

# Projeto CRUD



# Roi, turma 9 né?

Eu sou desenvolvedora back-end.  
Hoje sou Engenheira de Software trabalhando  
principalmente com Java no Banco Itaú.

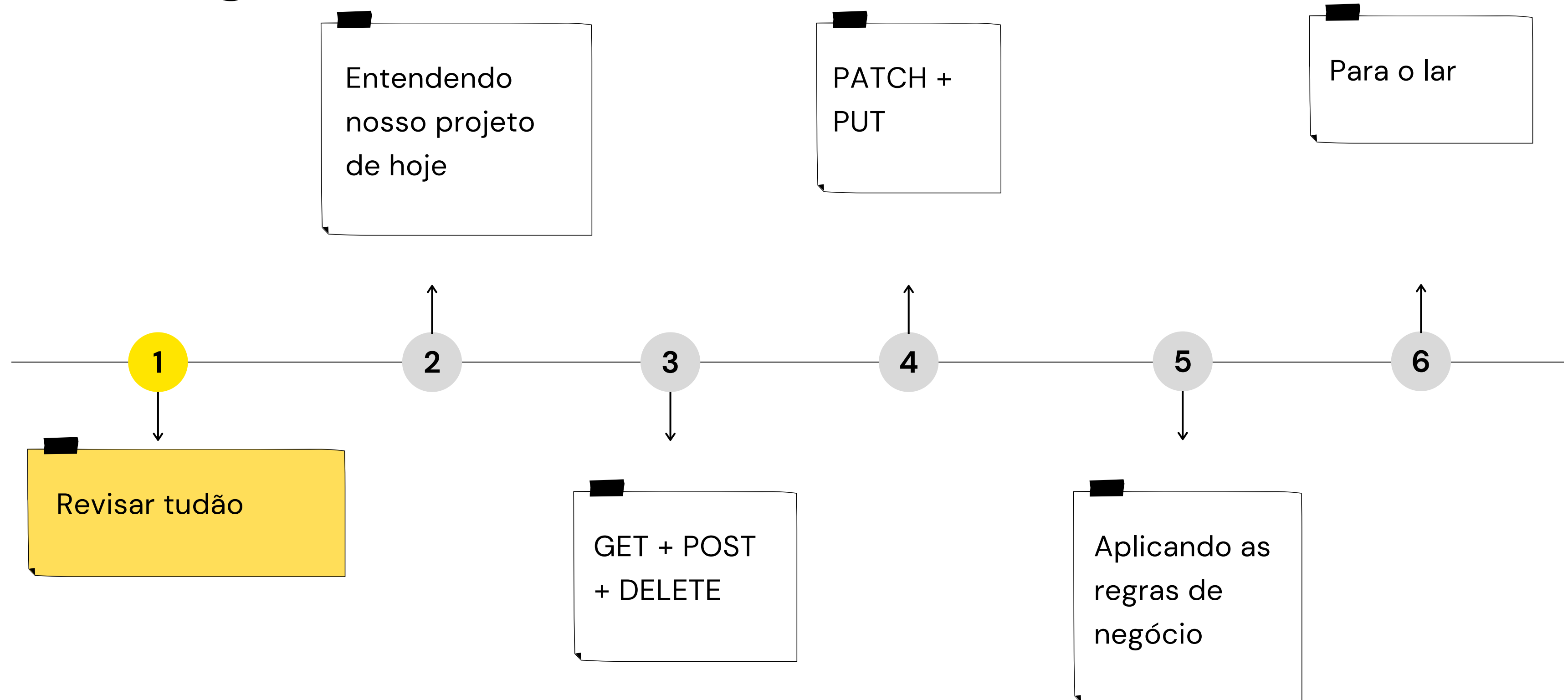
Pra me achar você pode procurar por  
**@analu.io** no instagram ou por  
**@sampaioaanaluiza** no linkedin e medium



Analu



# Cronograma



# Modelo Server-Client

## CLIENTE

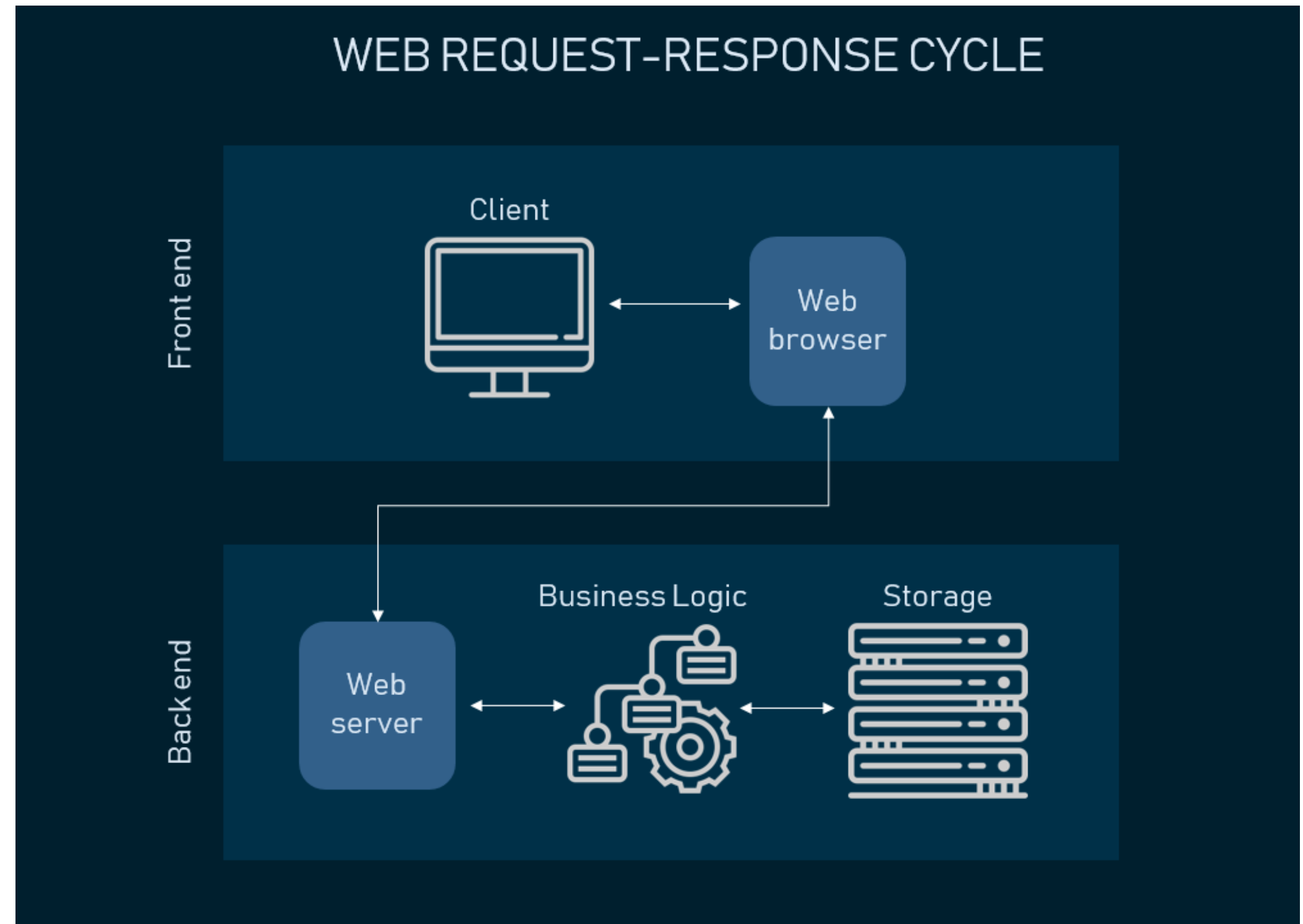
Entenda cliente como a interface que o usuário interage. É o Cliente que **solicita** serviços e informações de um ou mais servidores.

## 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.

Revisão



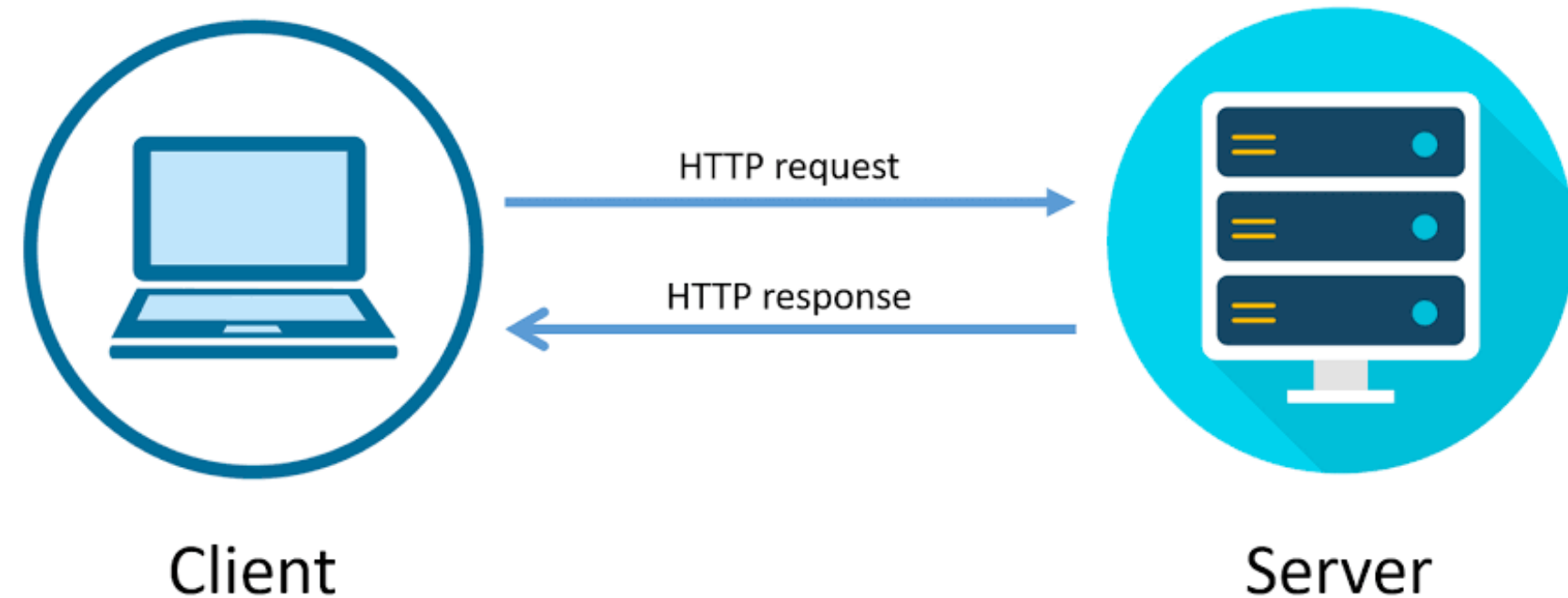
# HTTP

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.

<protocolo>://<servidor>:<porta>/<recurso>



Revisão



# HTTP – Status Code

Quando o Client faz uma requisição o Server responde com um código de status numérico também padronizado.

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi concluída. As respostas são agrupadas em cinco classes:

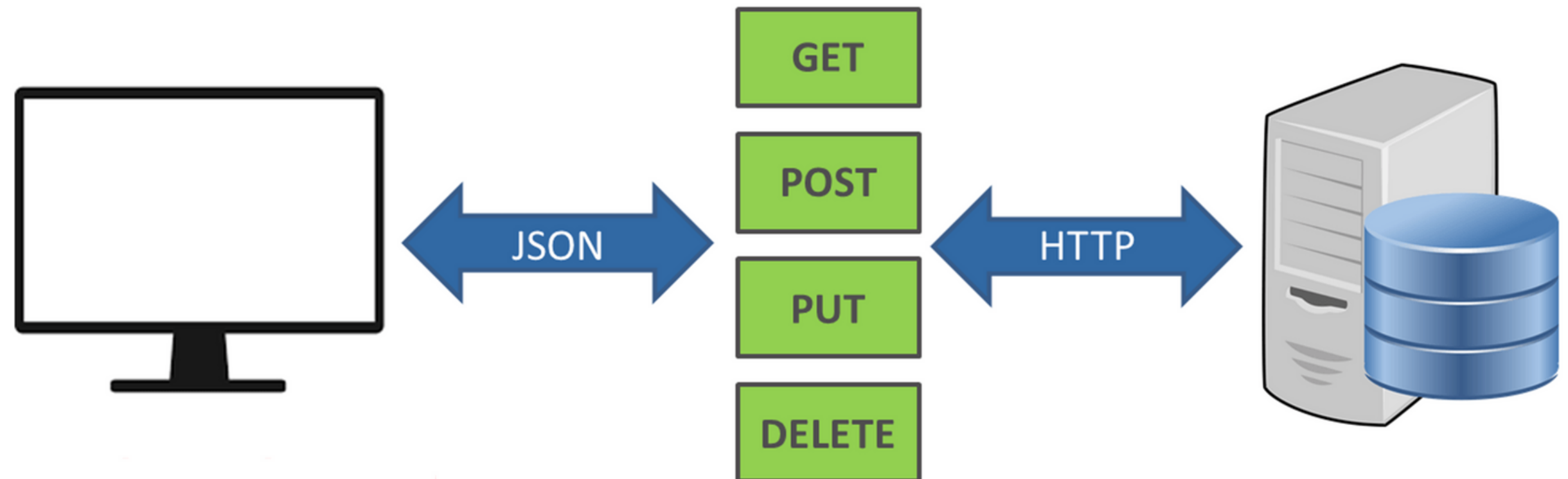
É a desenvolvedora Back end que coloca na construção do servidor quais serão as situações referentes a cada resposta.

Respostas de informação (100–199)  
Respostas de sucesso (200–299)  
Redirecionamentos (300–399)  
Erros do cliente (400–499)  
Erros do servidor (500–599)

# HTTP – Verbos

Os verbos HTTP são um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada.

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





# HTTP – 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

Revisão

## Operações CRUD com HTTP

Verbos HTTP	Operação CRUD
GET	Ler
POST	Criar
PUT	Substituir
PATCH	Modificar
DELETE	Excluir



# API

Interface de Programação de Aplicativos

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

Ela não necessariamente está num link na Web, ela pode ser uma lib ou um framework, uma função já pronta em uma linguagem específica, etc.

# 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.

# Dependências

só um pouquinho delas




# Express

npm i 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

4.17.1

License

MIT

Unpacked Size

208 kB

Total Files

16

Issues

97

Pull Requests

52

[Homepage](#)


Revisão

# nodemon

npm i nodemon

## nodemon

2.0.4 • Public • Published 4 months ago

 [Readme](#)

 [Explore](#) BETA

 10 Dependencies

 2.477 Dependents

 215 Versions



## 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.

### Install

```
> npm i nodemon
```

♥ [Fund this package](#)

### Weekly Downloads

2.893.116



Version

2.0.4

License

MIT

Unpacked Size

107 kB

Total Files

43

Revisão

# cors

npm i 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

Revisão ++

## cors

2.8.5 • Public • Published 2 years ago



Readme



Explore BETA



2 Dependencies



7.113 Dependents



34 Versions

## cors

npm v2.8.5 downloads 17M/month build passing coverage 100%

CORS is a node.js package for providing a **Connect/Express** middleware that can be used to enable **CORS** with various options.

Follow me (@troygoode) on Twitter!

- Installation
- Usage

### Install

```
> npm i cors
```

± Weekly Downloads

3.985.079

Version

2.8.5

License

MIT

# Arquitetura – MVC

MVC é um padrão de arquitetura de software, separando sua aplicação em 3 camadas. A camada de interação do usuário(view), a camada de manipulação dos **dados(model)** e a camada de **controle(controller)**

Já que estamos lidando com um projeto que tem somente back-end, não lidaremos com as views, porém lidaremos com as **rotas (routes)**.

O MVC nada mais é que uma forma de **organizar** o nosso código.

```

\--□ NOME-DO-SEU-SERVIDOR
|  server.js
|
\--□ src
|  app.js
|
□---controllers
|  NOMEController.js
|
□---models
|  NOME.json
|
□---routes
|  NOMERoute.js

```

## MODEL

Armazena os dados

## CONTROLLER

# Logica e regra de negócio

## ROUTES

## Definição das rotas e verbos

## APP

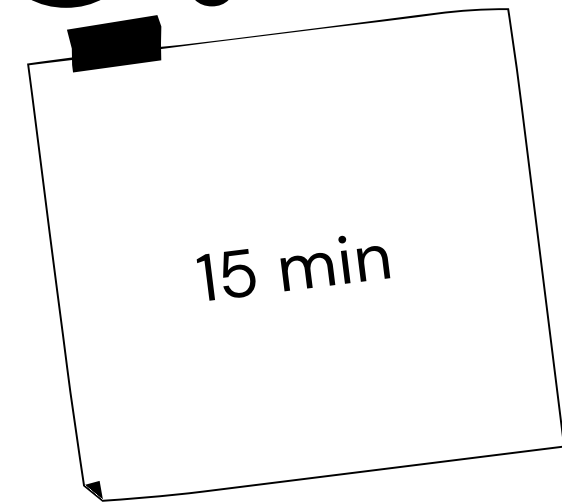
Indicação da  
rota raiz e  
configuração  
de requisição

## SERVER

configuração  
da porta,  
sobre o  
servidor

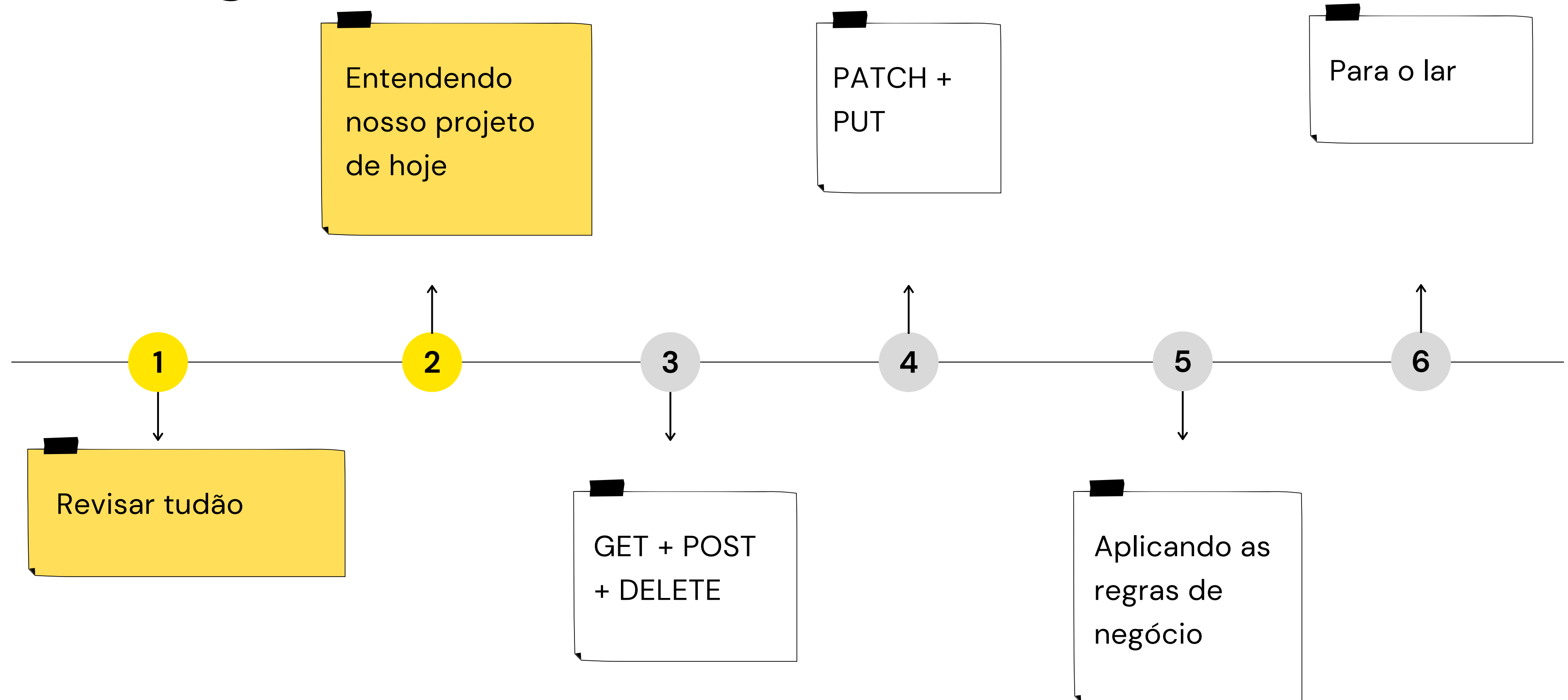
## Revisão

# Ai... vamo para um pouquinho?





# Cronograma



# Projeto de hoje



UHUUUL