

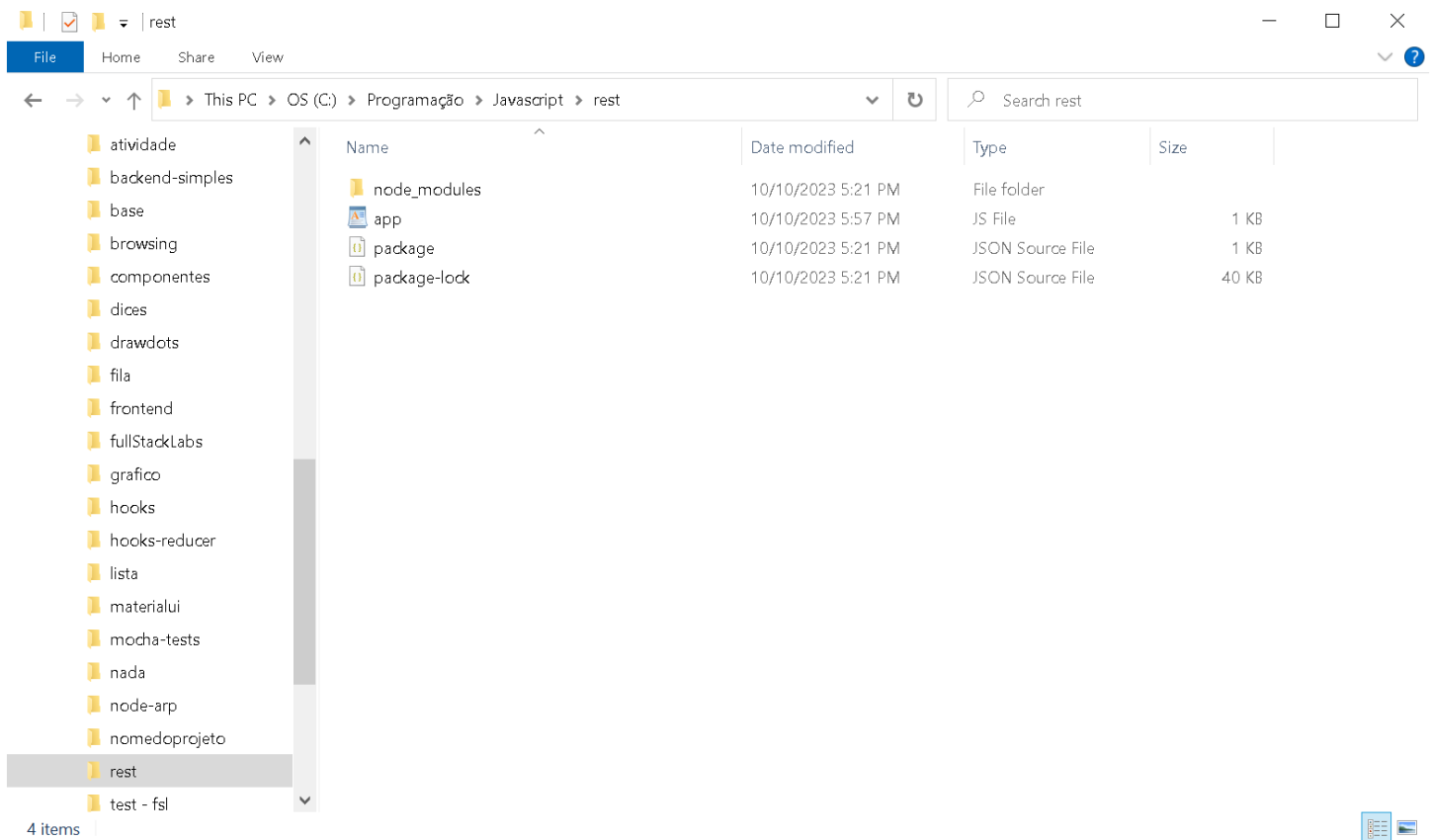
## PRÓXIMAS AULAS

- 31/10 - Arquitetura orientada a serviços (SOA) / Provisionamento e alocação de recursos em servidores / Infraestrutura em nuvem/cloud (Aplicação Prática);
- 07/11 - Web Services em Tempo Real - Web Socket (Aplicação Prática).

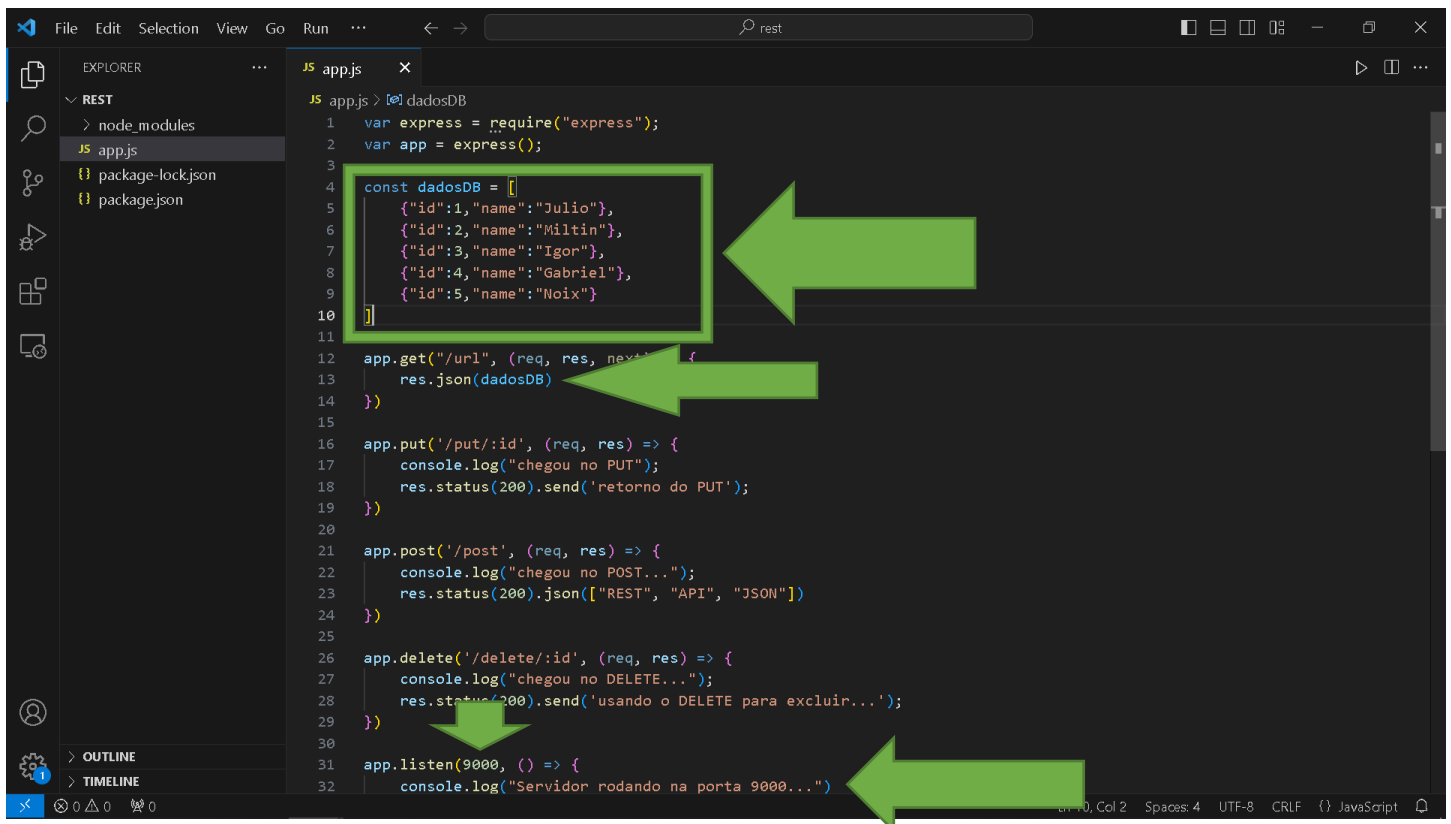
## AVALIAÇÃO V2

- 05/12 - V2 – APRESENTAÇÃO DE PROJETOS WEB (10,0)
- **Atividade:** criar um app web frontend e uma API separados, utilizando o conceito de desacoplamento entre frontend e backend para chamadas de APIs externas (básico). A atividade consiste em REALIZAR A INTEGRAÇÃO entre frontend e backend (com projetos independentes e separados, igual ao padrão de mercado). O servidor contém uma API básica com os métodos HTTP e recebe chamadas que são direcionadas/respondidas através das rotas. A tela do frontend deve ser capaz de captar os dados preenchidos pelo usuário e enviá-los através de um JSON para a API no servidor e receber uma resposta;
- **Conforme vastamente informado a todos os alunos nas duas aulas iniciais da disciplina, todas as atividades realizadas durante o semestre poderão ser feitas EM QUALQUER LINGUAGEM/Framework/IDE/TECNOLOGIA QUE OS ALUNOS DOS GRUPOS DESEJAREM, DESDE QUE DEMONSTREM AS ATIVIDADES NA PRÁTICA DURANTE O HORÁRIO DAS AULAS:**
- São utilizados, nessa aula, conceitos/desenvolvimentos similares aos das aulas 5, 6, 9 e 10;
- A partir da próxima página, estão instruções de como realizar esse desenvolvimento através dos frameworks React JS e Node JS;
- **OBSERVAÇÃO:** existe um outro arquivo que também trás uma forma similar de arquitetura monolítica ([https://docs.google.com/document/d/12nWK3x\\_3i-7c9IWzcbpDXUK02knBQ2in/edit?usp=drive\\_web&oid=113053781242364047423&rtpof=true](https://docs.google.com/document/d/12nWK3x_3i-7c9IWzcbpDXUK02knBQ2in/edit?usp=drive_web&oid=113053781242364047423&rtpof=true)). Entretanto, esse formato não atende aos objetivos dessa atividade (**NEM DA V2**), que foca em fazer projetos independentes de frontend e backend.

⇒ Vá até a pasta onde o projeto da **AULA 9** foi desenvolvido:



⇒ Abra o projeto no VS Code e altere a porta (utilizamos a 9000, para não dar conflito) e realize os ajustes abaixo:



⇒ Acesse a pasta do projeto da AULA 9 no terminal/cmd e rode o projeto:

```
C:\WINDOWS\system32\cmd.exe - node app.js
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\juiun>cd C:\Programação\Javascript\rest
C:\Programação\Javascript\rest>node app.js
Servidor rodando na porta 9000...
```

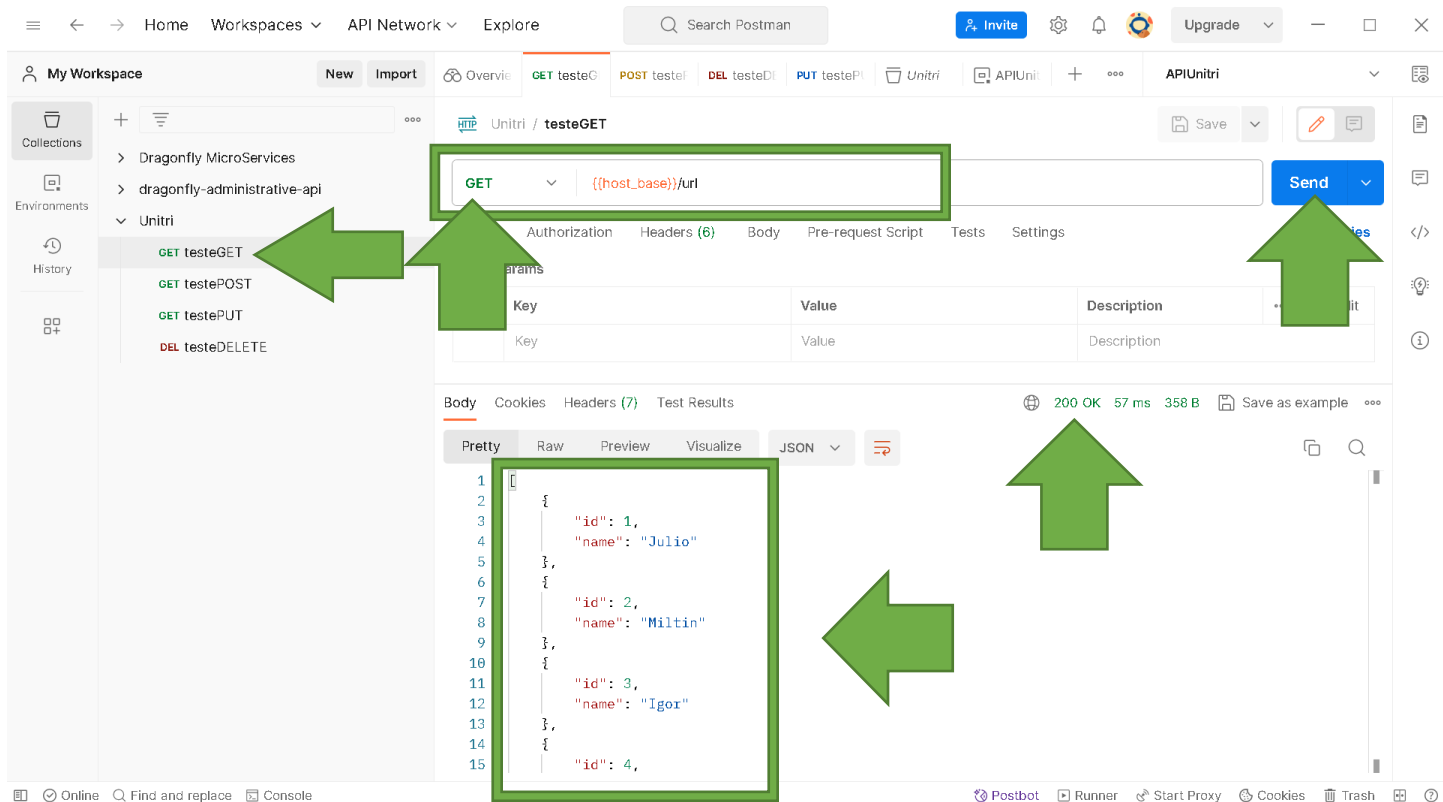
⇒ Ajuste o “environment” para a nova porta 9000 do servidor/API:

The screenshot shows the Postman API Platform interface. On the left, under 'My Workspace', the 'APIUnitri' environment is selected. The main panel displays the 'APIUnitri' environment variables table.

Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/> host_base	default	http://localhost:9000	http://localhost:9000

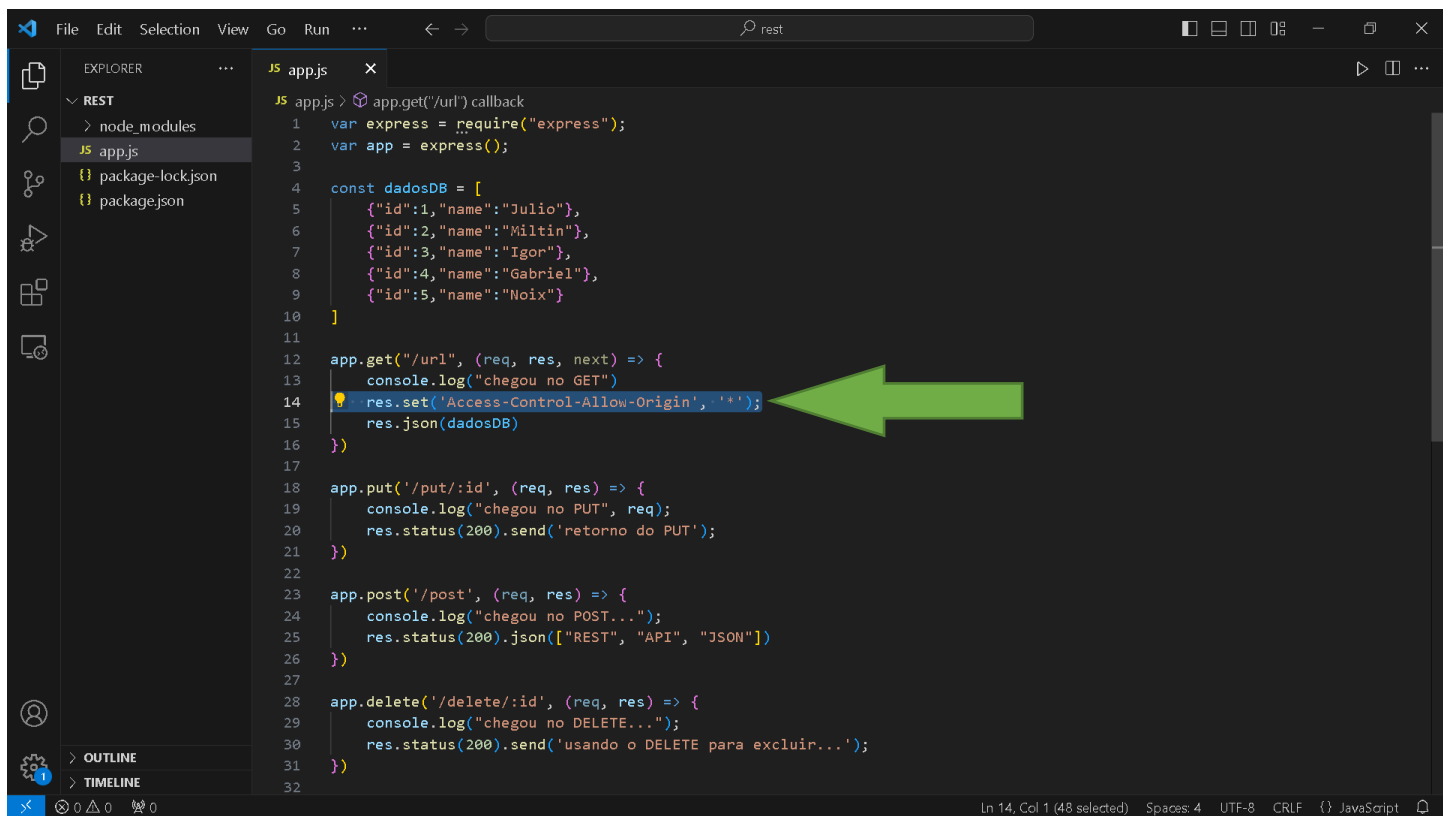
Below the table, a message box states: 'Use variables to reuse values and protect sensitive data'. It includes links for 'variable type' and 'variable values'.

⇒ Ajuste o “environment” para a nova porta 9000 do servidor/API:

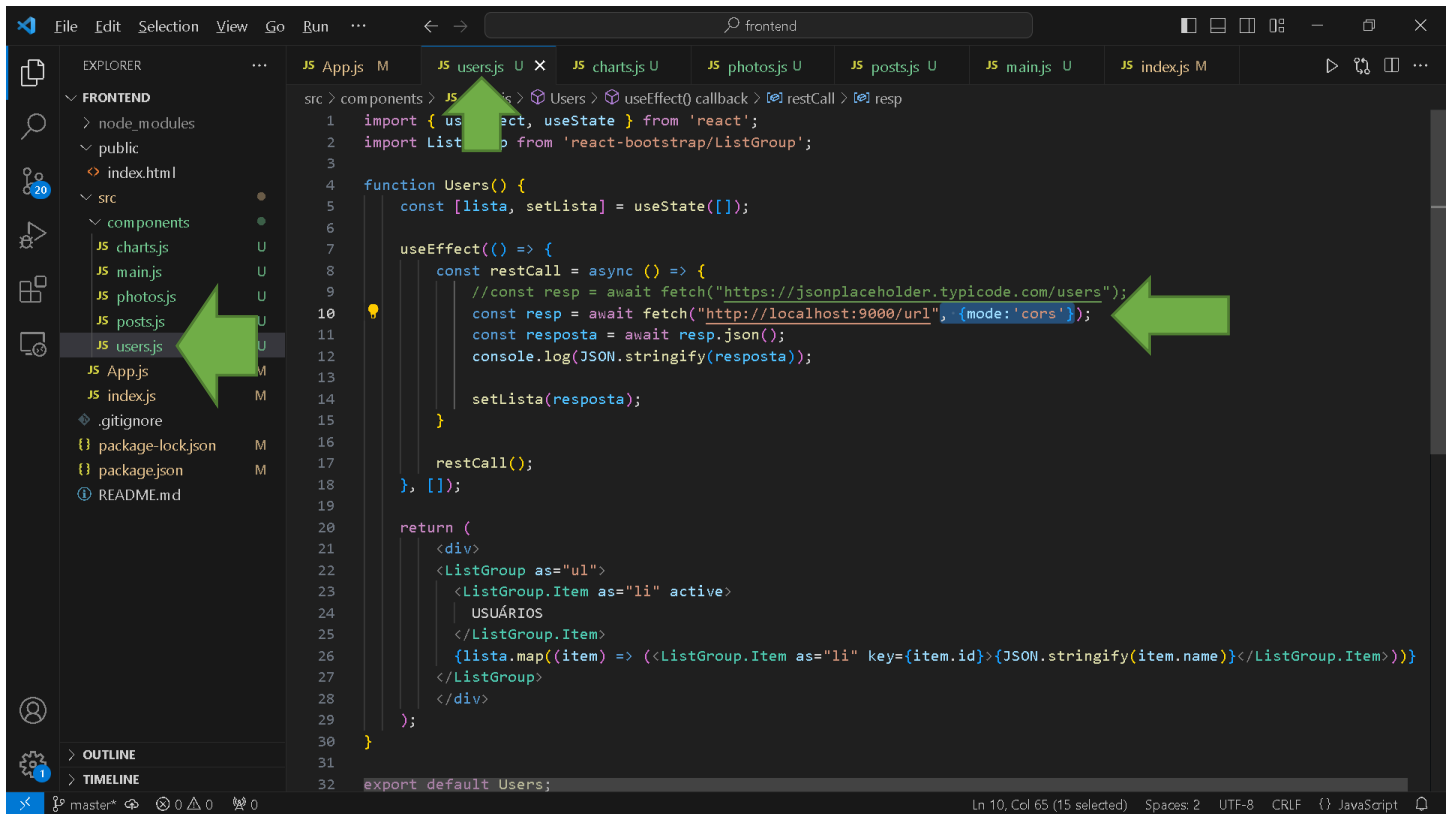


⇒ **PARA RODAR FRONTEND E BACKEND NO MESMO COMPUTADOR, É NECESSÁRIO REALIZAR A “LIBERAÇÃO” DO CORS:**

<https://www.stackhawk.com/blog/react-cors-guide-what-it-is-and-how-to-enable-it/>

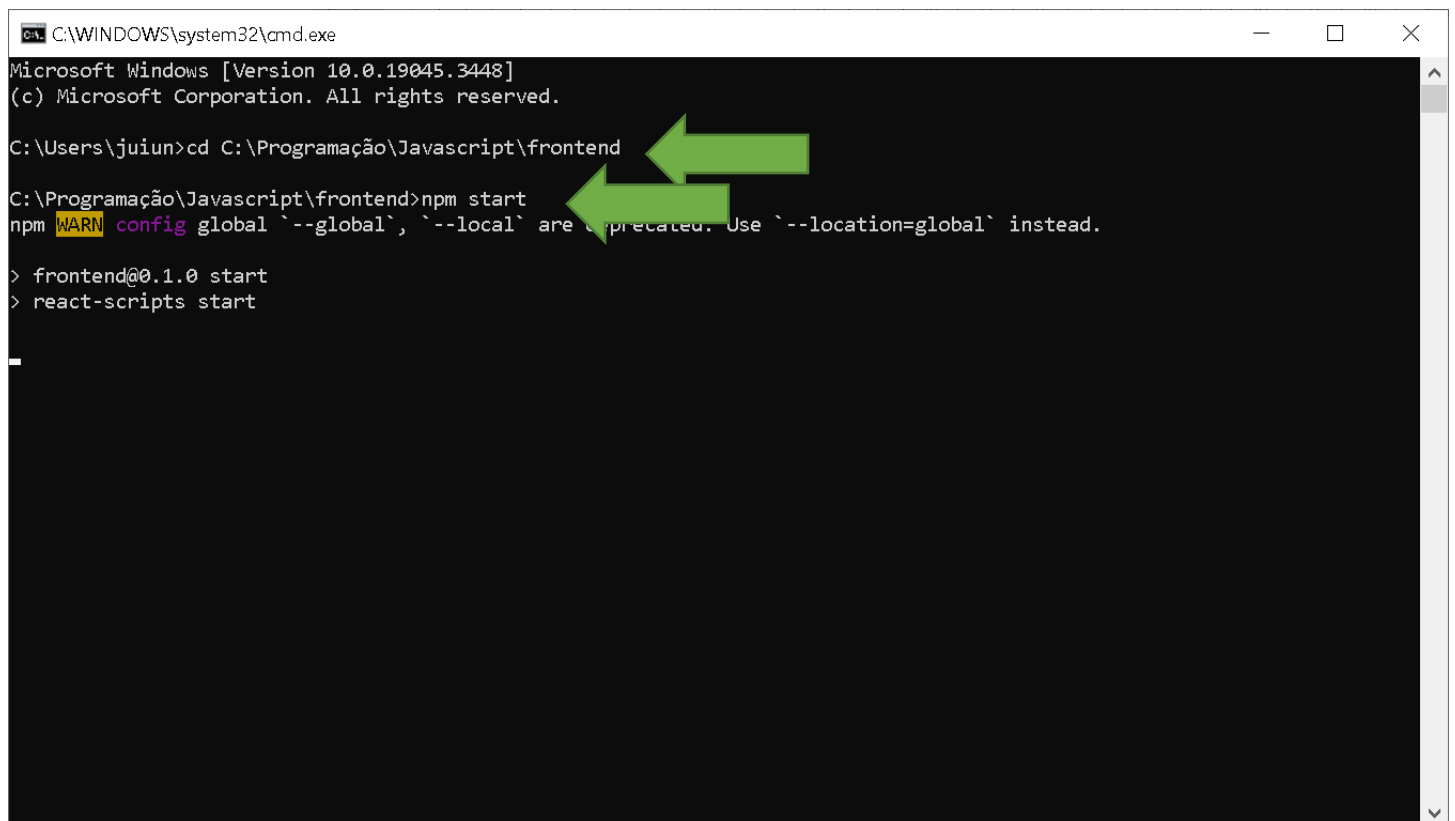


⇒ Abra o projeto da **AULA 6** em outra janela do VS Code e realize os seguintes ajustes no arquivo/componente “user.js”:



```
1 import { useState, useEffect } from 'react';
2 import ListGroup from 'react-bootstrap/ListGroup';
3
4 function Users() {
5   const [lista, setLista] = useState([]);
6
7   useEffect(() => {
8     const restCall = async () => {
9       //const resp = await fetch("https://jsonplaceholder.typicode.com/users");
10      const resp = await fetch("http://localhost:9000/unl", {mode: 'cors'});
11      const resposta = await resp.json();
12      console.log(JSON.stringify(resposta));
13      setLista(resposta);
14    }
15    restCall();
16  }, []);
17
18  return (
19    <div>
20      <ListGroup as="ul">
21        <ListGroup.Item as="li" active>
22          USUÁRIOS
23        </ListGroup.Item>
24        {lista.map((item) => (<ListGroup.Item as="li" key={item.id}>JSON.stringify(item.name)</ListGroup.Item>))}
25      </ListGroup>
26    </div>
27  );
28}
29
30 export default Users;
```

⇒ Rode o projeto da **AULA 6** com os ajustes acima:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\juun>cd C:\Programação\Javascript\frontend

C:\Programação\Javascript\frontend>npm start
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> frontend@0.1.0 start
> react-scripts start
```

⇒ Visualize os ajustes pelo navegador:

EMPRESA

USUÁRIOS

"Julio"
"Miltin"
"Igor"
"Gabriel"
"Noix"

LISTA DE USUÁRIOS (NOMES) DO OBJETO RECEBIDO

O OBJETO COMPLETO RECEBIDO PODE SER VISTO AO TECLAR "F12" NO BROWSER

```
[{"id":1,"name":"Julio"}, {"id":2,"name":"Miltin"}, {"id":3,"name":"Igor"}, {"id":4,"name":"Gabriel"}, {"id":5,"name":"Noix"}]
```

⇒ Pare o servidor e realize os ajustes abaixo no método POST para imprimir o objeto recebido:

```
1 var express = require("express");
2 var app = express();
3 app.use(express.json({type: '*/*'})); //instrução necessária para ver o LOG do JSON que chega do frontend
4
5 const dadosDB = [
6   {"id":1,"name":"Julio"},
7   {"id":2,"name":"Miltin"},
8   {"id":3,"name":"Igor"},
9   {"id":4,"name":"Gabriel"},
10  {"id":5,"name":"Noix"}
11 ]
12
13 app.get("/url", (req, res, next) => {
14   console.log("chegou no GET")
15   res.set('Access-Control-Allow-Origin', '');
16   res.json(dadosDB)
17 })
18
19 app.put('/put/:id', (req, res) => {
20   console.log("chegou no PUT", req);
21   res.status(200).send('retorno do PUT');
22 })
23
24 app.post('/post', (req, res) => {
25   console.log("chegou OBJETO no POST: ", req.body);
26   res.set('Access-Control-Allow-Origin', '');
27   res.status(200).json(['ENVIANDO UM JSON'])
28 })
29
30 app.delete('/delete/:id', (req, res) => {
31   console.log("chegou no DELETE...");
32   res.status(200).send('usando o DELETE para excluir...');
```

⇒ Rode o servidor novamente:

```
C:\WINDOWS\system32\cmd.exe - node app.js
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\juiun>cd C:\Programação\Javascript\rest

C:\Programação\Javascript\rest>node app.js
Servidor rodando na porta 9000...
chegou no GET
chegou no GET
^C
C:\Programação\Javascript\rest>node app.js
Servidor rodando na porta 9000...
```

⇒ Realize os seguintes ajustes no arquivo/componente “user.js”:


```
1 import { useEffect, useState } from 'react';
2 import ListGroup from 'react-bootstrap/ListGroup';
3
4 const dadosCliente = {
5   email: "julio.telles@gmail.com",
6   senha: "12345abcde",
7   novoRegistro: false
8 }
9
10 function Users() {
11   const [lista, setLista] = useState([]);
12
13   useEffect(() => {
14     const restCall = async () => {
15       //const resp = await fetch("https://jsonplaceholder.typicode.com/users");
16       const resp = await fetch("http://localhost:9000/url", {mode:'cors'});
17       const resposta = await resp.json();
18       console.log(JSON.stringify(resposta));
19
20       setLista(resposta);
21
22       resp = await fetch("http://localhost:9000/post",
23       {
24         method: 'post',
25         body: JSON.stringify(dadosCliente)
26       }, {mode:'cors'});
27       resposta = await resp.json();
28       alert(JSON.stringify(resposta));
29     }
30     restCall();
31   });
32 }
```

⇒ Observe a chamada chegando na API do servidor:

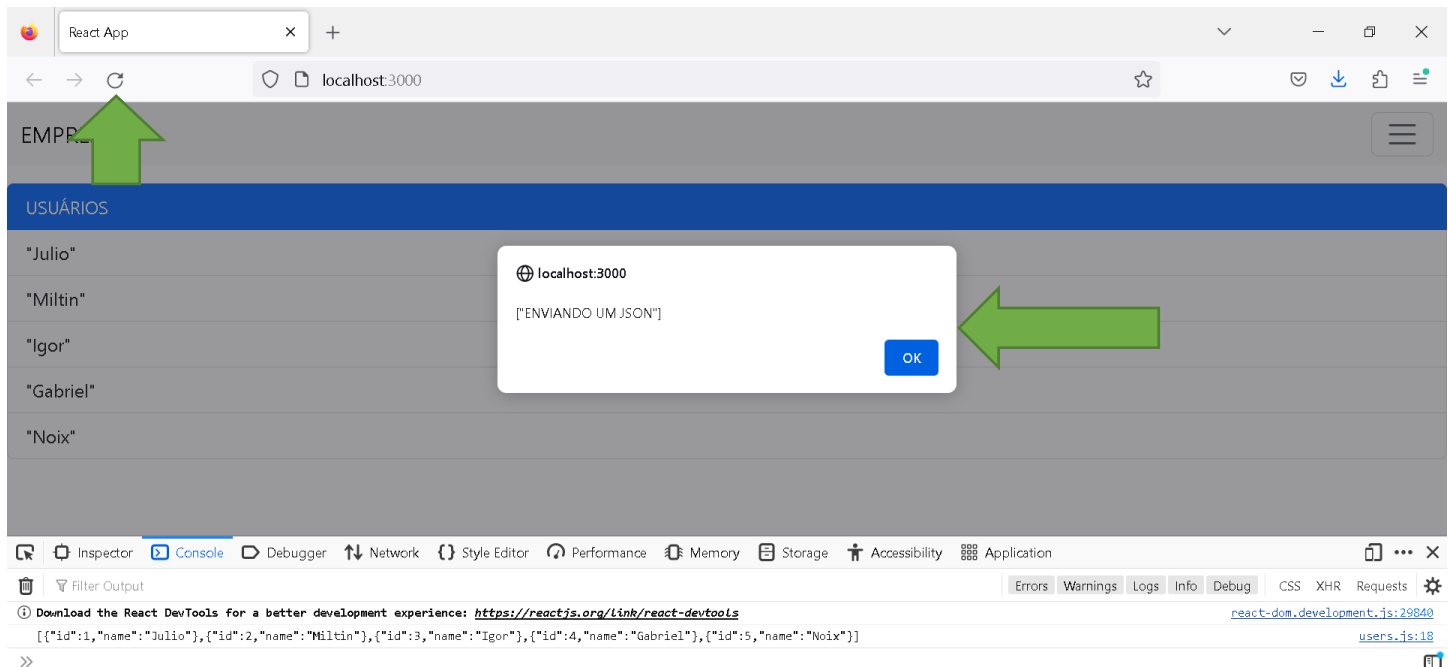
```
C:\WINDOWS\system32\cmd.exe - node app.js
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\juiun>cd C:\Programação\Javascript\rest

C:\Programação\Javascript\rest>node app.js
Servidor rodando na porta 9000...
chegou no GET
chegou no GET
^C
C:\Programação\Javascript\rest>node app.js
Servidor rodando na porta 9000...
chegou no GET
chegou OBJETO no POST: {
  email: 'julio.telles@gmail.com',
  senha: '12345abcde',
  novoRegistro: false
}
```



⇒ Observe o retorno da API:



⇒ **A partir desse ponto, já é possível observar a integração completa entre FRONTEND e BACKEND. O aluno, já está apto a desenvolver o aplicativo completo básico “full stack”. Prossiga com o desenvolvimento do banco de dados e integração com API para poder finalizar o projeto, assim como a troca de dados entre front e backend.**