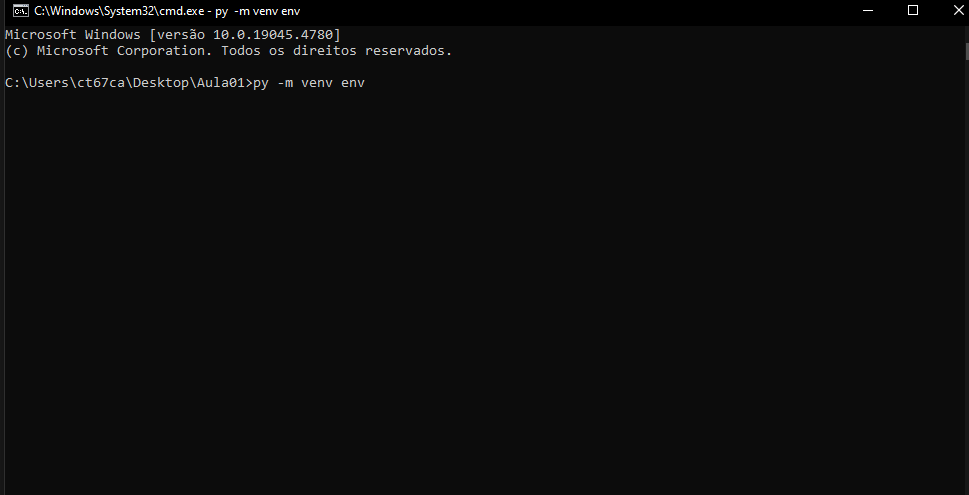
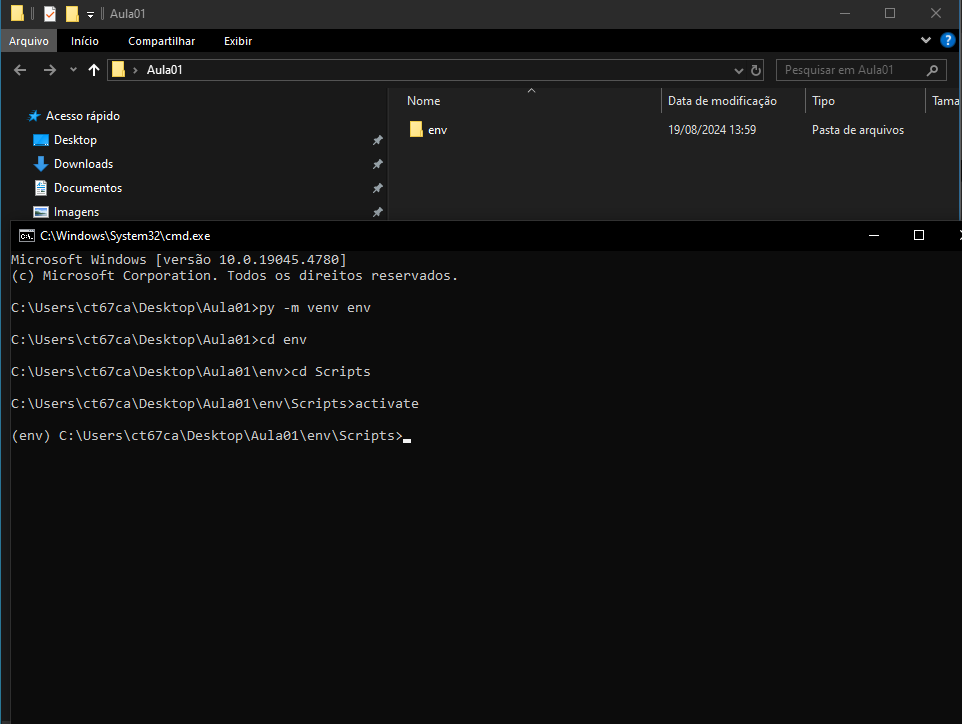
FAST API

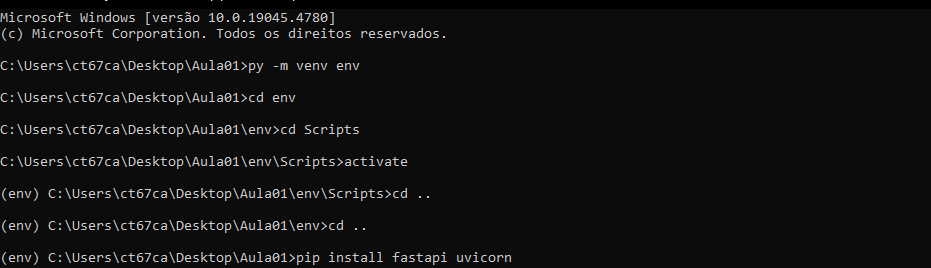
Comando que cria um ambiente virtual VENV (Não usar na pasta de compartilhado pq o ambiente é muito pesado, roda na área de trabalho) #quando a gente jogar essa pasta no compartilhado tem que tirar a venv dela pq é muito pesada.



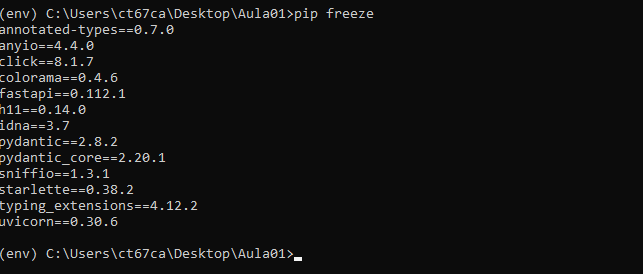
Ativar o ambiente virtual com a criação da pasta scripts tbm (quando o env aparece é o ambiente virtual foi ativado( você pode ativar com ‘env\scripts\activate’ também)



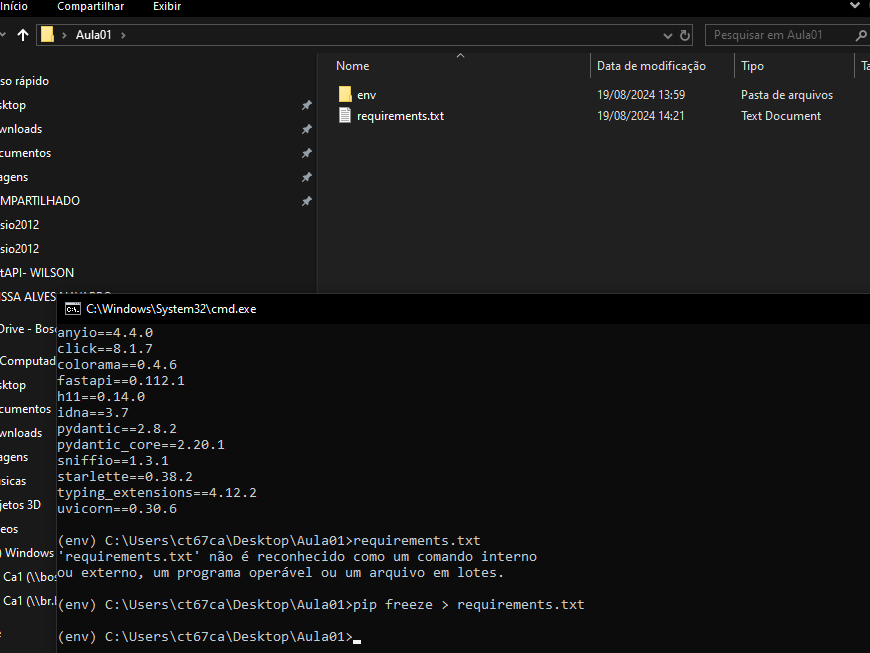
Voltar para a pasta inicial e baixar o fast api junto com o uvicorn (o pip permite isso) com o comando “cd ..” você consegue fazer isso



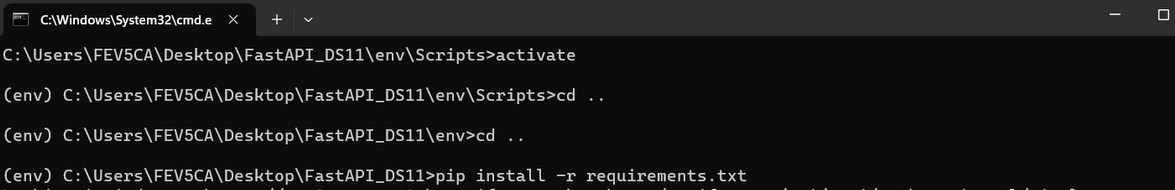
Esse comando mostra todas as bibliotecas que foram instaladas, mas você também pode olhar com o pip list e dir. a diferença do freeze para esses é que ele mostra a versão disponivel



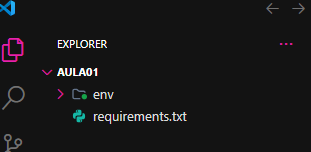
Comando que guarda em um arquivo txt as bibliotecas que tínhamos salvado antes( geralmente é chamado de Requirements)



Quando você tiver que criar a venv dnv por conta da pasta, para você não ter que instalar tudo dnv você usa outro comando (ele instala novamente e você tem que ativar a venv novamente)



Entrar no vscode pela pasta. (cmd e code .)



- Criar um arquivo de py fora da venv, tem que ser no arquivo da pasta mesmo.

- Importar a biblioteca do FastApi.

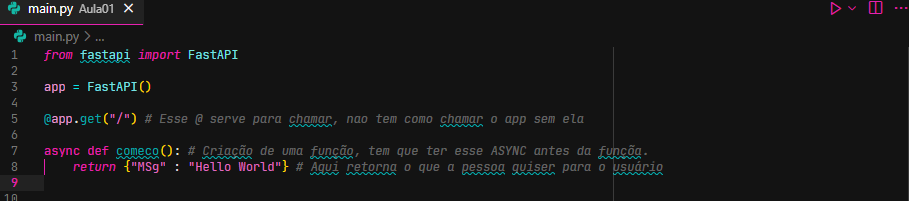
- Instanciar o FastApi com algum outro nome pra não ter que ficar digitando toda hora (app)



- Chamar a o FastApi com o que você quer que ela realize

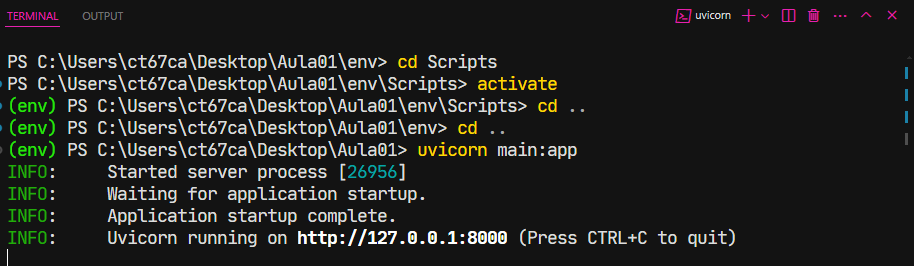
- Criar uma função usando async que é o padrão

- Retornar alguma coisa abaixo

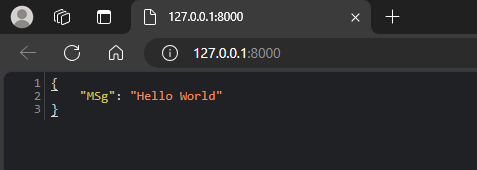


- Ativar a venv novamente no terminal do código

- Usar o uvicorn para ativar o código



- Puxar a url que foi dada e abrir no explorer

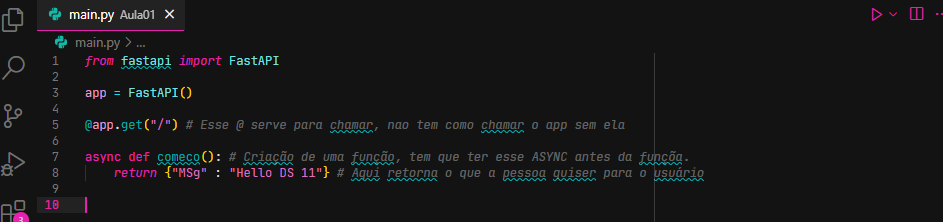


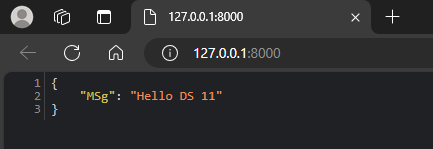
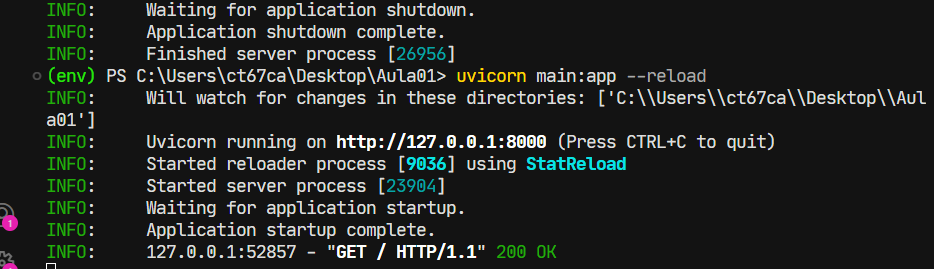
- Mudar a msg do return

- Dar Ctrl C no terminal para conseguir escrever novamente no terminal

- Dar o comando que atualiza a pagina com a modificação que você fez

- Visualizar como ficou a página

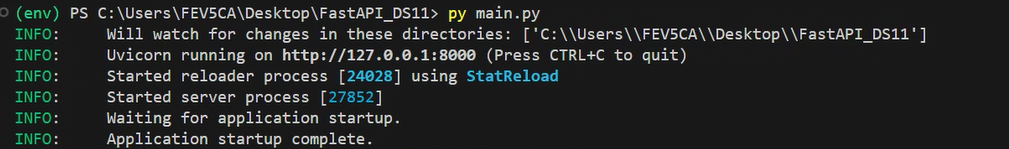




- Criar um if name pra não ter que fazer o uvicorn no terminal sempre

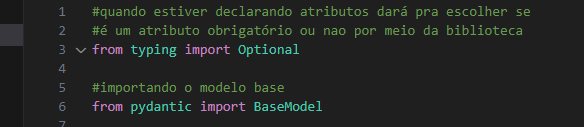


- Depois disso você só precisa chamar no terminal o py



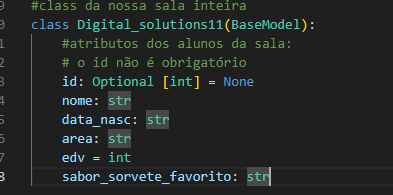
Vamos criar uma api da ds11:

1 passo: criar um arquivo model.py



Criando a class da turma inteira:

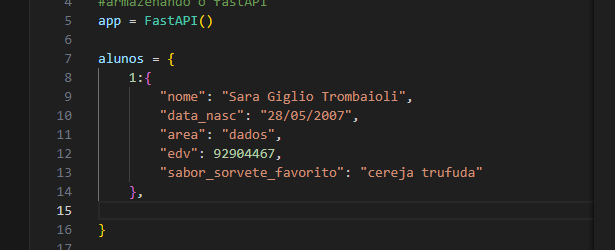


Dando atributos aos participantes da sala: Aqui esta todos atributos da turma ds11

Volta para o arquivo main.py

E faz como se fosse seu banco de dados, dessa forma

* A variavel tem que ser igual o do arquivo model – se colocou nome la – coloca nome aqui tb



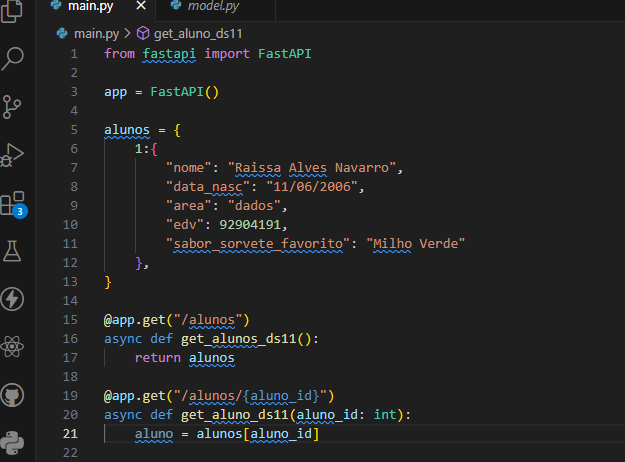
Agora, instalar o THUNDER CLIENT

- Criar ds11

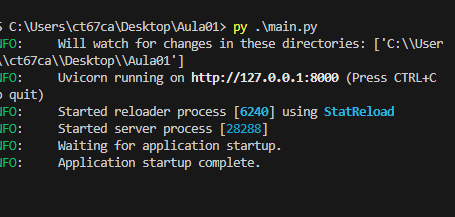
- Criar get alunos

- Colocar o endereço dado + /alunos

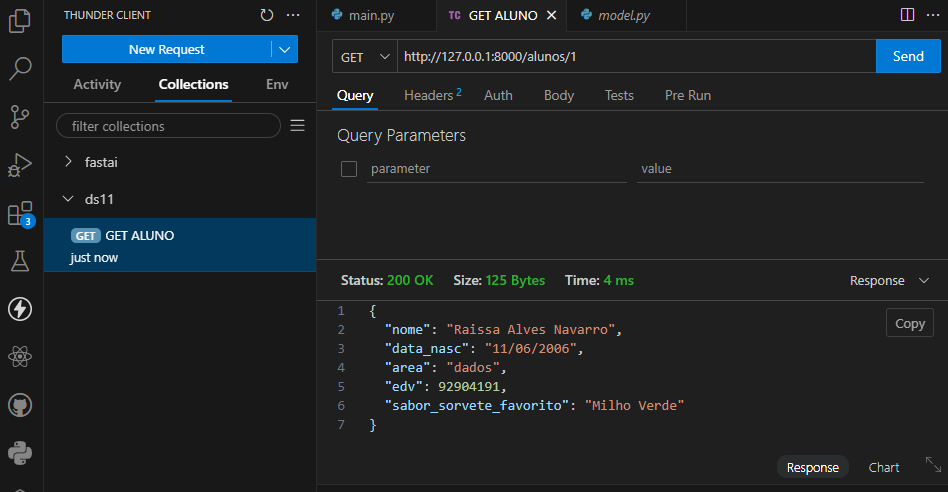
O app.get vai puxar o id de algum aluno e pegar daquela variavel de alunos, se tiver vários a pessoa vai passar um id e vai retornar o aluno que a pessoa quer ver as informações



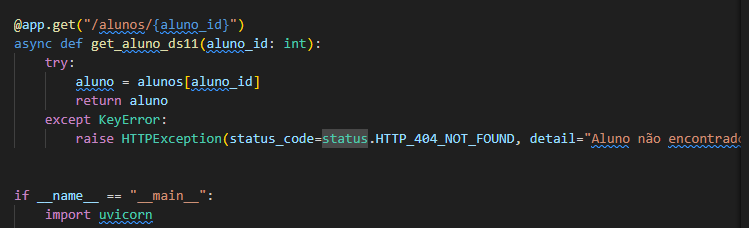
Tem que fazer esse comando do py main pra direcionar esse arquivo para o Thunder client



Entrar no Thunder Client em collections e criar um new request, colocando a url que o seu arquivo abre

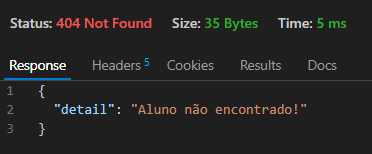


Aqui ele puxou o id 1 e mostrou esse aluno

- Aqui você vai um try para evitar erros e tem que importar mais algumas bibliotecas

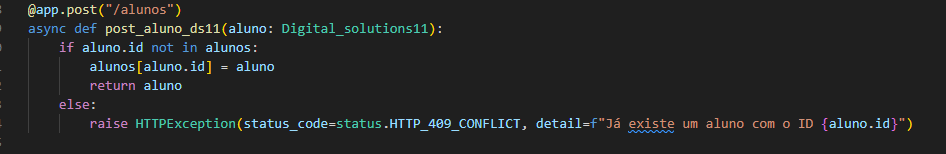


- Agora quando você for olhar no Thunder Client e procurar por um aluno que não existe o status vai ser 404 e vai te retornar que o aluno não foi encontrado.



Agora você cria outra função com async que vai pegar um aluno com o post e vai ter um if com a condição de se já existir esse aluno ele não é adicionado

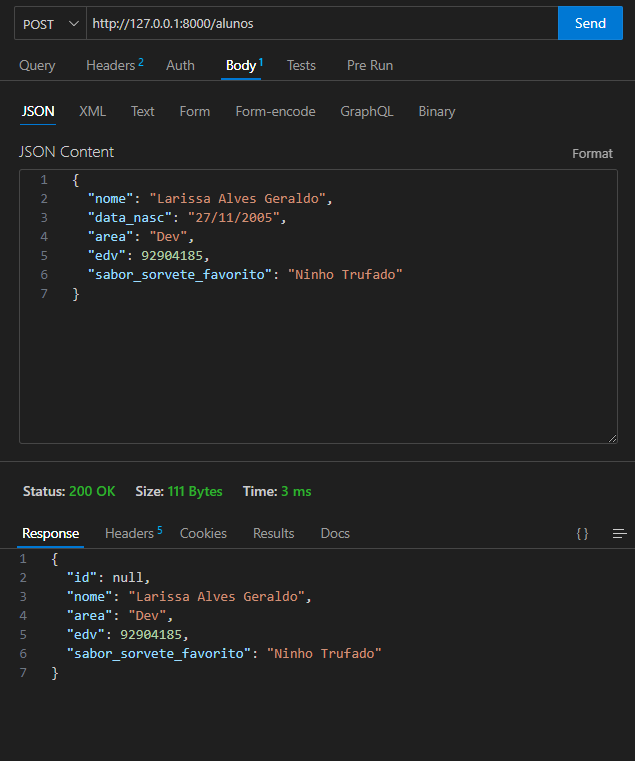
O 409 é um erro de quando da algum conflito no código, não é um urro mas é um conflito





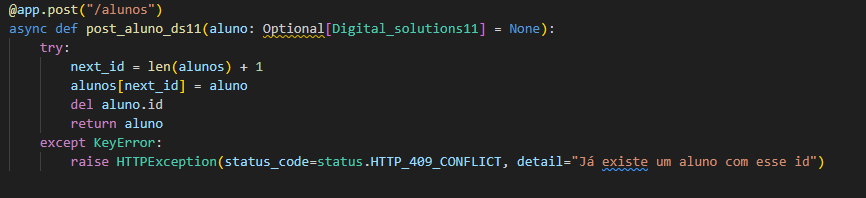
Você vai descobrir lá no Thunder Client se tem ou não esse aluno, o Thunder é como se fosse um inmsonnia do Lindomar so que dentro do vs code

-Aqui como não existe nenhum aluno com esses parâmetros ele cria nesse post aluno



Também possui outra forma de fazer essa busca pelo aluno e caso não tenha ele adiciona

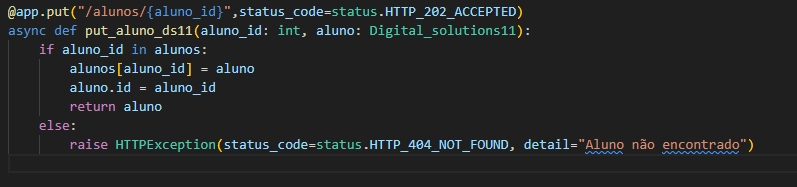
Nesse é utilizado o Optional, o try é usado para ver quantos alunos tem dentro de alunos somando mais 1, ele puxa ele aluno com o parâmetro e deleta o id dessa nova pessoa que esta sendo colocada pq as outras pessoas não tem e ai anula das pessoas novas que são adicionadas.



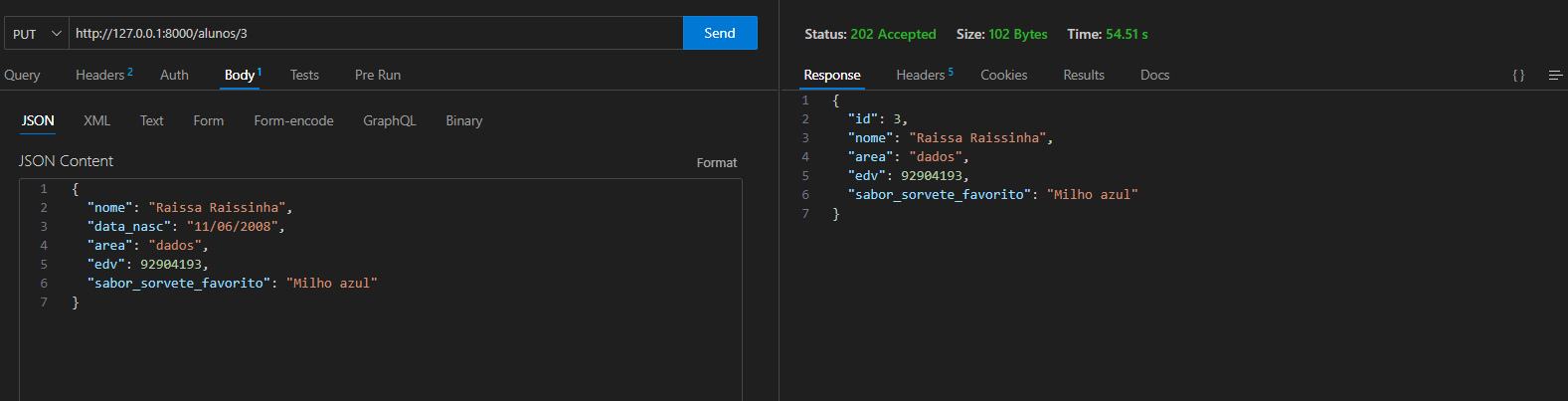
Colocar esse 201 é uma boa pratica pra aparecer que vc criou algo la no Thunder



O put serve para atualizar um arquivo ou um conteúdo, no nosso contexto seria atualizar um aluno já existente, eu posso mudar o edv do aluno, ou qualquer outro parâmetro dele através desse comando



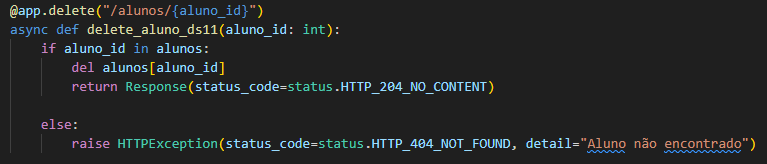
O put atualizou o meu id 2 para outro nome



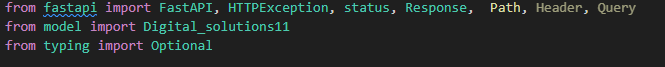
Tinha que ter importado o optional para funcionar



Para deletar um aluno da lista tem que criar essa função



Para o próximo passa tem que importar essas bibliotecas

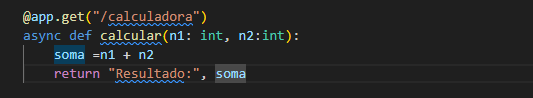


Esse Path serve para o usuário entender pq deu erro

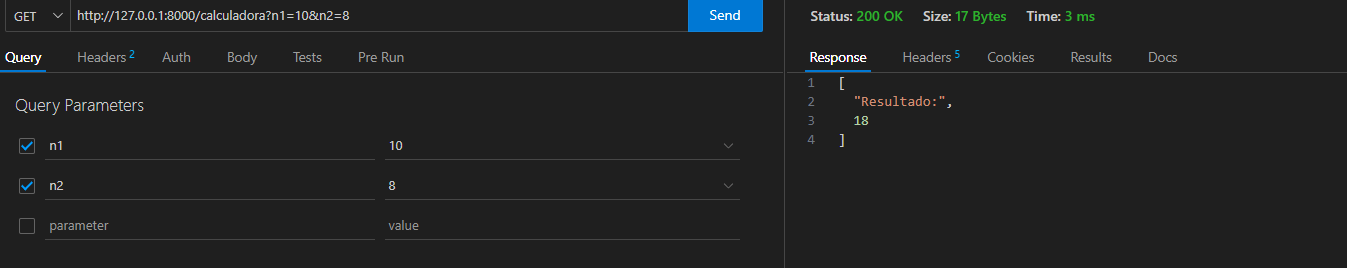


Vai aparecer isso, mostrando o motivo pelo qual deu problema

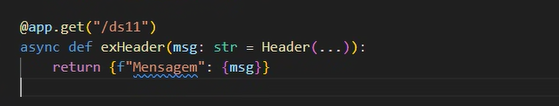


Tem que fazer função da calculadora 

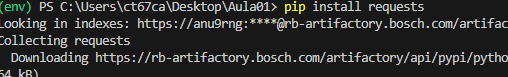
Para fazer uma calculadora usando



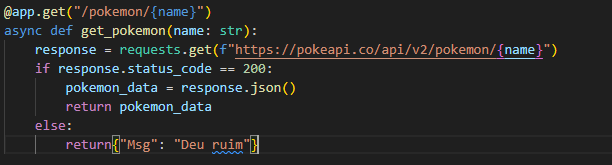
Essa parte do header



Instalar a biblioteca requests (já usei ela na área para filtrar infos das api)



Com esse requests.get eu consigo puxar um dado da api utilizando a url dela, quando eu entrar no Thunder Client e colocar o http com pokémon eu consigo puxar as info de um pokémon com o nome dele



Dando esse resultado

