# PROGRAMA STARTER
# FULL STACK WEB DEVELOPER

## Back-end

Atividade Prática - API Transações

## Termo de uso



## Objetivo do documento

Este material tem como objetivo descrever a atividade prática que realizaremos durante as aulas para fixação do conteúdo.

# Vamos praticar!

Chegou a hora de aplicar o conhecimento adquirido em nosso encontro. Lembrando sempre que os exercícios e desafios serão nossos principais indicadores sobre o conhecimento de vocês, tanto para ajudá-los como na hora do direcionamento para as vagas.

**Descrição**

Criar uma API REST para transações bancárias que fará o controle de valores recebidos e retirados de uma conta de um usuário. Para isso utilize node, typescript e express.

**Instruções**

- Todas as rotas deverão possuir validação dos parâmetros recebidos, retornando uma mensagem de erro e status adequado para a situação.

- Todas as rotas que possuírem consulta à alguma informação (GET, PUT, DELETE), deverão validar se o recurso acessado existe antes de efetuar a ação, retornando uma mensagem de erro e status adequado para a situação.

- Os usuários deverão ser salvos em um array específico, para serem utilizados nas demais rotas.

- Criar uma classe **User**, que deverá ter como propriedades **name**, **cpf**, **email**, **age** e **transactions** (sendo esse um array).

- Criar uma classe **Transaction**, que deverá ter como propriedades **title**, **value** e **type**.

- Ambas as classes **User** e **Transaction** deverão ter uma propriedade **id**, que deverá ser gerada automaticamente, sendo este um valor numérico único ou um UUID.

- POST /users: A rota deverá receber **name**, **cpf**, **email** e **age** dentro do corpo da requisição, sendo que o **cpf** deve ser único por usuário. Criar uma instância da classe **User** com os dados recebidos, e adicionar no array de usuários.
    - Esta rota deve conter um middleware para verificar se já existe um usuário com o CPF informado.

- GET /users/:id: A rota deverá retornar um único usuário de acordo com o parâmetro recebido. Não deverá retornar as transações do usuário nessa rota.

```
{
    "id": 1,
    "name": "Vinicius",
    "cpf": "000.000.000-00",
    "email": "email@email.com",
    "age": 18
}
```

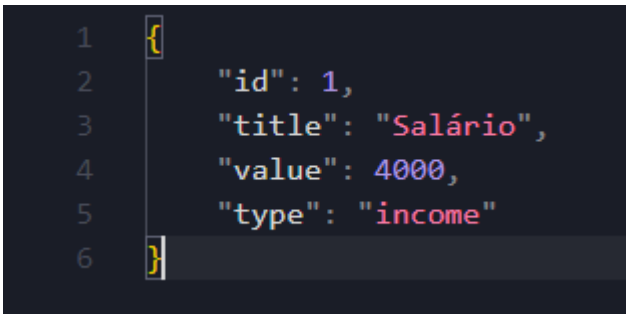
- GET /users: A rota deve retornar uma listagem com todos os usuários que você cadastrou até o momento. Esta rota deve poder filtrar os usuários pelo nome, email ou cpf. Não deverá retornar as transações do usuário nessa rota.

```
[
    {
        "id": 1,
        "name": "Vinicius",
        "cpf": "000.000.000-00",
        "email": "email@email.com",
        "age": 18
    },
    {
        "id": 2,
        "name": "Dirceu",
        "cpf": "000.000.000-00",
        "email": "email@email.com",
        "age": 21
    },
    {
        "id": 3,
        "name": "Roger",
        "cpf": "000.000.000-00",
        "email": "email@email.com",
        "age": 20
    }
]
```

- PUT/DELETE /users/:id: A rota deverá editar ou deletar usuários.

- POST /user/:userId/transactions: A rota deverá receber **title**, **value**, **type** dentro do corpo da requisição, sendo **type** o tipo da transação, que deve ter como valor de entradas **income** (depósitos) e **outcome** para saídas (retiradas). Criar uma instância da classe **Transaction**, e adicioná-la ao usuário responsável salvo anteriormente no array de usuários.

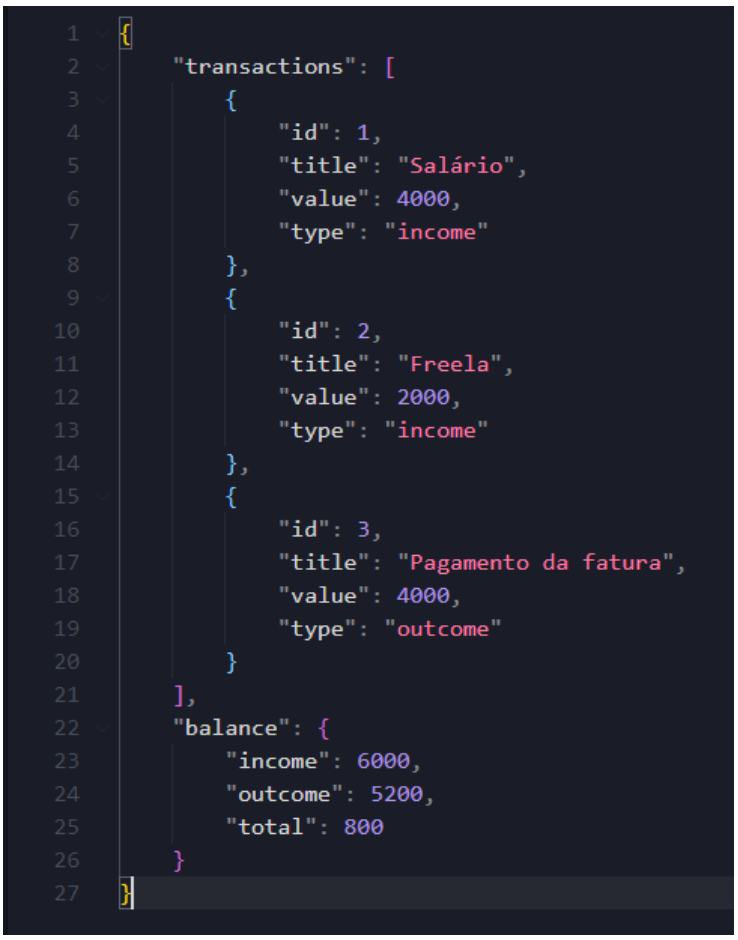
- GET /user/:userId/transactions/:id: A rota deverá retornar uma única transação cadastrada previamente



```
{
    "id": 1,
    "title": "Salário",
    "value": 4000,
    "type": "income"
}
```

- GET /users/:userId/transactions: A rota deverá retornar uma listagem com todas as transações que você cadastrou até o momento para um usuário específico, junto com o valor da soma de

entradas, retiradas e total de crédito. Esta rota deve poder filtrar as transações pelo título e tipo de transação.



```
{
    "transactions": [
        {
            "id": 1,
            "title": "Salário",
            "value": 4000,
            "type": "income"
        },
        {
            "id": 2,
            "title": "Freela",
            "value": 2000,
            "type": "income"
        },
        {
            "id": 3,
            "title": "Pagamento da fatura",
            "value": 4000,
            "type": "outcome"
        }
    ],
    "balance": {
        "income": 6000,
        "outcome": 5200,
        "total": 800
    }
}
```

- PUT/DELETE /users/:userId/transactions/:id: Devem editar ou deletar transações.

- Para todas as rotas envolvendo as transações devem conter um middleware para verificar se o usuário informado na rota existe.

"Este exercício deve ser postado na Class até o horário estipulado da tarefa na plataforma. Suba no Github e poste o link para o respectivo repositório."

Para que possamos construir uma base sólida de aprendizado é preciso praticar.

Bora implementar tudo isso!

BORA GROWDEVERS!

# PROGRAMA STARTER FULL STACK WEB DEVELOPER