

Node e Package.json

Labenu_



O que vamos ver hoje? 🙄

- Noções de backend
- Node e NPM
- Primeiros passos para construir um backend usando o Node



Boas vindas ao Backend!

Vamos começar nossa ambientação recapitulando alguns pontos da jornada que nos trouxeram até aqui



Introdução

- No início do curso, nosso fluxo de trabalho consistia em criar e editar arquivos HTML, CSS e JS.
- Para executá-los, bastava abrir o **index.html** no navegador
- O que não havíamos discutido ainda era que o **navegador compila nosso código-fonte em tempo de execução e roda o binário resultante**. Chamamos esse processo de interpretação.



Introdução

- Nos módulos seguintes, aprendemos a usar o *React*, uma biblioteca de código JS que nos permite criar páginas mais performáticas e escaláveis
- Essa biblioteca é distribuída pelo **N**ode **P**ackage **M**anager. Assim, para termos acesso a ela, tivemos que instalar o Node (e com ele vem o NPM)



Introdução

- A partir de então, passamos a criar e rodar nossos projetos a partir do terminal
- O que não ainda discutimos mais a fundo é: **o que é o Node? O que acontece após o *npm run start*?**



Node.js

Labenu_



Node.js

- Alguns fatos sobre o Node:
 - Usa a engine V8 criada pelo Google, a mesma que o Chrome usa para executar JS, ou seja, **é um interpretador de JavaScript.**
 - Rápido crescimento e adoção pela comunidade dada sua performance e facilidade de uso.
 - Tornou possível o uso do JS do lado do servidor (backend)



NPM

Labenu_





NPM



- **N**ode **P**ackage **M**anager, é o gerenciador de pacotes do Node
- De acordo com a documentação oficial, pacote é qualquer arquivo ou pasta contendo um programa descrito por um **package.json** (ex: React)

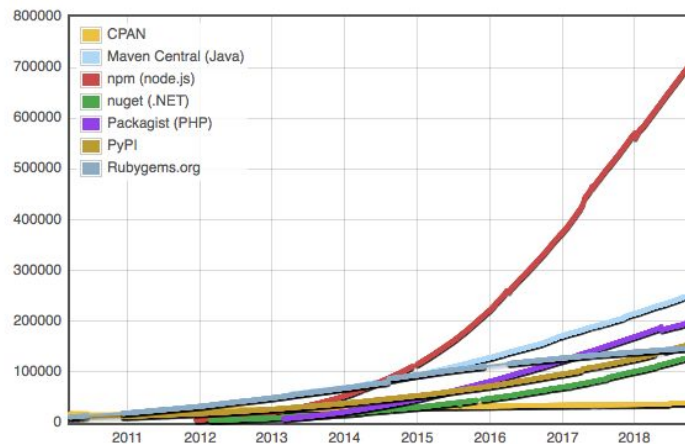


NPM



- Um dos principais motivos pela escolha do uso de JS pelas empresas e devs é a grande quantidade de pacotes disponíveis
- npm tem mais de 10 milhões de usuários, que instalam mais de 30 **bilhões** de **pacotes por mês**
- Em uma terça-feira comum (dia mais movimentado no NPM) usuários baixam mais do que 1.3 bilhões de pacotes

Module Counts



package.json

Labenu_



package.json

- Arquivo que mora **na raiz** de projetos JS e que utiliza NPM como administrador de dependências.
- Neste arquivo estão contidas todas as informações sobre o que o projeto se trata, quem é o autor, quais são suas dependências e scripts customizados.



package.json 🎬

- É a referência para os comandos do NPM:
 - **npm install**: cria a pasta `node_modules` com as dependências listadas atualmente no `package.json`
 - **npm install nomeDoPacote**: inclui o pacote especificado em `"dependencies"` e no `node_modules`
 - **npm install nomeDoPacote --save-dev**: inclui o pacote especificado em `"devDependencies"` e no `node_modules`
 - **npm run nomeDoComando**: procura o comando especificado em `"scripts"` e o executa



package.json

- Dependencies
 - Define todos os pacotes e suas respectivas versões que seu projeto depende para poder **ser executado**.
 - São as dependências do projeto.
- devDependencies
 - Define todos os pacotes e suas respectivas versões que seu projeto depende para poder **ser desenvolvido**
 - São as dependências das dependências.



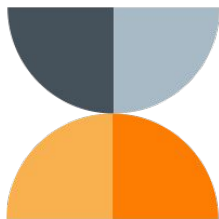
package.json 🎬

- É a referência para os comandos do NPM:
 - **npm init**: cria um package.json (deve ser preenchido)
 - **npm init --yes**: cria um package.json (com valores padrão)

Vamos ver na prática! 🧪



Pausa para relaxar 🤔



- **NPM** é uma ferramenta que facilita o acesso a bibliotecas de código JS
- Criamos o package.json com o comando ***npm init***
- O comando ***npm install*** instala dependências, que podem ser *dependencies* ou *devDependencies*



Backend com Node.js

Labenu_



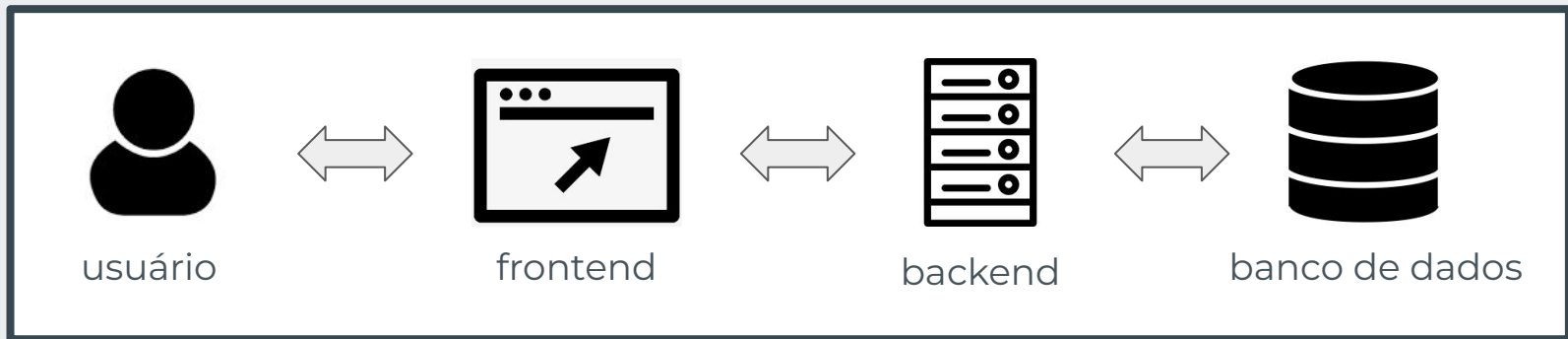
Backend com Node.js

- O Node é pouco perceptível no front, pois os projetos em React continuam sendo executados no navegador
- No entanto, aqui no backend nós **não estamos mais no navegador**.
- Isso significa que não temos mais acesso aos seus objetos globais, como *window* (e seus métodos: *alert*, *confirm*, *prompt...*), *document*, *event* e *localStorage*



Backend com Node.js

- A partir de agora, estaremos do lado do backend.
De maneira simplificada, ele é um programa composto por funções invocadas através do protocolo HTTP, que gerenciam o acesso a uma base de dados.



Backend com Node.js

- Durante essa semana, ainda não teremos um banco de dados. A informação persistirá na memória dos processos, apenas enquanto a aplicação estiver rodando
- Por hoje, iremos testar um pouco o node simulando funções com entradas e saídas utilizando algumas ferramentas do node



Backend com Node.js

- Para simular a chegada de uma requisição, passaremos valores de **entrada** pelo terminal, que serão acessados no código pela propriedade **process.argv**
- O terminal também é o console do Node, ou seja, ainda teremos disponível o método **console.log**, que será um grande recurso visual no backend
- Para rodar um arquivo javascript pelo terminal, utilizaremos o comando **node nomeDoArquivo.js**

Vamos ver na prática! 



O `process.argv`

- A propriedade **`process.argv`** é nativa do Node, e consiste em um array de strings que são os argumentos para que o processo seja executado. Os dois primeiros argumentos são fixos:
 - **`process.argv[0]`**: o primeiro argumento é o próprio node
 - **`process.argv[1]`**: o segundo argumento é o arquivo que vamos executar
- A partir do **`process.argv[2]`**, nós podemos atribuir valores

Vamos ver na prática! 



Criando scripts personalizados

Labenu_



Scripts personalizados

- Durante o front-end, nós utilizamos o comando **npm run start** para executar os nossos projetos;
- Isso acontece pois o react cria **scripts** no package.json para que certas sequências de comandos sejam executadas;
- Podemos fazer o mesmo com nossos projetos, alterando a área de **scripts** do arquivo package.json



Scripts personalizados

- Por padrão, esta área vem preenchida desta forma:

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



Scripts personalizados

- E podemos adicionar múltiplos scripts, com diferentes comandos em cada um

```
{
  "name": "exemplo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "clear && node ./index.js",
    "serve": "clear && echo \"Hello\" && node ./index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

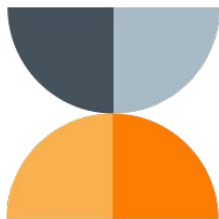
Vamos ver na prática! 



Pausa para relaxar 🧘

Para os nossos testes de hoje, usaremos:

- **Entradas:** `process.argv[2]` (3, 4, 5...)
- **Saídas:** `console.log` / `console.table`
- **Rodar o projeto:** `node nomeDoArg.js`



Prática!

Labenu_



Coding Together

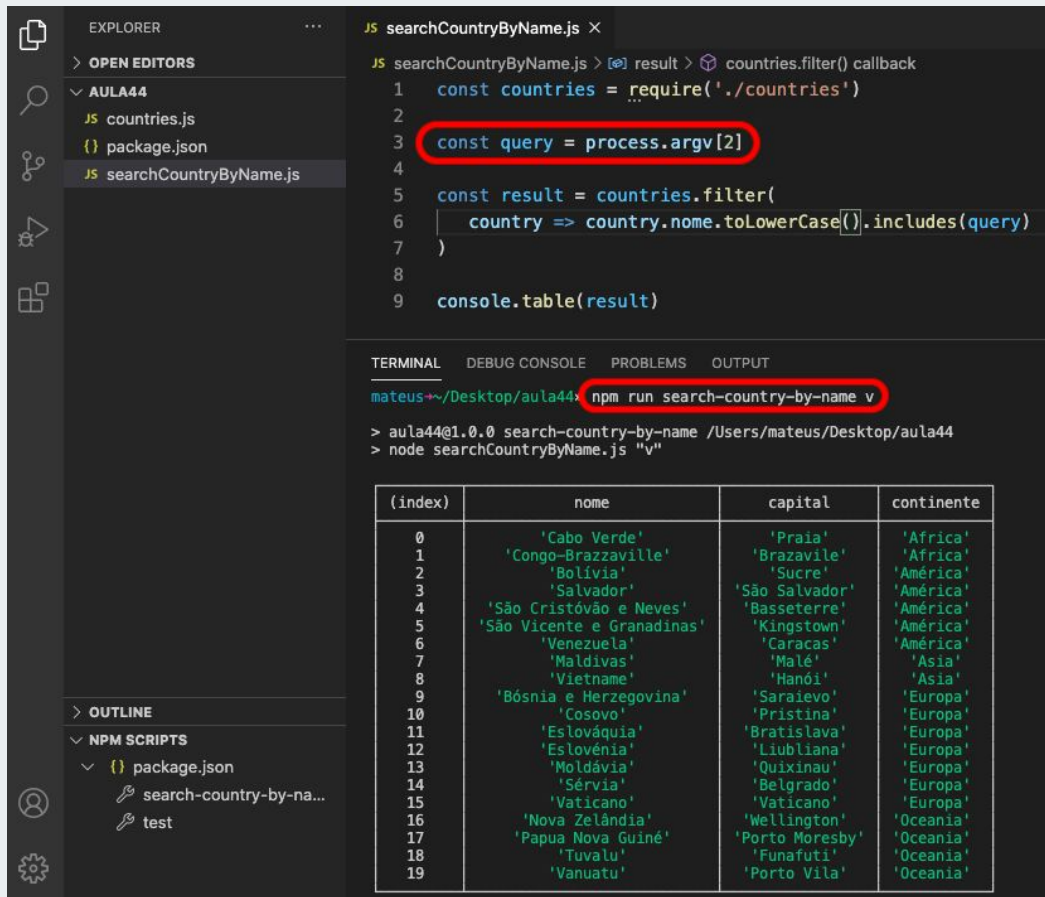
Exercício

Crie um pacote do Node contendo um script chamado ***search-country-by-name***. Ele deve receber uma string (*letra ou palavra*) pelo terminal e imprimir uma tabela com os países cujo nome contém a *string* informada

Dica: use o `console.table()`



Coding Together



The image shows a VS Code editor window with a file named `searchCountryByName.js`. The code in the editor is as follows:

```
JS searchCountryByName.js
JS searchCountryByName.js > [0] result > countries.filter() callback
1  const countries = require('./countries')
2
3  const query = process.argv[2]
4
5  const result = countries.filter(
6    country => country.nome.toLowerCase().includes(query)
7  )
8
9  console.table(result)
```

The terminal output shows the command `npm run search-country-by-name v` being executed, resulting in a table of country data:

```
mateus@~/Desktop/aula44: npm run search-country-by-name v
> aula44@1.0.0 search-country-by-name /Users/mateus/Desktop/aula44
> node searchCountryByName.js "v"
```

(index)	nome	capital	continente
0	'Cabo Verde'	'Praia'	'Africa'
1	'Congo-Brazzaville'	'Brazaville'	'Africa'
2	'Bolívia'	'Sucre'	'América'
3	'Salvador'	'São Salvador'	'América'
4	'São Cristóvão e Neves'	'Basseterre'	'América'
5	'São Vicente e Granadinas'	'Kingstown'	'América'
6	'Venezuela'	'Caracas'	'América'
7	'Maldivas'	'Malé'	'Asia'
8	'Vietname'	'Hanói'	'Asia'
9	'Bósnia e Herzegovina'	'Saraievo'	'Europa'
10	'Cosovo'	'Pristina'	'Europa'
11	'Eslováquia'	'Bratislava'	'Europa'
12	'Eslovénia'	'Liubliana'	'Europa'
13	'Moldávia'	'Quixinau'	'Europa'
14	'Sérvia'	'Belgrado'	'Europa'
15	'Vaticano'	'Vaticano'	'Europa'
16	'Nova Zelândia'	'Wellington'	'Oceania'
17	'Papua Nova Guiné'	'Porto Moresby'	'Oceania'
18	'Tuvalu'	'Funafuti'	'Oceania'
19	'Vanuatu'	'Porto Vila'	'Oceania'



Resumo

Labenu_



Resumo

- Node.js é um **interpretador** de Javascript
- Seu foco foi simplificar a manipulação de arquivos
- O Node.js não possui tudo que o navegador injeta para gente quando usamos JS em sites, porém possui recursos a mais



Resumo

- **npm** nos ajuda muito a cuidar do gerenciamento das **bibliotecas externas**
- Para iniciar novos projetos de Node.js, usamos o comando **npm init**
- Acessamos argumentos passados aos nossos scripts através de **process.argv**





Dúvidas?





Obrigado(a)!