Básico Front-End

Git

O Git é um programa de versionamento de código muito usado. Ele é muito parecido com CMD quando se quer navegar entre as pastas criar, deletar etc. Mas seu terminal se chama VIM.

Comandos CMD

dir: navegar;

mkdir: criar pastar;

rmdir /s /q: deletar pasta;

cd /: entrar em um disco;

cd (+ nome da pasta): entrar na pasta;

cd ..: Voltar;

cls: limpar;

echo: escrever;

echo + nome > "nome": criar arquivo para pasta;

del: deletar o que está na pasta.

Pwd: mostra o caminho;

Ls: mostra o conteúdo da pasta;

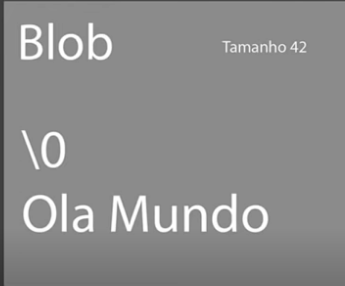
mv: Mover;

./ (+ pasta) : selecionar pasta.

Sha1: Criptografia base para controlar versões de arquivos do git, tendo sempre 40 caracteres que mudam conforme o arquivo é alterado.

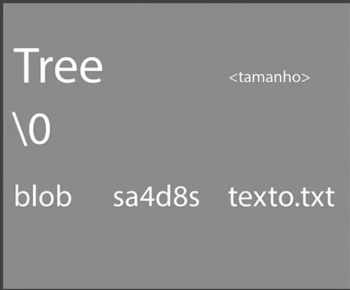
Objetos no Git

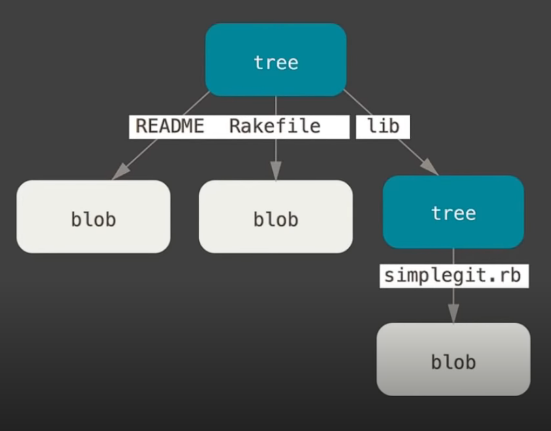
Blobs: Podemos usá-lo para obter o mesmo hash (ou id) do arquivo gerado pelo sha1. Contém meta dados em que há o tipo do objeto, o seu tamanho uma \0 e por fim seu conteúdo:





Trees: Servem para apontar os shars armazenados nos meta datas dos Blobs, guardando também o nome do arquivo e tamanho, montando toda a estrutura de onde estão os arquivos. Também podem apontar para outras Trees que consequentemente apontam para Blobs:

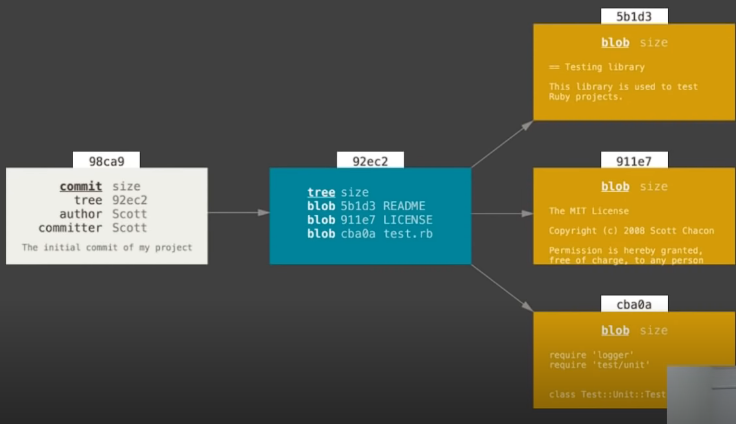




Commits: O mais importante pois dá sentido a toda alteração, apontando algum Commit realizado antes dele (parente), para uma Tree, autor e mensagem. Também informa o tamanho e o timestamp (Carimbo de tempo, data, hora, etc):



É basicamente isso:



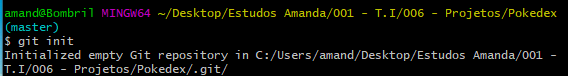
Github

Não tem como falar de Git sem falar do Github, é um servidor de hospedação de código ao público ou privado.

Chave SSH: Conexão segura e encriptada entre duas máquinas para o Github. Fazendo pelo Git Bash é só colocar o seguinte código: ssh-keygen -t rsa -C + Email.

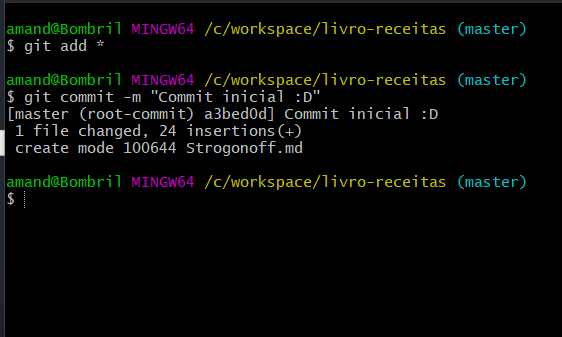
Passo a passo pra criar um versionamento e Commit

git innit: iniciar versionamento;

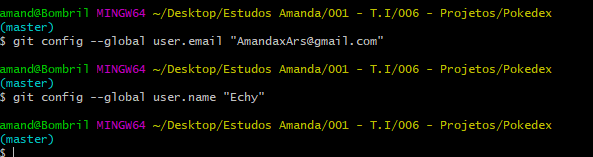


git add \*: adicionar pasta ou arquivo na pasta ao versionamento;

git commit: Criar commit;



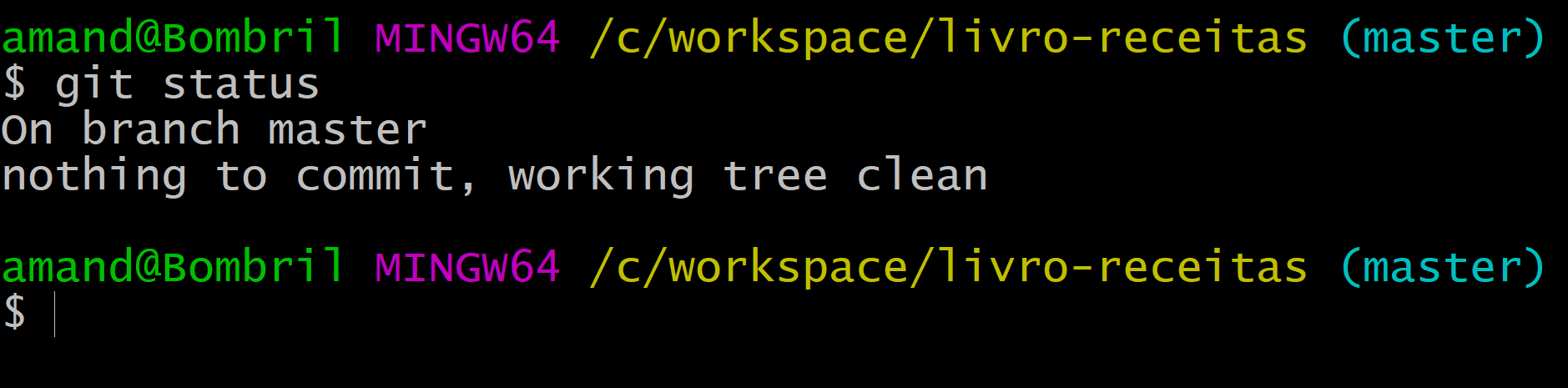
git config: configurar, para adicionar autor, etc;

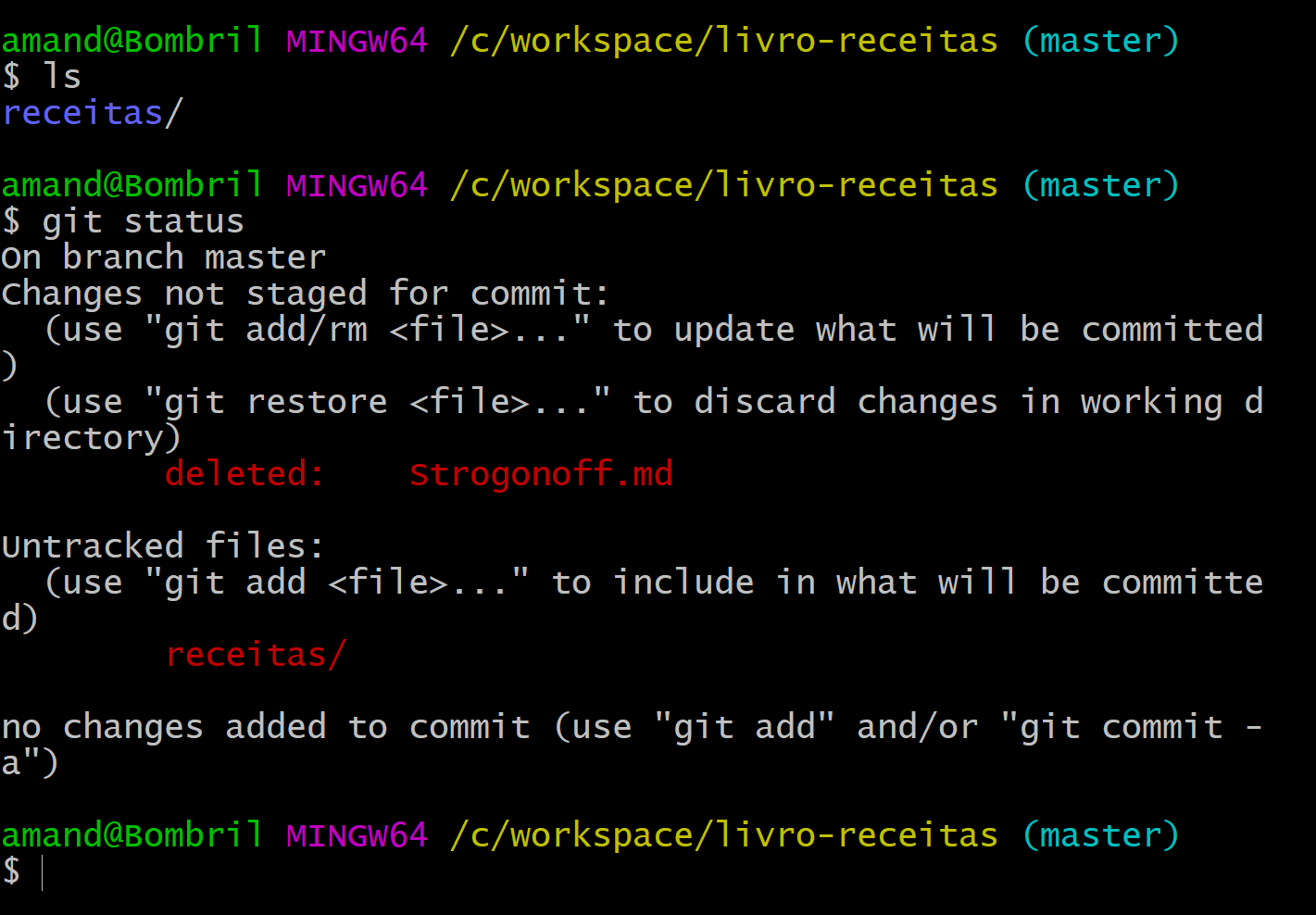


git config –list: Mostra a lista de configurações.

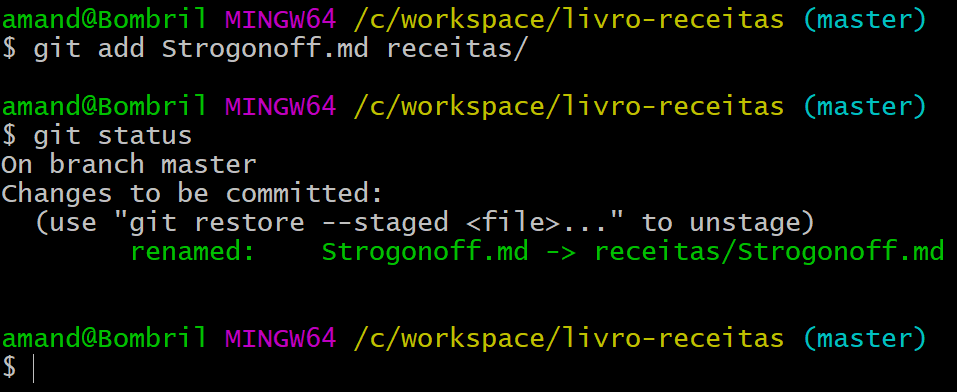
git status: mostra o status do versionamento.

Sem modificações:

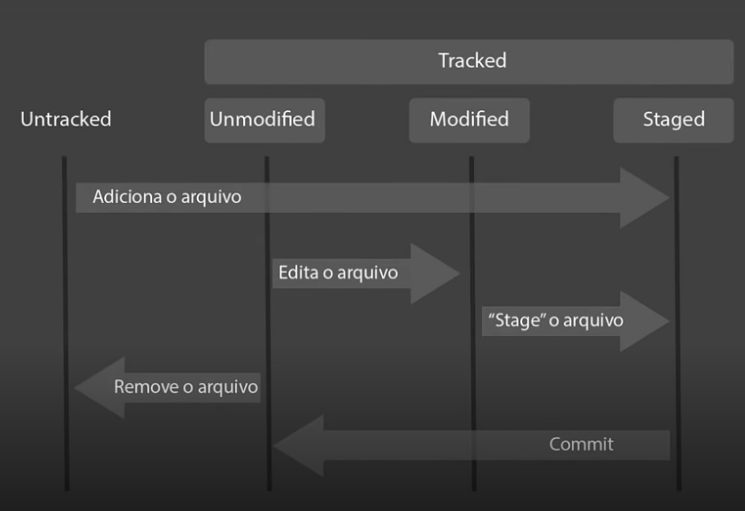


Modificado, mas não adicionado: 

Modificado e adicionado:



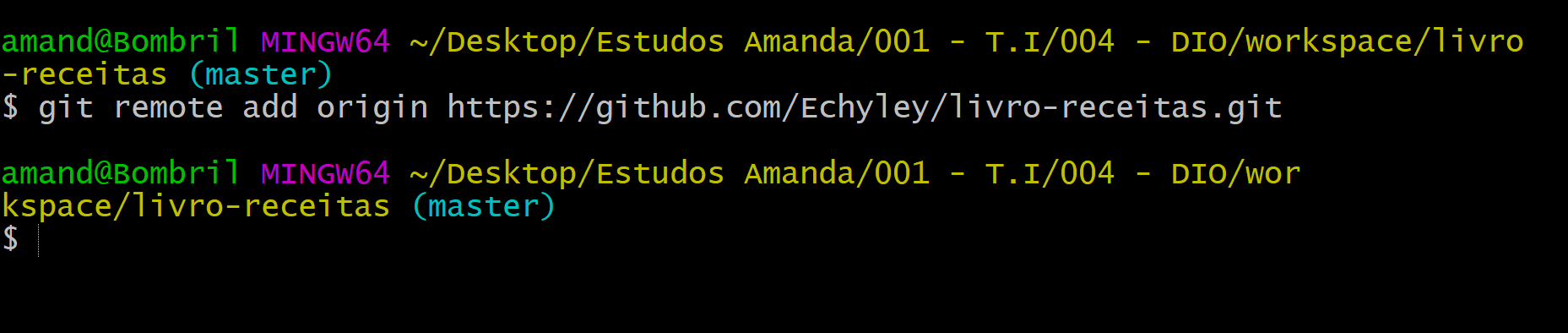
git restore (+ nome do arquivo ou pasta): Restaurar status anterior.

Resumo desde do versionamento ao commit

Repositório local ao remoto (Github)

Primeiro é preciso adicionar o ponto de origem do repositório remoto dentro do local:

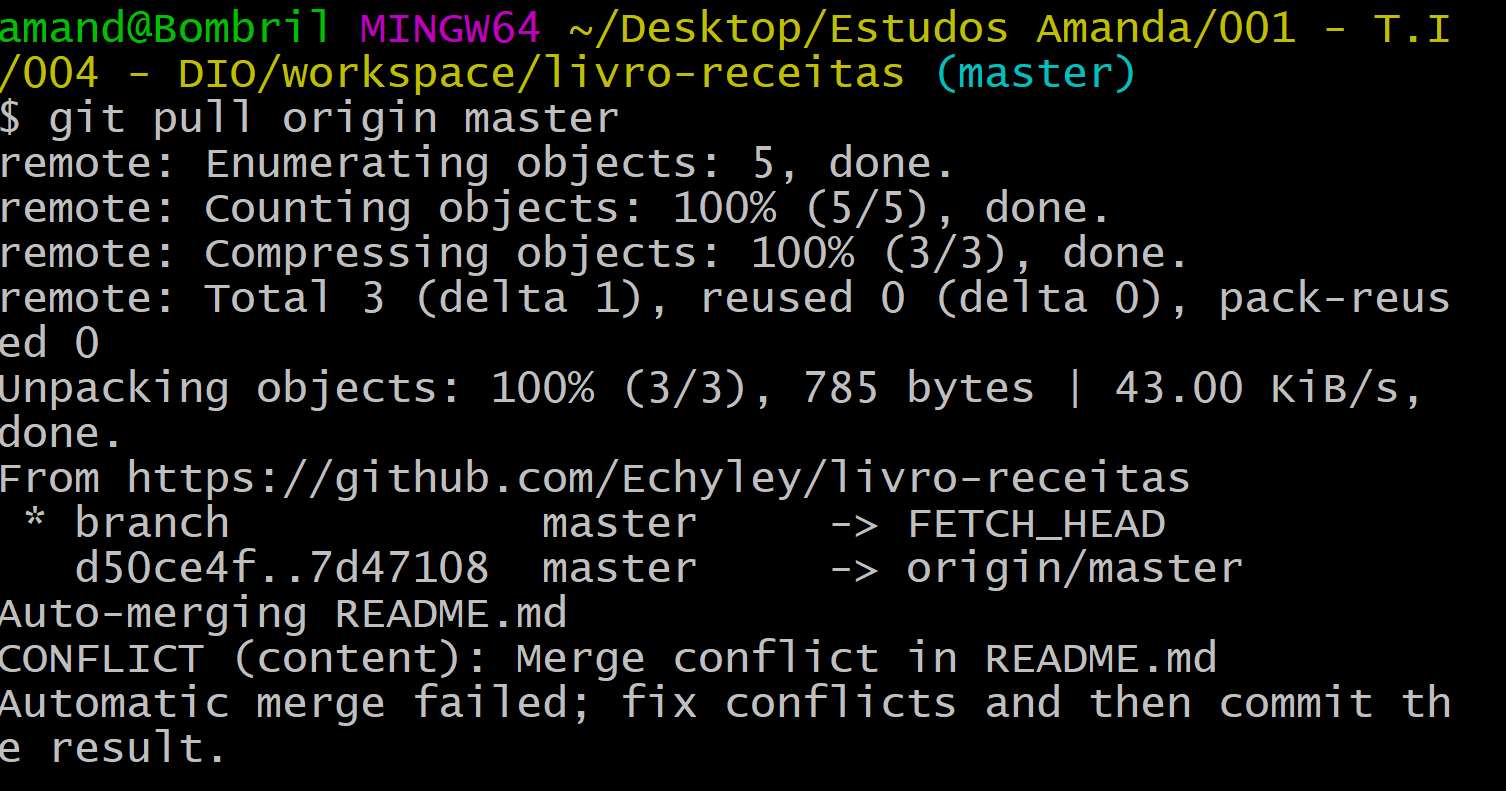
git remote add origin (+ URL): adiciona origem do repositório remoto.

git remote -v: Lista de url cadastrada no repositório.

git push (+ nick e Branch): Empurra o repositório local para o remoto pedindo pra logar no github se for a primeira vez.

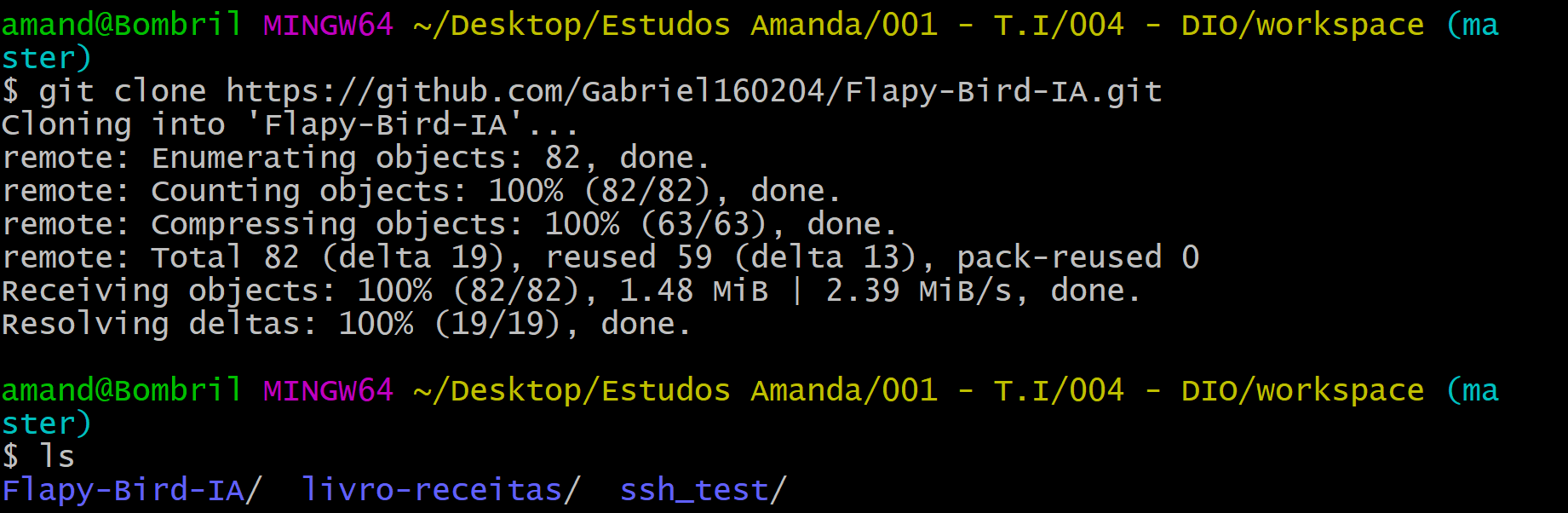


git pull (+ nick e branch): Puxa o repositório remoto para o local junto com suas mudanças.



Clonagem

git clone (+ URL): Clona um repositório remoto para o local.



Flags no git

-a: Mostra arquivos ou pastas ocultos;

--global: Deixar para acesso global;

- m: usado pra dar commit;

API

São servidores que podem nos fornecer dados já prontos e adiciona-los ao nosso código (poupa muito, mas muito tempo). São transferidos em sua maioria pelo formato de protocolo HTTP-Server.

Requisições: Na hora da requisição, quando o cliente solicita os dados, o sistema simplesmente lhe retorna o que foi solicitado. Requisições API sempre serão feitas por uma linguagem de programação.

Estrutura de requisições API

Url path: Endereço, caminho, Porta do servidor;

Method: Tipo de solicitação Definida pela linguagem de programação ou pelo próprio navegador. Os seguintes são os mais básicos:

* Get: Busca um recurso;
* Post: Inserir algo novo na API;
* Put: Atualizar algo na API;
* Delete: Exclui;

Patch Params: São formas de buscar por algo dentro da própria url;

Query String: É um filtro atribuído de acordo com o que o cliente busca. Uma estrutura de chave e valor. Na Url vem depois de um “?”.

Response e Request Header: Configurações ou “regras” que são compartilhadas nessa área de transmissão e recepção de dados entre servidor e cliente;

Body: Massa principal, dependendo do tipo recebe grandes informações;

Status Code: Resposta do servidor a requisição;

Java Script para web

Callback: É quando há várias funções dentro de funções chamando outras funções, isso complica o código e não é bom.

ArrowFuncions "=>" é uma sintaxe reduzida em relação a uma função e simplificam o código.

O fetch retorna uma resposta promise, que não chega na hora. Ela precisa passar por um caminho antes.

Json é um formato JS que cria um objeto convertido para se utilizar numa aplicação (nesse caso na API);

O catch manipula o fracasso;

O finally é executado independete do sucesso ou fracasso de uma requisição: .finally(function () {console.log('Requisição concluida!')})

Map: Transforma um elemento em outro.

Join: Junta elementos ou objetos;