

# PROGRAMAÇÃO WEBII

1

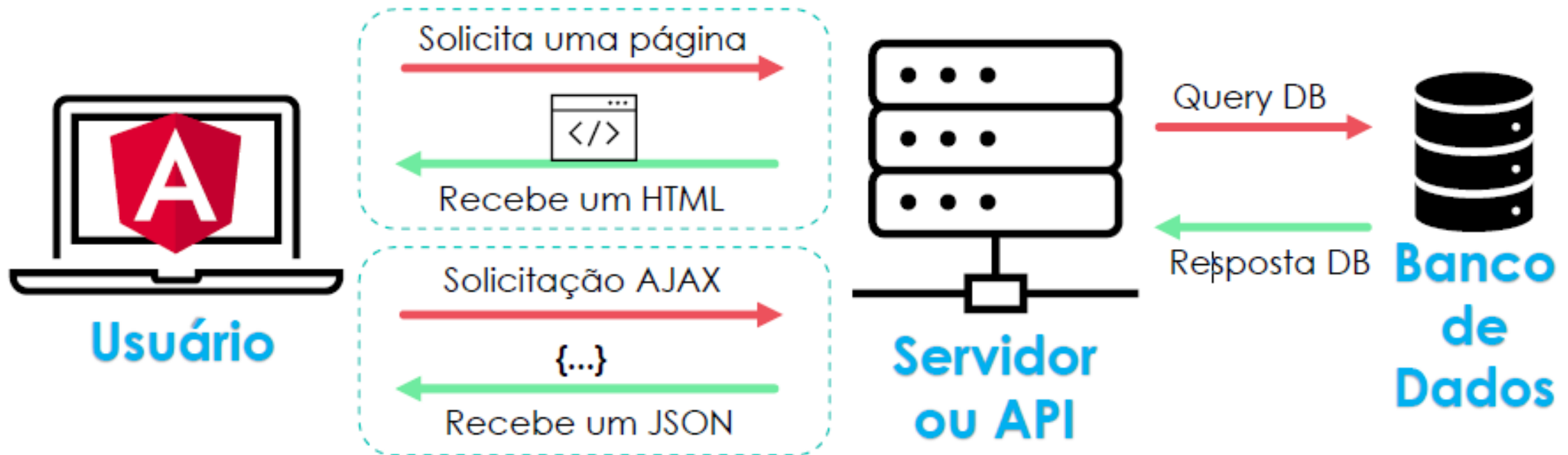
# Criando uma API REST de testes com JSO-SERVER

Projetos Iniciais do Angular

# Relembrando... **Funcionamento - SPA**

- Uma única página é carregada com todos os códigos HTML, CSS e Javascript(JS) que compõe a aplicação completa.
- Embora a URL mude, somente os dados e imagens são carregados dinamicamente, geralmente recebidos no formato JSON através de AJAX.
- Todo o processamento ocorre no lado do cliente, utilizando APIs para comunicação com o bancos de dados.

# Relembrando... Funcionamento - SPA



# Projeto JSON Server

<https://github.com/typicode/json-server>

routes.json

Update home

☰ README.md

## JSON Server

build passing

npm package 0.16.3

Get a full fake REST API with **zero coding** in **less than 30 seconds** (seriously)

Created with <3 for front-end developers who need a quick back-end for prototyping and mocking.

- [Egghead.io free video tutorial - Creating demo APIs with json-server](#)
- [JSONPlaceholder - Live running version](#)
- [My JSON Server](#) - no installation required, use your own data

See also:

- 🐶 [husky](#) - Git hooks made easy
- 🦊 [lowdb](#) - local JSON database
- ❌ ✅ [xv](#) - the most minimalist test runner

# Vamos lá!

- Abrir o prompt de comando;
- Ir para a pasta do diretório criado por vocês.

# Vamos lá!

- Vamos criar dois projetos: backend e frontend
- Vamos criar agora, nosso projeto backend:
- Dentro da pasta angular, criar a pasta backend

▷ C:\angular>MD backend <enter>

▷ Cd\backend <enter>

▷ Vamos criar o projeto do tipo node:

▷ npm init -y <enter>  -y -> cria com todas as configurações padrão

```
Wrote to C:\angular\cursoCPS\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

# Vamos lá!

- ▶ Vamos instalar o JSON SERVER.
- ▶ Se vc for na página deles, eles indicam que vc faça uma instalação global, mas como queremos algo pontual, vamos aprender como fazer um json server somente para esse projeto
- ▶ Acesse a documentação anterior:

## json-server

Node.js CI **passing**

 **Important**

Viewing alpha v1 documentation – usable but expect breaking changes. For stable version, see [here](#)



# Vamos lá!

- ▶ Copie a primeira linha do código, tirando o `-g` (global):

## Getting started

---

Install JSON Server

```
npm install -g json-server@0.17.4    # NPM  
yarn global add json-server@0.17.4    # Yarn  
pnpm add -g json-server@0.17.4        # PNPM
```

# Vamos lá!

- Estando na pasta do projeto backend, digite o comando abaixo:
  - ▷ `npm i json-server@0.17.4 <ENTER>`
    - ▷ \*\* Irá instalar json-server
    - ▷ Ficará disponível só no meu projeto.
    - ▷ Aguardar a instalação.
- \*\* Lembrando sempre ... Não pode clicar dentro do terminal na tela senão o processo para até que alguma tecla libere novamente (tecla para cima, direita..).

# Vamos lá!

Instalação finalizada:

```
C:\angular\cursopwebII\backend>npm i json-server
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN backend@1.0.0 No description
npm WARN backend@1.0.0 No repository field.

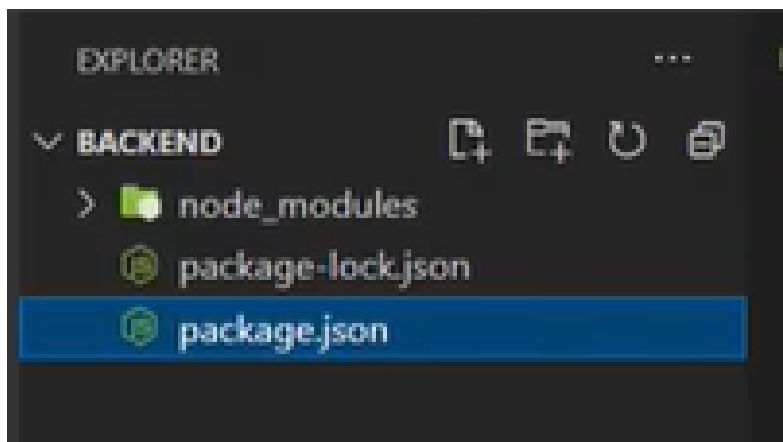
+ json-server@0.16.3
added 182 packages from 80 contributors and audited 182 packages in 8.256s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

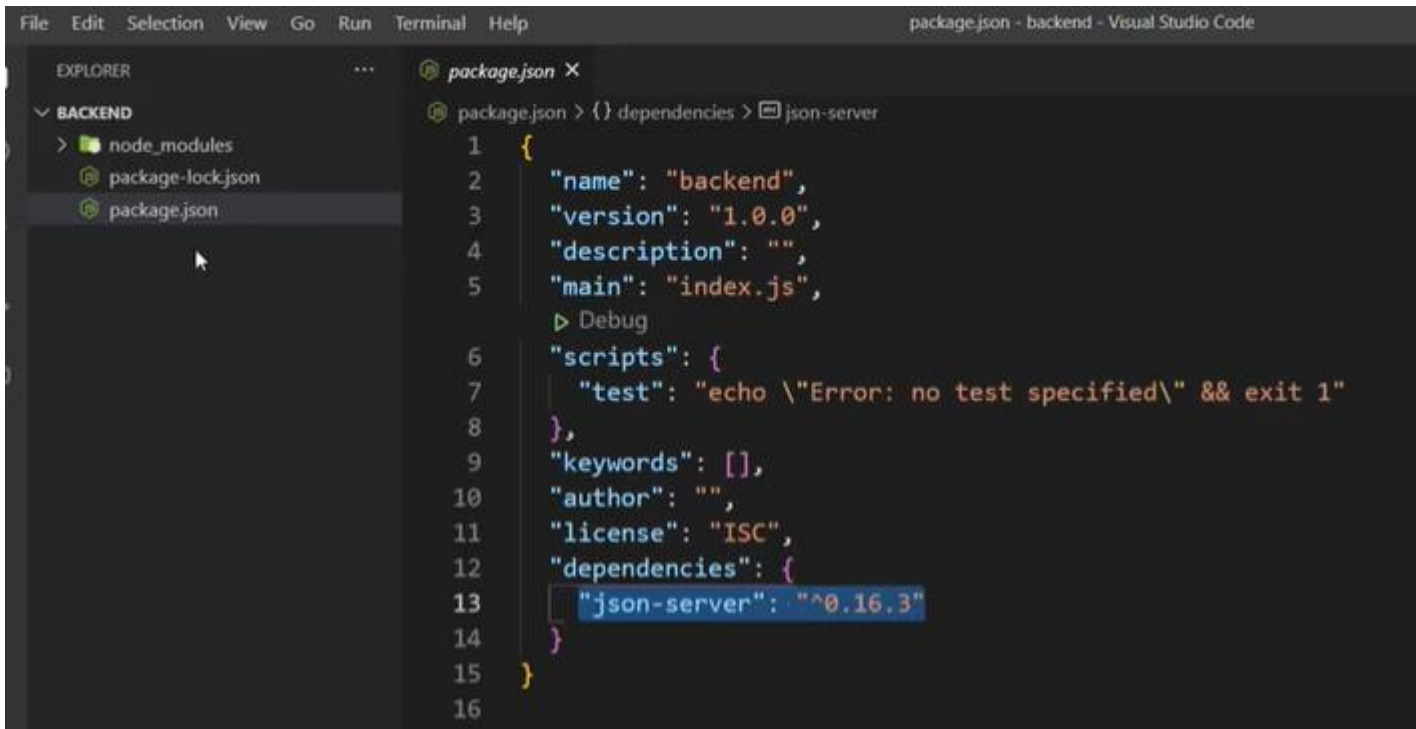
# Vamos lá!

- Vamos abrir o visual code dentro da pasta:
  - ▶ `code .`
- Vcs irão ver agora que temos um `package.json` que é onde temos as configurações dos nossos projetos.



# Vamos lá!

Inclusive já adicionou como dependência o JSON Server:

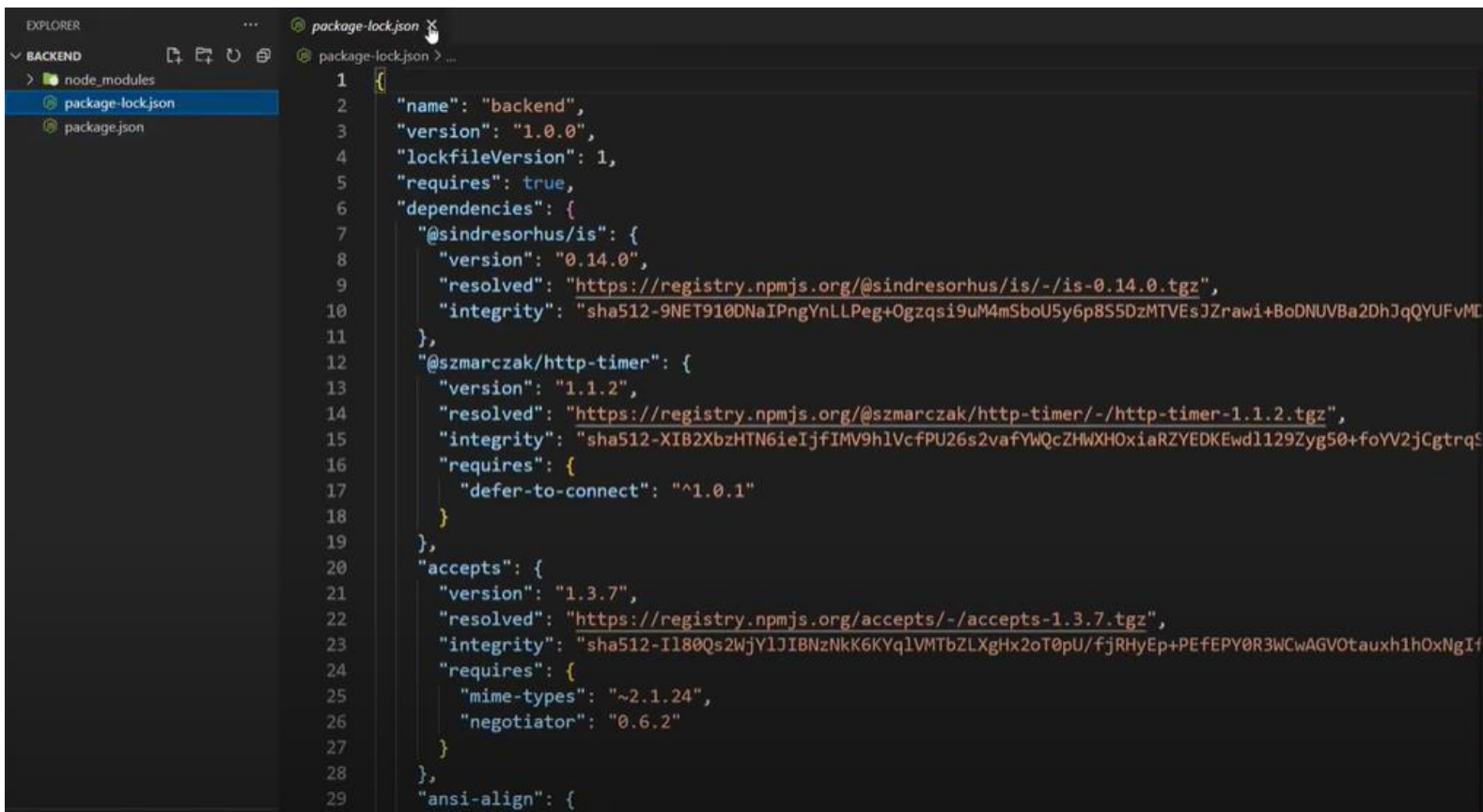


The screenshot shows the Visual Studio Code interface with the `package.json` file open. The Explorer sidebar on the left shows the project structure with `node_modules`, `package-lock.json`, and `package.json`. The main editor displays the `package.json` file with the following content:

```
1 {
2   "name": "backend",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "json-server": "^0.16.3"
14  }
15 }
```

# Vamos lá!

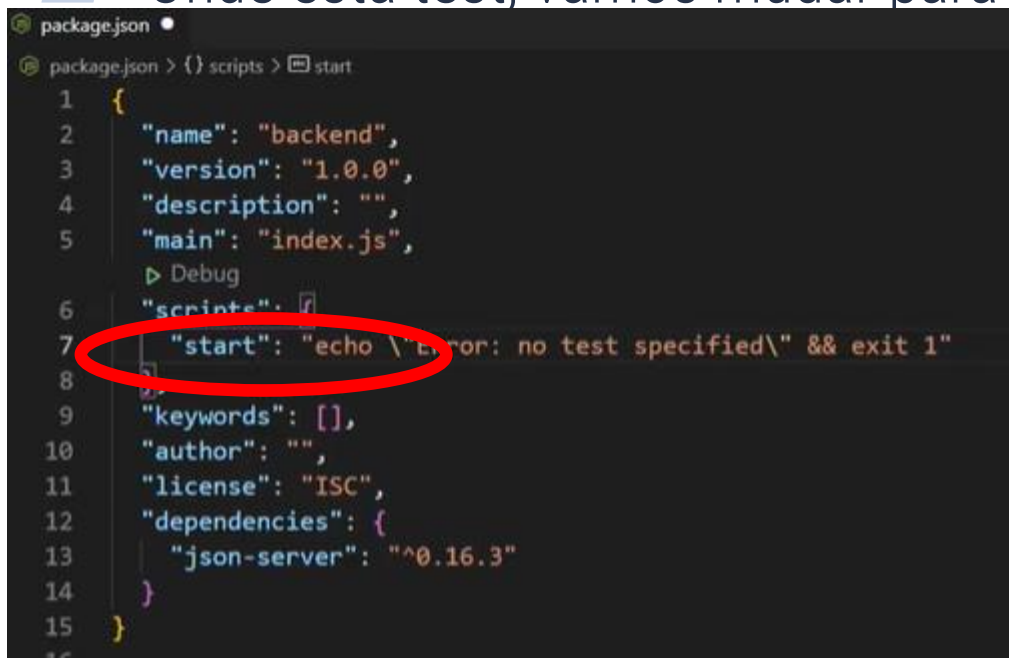
Ele tem o arquivo package-lock.json que é uma questão de integridade do node,



```
1 {
2   "name": "backend",
3   "version": "1.0.0",
4   "lockfileVersion": 1,
5   "requires": true,
6   "dependencies": {
7     "@sindresorhus/is": {
8       "version": "0.14.0",
9       "resolved": "https://registry.npmjs.org/@sindresorhus/is/-/is-0.14.0.tgz",
10      "integrity": "sha512-9NET910DNAIPngYnLLPeg+Ogzqsi9uM4mSboU5y6p8S5DzMTVEsJZrawi+BoDNUVBa2DhJqQYUFvME",
11     },
12     "@szmarczak/http-timer": {
13       "version": "1.1.2",
14       "resolved": "https://registry.npmjs.org/@szmarczak/http-timer/-/http-timer-1.1.2.tgz",
15       "integrity": "sha512-XIB2XbzHTN6ieIjfIMV9h1VcfFPU26s2vafYwQcZHwXHOxiaRZYEDKEwdl129Zyg50+foYV2jCgtrqS",
16       "requires": {
17         "defer-to-connect": "^1.0.1"
18       }
19     },
20     "accepts": {
21       "version": "1.3.7",
22       "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.7.tgz",
23       "integrity": "sha512-IL80Qs2WjY1JIBNzNkK6KYqlVMTbZLXgHx2oT0pU/fjRHxEp+PEfEPY0R3WCwAGVotauxh1hOxNgIf",
24       "requires": {
25         "mime-types": "~2.1.24",
26         "negotiator": "0.6.2"
27       }
28     },
29     "ansi-align": {
```

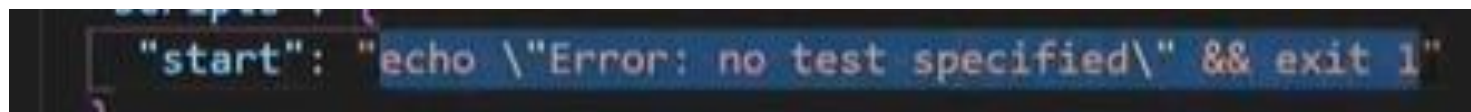
# Vamos lá!

- Dentro da pasta node\_modules ele baixou tudo o que ele precisasse para que o json server funcionasse, inclusive o json server está nessa lista gigante.
- Vamos recolher a pasta
- Vamos retornar no arquivo package.json e agora vamos fazer a configuração.
- Onde está test, vamos mudar para start



```
package.json
package.json > {} scripts > start
1 {
2   "name": "backend",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "echo \Error: no test specified\ && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "json-server": "^0.16.3"
14  }
15 }
```

O que são esses scripts? São scripts que consigo rodar através do comando npm. A partir de agora eu consigo digitar:  
Npm start e o node vai executar o que estiver dentro das aspas



```
"start": "echo \Error: no test specified\ && exit 1"
```

# Vamos lá!

Vamos retornar na documentação e ver como fazemos o servidor funcionar. Ir no github, copiar e colar:

<https://github.com/typicode/json-server>

Clique aqui:

## json-server

Node.js CI passing

### Important

Viewing alpha v1 documentation – usable but expect breaking changes. For stable version, see [here](#)



# Vamos lá!

Vamos retornar na documentação e ver como fazemos o servidor funcionar. Ir no github, copiar e colar:

<https://github.com/typicode/json-server>

Start JSON Server

```
json-server --watch db.json
```

```
json-server --watch db.json
```

```
package.json > {} scripts > start
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "json-server --watch db.json"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "json-server": "^0.16.3"
14   }
15 }
16
```

# Vamos lá!

Está feito... Por padrão, o JSON SERVER sobe na porta 3000:

Start JSON Server

```
json-server --watch db.json
```

Now if you go to <http://localhost:3000/posts/1>, you'll get

Caso vc esteja com essa porta ocupada ou queira mudar, vc pode correr na documentação, tem uma documentação bem legal e tem uma configuração que vc pode passar a porta que vc quiser. Vamos utilizar a porta 3000 mesmo.

## CLI usage

```
json-server [options] <source>
```

### Options:

<code>--config, -c</code>	Path to config file	[default: "json-server.json"]
<code>--port, -p</code>	Set port	[default: 3000]
<code>--host, -H</code>	Set host	[default: "localhost"]
<code>--watch, -w</code>	Watch file(s)	[boolean]
<code>--routes, -r</code>	Path to routes file	
<code>--middlewares, -m</code>	Paths to middleware files	[array]
<code>--static, -s</code>	Set static files directory	
<code>--read-only, --ro</code>	Allow only GET requests	[boolean]
<code>--no-cors, --nc</code>	Disable Cross-Origin Resource Sharing	[boolean]
<code>--no-gzip, --ng</code>	Disable GZIP Content-Encoding	[boolean]
<code>--snapshots, -S</code>	Set snapshots directory	[default: "."]
<code>--delay, -d</code>	Add delay to responses (ms)	
<code>--id, -i</code>	Set database id property (e.g. <code>_id</code> )	[default: "id"]
<code>--foreignKeySuffix, --fks</code>	Set foreign key suffix, (e.g. <code>_id</code> as in <code>post_id</code> )	[default: "Id"]
<code>--quiet, -q</code>	Suppress log messages from output	[boolean]
<code>--help, -h</code>	Show help	[boolean]
<code>--version, -v</code>	Show version number	[boolean]

# Vamos lá!

- Vamos voltar e verifiquem que qdo tem uma bolinha significa que o arquivo ainda não foi salvo, vamos salvar:

- Note que ele vai querer utilizar um arquivo db.json

```
"scripts": {  
  "start": "json-server --watch db.json"  
},  
"devDependencies": {}  
}
```



```
package.json •  
package.json > {} scripts > start  
1 {  
2   "name": "backend",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "start": "json-server --watch db.json"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC",  
12  "dependencies": {  
13    "json-server": "^0.16.3"  
14  }  
15 }  
16
```

# Vamos lá!

- Não vamos inicializar esse arquivo para vcs verem o que acontece quando ele cria o projeto e como é a estrutura desse db.json sem precisar fazer do zero.
- Vamos retornar no terminal e rodar o comando :
  - ▷ `npm start <ENTER>`

# Vamos lá!

- Ele iniciou o meu servidor backend@1.0
- Ai ele falou... Opa, não encontrei o arquivo db.json, então vou criar ele...
- Quando ele cria, por padrão já traz alguns recursos (<http://localhost:3000/posts>, etc., os famosos endpoints das apis rest. Já vamos ter um endpoint para posts, comments e profile

```
> backend@1.0.0 start C:\angular\cursoCPS\backend
> json-server --watch db.json

\{^_^}/ hi!

Loading db.json
Oops, db.json doesn't seem to exist
Creating db.json with some default data

Done

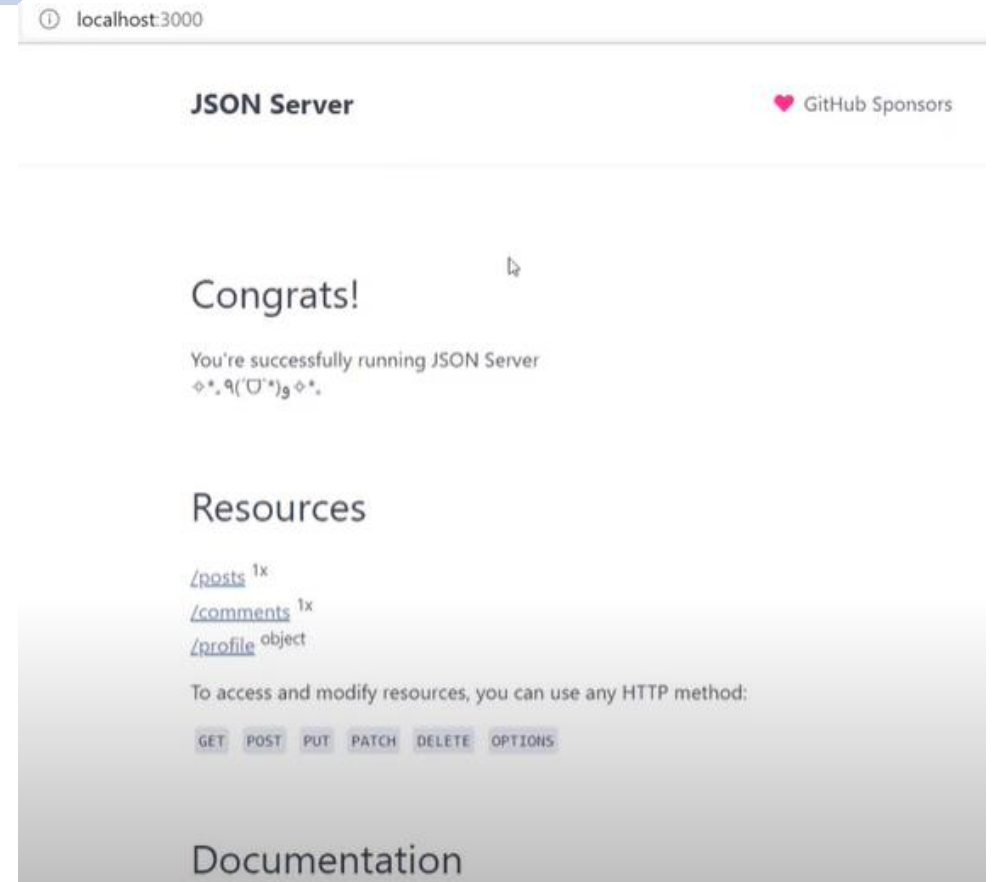
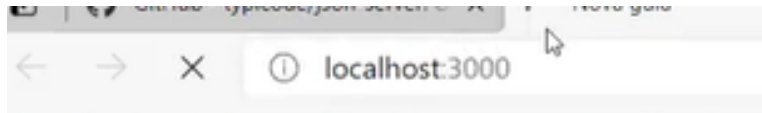
Resources
http://localhost:3000/posts
http://localhost:3000/comments
http://localhost:3000/profile

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
Watching...
```

# Vamos lá!

- Vamos ver nosso servidor funcionando...
- Abra um browser e digite localhost:3000
- Ele está me avisando que conseguiu abrir o servidor e
- Já mostra quais recursos eu possuo:
  - ▶ Posts, comments e profile



# Vamos lá!

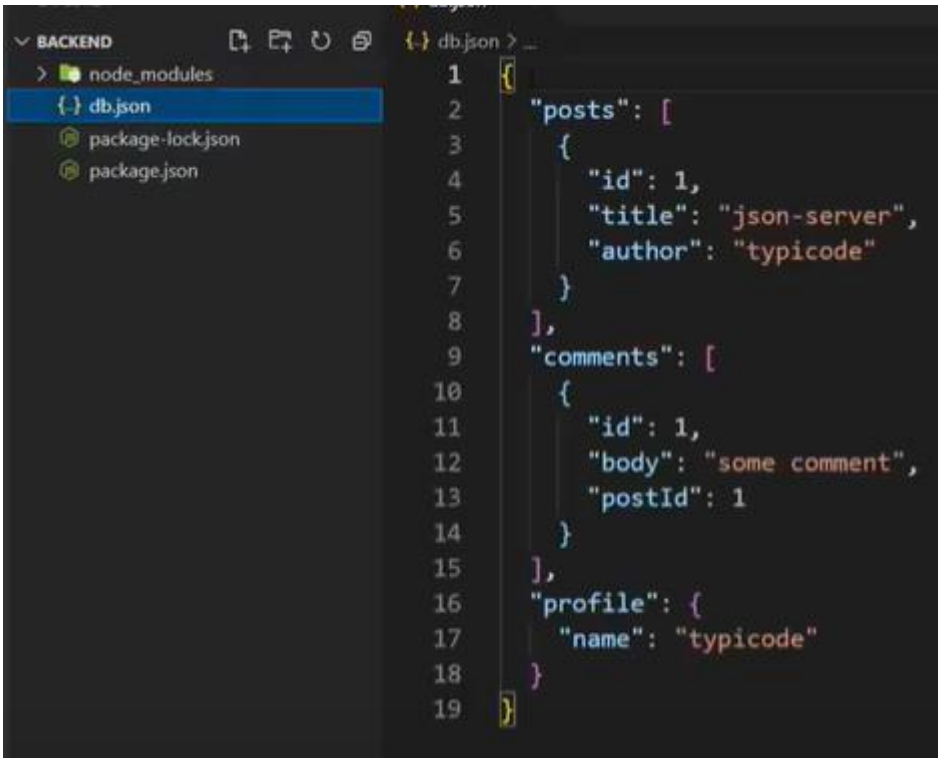
- Clicando dentro de posts ele vai para o endpoint de posts e vai dar um get mostrando todos os registros que tenho ali.

```
[  
  {  
    "id": 1,  
    "title": "json-server",  
    "author": "typicode"  
  }  
]
```

- Vamos conferir como é a estrutura desse arquivo db.json, voltando para o VSCode

# Vamos lá!

Vamos conferir como é a estrutura desse arquivo db.json, voltando para o VSCode. Clicando no db.json:

A screenshot of the Visual Studio Code editor interface. On the left, the 'EXPLORER' sidebar shows a project named 'BACKEND' with a file tree containing 'node\_modules', 'db.json', 'package-lock.json', and 'package.json'. The 'db.json' file is selected and highlighted in blue. The main editor area displays the content of 'db.json' with line numbers from 1 to 19. The JSON structure is as follows:

```
1 {  
2   "posts": [  
3     {  
4       "id": 1,  
5       "title": "json-server",  
6       "author": "typicode"  
7     }  
8   ],  
9   "comments": [  
10    {  
11      "id": 1,  
12      "body": "some comment",  
13      "postId": 1  
14    }  
15  ],  
16  "profile": {  
17    "name": "typicode"  
18  }  
19 }
```

Ele cria um objeto, já que é um arquivo .json.  
Vamos ter o posts que é um array de itens. Ele já atribuiu um ID,  
Um título e um autor

A close-up screenshot of the 'posts' array in the 'db.json' file. It shows a single object within the array:

```
{  
  "posts": [  
    {  
      "id": 1,  
      "title": "json-server",  
      "author": "typicode"  
    }  
  ],  
}
```



# Vamos lá!

## Formatação do arquivo

Além da terminação .json em todos os arquivos que utilizam esse formato, os dados armazenados devem seguir uma notação específica, ou seja, precisam ser organizados com os seguintes elementos básicos:

- **chaves** { } para delimitar os objetos e obrigatórias para iniciar e encerrar o conteúdo;
- **colchetes** [ ] para indicar um array;
- **dois pontos** : para separar a chave e seu valor correspondente;
- **vírgula** , para indicar a separação entre os elementos.

Veja, a seguir, alguns exemplos de como os dados devem ser relacionados em um arquivo .json.

```
{  
  "posts": [  
    {  
      "id": 1,  
      "title": "json-server",  
      "author": "typicode"  
    }  
  ],  
}
```

# Vamos lá!

- Depois tem os comentários. Ele relaciona o meu comentário a um post.
- E depois tenho o profile.
- Esse arquivo, nós iremos apagar no futuro e colocar os dados do banco de dados do curso.
- Por hoje temos nosso servidor funcionando e no próximo arquivo iremos ver como testar o servidor usando o Insomnia.

