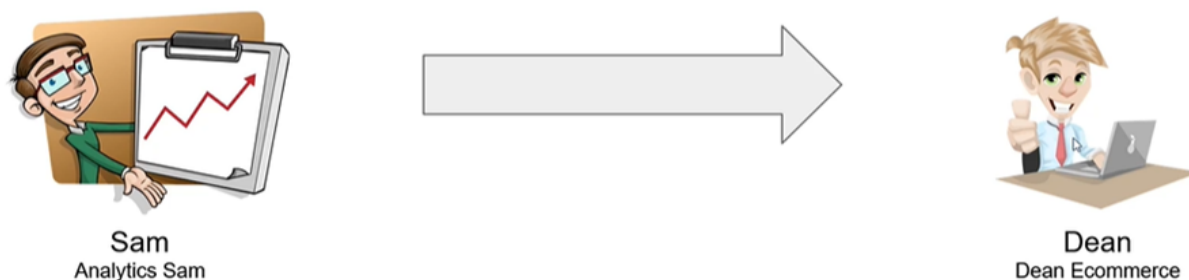


API MEDICATOR

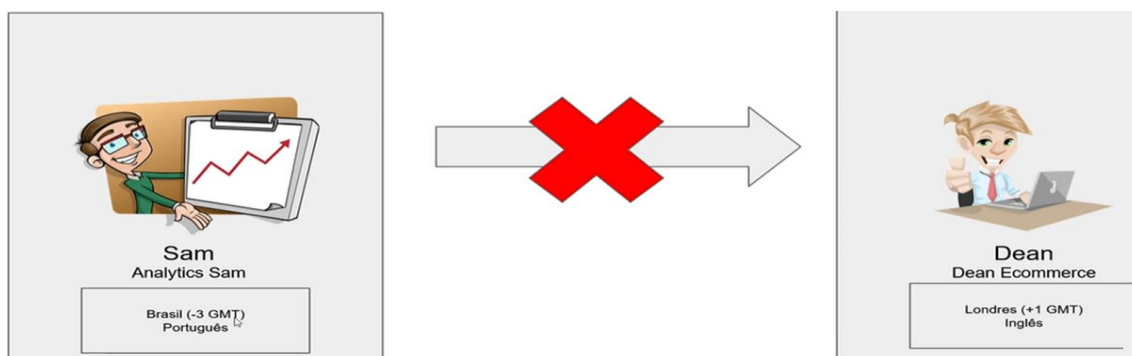
Este manual traz as orientações de como fazer a API Medicator, desenvolvida pelo time L3 da NFE.io.

Em primeiro lugar, vamos esclarecer o que é uma API.

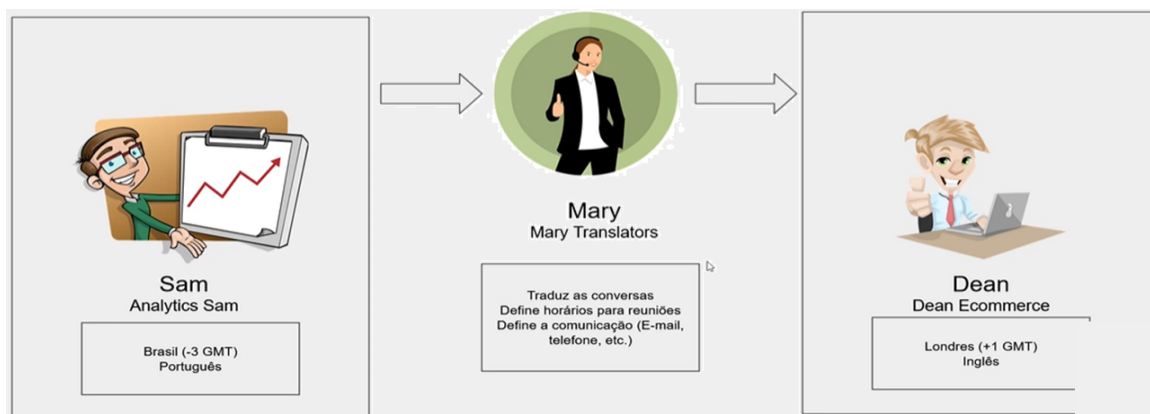
Imaginem a seguinte situação:



O Sam quer se comunicar com o Dean



Mas o Sam não fala em Inglês

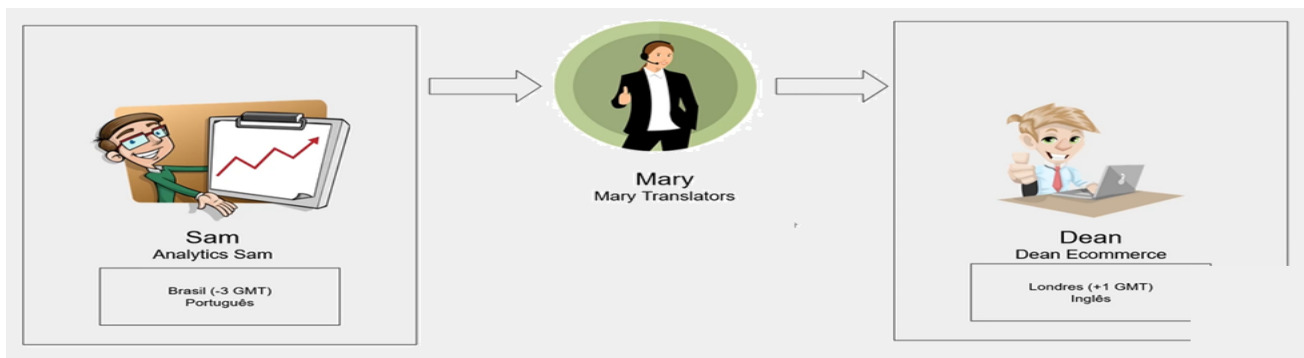


Então eles precisam de uma interprete!

Uma API REST é uma aplicação que permite a integração entre diferentes sistemas padronizando o fluxo de requisições, a forma de envio e recebimento de informações e o protocolo de codificação das mensagens.

O REST é um conjunto de convenções arquiteturais que utiliza algumas camadas do HTTP para o envio e recebimento de informações.

Logo, uma API REST é uma API que permite a integração entre sistemas utilizando as convenções arquiteturais do REST.



REST API (meio de campo entre sistemas)

O que faz:

- Define linguagem de comunicação (JSON).
- Define a ordem de requisições.
- Define como os dados são enviados e recebidos.

No nosso dia a dia, já utilizamos APIS como:

YouTube

<https://developers.google.com/youtube/v3>

⇒ Gerenciamento de dados de canais no YouTube

Instagram

<https://developers.facebook.com/docs/instagram-api/>

⇒ Gerenciamento de presença na plataforma

Mais de 1.400 APIs públicas

<https://any-api.com>

Desenvolvimento

Para começarmos, precisamos definir a finalidade que uma API terá.

A Medicator tem como finalidade facilitar o acesso aos dados públicos sobre preços de medicamentos no Brasil, atendendo várias necessidades de uso dos cidadãos.

Exemplo de uso:

- Quem: pessoas com doenças crônicas ou que precisem eventualmente de uma medicação
- Quando: buscam em períodos determinados os mesmos medicamentos, genéricos, marcas, quantidades.
- Por que: facilidade na busca, dar facilidade para encontrar todos os dados sobre um medicamento em um mesmo lugar.

Com os dados acima, fazemos a descrição final do que será feito. No caso da Medicator, irá usar uma planilha no formato CSV da ANVISA, a qual contém preços de medicamentos no contexto brasileiro. Será utilizado como referência o Manual de Dados do Consumidor que contém as siglas utilizadas na planilha CSV.

Esses dados são fornecidos no site Portal Brasileiro de Dados Abertos, disponível em <https://dados.gov.br/dataset/preco-de-medicamentos-no-brasil-governo>.

The screenshot shows the website dados.gov.br, the Portal Brasileiro de Dados Abertos. The header includes navigation links like BRASIL, CORONAVÍRUS (COVID-19), Simplifique!, Participe, Acesso à informação, Legislação, and Canais. The main content area displays the dataset 'Preço de Medicamentos no Brasil - Governo' by the Agência Nacional de Vigilância Sanitária - ANVISA. It shows 0 followers and a description of the data: 'Os dados representam a lista de preços de Medicamentos, contemplando o preço Fábrica, ou preço fabricante (PF), que é o preço máximo praticado que pode ser praticado pelas empresas produtoras ou importadoras do produto e pelas empresas distribuidoras. O PF indica o preço máximo permitido para venda a farmácias e drogarias e o Preço Máximo de Venda ao Governo (PMVG) indica o preço teto de venda aos entes da administração pública quando for aplicável o desconto do Coeficiente de Adequação de Preços (CAP), quando não for o preço teto é o PF.' Below the description, there are filters for ANVISA, MEDICAMENTO, and PREÇO. A feedback section asks 'Estes dados estão disponíveis como o esperado?' with a 'Sim' button (12 votes) and a 'Não' button (2 votes).

(imagem do site que disponibiliza esses dados)

Ferramentas

Após definir o que a API vai fazer, vamos ao seu desenvolvimento. Para isso devemos definir quais ferramentas serão utilizadas.

No caso da Medicator foram utilizadas:

- Visual Studio Code ou VsCode
- JavaScript
- JQuery
- Node Js
- Koa Js

Endpoints

Precisamos definir também quais serão os endpoints. Os endpoints são os principais dados necessários para um sistema se comunicar com uma Api (são os dados que as pessoas querem consultar). São basicamente a URL onde um serviço pode ser acessado. Cada endpoint tem o método HTTP conveniente. Pode ser um GET onde você está "pegando" coisas ou um POST, onde você está "inserindo".

A Api Medicator tem os seguintes Endpoints: Nome do medicamento, princípio ativo, restrição, tarja, código de barras e o código GGREM.

Outros métodos:

- GET
Envio de parâmetros via URL
Recebimento de informações no corpo da resposta
- POST
Envio de parâmetros via corpo
Recebimento de informações no corpo da resposta
- PUT
Envio de parâmetros via URL e via corpo
Recebimento de informações no corpo da resposta
- DELETE
Envio de parâmetros via URL
Recebimento de informações no corpo da resposta
- PATCH
Envio de parâmetros via URL e corpo
Recebimento de informações no corpo da resposta
- OPTIONS
Envio de parâmetros via URL
Recebimento de informações no cabeçalho da resposta
- HEAD
Envio de parâmetros via URL
Recebimento de informações no cabeçalho da resposta

A Mediator só usará métodos Gets.

Statuscode

Temos também o Statuscode (Código de status de resposta HTTP) que são usados para dar retorno ao usuário.

Os códigos de status são emitidos por um servidor em resposta a uma solicitação do cliente feita ao servidor.


Definição de status code

- Padrão de 3 dígitos que indica o resultado da tentativa de tratar a requisição
- São divididos em 4 classes, identificadas pelo primeiro dígito




Retornos mais comuns

Sucesso

Successful - 2xx 	
200 OK	Status genérico de sucesso. Normalmente usado como resposta a GETs ou atualizações com PUT/PATCH.
201 Created	Indica que um recurso foi criado. Normalmente usado para responder a requisições com PUTs e POSTs.
202 Accepted	Indica que a requisição foi aceita para processamento. Normalmente utilizada em chamadas assíncronas.
204 No Content	A requisição obteve sucesso, mas não há nada para mostrar. Normalmente usada como resposta a DELETEs.
206 Partial Content	O retorno está incompleto. Normalmente usado em recursos com paginação.




Erro

Client Error - 4xx 	
400 Bad Request	Status genérico de erro para requisições que não puderam ser processadas.
401 Unauthorized	O servidor não reconheceu você por falta de credenciais válidas para o recurso solicitado.
403 Forbidden	Sua credencial não tem privilégios suficientes para acessar o recurso que está solicitando.
422 Unprocessable Entity	Ocorreu algum erro de negócio com sua mensagem. Sintaticamente correto, semanticamente não.



Servidor

Server Error - 5xx 	
500 Internal Server Error	A requisição está certa, porém algum erro aconteceu no servidor. Não adianta mandar de novo agora!
503 Service Unavailable	O servidor não consegue processar agora por sobrecarga ou manutenção.



Framework

Mas afinal o que é isso e para que serve?

O framework é um pacote de códigos prontos que podem ser utilizados no desenvolvimento de sistemas. A proposta de uso desta ferramenta é aplicar funcionalidades, comandos e estruturas já prontas para garantir qualidade no projeto e produtividade.

Na API Medicator foi utilizado o framework Koa (disponível em <https://koajs.com/>).

O Koa é uma nova estrutura da web, que visa ser uma base menor, mais expressiva e mais robusta para aplicativos da web e APIs. Ele permite que você evite retornos de chamada e aumente muito o tratamento de erros, fornecendo um elegante conjunto de métodos que tornam a escrita de servidores rápida e agradável.

Se não usássemos um framework, teríamos que fazer tudo do zero, tendo muito mais trabalho e probabilidades de erros.

Node

O que é Node.js?

Node.js não é uma linguagem de programação. Você programa utilizando a linguagem JavaScript, a mesma usada há décadas no client-side das aplicações web. Javascript é uma linguagem de scripting interpretada.

Node.js não é um framework Javascript. Ele está mais para uma plataforma de aplicação, na qual você escreve seus programas com Javascript que serão compilados, otimizados e interpretados.

O Node.js se caracteriza como um ambiente de execução JavaScript. Com ele, o usuário pode criar aplicações sem depender do browser para isso. Com alta capacidade de escalabilidade, boa flexibilidade, arquitetura e baixo custo, tornam-se uma ótima opção para programação.

Npm(npm)

Node Package Manager ou npm (originalmente abreviação de Node Package Manager) é um gerenciador de pacotes para a linguagem de programação JavaScript mantido pela npm, Inc. npm é o gerenciador de pacotes padrão para o ambiente de tempo de execução JavaScript Node.

Package. Json

O arquivo json é o coração de qualquer projeto do Node. Ele registra metadados (metadados são informações que crescem aos dados e que têm como objetivo informar-nos sobre eles para tornar mais fácil a sua organização.) importantes sobre um projeto que são necessários antes de publicar no NPM e também define os atributos funcionais de um projeto que o npm usa para instalar dependências, executar scripts e identificar o ponto de entrada para nosso pacote.

Mão na massa!!

Vamos começar instalando as ferramentas:

- ⇒ Instale o Visual Studio Code disponível em <https://code.visualstudio.com>
 - Escolha a versão adequada para a o seu sistema operacional.
 - Siga as instruções de instalação.
- ⇒ Instale o Node Js disponível em <https://nodejs.org>
 - Escolha a opção LTS.
 - Escolha a versão adequada para a o seu sistema operacional.
 - Siga as instruções de instalação.
- ✓ Vídeo explicativo de como instalar o ambiente:
<https://www.youtube.com/watch?v=55koWQdsljE>

Para esta Api vamos usar o arquivo CSV (que contem os dados que serão consultados, fornecido pelo site). Foi usada a ferramenta online CSV JASON <https://csvjson.com/csv2json>) para a conversão do arquivo de CSV para JSON. Não será feita a atualização automática desses dados, sendo assim este arquivo deverá ser atualizado manualmente de tempo em tempo na API.

Depois que o ambiente estiver todo instalado e pronto para usar:

- ✓ Localizar e baixar o arquivo CSV (http://landpage-h.cgu.gov.br/dadosabertos/index.php?url=https://dados.anvisa.gov.br/dados/DADOS_ABERTOS_MEDICAMENTOS.csv);
- ✓ Usar a ferramenta CSV JASON para transformar o arquivo CSV de medicamentos em JSON (é uma notação, uma forma de se escrever) objetos em Javascript. Pode ser visto como um formato "universal" que é muito conveniente para troca de informações entre aplicações através de protocolos diversos;
- ✓ Crie uma pasta com um nome de sua preferência, no local que preferir, no caso da Medicator, o nome da pasta se chama Api_Consulta e está salva em meus documentos. Esta pasta terá todos os arquivos relacionados ao desenvolvimento da Api;
- ✓ Abra a pasta que foi criada;
- ✓ Crie outra pasta dentro da pasta da Api com o nome Preco_csv e coloque o arquivo json que foi criado no CSV JASON online (aquele csv que foi baixado e depois convertido).

- ✓ Abra o Visual Studio Code ou VsCode (normalmente ele está em inglês), caso queira mudar para o idioma português, pressione as teclas Ctrl + Shift + P e procure pela opção Configure Display Language para mudar;
- ✓ Vá em File(Arquivo), Open Folder...(Abrir pasta...) e selecione a pasta da Api;
- ✓ Precisamos criar o package. Json
 - Para que serve o package json? O arquivo package.json é o ponto de partida de qualquer projeto NodeJS. Ele é responsável por descrever o seu projeto, informar as engines (versão do node e do npm), url do repositório, versão do projeto, dependências de produção e de desenvolvimento dentre outras coisas.
 - Abrir o terminal (Terminal → New Terminal), estar atento para que se esteja trabalhando dentro da pasta da Api, e digitar npm init e dar enter:

```
PS C:\Users\rpaga\Documents\Api_consulta> npm init
```

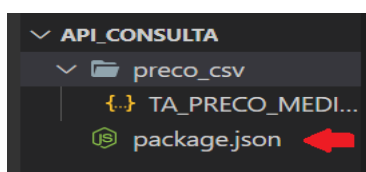
- No momento em que o Enter é dado, aparecerá na tela algumas perguntas. Essa perguntas são informações sobre o projeto, devemos preenche-las ou somente dar Enter para aceitar a sugestão que é dada ou deixar em branco:

```
Press ^C at any time to quit.
package name: (api_consulta)
version: (1.0.0)
description: Api para consulta de medicamentos
entry point: (index.js) no
test command:
git repository:
keywords:
author: Rafaela Paganotto
license: (ISC)
About to write to C:\Users\rpaga\Documents\Api_consulta\package.json:

{
  "name": "api_consulta",
  "version": "1.0.0",
  "description": "Api para consulta de medicamentos",
  "main": "no",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rafaela Paganotto",
  "license": "ISC"
}

Is this OK? (yes)
```

- No momento que confirmo (yes) será criado o arquivo package.json:



- Para usarmos o Koa e seus recursos precisamos instalá-lo no nosso projeto:
 - Abra um terminal (Terminal => Novo Terminal) e digite:
 - `npm install koa` (enter)

```
PROBLEMAS  CONSOLE DE DEPURACÃO  SAÍDA  TERMINAL
PS C:\Users\rpaga\Documents\Api_consulta> npm install koa
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN api_consulta@1.0.0 No repository field.

+ koa@2.13.1
added 44 packages from 23 contributors and audited 44 packages in 3.594s

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

found 0 vulnerabilities

PS C:\Users\rpaga\Documents\Api_consulta>
```

- `npm install --save koa-router` (enter)

```
PS C:\Users\rpaga\Documents\Api_consulta> npm install --save koa-router
npm WARN api_consulta@1.0.0 No repository field.

+ koa-router@10.1.1
updated 1 package and audited 52 packages in 1.345s

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

found 0 vulnerabilities

PS C:\Users\rpaga\Documents\Api_consulta>
```

- `npm install koa-json` (enter)

```
PS C:\Users\rpaga\Documents\Api_consulta> npm install koa-json
npm WARN api_consulta@1.0.0 No repository field.

+ koa-json@2.0.2
updated 1 package and audited 62 packages in 1.349s

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

found 0 vulnerabilities

PS C:\Users\rpaga\Documents\Api_consulta>
```

- `npm install koa/cors` (enter)

```
PS C:\Users\rpaga\Documents\Api_consulta> npm install @koa/cors
npm WARN api_consulta@1.0.0 No repository field.

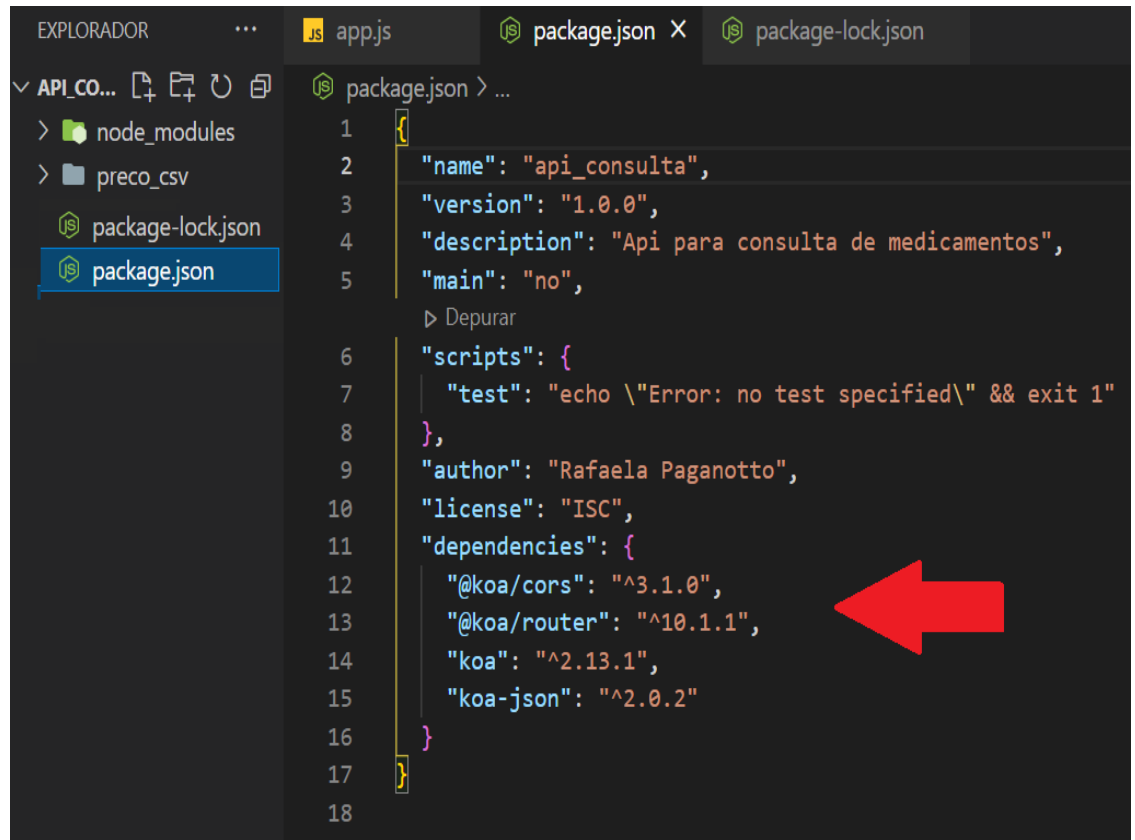
+ @koa/cors@3.1.0
added 1 package from 1 contributor and audited 63 packages in 1.903s

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

found 0 vulnerabilities

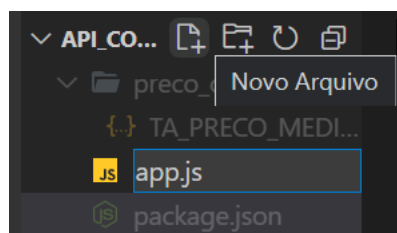
PS C:\Users\rpaga\Documents\Api_consulta>
```

- Observe que tudo que instalamos, vai sendo adicionado as dependências do package.json do projeto:

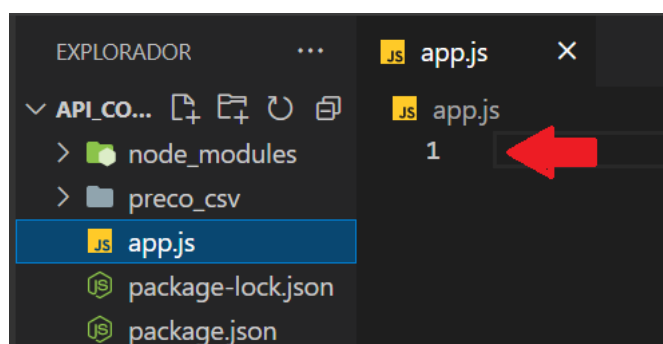


```
1 {
2   "name": "api_consulta",
3   "version": "1.0.0",
4   "description": "Api para consulta de medicamentos",
5   "main": "no",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "Rafaela Paganotto",
10  "license": "ISC",
11  "dependencies": {
12    "@koa/cors": "^3.1.0",
13    "@koa/router": "^10.1.1",
14    "koa": "^2.13.1",
15    "koa-json": "^2.0.2"
16  }
17 }
```

- Note que outro arquivo (package-lock.json) foi criado automaticamente.
- ✓ Agora que está tudo instalado, crie um arquivo, agora pelo VsCode, e coloque o nome de app.js;



-
- Dentro do arquivo app.js vamos colocar o código:



-

- Vamos por partes (**observe a numeração das linhas a direita para se orientar, elas terão continuidade**)

```
1 // Usamos // para comentar o nosso código
2
3 // Precisamos dizer que queremos usar o arquivo json no nosso projeto
4 const data = require("./preco_csv/TA_PRECO_MEDICAMENTO_GOV.json");
5 //-----
6 // Vamos usar o Koa em nosso projeto, para isso precisamos fazer as requisições (require)
7 const koa = require("Koa");
8 const koaRouter = require("koa-router");
9 const json = require("koa-json"); // padrões do koa
10
```

○

- Começamos com as requisições(require)

```
11
12 //-----
13 // para usar os recursos precisamos instanciar as classes
14
15 const app = new koa();
16 const router = new koaRouter({}); // para consultar no jason
```

○

▪ Configurações

```
17
18 //-----
19 // Compartilhamento de recursos com origens diferentes
20 app.use(cors());
21 // // Converte a resposta da requisição em json
22 app.use(json());
23
24 app.listen(3000, () => { // 3000 é a porta que iremos usar
25 | console.log("Rodando!"); // Esta mensagem aparecerá se tudo der certo e o servidor rodar
26 | });
27 // Rotas
28 app.use(router.routes());
29 // ctx agrupa tanto a request quanto a response
30 app.use(function (ctx) {
31 | ctx.body = { error: "Url inválida" };
32 | });
33 //-----
34
```

○

▪ Rotas (como respondem às solicitações do cliente)

```
35 //Router
36
37 router.get("/get_by_name", async (ctx) => await getByName(ctx));
38 router.get("/get_by_active", async (ctx) => await getByActive(ctx));
39 router.get("/get_by_cod", async (ctx) => await getByCod(ctx));
40 router.get("/get_by_ggrem", async (ctx) => await getByGgrem(ctx));
41
42 //-----
```

○

- Lógica da API(Como serão buscados e tratados os Endpoints)

```
43 //Funções para busca e tratamento dos endpoints
44
45 // retira espaços e acentos dos parametros
46 function normalizer(string) {
47   return string.normalize("NFD").replace(/[\u0300-\u036f]/g, "").trim();
48 }
```

```
49
50 // busca medicamento pelo nome
51 function getByNome(ctx) {
52   ctx.query.name = normalizer(ctx.query.name);
53
54   if (!/^[^a-zA-Z0-9\s]+/g.test(ctx.query.name)) {
55     //test retorna true ou false
56     ctx.query.name = ctx.query.name.toUpperCase();
57   } else {
58     ctx.body = { message: "Not Found" };
59     ctx.status = 400;
60     return;
61   }
62
63   let filtered = data.filter((obj) =>
64     obj["PRODUTO"].toUpperCase().includes(ctx.query.name)
65   );
66
67   if (filtered.length > 0) {
68     ctx.body = filtered;
69   } else {
70     ctx.body = { message: "Produto não encontrado" };
71     ctx.status = 404;
72   }
73 }
```

```
74
75 // busca medicamento pelo principio ativo
76 function getByActive(ctx) {
77   ctx.query.active = normalizer(ctx.query.active);
78
79   if (!/^[^a-zA-Z0-9\s]+/g.test(ctx.query.active)) {
80     ctx.query.active = ctx.query.active.toUpperCase();
81   } else {
82     ctx.body = { message: "Not Found" };
83     ctx.status = 400;
84     return;
85   }
86
87   let filtered = data.filter((obj) =>
88     obj["PRINCÍPIO ATIVO"].includes(ctx.query.active)
89   );
90
91   if (filtered.length > 0) {
92     ctx.body = filtered;
93   } else {
94     ctx.body = { message: "Princípio ativo inexistente" };
95     ctx.status = 404;
96   }
97 }
```

```

98
99 // busca medicamento pelo código de barras
100 function getByCod(ctx) {
101   if (/[0-9]+/g.test(ctx.query.codigo)) {
102   } else {
103     ctx.body = { message: "Not Found" };
104     ctx.status = 400;
105     return;
106   }
107
108   let filtered = data.filter(
109     (obj) => obj["EAN 1" || "EAN 2" || "EAN 3"] == ctx.query.codigo
110   );
111
112   if (filtered.length > 0) {
113     ctx.body = filtered;
114   } else {
115     ctx.status = 404;
116     ctx.body = { message: "Parametro Inválido" };
117   }
118 }

```

○

```

119
120 // busca medicamento pelo código GGREM
121 function getByGgrem(ctx) {
122
123
124   if (/[0-9]+/g.test(ctx.query.ggrem)) {
125   } else {
126     ctx.body = { message: "Parametro inválido" };
127     ctx.status = 400;
128     return;
129   }
130
131   let filtered = data.filter((obj) => obj["CÓDIGO GGREM"] == ctx.query.ggrem);
132
133   if (filtered.length > 0) {
134     ctx.body = filtered;
135   } else {
136     ctx.status = 404;
137     ctx.body = { message: "Not Found" };
138   }
139 }
140

```

○

Uma boa prática é separarmos em arquivos as rotas, as funções, o arquivo que terá somente as chamadas.

Nesta Api ficou tudo no arquivo app.js.

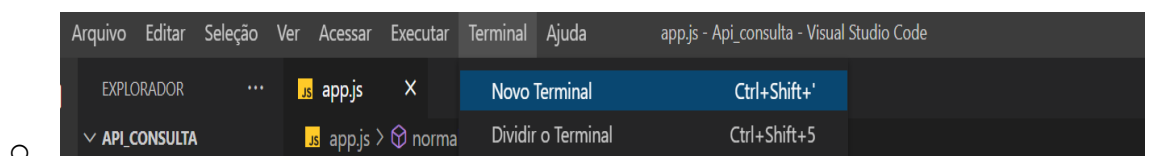
Devemos também sempre ir salvando as alterações feitas no código durante o desenvolvimento.

Até este ponto fizemos o que chamamos de Back End.

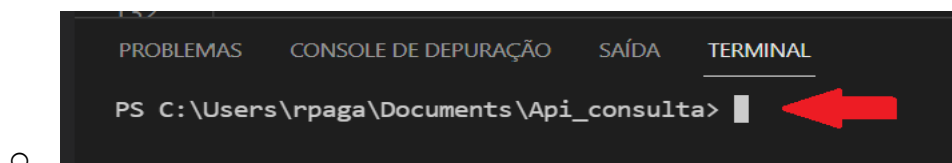
Back End, como o próprio nome sugere, vem da idéia do que tem por trás de uma aplicação. Pode ficar meio abstrato em um primeiro momento, mas pense que para conseguir usar o Facebook no dia a dia, os dados do seu perfil, amigos e publicações precisam estar salvos em algum lugar, sendo esse lugar um banco de dados e processados a partir de lá. Não basta apenas o Front End em HTML e CSS! O Back End trabalha em boa parte dos casos fazendo a ponte entre os dados que vem do navegador rumo ao banco de dados e vice-versa, sempre aplicando as devidas regras de negócio, validações e garantias em um ambiente onde o usuário final não tenha acesso e possa manipular algo.

Podemos testar se as consultas já estão funcionando, sem ter o Front(a parte bonita de interação) para isso precisamos fazer o seguinte:

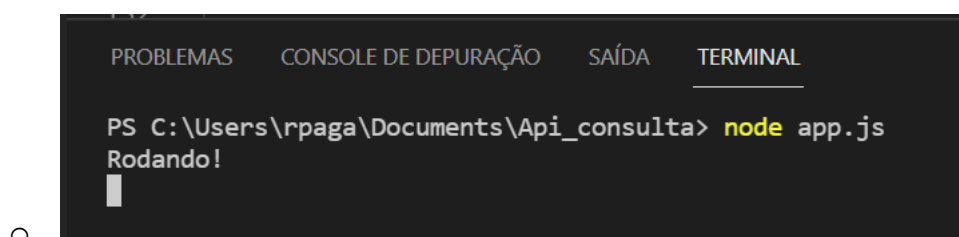
✓ Abra o terminal no VsCode:



- Certifique-se que o caminho do terminal esteja correto, que a pasta da Api esteja aberta:



- Precisamos ativar o servidor local com o comando **node** e o nome do arquivo que estão nossas configurações. Se correr tudo bem, se tudo estiver certo, vai aparecer a mensagem que colocamos lá no código “Rodando”:



- ✓ Com o servidor rodando, abra o navegador e digite uma das rotas que foram especificadas, por exemplo:

- `http://localhost:3000/get_by_name/?name=paracetamol`

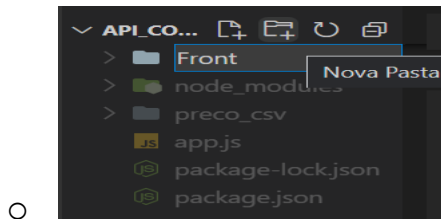
```
PRINCÍPIO ATIVO: "PARACETAMOL;FOSFATO DE CODEÍNA",
CNPJ: "61.190.096/0001-92",
LABORATÓRIO: "EUROFARMA LABORATÓRIOS S.A.",
CÓDIGO GGREM: 508028401112111,
REGISTRO: 1004311350087,
EAN 1: 7891317000110,
EAN 2: "-",
EAN 3: "-",
PRODUTO: "PARACETAMOL + FOSFATO DE CODEÍNA",
APRESENTAÇÃO: "500MG + 30MG COM CT BL AL PVC/PCTFE TRANS X 12",
CLASSE TERAPÊUTICA: "N2A - ANALGÉSICOS NARCÓTICOS",
TIPO DE PRODUTO (STATUS DO PRODUTO): "Genérico",
REGIME DE PREÇO: "Regulado",
PF Sem Impostos: "11,73",
PF 0%: "13,14",
PF 12%: "15,18",
PF 17%: "16,24",
PF 17% ALC: "14,14",
PF 17,5%: "16,35",
PF 17,5% ALC: "14,22",
PF 18%: "16,47",
PF 18% ALC: "14,31",
PF 20%: "16,94",
PMVG Sem Impostos: "9,20",
PMVG 0%: "10,31",
PMVG 12%: "11,91",
PMVG 17%: "12,74",
PMVG 17% ALC: "11,10",
PMVG 17,5%: "12,83",
PMVG 17,5% ALC: "11,16",
PMVG 18%: "12,92",
PMVG 18% ALC: "11,23",
PMVG 20%: "13,29",
RESTRIÇÃO HOSPITALAR: "Não",
CAP: "Não",
CONFAZ 87: "Não",
ICMS 0%: "Não",
ANÁLISE RECURSAL: "",
LISTA DE CONCESSÃO DE CRÉDITO TRIBUTÁRIO (PIS/COFINS): "Negativa",
COMERCIALIZAÇÃO 2020: "Sim",
TARJA: "Tarja Vermelha",
: "",
__1: 0,
```

○

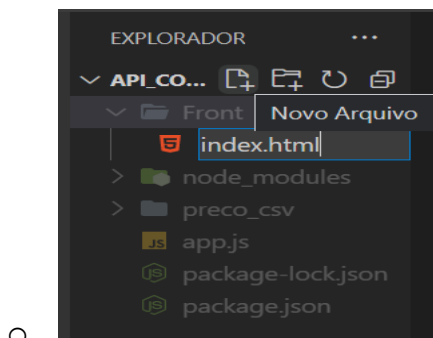
- Vai retornar todos os medicamentos que contenham a palavra paracetamol.
- Após fazer as consultas devemos parar o servidor local, se fizermos qualquer alteração no código, também devemos parar e reativá-lo novamente para ficar atualizado.
- O comando usado para parar o servidor é **Ctrl + C**, vai aparecer desta forma: `PS C:\Users\rpaga\Documents\Api_consulta> ^C`, e para voltar a rodar **node app.js**.

Agora vamos ver o Front End que é o que podemos classificar como a parte visual de um site, aquilo que nós como usuários interagimos.

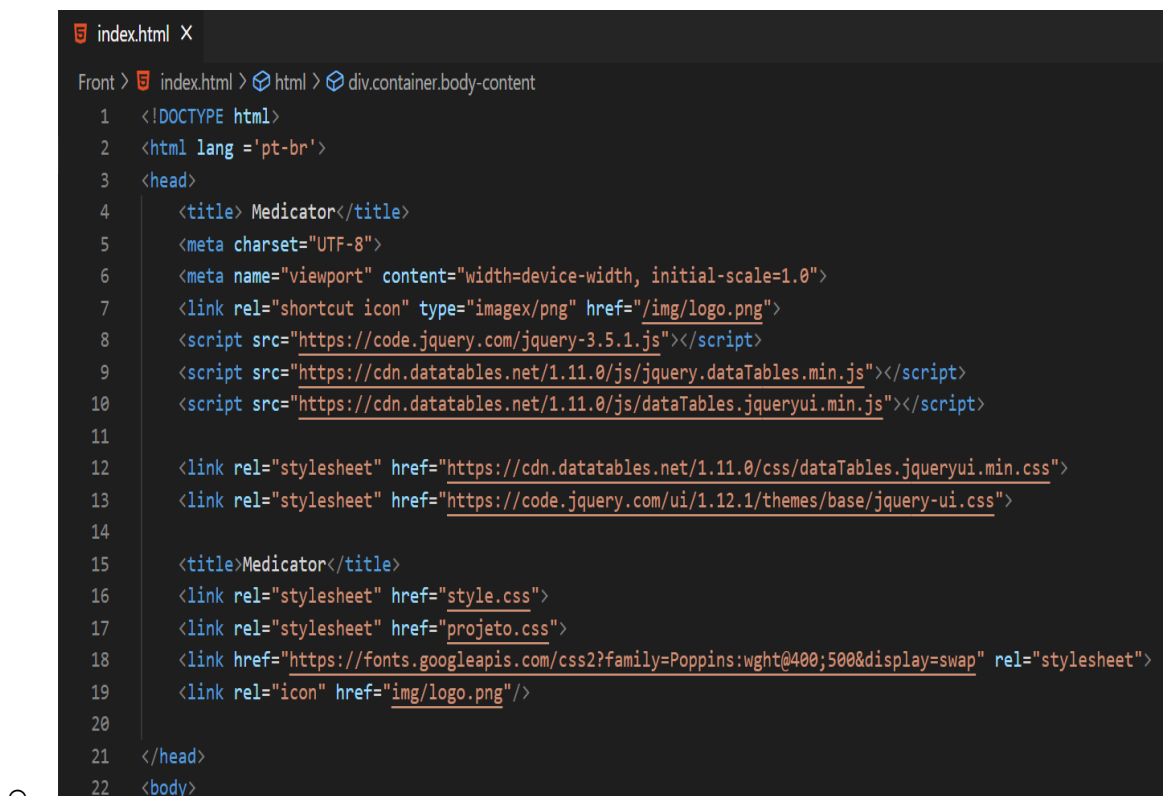
- ✓ No mesmo projeto, crie uma pasta com o nome Front:



- ✓ Dentro da pasta Front, criar um arquivo com o nome index.html:



- ✓ O arquivo index.html terá toda a estrutura da página, dentro dele vamos colocar o seguinte código (observe a sequência da numeração das linhas e digite tudo exatamente igual):



```

23
24 <!-- Inicio do menu -->
25
26 <nav class="navigation-bar">
27   
28   <a href="index.html">Home</a>
29   <a href="projeto.html">Equipe</a>
30   <a href="contato.html">Contato</a>
31   <a href="https://www.notion.so/Documenta-o-API-8edfc00857244bedab6c0607b47f73d9">
32     <button class="buttonD">Documentação </button>
33
34   </a>
35 </nav>
36
37 <div class='login-page'>
38   <div class='form' ng-app="dmExemploApp">
39     <h3>Consulte agora</h3> <br>
40
41     <div>

```

```

42       <!-- codigo de barras -->
43       <ul class="confirmacoes">
44         <li>
45           <label for='Barras'>
46             <input type='checkbox' class="label-input" name='rdo' id='barras'>
47             <span class='label'></span> Código de Barras
48           </label>
49         </li>
50         <li>
51           <!-- nome do produto -->
52           <label for='nome' class="label-input">
53             <input type='checkbox' name='rdo' id='nome'>
54             <span class='checkmark'></span> Nome do Produto
55           </label>
56         </li>
57
58         <li>
59           <label for='nome' class="label-input">
60             <input type='checkbox' name='rdo' id='subs'>
61             <span class='checkmark'></span> Princípio Ativo
62           </label>
63

```

```

64       </li>
65
66       <li>
67         <!-- Código do GGREM -->
68         <label for='nome' class="label-input" id="last-label-input">
69           <input type='checkbox' name='rdo' id='ggrem'>
70           <span class='checkmark'></span> Código GGREM
71         </label>
72       </li>
73     </ul>
74     <!-- Input + botão de Enviar -->
75     <input type='text' placeholder='Pesquisar' id='search' name='search'>
76     <i class="fas fa-lock icon-modify"></i><br>
77     <button type='submit' class="btn btn-second">Enviar</button>
78   </div ng-controller="ctrlExemplo">
79 </div>
80 </form>
81 </div>
82

```

```

83
84     <div id="result">
85
86     </div>
87
88     </body>
89     <div class="container body-content">
90
91     </div>
92     <footer class="fixar-rodape"> <br>
93     |     <p>&copy; Desenvolvido pelo time L3 </p>
94     </footer>
95 </html>

```

```

96
97     <script>
98     $(".btn").click(async function(){
99         await local(document.getElementById('search').value);
100         $("#table_id").DataTable( {
101             scrollY: 300,
102             "stripeClasses": [],
103             "language": {
104                 "lengthMenu": "Mostrar _MENU_ linhas por página",
105                 "zeroRecords": "Nenhum resultado encontrado",
106                 "info": "Mostrando _PAGE_ de _PAGES_ páginas",
107                 "infoEmpty": "Páginas não encontradas",
108                 "infoFiltered": "(Filtrado de _MAX_ total linhas)",
109                 "search": "Procurar:",
110                 "paginate": {
111                     "first": "Primeiro",
112                     "last": "Último",
113                     "next": "Próximo",
114                     "previous": "Anterior"
115                 },
116             },
117         });
118     })
119 </script> <!-- JQUERY DATA TABLE -->
120

```

```

121 <script>
122     // limita a Quantidade de cliques.
123     $(document).ready(function () {
124         $('input[type=checkbox]').change(function() {
125             $('input[type=checkbox]:checked').not(this).prop('checked', false);
126         });
127     });
128 </script>
129
130 <script src = 'main.js'></script>
131

```

- ✓ Esse arquivo index.html faz a marcação da página. Se abrissemos apenas ele no navegador ficaria desta forma:



Consulte agora

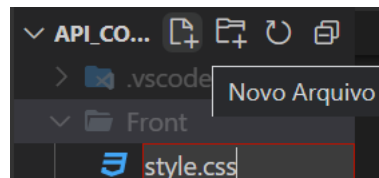
- ☐ Código de Barras
- ☐ Nome do Produto
- ☐ Princípio Ativo
- ☐ Código GGREM

© Desenvolvido pelo time L3

○

- ✓ Precisamos definir as cores, fontes, estilos por isso vamos criar outro arquivo:

- Dentro da pasta Front, criar um arquivo com o nome style.css:



○

- E colocar o seguinte código:

```
1  /* Configuração do corpo do site */
2  body {
3      zoom: 90%;
4      background-color: #d5d9df;
5      font-family: "Roboto", sans-serif;
6  }
7
8  @import url(https://fonts.googleapis.com/css?family=Roboto:300);
9
10 .login-page {
11     width: 360px;
12     padding: 5% 0 0;
13     margin: auto;
14     margin-top: -20px;
15 }
16
17 .login-page:hover {
18     box-shadow: 0 5px 0px rgba(0, 0, 0, 0.8);
19 }
20
```

○


```

21  /* configurações do formulario de pesquisa */
22  .form {
23      /* position: ; */
24      z-index: 1;
25      background: #ffffff;
26      max-width: 360px;
27      margin: 0 auto 100px;
28      padding: 45px;
29      text-align: center;
30      box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
31  }
32
33  /* Caixa de pesquisa */
34  .form input {
35      font-family: "Roboto", sans-serif;
36      outline: 0;
37      background: #ffffff;
38      width: 100%;
39      border: 1px solid #ccc;
40      margin: 0 0 16px;
41      padding: 15px;
42      box-sizing: border-box;
43      font-size: 14px;
44  }
45

```

```

46  /* Botão de pesquisa */
47  .form button {
48      font-family: "Roboto", sans-serif;
49      text-transform: uppercase;
50      outline: 0;
51      background-color: #0069d9;
52      width: 100%;
53      border: 0;
54      padding: 10px;
55      color: #ffffff;
56      font-size: 14px;
57      -webkit-transition: all 0.3 ease; /* controla a velocidade de uma animação*/
58      transition: all 0.3 ease; /*definir a transição entre dois estados de um elemento*/
59      cursor: pointer; /* controla a aparência do cursor quando estiver localizado sobre o elemento*/
60  }
61
62  /*efeitos no botão de enviar*/
63  .form button:hover,
64  .form button:active,
65  .form button:focus {
66      background: #2c4af1;
67      background-color: #0069d9;
68      color: white;
69      box-shadow: 0 10px 20px 0 rgba(80, 123, 252, 0.5);
70  }

```

```

71
72     .form .message {
73         margin: 15px 0 0;
74         color: #e6eff0;
75         font-size: 12px;
76     }
77
78     .form .message a {
79         color: #000000;
80         text-decoration: none;
81     }
82
83     .form .register-form {
84         display: none;
85     }
86
87     @media screen and (min-width: 800px) {
88         .container {
89             margin: 1em 2em;
90         }
91     }

```

```

92
93     /* posição do .container */
94     .container {
95         position: relative;
96         z-index: 1; /* z-index propriedade especifica a ordem da pilha de um elemento */
97         max-width: 300px; /* define a largura maxima */
98         margin: 0 auto; /* Significa que o browser dará uma margem automática para todos os lados */
99     }
100
101     .container:before,
102     .container:after {
103         content: "";
104         display: block; /* faz com que o elemento HTML seja renderizado como bloco */
105         clear: both;
106     }
107
108     .container .info {
109         margin: 50px auto;
110         text-align: center;
111     }
112
113     .container .info h1 {
114         margin: 0 0 15px;
115         padding: 0;
116         font-size: 36px;
117         font-weight: 300;
118         color: #1a1a1a;
119     }
120

```

```

121     .container .info span {
122         color: #4d4d4d;
123         font-size: 12px;
124     }
125
126     .container .info span a {
127         color: #000000;
128         text-decoration: none;
129     }
130
131     /* footer */
132
133     div.body-content {
134         /* Essa margem vai evitar que o conteúdo fique por baixo do rodapé */
135         margin-bottom: 20px;
136     }

```

```

137
138  /* footer */
139
140  footer {
141      width: 100%;
142      height: 50px;
143      margin: auto;
144      bottom: 0;
145      position: fixed;
146      text-align: center;
147      background-color: #ffffff;
148      box-shadow: 0 0 8px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
149  }
150
151  /* Encerra o footer */
152  .confirmacoes li {
153      font-size: 14px;
154      display: flex;
155  }
156
157  .confirmacoes input {
158      width: 16px;
159      height: 16px;
160      margin-right: 5px;
161      position: relative;
162  }

```

```

163
164  @media only screen and (min-width: 480px) {
165      .navigation-bar {
166          width: 100%;
167          height: 80px;
168          box-shadow: 0 0 8px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
169      }
170
171      .logo {
172          display: inline-block;
173          vertical-align: top;
174          margin-right: 90px;
175          margin-top: -20px;
176          max-height: 150px;
177          max-width: 200px;
178          width: 120px;
179          height: 120px;
180      }
181      .navigation-bar > a {
182          display: inline-block;
183          vertical-align: top;
184          margin-right: 30px;
185          line-height: 70px;
186      }

```

```
187 .logo {
188     float: left;
189     margin-top: -8px;
190 }
191
192 /* ~~ Top Navigation Bar ~~ */
193 #navigation-container {
194     width: 100% / px;
195     margin: 0 auto;
196     height: 70px;
197 }
198
199 .navigation-bar {
200     background-color: ■ #ffffff;
201     height: 70px;
202     width: 100%;
203     width: calc(50% + 583 - 366);
204     border-radius: □ #000;
205 }
206
207 #navigation-container img {
208     float: left;
209 }
210
```

```
211 #navigation-container ul {
212     padding: 0px;
213     margin: 0px;
214     text-align: center;
215     display: inline-block;
216 }
217
218 #navigation-container li {
219     list-style-type: none;
220     padding: 0px;
221     height: 24px;
222     margin-top: 4px;
223     margin-bottom: 4px;
224     display: inline;
225 }
226
227 #navigation-container li a {
228     color: ■ white;
229     font-size: 16px;
230     font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
231     text-decoration: none;
232     line-height: 70px;
233     padding: 5px 15px;
234     opacity: 0.7;
235 }
```

```

236
237     #menu {
238         float: right;
239     }
240 }
241
242 a {
243     text-decoration: none;
244     color: #000;
245 }
246
247 a:hover {
248     text-decoration: underline;
249     color: #0069d9;
250     text-transform: uppercase;
251     text-decoration: none;
252 }

```

```

253
254 @import url(https://fonts.googleapis.com/css?family=Roboto:400,300,600,400italic);
255 * {
256     margin: 0;
257     padding: 0;
258     box-sizing: border-box;
259     -webkit-box-sizing: border-box;
260     -moz-box-sizing: border-box;
261     -webkit-font-smoothing: antialiased;
262     -moz-font-smoothing: antialiased; /* Renderização em subpixel gera um melhor resultado em monitores atuais, */
263     -o-font-smoothing: antialiased; /*Permita que o navegador selecione uma otimização para suavização de fonte*/
264     text-rendering: optimizeLegibility; /*otimizar a velocidade do carregamento dos textos*/
265 }
266
267 .container {
268     max-width: 370px;
269     width: 100%;
270     margin: 0 auto;
271     /* position: relative; */
272 }
273
274 #result {
275     margin: 10px;
276     margin-bottom: 100px;
277 }

```

```

278
279 table,
280 th,
281 td {
282     border: 2px solid black;
283     border-collapse: collapse;
284     padding: 5px;
285 }
286
287 th {
288     width: 10px;
289     text-align: left;
290 }
291
292 thead {
293     color: white;
294 }
295

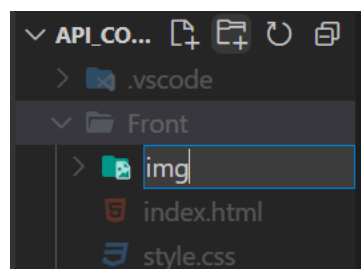
```

```

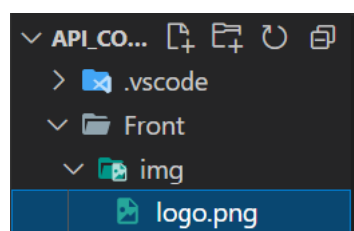
296 .buttonD {
297     font-family: "Roboto", sans-serif;
298     text-transform: uppercase;
299     outline: 0;
300     background-color: #0069d9;
301     width: 100%;
302     border: 0;
303     padding: 10px;
304     color: #ffffff;
305     font-size: 14px;
306     -webkit-transition: all 0.3 ease;
307     transition: all 0.3 ease;
308     cursor: pointer;
309     margin-left: 500%;
310     min-width: 50px;
311     max-width: 300px;
312 }
313
314 .buttonD:hover,
315 .buttonD:active,
316 .buttonD:focus {
317     background: #2c4af1;
318     background-color: #0069d9;
319     color: white;
320     box-shadow: 0 10px 20px 0 rgba(80, 123, 252, 0.5);
321 }
322

```

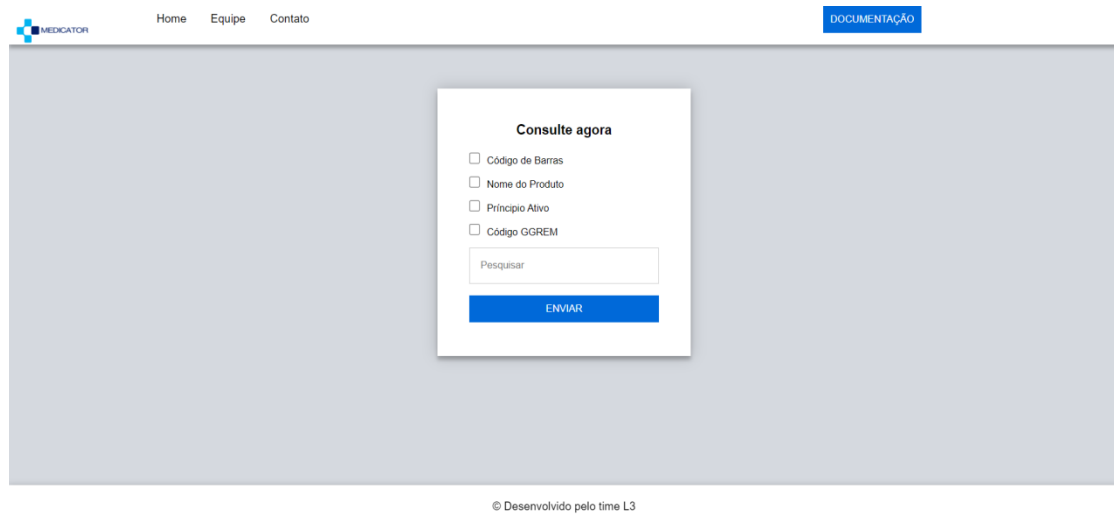
- ✓ Precisamos também colocar o logo na nossa api. Sempre colocamos as imagens em uma pasta separada e dentro da pasta, no nosso caso, Front:



- Procure por uma imagem e salve-a dentro desta pasta:



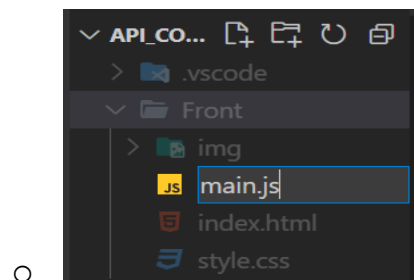
- ✓ Agora com estes dois arquivos, a aparência fica assim:



- Observe e compare as mudanças.

Mas ainda não está pronto! Precisamos ligar o Front End com o nosso Back End (index.html e app.js).

- ✓ Precisamos de mais um arquivo, vamos chamá-lo de mais.js:



- E colocar o seguinte código:

```
1  const local = async (input) => {
2    if (nome.checked) {
3      if (input == "") {
4        window.alert("Por favor, informe o nome do produto.");
5      } else if (!/[A-Z0-9]/gi.test(input)) {
6        window.alert("Apenas caracteres alfanuméricos são permitidos!");
7      } else {
8        try {
9          let response = await fetch(
10             "http://localhost:3000/get_by_name/?name=" + input
11           ).then((body) => body.json());
12          createTableFromJSON(response);
13        } catch (error) {
14          console.log(error); /* Tratar error */
15        }
16      }
17    } else if (barras.checked) {
18      if (input == "") {
19        window.alert("Por favor, informe o código de barras do produto.");
20      } else if (!/[0-9]/gi.test(input)) {
21        window.alert("Apenas caracteres numéricos são permitidos!");
22      } else {
```

```

23     try {
24         let response = await fetch(
25             "http://localhost:3000/get_by_cod/?codigo=" + input
26         ).then((body) => body.json());
27         createTableFromJSON(response);
28     } catch (error) {
29         console.log(error); /* Tratar error */
30     }
31 }
32 } else if (subs.checked) {
33     if (input == "") {
34         window.alert("Por favor, informe a nome do princípio Ativo");
35     } else if (!/[A-Z0-9]/gi.test(input)) {
36         window.alert("Apenas caracteres alfanuméricos são permitidos!");
37     } else {
38         try {
39             let response = await fetch(
40                 "http://localhost:3000/get_by_active/?active=" + input
41             ).then((body) => body.json());
42             createTableFromJSON(response);
43         } catch (error) {
44             console.log(error); /* Tratar error */
45         }
46     }

```

```

47 } else if (ggrem.checked) {
48     if (input == "") {
49         window.alert("Por favor, informe o código ggrem.");
50     } else if (!/[0-9]/gi.test(input)) {
51         window.alert("Apenas caracteres numéricos são permitidos!");
52     } else {
53         try {
54             let response = await fetch(
55                 "http://localhost:3000/get_by_ggrem/?ggrem=" + input
56             ).then((body) => body.json());
57             createTableFromJSON(response);
58         } catch (error) {
59             console.log(error); /* Tratar error */
60         }
61     }
62 } else {
63     window.alert("Por favor, selecione um campo!");
64 }
65 };

```

```

66
67 function createTableFromJSON(response) {
68     var divContainer = document.getElementById("result");
69
70     var col = [];
71     for (var i = 0; i < response.length; i++) {
72         for (var key in response[i]) {
73             if (col.indexOf(key) === -1) {
74                 col.push(key);
75             }
76         }
77     }

```

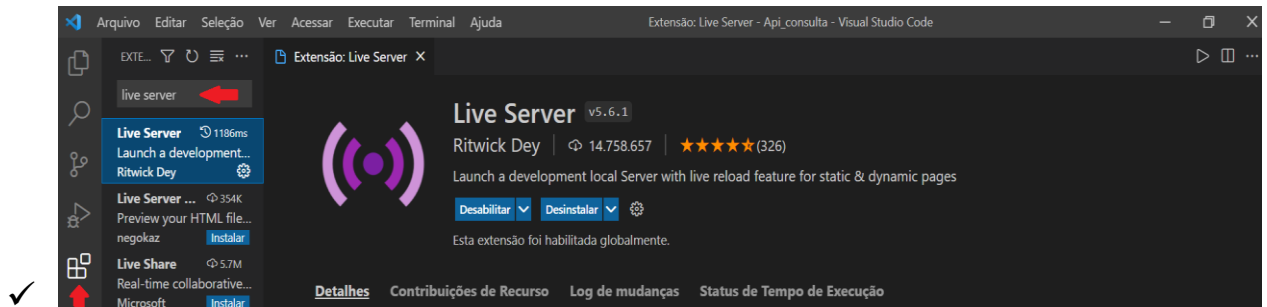
```

78
79 // CREATE DYNAMIC TABLE.
80 var table = document.createElement("table");
81 let theader = table.createTHead(); /* JQUERY DATA TABLE */
82 let tbody = table.createTBody(); /* JQUERY DATA TABLE */
83 table.id = "table_id"; /* JQUERY DATA TABLE */
84 table.className = "display"; /* JQUERY DATA TABLE */
85
86 // CREATE HTML TABLE HEADER ROW USING THE EXTRACTED HEADERS ABOVE.
87
88 var tr = theader.insertRow(-1); // TABLE ROW.
89
90 let header = [
91     "Princípio Ativo",
92     "Laboratorio",
93     "Produto",
94     "Apresentação",
95     "Tipo Produto",
96     "Regime Preço",
97     "Preço Final",
98     "Restricao Hospitalar",
99     "Tarja",
100 ];
101
102 for (var i = 0; i < header.length; i++) {
103     var th = document.createElement("th"); // TABLE HEADER.
104     th.innerHTML = header[i];
105     tr.appendChild(th);
106 }
107
108 // ADD JSON DATA TO THE TABLE AS ROWS.
109 for (var i = 0; i < response.length; i++) {
110     tr = tbody.insertRow(-1);
111
112     for (var j = 0; j < col.length; j++) {
113         if (
114             j == 0 ||
115             j == 2 ||
116             j == 8 ||
117             j == 9 ||
118             j == 11 ||
119             j == 12 ||
120             j == 13 ||
121             j == 32 ||
122             j == 39
123         ) {
124             var tabCell = tr.insertCell(-1);
125             tabCell.innerHTML = response[i][col[j]].replace(/;/gi, " ");
126         }
127     }
128 }
129
130 // FINALLY ADD THE NEWLY CREATED TABLE WITH JSON DATA TO A CONTAINER.
131 divContainer.innerHTML = "";
132 divContainer.appendChild(table);
133 }

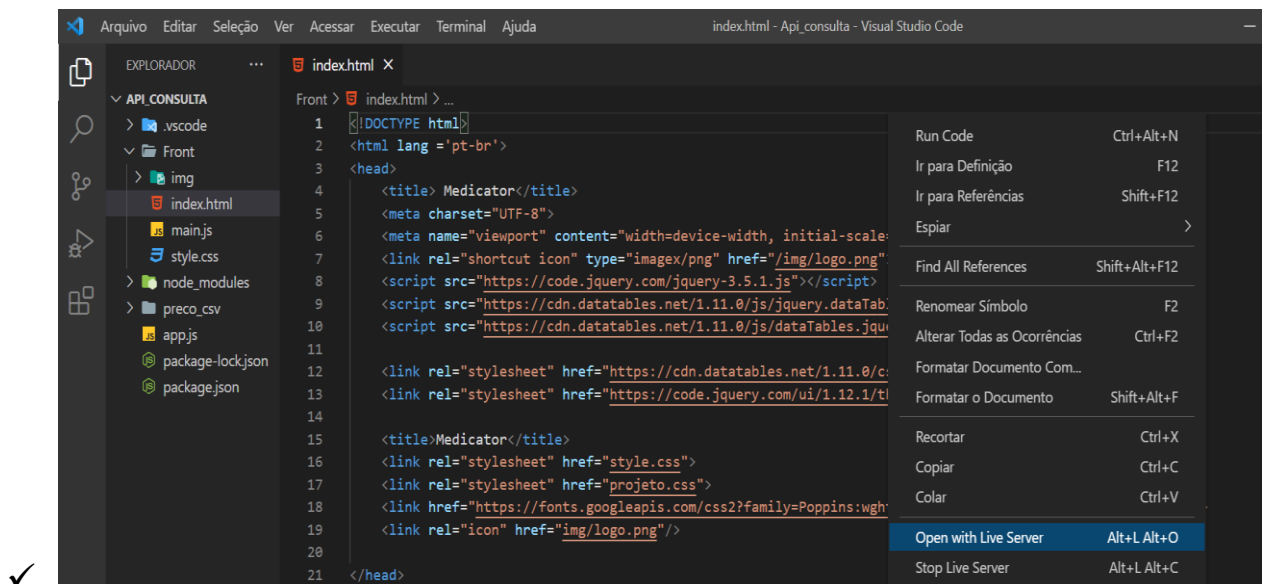
```

Para executarmos este projeto, temos mais de uma forma. Neste caso ele está sendo executado com uma extensão do Visual Code: Live Server.

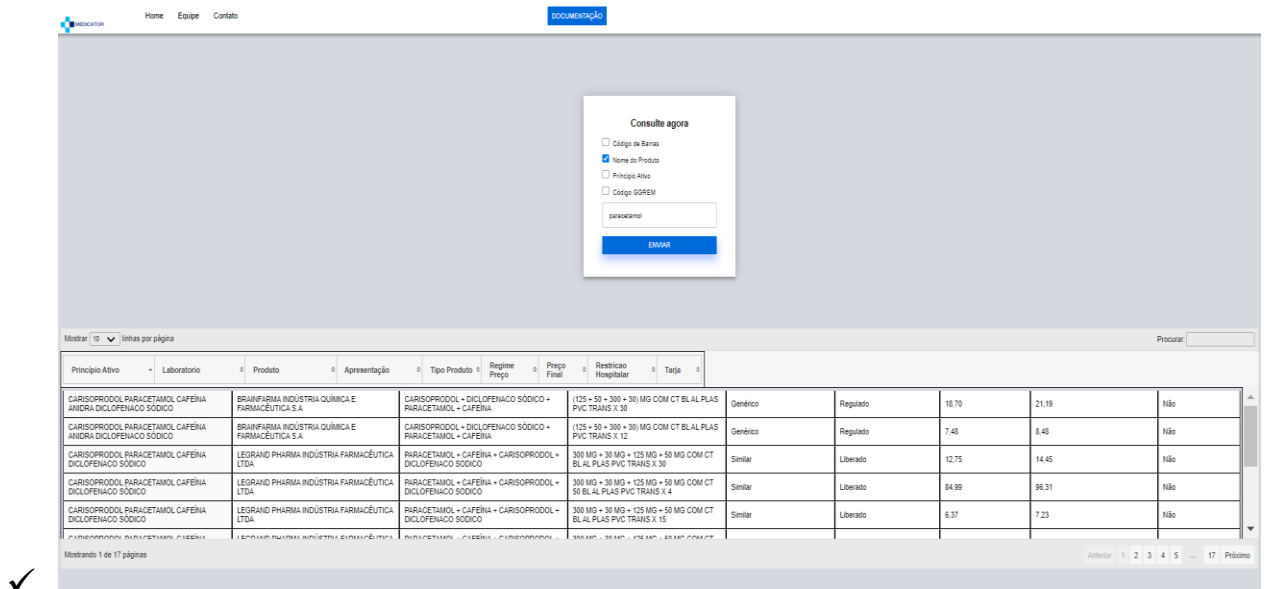
Para instalar, devemos ir em extensões e procurar pelo nome, assim que localizada só instalar.



No VsCode, clicar com o botão direito sobre o arquivo index.html e escolher a opção Open with Live Server:

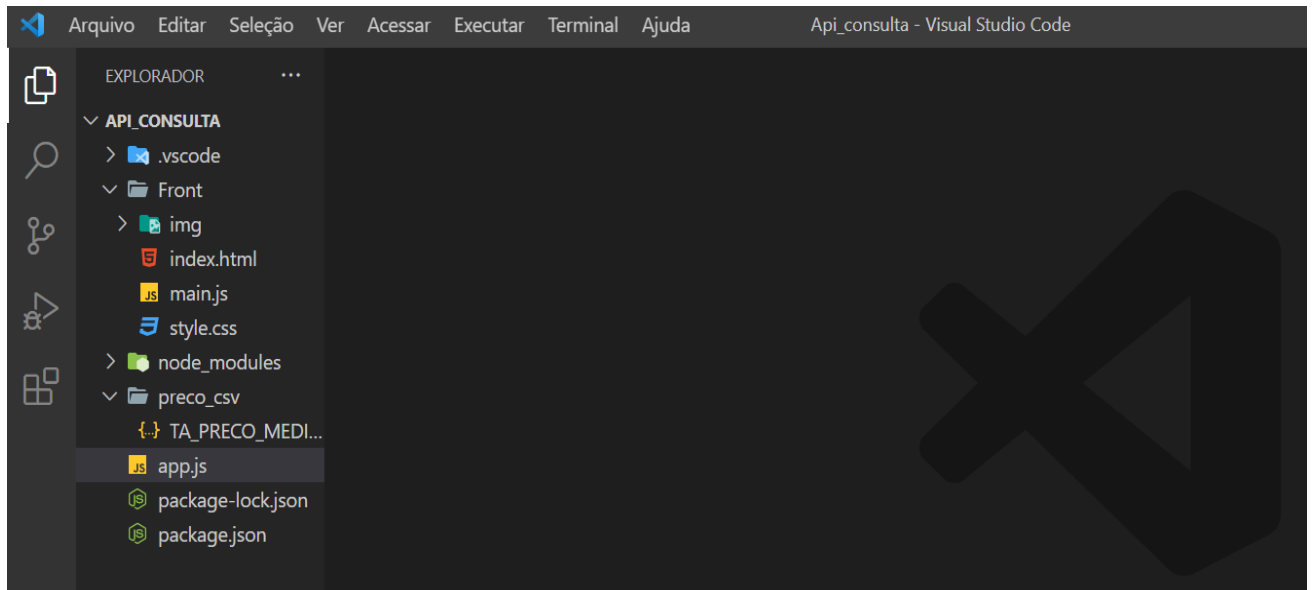


O projeto será aberto no seu navegador desta forma:



Agora é só escolher o tipo de pesquisa e digitar os dados correspondentes. Na imagem acima foi pesquisado pelo nome do produto, note que os dados aparecerão em uma tabela logo abaixo do menu de opções.

O projeto ficou com as seguintes pastas e arquivos:



Agora é com você!