09

Gerando dados dinâmicos

O nosso objetivo é construir gráficos, para isso, precisamos de dados. Só que agora, os dados que alimentam o nosso gráfico serão atualizados constantemente, e, para não precisarmos fazer cada atualização na mão, podemos automatizar isso e fazer com que o gráfico seja atualizado na medida em que os dados mudarem.

Para isso, precisaremos de um local onde o Bytebank possa guardar esses dados e de onde também possamos pegar esses dados para alimentar o gráfico. Esse lugar é o servidor do Bytebank, que criamos com o código abaixo e hospedamos no Heroku.

```
const servidor = require('http').createServer();
const gerarValores = require('./gerarValores');
const socket = require('socket.io')(servidor);

setInterval(function repeticao(){
    socket.emit('atualizaBarras', gerarValores());
}, 5000);

servidor.listen(process.env.PORT || 3000);
```

E, para que o nosso cliente Bytebank possa visualizar o gráfico, precisamos do arquivo json. Esse json vai mudar o nosso html cada vez que fizermos uma tabela com os seus dados.

Já temos um servidor que está pronto com esses dados. Então iremos pegar o json e mudar alguns campos para que eles recebam esses dados dinâmicos.

Olhando para o nosso json, quais dados precisarão ser atualizados? Esses que estão dentro dos valores das células, certo? Então vamos separá-los dentro de um array chamado dados.

```
const dados = [
    {"c": [
    { "v": "João Antônio Marques", "f": null},
    { "v": 25110, "f": null},
    { "v": "R$25.110", "f": null}
    ]
    },

    {"c": [
        { "v": "Ana Siqueira", "f": null},
        { "v": 10020, "f": null},
        { "v": "R$10.020", "f": null}
        ]
    },

    {"c": [
        { "v": "Alcides V. de Oliveira", "f": null},
        { "v": 30240, "f": null},
        { "v": "R$30.240", "f": null}
        ]
    },
```

```
{"c": [
    { "v": "Rafael Antunes", "f": null},
    { "v": 2310, "f": null},
    { "v": "R$2.310", "f": null}
    ]
},

{"c": [
    { "v": "Bianca M. Sampaio", "f": null},
    { "v": 8542, "f": null},
    { "v": "R$8.542", "f": null}
    ]
},

{"c": [
    { "v": "Victor Barbosa", "f": null},
    { "v": 6521, "f": null},
    { "v": "R$6.521", "f": null}
    ]
},

{"c": [
    { "v": "Maria do Carmo Prado", "f": null},
    { "v": 12327, "f": null},
    { "v": "R$12.327", "f": null}
    ]
},

{"c": [
    { "v": "Maria Heloisa de Almeida", "f": null},
    { "v": 9764, "f": null},
    { "v": "R$9.764", "f": null}
    ]
},

{"c": [
    { "v": "Rita de Cássia Cuaron", "f": null},
    { "v": 20174, "f": null},
    { "v": "R$20.174", "f": null}
    ]
}];
```

Beleza, com isso temos a nossa estrutura geral, agora vamos criar uma função que vai gerar os novos valores. Vou chamá-la de gerarValores().

```
function gerarValores(){}
```

Essa função irá retornar as nossas colunas e os novos dados das linhas. Então, vou incluir as cols e rows do json dentro dela:

```
function gerarValores(){
    return {
    "cols":[
                {
```

```
                    "label": "Nome",
                    "type":"string",
                },

                {
                    "label": "Valor",
                    "type":"number"
                },

                {
                    "type": "string",
                    "role": "annotation",
                }
            ],

    "rows": []    }
};
```

Dentro das nossas linhas ou rows, precisamos escrever o código que irá gerar esses novos valores no json, e então na nossa tabela e no nosso gráfico. Para fazermos isso, vamos mapear esses dados novos em cima dos dados antigos, então vamos pegar o nosso array chamado dados e usar um método do javascript chamado mapear em inglês, map.

```
        "rows": dados.map( )
```

Dentro de map vamos escrever uma função que vai pegar cada item das nossas células, transformar o seu valor, somá-lo ao valor original para termos certeza de que estamos aumentando o número e então retorná-lo. Para melhorar o retorno desses dados, vamos deixar claro, tratando no servidor, que o último valor da célula é em reais.

```
   "rows": dados.map(function(item){
         const numero = parseFloat((Math.random()*10).toFixed(2));
         item.c[1].v += numero;
         item.c[2].v =  item.c[1].v.toLocaleString('pt-BR', { style: 'currency', currency: 'BRL'});
         return item;
})
```

Nosso json já está certo, o que precisamos fazer agora é ligar essa parte do código com o resto do código, então iremos escrever um código que possa usar essa função e executá-la

```
module.exports = function gerarValores(){
    return {
    "cols":[
                {
                    "label": "Nome",
                    "type":"string",
                },

                {
                    "label": "Valor",
                    "type":"number"
                },

                {
```

```
                "type": "string",
                "role": "annotation",
            }
        ],

    "rows": dados.map(function(item){
        const numero = parseFloat((Math.random()*10).toFixed(2));
        item.c[1].v += numero;
        item.c[2].v =  item.c[1].v.toLocaleString('pt-BR', { style: 'currency', currency: 'BRL'});
        return item;
        })
    }
```

E é assim que os nossos dados estão ficando dinâmicos lá no servidor do Bytebank.