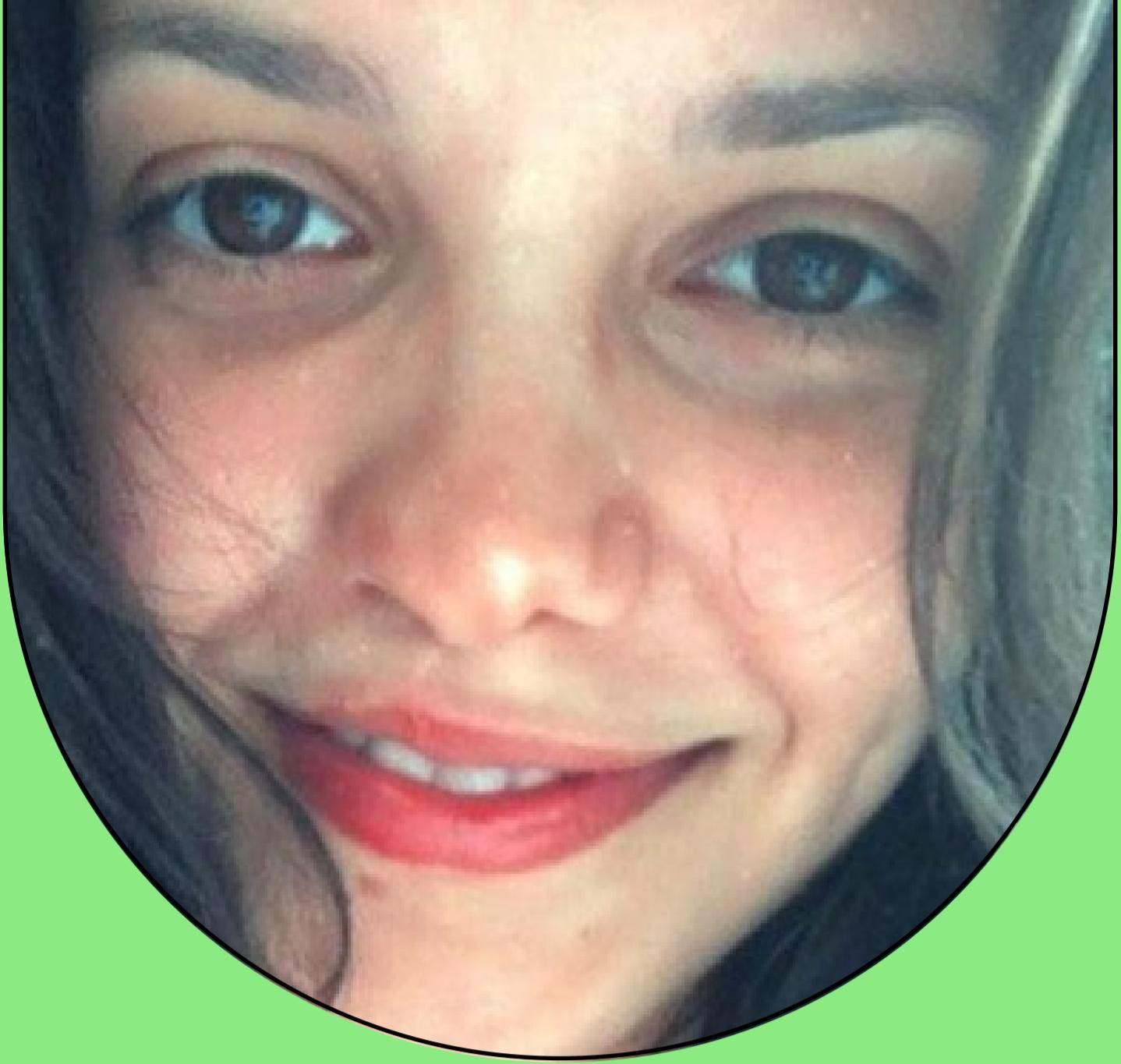




Criando uma **REST API**

Node.js



Marianne Salomão

Cloud Engineer na IBM Brasil,
desenvolvedora de software full - stack e instrutora de TI.

 /mariannesalomao

 /mariannesalomao

Índice

1. Anatomia de uma REST API

2. Entendendo uma Requisição

3. Os métodos HTTP

4. HTTP

5. Autenticação X Autorização

6. API Gateway

7. Versionamento APIs

8. Documentação

9. O que é DX?

10. Tutorial Swagger

11. Prática de Express

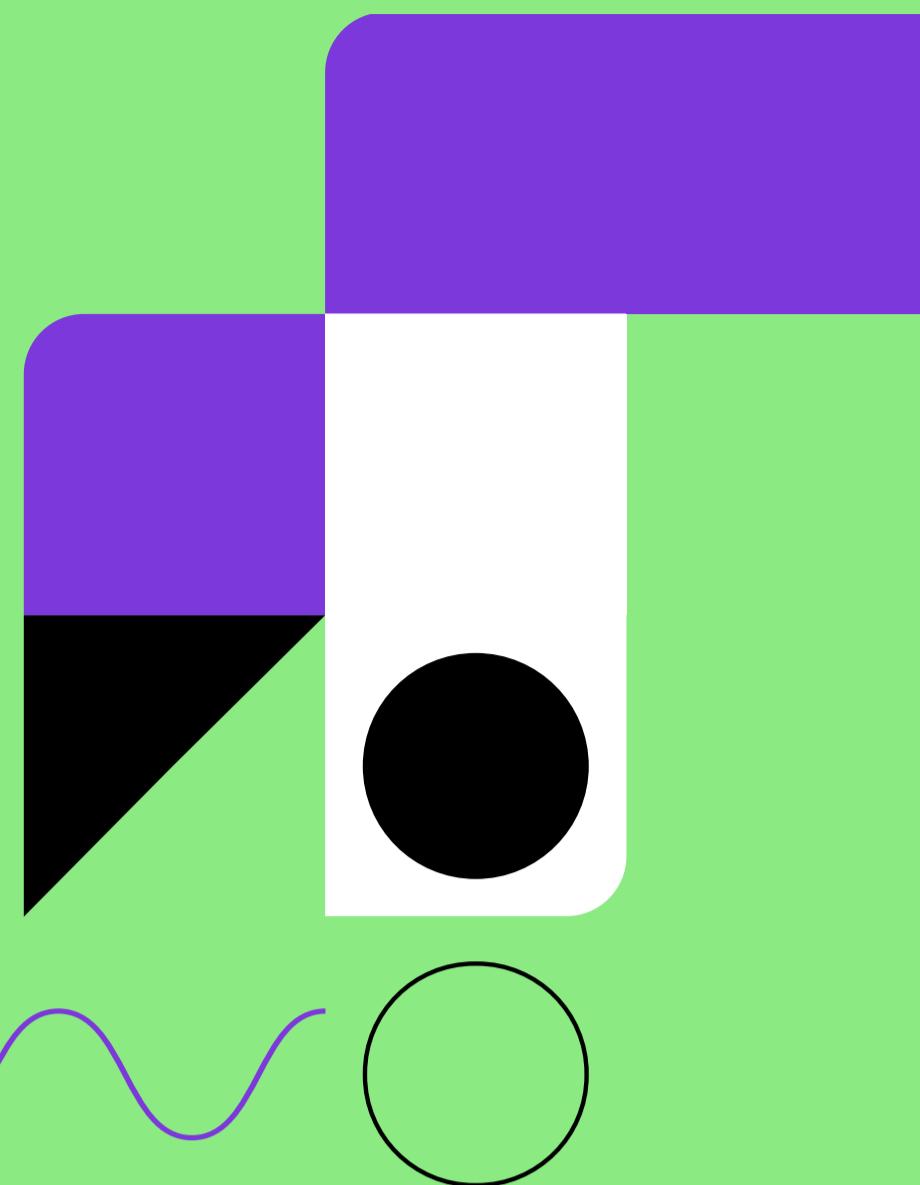
12. Express

13. Segurança

Introdução

Quando falamos de APIs toda a comunicação dessa interface é feita via web, ou seja, tudo é feito através de uma requisição a uma URL, que por sua vez, traz uma resposta.



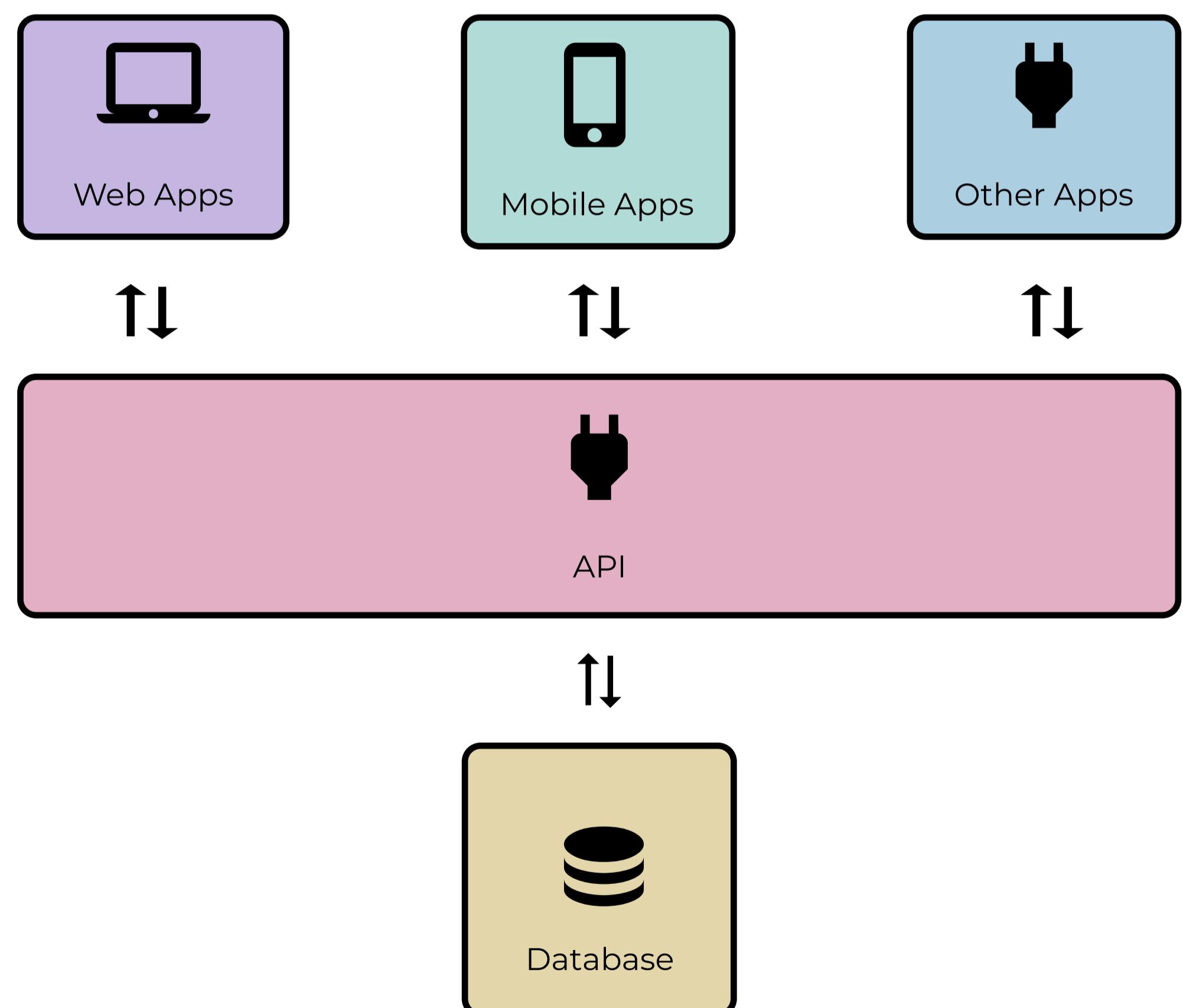


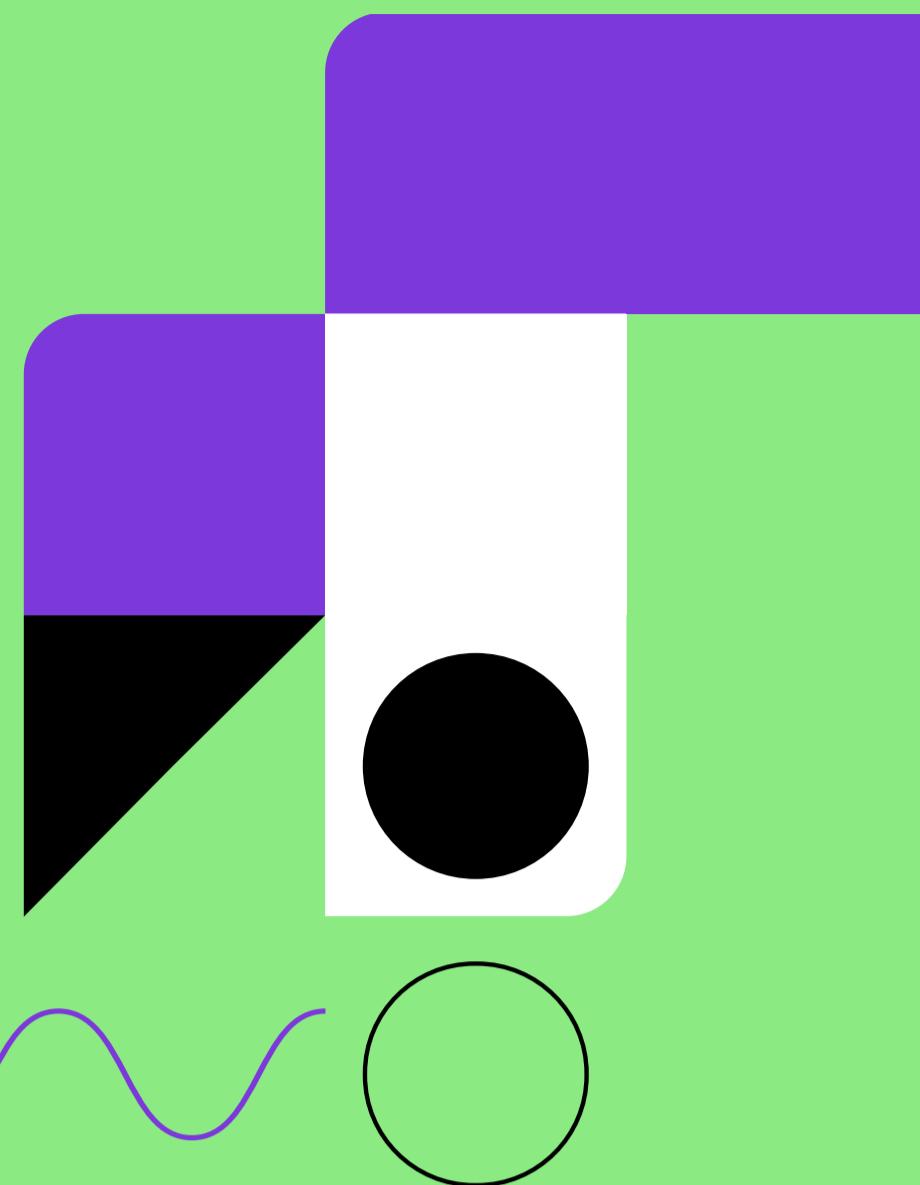
Anatomia REST

API quer dizer "**Application Programming Interface**", em outras palavras, é uma interface de comunicação entre desenvolvedores.

Em relação ao **REST (Representational State Transfer)**, ele é um conjunto de regras e boas práticas para o desenvolvimento dessas APIs.

Anatomia





Entendendo uma Requisição

A URL nada mais é que o caminho para fazer a requisição, porém é interessante ressaltar que ela segue a seguinte estrutura:

Base URL

- Esse é o início da URL da requisição, aqui você basicamente falará a informação de domínio que se repete em qualquer requisição. Por exemplo:

Exemplo de url:

<https://api.minhaapi.com>

Resource ou Path

- O recurso é o tipo de informação que você está buscando, ou seja, vamos simular que estamos buscando saber sobre vinhos, então acrescentamos o recurso vinhos:

Exemplo de resource/path:

<https://api.minhaapi.com/endereco>



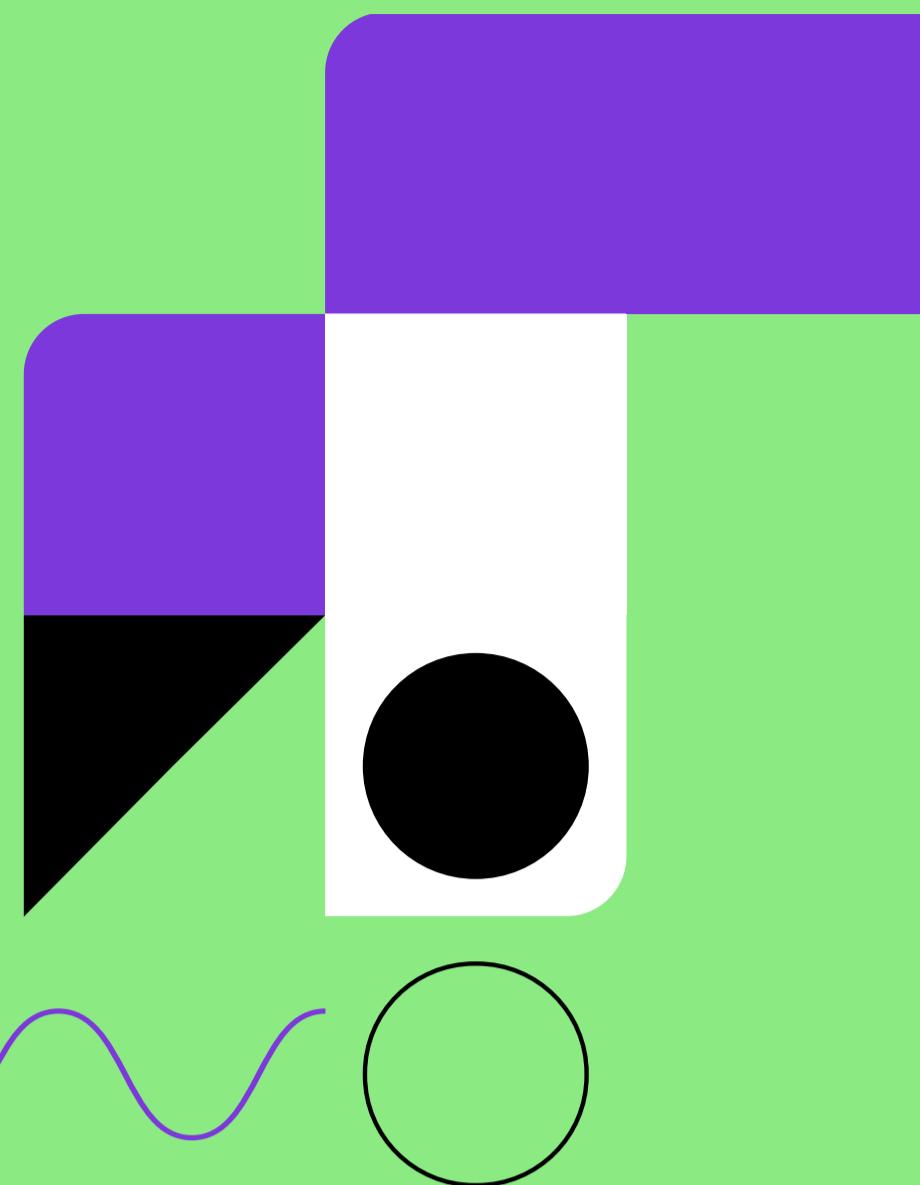
A URL nada mais é que o caminho para fazer a requisição, porém é interessante ressaltar que ela segue a seguinte estrutura:

Query String

- A query string são os parâmetros daquela requisição, então, se quisesse saber os melhores vinhos da região sul do Brasil, você incluiria esses parâmetros **?pais=brasil®iao=sul** e nossa URL ficaria assim:

Exemplo de query string:

<https://api.minhaapi.com/enderecopais=brasil®iao=sul>



Método HTTP

O método te ajuda a informar o tipo de ação que você está fazendo naquela requisição.

Dentre os principais métodos, temos:

GET

Busca dados

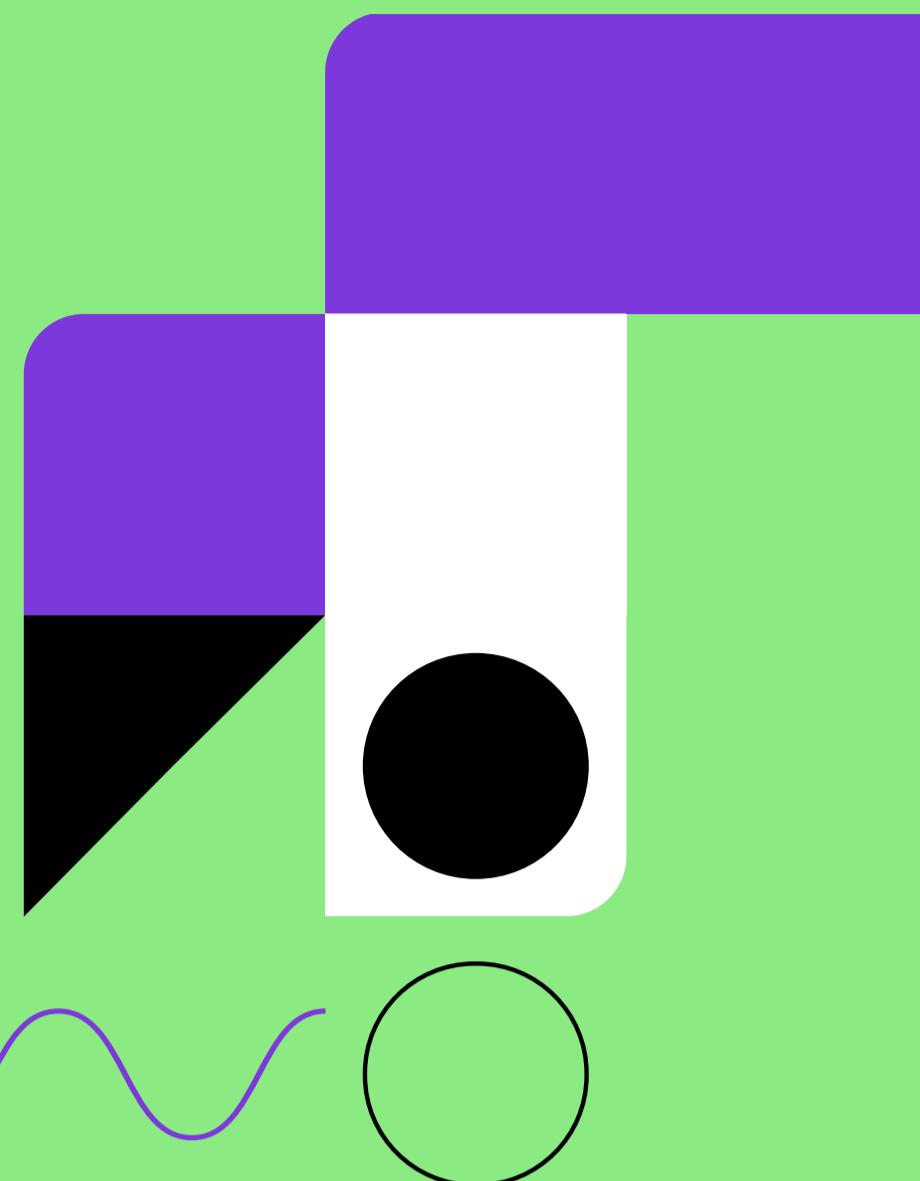
POST e PUT

Envia dados

PUT e PATCH

Atualiza dados





HTTP

Headers

Headers ou cabeçalhos permitem que você envie informações adicionais na requisição.

Ele pode ser utilizado para inúmeras funções, como: autenticação, formatação de objeto, e muito mais.

Para utilizá-lo é simples, você coloca a propriedade, seguido dois pontos e o valor, tudo entre aspas, exemplo:

```
"Authorization: token12342343534"
```

Body

O body é o corpo da mensagem que você quer enviar na requisição. Ele é utilizado somente nos métodos de POST, PUT, PATCH, ou seja, ele contém o dado a ser processado pela API, e por isso ele não é necessário em métodos de leitura de dados.



HTTP Status Codes

Para facilitar o entendimento das respostas das APIs existem padrões de códigos de status que podem ser utilizados.

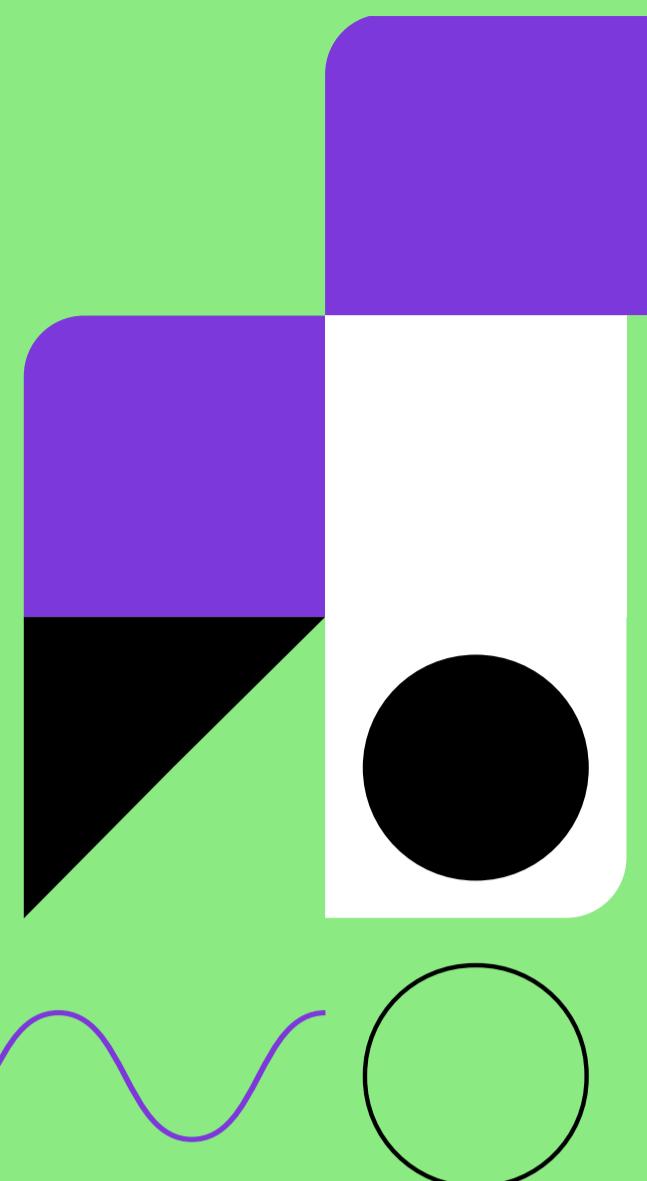
Os códigos mais utilizados para as respostas de uma requisição são os:

- > **200 (OK)**
- > **201 (created)**
- > **204 (no content)**
- > **404 (not found)**
- > **400 (bad request)**
- > **500 (internal server error).**

Existem vários outros códigos de resposta do protocolo HTTP que podem ser utilizados.

Nesse link [https://www.restapitutorial.com/
httpstatuscodes.html](https://www.restapitutorial.com/httpstatuscodes.html)) temos a lista completa.

Como padrão, os códigos de sucesso tem o **prefixo 20x, os de redirecionamento 30x, os de erro do cliente 40x e os de erro de servidor 50x.**



Autenticação e Autorização

Autorização

Obviamente não podemos falar de APIs sem segurança, afinal estamos falando da WEB.

Basic authentication

- Baseado em usuário e senha codificados em Base64 e utilizado no header da requisição.

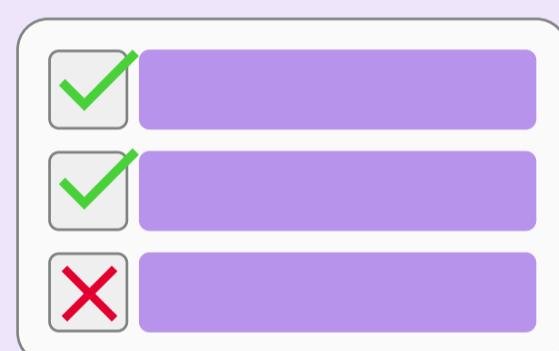
Secret token

- Token de acesso que pode ser limitado a escopo, e que é enviado na requisição pelo Header ou pela Query String.
- Nesse caso temos padrões famosos como oAuth e JWT.

Autenticação

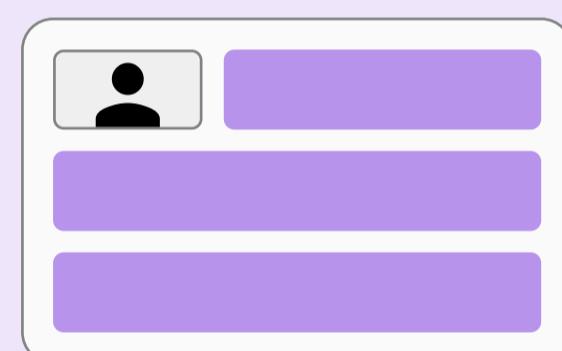
Autenticação é como se você fosse para uma reunião, e o segurança exigisse suas credenciais como nome na lista e RG para liberar sua entrada.

Autorização é como se você já estivesse dentro da reunião e quisesse falar no palanque principal da reunião, mas para isso você precisaria ser autorizado pelo segurança.



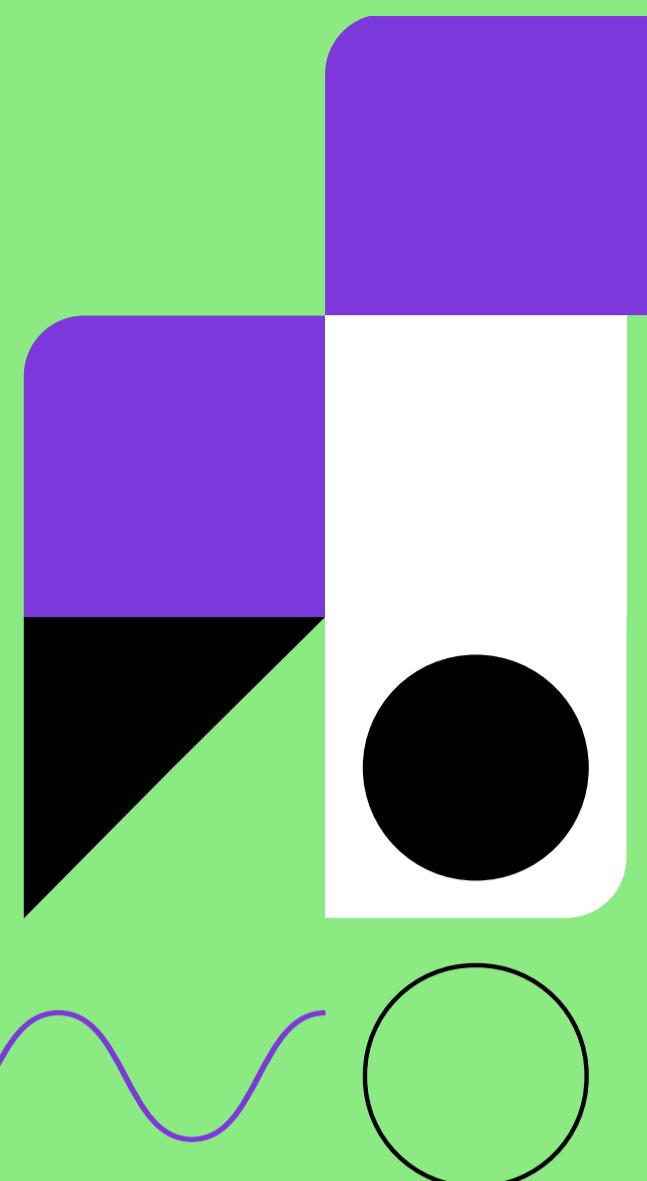
Autorização

O que você pode fazer



Autorização

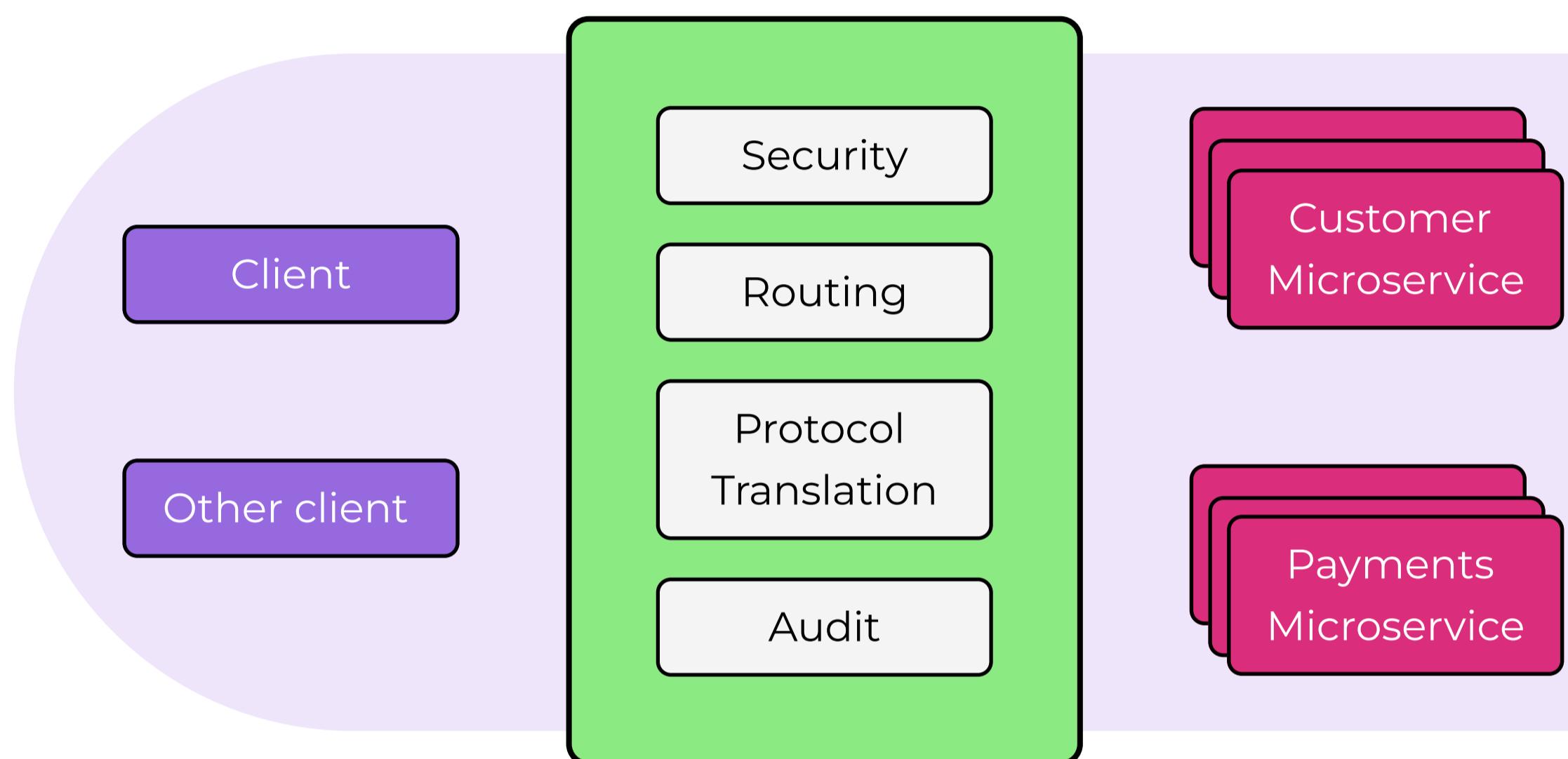
Quem você é



API Gateway

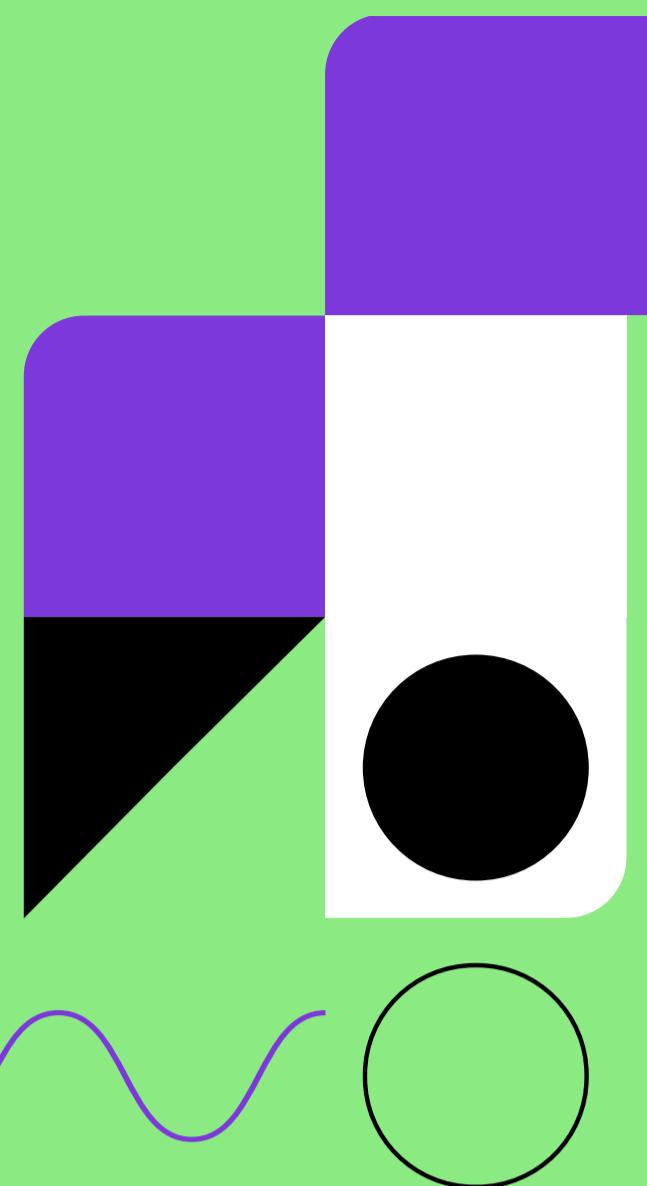
Existem diversas ferramentas que facilitam as implementações de uma API gateway centralizando toda a inteligência de segurança em uma camada específica de arquitetura.

Nesse desenho conseguimos entender esse modelo.



Algumas ferramentas recomendadas para essa arquitetura:

- > LinkApi
- > Kong
- > WSO2
- > Apigee



Versionamento

Sabe aquela situação em que um aplicativo para de funcionar de repente?

- Isso algumas vezes acontece porque a aplicação foi atualizada para uma nova versão.

Quando versionamos pela URL temos três maneiras de fazer isso:

- subdomínio
- path
- query string



Versionamento pela URL

Na forma de **subdomínio** você especifica a versão logo no início da URL, por exemplo:

HTTP GET

```
https://api-v1.minhaapi/endereco
```

O v1 especifica a versão.

No modelo de **path** você especifica a versão após a base url, por exemplo:

HTTP GET

```
https://api.minhaapi/v1/endereco
```

O v1 especifica a versão.

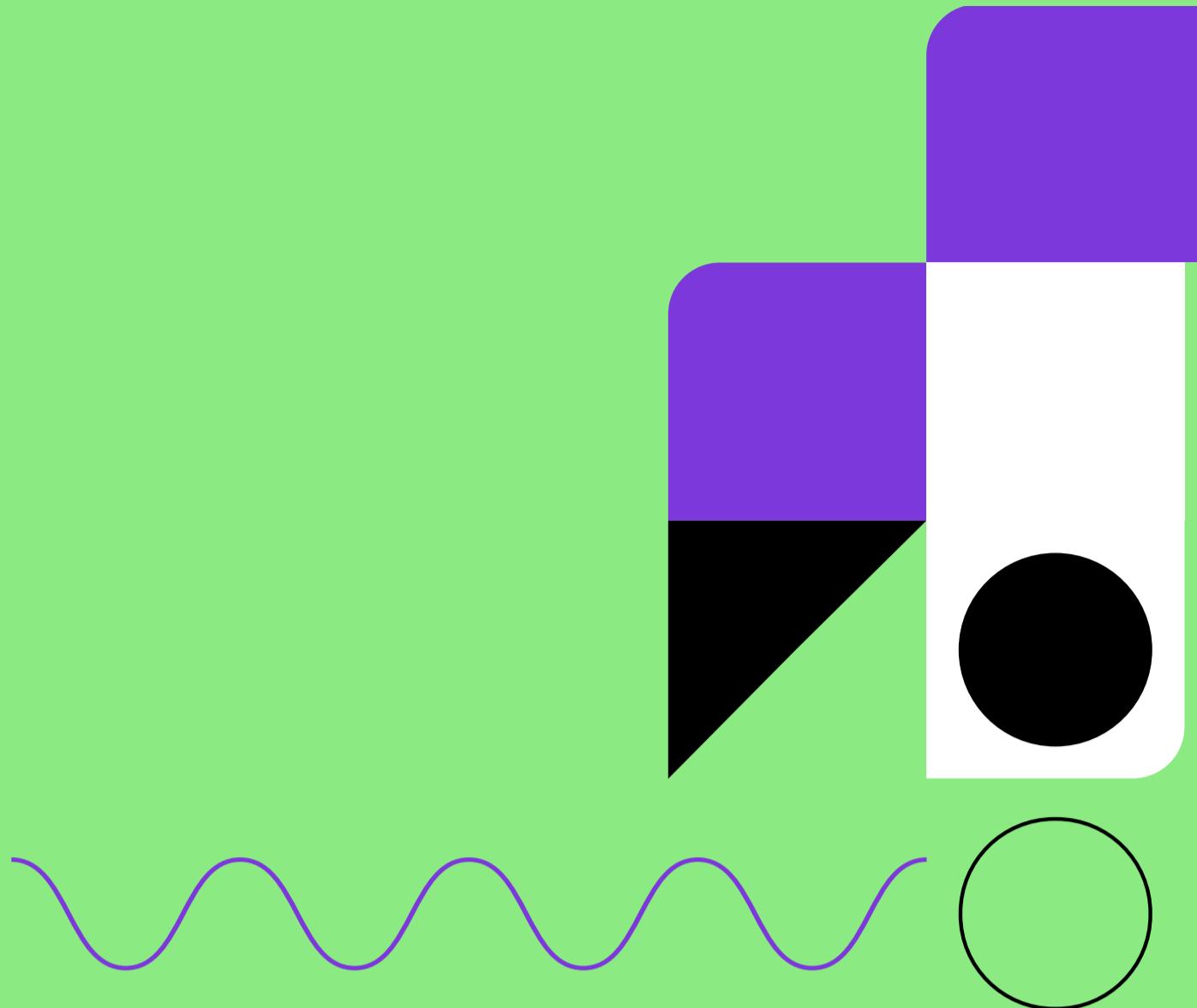
No modelo de **query string** você especifica a versão dessa forma:

HTTP GET

```
https://api.minhaapi/endereco version=1.0&pais=brasil
```

O v1 especifica a versão.





Documentação



Documentação de API é um texto escrito explicativo (ou manual de referência) que acompanha uma API.

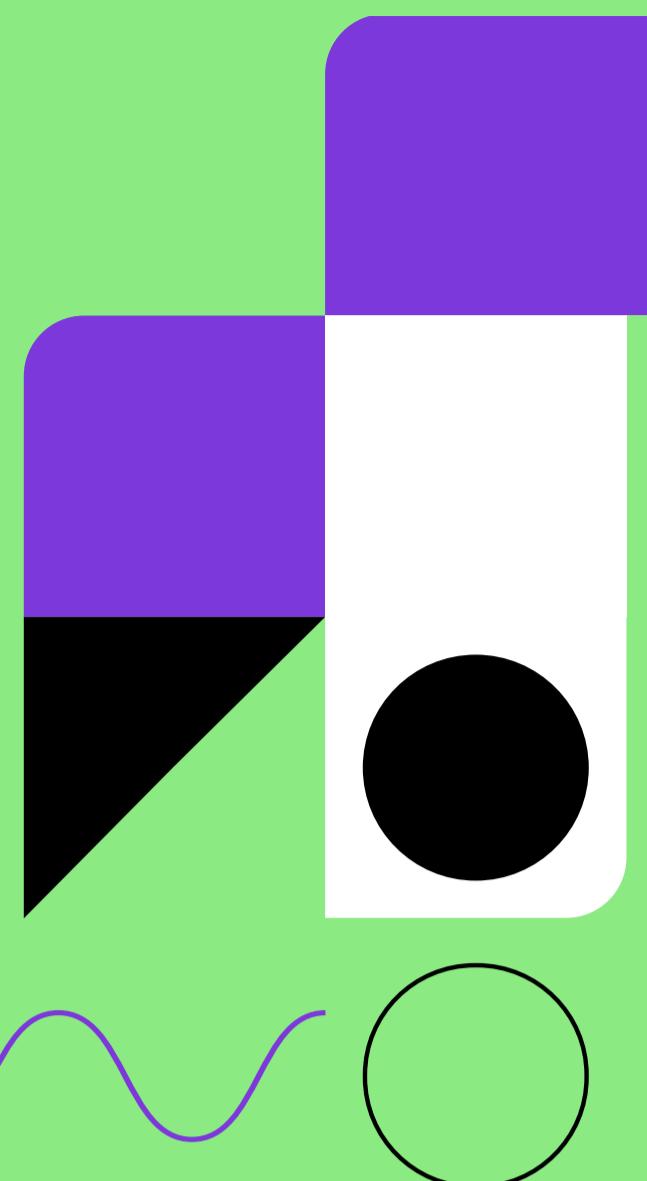
Este conteúdo explica como usar uma API de maneira correta.

Quais comandos são necessários para que sua API rode.

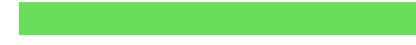
Esta documentação pode ser criada manualmente ou usando softwares e ferramentas específicas para este fim, utilizando Swagger por exemplo.

Exemplo de documentação de API do Spotify:

<https://developer.spotify.com/console/get-album/>



DX (Developer Experience)

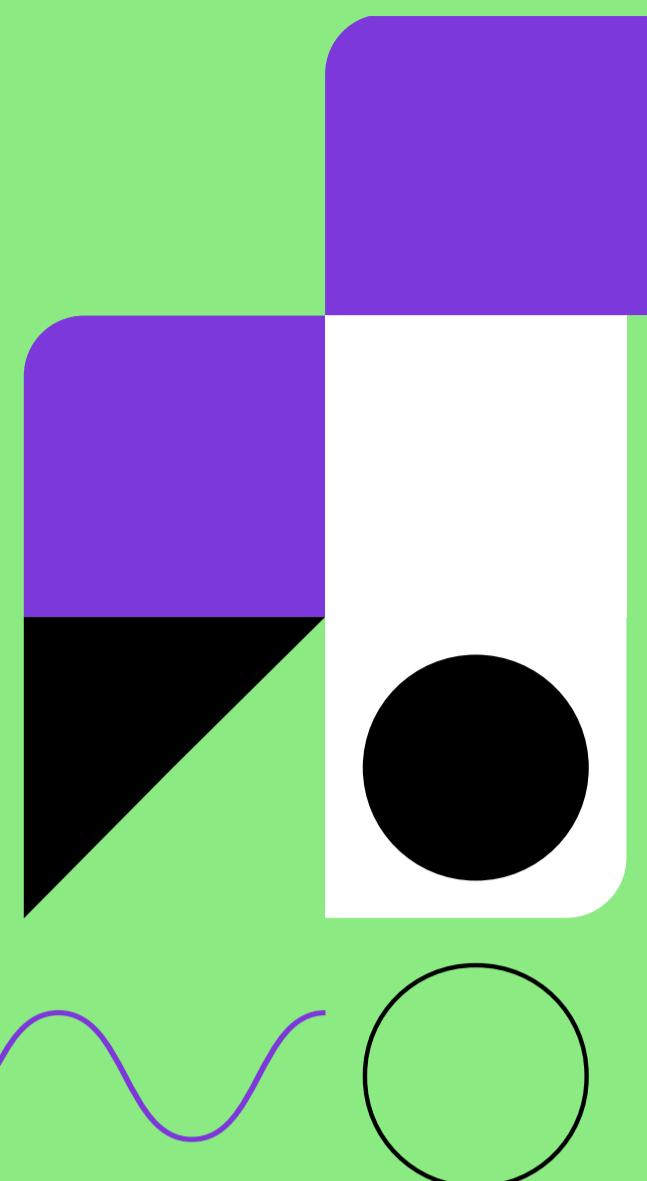


Se UI/UX é relativo à experiência do usuário, **Developer Experience** é o equivalente à experiência do Desenvolvedor.

O usuário principal do produto, nesse caso, é um **developer**.

DX se preocupa com a experiência que o developer tem ao usar um produto, por exemplo:

- > suas libs
- > SDKs
- > documentações
- > frameworks
- > soluções open-source
- > ferramentas de forma geral
- > APIs, etc.



Tutorial Swagger

Após instalar as dependências, vamos criar o arquivo **index.js** e colocar o seguinte código:

```
1 const express = require("express");
2 const app = express();
3 const swaggerUi = require('swagger-ui-express'),
4 swaggerDocument = require('./swagger.json');
5
6 app.use(
7 '/api-docs',
8 swaggerUi.serve,
9 swaggerUi.setup(swaggerDocument)
10 );
11
12 app.listen(8001, () => {
13 console.log("server listening on port 8001");
14 })
```

Em seguida criamos o arquivo **swagger.json**. Para este exemplo vamos definir apenas para a operação **SOMA**.

O código está na próxima página.



```
1  {
2    "swagger": "2.0",
3    "info": {
4      "title": "Calculadora",
5      "description": "API Calculadora com swagger",
6      "version": "1.0"
7    },
8    "host": "localhost:8001",
9    "basePath": "/",
10   "schemes": [
11     "http"
12   ],
13   "paths": {
14     "/soma/{a}&{b)": {
15       "get": {
16         "description": "Escolha dois números.",
17         "operationId": "SOMA",
18         "parameters": [
19           {
20             "name": "a",
21             "in": "path",
22             "description": "Primeiro número. Valor por omissão <code>10</code>.",
23             "required": true,
24             "default": "10",
25             "enum": [
26               "10"
27             ]
28           },
29           {
30             "name": "b",
31             "in": "path",
32             "description": "Segundo número. Valor por omissão <code>15</code>.",
33             "required": true,
34             "default": "15",
35             "enum": [
36               "15"
37             ]
38           }
39         ],
40         "responses": {}
41       }
42     }
43   }
44 }
```



Agora vamos executar o nosso projeto usando o comando:

node index.js

Abram o endereço **http://localhost:8001/api-docs/** e deverão ver a seguinte interface:

The screenshot shows a web browser window with the address bar set to `localhost:8001/api-docs/#/default/SOMA`. The page is titled "Calculadora 1.0" and includes the note "[Base URL: localhost:8001/]". Below this, it says "API Calculadora com swagger". A dropdown menu labeled "Schemes" is set to "HTTP". Under the "default" section, there is a "GET /soma/{a}&{b}" endpoint with the instruction "Escolha dois números.".

Para testar se tudo está operacional, carreguem em Try Out, indiquem dois números...

The screenshot shows the "Try Out" interface for the "/soma/{a}&{b}" endpoint. It displays two input fields: "a" with the value "10" and "b" with the value "15". At the bottom, there is a blue "Execute" button.

E depois rodem o Execute. O resultado deverá aparecer no campo **Response body**.

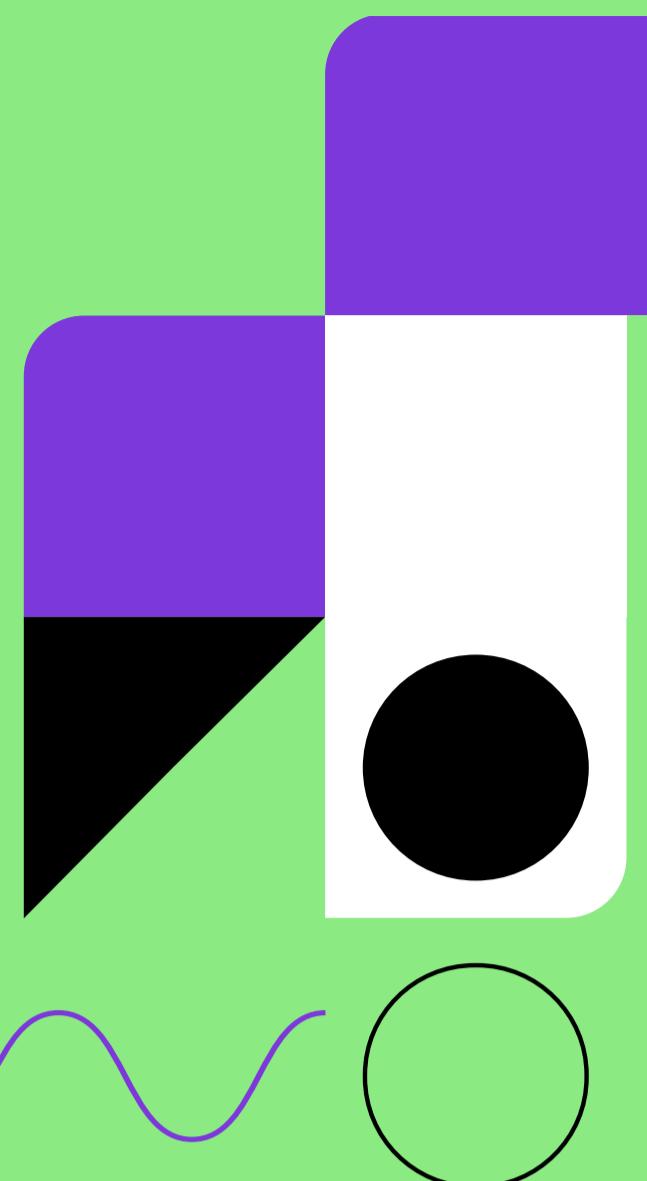


The screenshot shows a 'Responses' panel from a REST API tool. It includes a 'Curl' command, a 'Request URL' (http://localhost:8001/soma/10&15), and a 'Server response' table. The table has two rows: one for code 200 (Undocumented) and one for code 25. A red arrow points to the 'Response body' column under the 200 row, which is currently empty. The 'Response headers' section lists standard HTTP headers.

Code	Details
200 Undocumented	Response body
25	

Response headers

```
connection: keep-alive
content-length: 2
content-type: text/html; charset=utf-8
date: Thu,22 Apr 2021 13:19:13 GMT
etag: W/"2-9uE5bO3r8j4UY67nP53wh4NkBAA"
keep-alive: timeout=5
x-powered-by: Express
```



Prática de Express

O **Express.js** é uma framework para Node.js que permite o desenvolvimento de aplicações Web de uma forma muito simples.

A instalação do Express.js no ambiente de programação deve ser feito através do NPM.

O **NPM** é um gestor de pacotes para javascript.

Express

Para instalar o Express.js execute este comando, dentro do diretório de trabalho:

```
npm install express -save
```

Crie um arquivo chamado **server.js** e nele vamos iniciar o servidor e suas rotas:

```
1 // Importando o express
2 var express = require('express');
3 // Chamando o express
4 var app = express();
5 // Definindo a rota principal (home) e enviando uma
6 resposta: "Site de Tecnologia"
7 app.get('/',function(req,res){
8 res.send("Site de Treinamento");
9 });
10 // Definindo a rota /aula
11 app.get('/aula',function(req,res){
12 res.send("Categoria de Aulas");
13 });
14 // Definindo a rota /curso
15 app.get('/curso',function(req,res){
16 res.send("Categoria de Cursos");
17 });
18 // Definindo a rota /turma
19 app.get('/turma',function(req,res){
20 res.send("Categoria de Turmas");
21 });
22 // Definindo a porta que esse servidor irá executar
23 app.listen(8080,function(){
24 console.log("Servidor ativo na porta 8080");
25});
```



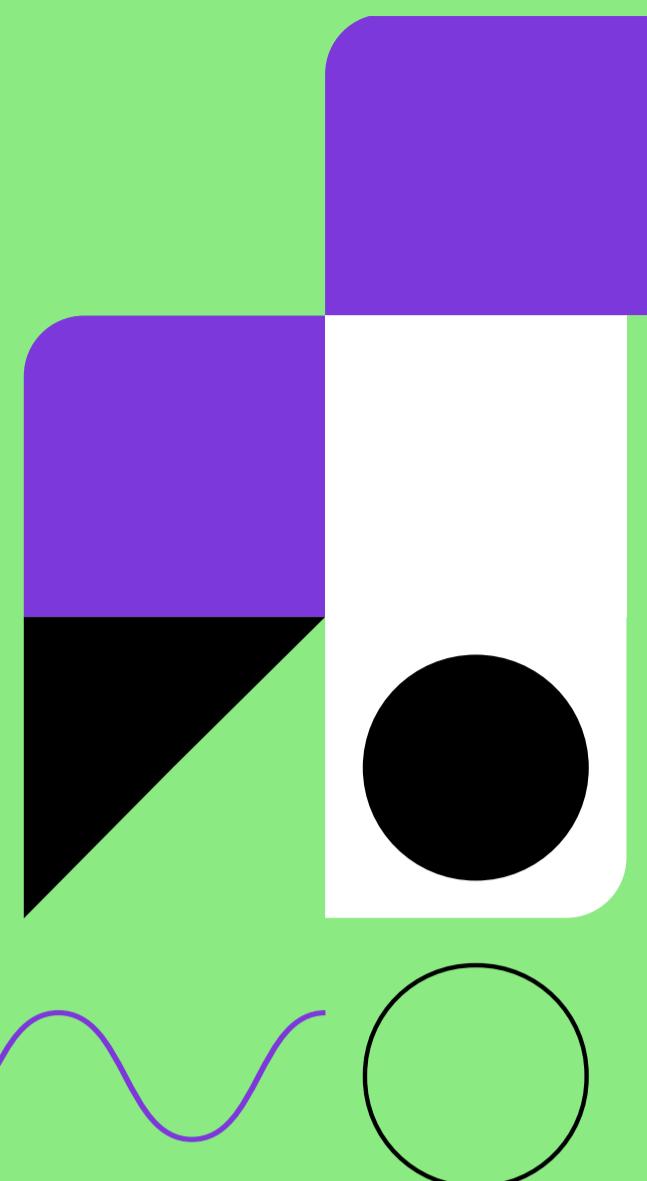
Depois no terminal, execute o comando abaixo:

```
node index.js
```

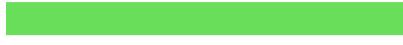
Depois, abra o browser e coloque o endereço da porta do servidor que acabou de criar e as outras rotas:

- > localhost:8080
- > localhost:8080/aula
- > localhost:8080/cursos
- > localhost:8080/turma





Segurança



A **Tecnologia da Informação** é uma área com amplos serviços. E se precisa de segurança nessas amplas áreas. A segurança da informação nada mais é do que um conjunto de estratégias que são adotadas para proteger dados e informações relacionadas à tecnologia.

Essas estratégias evitam que pessoas mal-intencionadas accessem dispositivos físicos como computadores, redes como a internet e os sistemas computacionais de uma empresa.

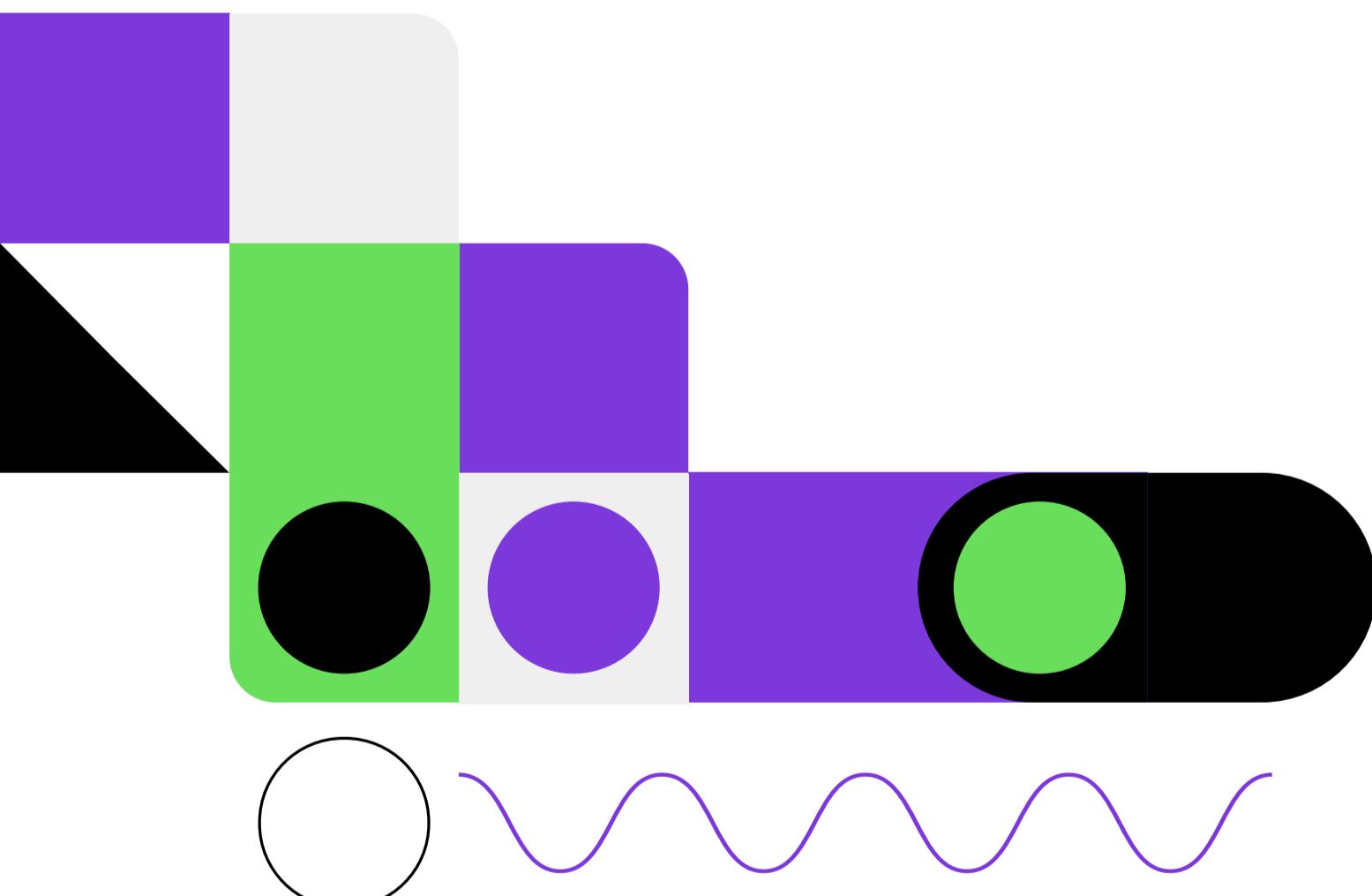
Imagine o prejuízo que é para uma empresa ter seu sistema acessado por uma pessoa não autorizada! Que pega todas essas informações e passa para uma empresa concorrente?

Investir em segurança é imprescindível nos dias de hoje, significa inclusive o sucesso da sua empresa.

Fecha mento

Vimos diversos assuntos até aqui, ainda há muito a ser abordado. Esse material pode servir como consulta e guia de estudos.

Até a próxima pessoal!





A sua aceleradora de carreiras.