API – GET & POST

de novo, Analu.

Sou desenvolvedora Full-Stack.
Fui da primeira turma de Back-End da Reprograma
Hoje trabalho como Engenheira de Software no Banco Itaú.

Email: sampaioaanaluiza@gmail.com

Objetivos

- [x] Node.js
- [x] Gerenciadores de pacotes
- [x] Dependências
- [x] Nosso primeiro servidor
- [x] Nossa primeira API
- [x] path params e query params
- [ ] GET e POST
- [ ] Para o lar

CRUD

CRUD é a composição da primeira letra de 4 operações básicas de um banco de dados, e são o que a maioria das aplicação faz

✅ C: Create (criar) - criar um novo registro
R: Read (ler) - exibir as informações de um registro
♻️ U: Update (atualizar) - atualizar os dados do registro
❌ D: Delete (apagar) - apagar um registro

Cada um deles corresponde a uma ação real no banco de dados.

GET	ler
POST	criar
PUT	substituir
PATCH	modificar
DELETE	excluir

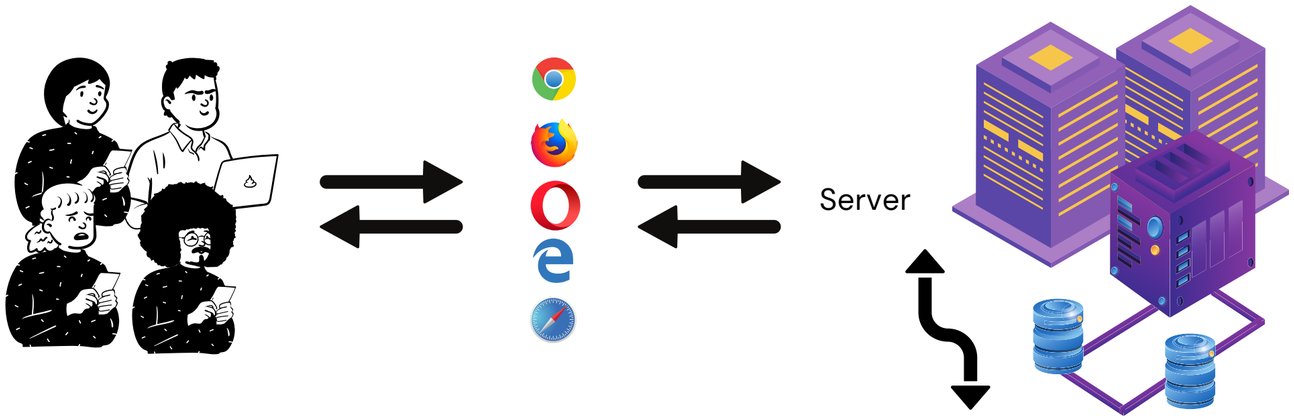
Modelo Server-Client

CLIENTE

Entenda cliente como a interface que o usuário interage. É o Cliente que **solicita** serviços e informações de um ou mais servidores.

SERVIDOR

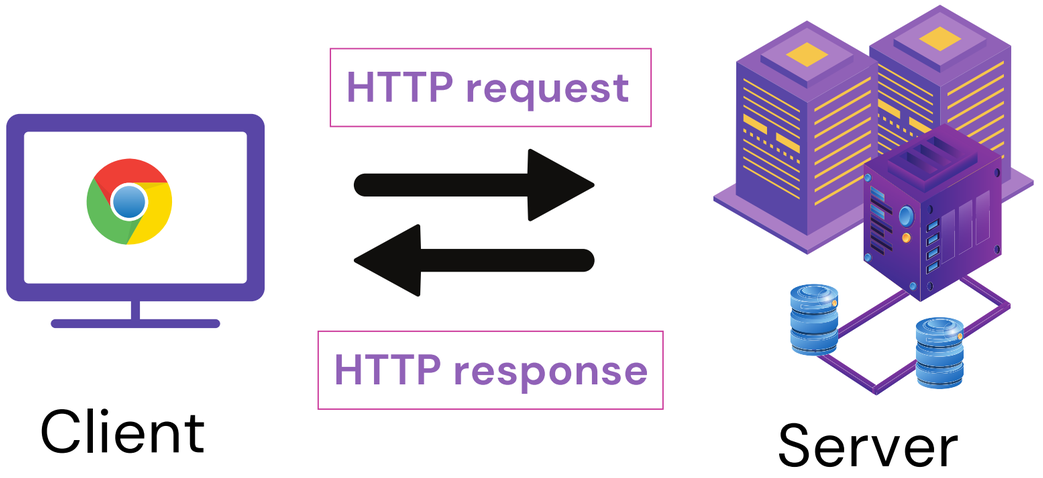
E o Servidor é o responsável pelo processo, organização e gerenciamento das informações. É ele que **responde** às solicitações feitas pelo usuário.
Ele é um processo reativo, disparado pela chegada de pedidos de seus clientes.



HTTP

Protocolo de Transferência de Hipertexto é um protocolo usado dentro do modelo Client/Server é baseado em pedidos (requests) e respostas (responses).
Ele é a forma em que o Cliente e o Servidor se comunicam.

Pensando em uniformizar a comunicação entre servidores e clientes foram criados **códigos** e **verbos** que são usados por ambas as partes, e essas requisições são feitas em **URLs** que possuem uma estrutura específica.



http://	www.minasprogramam.com	/courses
<protocolo>	<endereço do servidor>	<recurso>

Client

HTTP request

HTTP response

GET

POST

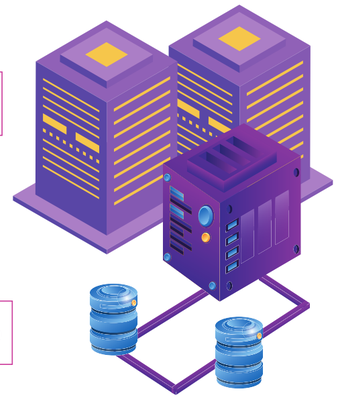
PUT

PATCH

DELETE

HTTP request

HTTP response



Server

verbo	ação
GET	ler
POST	criar
PUT	substituir
PATCH	modificar
DELETE	excluir

código	tipo de resposta
100-199	informação
200-299	sucesso
300-399	redirecionamento
400-499	erro do cliente
500-599	erro de servidor

API

Interface de Programação de Aplicativos

API busca criar formas e ferramentas de se usar uma funcionalidade ou um dado sem realmente ter que "reinventar a tal função."

Ela não necessariamente está num link na Web.

Web API e API REST

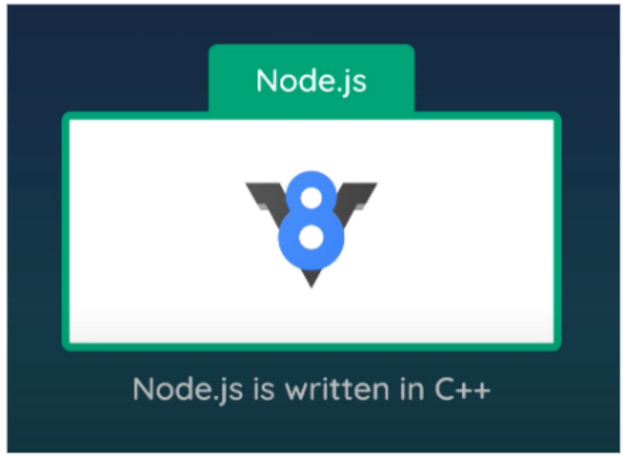
Web API é uma interface que é disponibilizada de forma remota, pela web, que possibilita a programação aplicativos e softwares.

E as APIs RESTfull são aquelas que são capazes de fazer o REST. Que nada mais é uma API que usa os protocolos HTTP para comunicação entre o usuário e o servidor.

Node.js

Node.js

Antes de tudo, um pouco de história!
Tudo começou em 2009...



Interpretador JavaScrypt que não precisa de navegador.
Ele pode:

- Ler e escrever arquivos no seu computador
- Conectar com um banco de dados
- Se comportar como um servidor

Objetivos

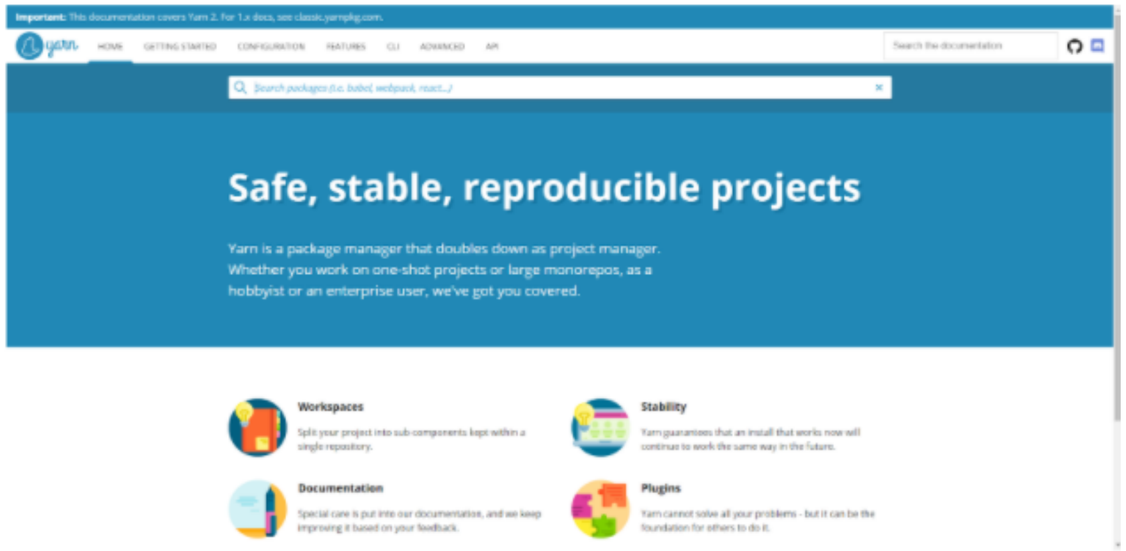
- [x] Node.js
- [x] Nossa primeira API
- [x] Gerenciadores de pacotes
- [x] path params e query params
- [x] Dependências
- [ ] GET e POST
- [x] Nosso primeiro servidor
- [ ] Para o lar

Servidores com Node.js

e a API?

Node.js

e os gerenciadores de pacotes



npm X yarn

... e a guerra dos 100 anos

Objetivos

- [x] Node.js
- [x] Gerenciadores de pacotes
- [x] Dependências
- [x] Nosso primeiro servidor
- [x] Nossa primeira API
- [x] path params e query params
- [ ] GET e POST
- [ ] Para o lar

@analu.io

{ Reprograma }

init

npm init

@analu.io

{ Reprograma }

Dependências

Só um pouquinho delas

Package.json

O arquivo package.json é o ponto de partida de qualquer projeto NodeJS. Ele é responsável por:

- descrever o seu projeto
- informar a versão do node e do npm
- url do repositório
- versão do projeto
- dependências de produção e de desenvolvimento

Express

npm install express

express

4.17.1 • Public • Published a year ago

Readme | Explore BETA | 30 Dependencies | 46.033 Dependents | 264 Versions

express

Fast, unopinionated, minimalist web framework for node.

npm v4.17.1 | downloads 58M/month | linux passing | windows passing | coverage 100%

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Install

```
> npm i express
```

Weekly Downloads

13.961.907

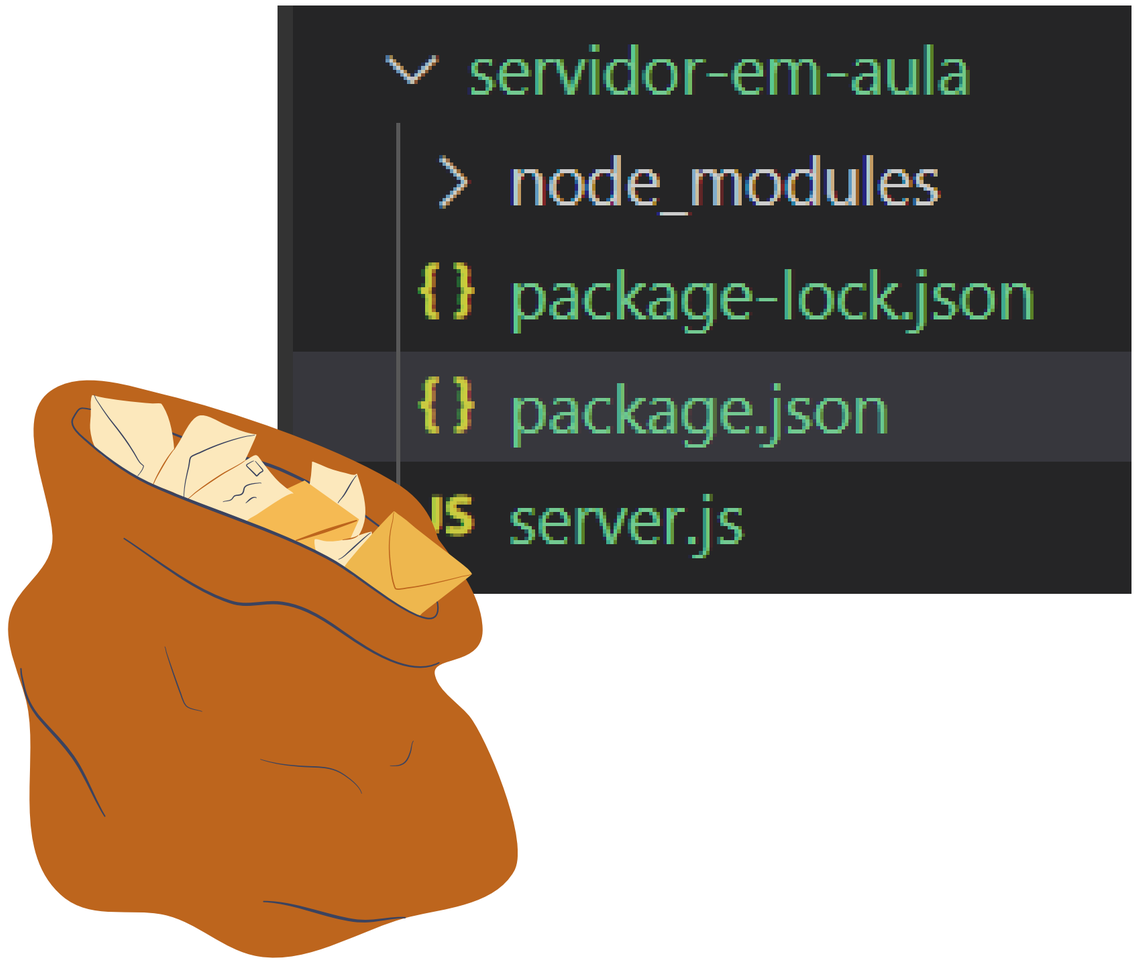
Version	License
4.17.1	MIT

Unpacked Size	Total Files
208 kB	16

Issues	Pull Requests
97	52

Homepage

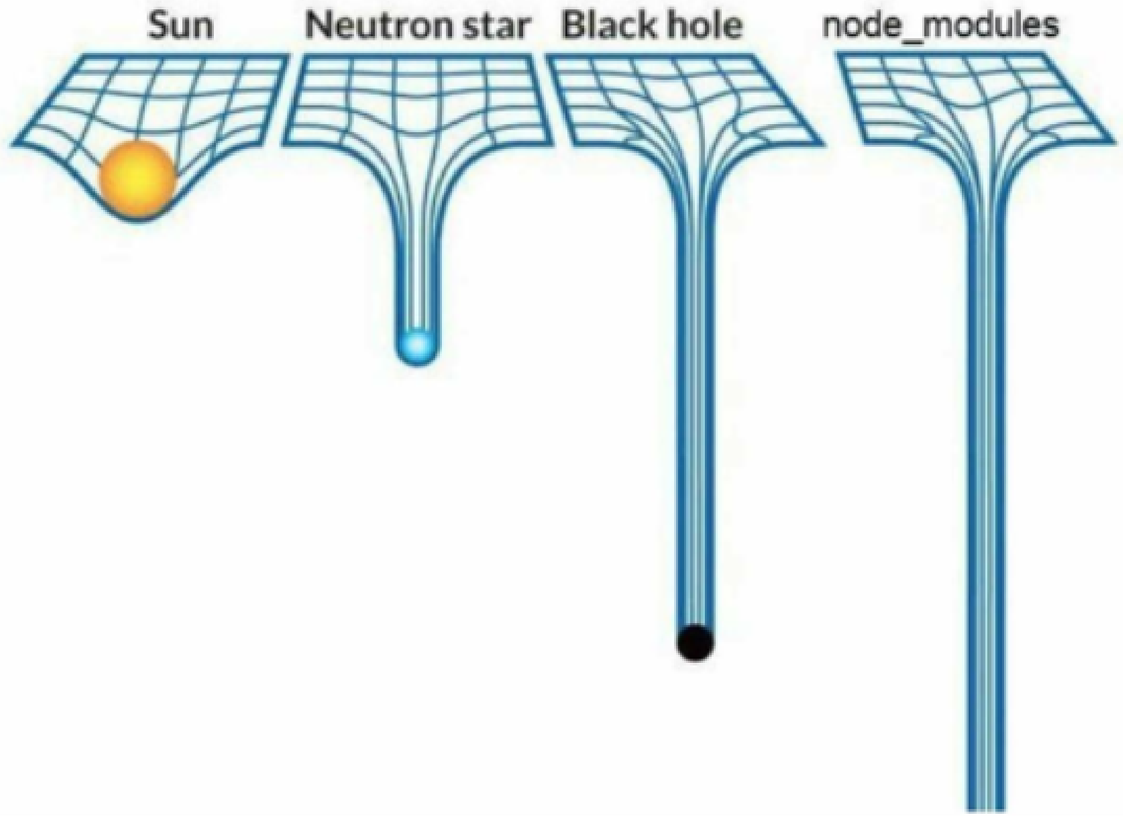
Pacotes



Sempre que você instalar um pacote do npm, ele será referenciado no **package-lock.json** e será instalado na pasta **node_modules**.

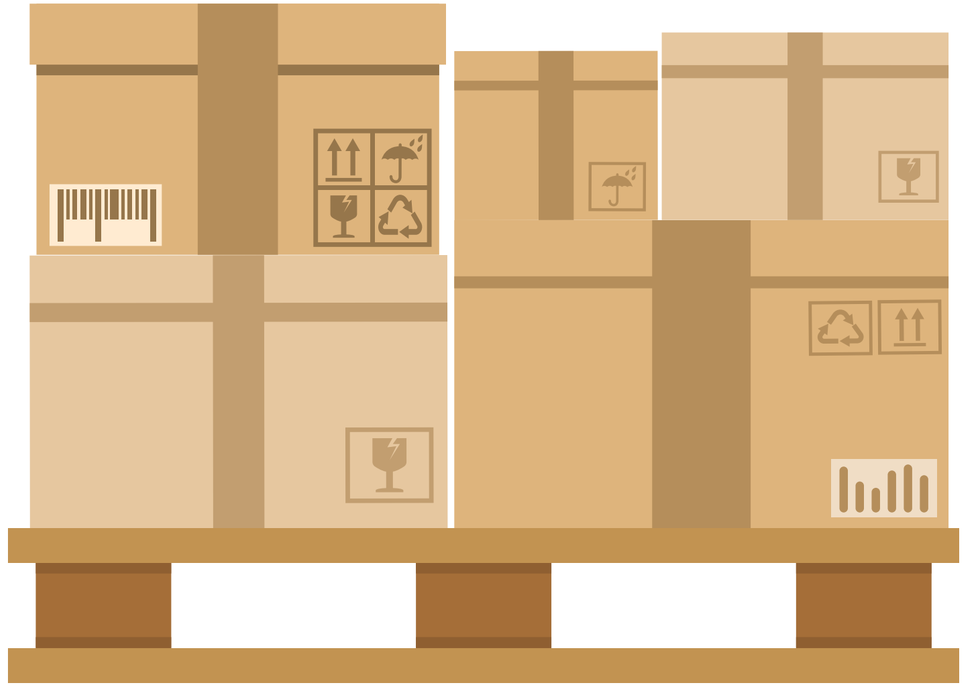
node_modules

Na node_modules estarão baixadas as dependências que o seus pacotes precisarão pra funcionar



Package-lock.json

O package-lock especifica a versão e suas dependências próprias, assim, a instalação criada será sempre a mesma, toda vez.



.gitignore

vamo ignorar ela, gente

Nós ignoramos a node_modules pois nela estão todos os downloads de todas as dependências do projeto, fica MUITO pesado subir ela no git, por exemplo

Se apagarmos ela só precisamos dar o comando *npm install* que as dependências serão baixadas de novo e pasta node_modules reaparecerá.

O npm sabe quais dependências baixar pois elas estão referenciadas no package.json e no package-lock.json.

Objetivos

- [x] Node.js
- [x] Gerenciadores de pacotes
- [x] Dependências
- [x] Nosso primeiro servidor
- [x] Nossa primeira API
- [x] path params e query params
- [ ] GET e POST
- [ ] Para o lar

Nosso primeiro servidor

O mais fácil do mundo

API Rest

só o básico

- porta
- rota
- http response
- response

http://localhost:PORTA/recurso

só o básico

nodemon

pra parar de parar

nodemon

2.0.4 • Public • Published 4 months ago

Readme | Explore BETA | 10 Dependencies | 2.477 Dependents | 215 Versions

Install

```
> npm i nodemon
```

♥ Fund this package

Weekly Downloads

2.893.116

Version	License
2.0.4	MIT

Unpacked Size	Total Files
107 kB	43

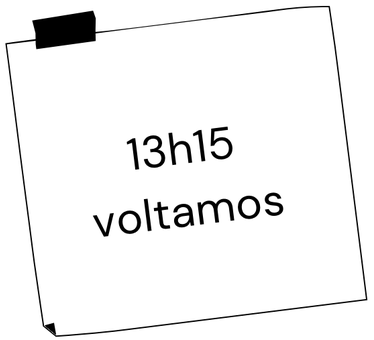
nodemon

nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

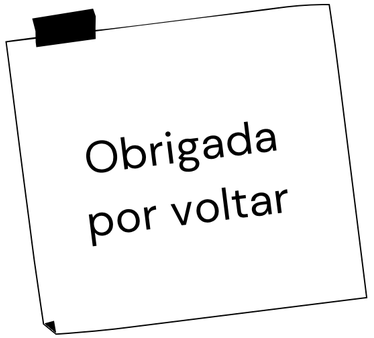
Objetivos

- [x] Node.js
- [x] Nossa primeira API
- [x] Gerenciadores de pacotes
- [x] path params e query params
- [x] Dependências
- [ ] GET e POST
- [x] Nosso primeiro servidor
- [ ] Para o lar

Ai, que fome...



Nossa, você aqui de novo?



duvidas?

Node.js

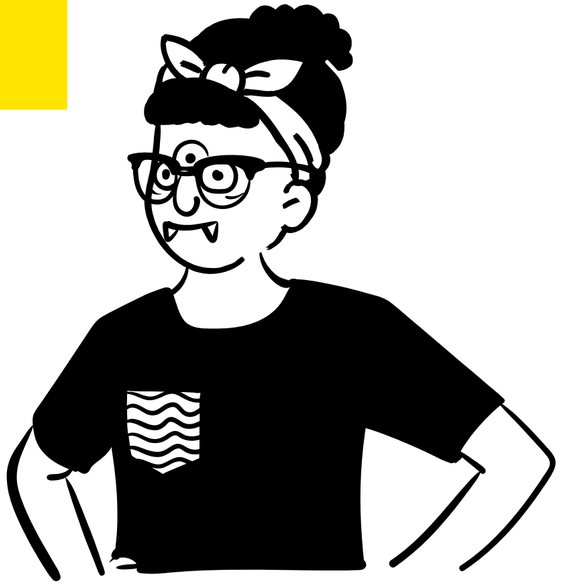
Gerenciadores de pacotes

Dependências

Nosso primeiro servidor

API em aula

agora vai!



API em aula

o negócio

- quero poder consultar uma lista dos filmes da ghibli por titulo, id e também a lista completa.
- se o usuário digitar errado o nome do filme quero retorno do erro
- quero poder cadastrar um novo filme

iniciando o projeto

npm init

instalando as dependências

npm install NOME-DA-DEPENDÊNCIA

estartando o servidor

npm start

instalando dependencias de um servidor já iniciado

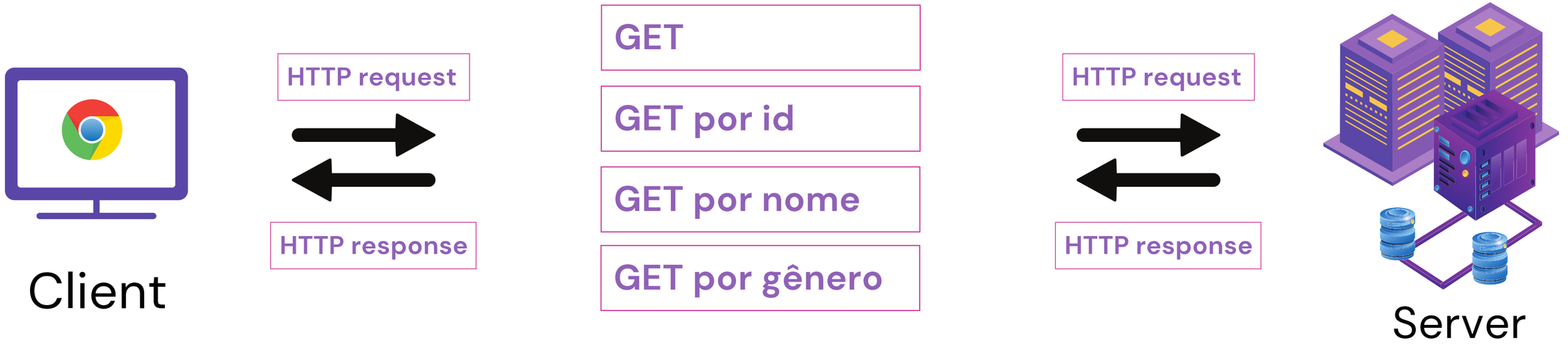
npm install

GET

Dentro do CRUD o método GET é representado pela letra R

R: Read (ler) - exibir as informações de um registro

O Client manda um **request** solicitando realizar um GET e o Server deve estar preparado para receber esse GET e responde-lo com um **response**.



Parametros

Path params

- são aqueles que são adicionados diretamente na URL
- "/rota/:id"
- request.params.id
- bons, porem limitantes, por exemplo, se quisermos filtrar por uma string

Query params

- são aqueles que são adicionados a chave e o valor desejados
- "/rota?id=1234"
- request.query.id
- a forma mais efetiva de fazer requests com strings ou diversos valores

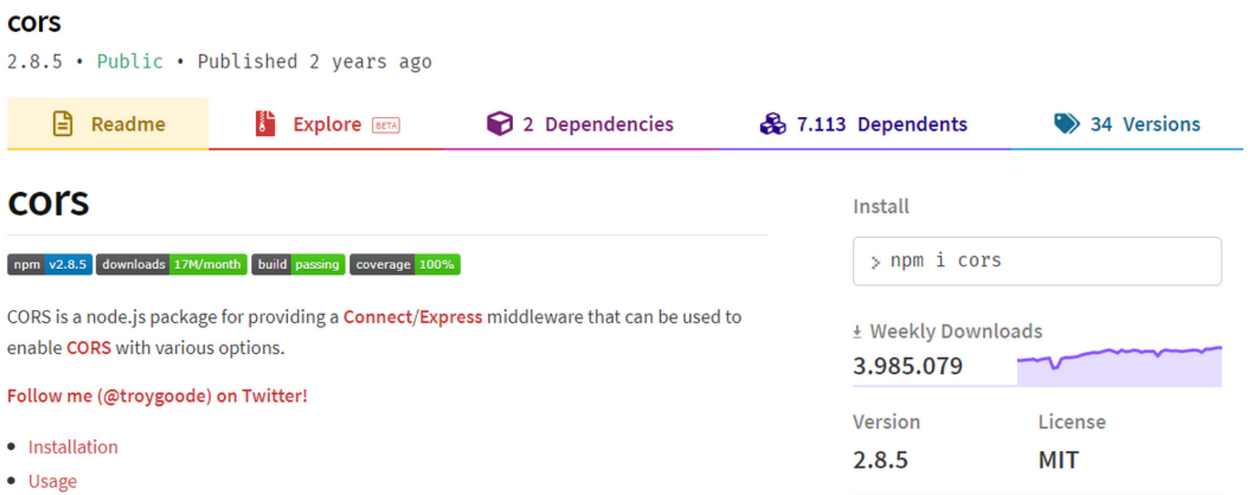
Post

✅ C: Create (criar) - criar um novo registro

O Client manda um **request** solicitando realizar um POST e envia no **Body** o registro que quer cadastrar, o Server deve estar preparado para receber esse POST, incluir o novo item solicitado e responde-lo com um **response**.

cors

npm install cors



CORS (Cross-Origin Resource Sharing) é uma especificação permite que um site acesse recursos de outro site mesmo estando em domínios diferentes.

Os navegadores fazem uso de uma funcionalidade de segurança chamada **Same-Origin Policy**: um recurso de um site só pode ser chamado por outro site se os 2 sites estiverem sob o mesmo domínio.

Isso porque o navegador considera recursos do mesmo domínio somente aqueles que usam o **mesmo protocolo** (http ou https), a mesma **porta** e o **mesmo endereço**

Pensando assim, o front-end e o back-end deveriam estar no mesmo Servidor e na mesma camada, o que não acontece!

Para resolver esse problema usamos o CORS

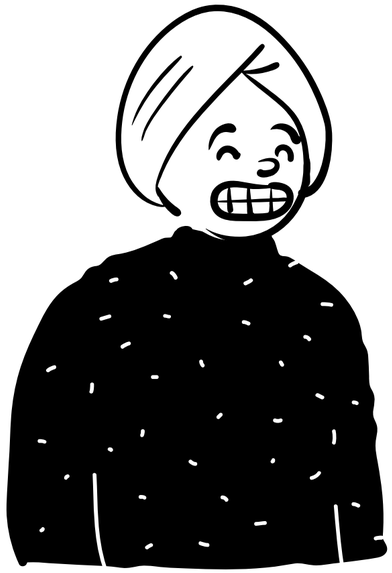
Body

- são usados nos métodos POST, PATCH e PUT
- enviam dados a serem cadastrados no banco de dados
- request.body

```
{
    "descricao": "exemplo de Body",
    "nomeColaborador": "Ana"
}
```

Body Parse

app.use(express.json())



Quando recebemos um request os dados do body são enviados de uma forma que não conseguimos facilmente acessar e manipular.

Por isso, devemos "parsear" o body: essa função analisa a transforma num json manipulavel

Parâmetros

Tanto o body quando o query e o path são parâmetros enviados na requisição e podem ser acessados pelo servidor afim de definir a requisição e as ações.

request.params

usado para pesquisa simples, enviado diretamente na rota

request.query

usado para pesquisa de uma ou multiplas strings

request.body

usado para enviar dados que serão cadastrados no banco, podem ser combinados com o query ou o path params

Objetivos

- [x] Node.js
- [x] Gerenciadores de pacotes
- [x] Dependências
- [x] Nosso primeiro servidor
- [x] Nossa primeira API
- [x] path params e query params
- [x] GET e POST
- [ ] Para o lar

Para o lar

UHUUUL