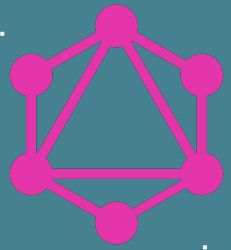


Introdução a GraphQL com Django

Hello!

Eu sou Sarah Raquel

Você pode me encontrar em @sarahhaquel

*API's se tornaram componentes ubíquos em infraestruturas de software. Em resumo, uma API define **como** um cliente pode pegar dados de um servidor.*

1

Revisando REST

- Stateless
- Interface Uniforme
- Cliente-Servidor

grupy.rn

/users

Lista de Inscritos

João

Laura

José

Maria

Lucas

grupy.rn

/users/:id
/users/:id/talks
/users/:id/meetups

Detalhe do Inscrito

João
1 palestra
5 meetups

```
{
    “name”: “João”,
    “nickname”: “Jão”,
    “interests”: “Python, IoT”,
    “job”: “Júnior developer”,
    “img_url”: “https://imgurl.com”,
    “profile_description”: “...”
    // more data
}
```

grupy.rn

/users/:id
/users/:id/talks
/users/:id/meetups

Detalhe do Inscrito

João
1 palestra
5 meetups

Várias requisições

Código mais complexo

Dados não usados

E quanto a custom endpoints?

1 endpoint por view

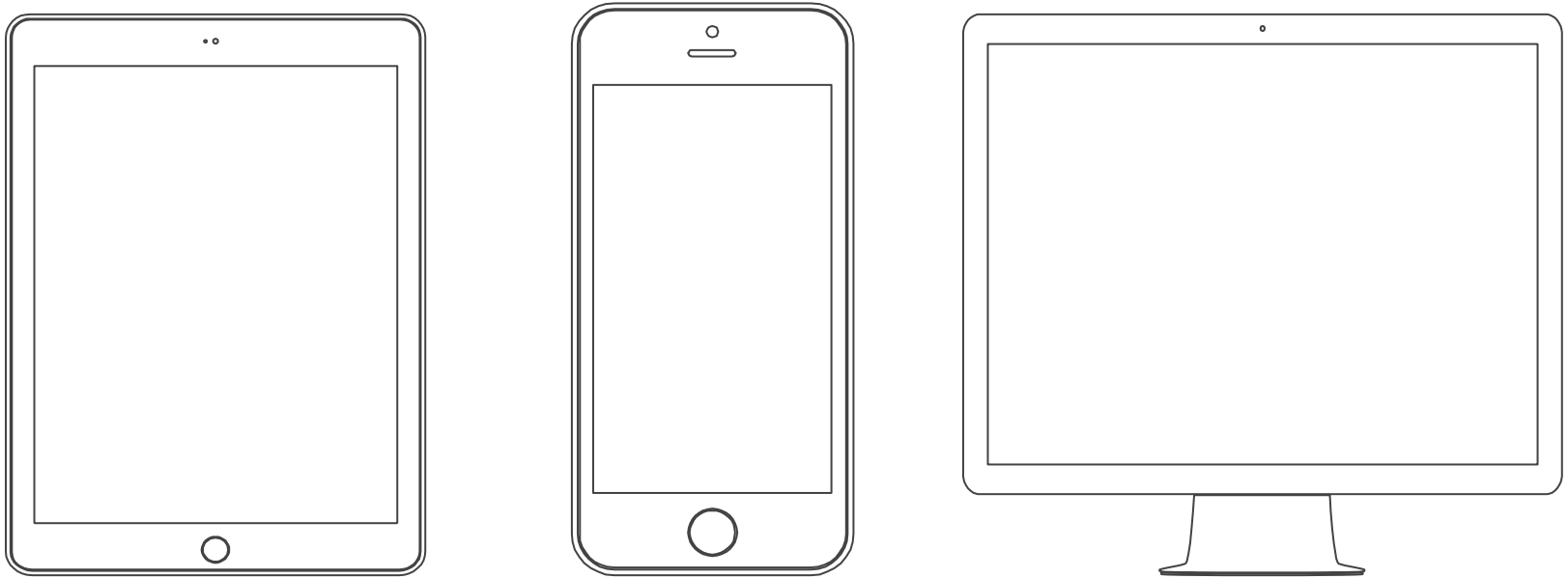
E quanto a custom endpoints?

1 endpoint por view

Código mais simples
Menos requests

x

Maior acoplamento
Vários clientes



grupy.rn

/users
/users/:id/talks
/users/:id/meetups

Lista de Inscritos

João
1 palestra
5 meetups

Laura
3 palestras
10 meetups

José
Nenhuma palestra
2 meetups

Maria
1 palestra
5 meetups

Lucas
Nenhuma palestra
Nenhum meetup

GraphQL

- QL = “Query Language”
- Open Source
- Database agnostic
- Desenvolvimento mais rápido
- POST
- Um endpoint


coursera


SERVERLESS


Product Hunt


yelp


METEOR

Conceitos principais

- SDL Schema Definition Language

```
type Person {
  name: String!
  age: Int!
}
```

```
type Post {
  title: String!
  author: Person!
}
```

```
type Person {
  name: String!
  age: Int!
  posts: [Post!]!
}
```

Rodando queries

```
{
    allPersons {
        name
  }
}
```

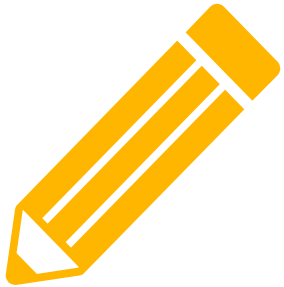
```
{
  "allPersons": [
    { "name": "Johnny" },
    { "name": "Sarah" },
    { "name": "Alice" }
  ]
}
```

```
{
    allPersons {
        name
        age
        posts {
            title
        }
  }
}
```

```
{
    allPersons(last: 2) {
        name
    }
}
```

```
{
    "data": {
        "allPersons": [
            { "name": "Sarah" },
            {  "name": "Alice" }
        ]
    }
}
```

Mutações



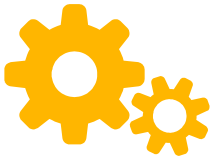
```
mutation {
    createPerson(name: "Bob", age: 36) {
        name
        age
    }
}
```

```
"createPerson": {
  "name": "Bob",
  "age": 36,
}
```

Getting started

https://github.com/sarahraquel/django-blog

mysite/settings.py



```
pip install graphene_django
```

Add `'graphene_django'` ao seu settings.py

mysite/settings.py

```
GRAPHENE = {
    # Where your Graphene schema lives
    'SCHEMA': 'mysite.schema.schema',
}
```

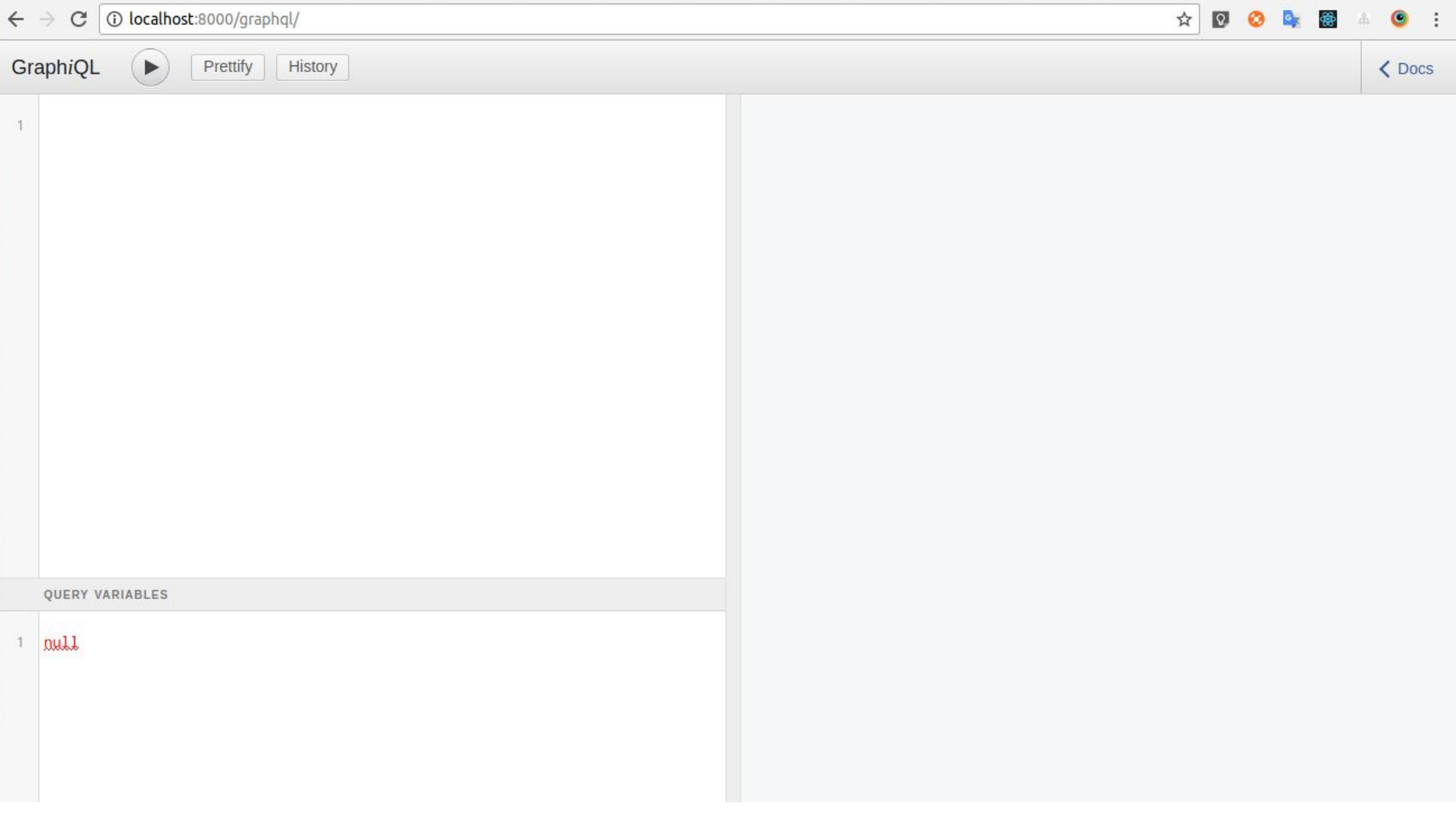
mysite/urls.py

```
from graphene_django.views import GraphQLView


urlpatterns = [
    ...
    url(r'^graphql', GraphQLView.as_view(graphiql=True)),
]
```

mysite/urls.py

```
from graphene_django.views import GraphQLView


urlpatterns = [
    ...
    url(r'^graphql', GraphQLView.as_view(graphiql=True)),
]
```

Modo interativo


localhost:8000/graphql/
GraphiQL
Prettify
History
Docs
1
QUERY VARIABLES
1
null

blog/models/post.py
blog/models/comment.py

```
class Post(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=80)
    body = models.TextField()
    pub_date = models.DateTimeField('date published', auto_now_add=True)

class Comment(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    body = models.TextField()
    pub_date = models.DateTimeField('date published', auto_now_add=True)
```

blog/schema.py

```
import graphene
from graphene_django.types import DjangoObjectType
from blog.models.post import Post
from blog.models.comment import Comment

class PostType(DjangoObjectType):
    class Meta:
        model = Post

class CommentType(DjangoObjectType):
    class Meta:
        model = Comment
```

blog/schema.py

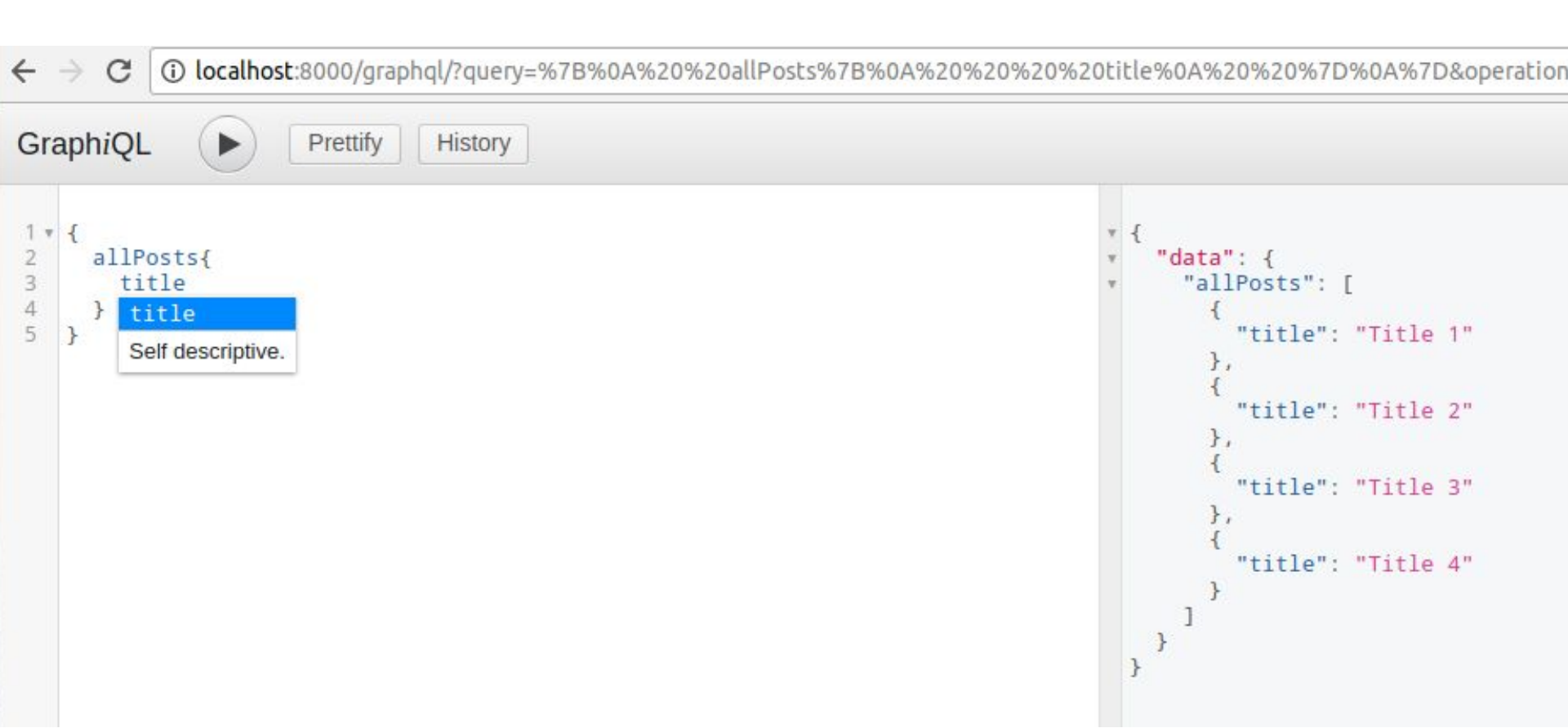
```
class Query(object):
    all_posts = graphene.List(PostType)
    all_comments = graphene.List(CommentType)
    post = graphene.Field(PostType, id=graphene.Int(),
                                title=graphene.String())


    def resolve_all_posts(self, info, **kwargs):
        return Post.objects.all()


    def resolve_all_comments(self, info, **kwargs):
        return Comment.objects.select_related('post').all()
```

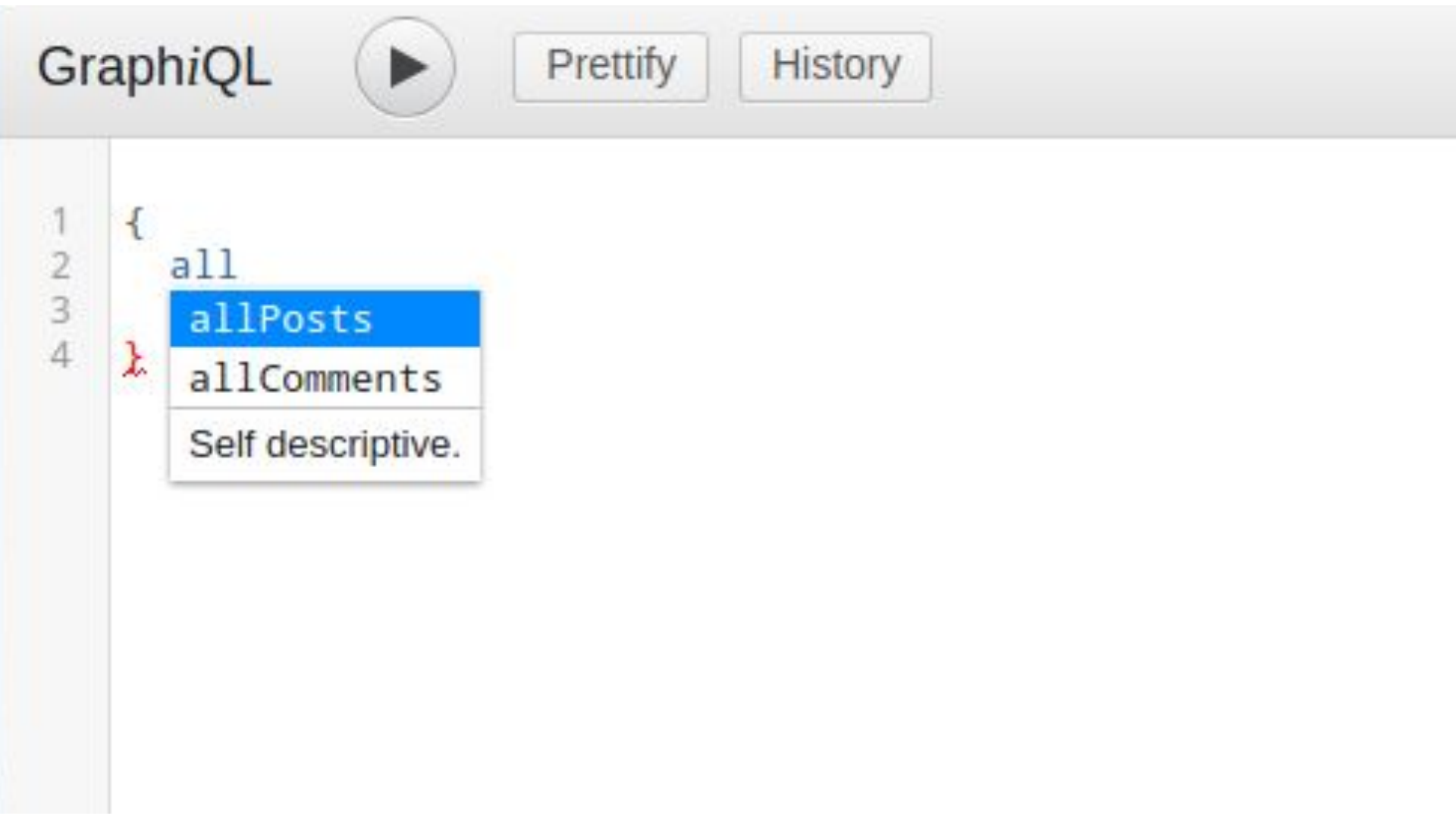
blog/schema.py

```
def resolve_post(self, info, **kwargs):
        id = kwargs.get('id')
        title = kwargs.get('title')


        if id is not None:
            return Post.objects.get(pk=id)


        if title is not None:
            return Post.objects.get(title__contains=title)


        return None
```

mysite/schema.py

```
import graphene
import blog.schema

class Query(blog.schema.Query, graphene.ObjectType):
    # This class will inherit from multiple Queries
    # as we begin to add more apps to our project
    pass


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


localhost:8000/graphql/?query=%7B%0A%20%20allPosts%7B%0A%20%20%20%20title%0A%20%20%7D%0A%7D&operation
GraphiQL
Prettify
History
{
allPosts{
title
}
}
title
Self descriptive.
{
"data": {
"allPosts": [
{
"title": "Title 1"
},
{
"title": "Title 2"
},
{
"title": "Title 3"
},
{
"title": "Title 4"
}
]
}
}


GraphiQL
Prettify
History
1
2
3
4
{
all
allPosts
allComments
Self descriptive.


GraphiQL
Prettify
History
{
post(title: "1"){
title
}
}

Como escolher

Obrigada!

Perguntas?

Você pode me encontrar em

@sarahhaquel

@sarahraquel