```python
class Query(graphene.ObjectType):
    hello = graphene.String()
    user = relay.Node.Field(UserNode)
    users = DjangoFilterConnectionField(UserNode)

    course = relay.Node.Field(CourseNode)
    courses = DjangoFilterConnectionField(CourseNode)

    me = graphene.Field(UserNode)

    debug = graphene.Field(DjangoDebug, name='__debug')

    def resolve_me(self, info):
        return UserNode.get_node(info, info.context.user.id)

    def resolve_hello(self, info, **kwargs):
        return 'world'


class Mutations(graphene.ObjectType):
    create_course = CreateCourse.Field()
    update_course = UpdateCourse.Field()
    update_profile = UpdateProfile.Field()


class Subscription(graphene.ObjectType):

    count_seconds = graphene.Int(up_to=graphene.Int())
    sub_user = graphene.Field(
        UserNode, description='subscribe to updated product', username=graphene.String())

    def resolve_count_seconds(root, info, up_to=5):
        return Observable.interval(1000)\
                         .map(lambda i: "{0}".format(i))\
                         .take_while(lambda i: int(i) <= up_to)

    def resolve_sub_user(root, info, *args, **kwargs):
        username = kwargs.get('username')

        def get_object(observer):
            instance = User.objects.get(username=username)
            return instance
        return Observable.interval(1000) \
            .map(lambda s: get_object(s)) \
            .share()
```

CREANDO UN API GRAPHQL CON

CARLOS MARTINEZ

Desarrollador Backend en comparamejor

twitter/carlosmart626

github/carlosmart626

https://carlosmart.co

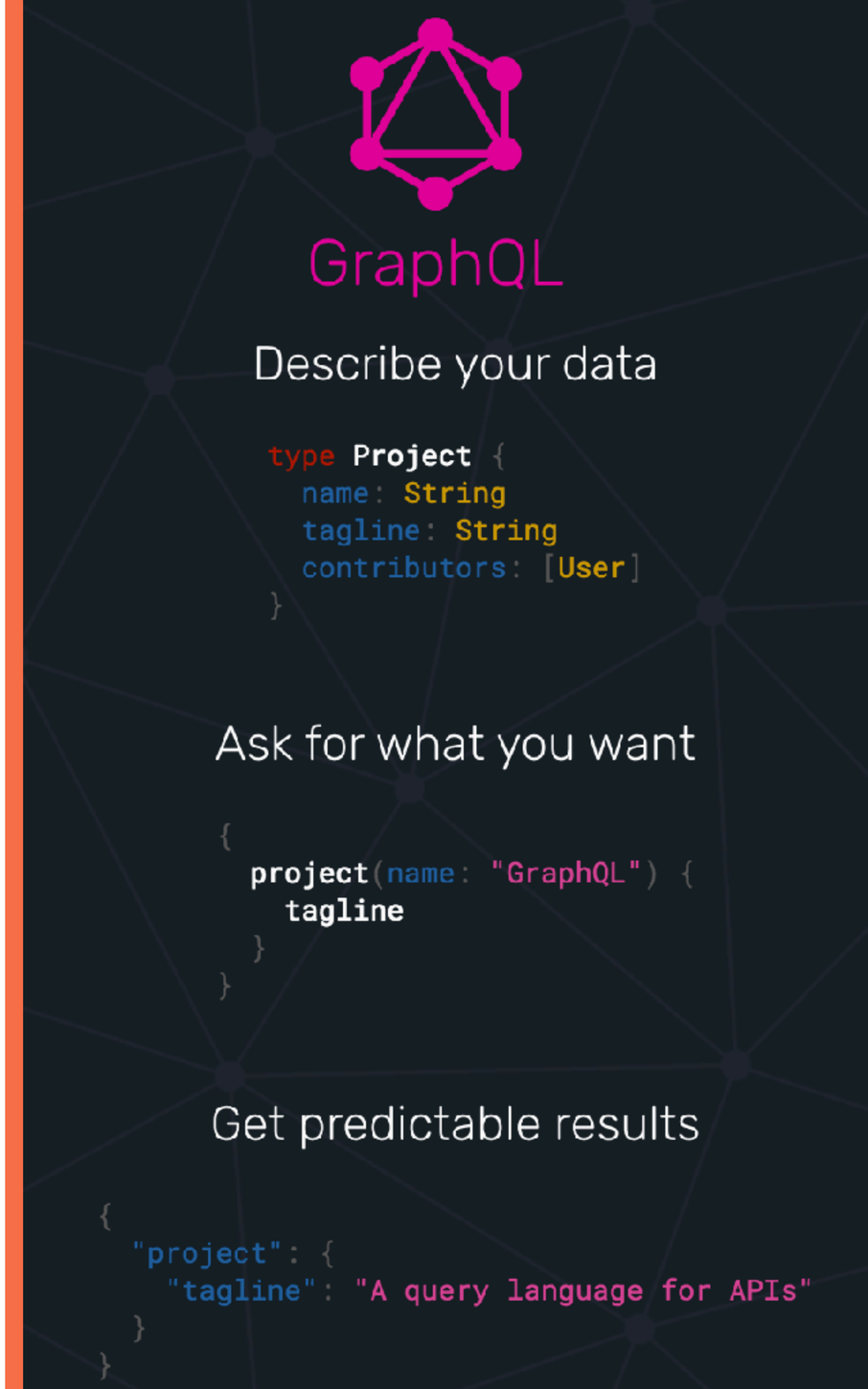
```
class Query(gr
    hello = gr
    user = rel
    users = Dj

    course = r
    courses =

    me = graph

    debug = gr

    def resolv
        return

    def resolv
        return

class Mutation
    create_cou
    update_cou
    update_pro

class Subscrip

    count_seco
    sub_user =
        UserNo

    def resolv
        return

    def resolv
        userna

        def ge
            in
            re
        return
            .m
            .s
```

QUE ES GRAPHQL?

QUE ES GRAPHQL?

GraphQL es un lenguaje de "query" de datos para tu API para ejecutar consultas usando un sistema de tipos definidos para tus datos, fue desarrollado por Facebook en 2012 y fue liberado públicamente en 2015. Provee una alternativa a REST.

Un servicio de GraphQL se crea definiendo tipos y campos en esos tipos, luego proveer funciones para cada uno de esos tipos definidos.



```python
from courses.mutations import CreateCourse, UpdateCourse


class Query(graphene.ObjectType):
    hello = graphene.String()
    user = relay.Node.Field(UserNode)
    users = DjangoFilterConnectionField(UserNode)

    course = relay.Node.Field(CourseNode)
    courses = DjangoFilterConnectionField(CourseNode)

    me = graphene.Field(UserNode)

    debug = graphene.Field(DjangoDebug, name='__debug')

    def resolve_me(self, info):
        return UserNode.get_node(info, info.context.user.id)

    def resolve_hello(self, info, **kwargs):
        return 'world'


class Mutations(graphene.ObjectType):
    create_course = CreateCourse.Field()
    update_course = UpdateCourse.Field()
    update_profile = UpdateProfile.Field()


class Subscription(graphene.ObjectType):

    count_seconds = graphene.Int(up_to=graphene.Int())
    sub_user = graphene.Field(
        UserNode, description='subscribe to updated product', username=graphene.String())

    def resolve_count_seconds(root, info, up_to=5):
        return Observable.interval(1000)\
                         .map(lambda i: "{0}".format(i))\
                         .take_while(lambda i: int(i) <= up_to)

    def resolve_sub_user(root, info, *args, **kwargs):
        username = kwargs.get('username')

        def get_object(observer):
```

ESTRUCTURA GRAPHQL

ObjectTypes, Enum, Scalars

Nodos

Mutations

Schemas

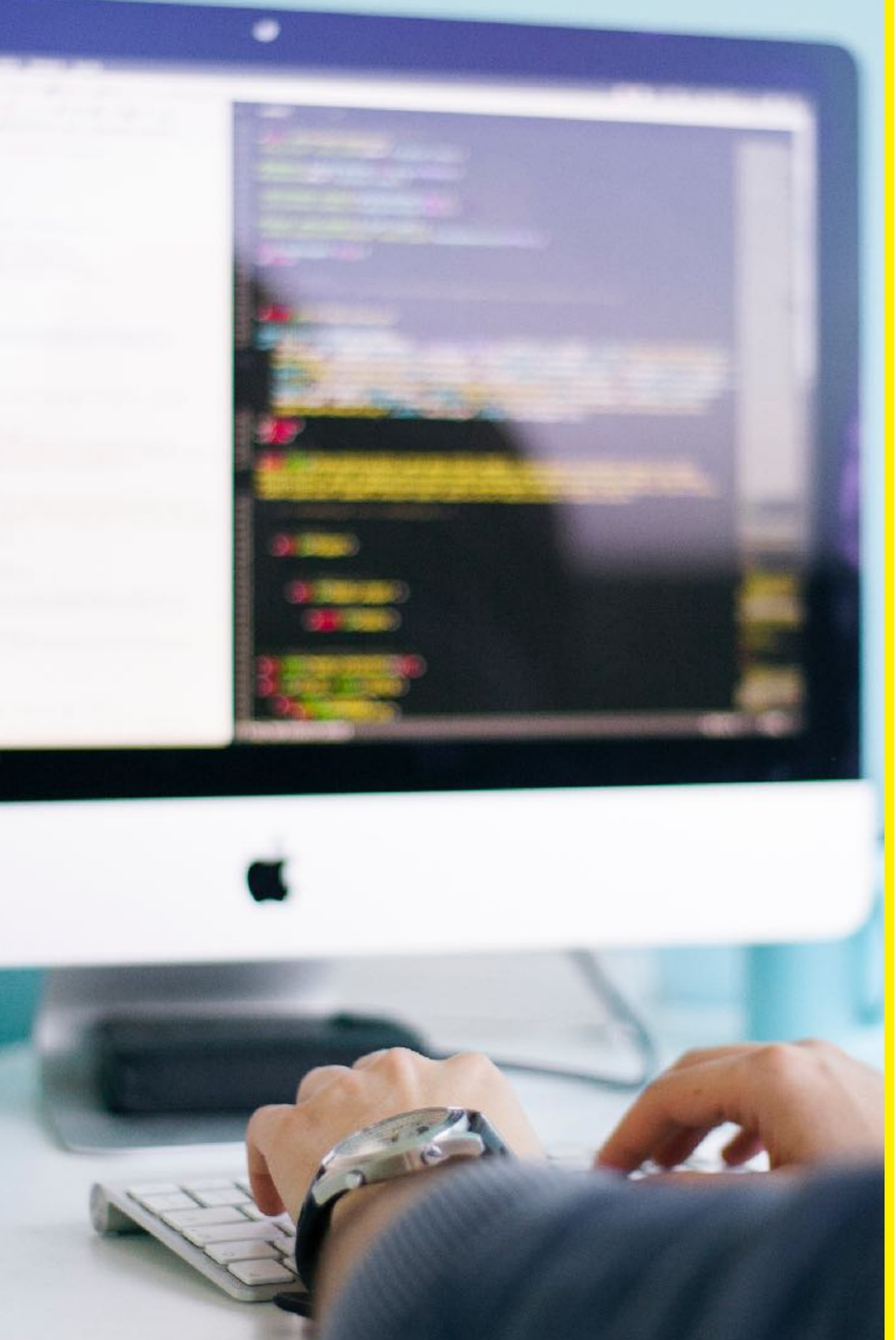
Un solo endpoint

GET, POST

POR QUÉ GRAPHQL?

GitHub

https://github.com/CarlosMart626/graphql-pycon.co2018



MANOS A LA OBRA

Graphene

Python

```
pip install graphene-django
```

TIPOS Y NODOS

```python
class Course(models.Model):
    title = models.CharField(max_length=150)
    description = models.TextField(max_length=500)
    teacher = models.ForeignKey(User, on_delete=models.CASCADE, related_name='courses')
    is_active = models.BooleanField(default=True)
    enrolled_students = models.ManyToManyField(User, blank=True)

    def __str__(self):
        return f'{self.title}'
```

```python
class CourseNode(DjangoObjectType):

    class Meta:
        model = Course
        filter_fields = {
            'title': ['exact', 'icontains', 'istartswith'],
            'teacher': ['exact'],
        }
        interfaces = (relay.Node, )
```

MUTACIONES [INPUT TYPES]

```python
class CourseInput(graphene.InputObjectType):
    title = graphene.String(required=True)
    description = graphene.String()
    teacher_username = graphene.Int(required=True)
```

MUTACIONES [INPUT TYPES]

```python
class CourseInput(graphene.InputObjectType):
    title = graphene.String(required=True)
    description = graphene.String()
    teacher_username = graphene.Int(required=True)


class CourseUpdateInput(CourseInput):
    id = graphene.Int(required=True)
    is_active = graphene.Boolean()
```

MUTACIONES

```python
class CreateCourse(graphene.Mutation):

    class Arguments:
        course_data = CourseInput()

    ok = graphene.Boolean()
    course = graphene.Field(CourseNode)

    def mutate(self, info, course_data):
        try:
            username = profile_data.pop('teacher_username')
            user = User.objects.get(username=username)
            course = Course.objects.create(
                title=title,
                description=description,
                teacher=user)
            ok = True

        except Exception as e:
            ok = False
            profile = None

        return CreateCourse(ok=ok, course=course)
```

MUTACIONES

```python
class CreateCourse(graphene.Mutation):

    class Arguments:
        title = graphene.String(required=True)
        description = graphene.String()
        teacher_username = graphene.Int(required=True)

    ok = graphene.Boolean()
    course = graphene.Field(CourseNode)

    def mutate(self, info, course_data):
        try:
            username = profile_data.pop('teacher_username')
            user = User.objects.get(username=username)
            course = Course.objects.create(
                title=title,
                description=description,
                teacher=user)
            ok = True

        except Exception as e:
            ok = False
            profile = None

        return CreateCourse(ok=ok, course=course)
```

```python
class CreateCourse(graphene.Mutation):

    class Arguments:
        course_data = CourseInput()

    ok = graphene.Boolean()
    course = graphene.Field(CourseNode)

    def mutate(self, info, course_data):
        try:
            username = profile_data.pop('teacher_username')
            user = User.objects.get(username=username)
            course = Course.objects.create(
                title=title,
                description=description,
                teacher=user)
            ok = True

        except Exception as e:
            ok = False
            profile = None

        return CreateCourse(ok=ok, course=course)
```

MUTACIONES

```python
@classmethod
def mutate(cls, info, **input):
    files = info.context.FILES
    # Process files
    return UploadFile(success=True)
```

SCHEMA

```python
class Query(graphene.ObjectType):
    hello = graphene.String()
    user = relay.Node.Field(UserNode)
    users = DjangoFilterConnectionField(UserNode)

    course = relay.Node.Field(CourseNode)
    courses = DjangoFilterConnectionField(CourseNode)

    me = graphene.Field(UserNode)

    debug = graphene.Field(DjangoDebug, name='__debug')

    def resolve_me(self, info):
        return UserNode.get_node(info, info.context.user.id)

    def resolve_hello(self, info, **kwargs):
        return 'world'


class Mutations(graphene.ObjectType):
    create_course = CreateCourse.Field()
    update_course = UpdateCourse.Field()
    update_profile = UpdateProfile.Field()
```

SCHEMA [RESOLVERS]

```
token = graphene.String()

def resolve_token(self, info, **args):
    if self.id != info.context.user.id and \
        not getattr(self, 'is_current_user', False):
            return None

    jwt_payload_handler = api_settings.JWT_PAYLOAD_HANDLER
    jwt_encode_handler = api_settings.JWT_ENCODE_HANDLER

    payload = jwt_payload_handler(self)
    token = jwt_encode_handler(payload)

    return token
```

SCHEMA

```
schema = graphene.Schema(
    query=Query,
    mutation=Mutations,
    subscription=Subscription)
```

```
url(r'^graphql',
    csrf_exempt(
        GraphQLView.as_view(
            schema=auth_schema, graphiql=True))),
```

GRAPHENE SETTINGS

```
GRAPHENE = {
    'SCHEMA': 'djcourses.schema.schema',
}
```

AUTENTICACIÓN

```python
from jwt_auth.mixins import JSONWebTokenAuthMixin


class AuthGraphQLView(JSONWebTokenAuthMixin, GraphQLView):
    pass
```

djangorestframework-jwt

django-jwt-auth

```python
url(r'^graphql',
    csrf_exempt(
        AuthGraphQLView.as_view(
            schema=schema, graphiql=True))),
```

TESTING GRAPHQL?

TESTING

```
query_me = '''
    query Me{
        me{
            email
            firstName
            lastName
            profile{
            bio
            birthDate
            }
        }
    }
'''
```

```
def test_query():
    request = Mock()

    result = schema.execute(
        query_external_sources,
        context_value=request
    )
    assert result.data['me'] is not None
    assert result.data['me']['email'] == 'me@carlosmart.co'
```

SUBSCRIPTIONS

DJANGO CHANNELS!!!

SUBSCRIPTIONS

```
CHANNELS_WS_PROTOCOLS = ["graphql-ws", ]

CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "asgi_redis.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("redis", 6379)],
        },
        "ROUTING": "djcourses.urls.channel_routing",
    },

}
```

```
pip install channels
```

SUBSCRIPTIONS

```
class Subscription(graphene.ObjectType):

    count_seconds = graphene.Int(up_to=graphene.Int())
    sub_user = graphene.Field(
        UserNode, description='subscribe to updated user', username=graphene.String())

    def resolve_count_seconds(root, info, up_to=5):
        return Observable.interval(1000)\
                         .map(lambda i: "{0}".format(i))\
                         .take_while(lambda i: int(i) <= up_to)

    def resolve_sub_user(root, info, *args, **kwargs):
        username = kwargs.get('username')

        def get_object(observer):
            instance = User.objects.get(username=username)
            return instance
        return Observable.interval(1000) \
            .map(lambda s: get_object(s)) \
            .share()
```

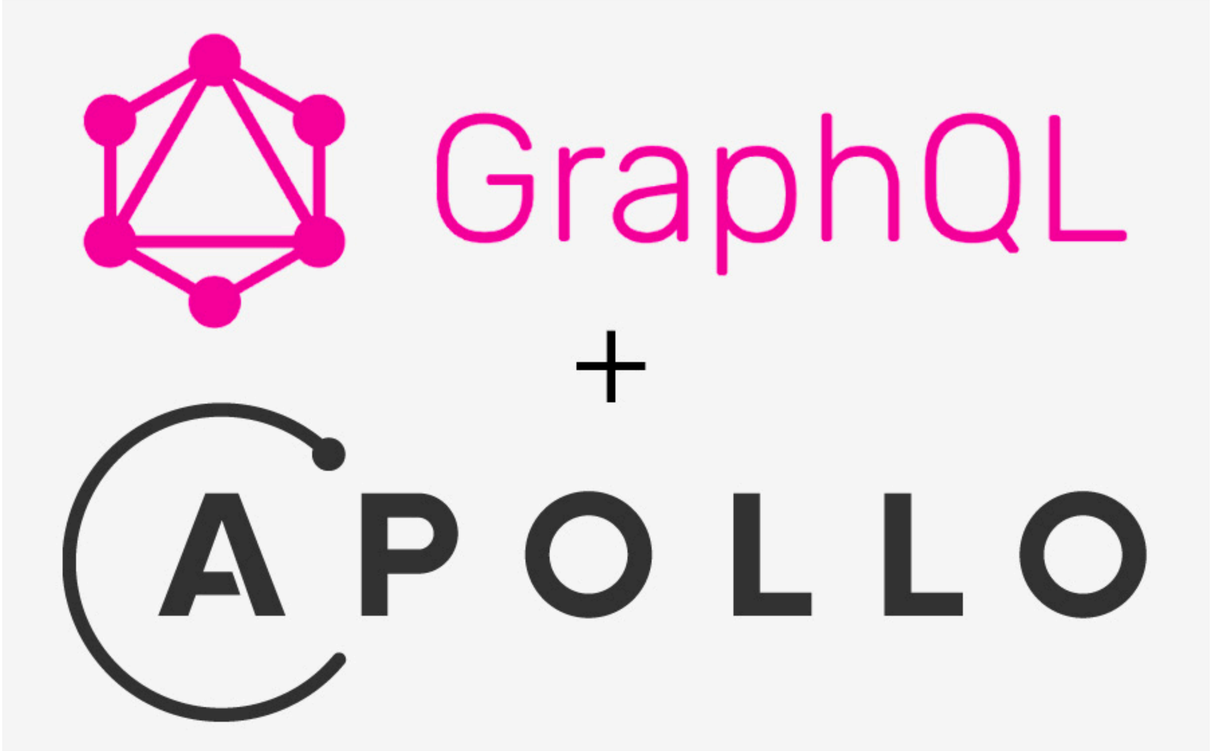
SUBSCRIPTIONS

```python
from graphql_ws.django_channels import GraphQLSubscriptionConsumer
from channels.routing import route_class


channel_routing = [
    route_class(GraphQLSubscriptionConsumer, path=r"^/subscriptions"),
]
```

DÓNDE PUEDO USAR GRAPHQL?


Apollo Android


Swift +


GraphQL
+
APOLLO

PREGUNTAS


GRACIAS
PASSION LED US HERE