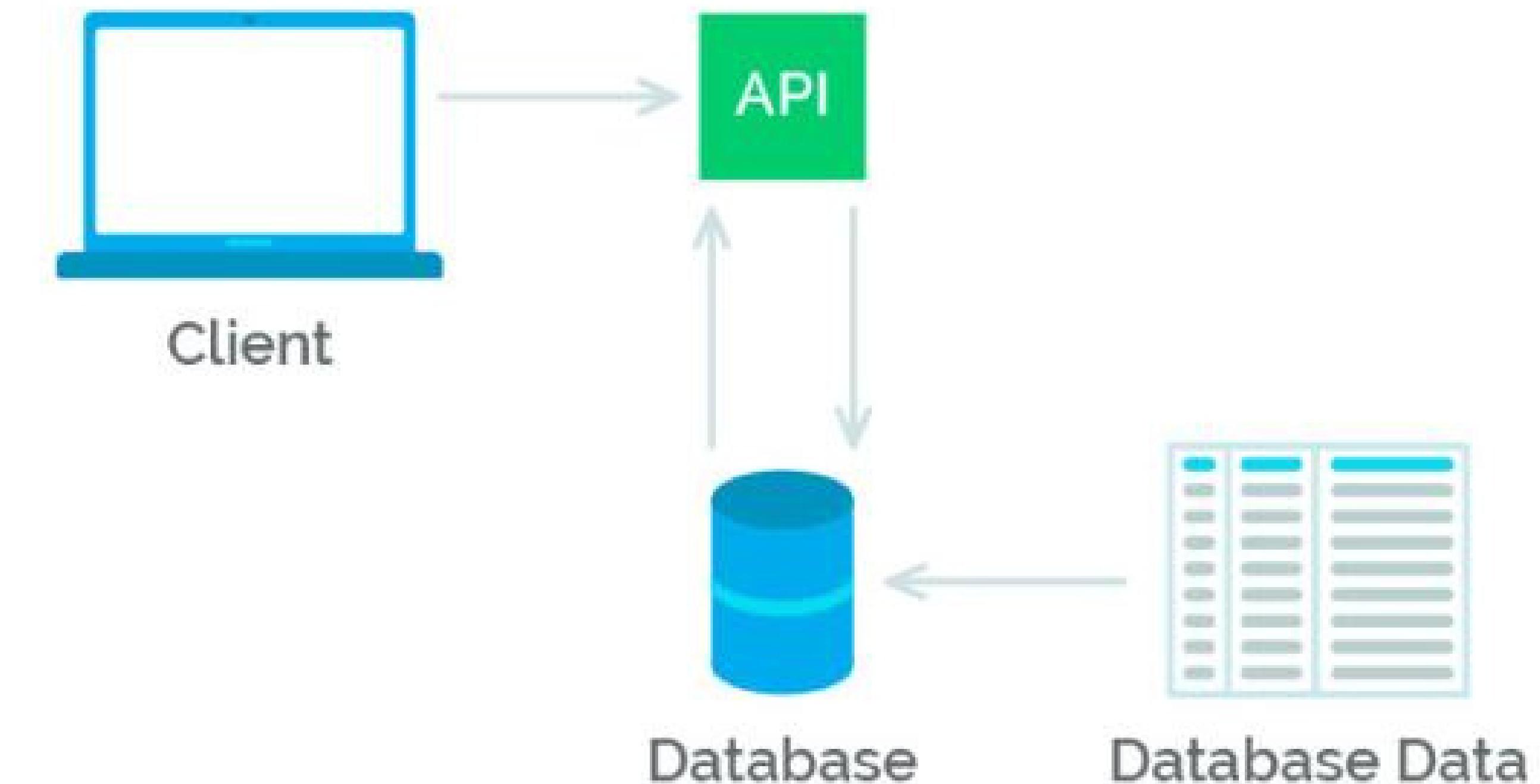


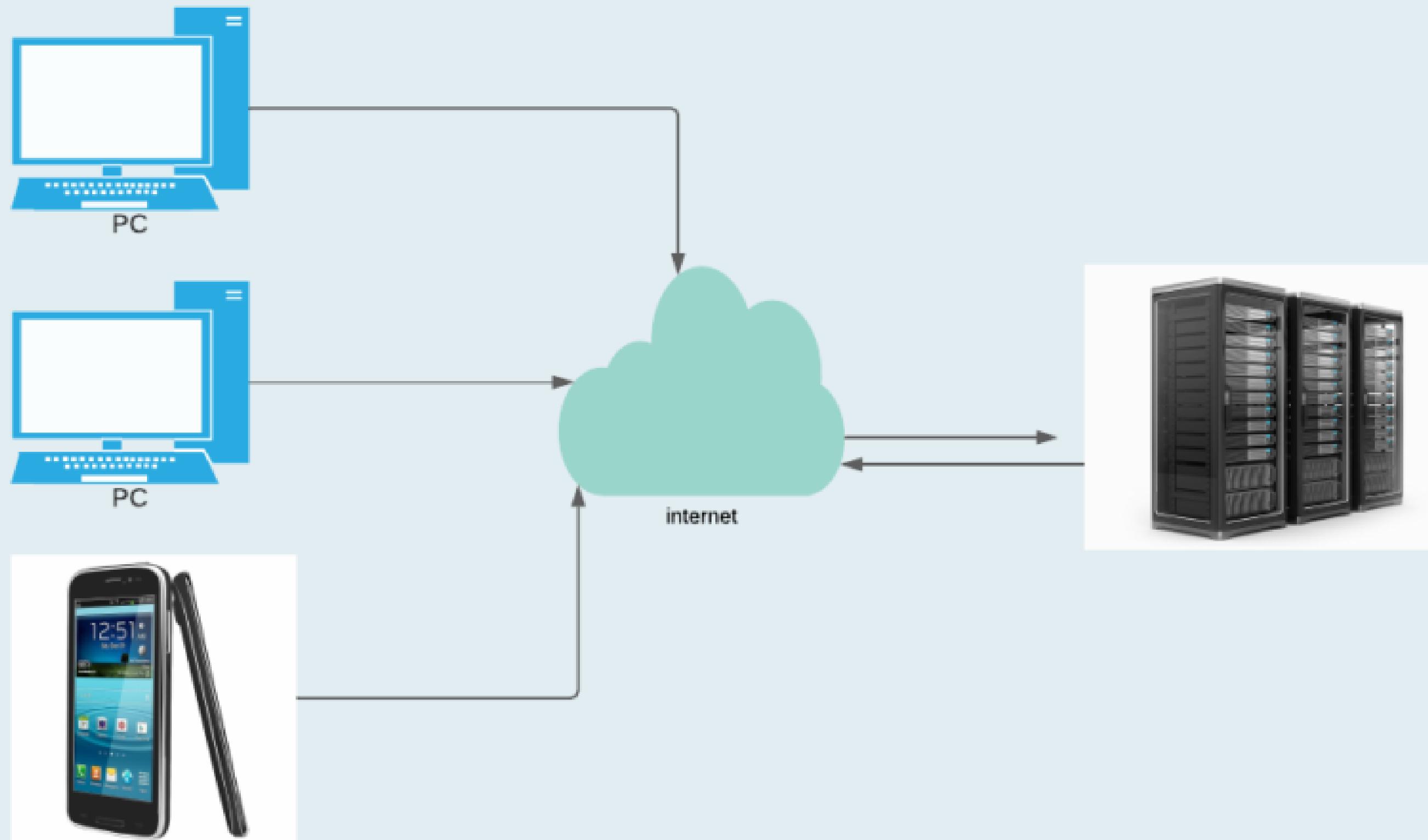
# **DESENVOLVIMENTO DE API'S: CONCEITOS E CASOS PORTICOS**



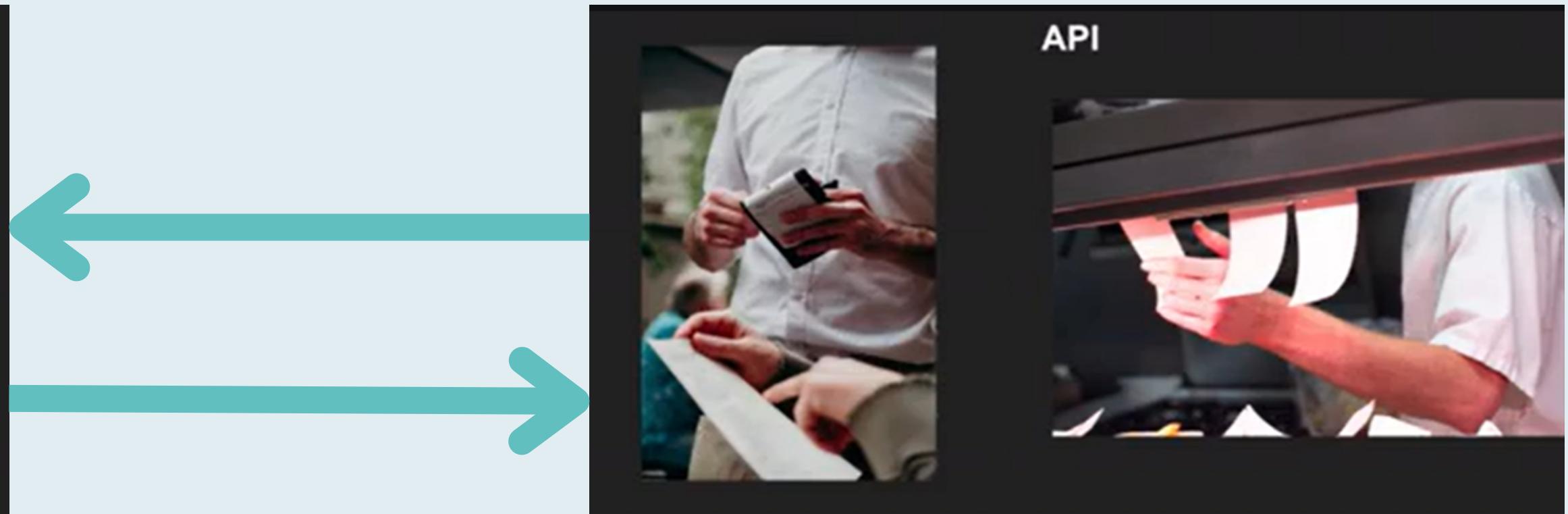
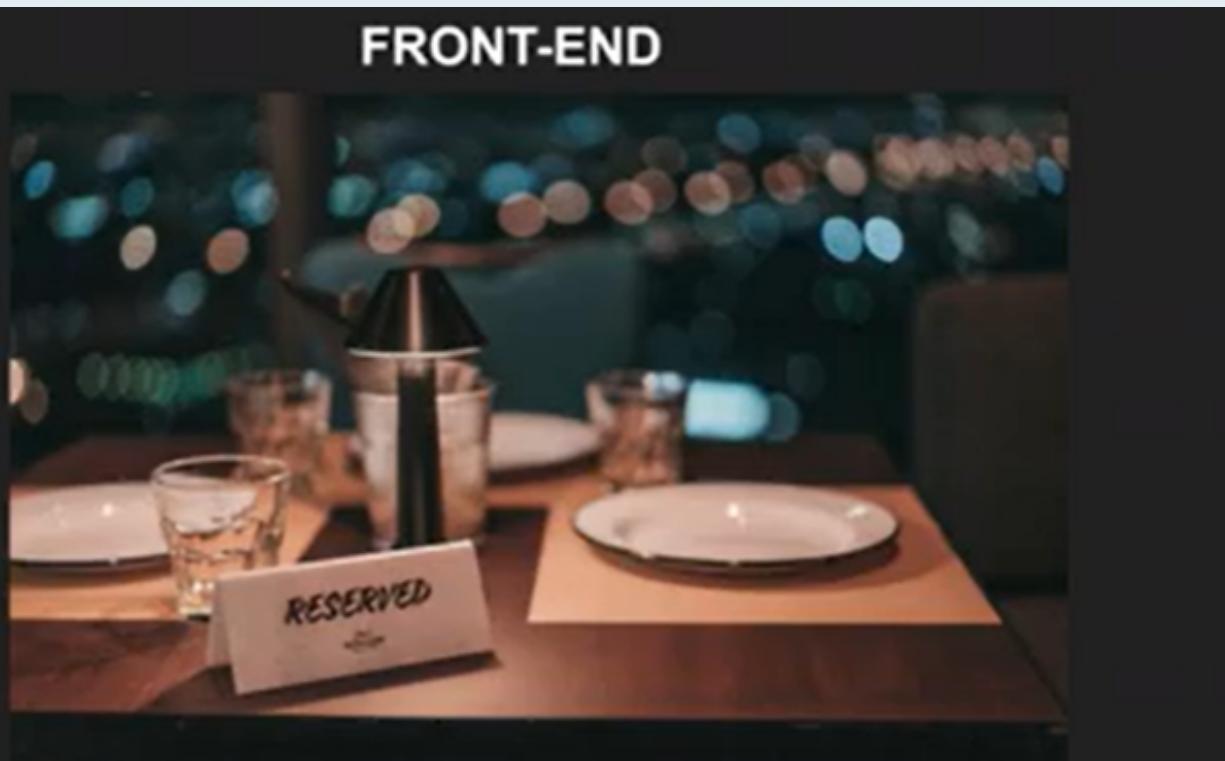
# AGENDA

- 1.Arquitetura Cliente-Servidor;
- 2.API sua contextualização;
- 3.Desenvolvimento de APIs
- 4.Protocolos da comunicação de uma API;
- 5.Códigos de status de respostas HTTP
- 6.Quais vantagens de uso de uma API;
- 7.Banco de Dados;
- 8.Graphql;

# ARQUITETURA CLIENTE-SERVIDOR



# CONTEXTUALIZACAO



# API

Application Programming Interface é um conjunto de padrões de programação que permite a construção de aplicativos e a sua utilização de maneira não tão evidente para os utilizadores.

A api do Google Maps;

Integração com apis de outros sistemas, como:

A integração de Facebook, messenger, whatsapp;

---

# TIPOS DE API

- APIs públicas ou abertas
  - APIs privadas ou internas
  - APIs de Parceiros
  - APIs compostas
-

# API PÚBLICAS

Estão disponíveis para desenvolvedores e outros utilizadores de forma pública para o consumo e com restrições mínimas. Elas podem exigir registro, uso de uma chave de API ou **OAuth**, ou mesmo podem ser completamente abertos.

---

# API PRIVADAS

São ocultadas de utilizadores externos e expostas apenas para sistemas internos de uma empresa. Elas não são usadas para o consumo fora da empresa, mas sim ao uso em diferentes equipes de desenvolvimento interno para melhor produtividade e reutilização de serviços.

---

# API DE PARCEIROS

Essas APIs são acessadas por outras pessoas fora da organização com permissões exclusivas. Normalmente, esse acesso especial é concedido a terceiros específicos para facilitar uma parceria comercial estratégica.

---

# API COMPOSTAS

Estas combinam duas ou mais APIs distintas para atender a requisitos ou comportamentos complexos do sistema.

---

# TABELA DEMOSTRAÇÃO

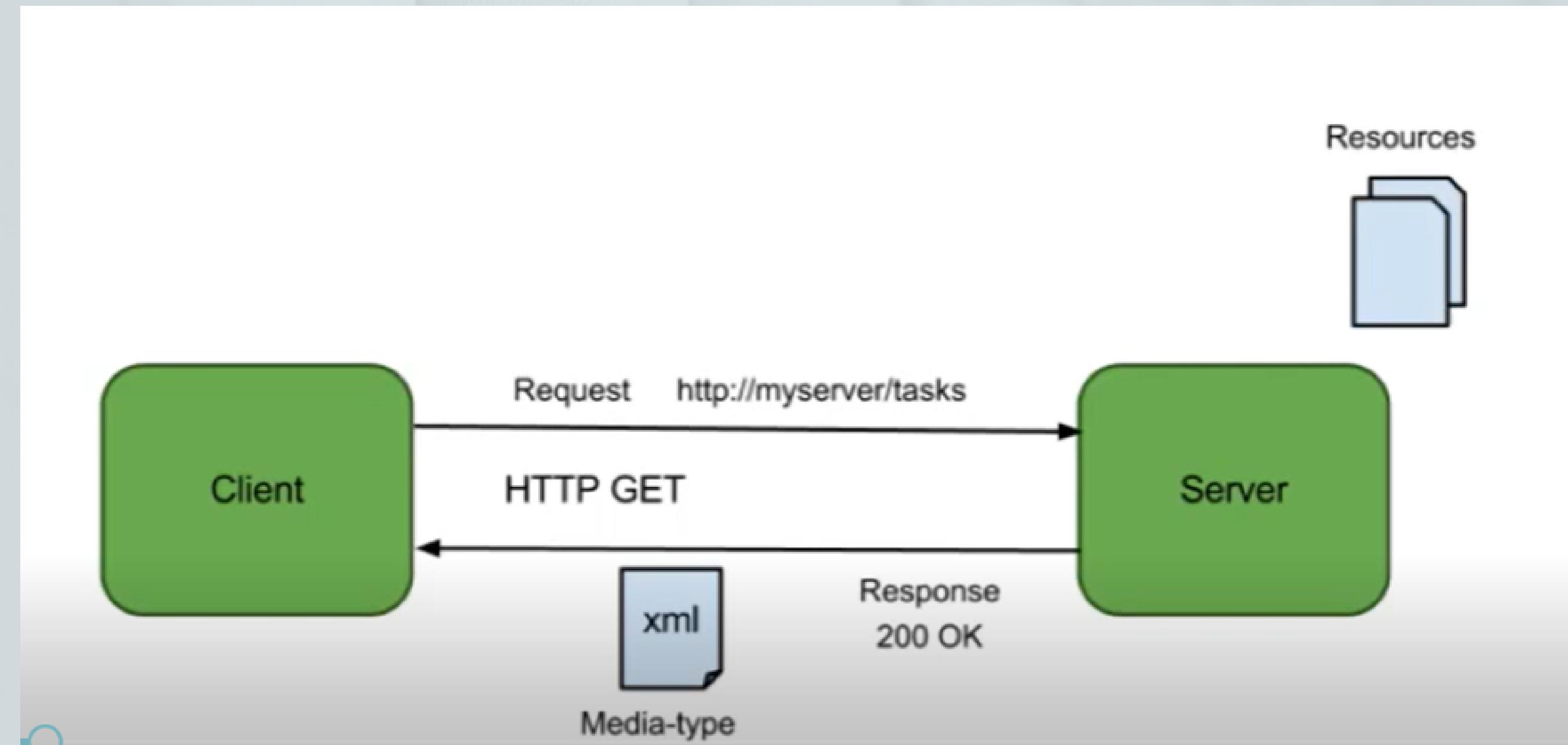
Quem usa a API	Acesso	Tipos de Apps	Exemplos
Desenvolvedores internos (APIs internas)		B2E, A2A, B2B, B2C	Empresas que mantém suas APIs para consumo interno
Desenvolvedores de Parceiros e Clientes (APIs de Parceiros e Clientes)		B2B, B2C	Empresas que precisam compartilhar dados com parceiros
Desenvolvedores externos (APIs públicas)		B2B, B2C	 Google Maps

# PROTOCOLOS DE COMUNICACAO

- **REST:** Representational State Transfer
  - **SOAP:** Protocolo de Acesso a Objeto Simples
  - **RPC:** protocolo de Chamada ProceduralRemota
-

# REST

E uma arquitetura que consiste em um conjunto de princípios, regras para a construção de serviços web, e por natureza, utiliza Arquitetura Cliente-Servidor.



# CÓDIGOS DE STATUS DE RESPOSTAS HTTP

->Indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

1.Respostas de informação (**100-199**);

2.Respostas de sucessos (**200-299**);

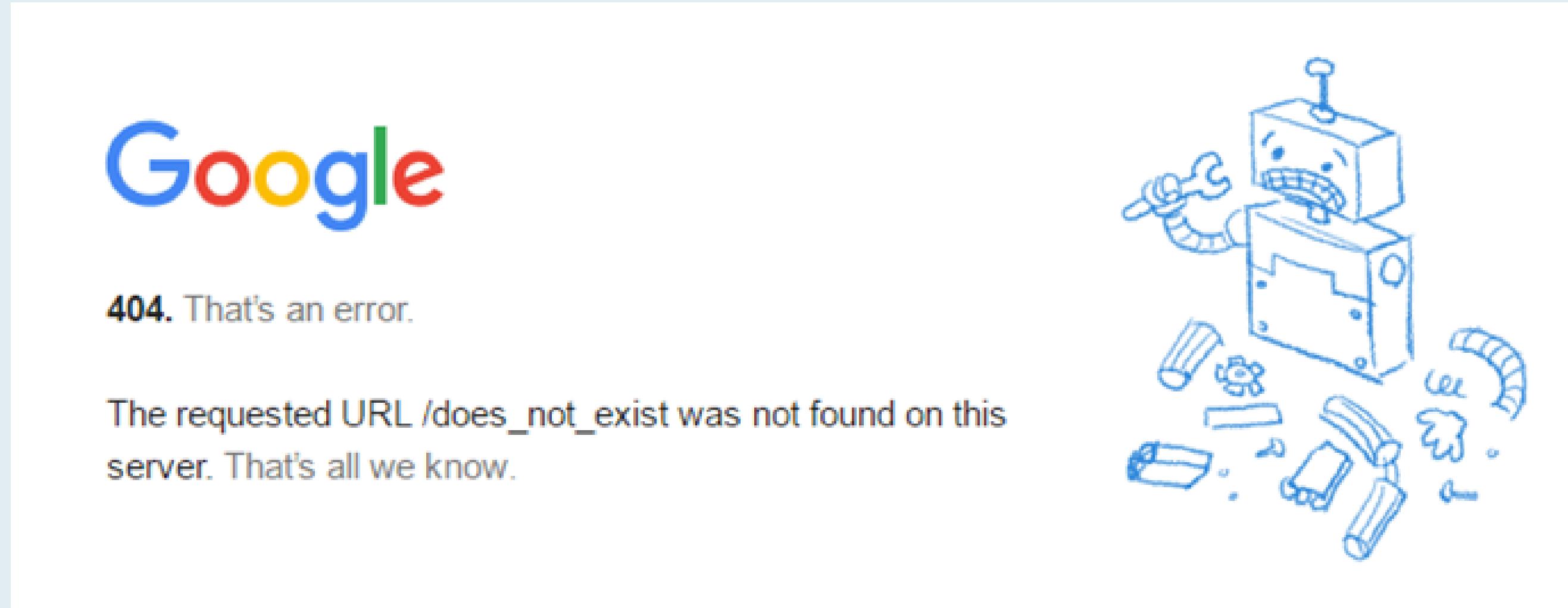
3.Recebimentos (**300-399**);

4.Recebimento do cliente (**400-499**);

5.Erros do servidor (**500-599**).

---

# CÓDIGOS DE STATUS DE RESPOSTAS HTTP



# PRINCÍPIOS DE UMA API RESTFUL EFICIENTE

Mas para que uma API seja RESTful, ela deve seguir as seguintes regras:

**Interface uniforme:** a comunicação entre cliente e servidor será através do protocolo HTTP, URIs(identificadores de recursos exclusivos), CRUD, e conexões Json.

**Cliente-servidor:**

**Cache:** o cliente deve armazenar em cache as respostas, pois isso melhora a experiência do usuário, tornando-as mais rápidas e eficientes.

**Em camadas:** concede que a solução seja menos complexa, altamente flexível e desacoplada.

---

# PROTOCOLO HTTP

**HTTP**, ou Hypertext Transfer Protocol, foi inventado por Tim Berners-Lee na década de 1980.

É um sistema de regras, protocolo, que serve de ligação entre as aplicações e a transferência de documentos de hipertexto;

---

# MÉTODOS DE REQUISIÇÃO HTTP

**GET:** Indica que queremos apenas consultar informações de uma aplicação.

**POST:** Serve para enviar informações para as nossas aplicações.

**PUT:** Serve para atualizar um determinado recurso.

**DELETE:** Remove um recurso específico

---

# JSON (JAVASCRIPT OBJECT NOTATION)

É um formato de arquivo para troca de informações.

```
{  
  "data":  
    {"estado":"inativo",  
     "data_marcacao":"2010-03-03",  
     "data_consulta":"2022-03-02",  
     "medico":2  
    }  
}
```



# O QUE É UM ENDPOINT DE API E POR QUE ELE É IMPORTANTE?

são os pontos de contato finais no sistema de comunicação da API. Estes incluem URLs de servidores, serviços e outros locais digitais específicos de onde as informações são enviadas e recebidas entre sistemas.

Os endpoints da API são fundamentais para as empresas por dois motivos principais:

---

# O QUE É UM ENDPOINT DE API E POR QUE ELE É IMPORTANTE?

## ● Segurança

Os endpoints da API tornam o sistema vulnerável a ataques. O monitoramento da API é crucial para impedir o uso indevido.

## ● Performance

Os endpoints da API, especialmente os de alto tráfego, podem causar gargalos e afetar a performance do sistema.

# COMO PROTEGER UMA API REST?

- **Tokens de autenticação**

Eles são usados para autorizar os usuários a fazer a chamada de API.

- **Chaves de API**

As chaves de API verificam o programa ou a aplicação que faz a chamada de API. Elas identificam a aplicação e garantem que ela tenha os direitos de acesso necessários para fazer a chamada de API específica.

# PROTOCOLOS DE COMUNICAÇÃO DE API

**SOAP:** significa Protocolo de Acesso a Objeto Simples

## Características:

- Surgiu nos anos 1990;
  - Possui regras rígidas;
  - padrões rígidos eram muito pesados;
  - Em algumas situações, consumiam muitos recursos.
-

# PROTOCOLOS DE COMUNICAÇÃO DE API

**RPC:** protocolo de Chamada Procedural Remota

## Objectivos:

- o cliente executasse o código em um servidor. XML-RPC usava XML para codificar suas chamadas, enquanto JSON-RPC usava JSON para a codificação.
-

# PROTOCOLOS DE COMUNICAÇÃO DE API

## RPC

**Vantagens:** Simples;

**Desvantagens:**

- vAPIs RPC são fortemente acopladas, então isso torna difícil mantê-las ou atualizá-las;
  - vPara fazer qualquer mudança, um novo desenvolvedor teria que passar por várias documentações de RPCs para entender como uma mudança afeta a outra.
-

# QUAIS VANTAGENS DE USO DE UMA API

## Segurança

As APIs são seguras, pois criam uma espécie de barreira que permite acesso apenas às informações que fazem parte daquela aplicação, e não ao sistema por inteiro.

## Menor volume de dados:

São inseridos no sistema apenas os dados que realmente são necessários para a ação esperada.

## Aumenta a eficiência de sistemas e aplicativos

# Banco de Dados

ARTIGO

## **BANCO DE DADOS**

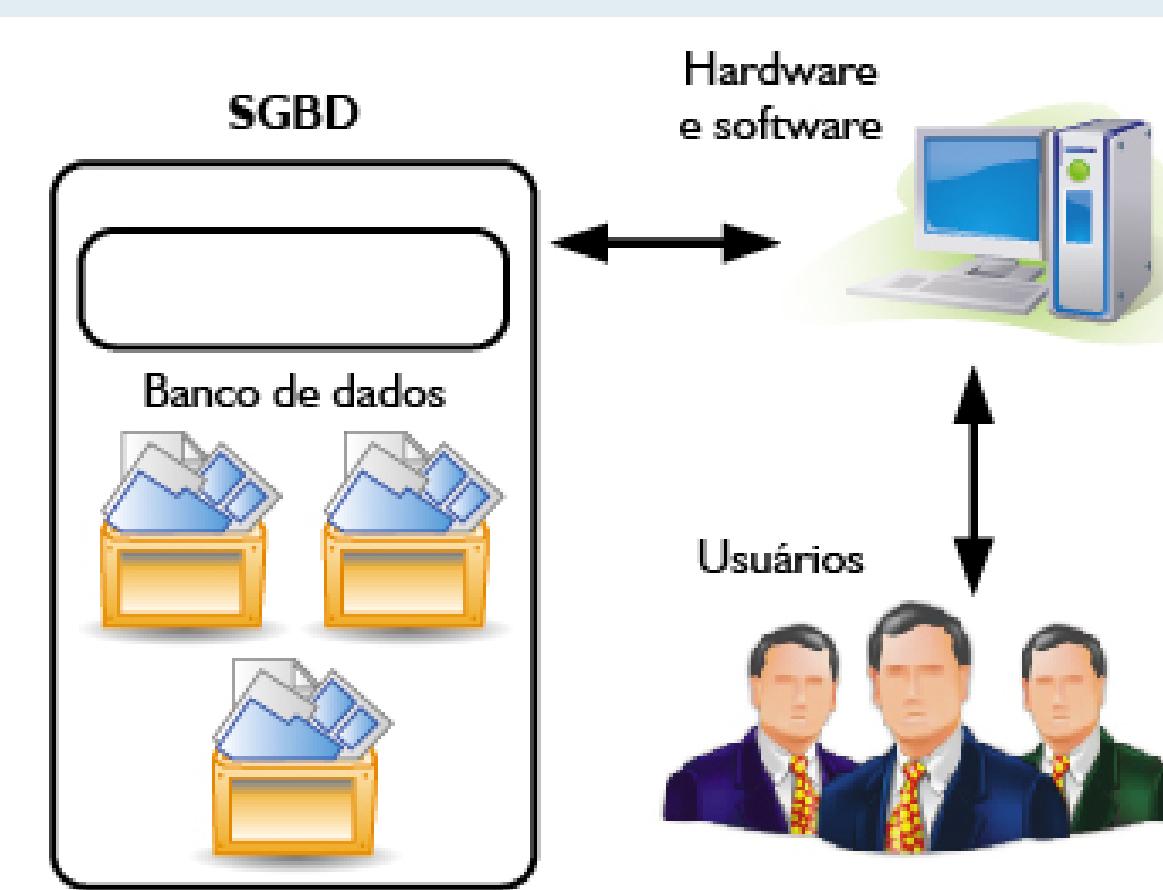
Conceitos Fundamentais



# ELEMENTOS QUE COMPÕE UM BANCO DE DADOS:



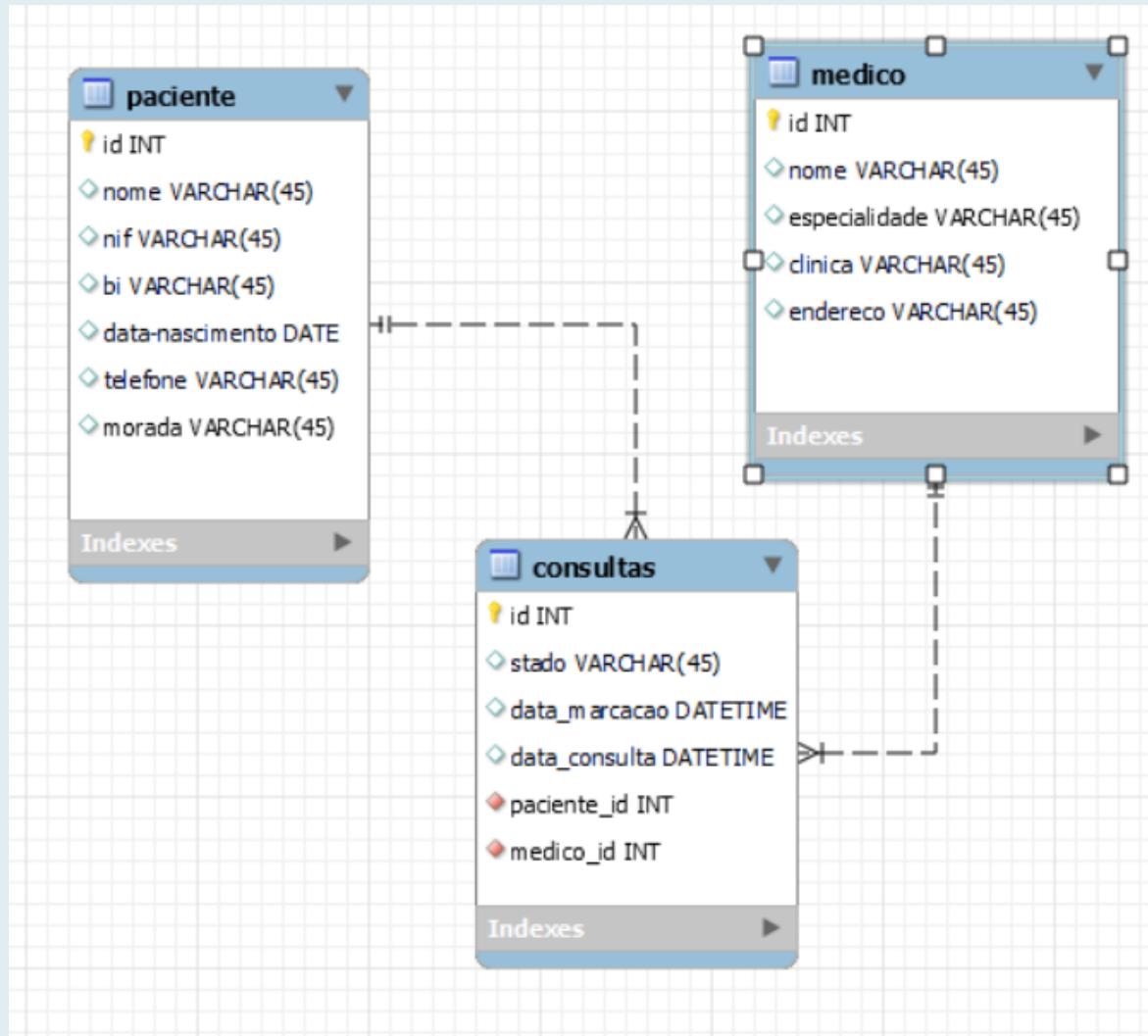
Arquivos de Banco de Dados;



Linguagem de Conversação

# CATEGORIAS DE BANCO DE DADOS

- Relacional (SQL)
- Nao Relaciona (NoSQL)



```
{  
  "data":{  
    "estado":"inativo",  
    "data_marcacao":"2010-03-03",  
    "data_consulta":"2022-03-02",  
    "medico":2,  
    "paciente":2  
  }  
}
```

# Tipos de Banco de Dados Relacional(SQL)



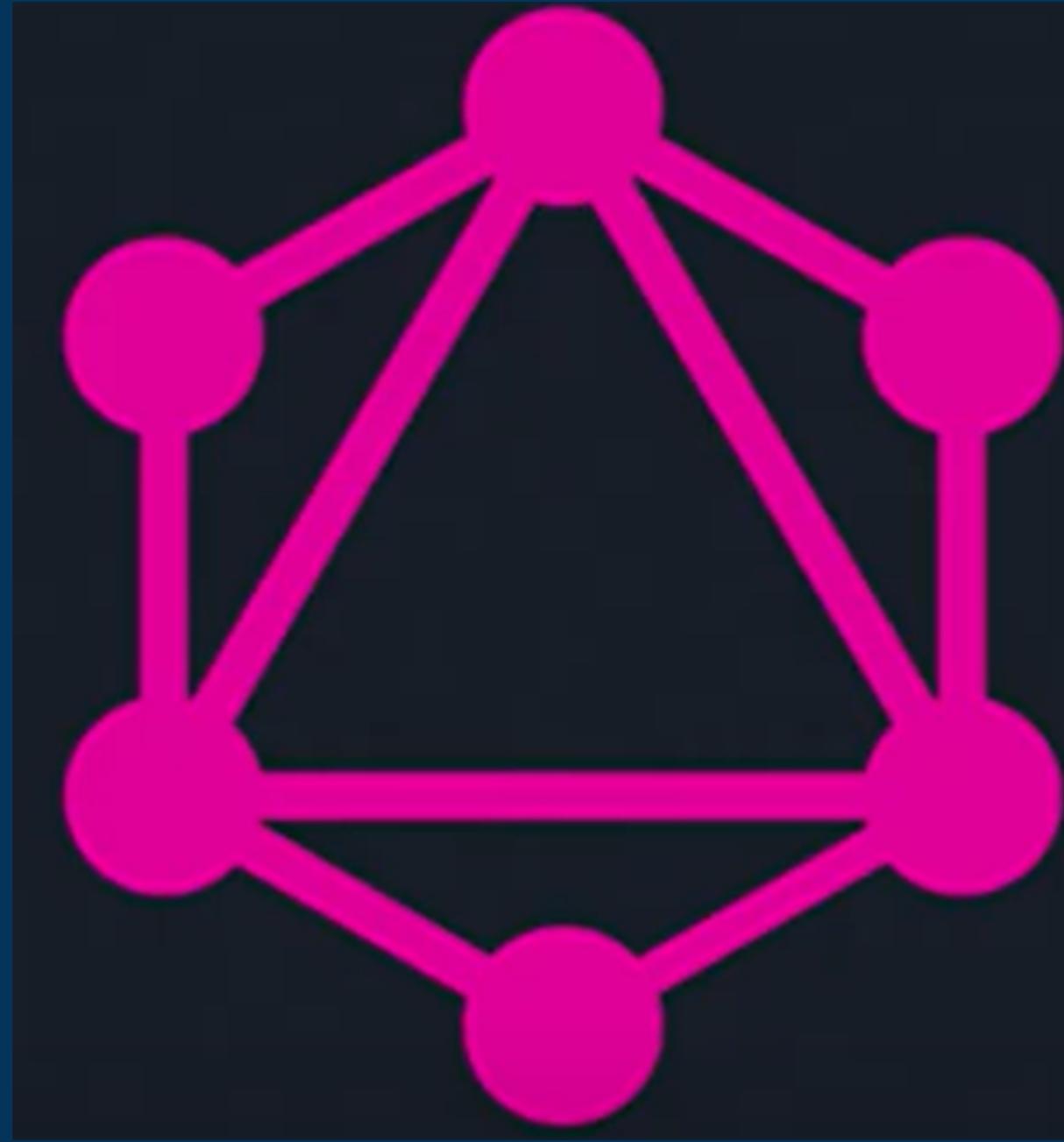
**ORACLE**



# Tipos de Banco de Dados Não Relacional(NoSQL)



# GRAPHQL



• Criada pelo Facebook  
em 2012 e lançada  
publicamente em 2015;

é uma linguagem de consulta e ambiente de execução voltada a servidores,  
para as interfaces de programação de aplicações (APIs) cuja prioridade é fornecer exatamente os dados que os clientes solicitam e nada além.

# GRAPHQL VS REST

- Rest:

Alta possibilidade de over (dados demasiados) ou under fetching (dados a menos).

Ex: No Frontend, ao listar dados de um utilizador, por vezes só precisamos apenas do nome do utilizador, mas no backend, criamos um endpoint que obtém dados completo do utilizador.

- Graphql:

O Graphql remove o requisitos de backend de criar endpoint específicos para cada tipo de dados necessários; Um desenvolvedor só precisa chamar um único endpoint e isso reduz o trabalho, sempre que algo novo ou alterações surgem nas especificações, etc.

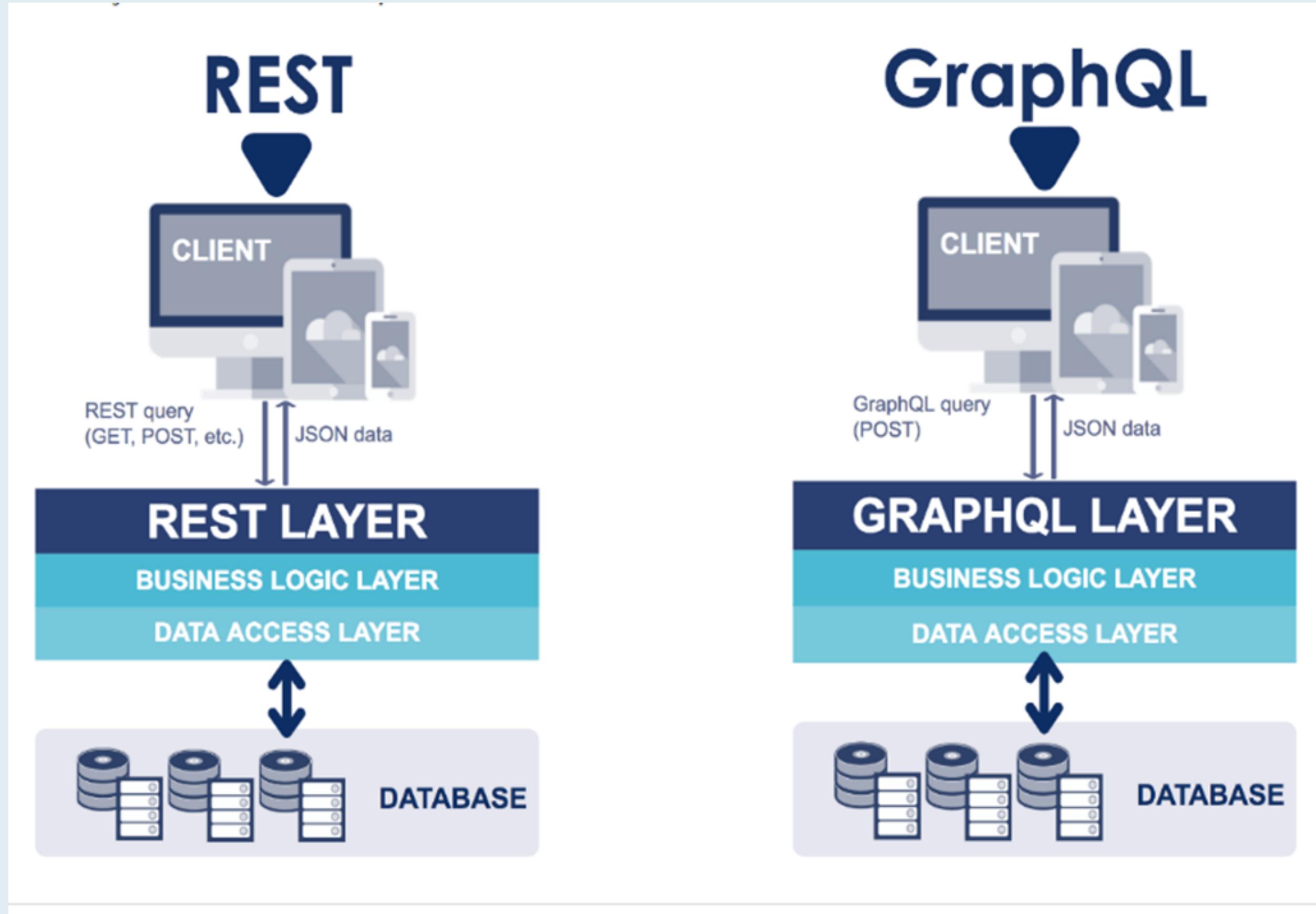
# UM RESUMO ENTRE REST E GRAPHQL

- **endpoints únicos que recebem parâmetros dinâmicos**
  - **permite um desenvolvimento mais dinâmico uma vez implementado**
  - **fornecce servidores sem estado e acesso controlado flexível aos recursos. Enquanto no REST, é controle completo e os dados são fornecidos com base no que o back-end codificou, o GraphQL coloca controles sobre o que pode ser acessado, mas deixa a estrutura dos dados para o front-end.**
  - **envia apenas uma única solicitação no corpo e inclui uma consulta que contém todos os requisitos de dados. Isso traz dados para o front-end e reduz a quantidade de carga de trabalho necessária no back-end para se ajustar às necessidades de mudança dos requisitos de dados.**
  - **o campo raiz retornará um campo de dados com todos os dados que você deseja e no formato que você precisa**
  - **permite iteração rápida e feedback mais rápido para o negócio**
-

# UM RESUMO ENTRE REST E GRAPHQL

## Rest

- vários endpoints
  - estrutura de dados é controlada pelo backend
  - fornece servidores sem estado e acesso estruturado a recursos
  - frontend não tem controle sobre como e como os dados se parecem
  - estruturado de acordo com a visão que serve. Isso significa que o back-end precisa criar novas APIs ou ajustar as existentes para fornecer a quantidade e o tipo de dados corretos. Isso pode levar a um maior tempo de produção e retardar o processo de desenvolvimento.
-



Arquitectura de sistema utilizando O Rest e GraphQL

# PORQUE USAR O GRAPHQL ?

## Por que usar:

- Fornece uma descrição completa e compreensível dos dados em sua API;
  - Oferece aos clientes o poder de solicitar exatamente o que eles precisam e nada mais.
  - Facilita a evolução das APIs ao longo do tempo e fornece poderosas ferramentas ao desenvolvedor.
-

# QUAIS AS OPERAÇÕES DO GRAPHQL ?

## ● Consultas

```
query {  
  locations {  
    data {  
      id  
      attributes {  
        country  
        City  
        geolocation  
      }  
    }  
  }  
}
```

## ● Mutações

```
mutation CreateLocation($location:  
  JSON, $city: String,  
  $country: String){  
  createLocation(data:{  
    geolocation:$location,  
    City: $city,  
    country: $country,  
  })  
  data{  
    id  
    attributes{  
      geolocation  
    }  
  }  
}
```

“

## CASO PRATICO:

Marcação de consultas

## OBJETIVO DE SISTEMA

Fazer uma demonstração de um sistema que por vezes pode ser utilizado numa aplicação de sistema de gerenciamento clínicos, pois demonstra um pequeno fluxo do processo de marcação de consultas de um paciente.

# CASO PRATICOS

## ● Caso Pratico I:Requisitos

- Requisitos:
- Node js;
- Editor de texto (Visual Studio Code);
- Google Maps;
- Postman;

## ● Caso Pratico II: Requisitos

- MySql WorkBench;
- Editor de texto (Visual Studio Code);
- Strapi io;
- Banco de Dados Sql lite/Postgres;
- Postman;
- Graphql;
- Docker;

# CONCLUSAO

- Com a crescente demanda por sistemas web encontramos sempre a necessidade de integração entre os diferentes sistemas no ambiente na internet, ou ate mesmo a distribuição de um grande sistema em fragmentos independentes. Que normalmente são conhecidos como Serviços. E muitas das vezes esses serviços são desenvolvidos em diferentes Linguagens de Programação.
  - Com isso, a cada dia são geradas grandes quantidades de informação, tecnologias, dispositivos e servidores que necessitam de comunicação entre si, graças a API.
-

# RECOMENDACOES

**R•What is a network?**

- What is the internet?**
  - What is an IP address?**
  - What is a router?**
  - What is an ISP?**
  - What are packets and how are they used to transfer data?**
  - What is a client?**
  - What is a server?**
  - What is a web page?**
  - What is a web server?**
  - What is a web browser?**
  - What is a search engine?**
  - What is a DNS request?**
  - Which browser are you currently using?**
  - In your own words, explain what happens when you run a search on google.com.**
-

# LINKS DE ACOMPANHAMENTO

[----GitHub](https://raw.githubusercontent.com/Chingling152/SQL-SPMedGroup/master/Modelos/ModelagemLogica.png)  
[----strapi token](https://docs.strapi.io/user-docs/latest/settings/managing-global-settings.html#managing-api-tokens)

[----Status do Servidor](https://developer.mozilla.org/pt-BR/docs/Web/HTTP>Status)

[--- Graphql vs Rest](https://www.tabnine.com/blog/rest-api-vs-graphql/?utm_term=&utm_source=google.com&utm_medium=cpc&utm_campaign=14854202152&utm_content=&gclid=Cj0KCQjwyMiTBhDKARIlsAAJ-9VuTG6LyvMUrQnDvd6KzIBy47IG5ZOkFp8QkHGeKsPlw2ozt2vYYe6oaAnX1EA_Lw_wcB)