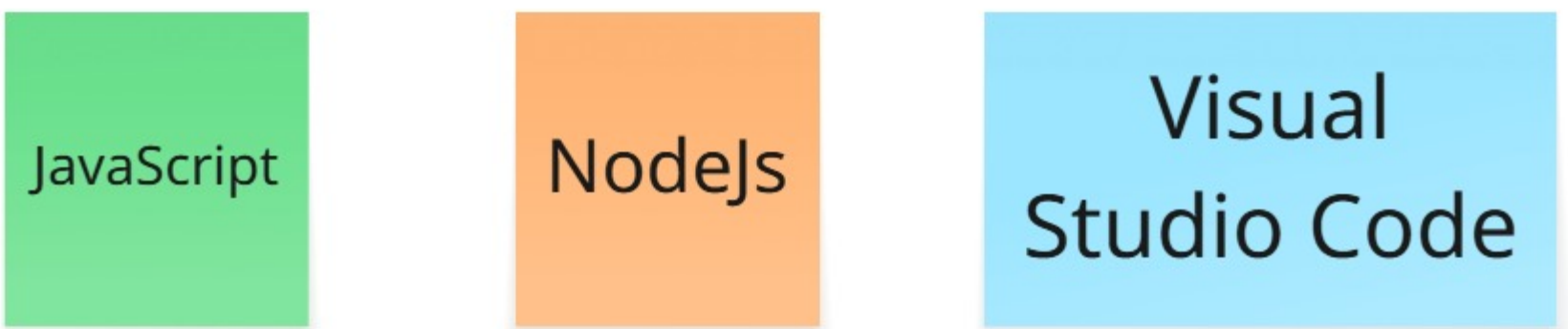
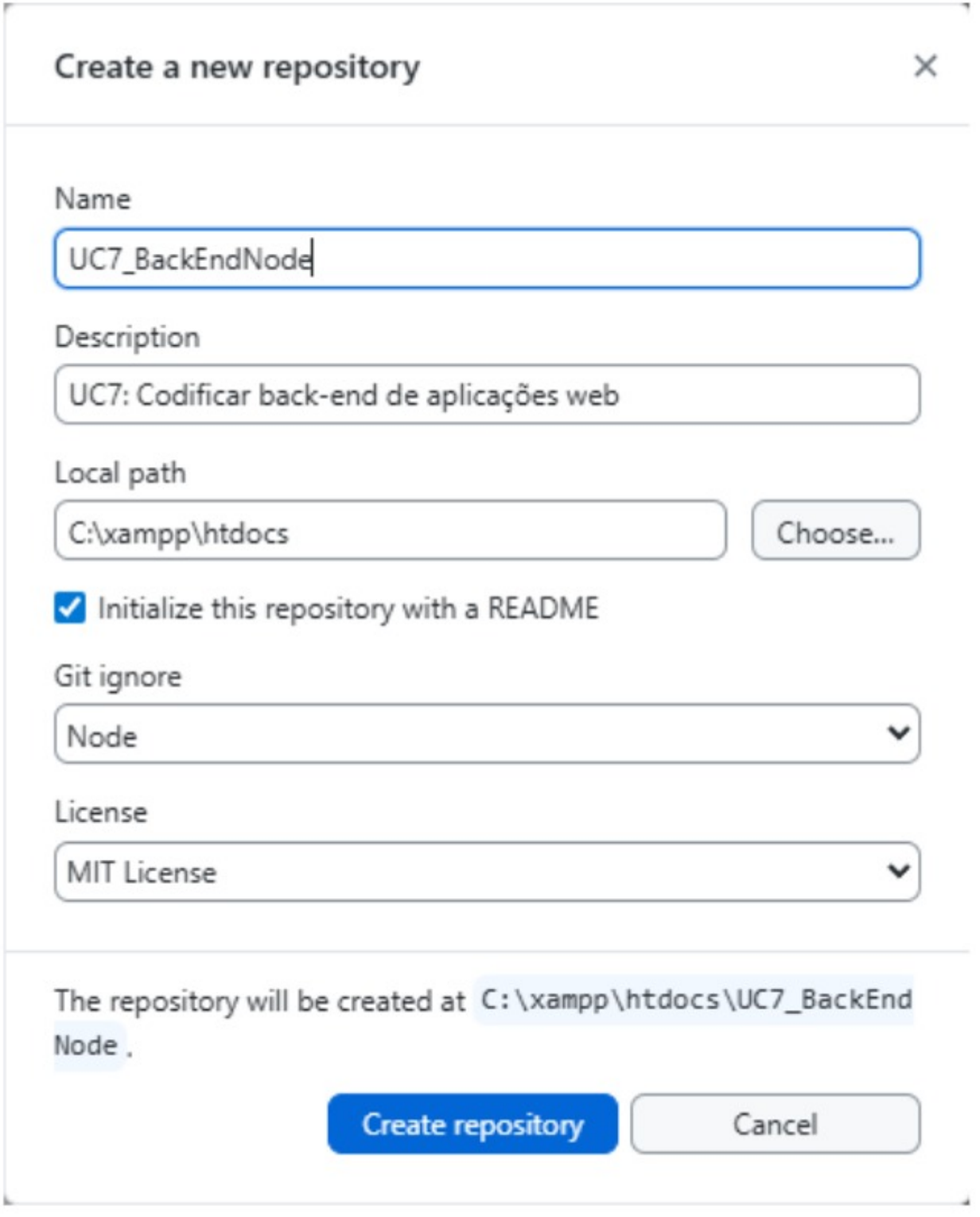
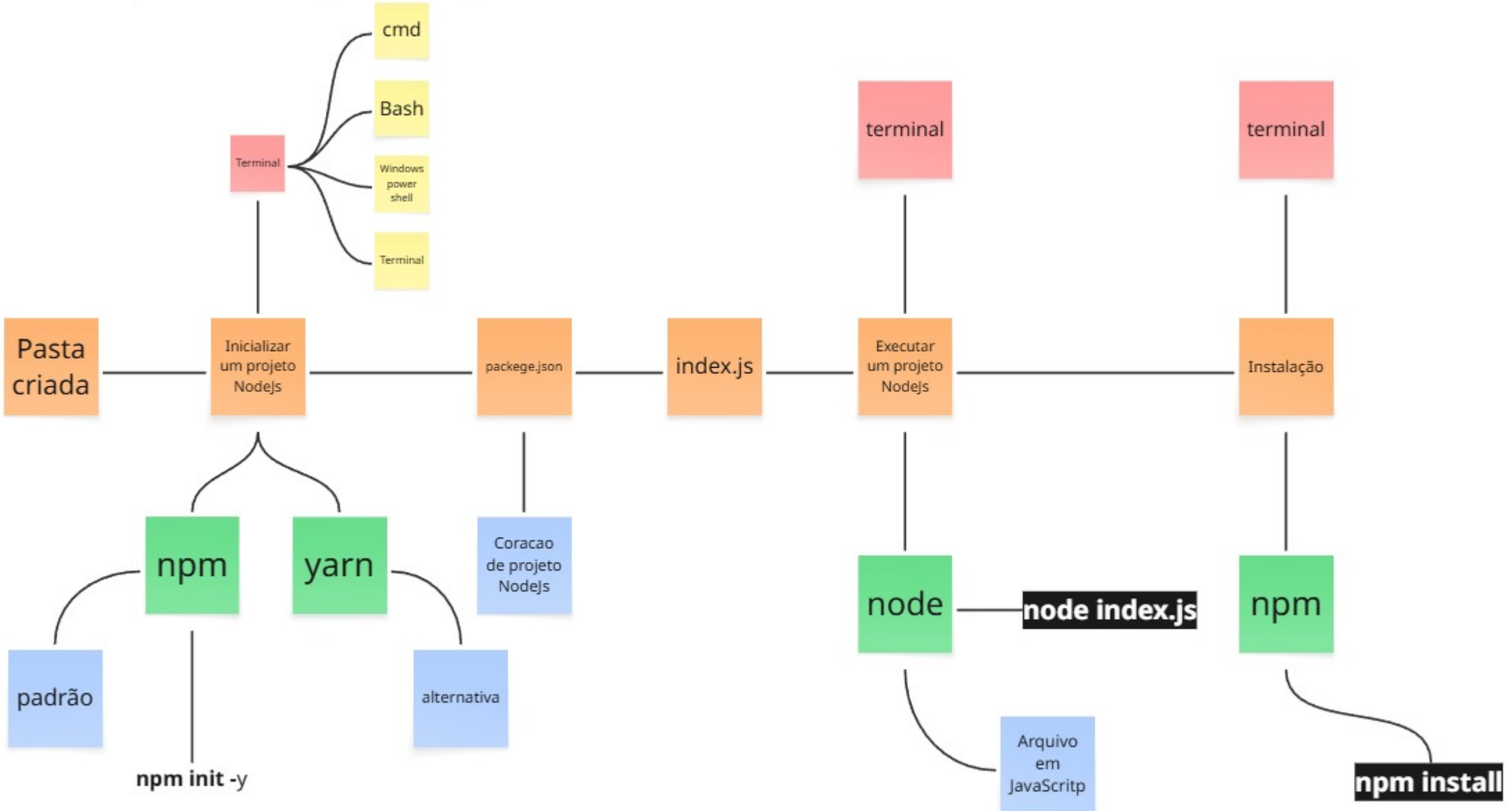
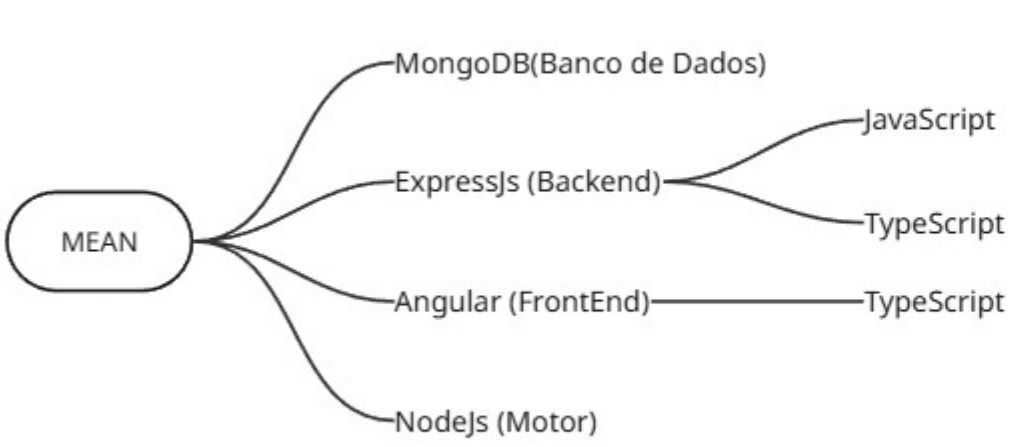

JavaScript
NodeJs
Visual
Studio Code

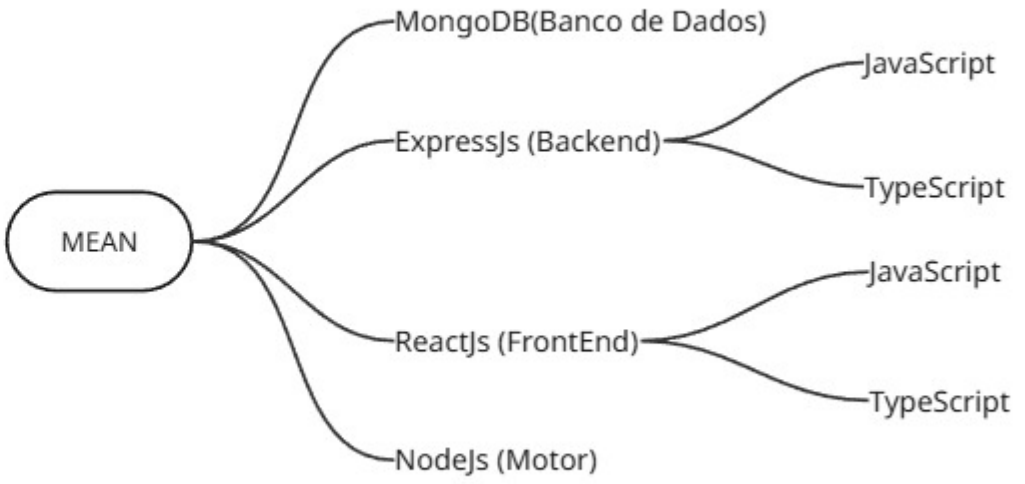

Create a new repository
Name
UC7_BackEndNode
Description
UC7: Codificar back-end de aplicações web
Local path
C:\xampp\htdocs
Choose...
Initialize this repository with a README
Git ignore
Node
License
MIT License
The repository will be created at C:\xampp\htdocs\UC7_BackEnd Node .
Create repository
Cancel

Fluxo de Criação de um Projeto NodeJs



stack

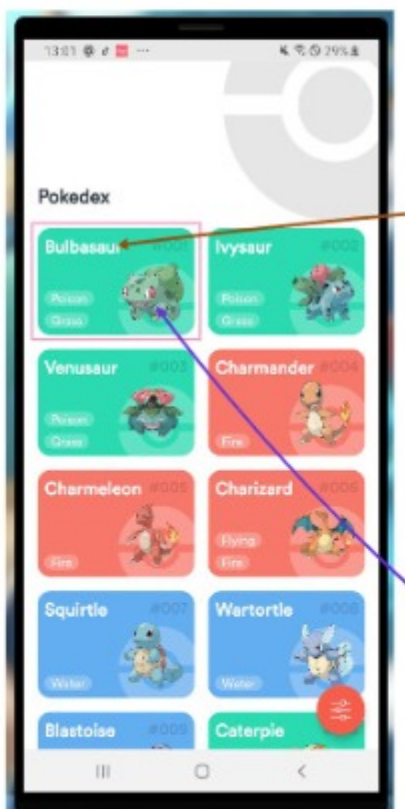

MEAN
MongoDB(Banco de Dados)
ExpressJs (Backend)
JavaScript
TypeScript
Angular (FrontEnd)
TypeScript
NodeJs (Motor)


MEAN
MongoDB(Banco de Dados)
ExpressJs (Backend)
JavaScript
TypeScript
ReactJs (FrontEnd)
JavaScript
TypeScript
NodeJs (Motor)

Direção do fluxo de dados

Site, App, Cliente, Client → Web -> HTML, CSS, JavaScript
App Android -> Java/Kotlin e XML

Frontend

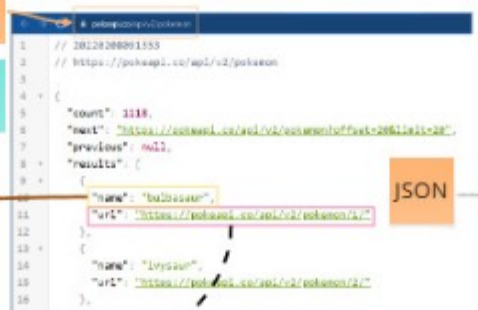


API, Servidor, Server → Python, JavaScript, Java, C#, C++, PHP, ... → JSON, XML

Backend

[GET] /pokemon

Read all



JSON

JSON é uma string (texto)

[GET] /pokemon/:id

Read by id

Database, DB → SQL (MySQL, Oracle, MariaDB, Postgres, SQLite, ...), NoSQL (MongoDB, Cassandra, Redis, ...)

Banco de dados

Express → Micro Framework → Servidor HTTP

https://www.npmjs.com/
https://www.npmjs.com/package/express

Pacotes do npm

- Biblioteca → Conjunto de códigos e arquivos que resolvem (geralmente) um problema
- Framework → Conjunto de bibliotecas que, juntas, resolvem um problema maior, através da resolução de vários problemas menores
- Micro Framework → Semelhante ao framework porém, ele não é opinativo

https://www.npmjs.com/package/is-odd

Semelhante ao framework porém, ele não é opinativo

Semelhante ao framework porém, ele não é opinativo

Index.js

2025.11.04_InicioNode > JS index.js > ...

```
1  const express = require("express")
2  const app = express()
3  app.get('/', function (req, res) {
4      res.send('Olá, Mundo')
5  })
6
7  app.listen(3000)
8
```

package.json

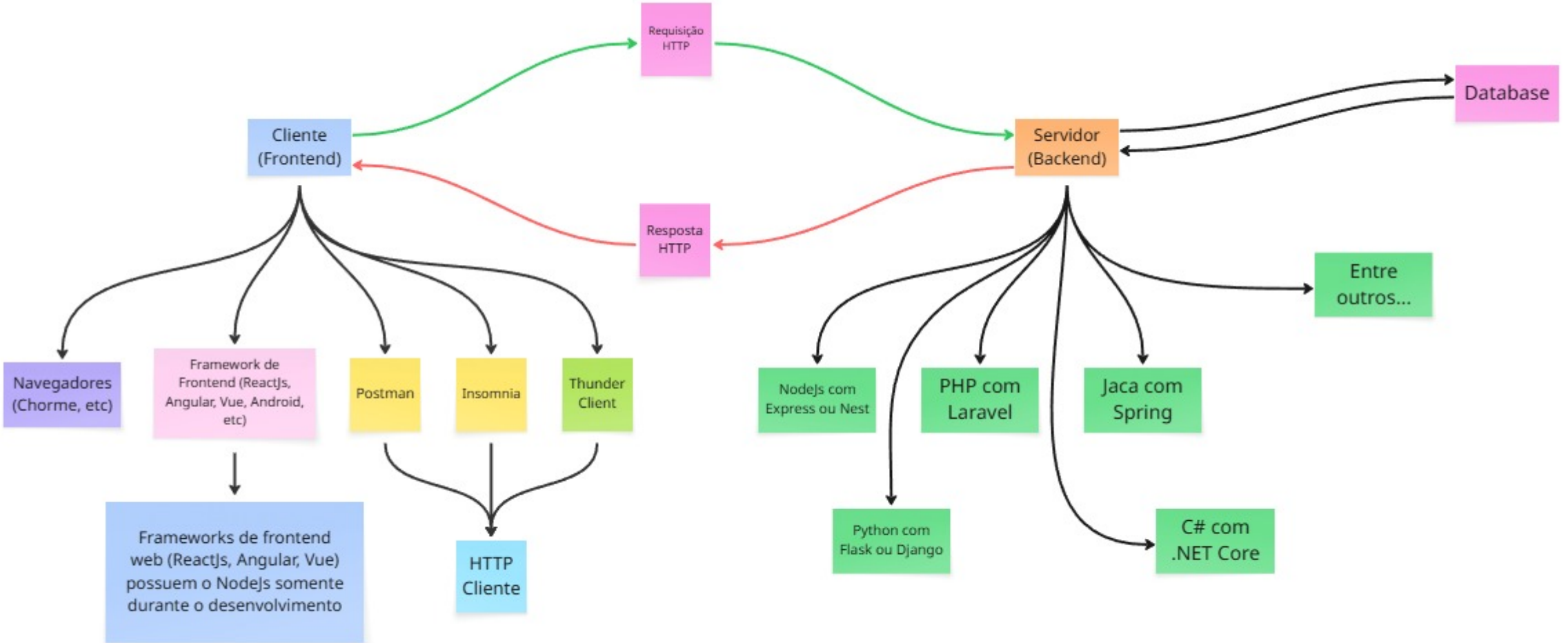
2025.11.04_InicioNode > package.json > {} scripts > start

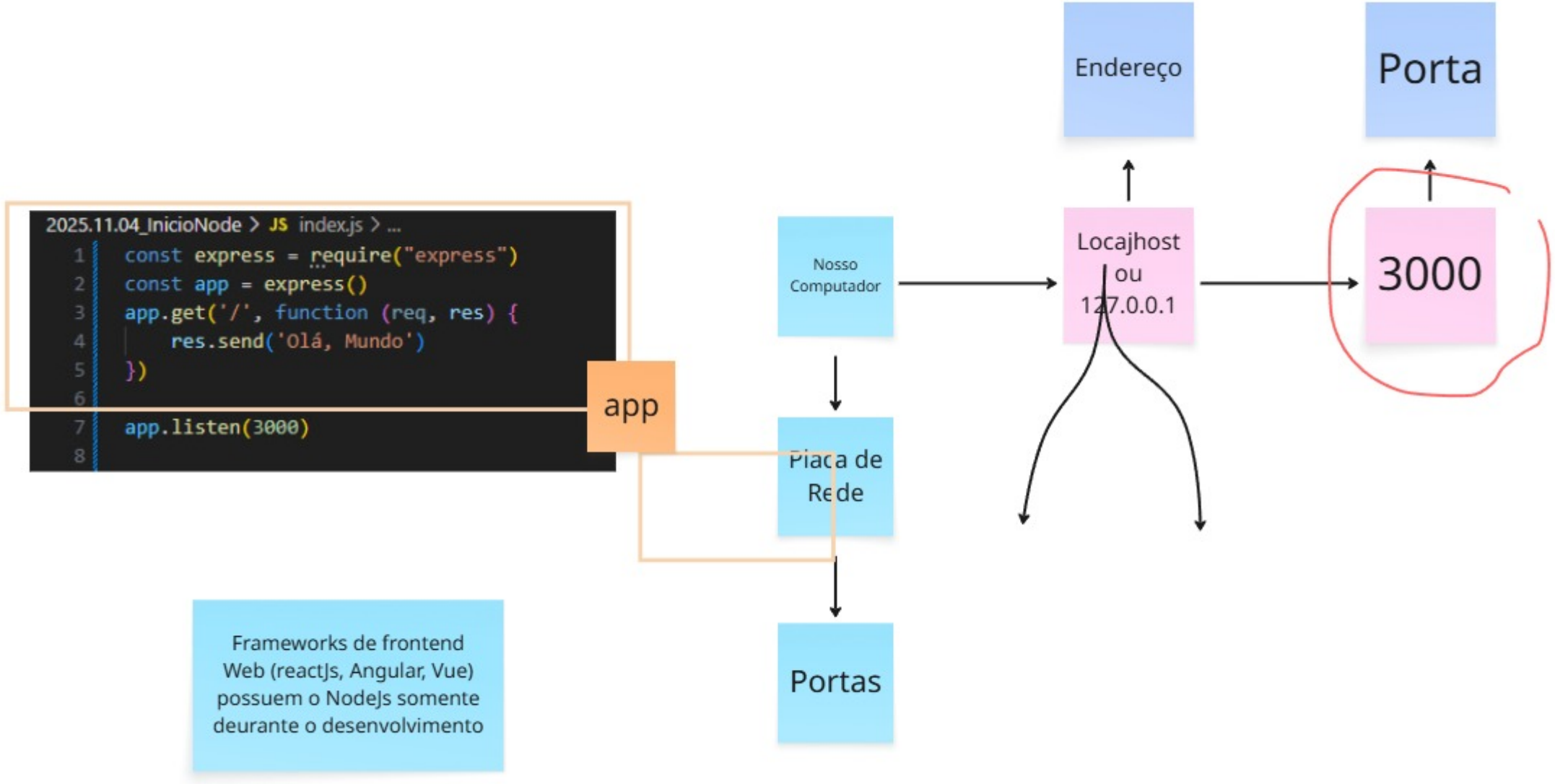
```
1  {
2    "name": "2025.11.04_inicionode",
3    "version": "1.0.0",
4    "description": "Primeiro projeto node",
5    "main": "index.js",
     ▷ Depurar
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start": "node index.js"
9    },
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "express": "^5.1.0"
14   }
15 }
```

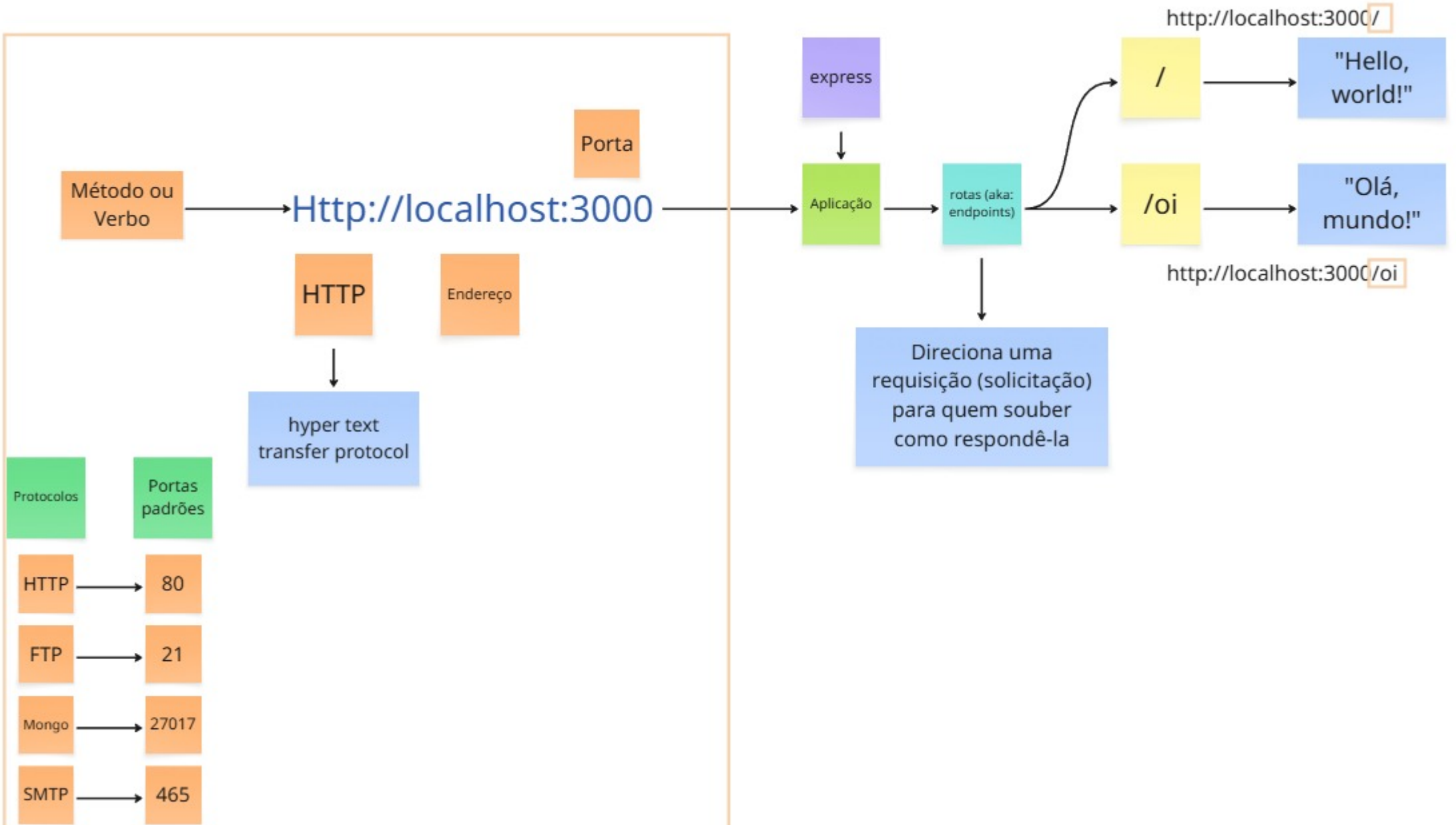
Para utilizar pacotes, precisamos instalá-los → npm install → nome_pacote → npm - install express → npm i express

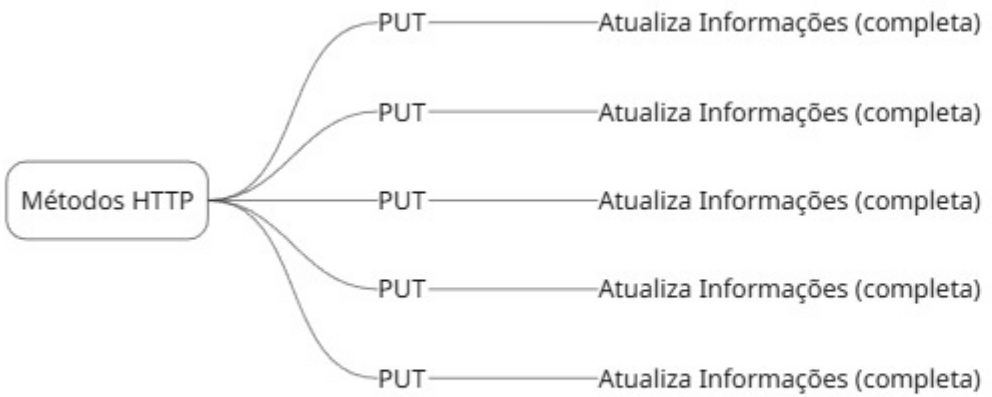
npm i

npm install --save-dev nodemon

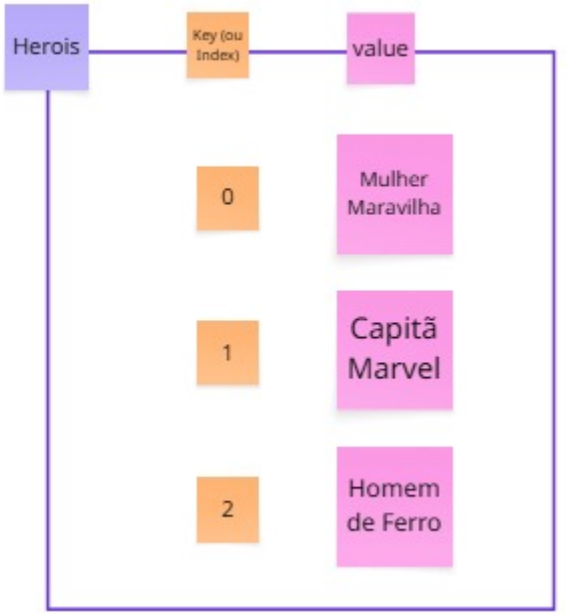

Requisição HTTP
Cliente (Frontend)
Servidor (Backend)
Database
Resposta HTTP
Navegadores (Chorme, etc)
Framework de Frontend (ReactJs, Angular, Vue, Android, etc)
Postman
Insomnia
Thunder Client
Entre outros...
NodeJs com Express ou Nest
PHP com Laravel
Jaca com Spring
Frameworks de frontend web (ReactJs, Angular, Vue) possuem o NodeJs somente durante o desenvolvimento
HTTP Cliente
Python com Flask ou Django
C# com .NET Core

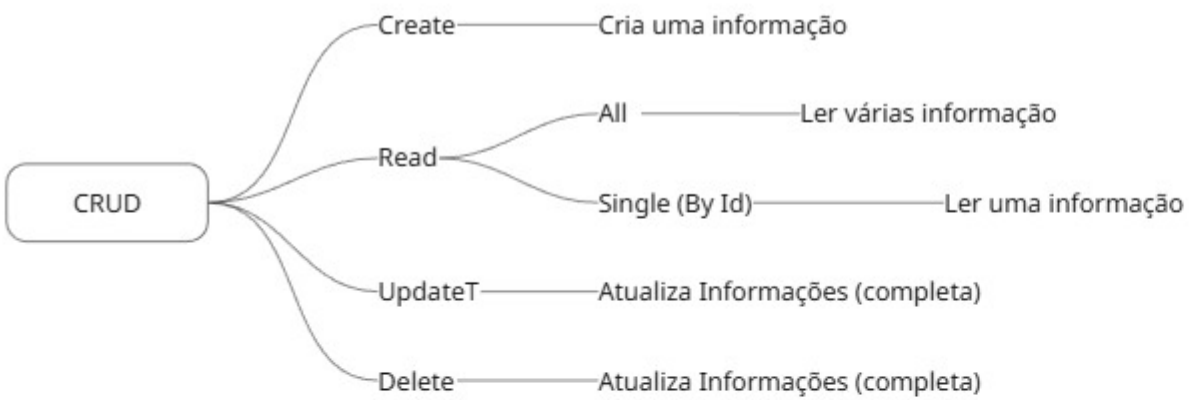

2025.11.04_InicioNode > JS index.js > ...
1 const express = require("express")
2 const app = express()
3 app.get('/', function (req, res) {
4 res.send('Olá, Mundo')
5 })
6
7 app.listen(3000)
8
app
Nosso Computador
Placa de Rede
Portas
Locajhost ou 127.0.0.1
Endereço
Porta
3000
Frameworks de frontend Web (reactJs, Angular, Vue) possuem o NodeJs somente deurante o desenvolvimento


http://localhost:3000/
express
Porta
/
"Hello, world!"
Método ou Verbo
Http://localhost:3000
Aplicação
rotas (aka: endpoints)
/oi
"Olá, mundo!"
HTTP
Endereço
http://localhost:3000/oi
hyper text transfer protocol
Direciona uma requisição (solicitação) para quem souber como respondê-la
Protocolos
Portas padrões
HTTP
80
FTP
21
Mongo
27017
SMTP
465

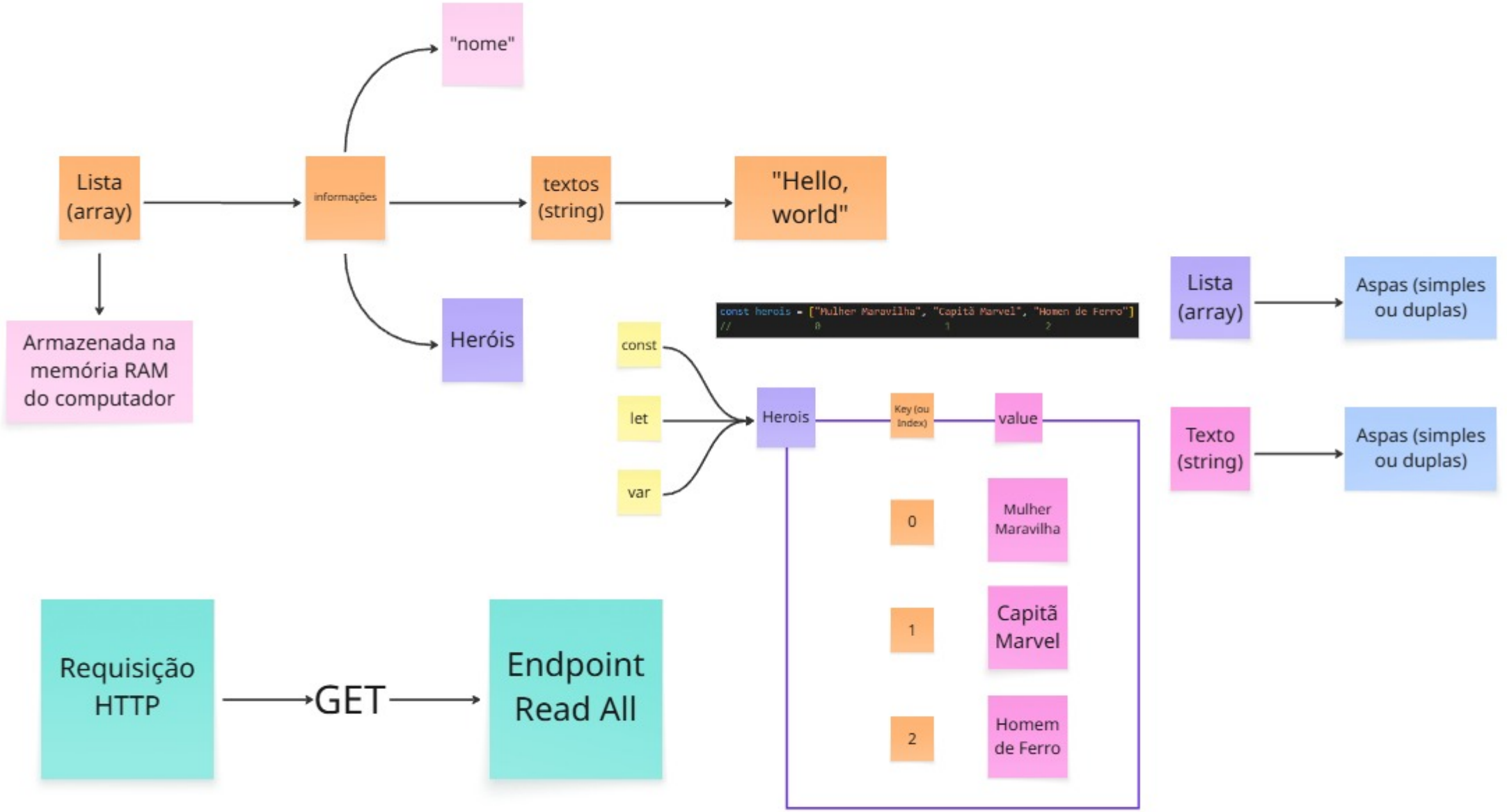

Métodos HTTP
PUT
Atualiza Informações (completa)
PUT
Atualiza Informações (completa)
PUT
Atualiza Informações (completa)
PUT
Atualiza Informações (completa)
PUT
Atualiza Informações (completa)

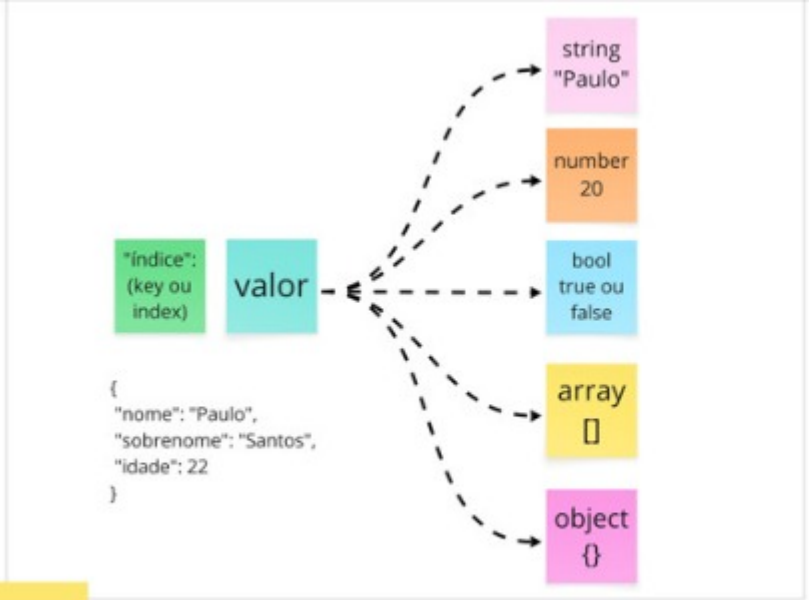
REST
RESTful

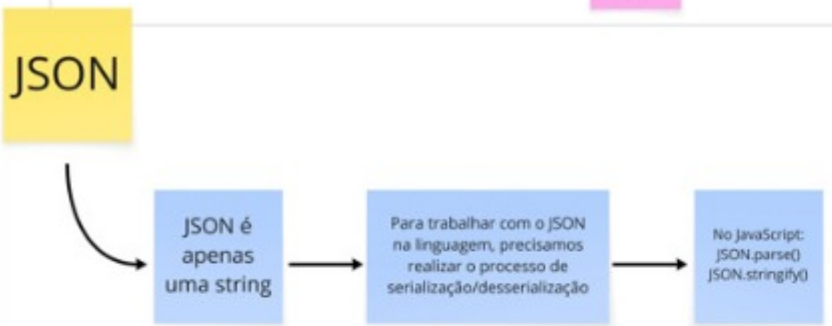

Herois
Key (ou Index)
value
0
Mulher Maravilha
1
Capitã Marvel
2
Homem de Ferro

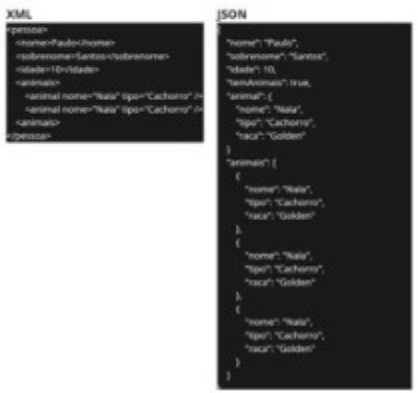

CRUD
Create
Cria uma informação
Read
All
Ler várias informação
Single (By Id)
Ler uma informação
UpdateT
Atualiza Informações (completa)
Delete
Atualiza Informações (completa)

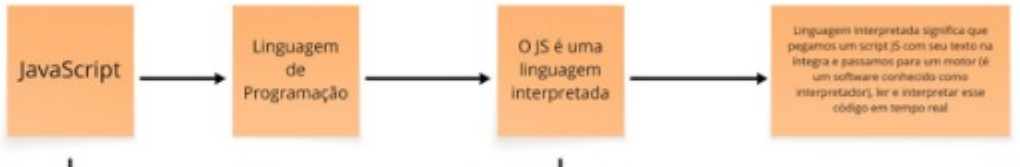
1ª etapa
Armazenamento na memória RAM
Reiniciou a aplicação, perdeu todos os dados
2ª etapa
Armazenamento no banco de dados
Pode reiniciar a aplicação, que tudo continua salvo


"nome"
Lista (array)
informações
textos (string)
"Hello, world"
Armazenada na memória RAM do computador
Heróis
const herois = ["Mulher Maravilha", "Capitã Marvel", "Homen de Ferro"]
// 0 1 2
const
let
var
Herois
Key (ou Index)
value
0
Mulher Maravilha
1
Capitã Marvel
2
Homem de Ferro
Lista (array)
Aspas (simples ou duplas)
Texto (string)
Aspas (simples ou duplas)
Requisição HTTP
GET
Endpoint Read All

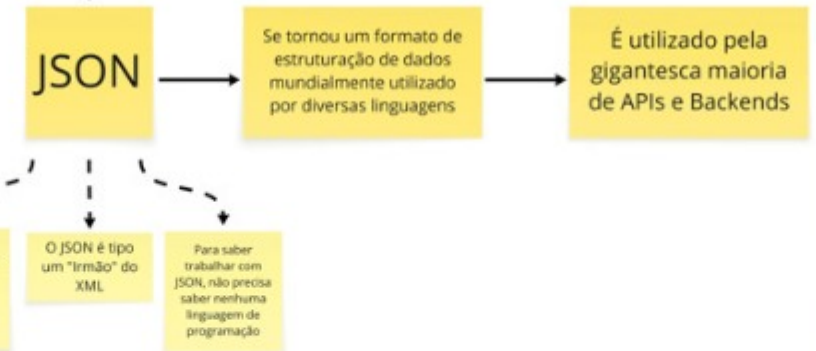

"índice": (key ou index)
valor
string "Paulo"
number 20
bool true ou false
array []
object {}
{
"nome": "Paulo",
"sobrenome": "Santos",
"idade": 22
}


JSON
JSON é apenas uma string
Para trabalhar com o JSON na linguagem, precisamos realizar o processo de serialização/desserialização
No JavaScript: JSON.parse() JSON.stringify()

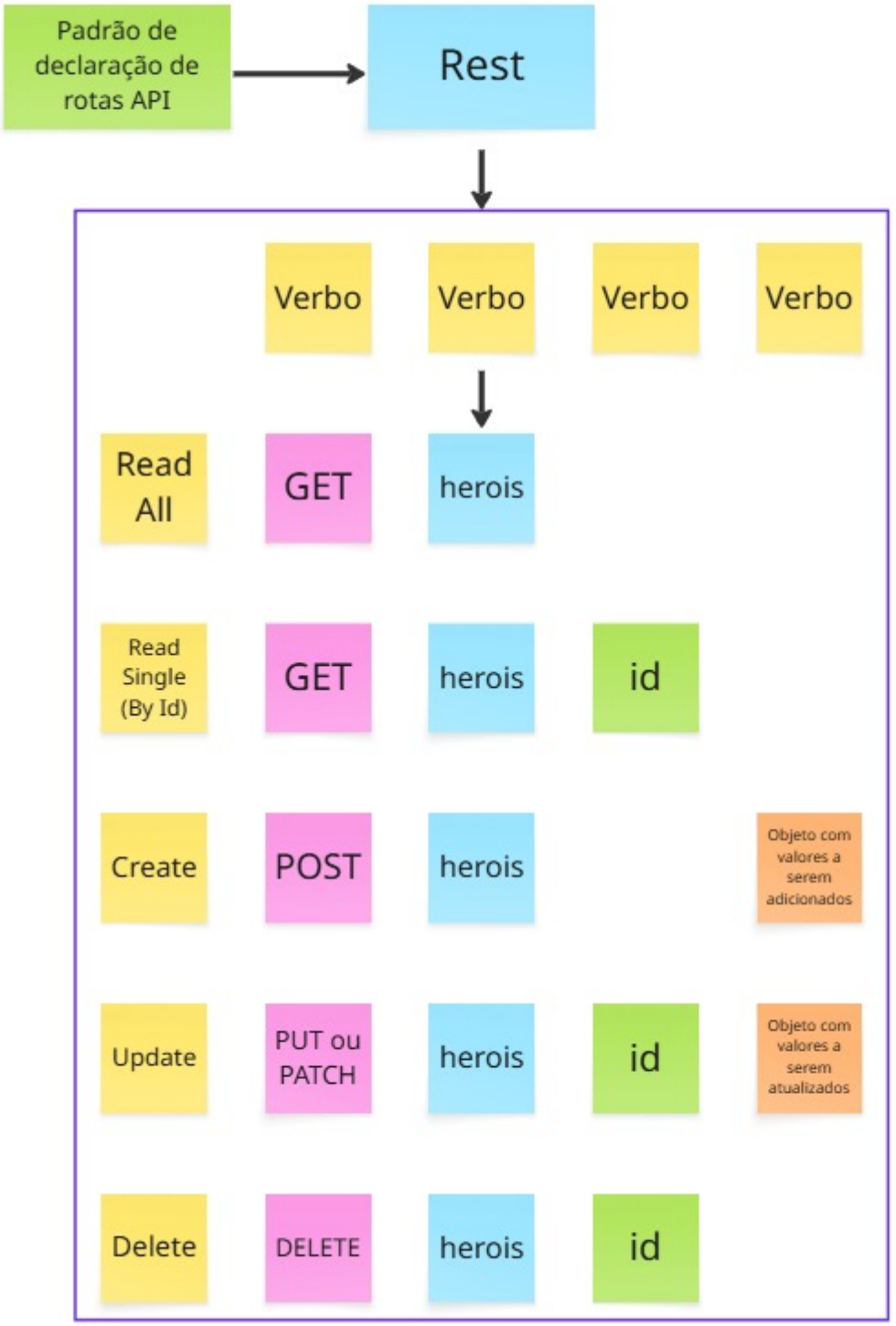

XML
JSON

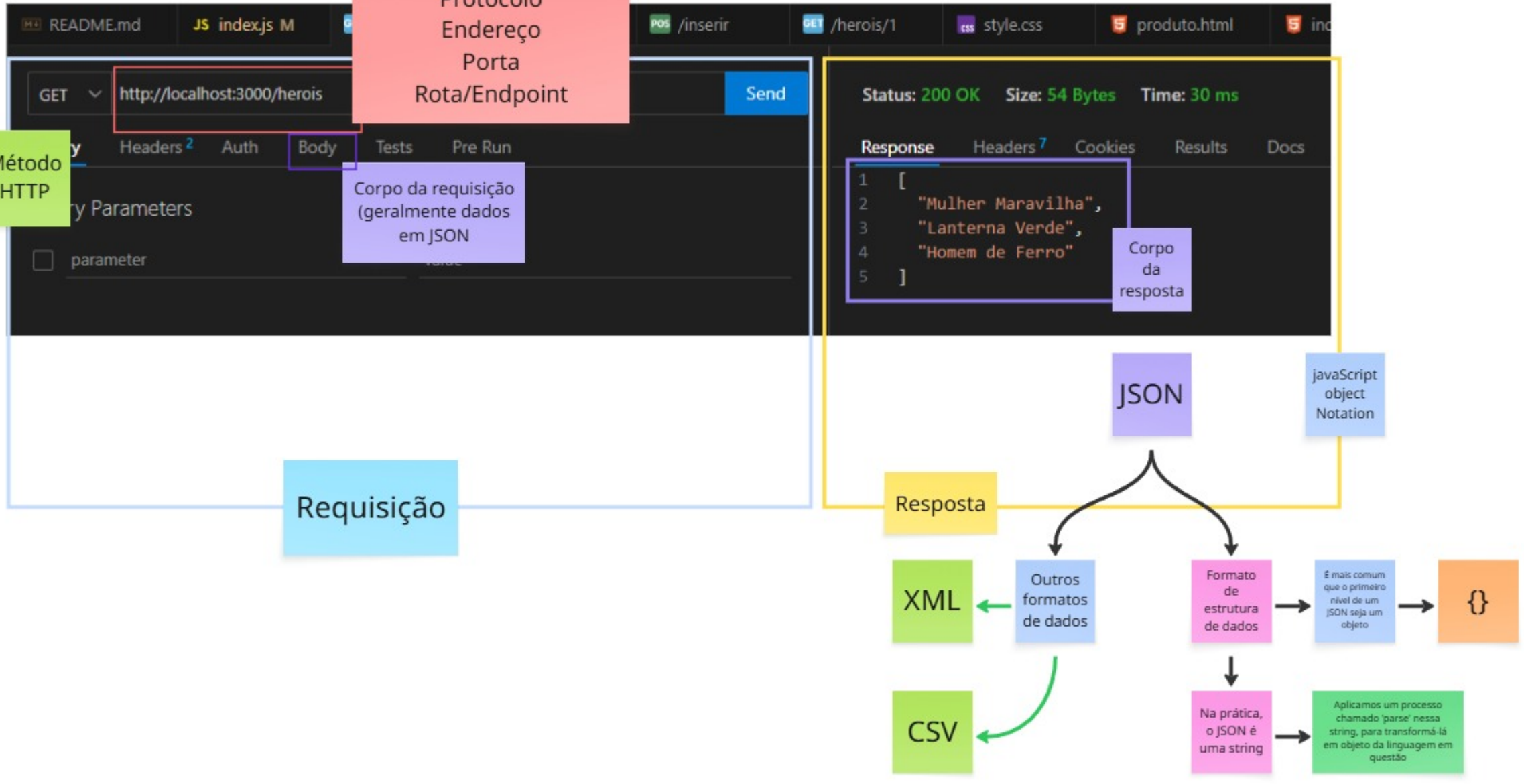

JavaScript
Linguagem de Programação
O JS é uma linguagem interpretada
Linguagem interpretada significa que pegamos um script JS com seu texto na íntegra e passamos para um motor (é um software conhecido como interpretador), ler e interpretar esse código em tempo real

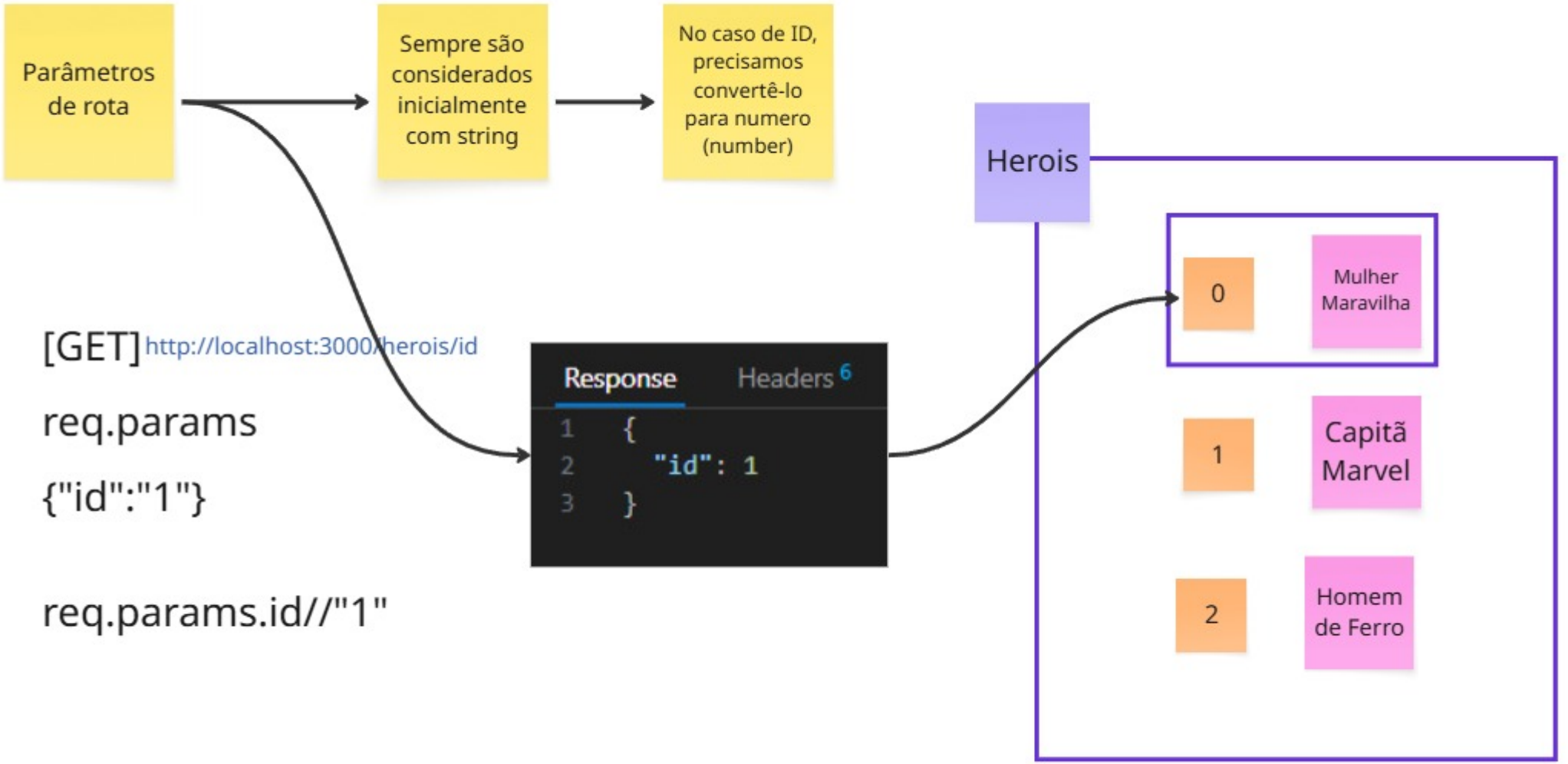
Além das linguagens interpretadas, temos também as linguagens compiladas
Uma linguagem compilada consiste em pegar um arquivo de texto contendo um código escrito em uma linguagem e passar para um compilador
Esse compilador é um software que tem a responsabilidade de transformar um código em um arquivo binário
Esse arquivo binário será executado direto pelo sistema operacional


JSON
Se tornou um formato de estruturação de dados mundialmente utilizado por diversas linguagens
É utilizado pela gigantesca maioria de APIs e Backends
JavaScript Object Notation
O JSON é tipo um "Irmão" do XML
Para saber trabalhar com JSON, não precisa saber nenhuma linguagem de programação

// 20220808091353
// https://pokeapi.co/api/v2/pokemon
{
"count": 1118,
"next": "https://pokeapi.co/api/v2/pokemon?offset=20&limit=20",
"previous": null,
"results": [
{
"name": "bulbasaur",
"url": "https://pokeapi.co/api/v2/pokemon/1/"
},
{
"name": "ivysaur",
"url": "https://pokeapi.co/api/v2/pokemon/2/"
},
{
"name": "venusaur",
"url": "https://pokeapi.co/api/v2/pokemon/3/"
},
{
"name": "charmander",
"url": "https://pokeapi.co/api/v2/pokemon/4/"
},
{
"name": "charmeleon",
"url": "https://pokeapi.co/api/v2/pokemon/5/"
},
{
"name": "charizard",
"url": "https://pokeapi.co/api/v2/pokemon/6/"


Padrão de declaração de rotas API
Rest
Verbo
Verbo
Verbo
Verbo
Read All
GET
herois
Read Single (By Id)
GET
herois
id
Create
POST
herois
Objeto com valores a serem adicionados
Update
PUT ou PATCH
herois
id
Objeto com valores a serem atualizados
Delete
DELETE
herois
id


README.md
index.js M
/inserir
/herois/1
style.css
produto.html
GET
http://localhost:3000/herois
Send
Protocolo
Endereço
Porta
Rota/Endpoint
Método HTTP
Headers 2
Auth
Body
Tests
Pre Run
Corpo da requisição (geralmente dados em JSON
parameter
Status: 200 OK
Size: 54 Bytes
Time: 30 ms
Response
Headers 7
Cookies
Results
Docs
1 [
2 "Mulher Maravilha",
3 "Lanterna Verde",
4 "Homem de Ferro"
5]
Corpo da resposta
Requisição
Resposta
JSON
javaScript object Notation
XML
Outros formatos de dados
CSV
Formato de estrutura de dados
É mais comum que o primeiro nível de um JSON seja um objeto
{}
Na prática, o JSON é uma string
Aplicamos um processo chamado 'parse' nessa string, para transformá-lá em objeto da linguagem em questão


Parâmetros de rota
Sempre são considerados inicialmente com string
No caso de ID, precisamos convertê-lo para numero (number)
[GET] http://localhost:3000/herois/id
req.params
{"id":"1"}
req.params.id//"1"
Response
Headers 6
1 {
2 "id": 1
3 }
Herois
0
Mulher Maravilha
1
Capitã Marvel
2
Homem de Ferro