AULA 04 – Vídeo 06

Desenvolvimento

Nesta aula, iremos implementar o método PUT, que servirá para edição dos perfis.

Para que possamos editar um perfil, a requisição deverá conter o ID do perfil, para que possamos saber qual perfil se quer editar, e os novos dados. O ID deverá ser passado por *query params*, de forma que nosso recurso será '/perfil/:id'. Os novos dados, por sua vez, deverão estar no corpo da requisição. Nosso código ficará assim:

```
    app.put("/perfil/:id", (req, res) => {
        let perfilID = req.params.id;
        let perfilEditado = req.body;

        if (perfilEditado) {
            let perfilIndex = perfis.findIndex((perfil) =>
perfil.id == perfilID);

            if (perfilIndex !== -1) {
                let perfilRetorno = perfis[perfilIndex];
                perfilEditado.id = perfilID;
                perfis.splice(perfilIndex, 1, perfilEditado);

                res.json(perfilRetorno);
            } else {
                res.json({
```

```
                    message: "Erro ao editar perfil : Perfil
não encontrado"
                });
            }
        } else {
            res.status(400).json({
                    message: "Erro ao editar perfil : Dados
incompletos"
            });
        }
    });
```

Tentemos entendê-lo.

Inicialmente, devemos perceber que se a requisição for feita para '/perfil' pelo método PUT sem o ID, a API retornará a informação "Cannot PUT /perfil". A API garante que a requisição conterá o ID, se for feita pelo método PUT. Assim sendo, na configuração da rota, podemos simplesmente acessar o ID passado na requisição pelo operador ponto. Colocamos o ID na variável 'perfilID':

```
    let perfilID = req.params.id;
```

Podemos também acessar os novos dados do perfil no *body* da requisição, também pelo operador ponto, e colocá-los na variável 'perfilEditado':

```
    let perfilEditado = req.body;
```

Agora, devemos garantir duas coisas: (i) que foram passados novos dados de perfil no *body* da requisição, ou seja, que o valor da variável 'perfilEditado' é diferente de nulo, de *undefined*, e de vazio; e (ii) que o ID passado se refere a um perfil existente em nosso banco de dados.

Em primeiro lugar, usamos uma decisão condicional para garantir a existência de 'perfilEditado':

```
    if (perfilEditado) {}
```

Em segundo lugar, buscamos em nossa base de dados um perfil cujo ID seja igual ao ID passado na requisição. Para isso, utilizamos o método de *array* 'findIndex()'. Esse método retorna o índice do elemento que obedece à condição passada como argumento por meio de uma função de callback. Colocamos o

resultado na variável 'perfilIndex'. Caso não seja encontrado um perfil que obedeça à condição, o resultado do método 'findIndex()' será -1:

```
    let perfilIndex = perfis.findIndex((perfil) => perfil.id
== perfilID);
```

Agora, podemos utilizar outra decisão, cuja condição será que 'perfilIndex' seja estritamente diferente de -1. Caso essa condição seja satisfeita, podemos proceder à edição dos dados na base de dados:

```
    if (perfilIndex !== -1) {}
```

Um comportamento comum às APIs, quando se trata de requisições pelo método PUT, é enviar como resposta o valor já existente na base de dados e alterar o valor na base de dados. Assim sendo, primeiramente guardamos o perfil já existente na variável 'perfilRetorno', antes de alterar os dados:

```
    let perfilRetorno = perfis[perfilIndex];
```

Antes de alterarmos os dados em nossa base de dados, devemos garantir que o novo perfil a ser inserido tem o mesmo ID. Podemos fazê-lo simplesmente atribuindo o valor de 'perfilID' ao atributo 'id' do objeto 'perfilEditado':

```
    perfilEditado.id = perfilID;
```

Para editarmos os dados, utilizamos o método de *arrays* 'splice()'. O método 'splice' recebe os seguintes parâmetros: o índice a partir do qual os elementos deverão ser removidos (em nosso caso, o valor de 'perfilIndex'), o número de elementos a serem removidos (ou seja, 1, pois queremos remover somente o perfil que estamos editando), e o dado que deverá ser inserido no lugar do elemento removido (ou seja, o 'perfilEditado'):

```
    perfis.splice(perfilIndex, 1, perfilEditado);
```

Finalmente, podemos implementar as respostas. Caso tudo tenha ocorrido sem problemas, a resposta é o perfil que já existia na base de dados:

```
    res.json(perfilRetorno);
```

Caso não tenha sido encontrado um perfil, a API retornará a mensagem de erro:

```
    res.json({
            message: "Erro ao editar perfil : Perfil não
encontrado"
                });
```

Por fim, caso os novos dados de perfil não tenham sido passados corretamente, a API retornará o *status* 400, erro de requisição, e a mensagem de erro:

```
    res.status(400).json({
       message: "Erro ao editar perfil : Dados incompletos"
            });
```

Feitas todas as implementações, podemos testar o recurso com o auxílio do Thunder Client. Tenha certeza de que o perfil que você está tentando editar existe na base de dados. Perceba que o perfil da Marcela, de ID 8, não foi implementado no *array* de perfis, e sim cadastrado por meio de uma requisição POST. Caso essa requisição não tenha sido feita, a requisição resultará em erro de perfil não encontrado. Uma requisição bem sucedida será semelhante a esta: