

LEVANDO O MUNDO GRAPHQL PARA O DJANGO COM O GRAPHENE

DANIEL CÂNDIDO
dcs2@cesar.org.br

MARLON CHALEGRE
mcp@cesar.org.br



QUEM SOMOS



DANIEL CÂNDIDO
ENGENHEIRO DE SOFTWARE



MARLON CHALEGRE
ENGENHEIRO DE SOFTWARE





C . E . S . A . R

Centro de Inovação que utiliza Design e Tecnologia da Comunicação e Informação para resolver problemas complexos de um mercado muito diverso.

Em torno de 650 colaboradores espalhados em alguns estados do país.

MAPA DA APRESENTAÇÃO

- INTRODUÇÃO AO GRAPHQL

MAPA DA APRESENTAÇÃO

● INTRODUÇÃO AO GRAPHQL

● OVERVIEW DO GRAPHENE

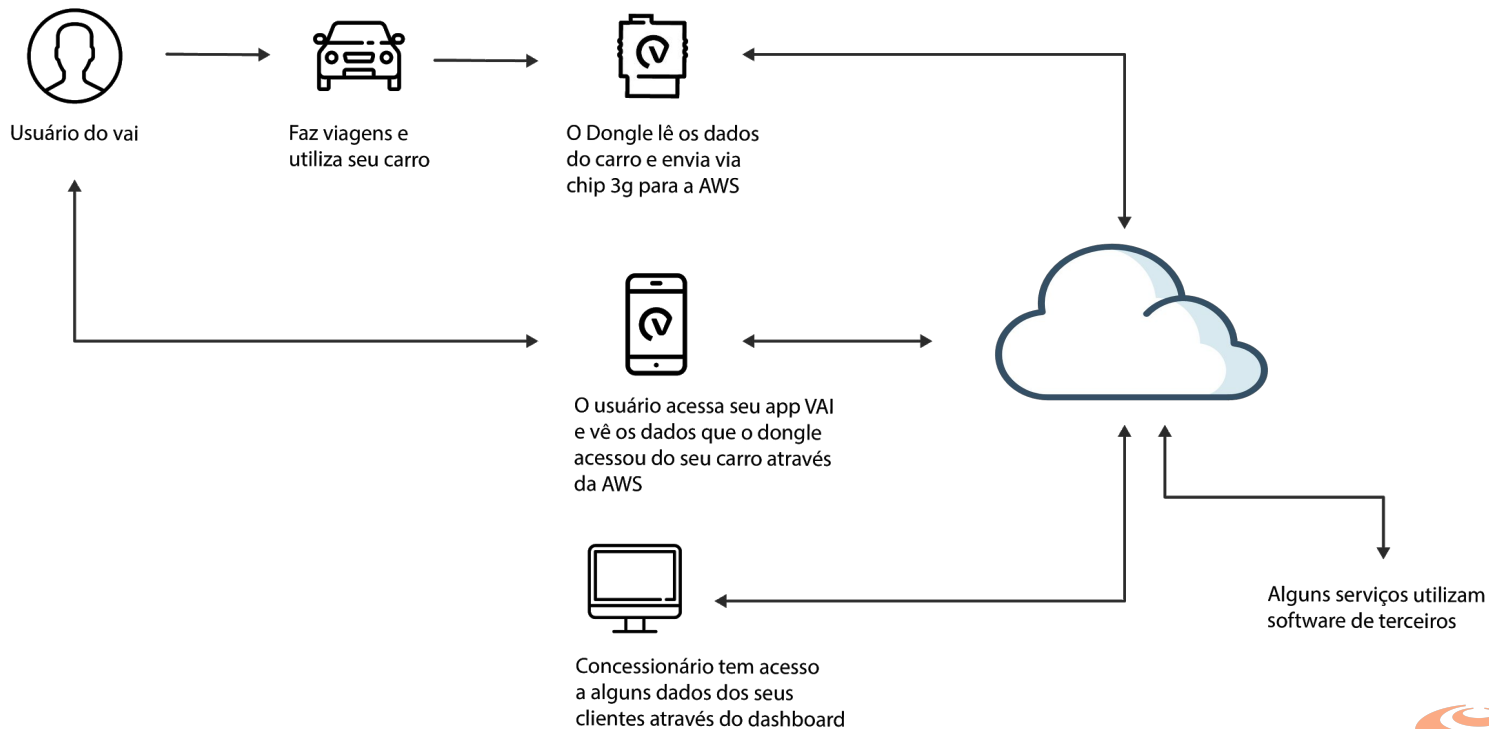
MAPA DA APRESENTAÇÃO

- INTRODUÇÃO AO GRAPHQL
- OVERVIEW DO GRAPHENE
- **SETUP DE SERVIDOR GRAPHQL COM
DJANGO E GRAPHENE**

MAPA DA APRESENTAÇÃO

- INTRODUÇÃO AO GRAPHQL
- OVERVIEW DO GRAPHENE
- SETUP DE SERVIDOR GRAPHQL COM DJANGO E GRAPHENE
- DÚVIDAS, DORES, AFLIÇÕES, FEEDBACKS (?)

O PROJETO



O TIME



DESIGNER



**FULL
STACK**



**DEV
BACKEND**



**DEV
FRONTEND**



DEV/QA



O TIME



DESIGNER



**FULL
STACK**



**DEV
BACKEND**



**DEV
FRONTEND**

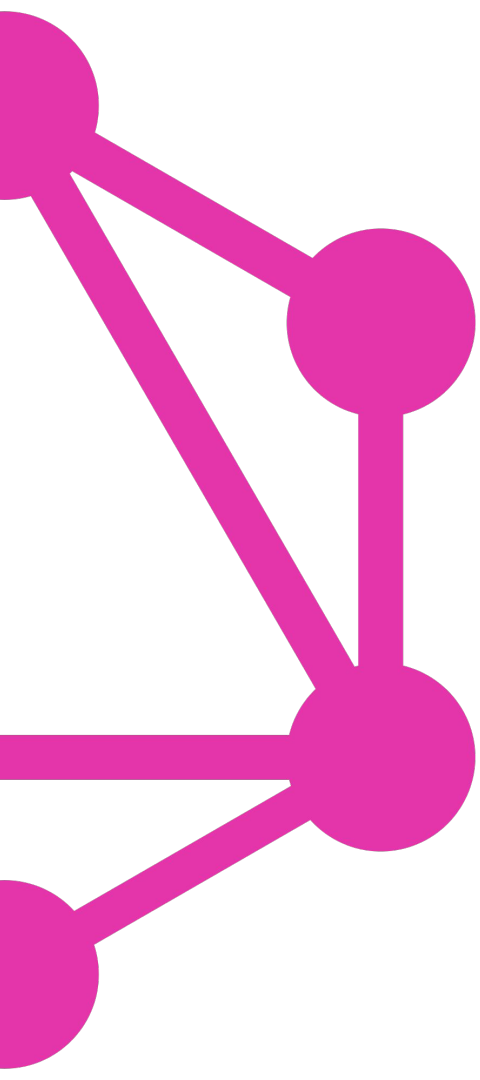


DEV/QA



GRAPHQL
PARA QUEM (NUNCA)
OUVIU FALAR DE
GRAPHQL

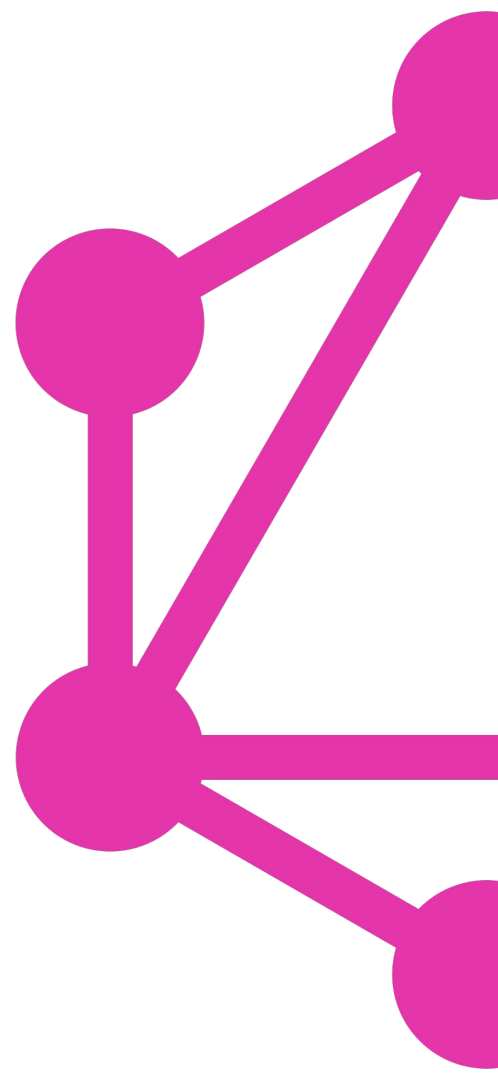




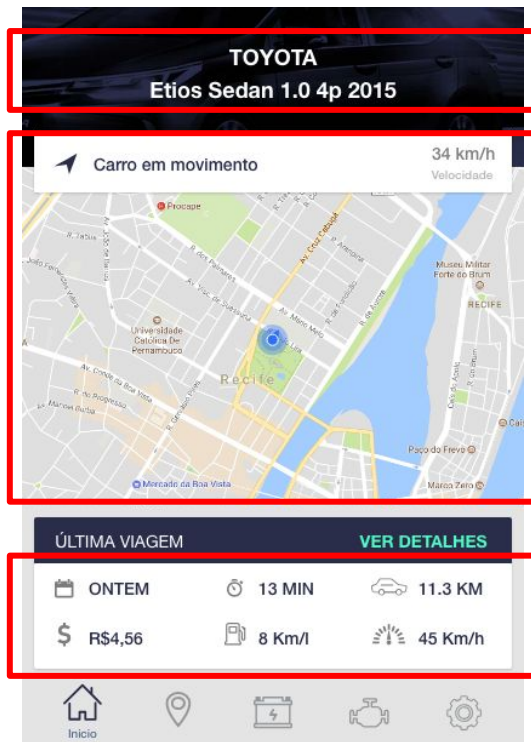
LINGUAGEM DE **CONSULTA**,
DESENVOLVIDA PELO
FACEBOOK EM 2015

ALTERNATIVA AO **REST**
PARA DESENVOLVIMENTO
DE API'S

O **FRONTEND** DECIDE
QUAIS **DADOS** QUER
CONSUMIR DA API



COM REST



GET -> /CARINFO/:CARID

GET -> /LOCATION/:CARID

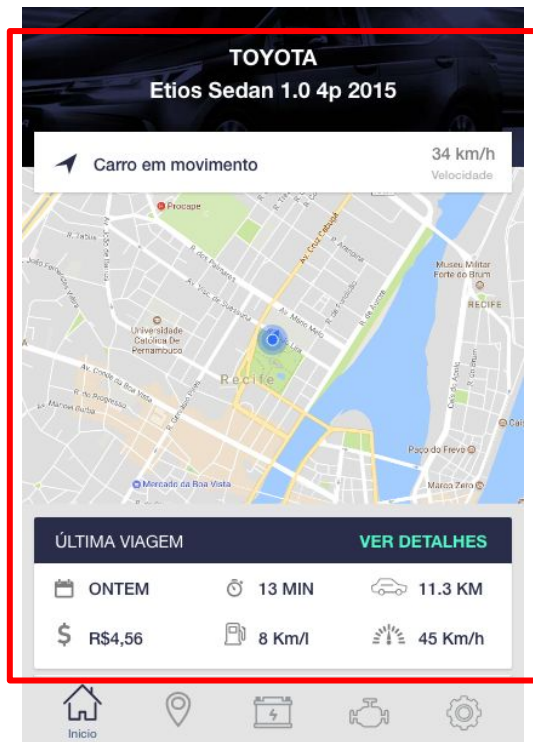
GET -> /TRIPS/:CARID/LAST

```
{
  "montadora": "Toyota",
  "modelo": "Etios Sedan 1.0 4p",
  "ano": 2015
  ...
}

{
  "latitude": -8.0418006,
  "longitude": -35.0787237,
  "velocidade": 34
  ...
}

{
  "data": "2019-07-16 13:50",
  "duracao": 13,
  "distancia": 11300,
  "custo": 456,
  "consumo": 8.0,
  "velocidade": 45
  ...
}
```

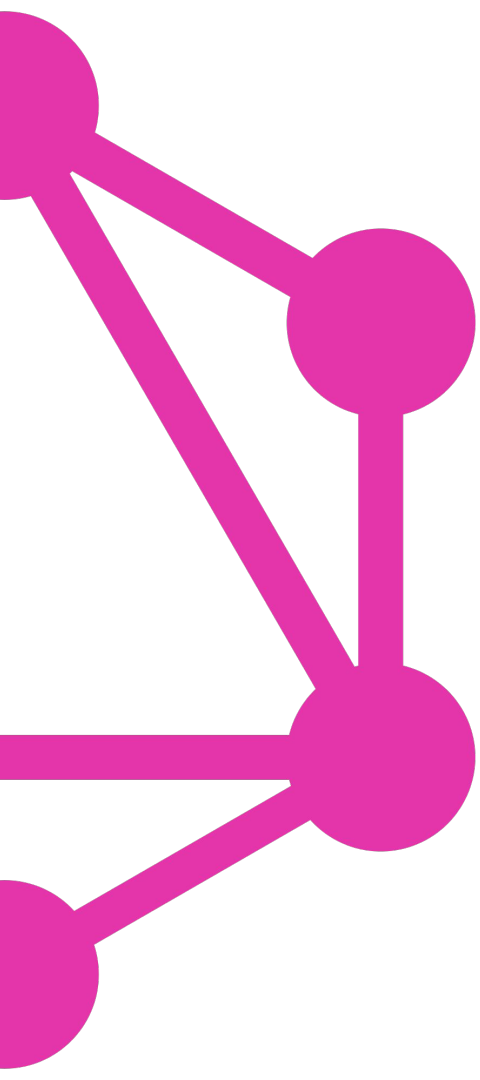
COM GRAPHQL



POST -> /GRAPHQL

```
{  
  car {  
    montadora  
    modelo  
    ano  
    location {  
      latitude  
      longitude  
      velocidade  
    }  
    lastTrip {  
      data  
      duracao  
      distancia  
      custo  
      consumo  
      velocidade  
    }  
  }  
}
```

```
{  
  car {  
    montadora: "Toyota"  
    modelo: "Etios Sedan 1.0 4p"  
    ano: 2015  
    location {  
      latitude: -8.0418006  
      longitude: -35.0787237  
      velocidade: 34  
    }  
    lastTrip {  
      data: "2019-07-16 13:50"  
      duracao: 13  
      distancia: 11300  
      custo: 456  
      consumo: 8.0  
      velocidade: 45  
    }  
  }  
}
```



O **BACKEND** NÃO
PRECISA SE **PREOCUPAR**
QUAIS DADOS SERÃO
CONSUMIDOS POR CADA
CLIENTE



APENAS **DISPONIBILIZA** OS DADOS ATRAVÉS DE UM **SCHEMA**

```
type Car {  
  montadora: String  
  modelo: String  
  ano: Int  
  location: Location  
  trips: [Trip]  
  lastTrip: Trip  
}
```

```
type Location {  
  latitude: Float  
  longitude: Float  
  velocidade: Int  
}
```

```
type Trip {  
  data: Date  
  duracao: Int  
  distancia: Int  
  custo: Int  
  consumo: Float  
  velocidade: Int  
}
```

OVERVIEW DO GRAPHENE





GRAPHENE

BIBLIOTECA MAIS POPULAR

O SCHEMA É DEFINIDO EM PYTHON

INTEGRAÇÃO COM DJANGO E
OUTROS WEB FRAMEWORKS

GRAPHENE

```
import graphene
```

```
class Patron(graphene.ObjectType):
```

```
    id = graphene.ID()
```

```
    name = graphene.String()
```

```
    age = graphene.Int()
```

```
class Query(graphene.ObjectType):
```

```
    patron = graphene.Field(Patron)
```

```
    def resolve_patron(self, info):
```

```
        return Patron(id=1, name="Syrus", age=27)
```

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

```
query = """
```

```
    query something{
```

```
        patron {
```

```
            id
```

```
            name
```

```
            age
```

```
        }
```

```
    }
```

```
"""
```

```
def test_query():
```

```
    result = schema.execute(query)
```

```
    assert not result.errors
```

```
    assert result.data == {"patron": {"id": "1",
```

```
        "name": "Syrus", "age": 27}}
```

```
if __name__ == "__main__":
```

```
    result = schema.execute(query)
```

```
    print(result.data["patron"])
```

INTEGRANDO GRAPHENE COM O DJANGO



PYTHON + DJANGO + GRAPHENE + GRAPHENE_DJANGO



django



Graphene
Python



**TALK IS CHEAP.
DEMONSTRAÇÃO**



SETUP

- GRAPHENE + GRAPHENE_DJANGO

SETUP

- GRAPHENE + GRAPHENE_DJANGO
- SETTINGS DO DJANGO

SETUP

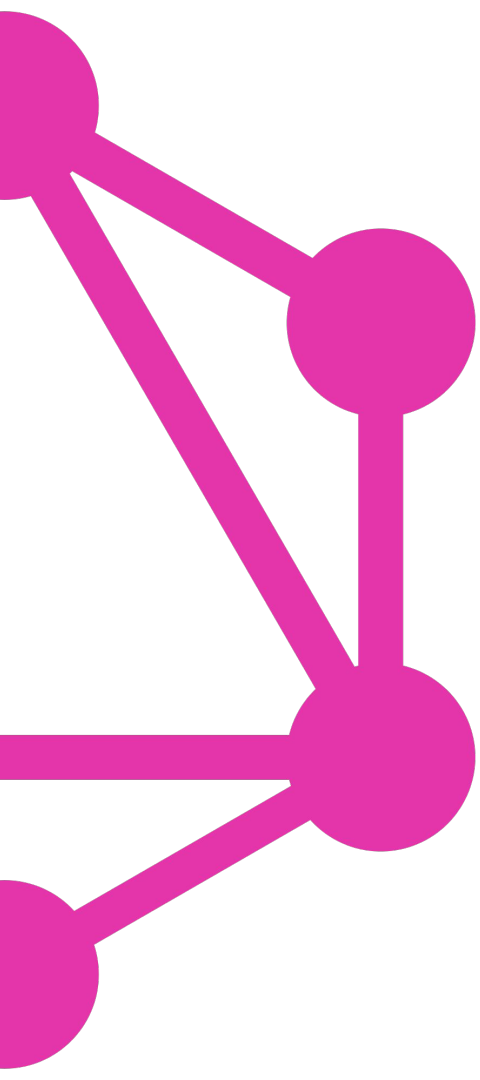
- GRAPHENE + GRAPHENE_DJANGO
- SETTINGS DO DJANGO
- CRIAR O APP + DEFINIR AS URLS

SETUP

- GRAPHENE + GRAPHENE_DJANGO
- SETTINGS DO DJANGO
- CRIAR O APP + DEFINIR AS URLS
- DECLARAR SCHEMA

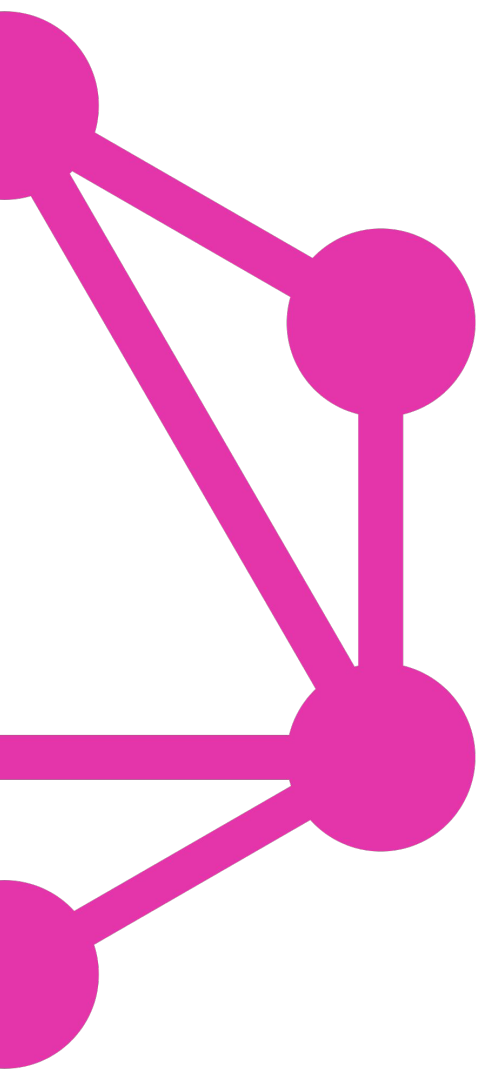
SETUP

- GRAPHENE + GRAPHENE_DJANGO
- SETTINGS DO DJANGO
- CRIAR O APP + DEFINIR AS URLS
- DECLARAR SCHEMA
- CRIAR QUERIES E MUTATIONS



COM GRAPHQL NEM TUDO
SÃO FLORES

O **BACKEND** PRECISA SE
PREOCUPAR COMO OS
DADOS SERÃO **BUSCADOS**
INTERNAMENTE



```
query maliciousQuery {  
  thread(id: "some-id") {  
    messages(first: 99999) {  
      thread {  
        messages(first: 99999) {  
          thread {  
            messages(first: 99999) {  
              thread {  
                # ...repeat times 10000...  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

DÚVIDAS, APLICABILIDADE DE PROJETO, EXPERIÊNCIAS?



VALEU!



TEMOS VAGAS!
cesar.org.br



CÓDIGO EM:
...

DANIEL CÂNDIDO
dcs2@cesar.org.br

MARLON CHALEGRE
mcp@cesar.org.br

vai.com.vc

