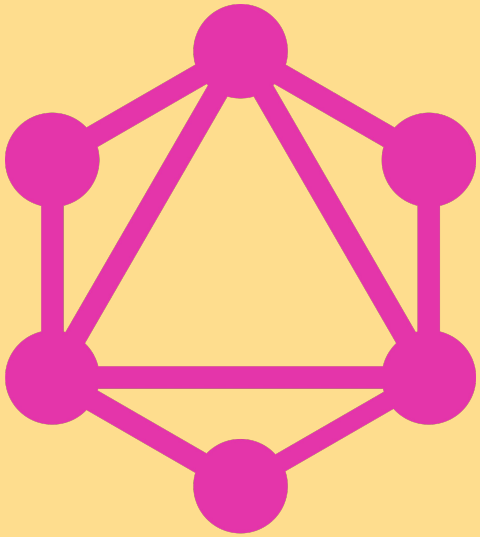


GraphQL

Live de Python # 185

Roteiro

1. O GraphQL

Um entendimento básico sobre o assunto

2. Strawberry

Nossa ferramenta de GraphQL em python

3. Construindo uma aplicação

Com FastAPI e SQLModel

4. Integrando tudo

A magia acontece aqui

picpay.me/dunossauro

apoia.se/livedepython

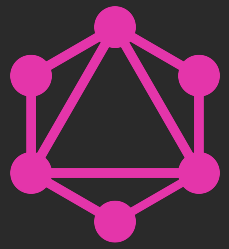
PIX

Ajude o projeto <3

A Earth, Acássio Anjos, Ademar Peixoto, Alex Lima, Alexandre Harano, Alexandre Santos, Alexandre Takahashi, Alexandre Tsuno, Alexandre Villares, Alynne Ferreira, Alysson Oliveira, Amaziles Carvalho, Ana Carneiro, Andre Azevedo, André Rocha, Antonio Neto, Apolo Santos, Arnaldo Turque, Artur Zalewska, Bruno Barcellos, Bruno Freitas, Bruno Guizi, Bruno Oliveira, Bruno Ramos, Caio Nascimento, Carlos Chiarelli, Carlos Eduardo, Cleber Santos, César Almeida, Dartz Dartz, David Kwast, Diego Guimarães, Diego Ubirajara, Dilenon Delfino, Dino Aguilar, Donivaldo Sarzi, Douglas Zickuhr, Emerson Rafael, Eric Niens, Eugenio Mazzini, Euripedes Borges, Fabiano Gomes, Fabio Barros, Fabio Castro, Felipe Rodrigues, Fernando Silva, Flavkaze Flavkaze, Flávio Meira, Francisco Alencar, Franklin Silva, Fábio Barros, Gabriel Sarmento, Gabriel Simonetto, Geandreson Costa, Guilherme Castro, Guilherme Felitti, Guilherme Gall, Guilherme Ostrock, Gustavo Suto, Henrique Junqueira, Henrique Machado, Ismael Ventura, Israel Fabiano, Israel Gomes, Italo Silva, Jair Andrade, Jairo Rocha, Johnny Tardin, Jonatas Leon, Jonatas Oliveira, Jorge Plautz, Jose Mazolini, José Gomes, José Prado, João Lugão, Juan Gutierrez, Julio Silva, Jônatas Silva, Kaio Peixoto, Kaneson Alves, Leandro Miranda, Leonardo Cruz, Leonardo Mello, Lidiane Monteiro, Lucas Barros, Lucas Mello, Lucas Mendes, Lucas Oliveira, Lucas Polo, Lucas Teixeira, Lucas Valino, Luciano Ratamero, Luciano Silva, Luciano Teixeira, Maiquel Leonel, Marcela Campos, Marcelino Pinheiro, Marco Yamada, Marcos Ferreira, Maria Clara, Marina Passos, Matheus Vian, Murilo Cunha, Márcio Martignoni, Natan Cervinski, Nicolas Teodosio, Osvaldo Neto, Patric Lacouth, Patricia Minamizawa, Patrick Brito, Patrick Gomes, Paulo Tadei, Pedro Henrique, Pedro Kulaif, Pedro Pereira, Peterson Santos, Priscila Santos, Rafael Lino, Reinaldo Silva, Renan Gomes, Renan Moura, Revton Silva, Richard Nixon, Riverfount Riverfount, Robson Maciel, Rodrigo Ferreira, Rodrigo Freire, Rodrigo O'neal, Rodrigo Vaccari, Ronaldo Silva, Rui Jr, Samanta Cicilia, Sandro Mio, Sara Selis, Silvio Xm, Thiago Araujo, Thiago Borges, Thiago Bueno, Thiago Moraes, Tony Dias, Tyrone Damasceno, Victor Wildner, Vinícius Bastos, Vlademir Souza, Vladimir Lemos, Vítor Gomes, Wellington Abreu, Wesley Mendes, Willian Lopes, Willian Rosa, Wilson Duarte, Yuri Fialho, Yury Barros, Érico Andrei

Obrigado você

Graph QL



Uma introdução a tecnologia

GraphQL

GraphQL é um novo padrão pensado para APIs. Uma contrapartida ao já estabelecido padrão REST.

- Criado pelo Facebook em 2012
- Aberto ao público em 2015 (GraphQL Foundation)
- Mantido pela comunidade [https://spec.graphql.org/]
- Agnóstico de linguagem

Veio como uma solução para simplificar o número de endpoints e a quantidade de trafego entre aplicações.

Mas o que é GraphQL?

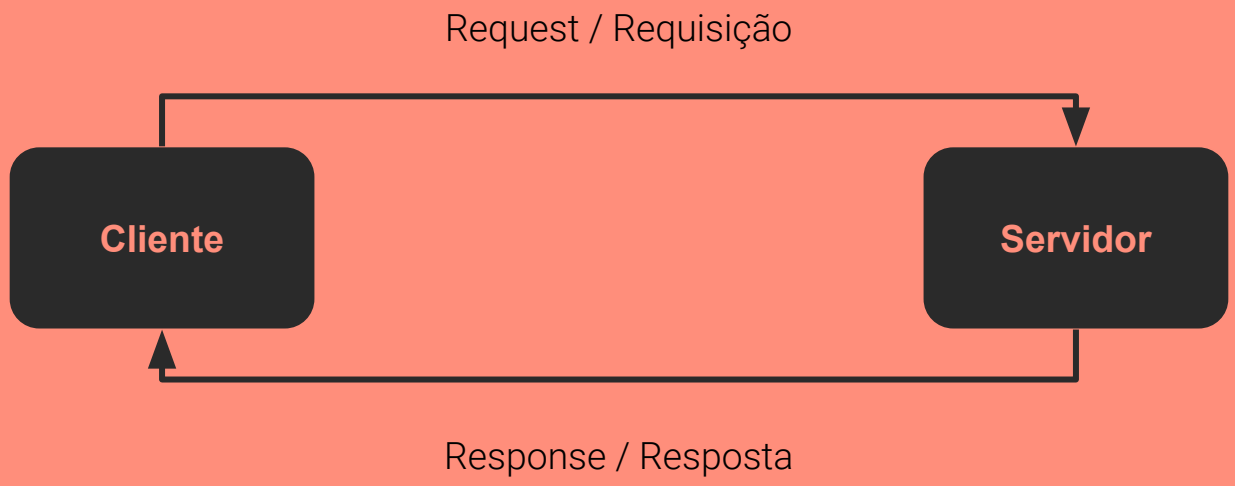
Segundo a especificação o graphQL:

"GraphQL é uma linguagem de consulta projetada para construir aplicativos cliente, fornecendo uma sintaxe e sistema intuitivos e flexíveis para descrever seus requisitos de dados e interações. "

https://spec.graphql.org/October2021/#sec-Overview

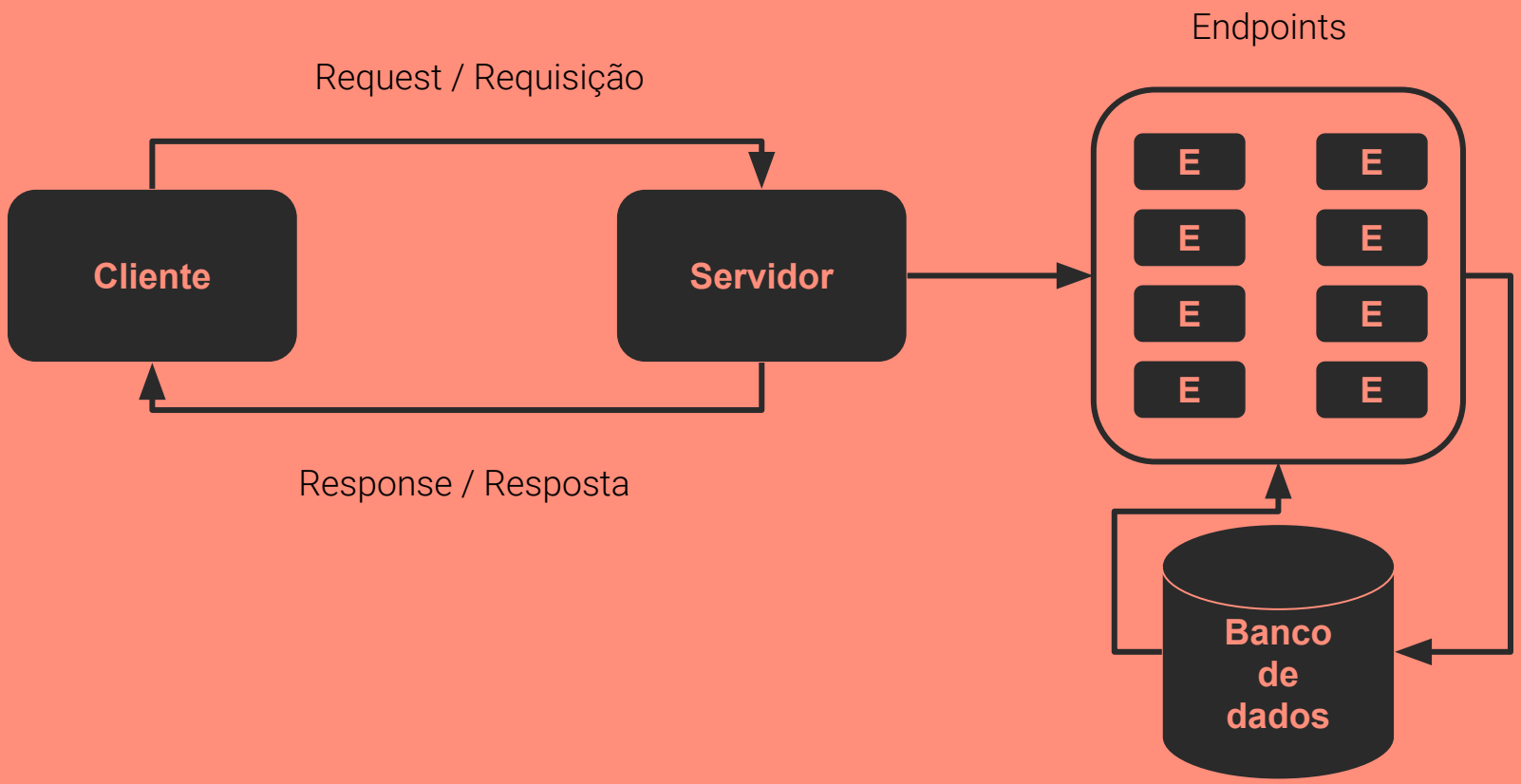
Linguagem de consulta?

Um exemplo de comunicação web



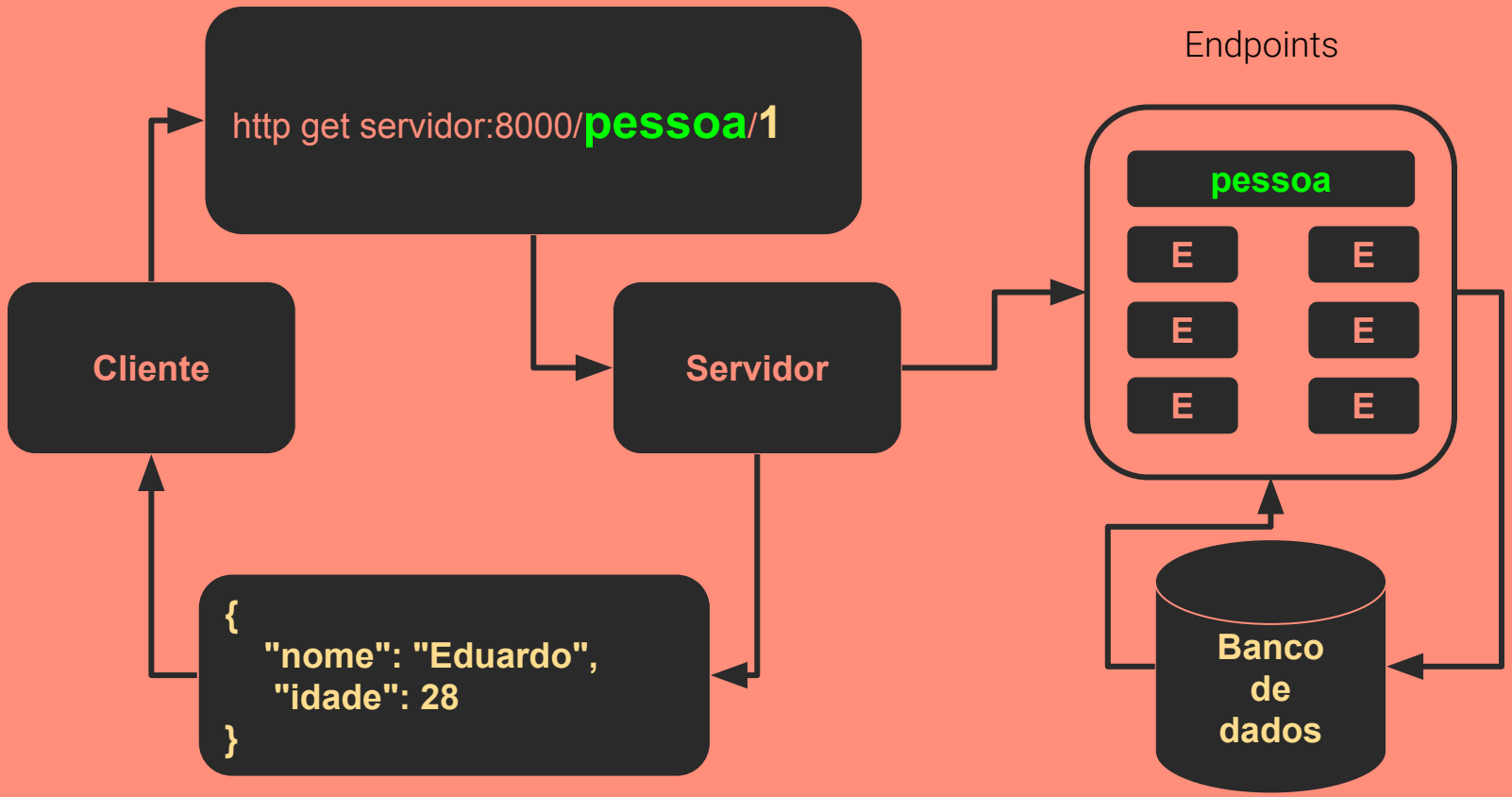
Linguagem de consulta?

Um exemplo de comunicação REST

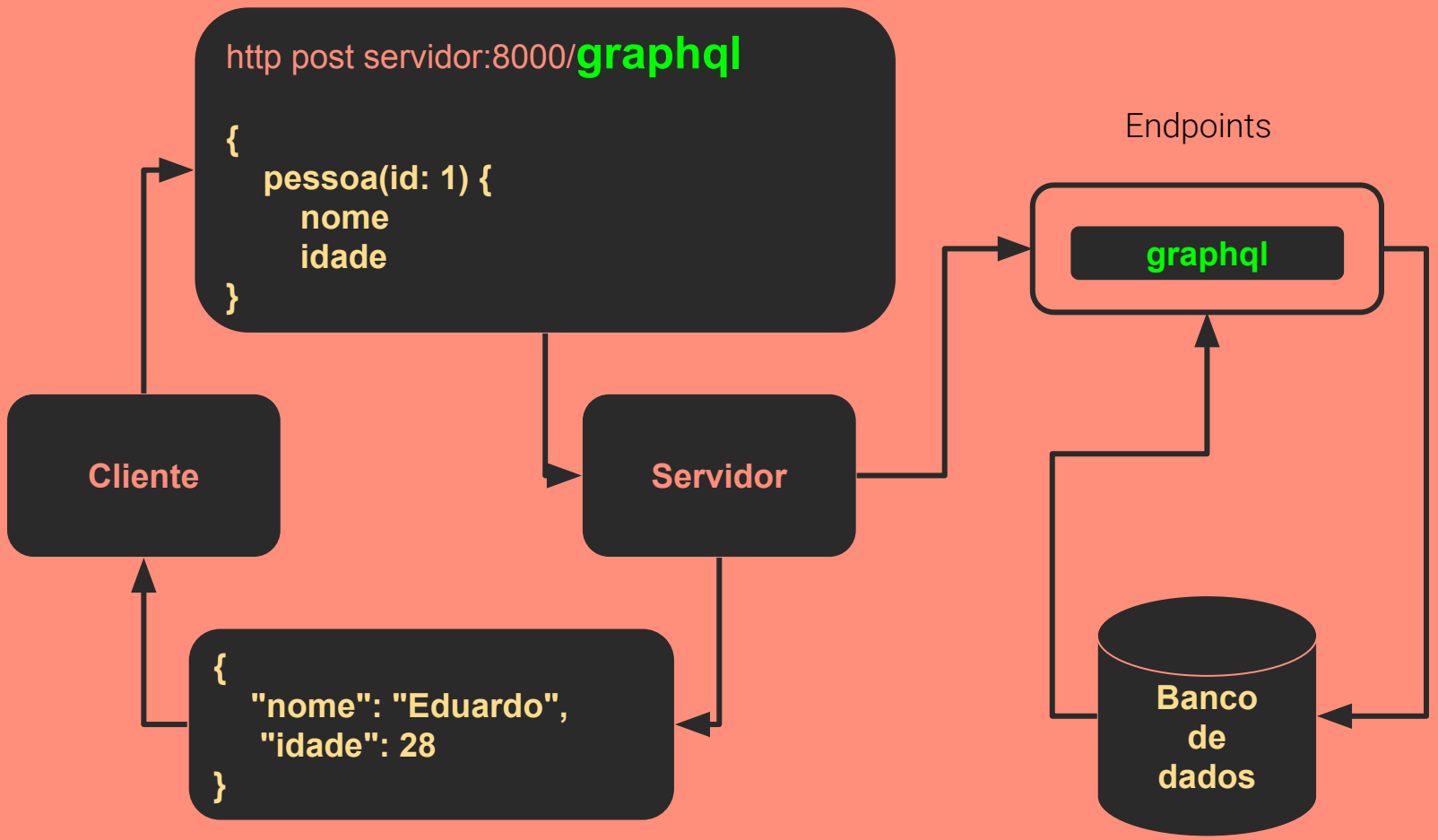


Linguagem de consulta?

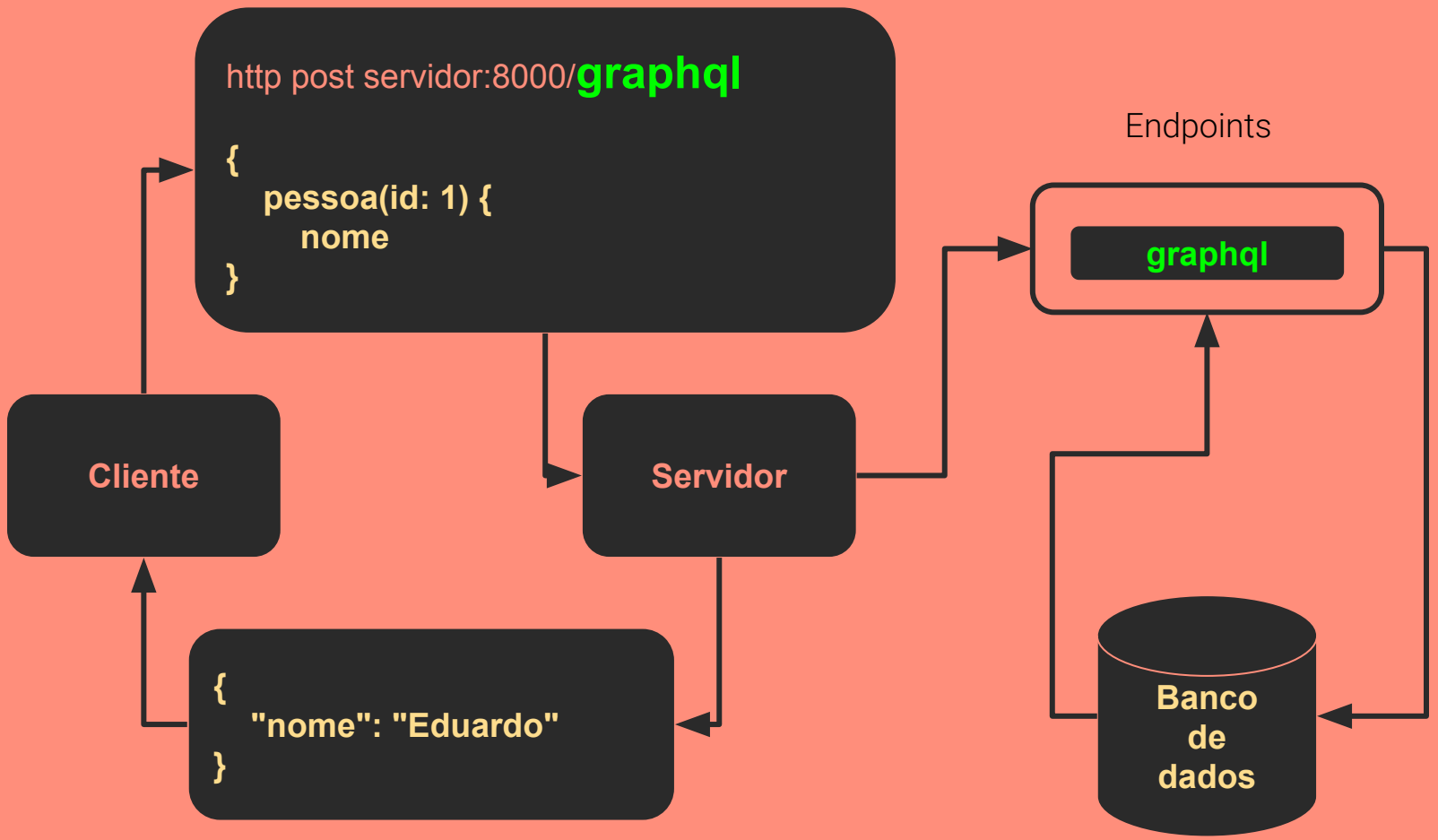
Um exemplo de comunicação REST



Linguagem de consulta?



Linguagem de consulta?



Tá, mas qual a vantagem?

APIs REST apresentam algumas desvantagens:

- Underfetching
- Overfetching
- Muitos endpoints
- Consumo de banda

pip install

- fastapi
- sqlmodel
- uvicorn[standard]

Tá, vamos ver isso na prática

GraphQL em Python

Straw berry

Strawberry

Strawberry é uma biblioteca python para GraphQL, baseada em **dataclasses** (**Live de Python #150**).

Oferece suporte a múltiplos frameworks web:

- Django
- Flask
- FastAPI
- Chalice
- Sanic
- Starlette
- AIOHTTP
- Pydandic (Experimental)
- ....

Tá, mas como o graphQL funciona?

Existem 4 conceitos fundamentais no GraphQL

1. Schema (**SDL**)
2. Queries
3. Mutations
4. Subscripton

pip install strawberry-graphql

Instalar o strawberry pra ficar legal

Tá, mas como o graphQL funciona?

Existem 4 conceitos fundamentais no GraphQL

1. Schema
    - A formação dos dados que vamos aceitar e responder
    - E seus respectivos tipos
2. Queries
    - Como buscar esses dados no banco e devolver para API
3. Mutations
    - Como alterar nossos dados
    - Como criar novos dados
4. Subscripton
    - Ouvir os eventos que acontecem no nosso servidor
    - websocket

Schema

O schema é a definição dos nossos dados ou **Tipos**.

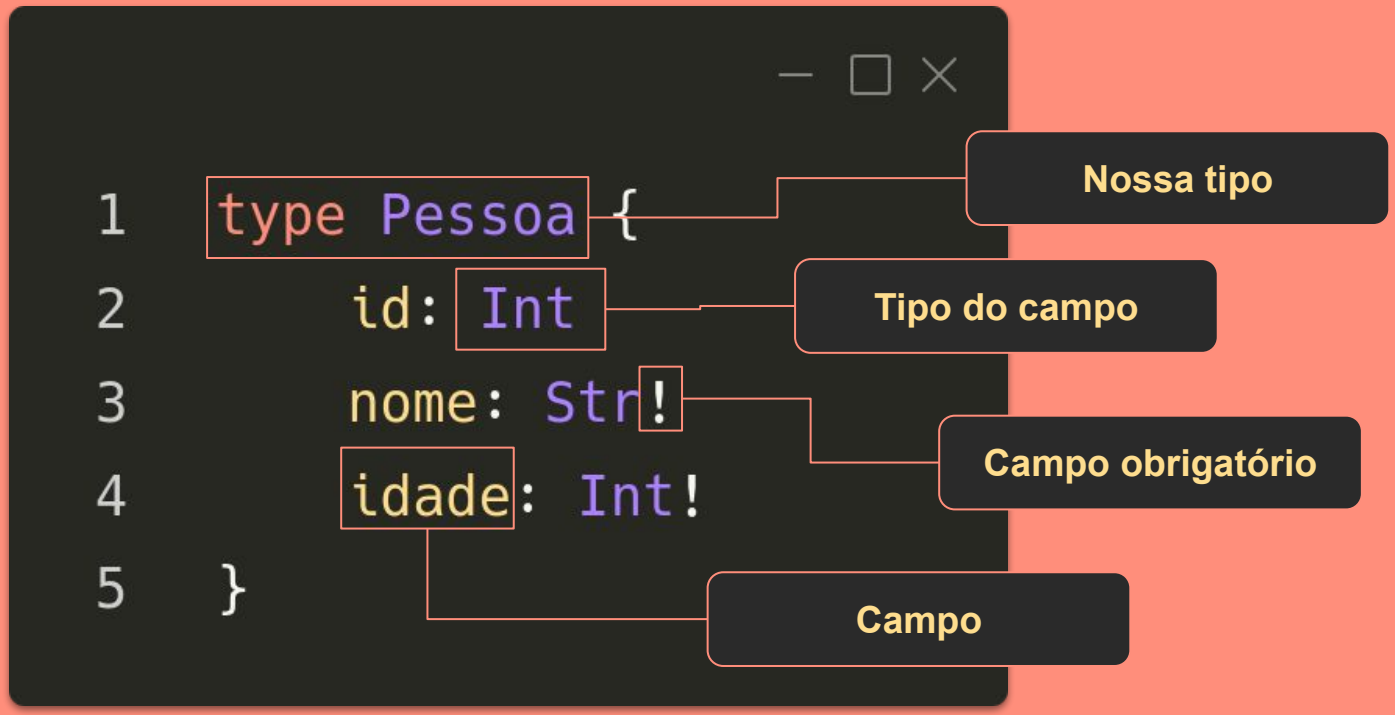
Por exemplo, nossa tabela `Pessoa`, pode ser representada como:

```
type Pessoa {
    id: Int
    nome: Str!
    idade: Int!
}
```

Schema

O schema é a definição dos nossos dados ou **Tipos**.

Por exemplo, nossa tabela `Pessoa`, pode ser representada como:



Um Tipo para fazer as Buscas

Definindo nossa Query

```
1   type Query {
2       allPessoas: [Pessoa!]!
3   }
```

Uma olhada para os dois tipos

```
type Pessoa {
    id: Int
    nome: Str!
    idade: Int!
}
```

```
type Query {
    allPessoas: [Pessoa!]!
}
```

Transcrevendo com Strawberry

```python
import strawberry

@strawberry.type
class Pessoa:
    """
    type Pessoa {
      id: int
      nome: str!
      idade: int!
    }
    """
      id: Optional[int]
      nome: str
      idade: int
```

```python
@strawberry.type
class Query:
    """
    type Query {
      allPessoas: [Pessoa!]!
    }
    """

    @strawberry.field
    def all_pessoas(self) -> list[Pessoa]:
        ...
```

Ultimo passo do Schema

Finalmente, dizemos que nosso schema tem o tipo **Query**, como tipo Raiz (**Root Type**)

```
1   schema {
2     query: Query
3   }
```

Ultimo passo do Schema

```
schema {
  query: Query
}
```

```
schema = strawberry.Schema(query=Query)
```

Uma comparação da especificação com o Python

```
type Pessoa {
  id: int
  nome: str!
  idade: int!
}

type Query{
  allPessoas: [Pessoa!]!
}

schema {
  query: Query
}
```

```python
import strawberry

@strawberry.type
class Pessoa:
  id: Optional[int]
  nome: str
  idade: int


@strawberry.type
class Query:

  @strawberry.field
  def all_pessoas(self) -> list[Pessoa]:
    ...


schema = strawberry.Schema(query=Query)

```

strawberry server nosso_aquivo_python

Hora de rodar <3

Um resumão até agora!

- Vimos os pontos baixos do REST
- Entendemos o básico da implementação de tipos do GraphQL
- Vimos como formar um Root Type para buscas
- Aplicamos isso no python
- Criamos o modelo strawberry

A linguagem das queries

Agora que temos a base da nossa aplicação rodando, podemos entrar no mundo das queries.

```
query {
  allPessoas {
    id
    nome
    idade
  }
}
```

A linguagem das queries

Agora que temos a base da nossa aplicação rodando, podemos entrar no mundo das queries.

http://0.0.0.0:8000/graphql

Bora brincar

Mutations

Como inserir / Alterar dados?

Mutations

Dentro do schema, ainda não definimos como alterar ou inserir dados. Isso é feito com outro root type, as mutations.

```
1  type Mutation{
2    createPessoa: Pessoa
3  }
4
5  schema {
6    query: Query
7    mutation: Mutation
8  }
```

Mutations no strawberry

```
type Mutation{
  createPessoa: Pessoa
}

schema {
  query: Query
  mutation: Mutation
}
```

```python
@strawberry.type
class Mutation:
  create_pessoa: Pessoa = strawberry.field(resolver=create_person)

schema = strawberry.Schema(query=Query, mutation=Mutation)
```

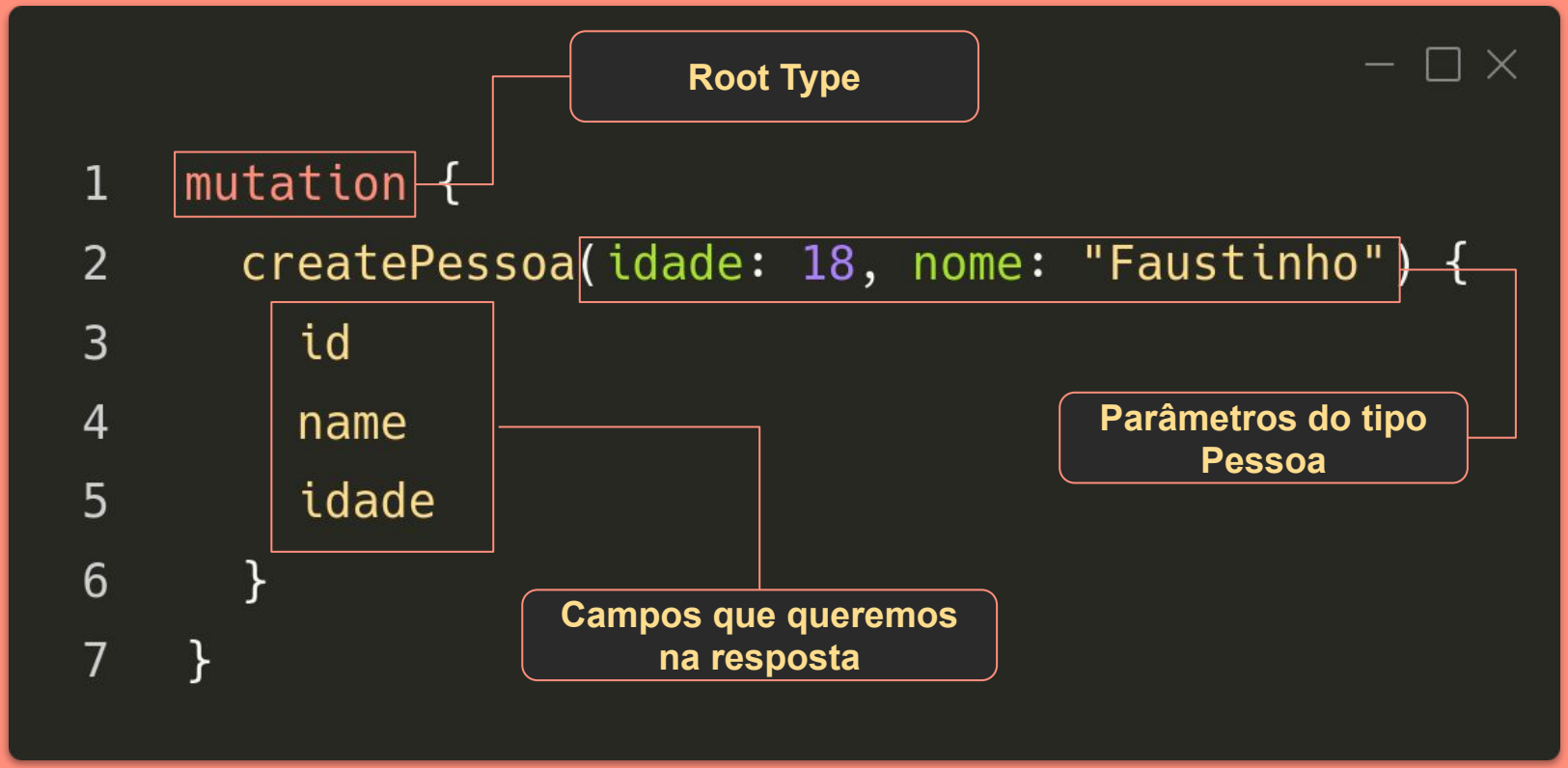
Query para mutations

Tá, mas como vamos inserir os dados?

```
1   mutation {
2     createPessoa(idade: 18, nome: "Faustinho") {
3       id
4       name
5       idade
6     }
7   }
```

Query para mutations

Tá, mas como vamos inserir os dados?



Como filtrar os dados?

Filtros

Bora direto pro código

```python
@strawberry.type
class Query:
  @strawberry.field
  def all_pessoas(self, limit: int = 10) -> list[Pessoa]:
      query = select(Person)
      if limit:
          query = query.limit(limit)

      with Session(engine) as session:
          result = session.execute(query).scalars().all()

      return result
```

Como ficam as queries com filtros?

```
{
    allPessoas(id: 1) {
        idade
        nome
    }
}
```

Integrando

Integrando com o framework web

picpay.me/dunossauro

apoia.se/livedepython

PIX

Ajude o projeto <3