Etec
Bento Quirino
Campinas

Criando uma REST API com JSON Server – parte 2

Agora que já sabemos como criar uma *REST API fake* utilizando o *JSON Server*, em que de forma fácil e rápida conseguimos ter um *back-end* para nos ajudar no desenvolvimento *front-end*, nesse tutorial iremos aprender como gerar dados *fake* em massa. Assim nossa simulação, prototipação e testes ficarão mais precisos.

Sabemos que gerar uma quantidade grande de dados manualmente é penoso, ainda mais tendo que muitas vezes, inventar dados para cada situação, tais como para um cliente, primeiro nome, último nome, endereço, número de telefone, e-mail etc. Haja tanta criatividade e tempo, não é mesmo?

Para resolvermos essa situação, podemos usar duas bibliotecas do *JavaScript* que de forma rápida conseguimos gerar esses dados randomicamente e com a quantidade que desejarmos. Essas bibliotecas são *Faker* e *Lodash*. A primeira gera grandes quantidades de dados falsos (mas realistas) para teste e desenvolvimento. A segunda oferece modularidade, desempenho e extras, tornando o *JavaScript* mais fácil, eliminando o incômodo de trabalhar com *arrays*, números, objetos, *string* etc.

Instalando as bibliotecas

Abra um terminal na pasta *"bakend"* do seu projeto e digite o comando **npm install faker** e em seguida **npm install lodash** .

Em seguida crie um arquivo chamado **"generate.js"** dentro da pasta *"backend"* conforme demonstrado na figura 1.

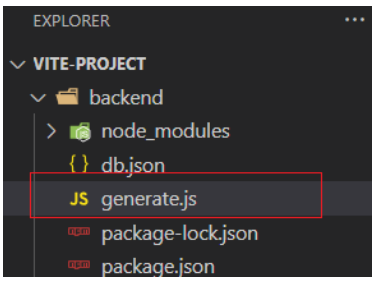

Figura 1 - Arquivo generate.js

Abra o arquivo **"generate.js"** e conforme a figura 2, digite o código que será explicado posteriormente.

```
backend > JS generate.js > ...
1   import { faker } from '@faker-js/faker/locale/pt_BR';
2   import lodash from 'lodash';
3   import fs from 'fs';
4
5   const peoples = lodash.times(50, function(n){
6       const firstName = faker.person.firstName();
7       const lastName = faker.person.lastName();
8       return {
9           id: n+1,
10          firstname: firstName,
11          lastname: lastName,
12          avatar: faker.image.avatar(),
13          address: faker.location.streetAddress(),
14          email: faker.internet.email({firstName: firstName.toLowerCase(), lastName: lastName.toLowerCase()})
15      }
16  });
17
18  const data = {};
19  data.peoples = peoples;
20  fs.writeFile('db.json', JSON.stringify(data), (err) => {
21      if(err) throw err;
22      console.log('Finalizado...');
23  });
```

Figura 2 - Código para gerar massa de dados fake

Explicando o código

Vamos à explicação, na **linha 1** é feita a importação da biblioteca *Faker* para geração de dados falsos. *Faker* oferece suporte a muitas localidades diferentes. Ao usar a instância padrão*, import {faker} from '@faker-js/faker'* você obtém dados em inglês. Para obter dados de uma localidade em português do Brasil, acrescente *'/locale/pt_BR'* na importação. Você poderá combinar várias localidades ou escolher a localidade desejada. Consulte o *link* https://fakerjs.dev/guide/localization.html para maiores informações quanto a localidades.

Na **linha 2** importamos a biblioteca *Lodash* para no ajudar na construção de nossos objetos que serão transformados em *JSON*. *Lodash* é ótimo para interagir com *arrays*, objetos e *string*, além de manipular e testar valores criando funções compostas. Para mais informações sobre *Lodash* visite o *site* https://lodash.com/ .

Terminando as importações, na **linha 3** importamos o *FS*, um módulo integrado do *Node.js* que fornece uma *API* para interagir com o sistema de arquivos do computador em que o *Node.js* está sendo executado. Ele permite a leitura, gravação, exclusão e manipulação de arquivos e diretórios. Para mais detalhes consulte a documentação disponível no *link* https://nodejs.org/docs/v0.3.1/api/fs.html .

Entre as **linhas 5 e 16** é construído uma lista de objetos com dados *fake* de pessoas usando o *Lodash* juntamente com o *Faker*. Na linha 5 usamos o método *times()* do *Lodash* onde é invocado *n* tempos de interação, retornando um *array* dos resultados de cada interação. Em nosso código estamos informando que ele terá 50 (cinquenta) interações e para cada interação executará uma função anônima que retornará um objeto *fake*. Você poderá informar a quantidade de interações necessárias para a geração de dados para os seus testes.

Nas **linhas 6 e 7** usamos o objeto o módulo *person* do *Faker* para gerar os dados *firstName* (primeiro nome) e *lastName* (último nome ou sobrenome) e armazenamos em duas constantes que serão utilizadas posteriormente. *Faker* possui vários módulos (categorias) de dados para serem gerados aleatoriamente tais como *Person*, *Finance*, *Image*, *Internet*, *Commerce*, *Company* e muitos outros, cada qual com seu conjunto de dados. Para saber quais dados *fake* você pode gerar com *Faker*, consulte sua *API* pelo *link* https://fakerjs.dev/api/ .

Seguindo as **linhas 9 a 14** é retornado dados *fakes* que irão compor uma pessoa.

Na **linha 18** é declarado uma constante **'***data'* que recebe uma lista de objetos vazia. Na sequência na **linha 19**, adicionamos um objeto **'***peoples'* a lista **'**data' atribuindo a lista de pessoas geradas pelo *Lodash*.

Finalmente na **linha 20**, usamos o método *fs.writeFile* para escrever no arquivo ***'db.json'*** a massa de dados gerada. Como o arquivo ***'db.json'*** deve ser composto por uma *String Json*, precisamos converter nossos dados gerados em *String*. Para isso usamos o método *stringify* do objeto *JSON*.

Executando o código

Com o entendimento do código, vamos executá-lo para gerar os dados no arquivo ***'db.json'*** que será usado pelo *JSON Serve*. Abra o terminal dentro da pasta *'backend'* e digite o comando

node generate.js

Pronto!!! O arquivo ***'db.json'*** foi atualizado com a massa de dados *fakes* conforme podemos verificar na figura 3. Basta agora executar o *JSON Serve* para servir uma *REST API* para ser consumida pela sua aplicação.

backend > {} db.json > [ ] peoples


```
{
  "peoples": [
    {
      "id": 1,
      "firstname": "Aline",
      "lastname": "Santos",
      "avatar": "https://avatars.githubusercontent.com/u/86743847",
      "address": "741 Lívia Avenida",
      "email": "aline.santos@live.com"
    },
    {
      "id": 2,
      "firstname": "Janaína",
      "lastname": "Pereira",
      "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/624.jpg",
      "address": "129 Núbia Rua",
      "email": "janaina16@gmail.com"
    },
    {
      "id": 3,
      "firstname": "Samuel",
      "lastname": "Albuquerque",
      "avatar": "https://avatars.githubusercontent.com/u/78643454",
      "address": "695 Emanuel Rua",
      "email": "samuel8@hotmail.com"
    },
    {
      "id": 4,
      "firstname": "Sara",
      "lastname": "Pereira",
      "avatar": "https://avatars.githubusercontent.com/u/48739768",
      "address": "28853 Silva Avenida",
      "email": "sara.pereira91@yahoo.com"
    },
```


Figura 3 - Resultado contendo a massa de dados gerada pelo Faker e Lodash

Concluindo...

Quando precisarmos de uma massa de dados para realizarmos a prototipação, apresentação ou até mesmo testes de nossas aplicações, podemos obtê-la por meio de dados fakes gerados aleatoriamente.

Nesse tutorial vimos como gerar essa massa por meio da biblioteca *Faker* juntamente com *Lodash*, de forma fácil e rápido gerando grandes quantidades de dados de diversos tipos, tais como dados pessoais, comercial, internet entre outros.

Existem várias situações a serem exploradas que não foram abordadas aqui, mas com esses conhecimentos básicos e um pouco de pesquisa, você poderá resolver qualquer situação. Explore mais sobre o tema.

Referências

Faker | Faker. Disponível em: <https://fakerjs.dev/>.
Lodash. Disponível em: <https://lodash.com/>.
EGGHEADIO. **Expert led courses for front-end web developers.** Disponível em: <https://egghead.io/lessons/javascript-creating-demo-apis-with-json-server>.