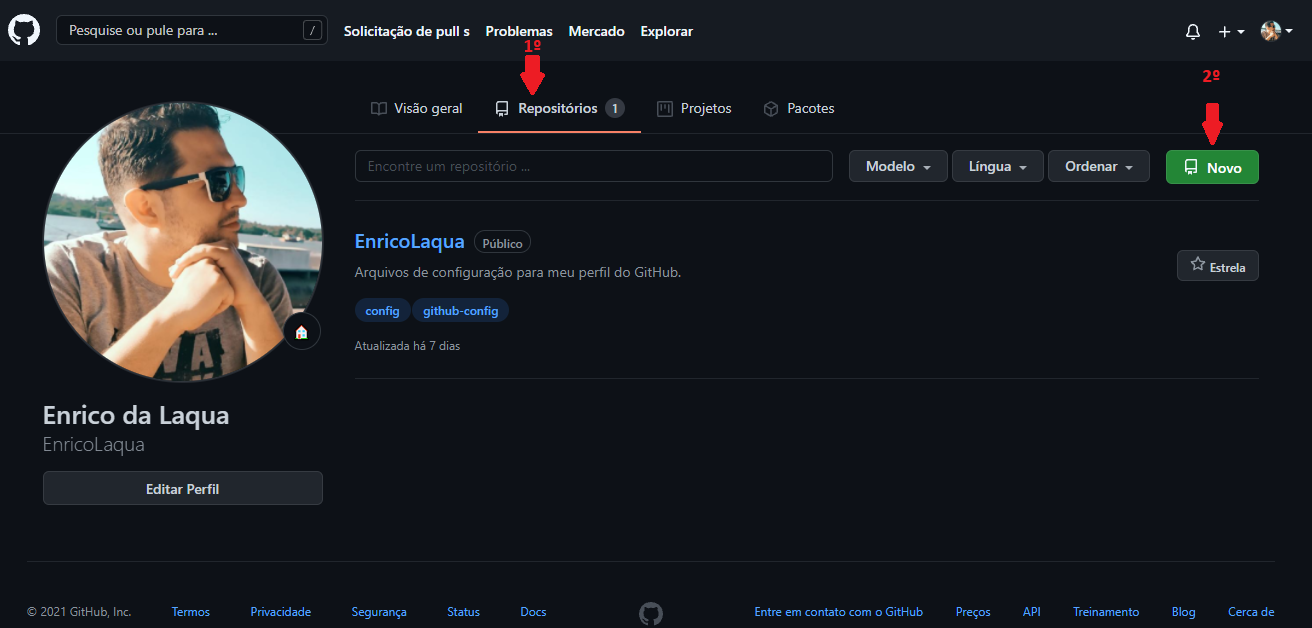
1. Dentro do diretório com o nome do projeto (BestTrip), clique com o botão direito do mouse, e selecione “Git Bash Here”.



1. Abrirá o prompt de comando do Git, que utilizaremos para realizar as alterações do projeto. O primeiro comando que iremos digitar será o “git init”, que criará o diretório “.git” com todos os arquivos necessários do git, indicando que aquele será o repositório do projeto.
2. Com o repositório criado, podemos iniciar o desenvolvimento do site, salvando todos os arquivos (“.html”, “.css”, “.jgp”...) dentro desta pasta “BestTrip”.

Com os arquivos criados, o próximo passo é voltar ao prompt do Git e digitar o comando “git add index.html” (ou simplesmente “git add .” para adicionar todos os arquivos do projeto de uma só vez). Conforme descrito anteriormente, este comando encaminhará os arquivos do projeto para a área de “staging”, onde ficará aguardando o commit.

1. O comando “git status” mostrará todos os arquivos aguardando commit na área de “staging”.
2. No próximo passo, usaremos o comando “git commit -m “primeira versão da home””, para dar commit no arquivo index.html e comentar as alterações que foram feitas no arquivo.
3. Agora precisamos criar um repositório no GitHub para que possamos salvar os arquivos de forma remota e fazer o controle de versionamento. Para isso, no site do GitHub, dentro da página principal do seu perfil, clique em “*Repositórios*” e depois em “*Novo*”.



1. Na tela a seguir, preencheremos o “Repository name” com o nome do projeto e em “Description”, uma breve descrição do projeto. Os outros campos manteremos como “dafault” e por fim, clicar em “Create Repository”.

Texto

Descrição gerada automaticamente

1. Ao criar o repositório, o próprio GitHub informa a lista de comandos para auxiliar na manipulação dos arquivos entre Git e GitHub. Observe que na parte de cima, existe uma linha com um link <https://github.com/EnricoLaqua/BestTrip.git>. Este é o caminho onde será guardado o repositório remoto no GitHub e com ele faremos a conexão entre o repositório local e remoto.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

1. Conforme mencionado acima, o repositório foi criado na máquina local e no GitHub, porém ainda não existe um link de conexão entre eles. Para fazer essa conexão, no prompt de comandos do Git, será inserida a entrada “git remote add origin <https://github.com/EnricoLaqua/BestTrip.git>” (colar -tecla “insert” ou “ctrl + shift + v”- o link do repositório que foi copiado na tela mencionada acima). Neste comando, estamos fazendo uma conexão do repositório local com o repositório do GitHub através do link e dando o nome do repositório do GitHub de “origin” (origem).
2. Neste próximo passo, faremos um “Push” para enviar o arquivo “index.html” para o GitHub, através do comando “git push -u origin master”. Conforme descrito acima, este comando envia as alterações locais para o repositório remoto (origin), na branch principal (main).
3. Abrirá uma tela de autenticação para que seja realizado login no GitHub. Clicaremos em “Sign in with your browser” e realizaremos o login. Após este passo, no prompt do Git irá retornar uma mensagem informando o “push” realizado. Ao atualizar a página do GitHub, será possível visualizar o arquivo index.html no repositório.
4. Por fim, enviaremos os demais arquivos do projeto para o GitHub simultaneamente através do comando “git add .”. Conforme já descrito acima, ele envia todos os arquivos para aguardar o commit.
5. Novamente utilizaremos o “git commit -m “Envio dos demais arquivos do site e atualização do index.html””.
6. E desta vez, para o “push”, utilizaremos o comando “git push origin master”, sem utilizar o “-u” pois a “origin” já foi definida anteriormente (no passo #10) como repositório remoto upstream no Git.
7. No GitHub será possível visualizar as alterações do index.html e os demais no repositório, com seus respectivos códigos.