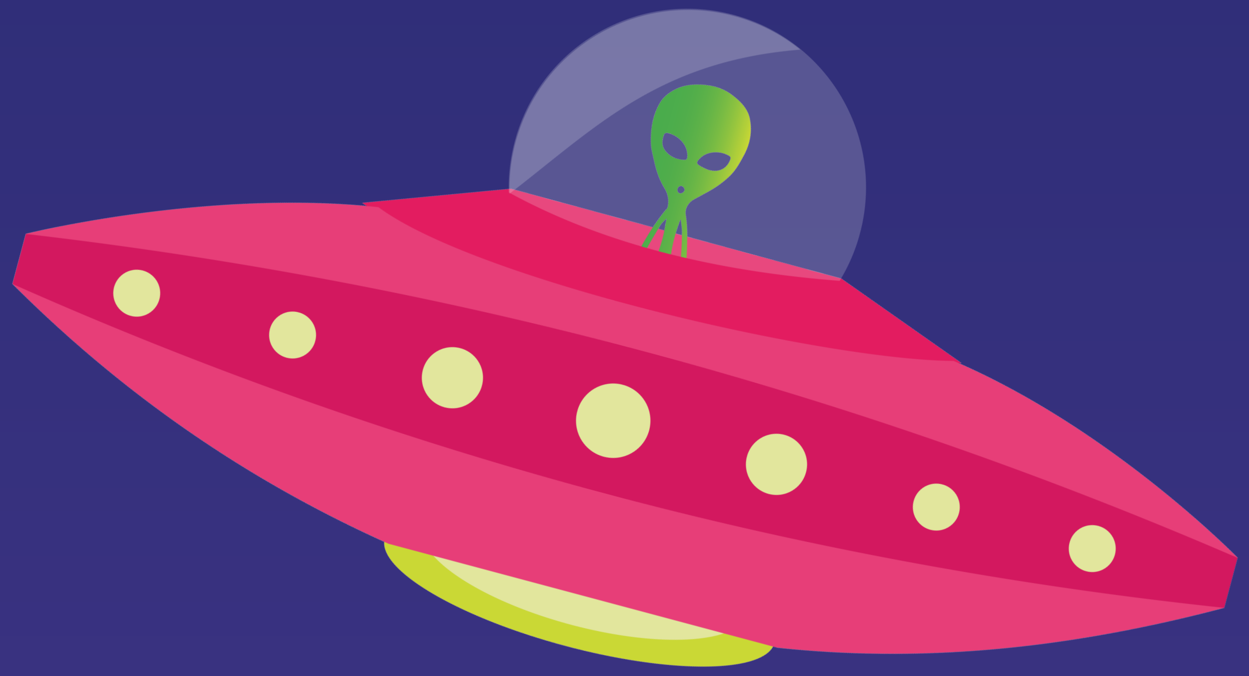
2013-2014

@BRWNGRLDEV

U.S.A

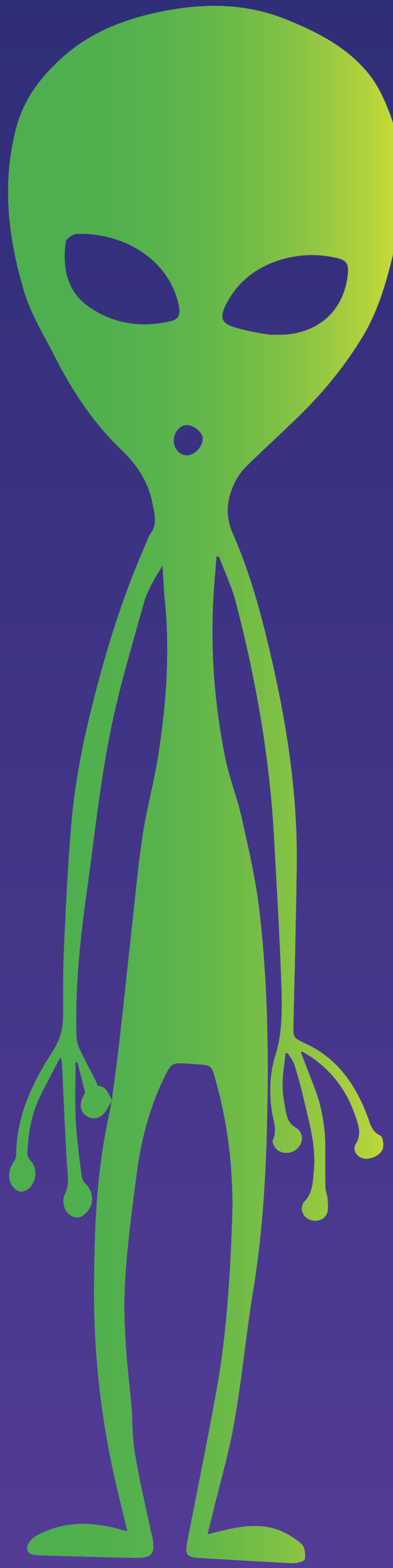


891 TIMES

GETTING A GRIP ON GRAPHQL

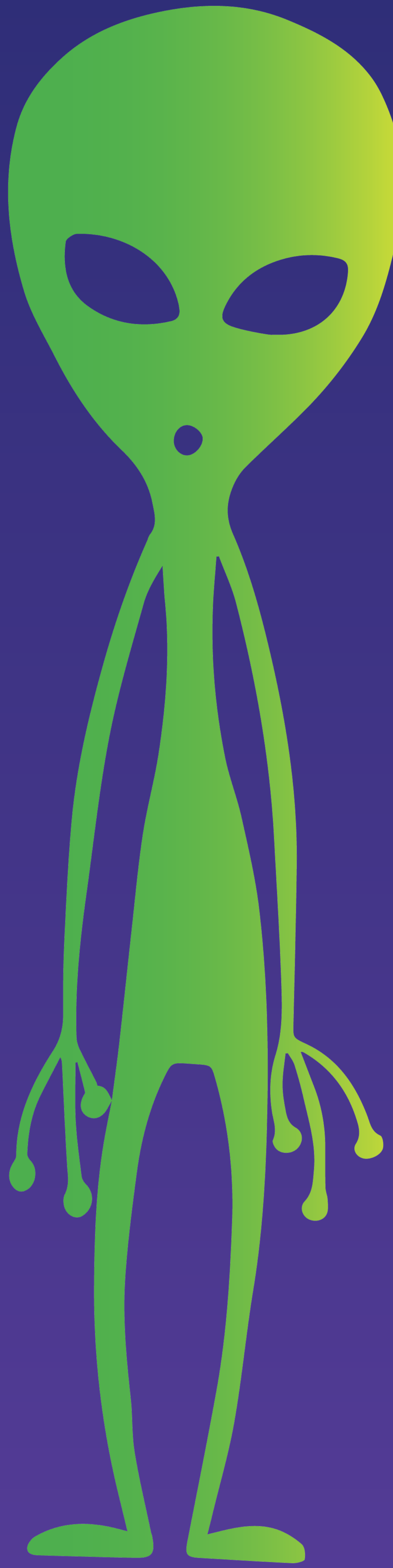
@BRWNGRLDEV

AGENDA



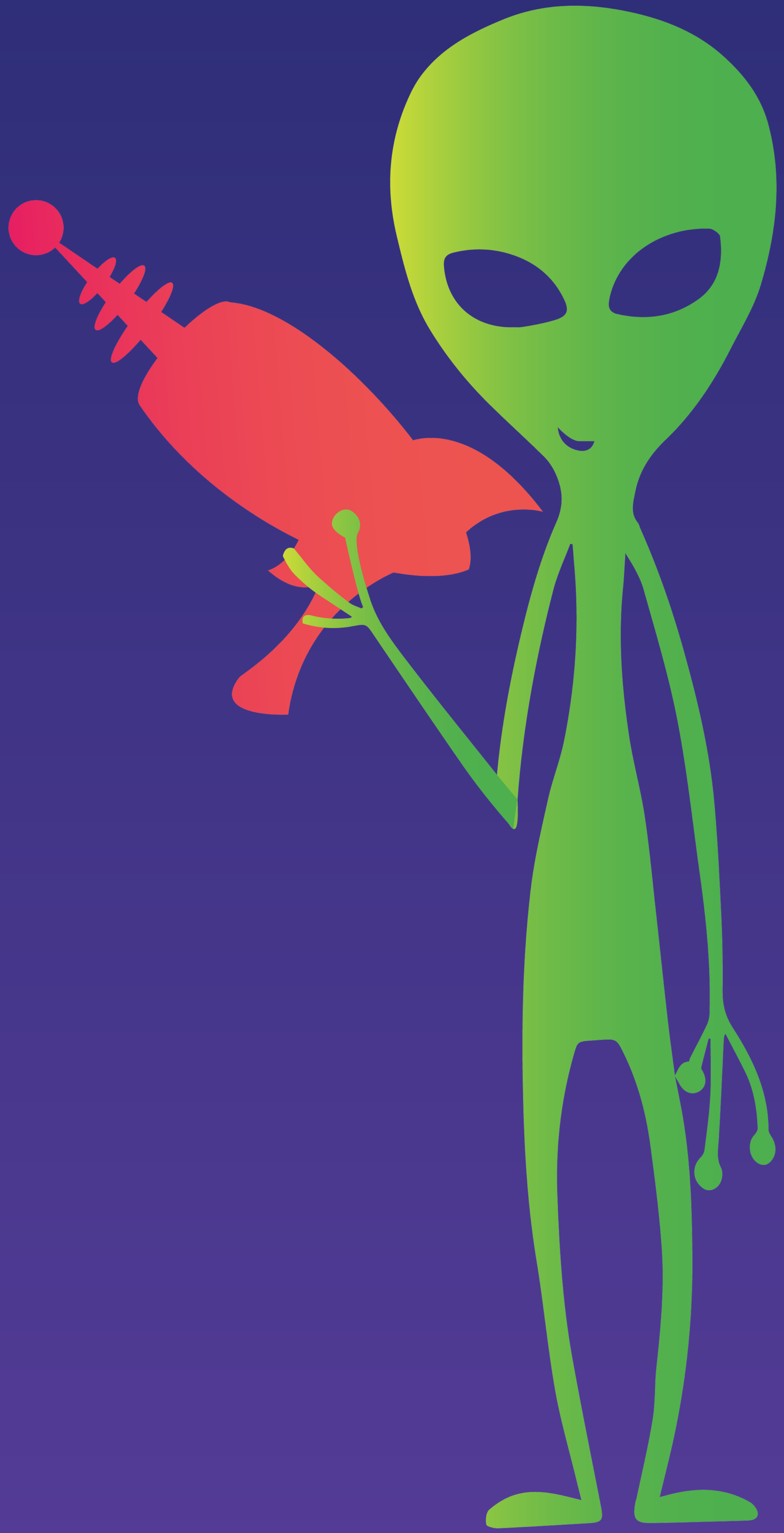
AGENDA

- Basics



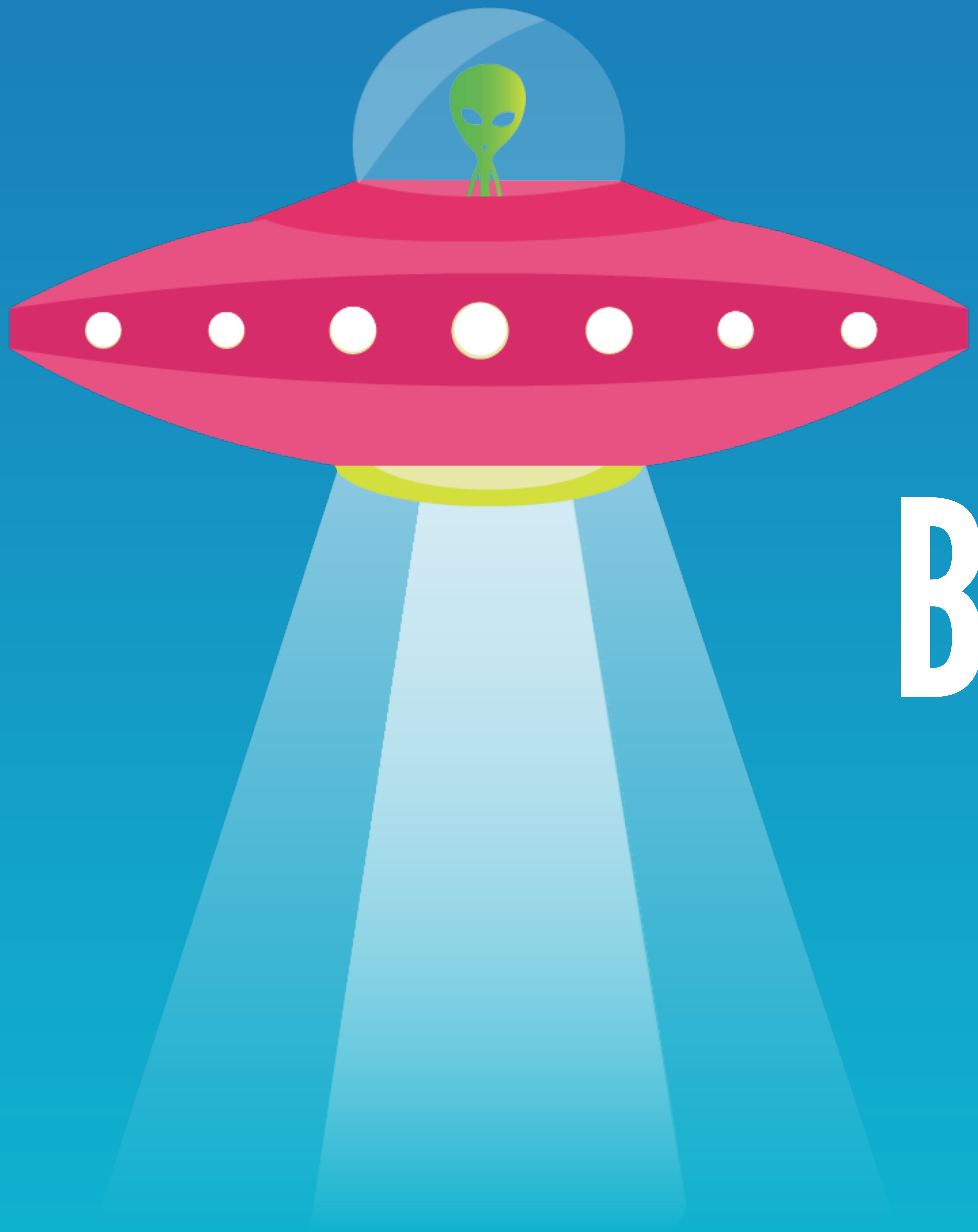
AGENDA

- Basics
- Server



AGENDA

- Basics
- Server
- Client



BASICS

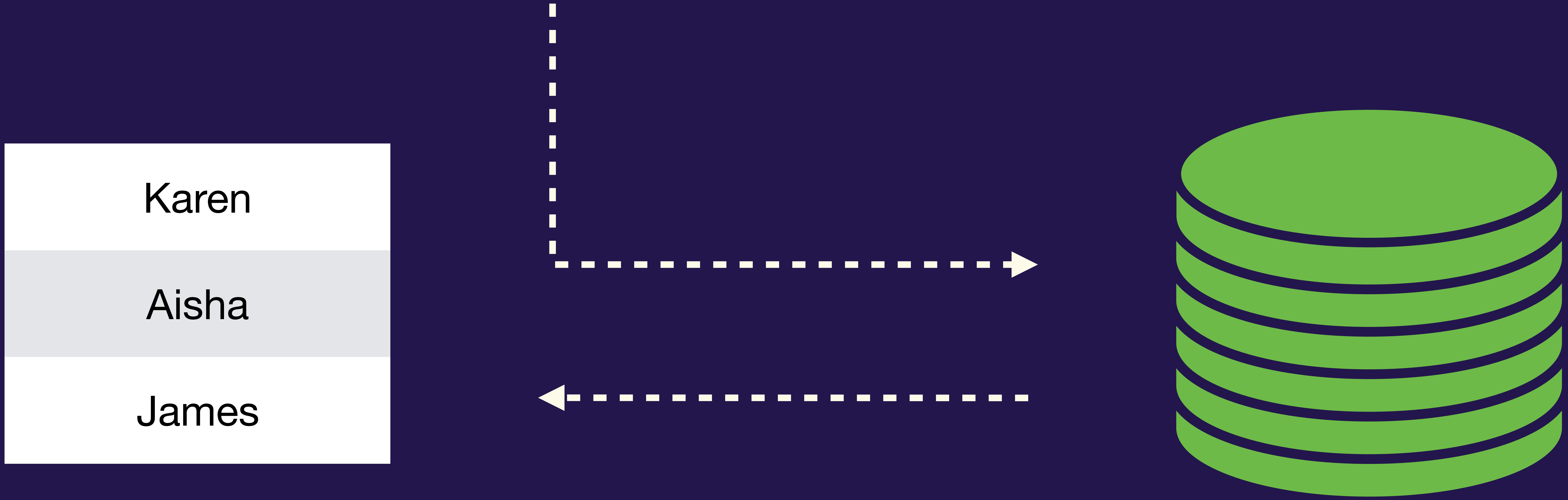
✓ GraphQL is...

A QUERY LANGUAGE

FOR YOUR API

```
SELECT name FROM users
SELECT name FROM users
SELECT name FROM users
```

```
SELECT name FROM users
```



Karen
Aisha
James

```
data class UFOSighting(
        var id: Int,
        var date: LocalDate,
        var city: String?,
        var state: String?,
        var country: String?,
        var shape: String?,
        var duration: Double,
        var comments: String?,
        var latitude: Double,
        var longitude: Double
)
```

```
query AllSightings {
  sightings {
    id
    shape
  }
}
```

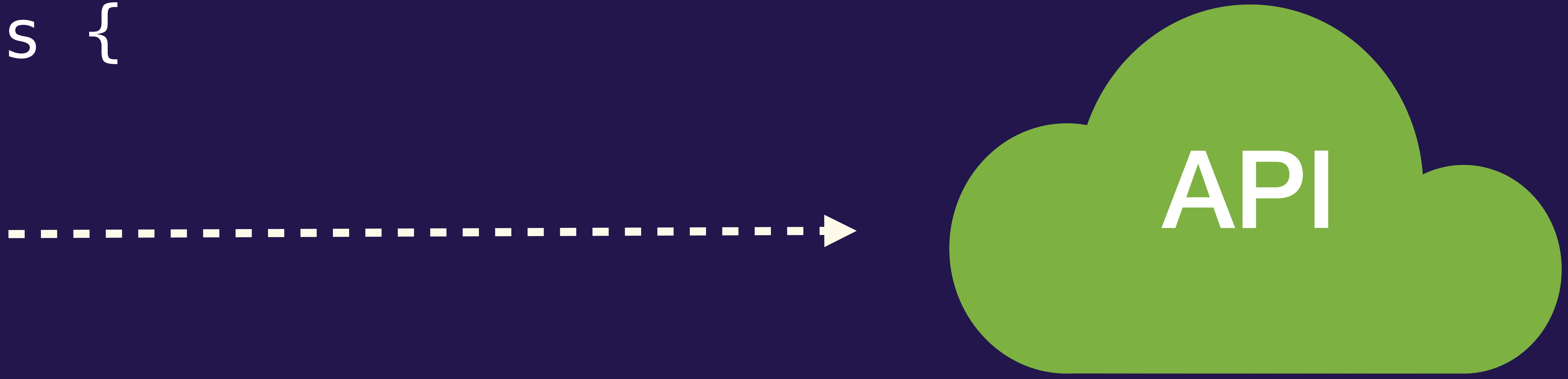
```
query AllSightings {
  sightings {
    id
    shape
  }
}
```

```
query AllSightings {
  sightings {
    id
    shape
  }
}
```

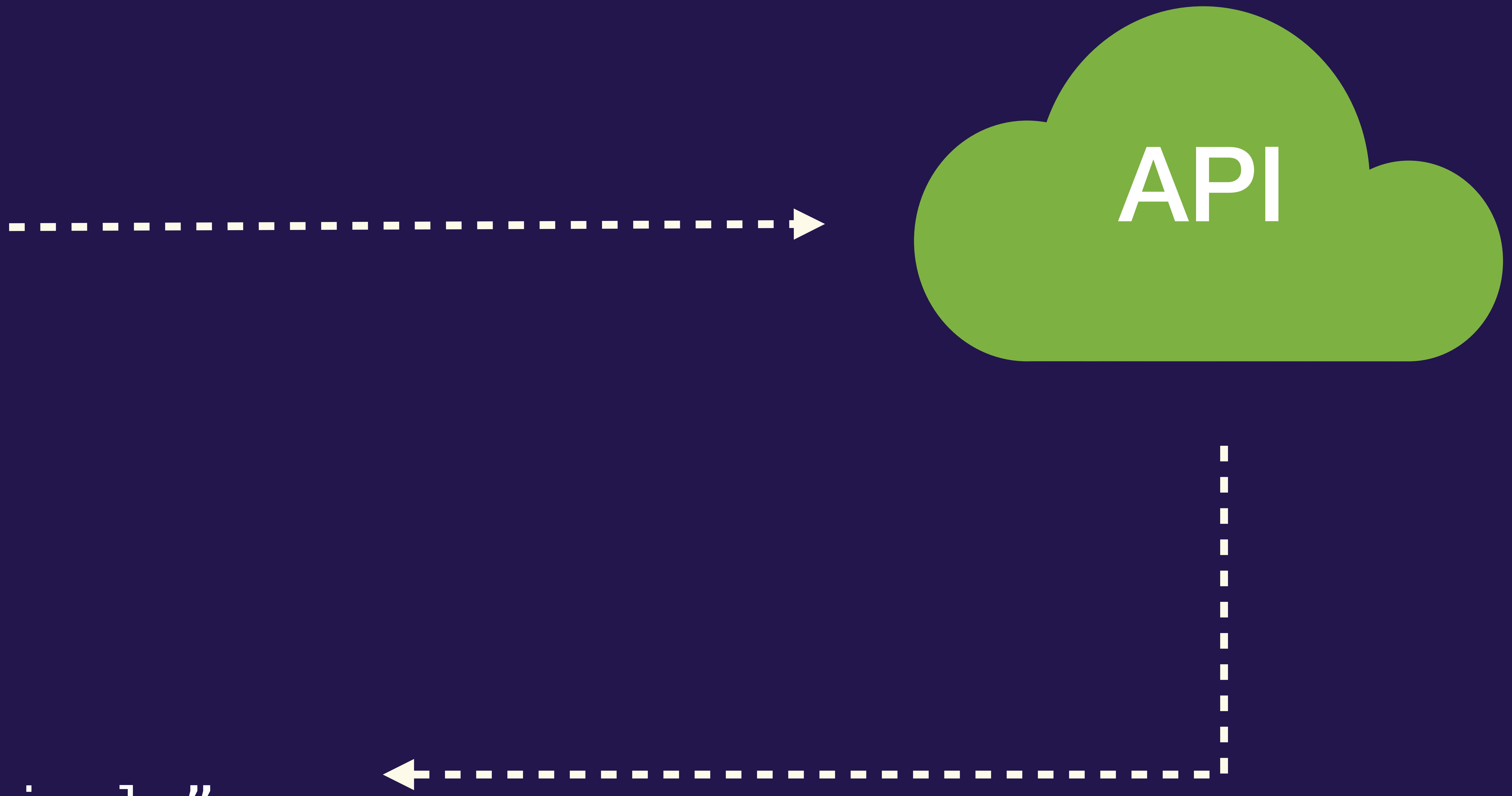
```
query AllSightings {
  sightings {
    id
    shape
  }
}
```

```
query AllSightings {
  sightings {
    id
    shape
  }
}
```

```
query AllSightings {
  sightings {
    id
    shape
  }
}
```



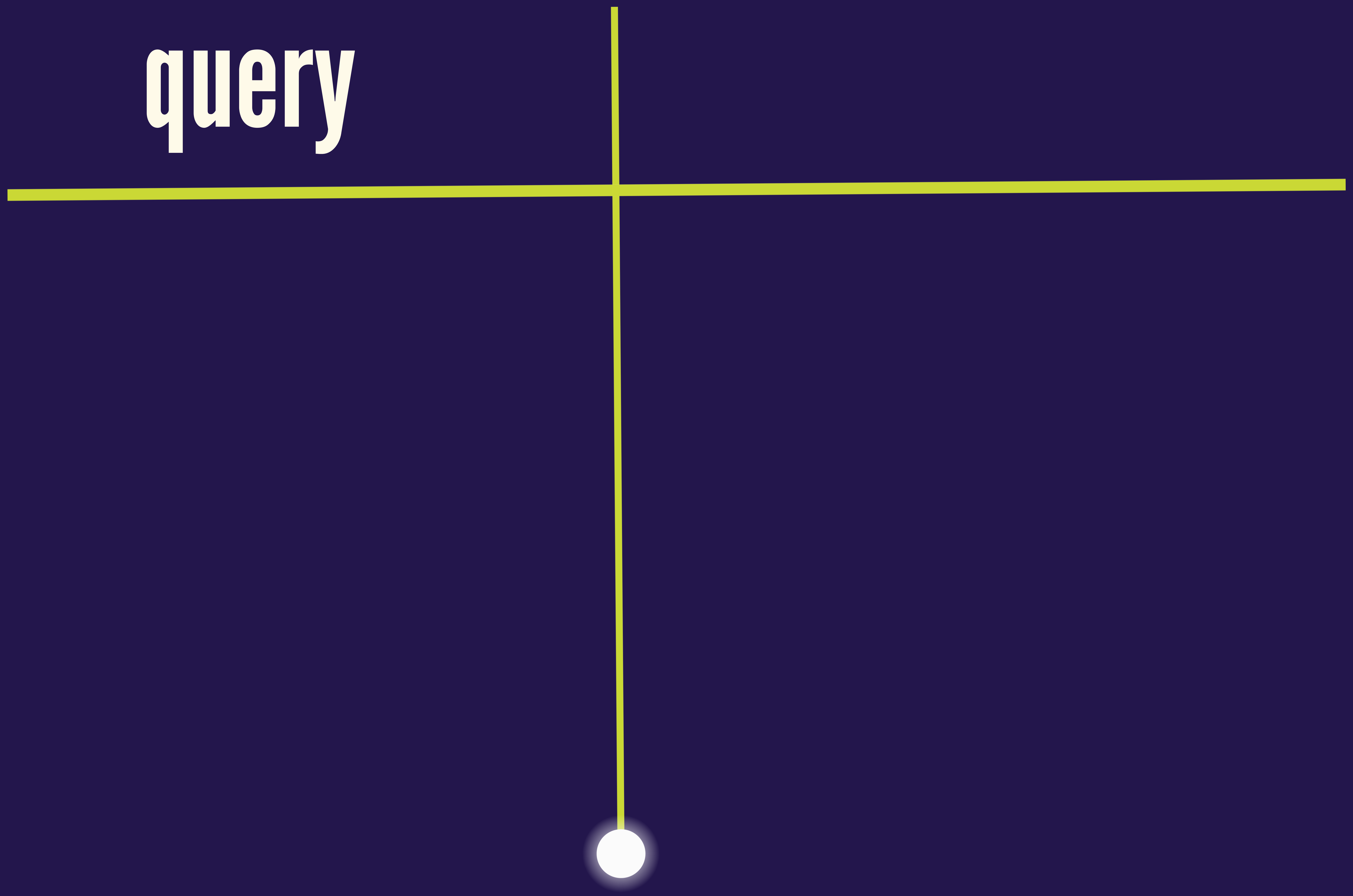
```
query AllSightings {
  sightings {
    id
    shape
  }
}
```



```
{
  “data” : {
    “sightings” : [
      {
        “id” : 1,
        “shape” : “circle”
      }
    ]
  }
}
```

```
{
  “data” : {
    “sightings” : [
      {
        “id” : 1,
        “shape” : “circle”
      }
    ]
  }
}
```

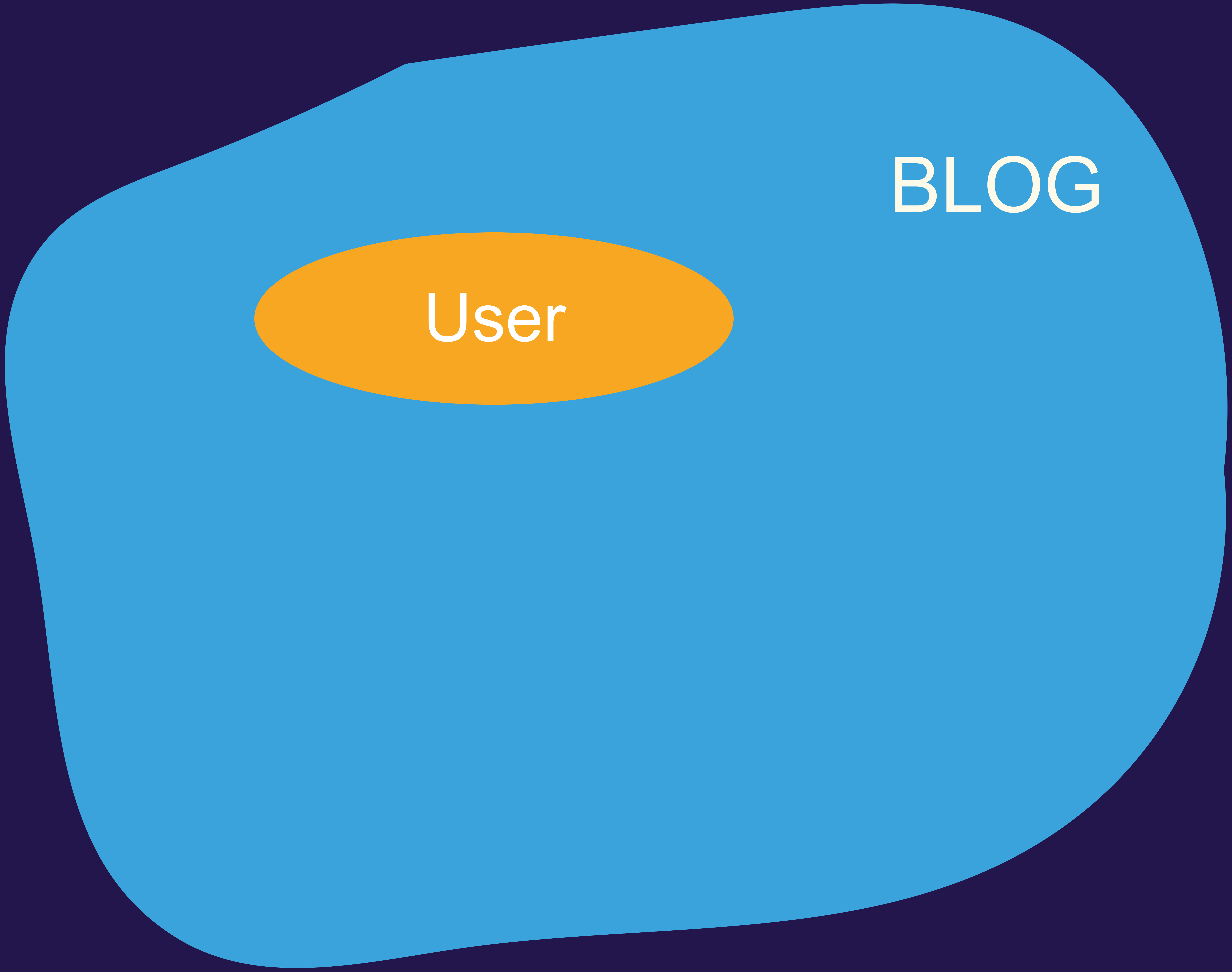
```
{
  “data” : {
    “sightings” : [
      {
        “id” : 1,
        “shape” : “circle”
      }
    ]
  }
}
```

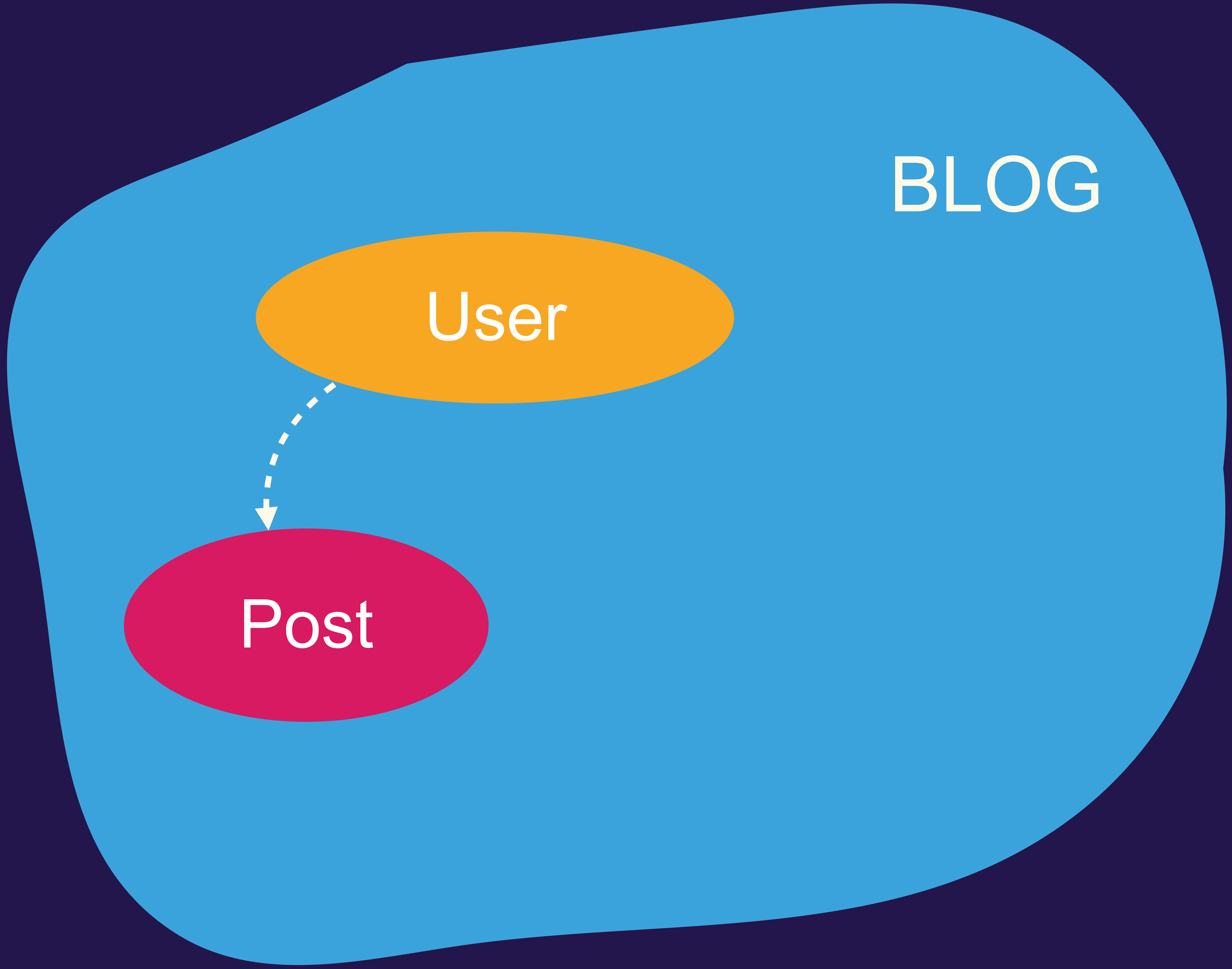

query

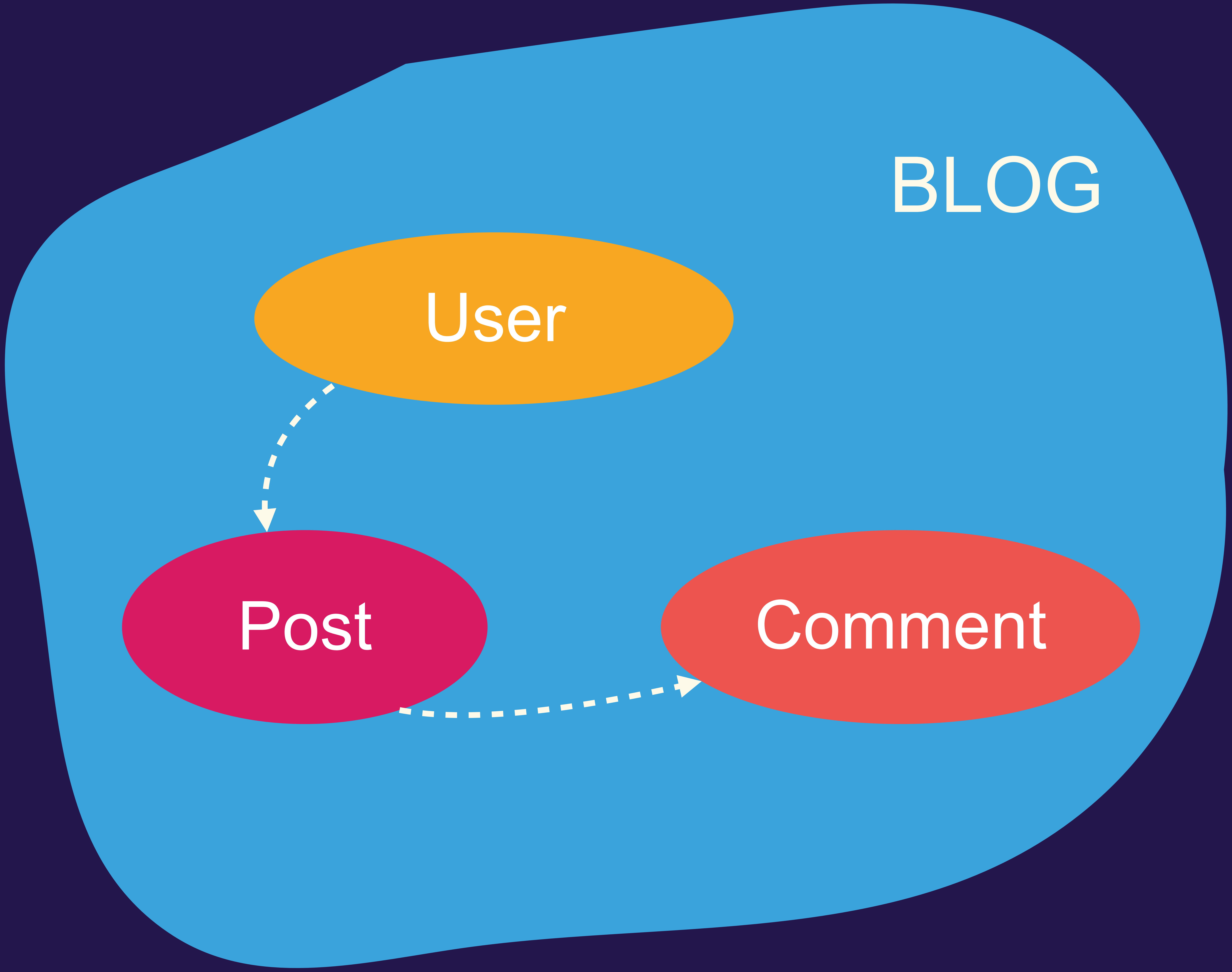
query	
GET	

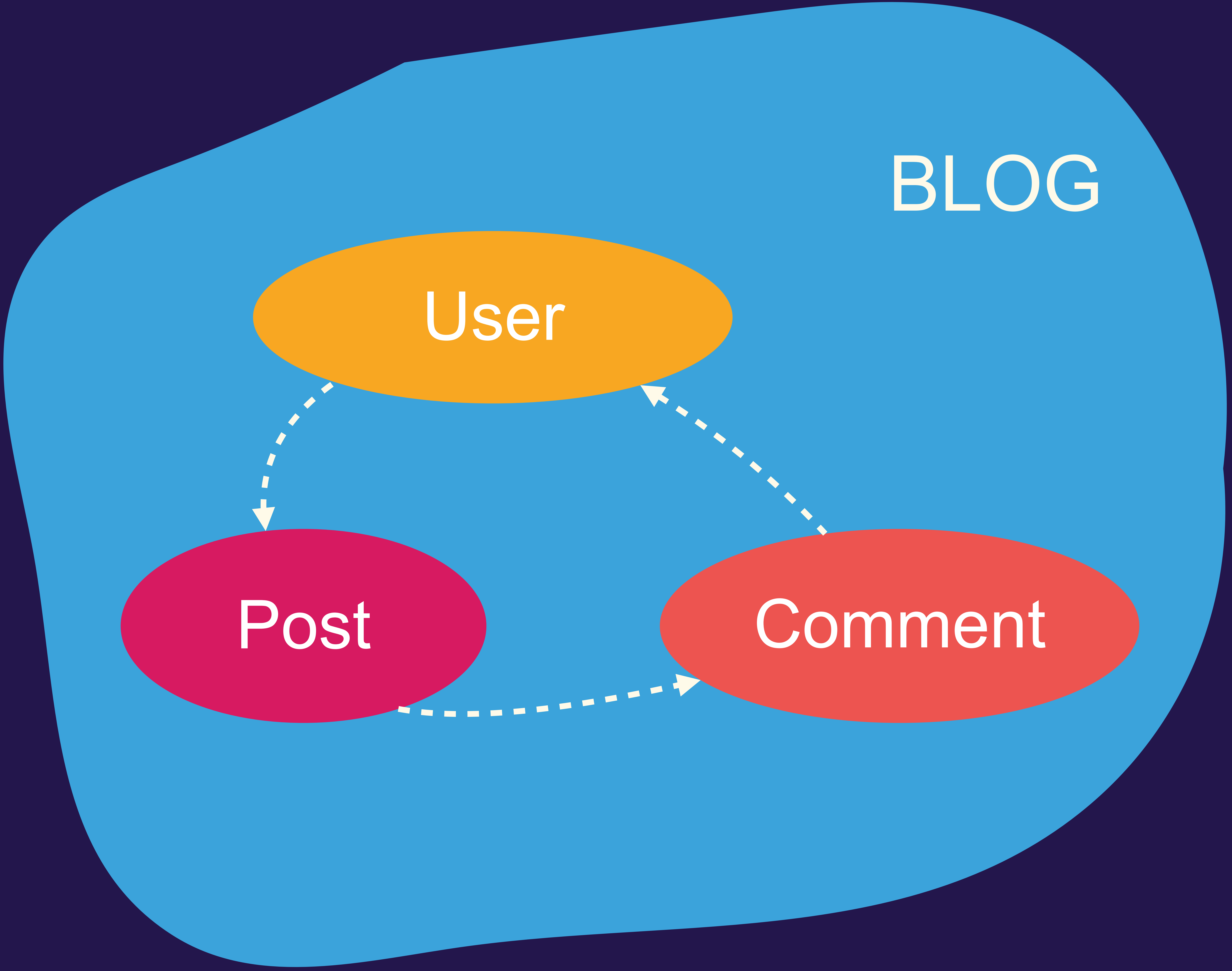
query	mutation
GET	POST
	PUT
	PATCH
	DELETE

BLOG


BLOG
User


BLOG
User
Post

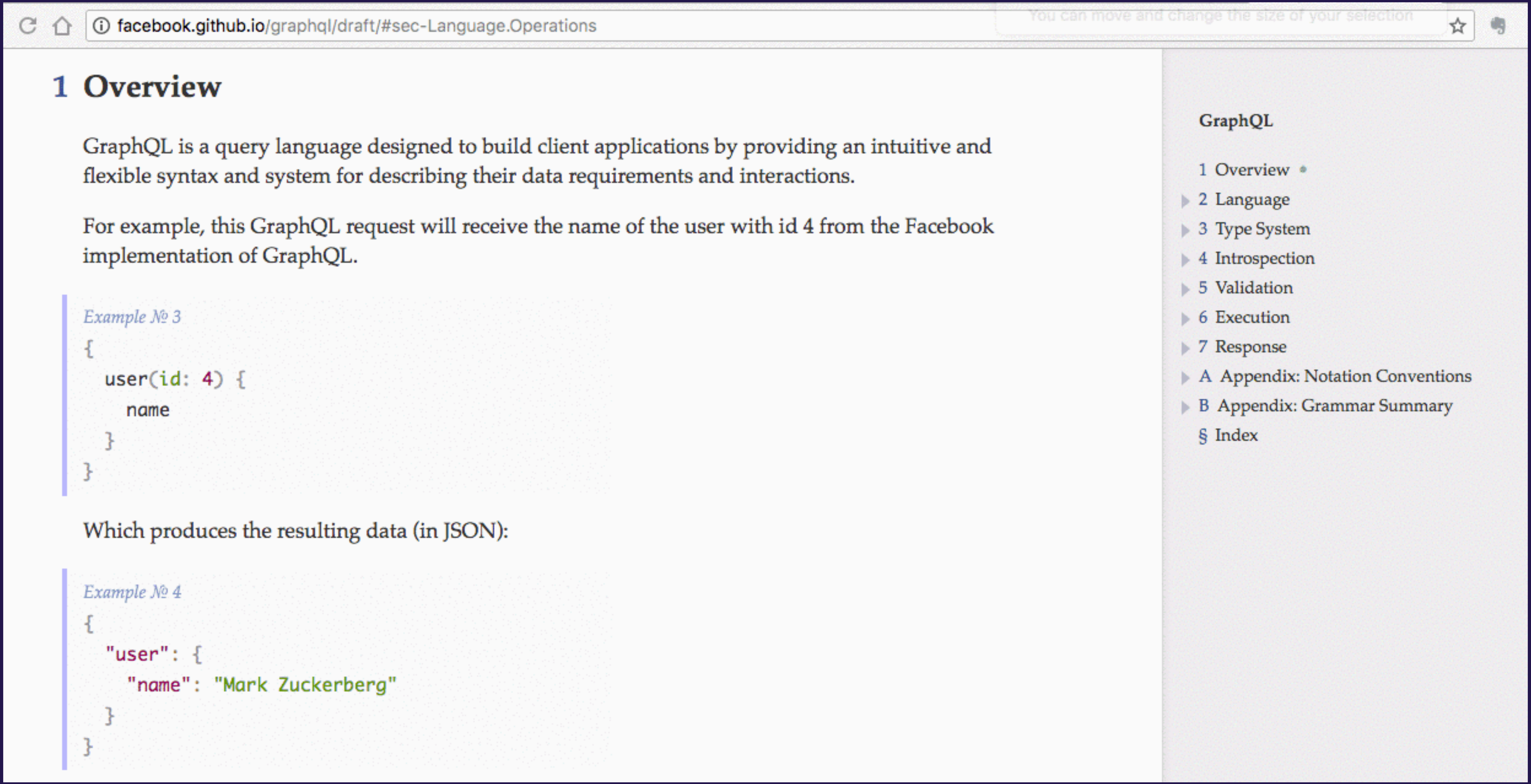

BLOG
User
Post
Comment


BLOG
User
Post
Comment

✓ GraphQL is...

A SPECIFICATION

@BRWNGRLDEV



facebook.github.io/graphql/draft/#sec-Language.Operations

1 Overview

GraphQL is a query language designed to build client applications by providing an intuitive and flexible syntax and system for describing their data requirements and interactions.

For example, this GraphQL request will receive the name of the user with id 4 from the Facebook implementation of GraphQL.

Example № 3

```
{
  user(id: 4) {
    name
  }
}
```

Which produces the resulting data (in JSON):

Example № 4

```
{
  "user": {
    "name": "Mark Zuckerberg"
  }
}
```

GraphQL

Give me some data!

You didn't say please!

Server

C# / .NET

Elixir

Kotlin

Java

JavaScript

Ruby

...

C# / .NET
Go
Java / Android
JavaScript
Python
Swift
...

Client

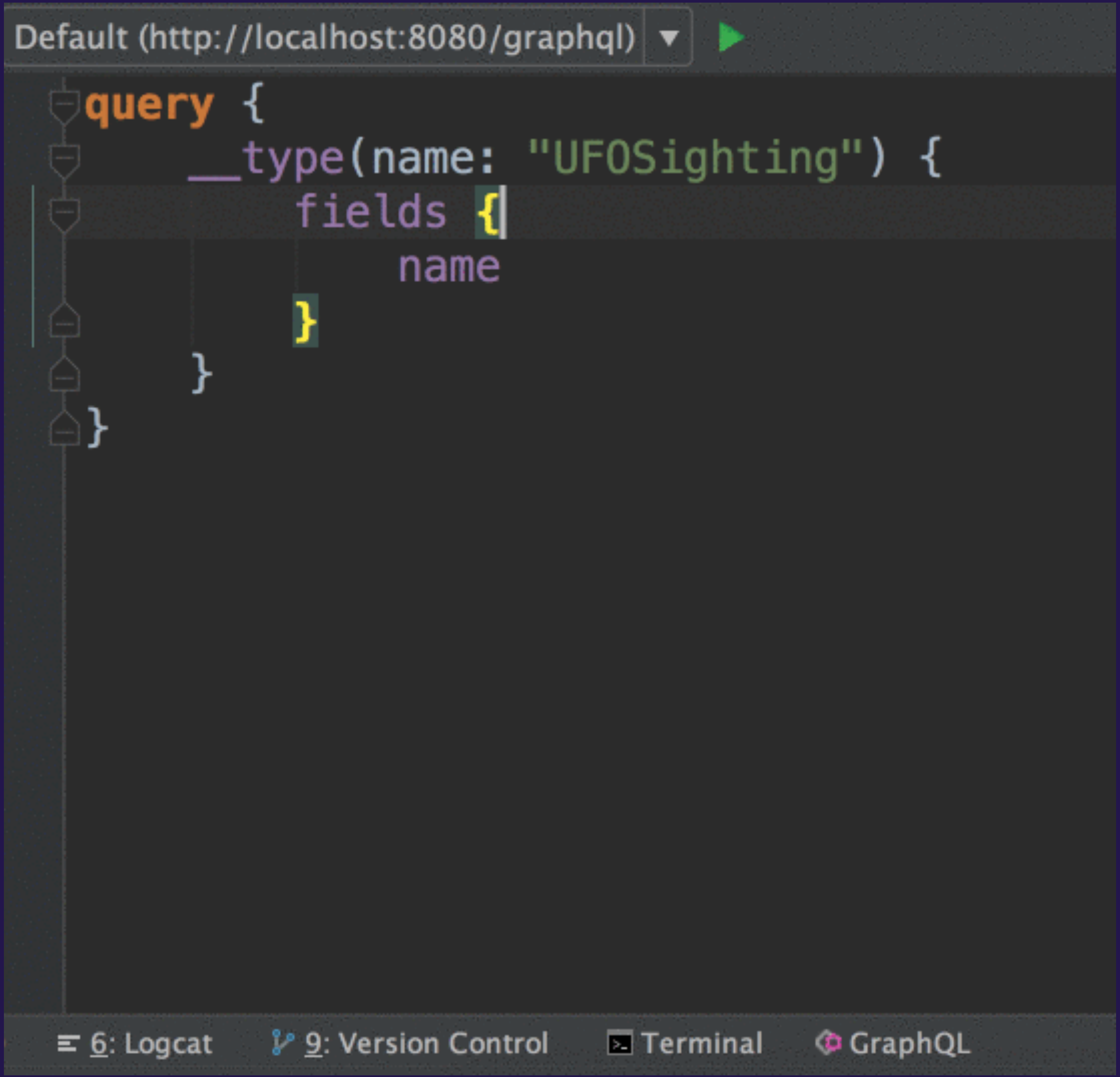
✓ GraphQL is...

INTROSPECTIVE

```
query {
  __type(name: "UFOSighting") {
    fields {
      name
    }
  }
}
```

```
query {
  __type(name: "UFOSighting") {
    fields {
      name
    }
  }
}
```

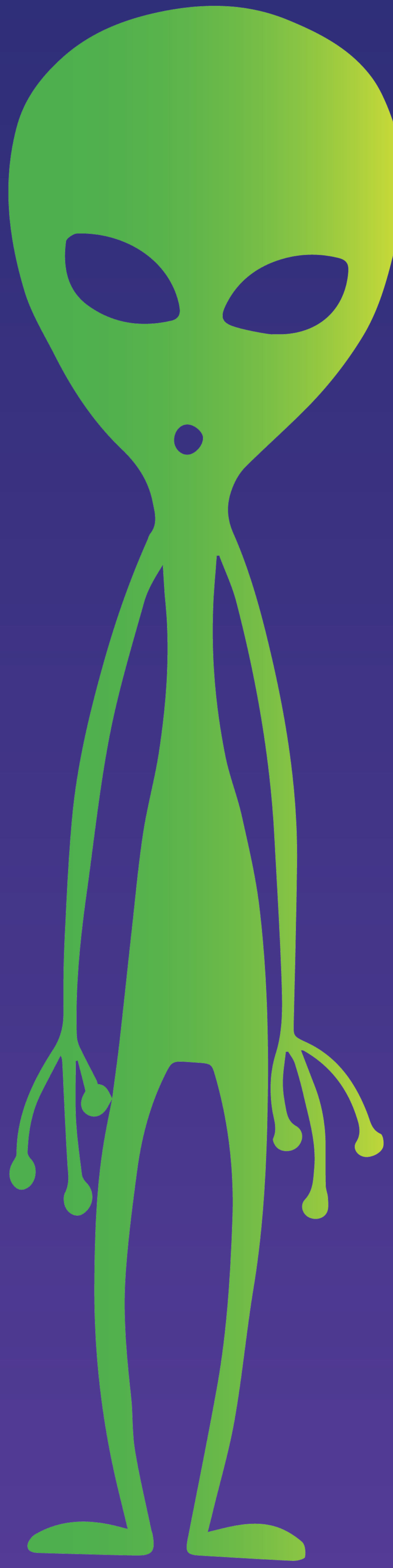
```
query {
  __type(name: "UFOSighting") {
    fields {
      name
    }
  }
}
```



```
query {
    __type(name: "UFOSighting") {
        fields {
            name
        }
    }
}
```

GraphQL is...

- ✓ a query language
- ✓ a specification
- ✓ introspective



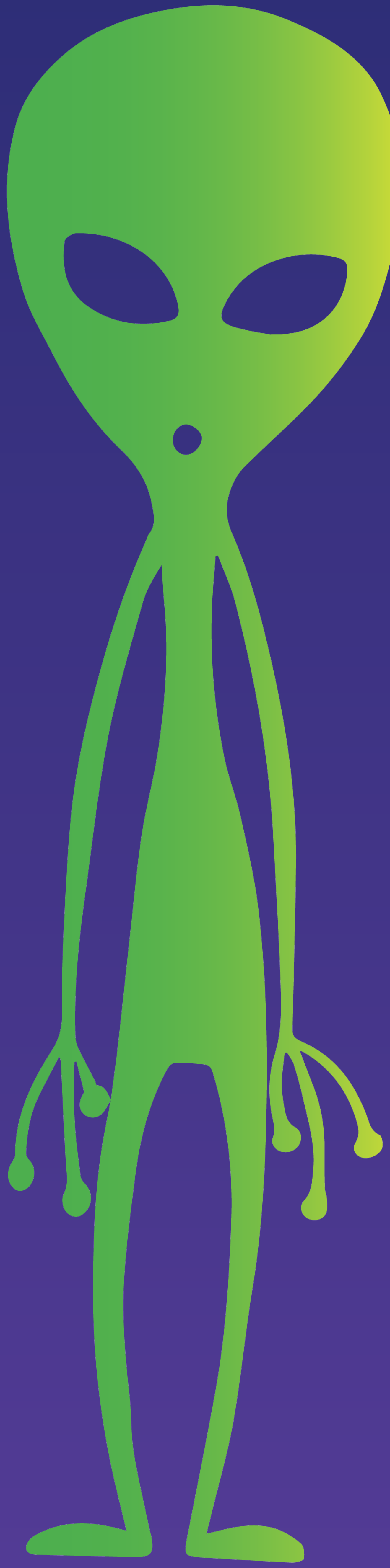
SO WHAT?!

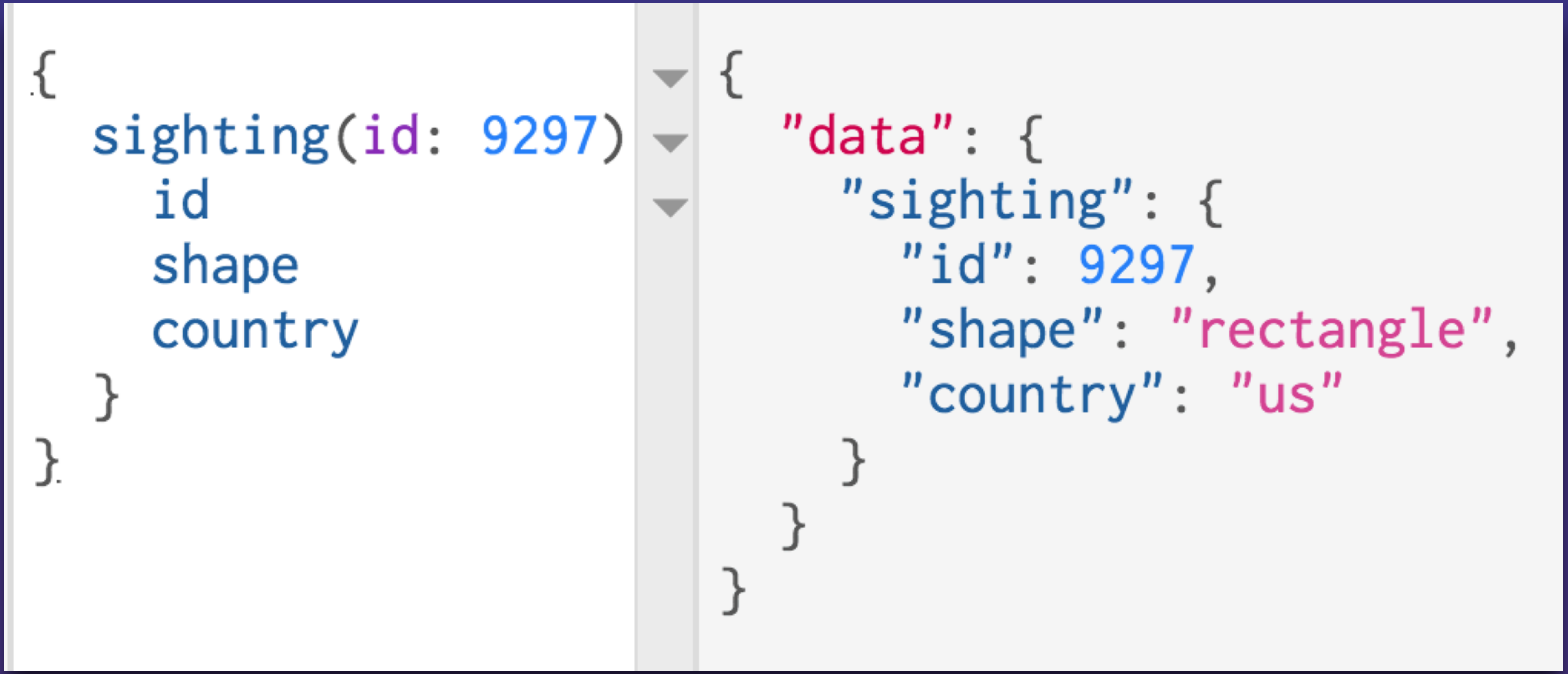
REST

/UFO-SIGHTINGS

```
[
      {
         "ID": 9298,
         "LONGITUDE": 145.722595,
         "LATITUDE": -38.626591,
         "STATE": "",
         "COUNTRY": "AU",
         "SHAPE": "LIGHT",
         "COMMENTS": "BRIGHT ORANGE LIGHT”
       },
       {
         "ID": 9297,
         "LONGITUDE": -90.0488889,
         "LATITUDE": 35.1494444,
         "STATE": "TN",
         "COUNTRY": "US",
         "SHAPE": "RECTANGLE",
         "COMMENTS": "STANDING AT MY WINDOW”
       },
       {
         "ID": 9287,
         "LONGITUDE": -3.1,
         "LATITUDE": 53.316667,
         "STATE": "YT",
         "COUNTRY": "GB",
         "SHAPE": "TRIANGLE",
         "COMMENTS": "((HOAX??)) LONG TRIANGLE OBJECT”
       },
…
```

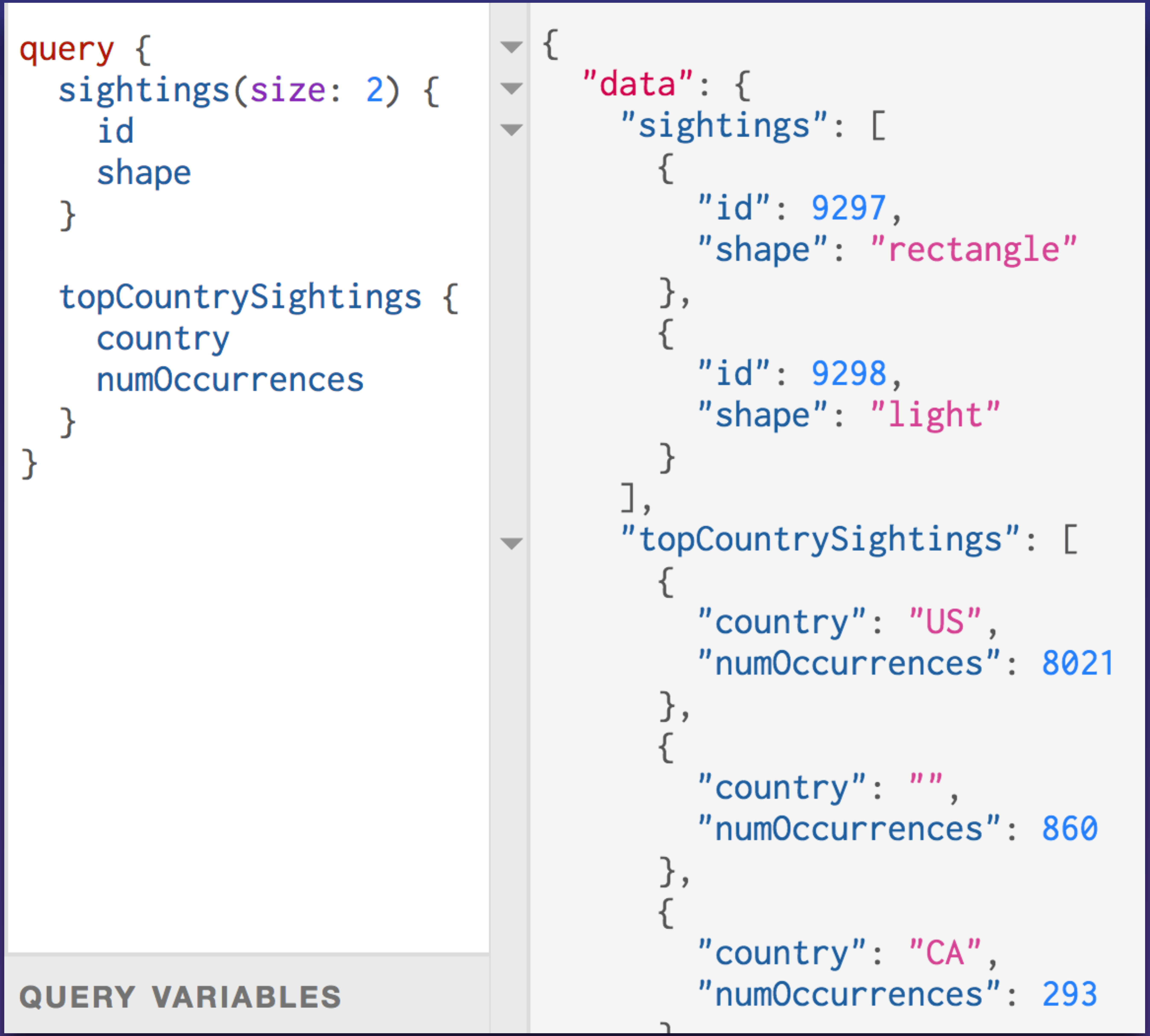
GRRAPSHQL





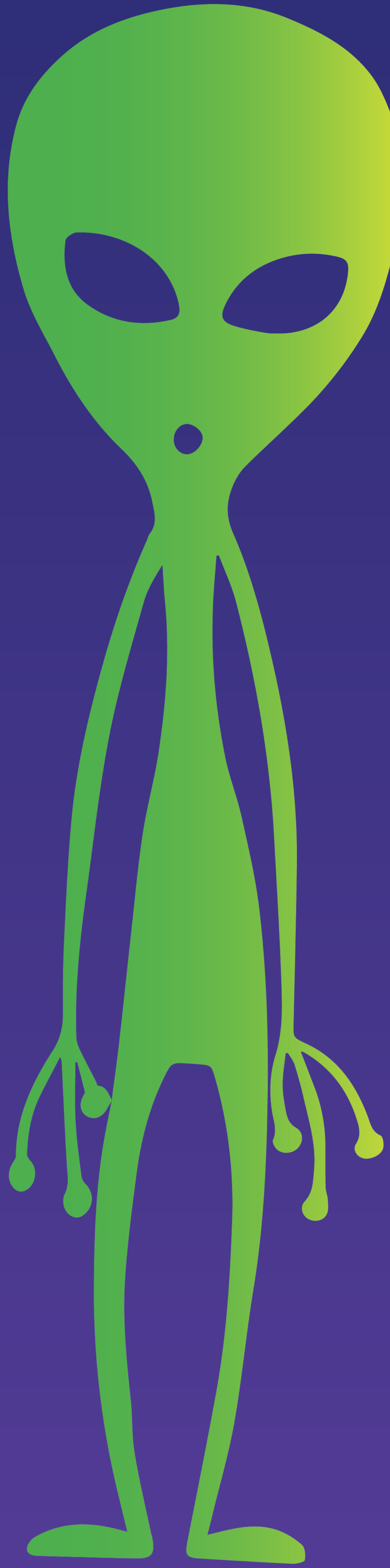
```
{
  sighting(id: 9297)
    id
    shape
    country
  }
}
```

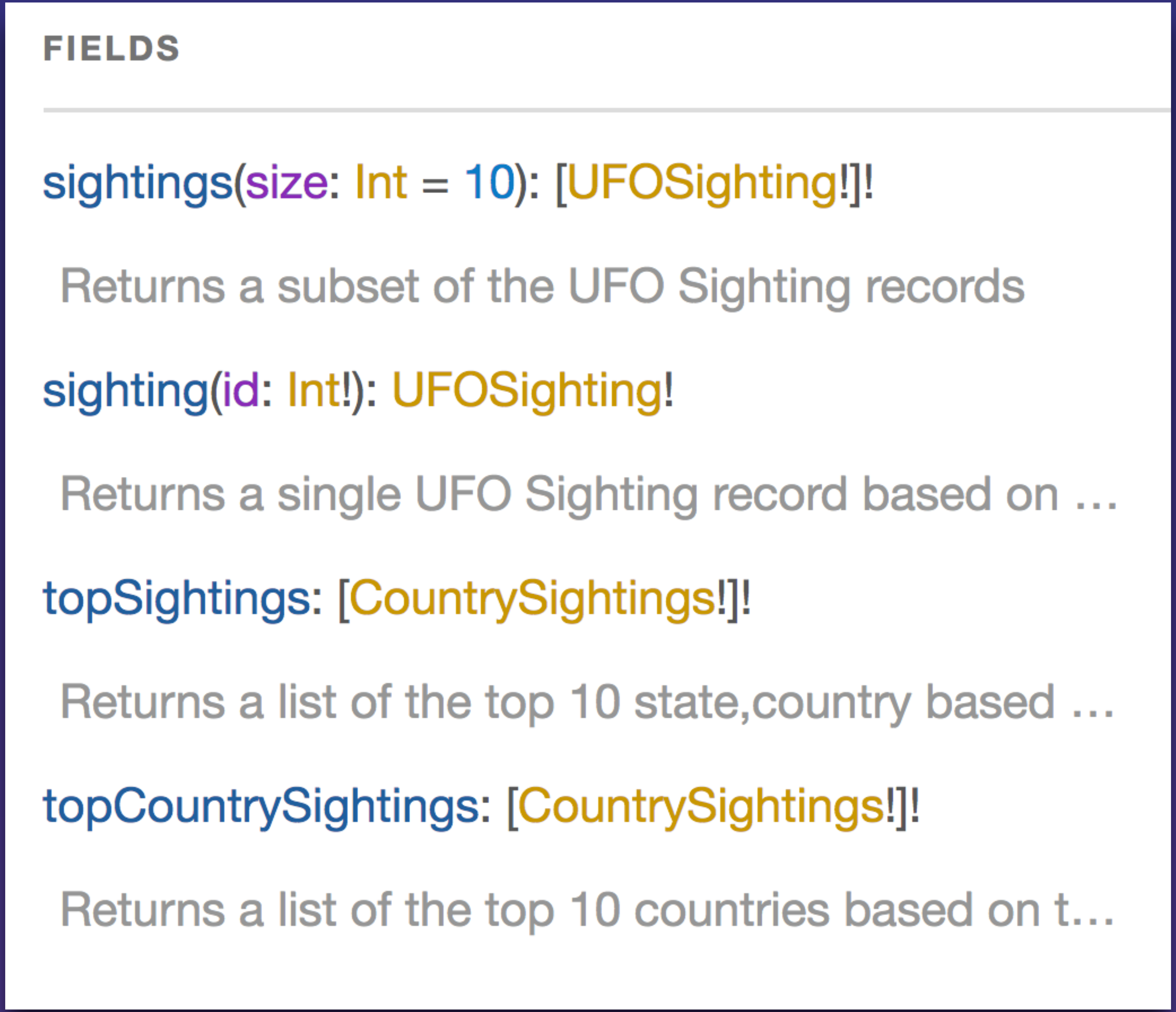
```
{
  "data": {
    "sighting": {
      "id": 9297,
      "shape": "rectangle",
      "country": "us"
    }
  }
}
```



```
query {
  sightings(size: 2) {
    id
    shape
  }

  topCountrySightings {
    country
    numOccurrences
  }
}
```

```
{
  "data": {
    "sightings": [
      {
        "id": 9297,
        "shape": "rectangle"
      },
      {
        "id": 9298,
        "shape": "light"
      }
    ],
    "topCountrySightings": [
      {
        "country": "US",
        "numOccurrences": 8021
      },
      {
        "country": "",
        "numOccurrences": 860
      },
      {
        "country": "CA",
        "numOccurrences": 293
```





FIELDS

sightings(size: Int = 10): [UFOSighting!]!

Returns a subset of the UFO Sighting records

sighting(id: Int!): UFOSighting!

Returns a single UFO Sighting record based on …

topSightings: [CountrySightings!]!

Returns a list of the top 10 state,country based …

topCountrySightings: [CountrySightings!]!

Returns a list of the top 10 countries based on t…

SERVER

Single Endpoint

/graphql

HTTP **GET**

/GRAPHQL?QUERY=<QUERY>

HTTP **GET**

/GRAPHQL?QUERY=<QUERY>

```
“{
  sightings {
    id
    shape
  }
}”
```

HTTP **POST**

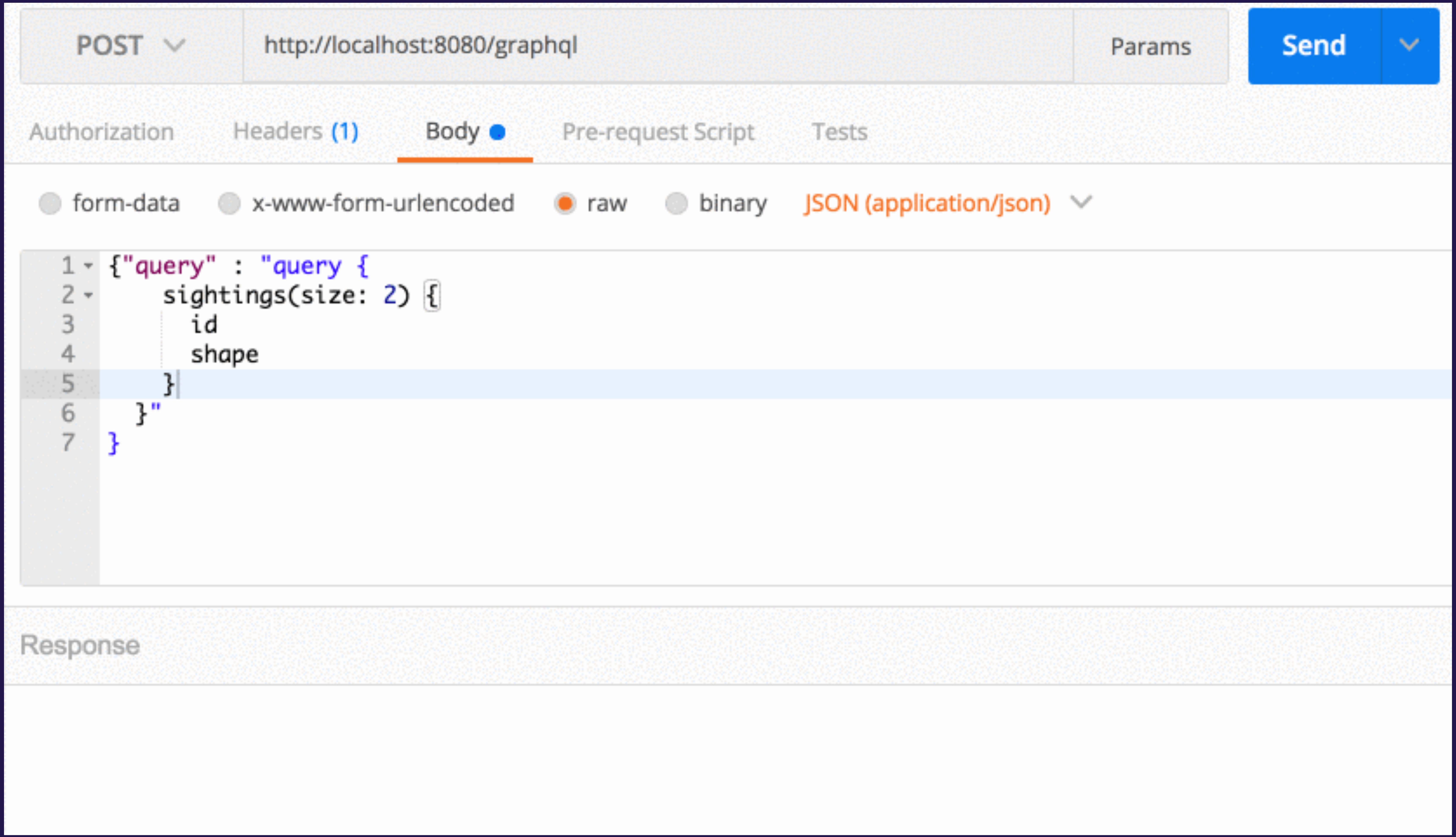
/GRAPHQL

HTTP **POST**

/GRAPHQL

```
{
  “query” : “{
                sightings {
                  id
                  shape
                }
              }”
}
```

Postman

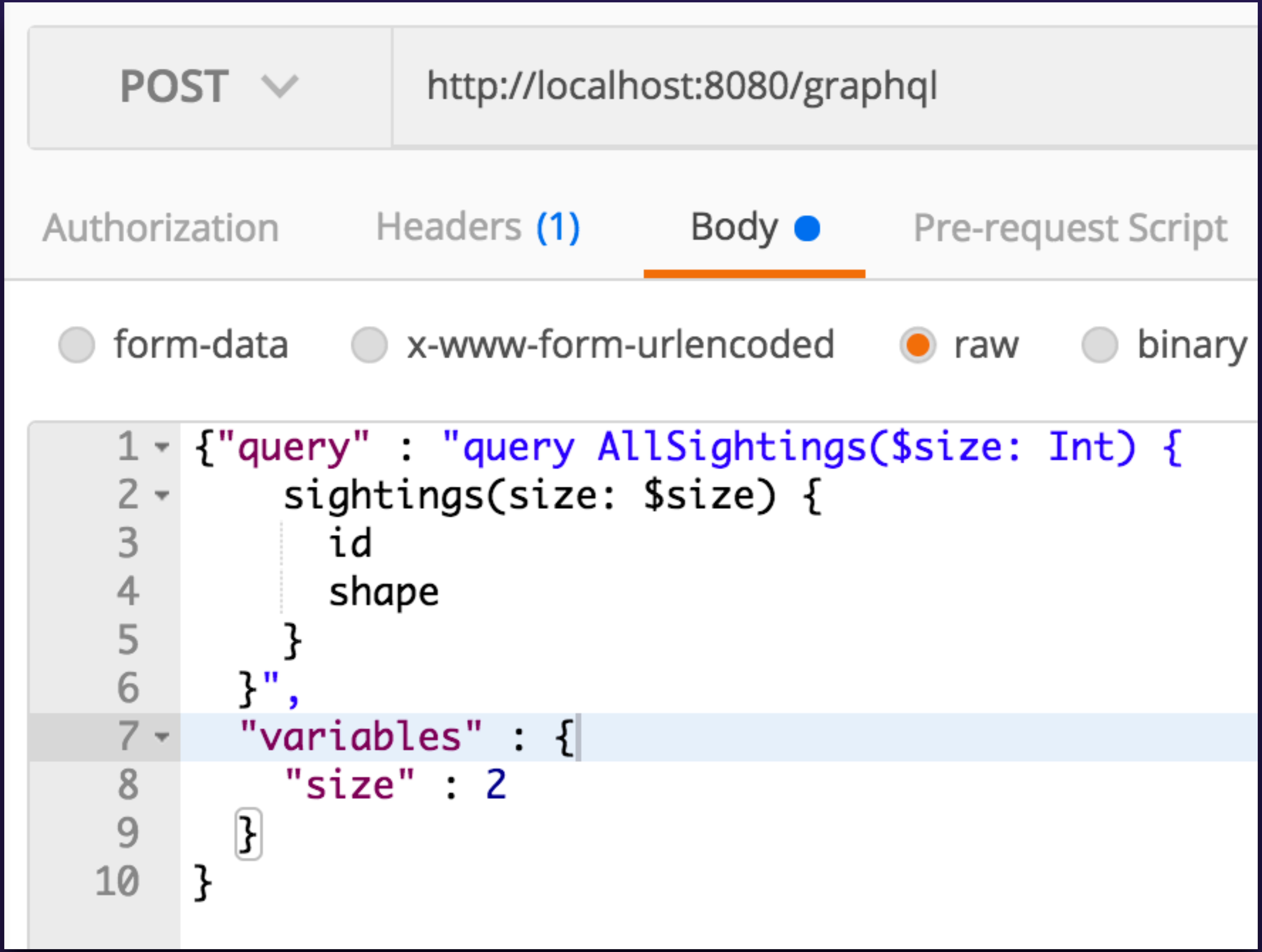


```
http://localhost:8080/graphql
```

raw — JSON (application/json)

```
{"query" : "query {
    sightings(size: 2) {
      id
      shape
    }
  }"
}
```

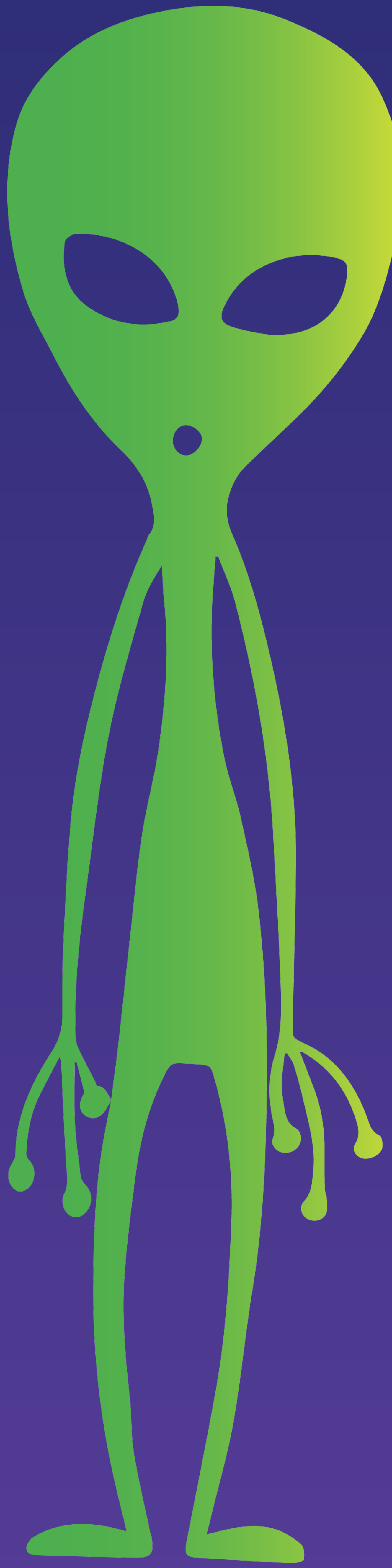
Postman



http://localhost:8080/graphql

[✓ raw] form-data x-www-form-urlencoded binary

```
{"query" : "query AllSightings($size: Int) {
    sightings(size: $size) {
      id
      shape
    }
  }",
  "variables" : {
    "size" : 2
  }
}
```



BUILDING OUR SERVER

Ktor – Server Framework

Ktor – Server Framework

Koin – Dependency Injection

Ktor – Server Framework

Koin – Dependency Injection

Squash – Database Access

Ktor – Server Framework

Koin – Dependency Injection

Squash – Database Access

KGraphQL – GraphQL Support

GraphQL Server...

✓ Types
✓ Schema
✓ Resolvers

C	S	S	H	N	A	S	V	Z	U	Y	C	J	H	M
N	I	W	P	X	O	D	C	B	Q	Q	G	L	F	T
C	P	B	U	H	Z	I	B	A	U	M	A	Q	S	V
Q	O	W	G	Y	F	M	T	E	L	M	A	H	F	H
R	E	V	L	O	S	E	R	A	E	A	T	P	I	R
K	F	S	R	R	S	Y	N	H	T	P	R	A	E	J
O	B	J	E	C	T	S	C	U	L	U	D	R	L	P
E	P	Y	T	L	O	S	X	L	U	B	M	G	D	X

@BRWNGRLDEV

C	S	S	H	N	A	S	V	Z	U	Y	C	J	H	M
N	I	W	P	X	O	D	C	B	Q	Q	G	L	F	T
C	P	B	U	H	Z	I	B	A	U	M	A	Q	S	V
Q	O	W	G	Y	F	M	T	E	L	M	A	H	F	H
R	E	V	L	O	S	E	R	A	E	A	T	P	I	R
K	F	S	R	R	S	Y	N	H	T	P	R	A	E	J
O	B	J	E	C	T	S	C	U	L	U	D	R	L	P
E	P	Y	T	L	O	S	X	L	U	B	M	G	D	X

```
type UFOSighting {
    id: Int!
    city: String
}
```

```
type UFOSighting {
    id: Int!
    city: String
}
```

```
type UFOSighting {
    id: Int!
    city: String
}
```

```
{
  sightings {
    id {
       ???
    }
    city
  }
}
```

```
type UFOSighting {
    id: Int!
    city: String
}
```

WRONG!

```
{
  sightings {
    i   {
          ??
    }
    city
  }
}
```

```
type<UFOSighting>
```

```
type<UFOSighting>
```

```
data class UFOSighting(
  var id: Int = -1,
  var city: String? = "",
)
```

```
type<UFOSighting>

data class UFOSighting(
   var id: Int = -1,
   var city: String? = "",
)
```

```
type<UFOSighting>

data class UFOSighting(
  var id: Int = -1,
  var city: String? = "",
)
```

```
type UFOSighting {
    id: Int!
    city: String
}
```

@BRWNGRLDEV

C	S	S	H	N	A	S	V	Z	U	Y	C	J	H	M
N	I	W	P	X	O	D	C	B	Q	Q	G	L	F	T
C	P	B	U	H	Z	I	B	A	U	M	A	Q	S	V
Q	O	W	G	Y	F	M	T	E	L	M	A	H	F	H
R	E	V	L	O	S	E	R	A	E	A	T	P	I	R
K	F	S	R	R	S	Y	N	H	T	P	R	A	E	J
O	B	J	E	C	T	S	C	U	L	U	D	R	L	P
E	P	Y	T	L	O	S	X	L	U	B	M	G	D	X

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

```
type Query {
  sighting(id: Int): UFOSighting
}

schema {
  query: Query
}
```

```
type UFOSighting {
  id: Int!
  city: String
}

type Query {
  sighting(id: Int): UFOSighting
}

schema {
  query: Query
}
```

Data, please!
Why, yes!

```
KGraphQL.schema {







}
```

```
KGraphQL.schema {

  type<UFOSighting>






}
```

```
KGraphQL.schema {

  type<UFOSighting>

  query("sighting") {
    resolver { id: Int -> …}
  }

}
```

@BRWNGRLDEV

C	S	S	H	N	A	S	V	Z	U	Y	C	J	H	M
N	I	W	P	X	O	D	C	B	Q	Q	G	L	F	T
C	P	B	U	H	Z	I	B	A	U	M	A	Q	S	V
Q	O	W	G	Y	F	M	T	E	L	M	A	H	F	H
R	E	V	L	O	S	E	R	A	E	A	T	P	I	R
K	F	S	R	R	S	Y	N	H	T	P	R	A	E	J
O	B	J	E	C	T	S	C	U	L	U	D	R	L	P
E	P	Y	T	L	O	S	X	L	U	B	M	G	D	X

Resolver

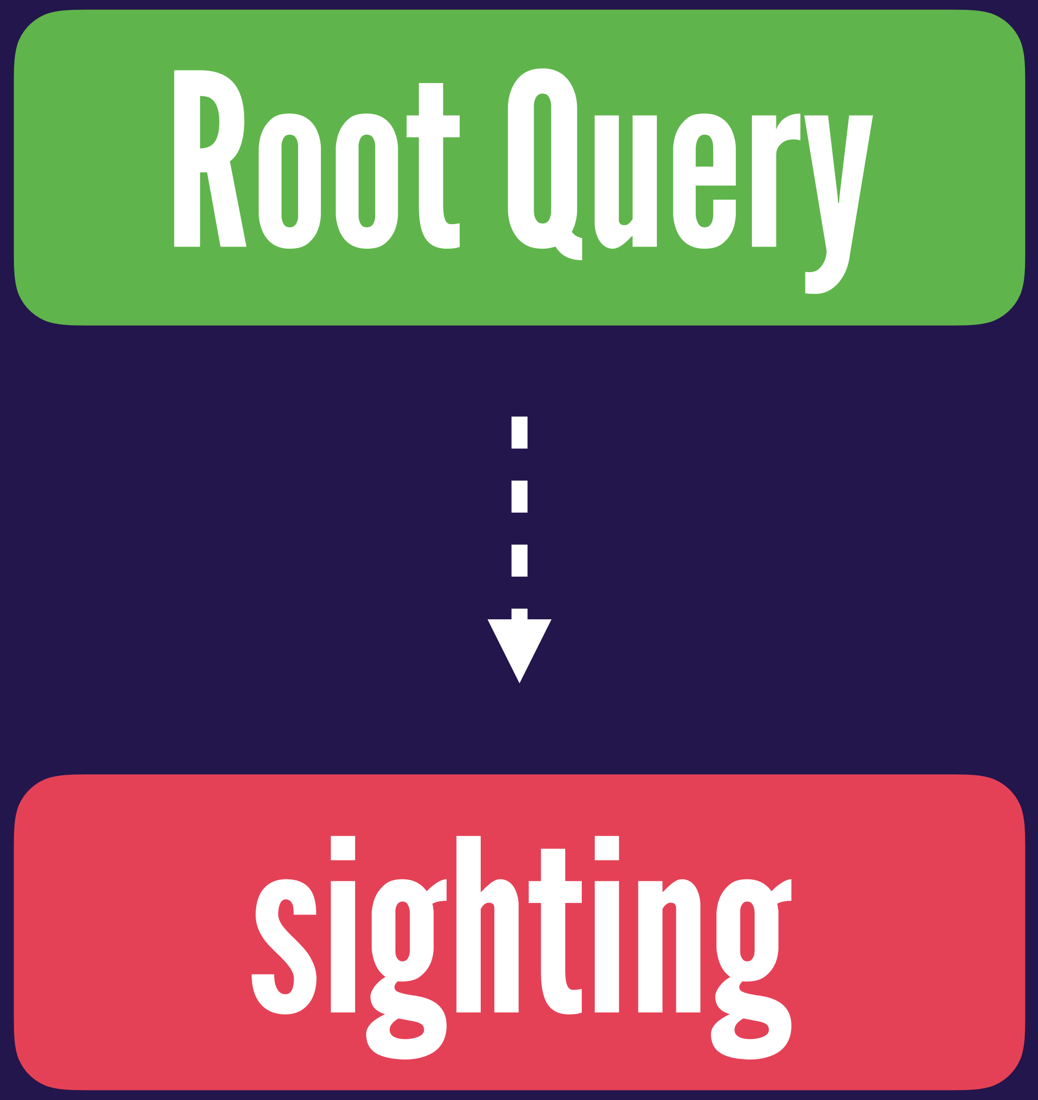
```
query("sighting") {
  resolver { id: Int ->
    storage.getSighting(id)
  }
}
```

Resolver

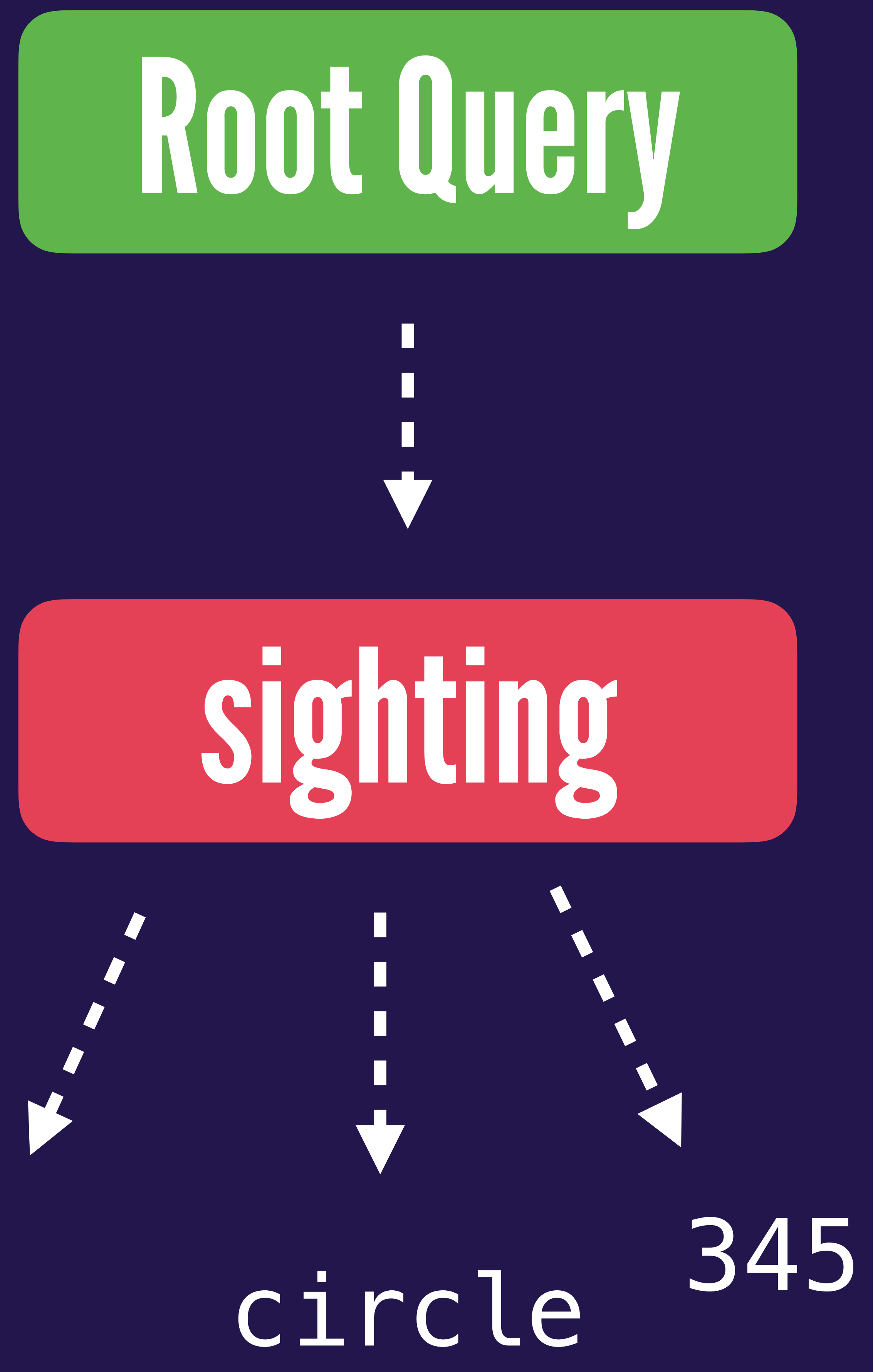
```
query("sighting") {
  resolver { id: Int ->
    “http://sightings/$id".httpGet()
  }
}
```

```
{
  sighting(id: 45) {
    id
    shape
    user {
      id
      name
    }
  }
}
```

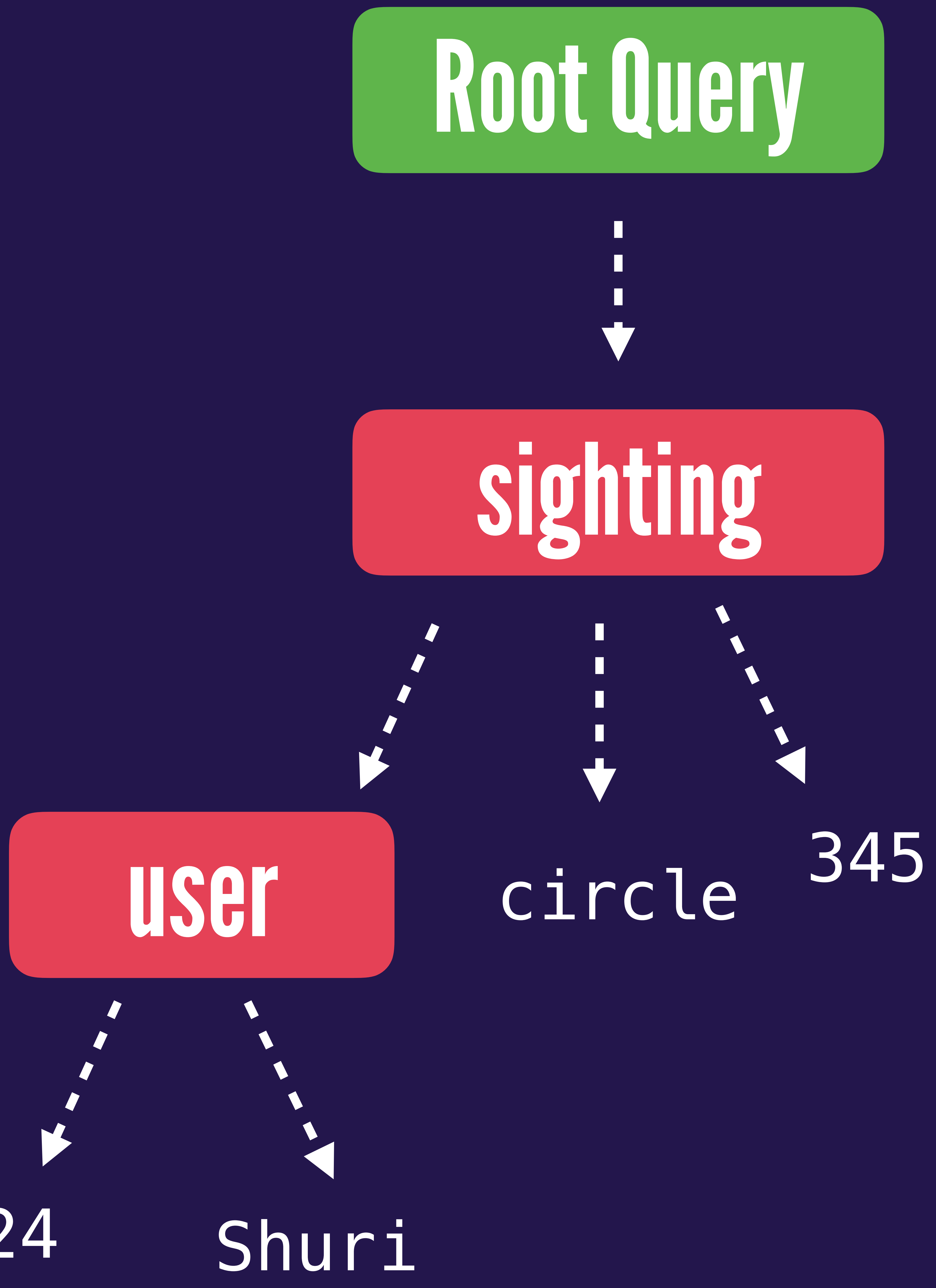
```
{
  sighting(id: 45) {
    id
    shape
    user {
      id
      name
    }
  }
}
```



```
{
  sighting(id: 45) {
    id
    shape
    user {
      id
      name
    }
  }
}
```



```
{
  sighting(id: 45) {
    id
    shape
    user {
      id
      name
    }
  }
}
```



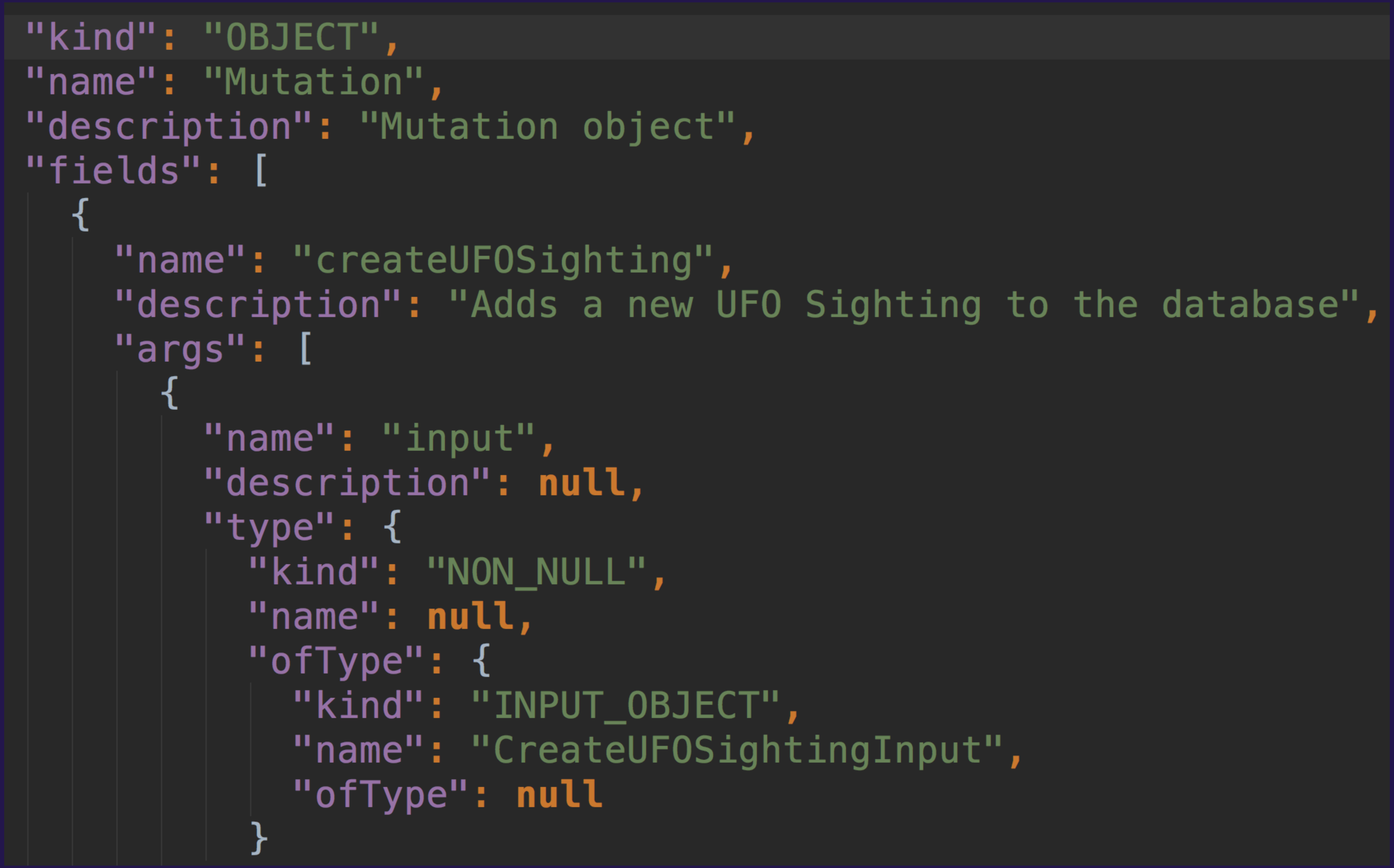
Resolver - Mutation

```
mutation("createUFOSighting") {
   description = "Adds a new UFO Sighting”




}
```

Resolver - Mutation

```
mutation("createUFOSighting") {
  description = "Adds a new UFO Sighting”

  resolver { input: CreateUFOSightingInput ->
    storage.createSighting(input…)
  }
}
```

Resolver - schema.json



```
"kind": "OBJECT",
"name": "Mutation",
"description": "Mutation object",
"fields": [
  {
    "name": "createUFOSighting",
    "description": "Adds a new UFO Sighting to the database",
    "args": [
      {
        "name": "input",
        "description": null,
        "type": {
          "kind": "NON_NULL",
          "name": null,
          "ofType": {
            "kind": "INPUT_OBJECT",
            "name": "CreateUFOSightingInput",
            "ofType": null
          }
```

/graphql Endpoint

```
fun Route.graphql(…) {
  post<GraphQLRequest> {
    val request = call.receive<GraphQLRequest>()

    val query = request.query
    val variables = gson.toJson(request.variables)

    val result = schema.execute(query, variables)
    call.respondText(result)
  }
}
```

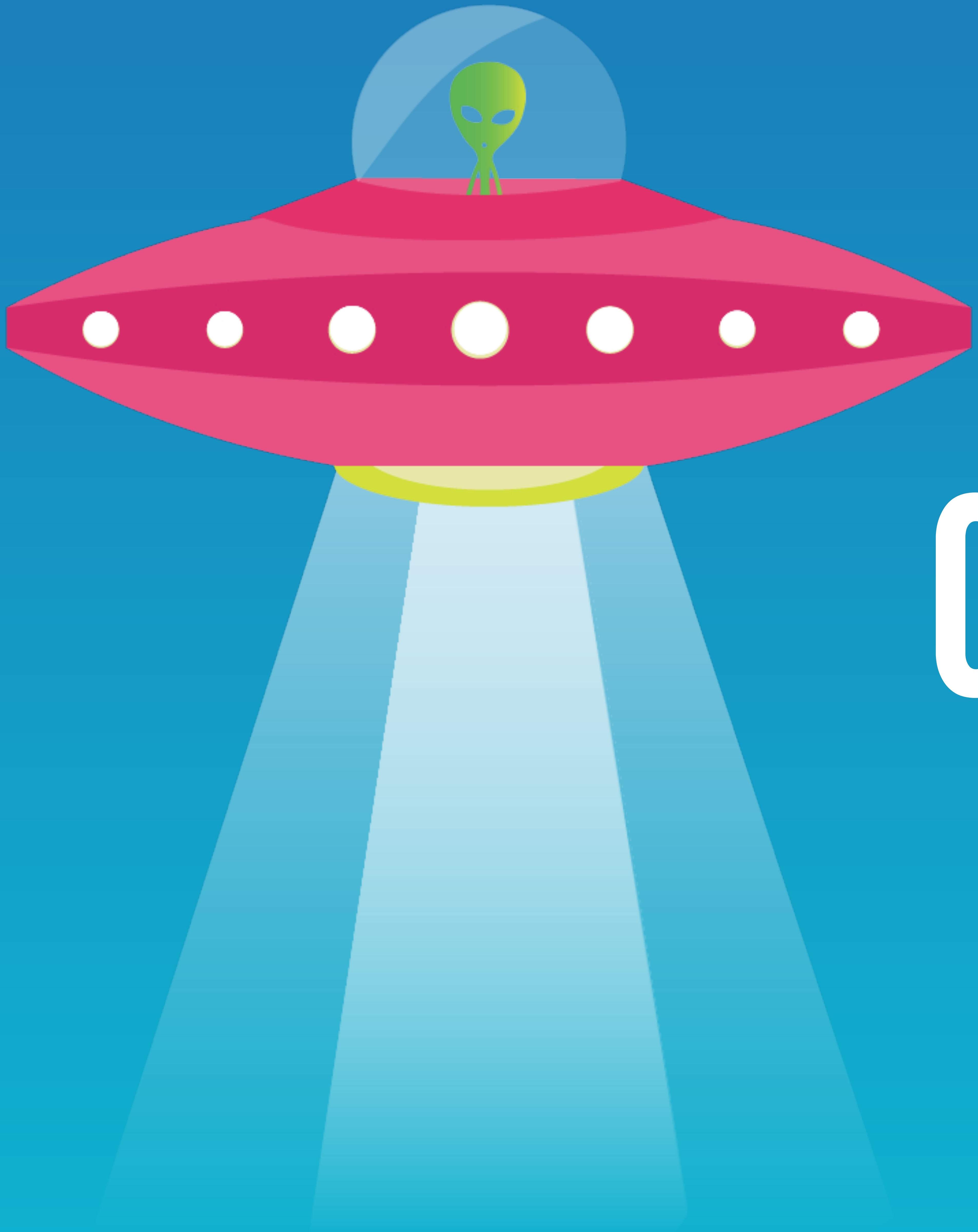
/graphql Endpoint

```
fun Route.graphql(…) {
  post<GraphQLRequest> {
    val request = call.receive<GraphQLRequest>()

    val query = request.query
    val variables = gson.toJson(request.variables)

    val result = schema.execute(query, variables)
    call.respondText(result)
  }
}
```

resolvers

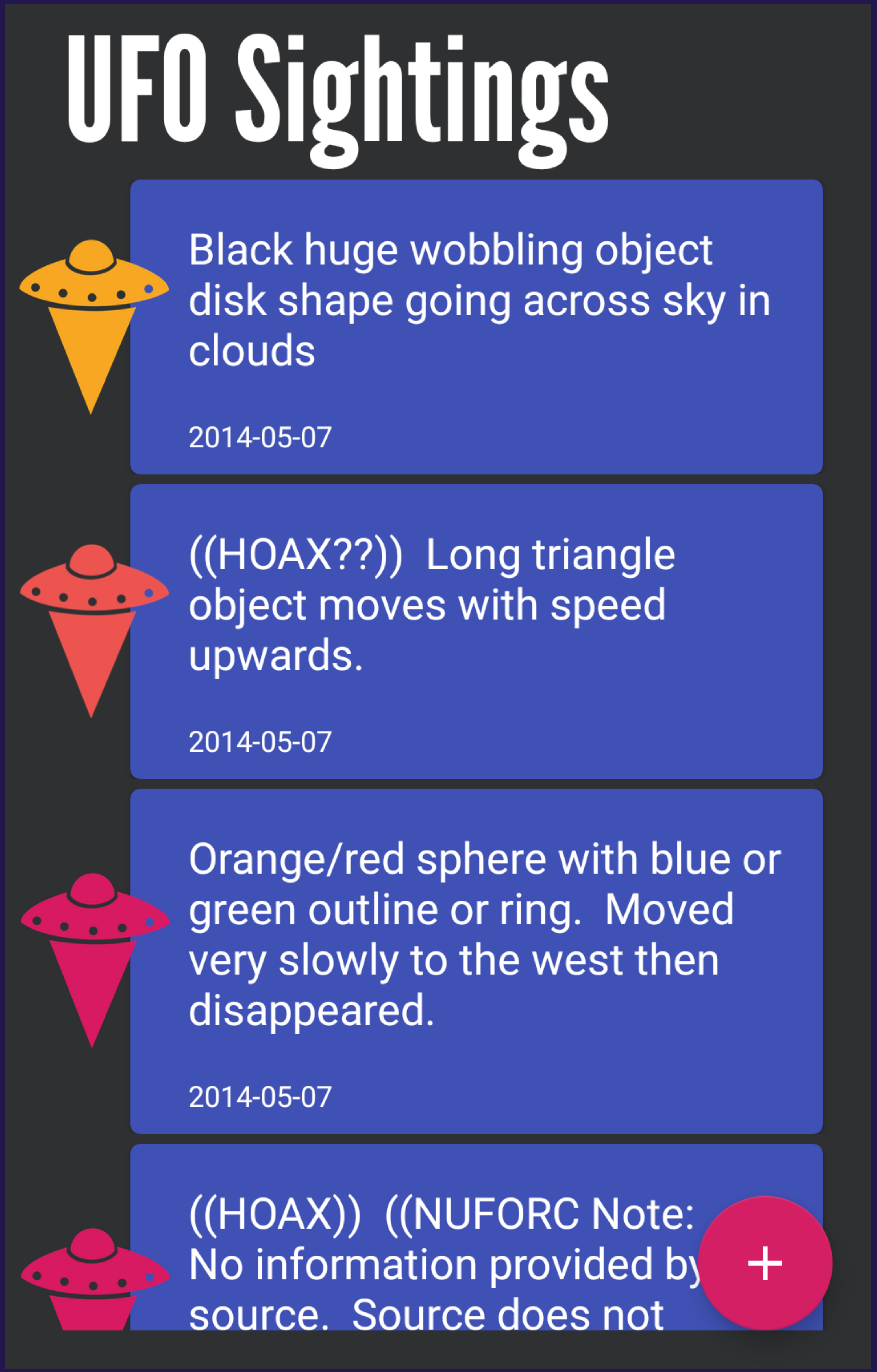
schema

/graphql



CLIENT

Sample Application


UFO Sightings

Black huge wobbling object disk shape going across sky in clouds

2014-05-07

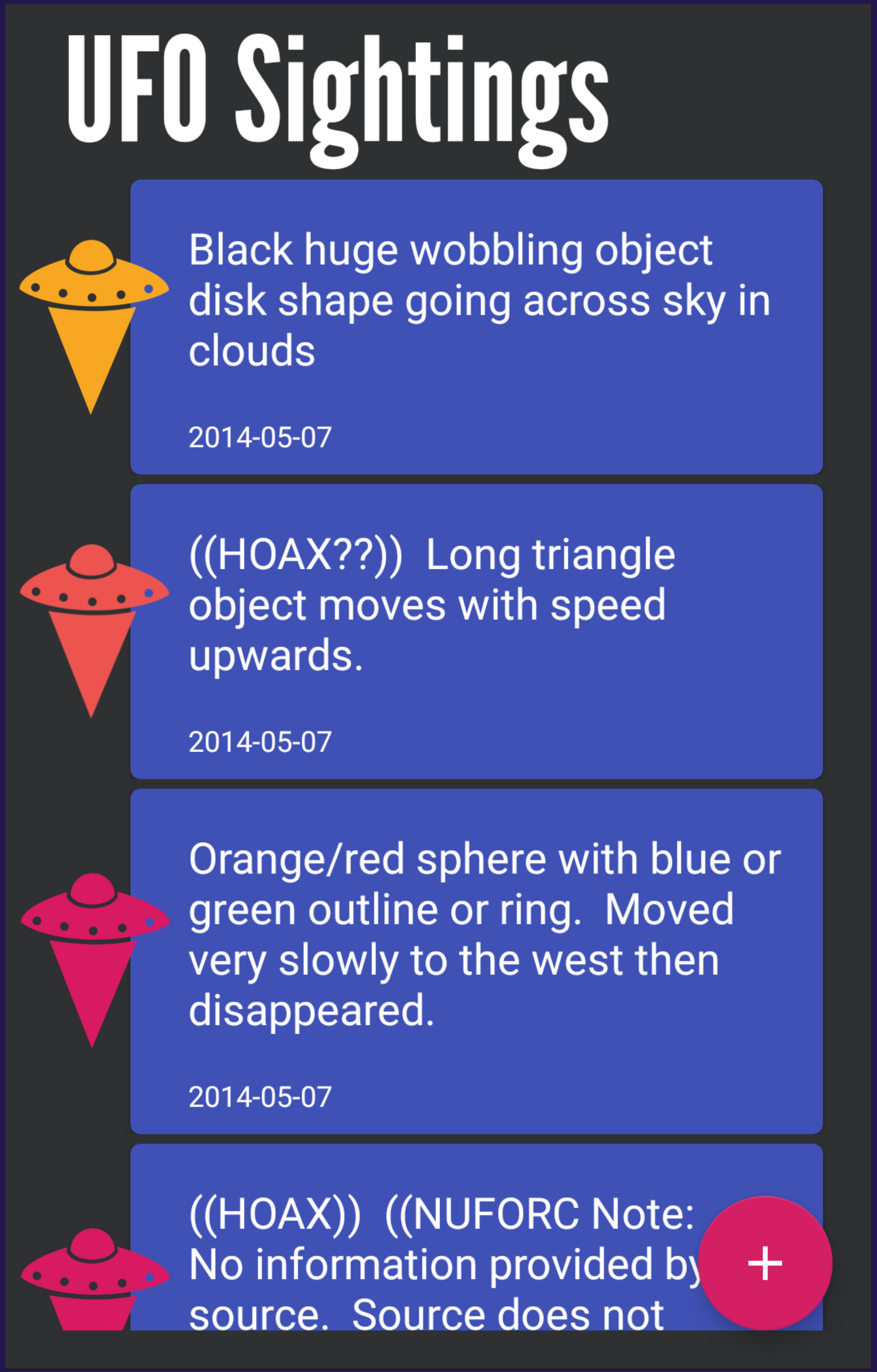
((HOAX??)) Long triangle object moves with speed upwards.

2014-05-07

Orange/red sphere with blue or green outline or ring. Moved very slowly to the west then disappeared.

2014-05-07

((HOAX)) ((NUFORC Note: No information provided by source. Source does not



UFO Sightings

Black huge wobbling object disk shape going across sky in clouds

2014-05-07

((HOAX??)) Long triangle object moves with speed upwards.

2014-05-07

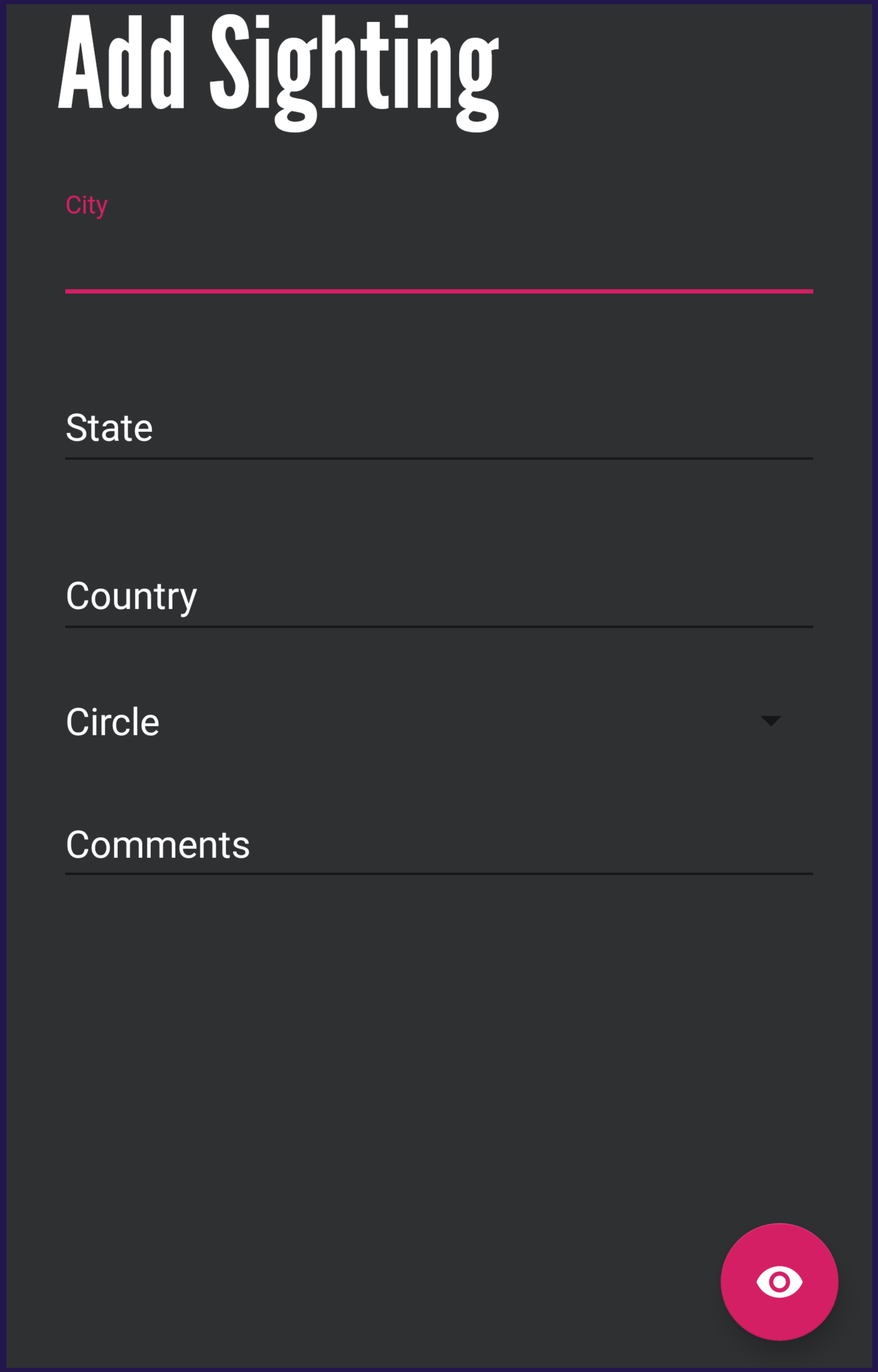
Orange/red sphere with blue or green outline or ring. Moved very slowly to the west then disappeared.

2014-05-07

((HOAX)) ((NUFORC Note: No information provided by source. Source does not

Sample Application

KOTLIN


Add Sighting

City

State

Country

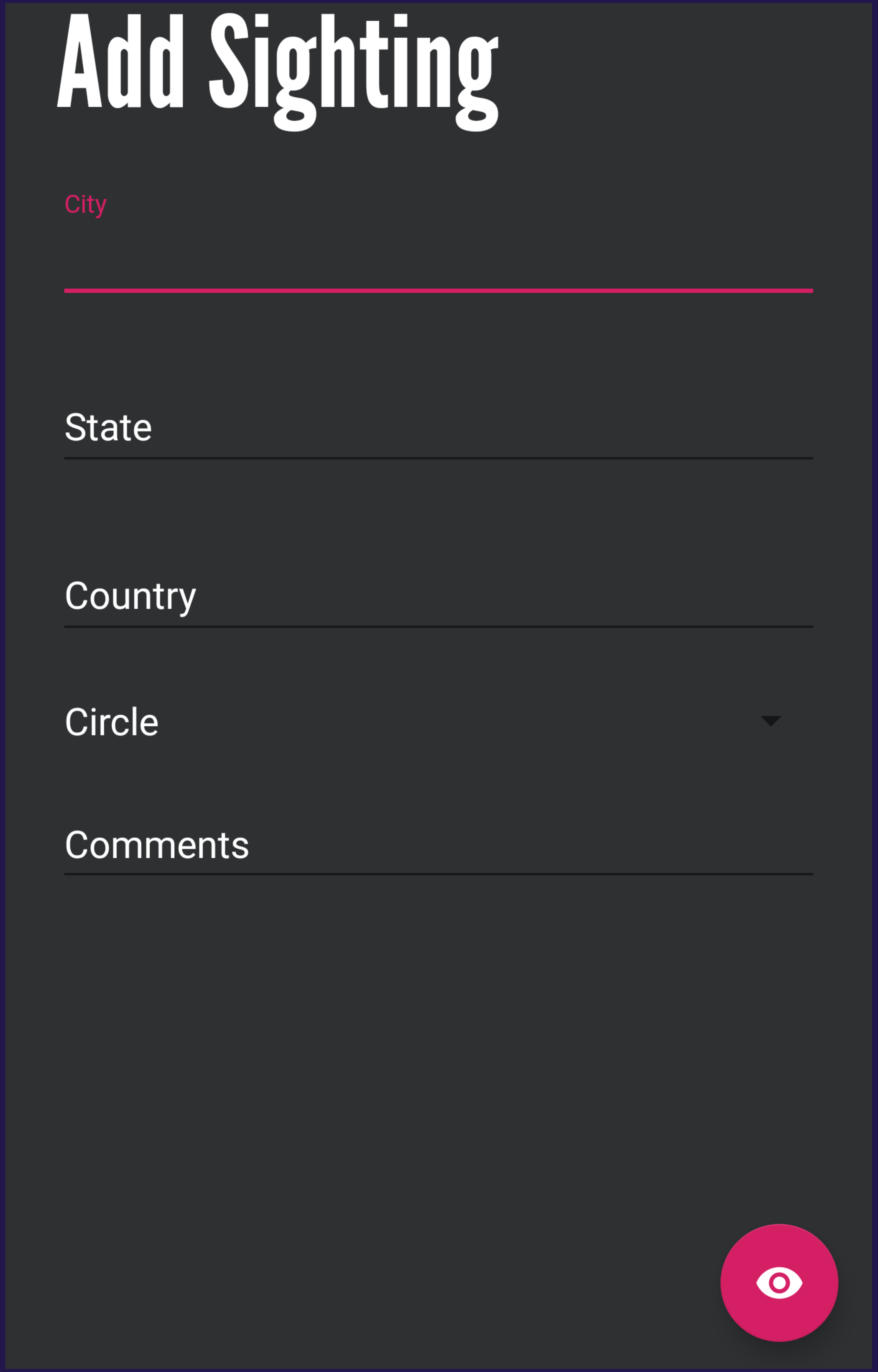
Circle

Comments

Sample Application

KOTLIN

ARCHITECTURE COMPONENTS



Sample Application

KOTLIN

ARCHITECTURE COMPONENTS

APOLLO ANDROID

GraphQL Client...

- ✓ Apollo Client
- ✓ Schema
- ✓ .graphql Files

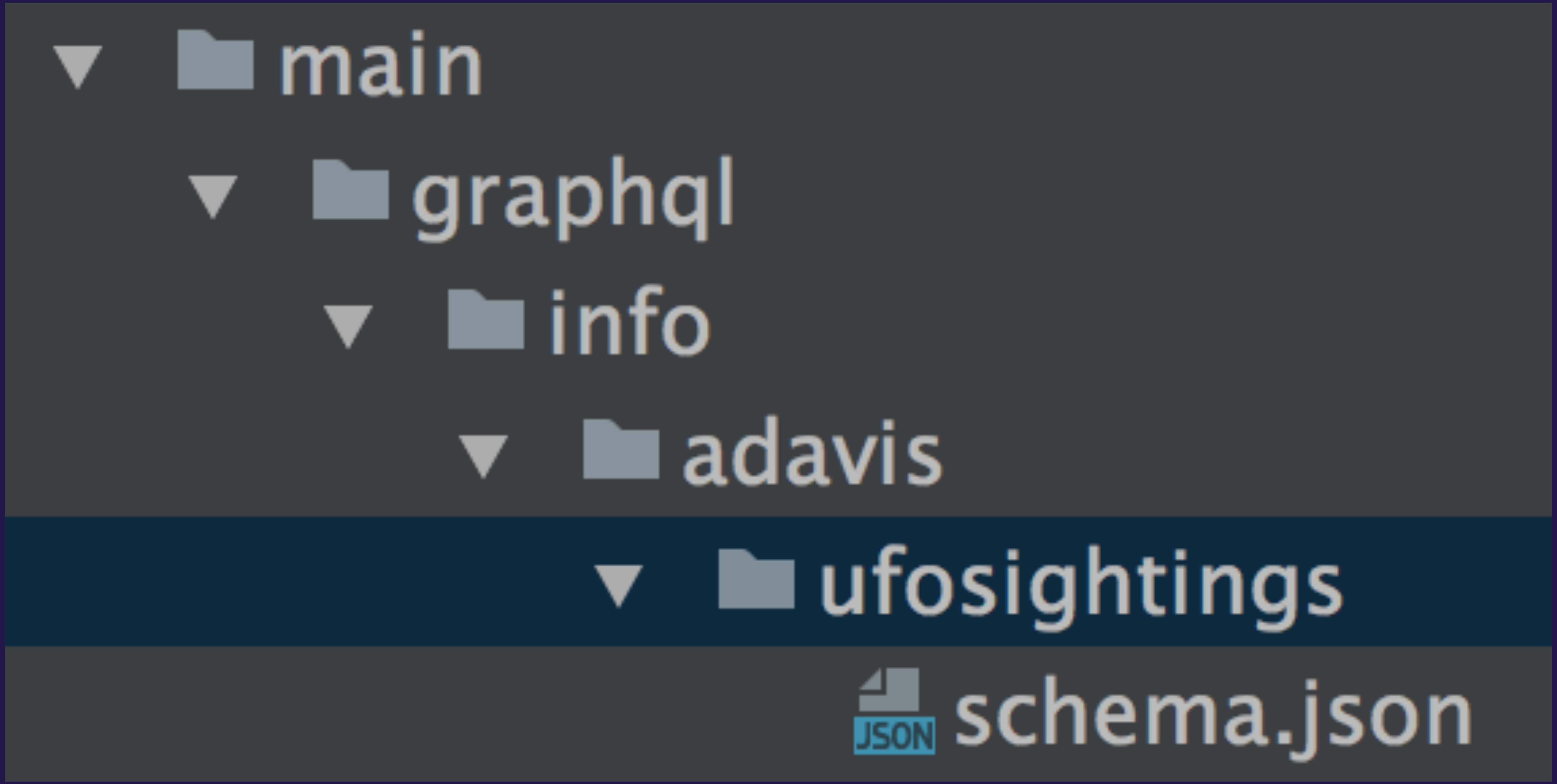
Apollo Client

```
ApolloClient.builder()
            .serverUrl(BASE_URL)
            .okHttpClient(okHttpClient)
            .build()
```

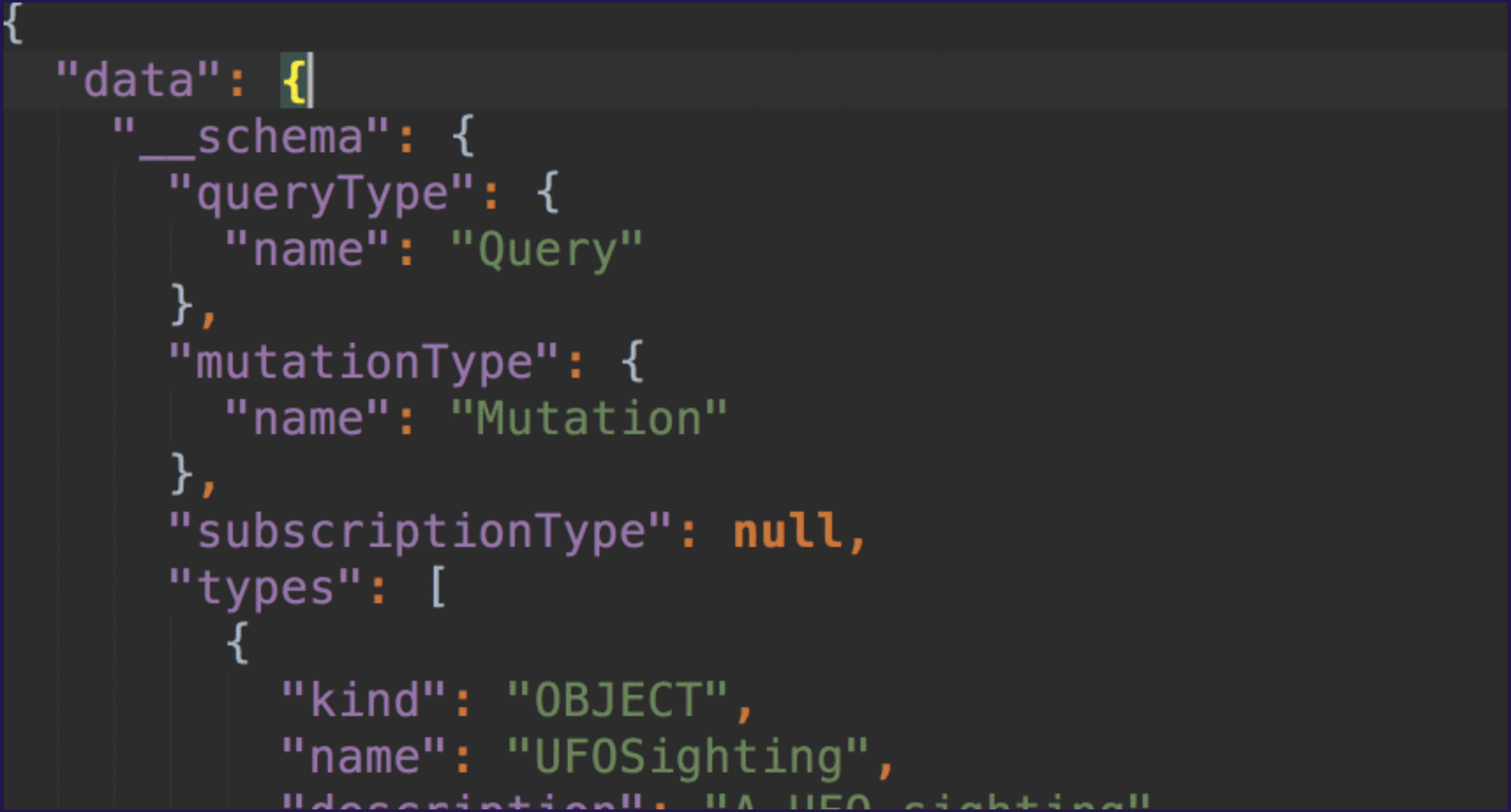
Apollo Client

```
ApolloClient.builder()
            .serverUrl(BASE_URL)
            .okHttpClient(okHttpClient)
            .build()
```

apollo-codegen download-schema

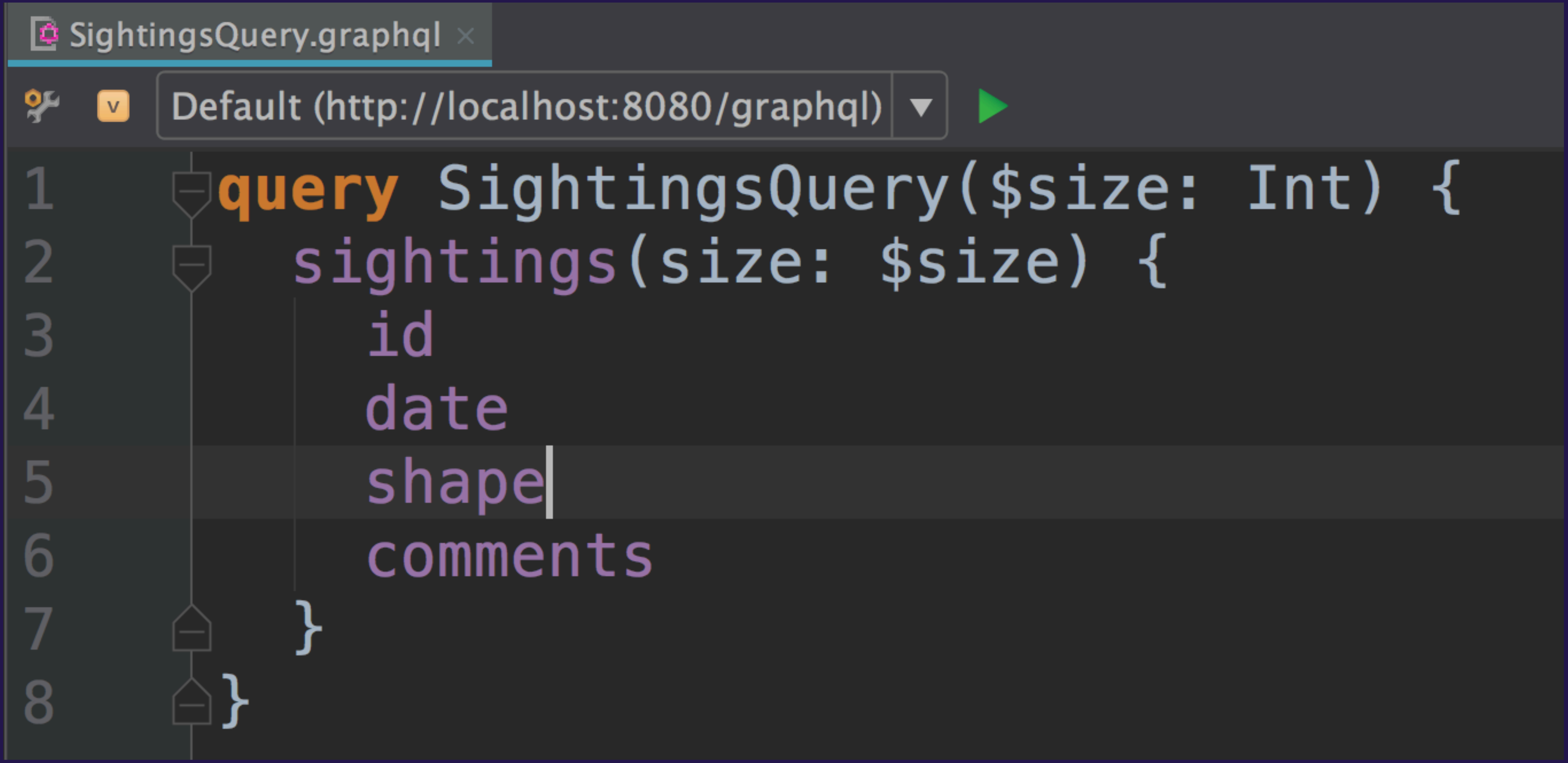


Schema



```
{
  "data": {
    "__schema": {
      "queryType": {
        "name": "Query"
      },
      "mutationType": {
        "name": "Mutation"
      },
      "subscriptionType": null,
      "types": [
        {
          "kind": "OBJECT",
          "name": "UFOSighting",
          "description": "A UFO sighting"
```

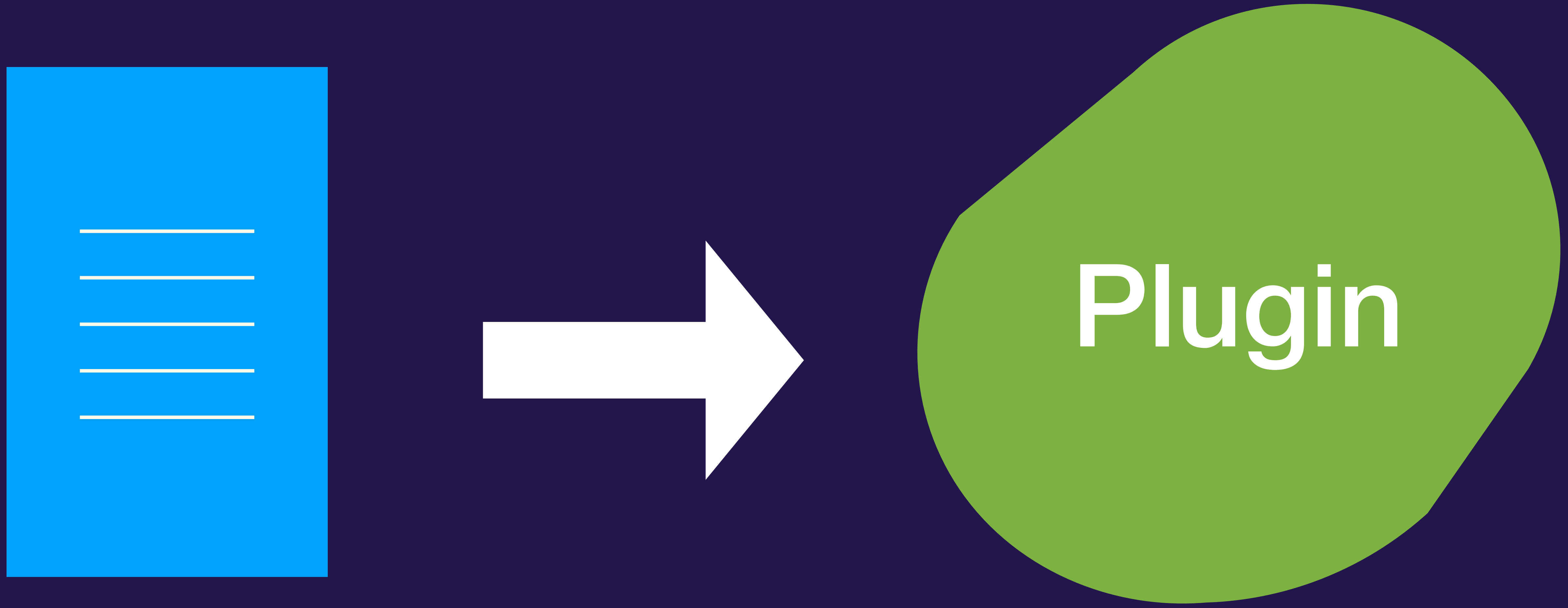
.graphql File



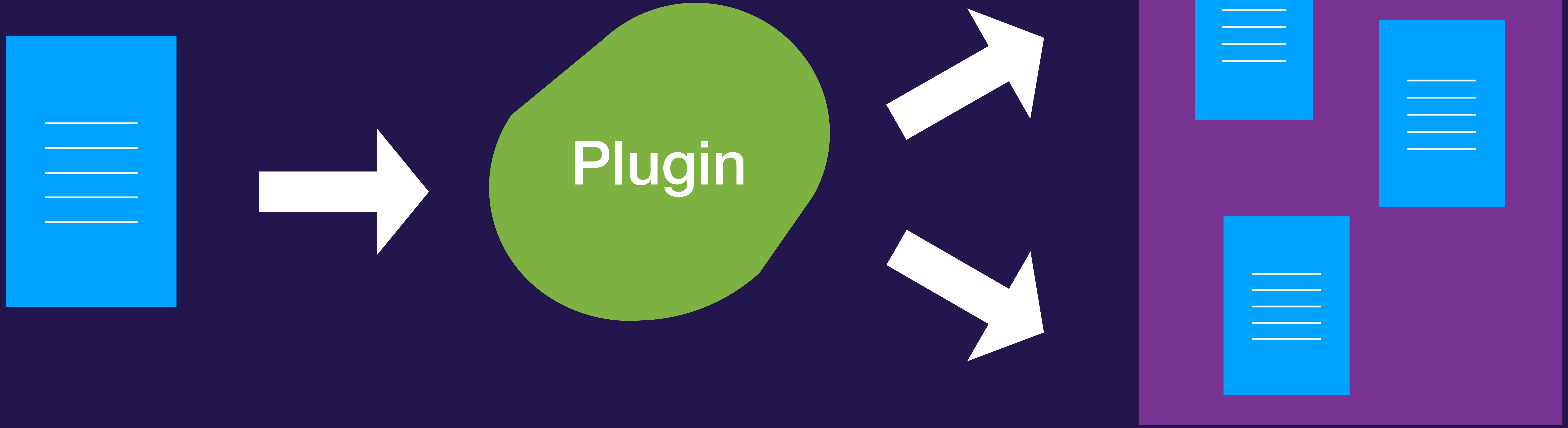
```graphql
query SightingsQuery($size: Int) {
  sightings(size: $size) {
    id
    date
    shape
    comments
  }
}
```

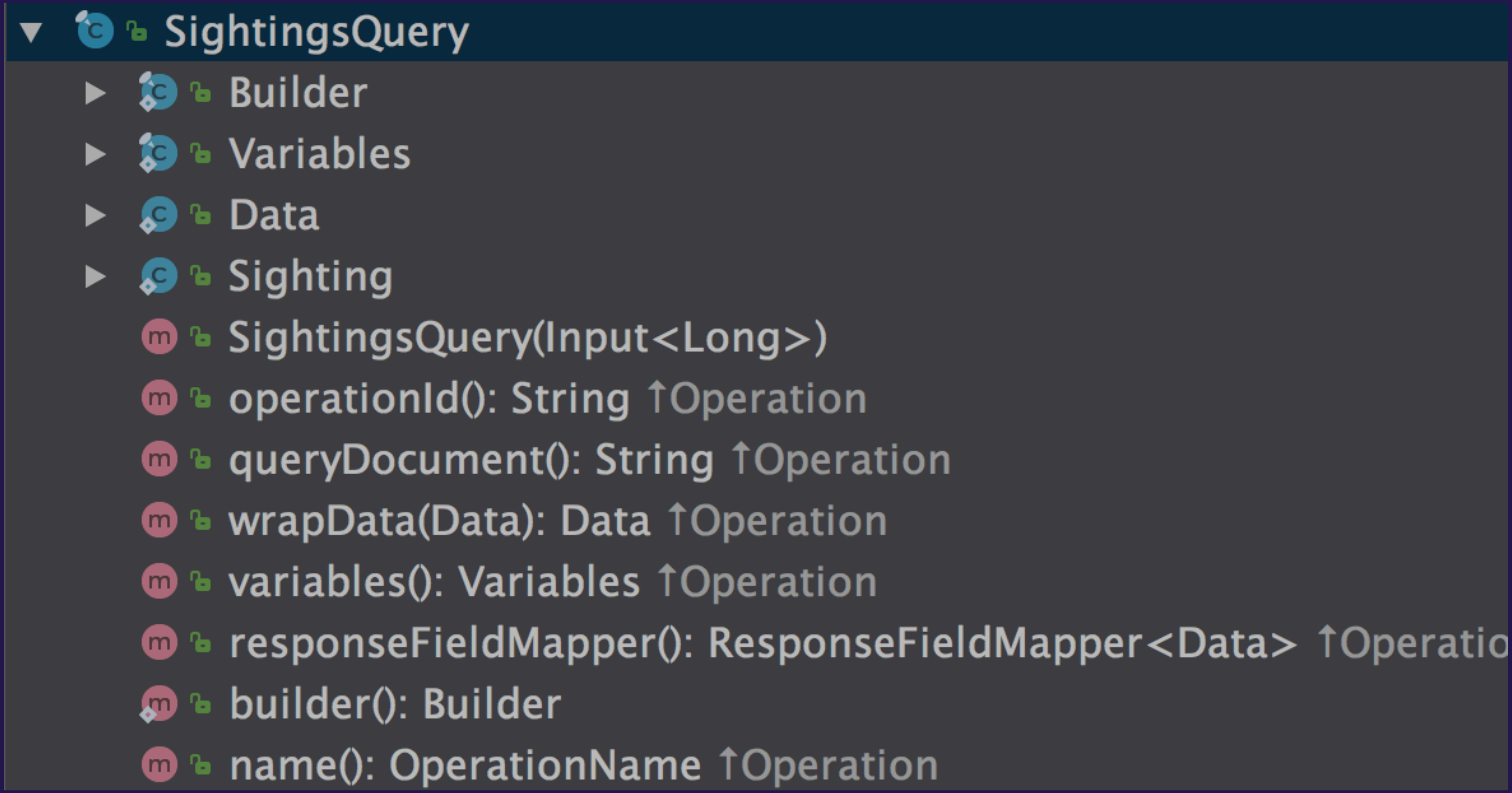
.graphql File

.graphql File



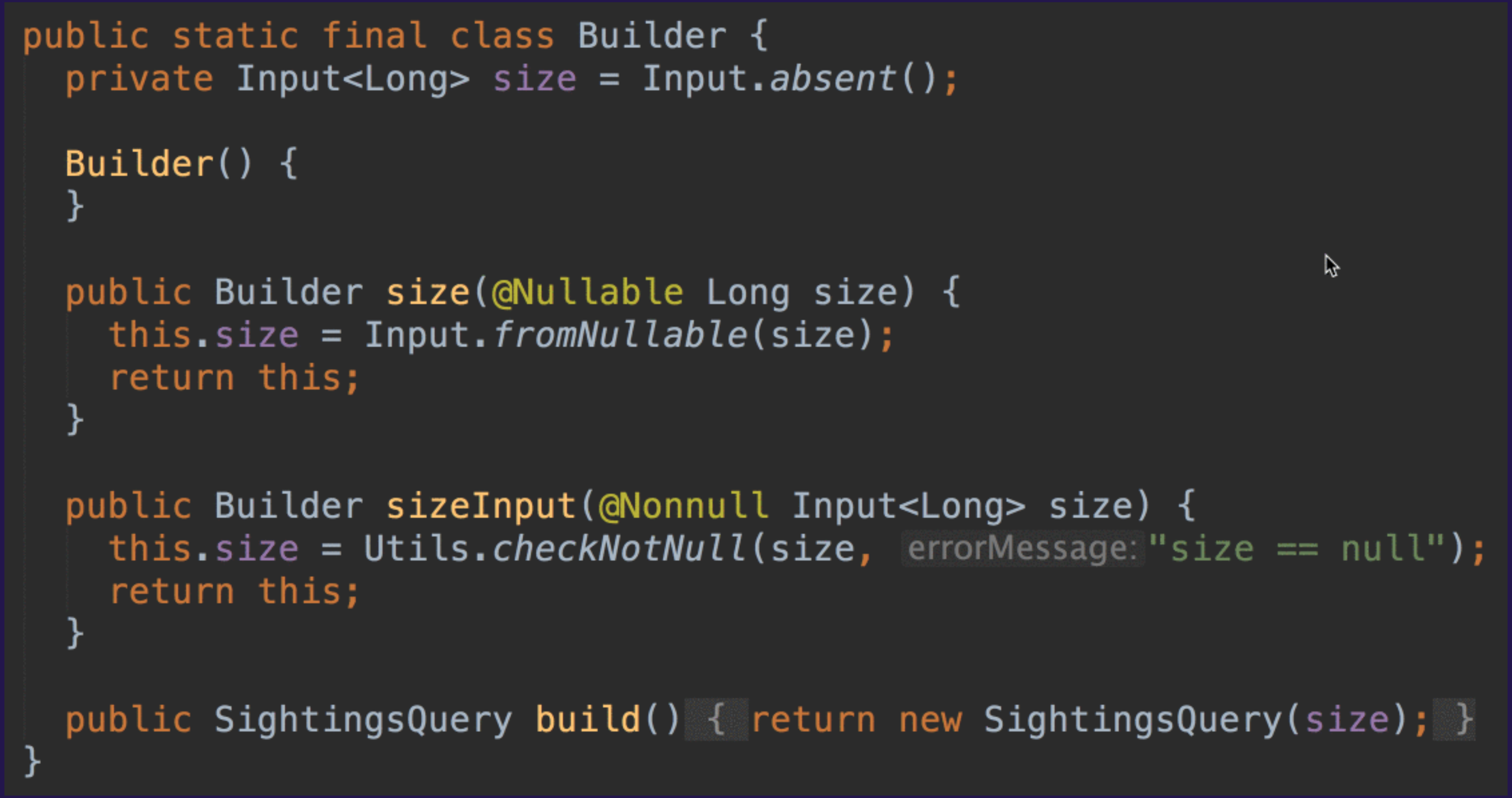
.graphql File





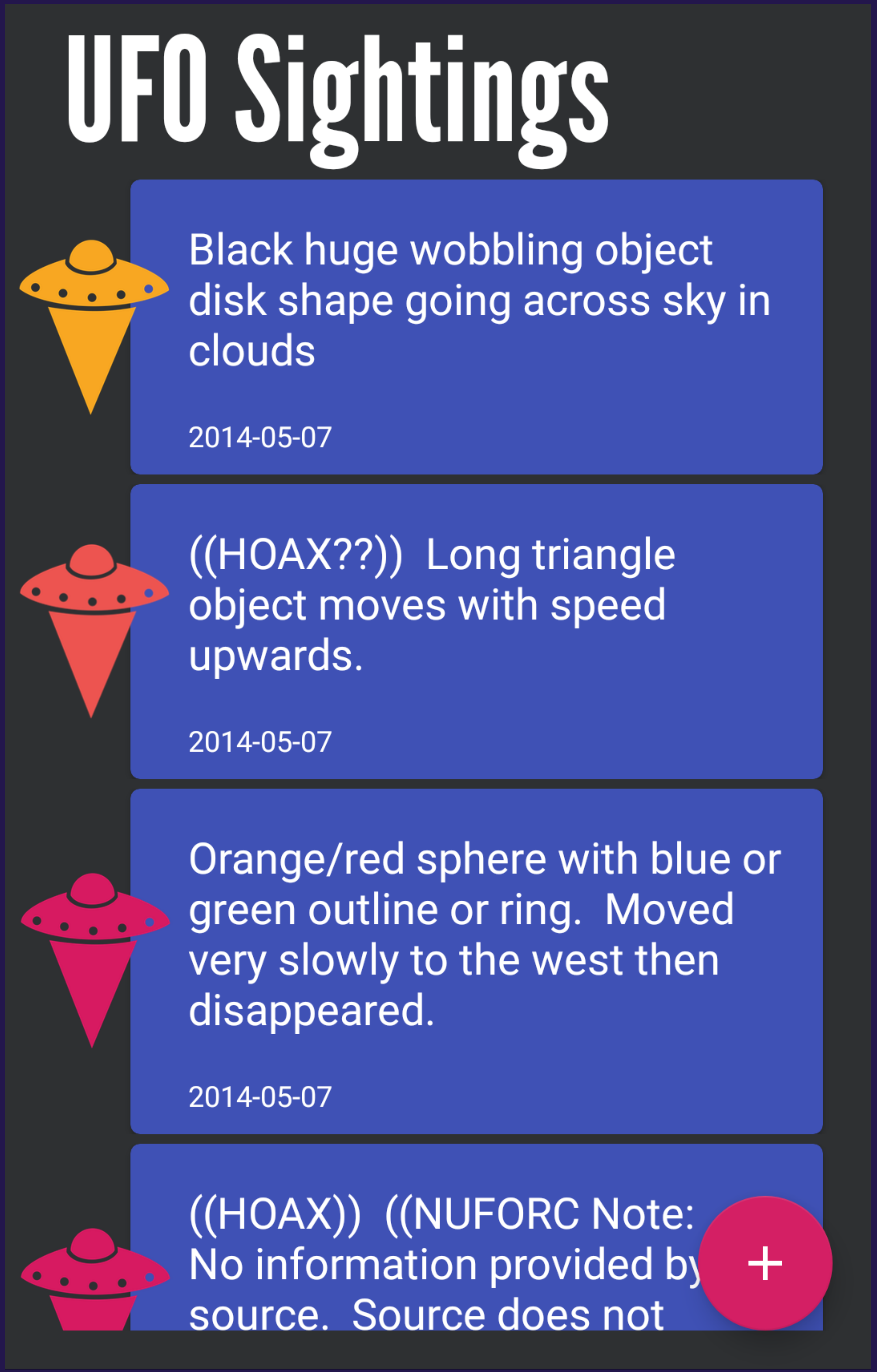
```
▼ SightingsQuery
    ▶ Builder
    ▶ Variables
    ▶ Data
    ▶ Sighting
      SightingsQuery(Input<Long>)
      operationId(): String ↑Operation
      queryDocument(): String ↑Operation
      wrapData(Data): Data ↑Operation
      variables(): Variables ↑Operation
      responseFieldMapper(): ResponseFieldMapper<Data> ↑Operatio
      builder(): Builder
      name(): OperationName ↑Operation
```

Generated Code...



```
public static final class Builder {
  private Input<Long> size = Input.absent();

  Builder() {
  }

  public Builder size(@Nullable Long size) {
    this.size = Input.fromNullable(size);
    return this;
  }

  public Builder sizeInput(@Nonnull Input<Long> size) {
    this.size = Utils.checkNotNull(size, errorMessage: "size == null");
    return this;
  }

  public SightingsQuery build() { return new SightingsQuery(size); }
}
```

- ✓ Apollo Client
- ✓ Schema
- ✓ .graphql Files


UFO Sightings

Black huge wobbling object disk shape going across sky in clouds

2014-05-07

((HOAX??)) Long triangle object moves with speed upwards.

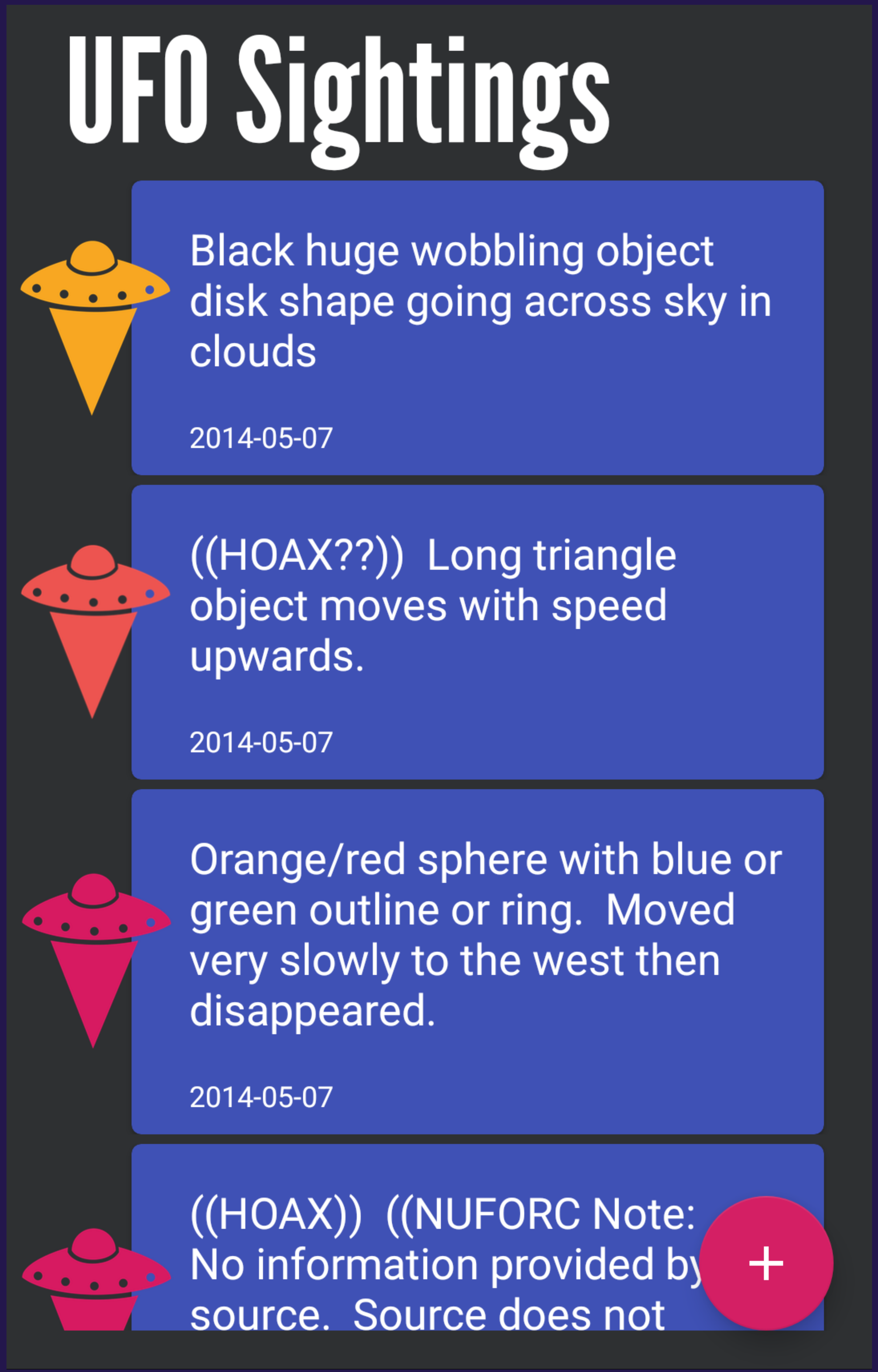
2014-05-07

Orange/red sphere with blue or green outline or ring. Moved very slowly to the west then disappeared.

2014-05-07

((HOAX)) ((NUFORC Note: No information provided by source. Source does not

1. Build our query
2. Enqueue the request
3. Handle the response


UFO Sightings

Black huge wobbling object disk shape going across sky in clouds

2014-05-07

((HOAX??)) Long triangle object moves with speed upwards.

2014-05-07

Orange/red sphere with blue or green outline or ring. Moved very slowly to the west then disappeared.

2014-05-07

((HOAX)) ((NUFORC Note: No information provided by source. Source does not

Build our query

```
SightingsQuery.builder()
              .size(30)
              .build()
```

Enqueue the request

```
apolloClient
    .query(query)
```

Enqueue the request

```
apolloClient
    .query(query)
    .enqueue(object : Callback<T>() {





})
```

Enqueue the request

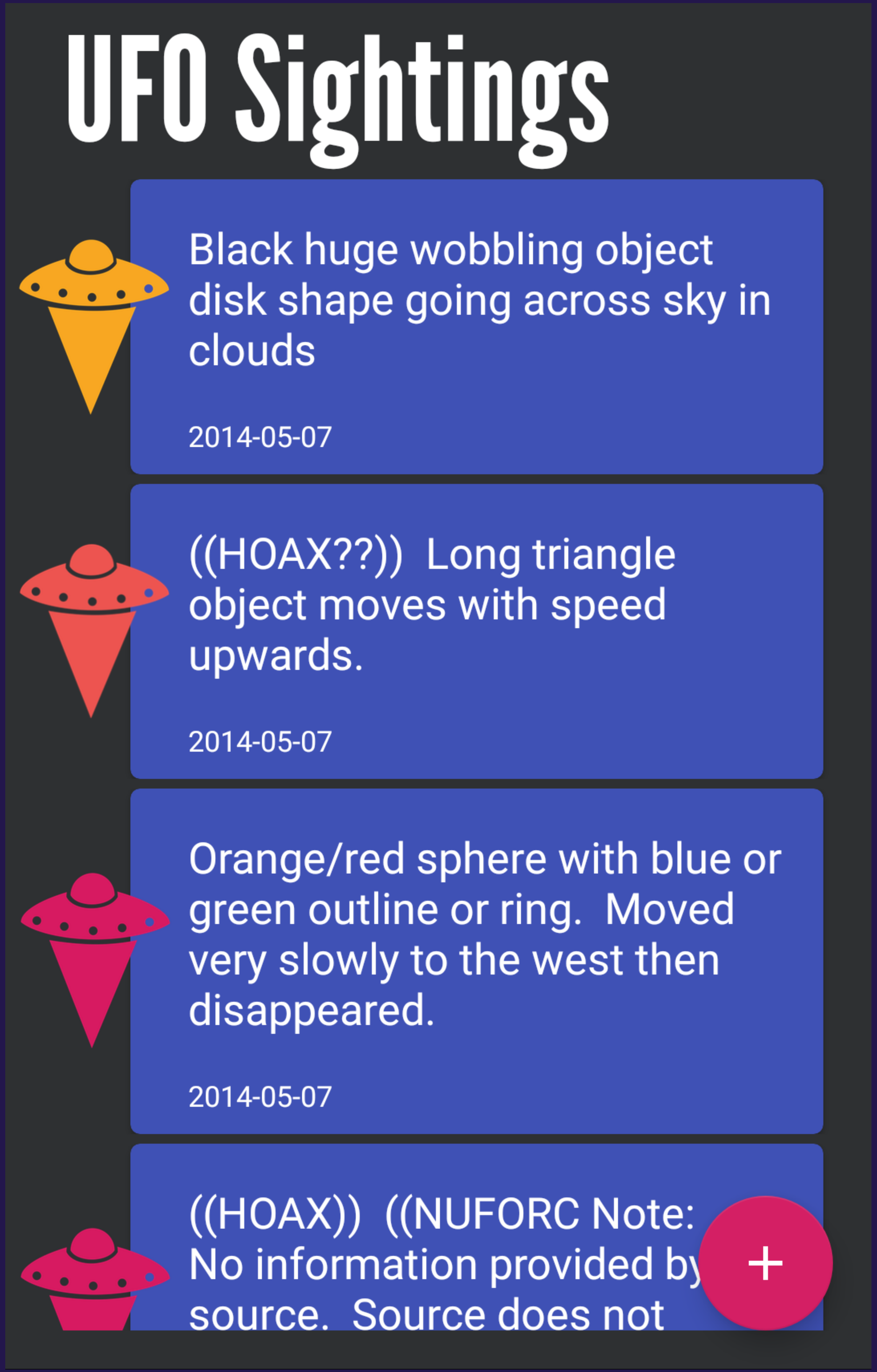
```
apolloClient
    .query(query)
    .enqueue(object : Callback<T>() {

      fun onResponse(response: Response<T>)

      fun onFailure(e: ApolloException)

})
```

Handle the response

```
{
   "data" : {
      "sightings" : [ {
         "__typename" : "UFOSighting",
         "id" : 9297,
         "date" : "2014-05-08",
         "shape" : "rectangle",
         "comments" : "Standing at my window one by one."
      } ]
   }
}
```

Handle the response

```
fun onResponse(response: Response<T>) {
    response.data()?.sightings()

    // notify your UI
}
```



UFO Sightings

Black huge wobbling object disk shape going across sky in clouds

2014-05-07

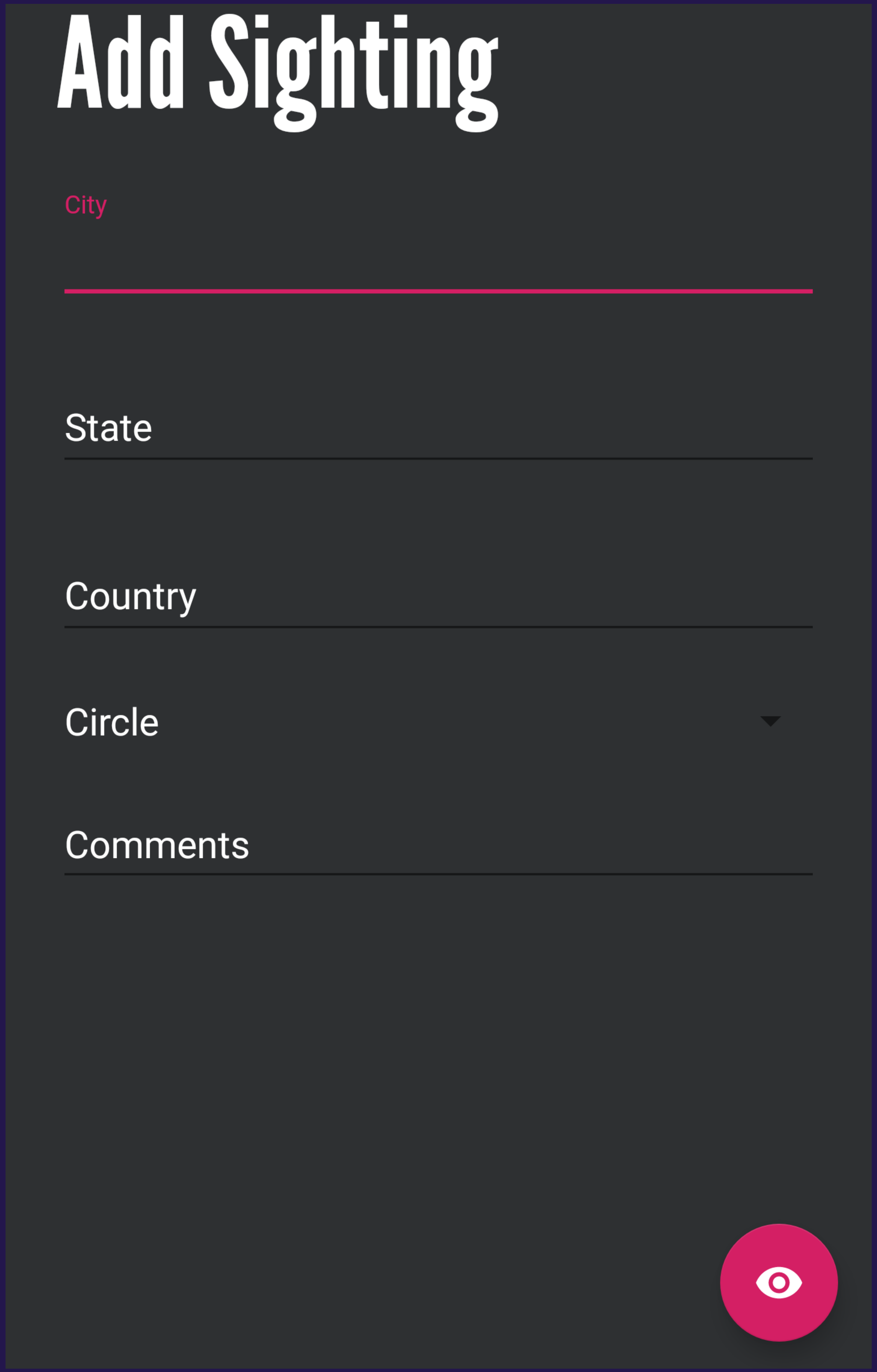
((HOAX??)) Long triangle object moves with speed upwards.

2014-05-07

Orange/red sphere with blue or green outline or ring. Moved very slowly to the west then disappeared.

2014-05-07

((HOAX)) ((NUFORC Note: No information provided by source. Source does not



Add Sighting

City

State

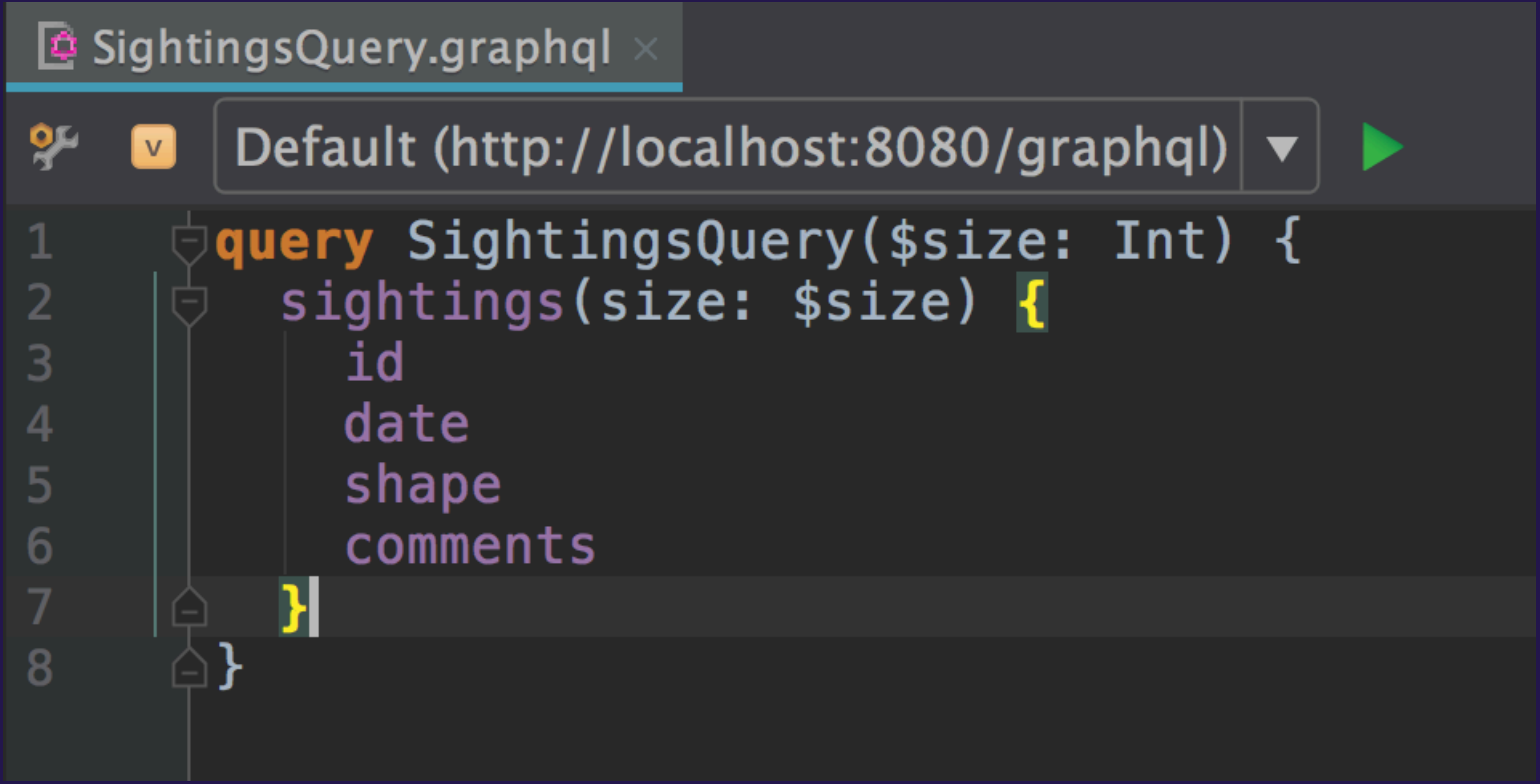
Country

Circle

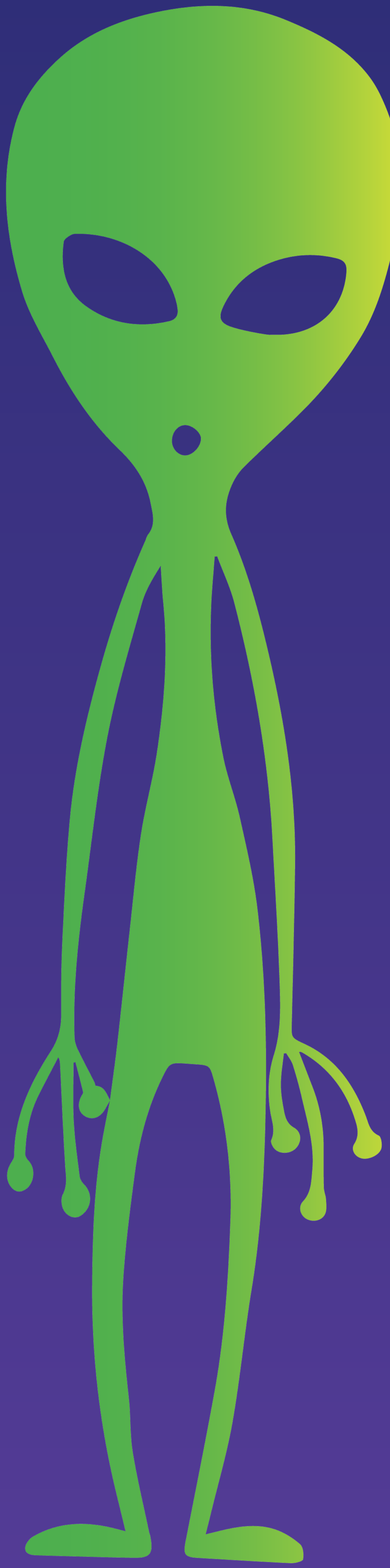
Comments

TIPS & TRICKS

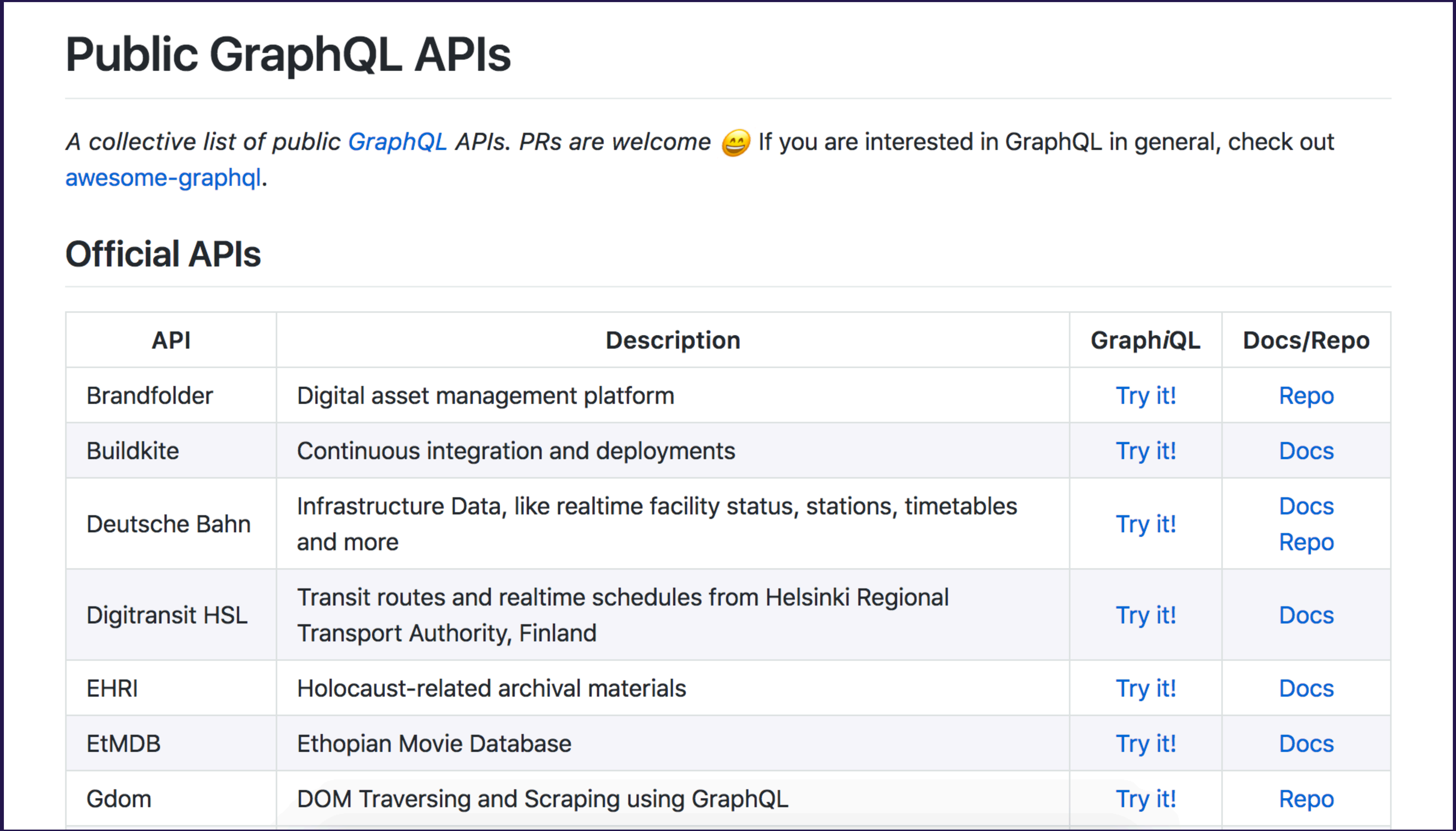
IntelliJ GraphQL Plugin...



```graphql
query SightingsQuery($size: Int) {
  sightings(size: $size) {
    id
    date
    shape
    comments
  }
}
```



A CLIENT IS OPTIONAL



Public GraphQL APIs

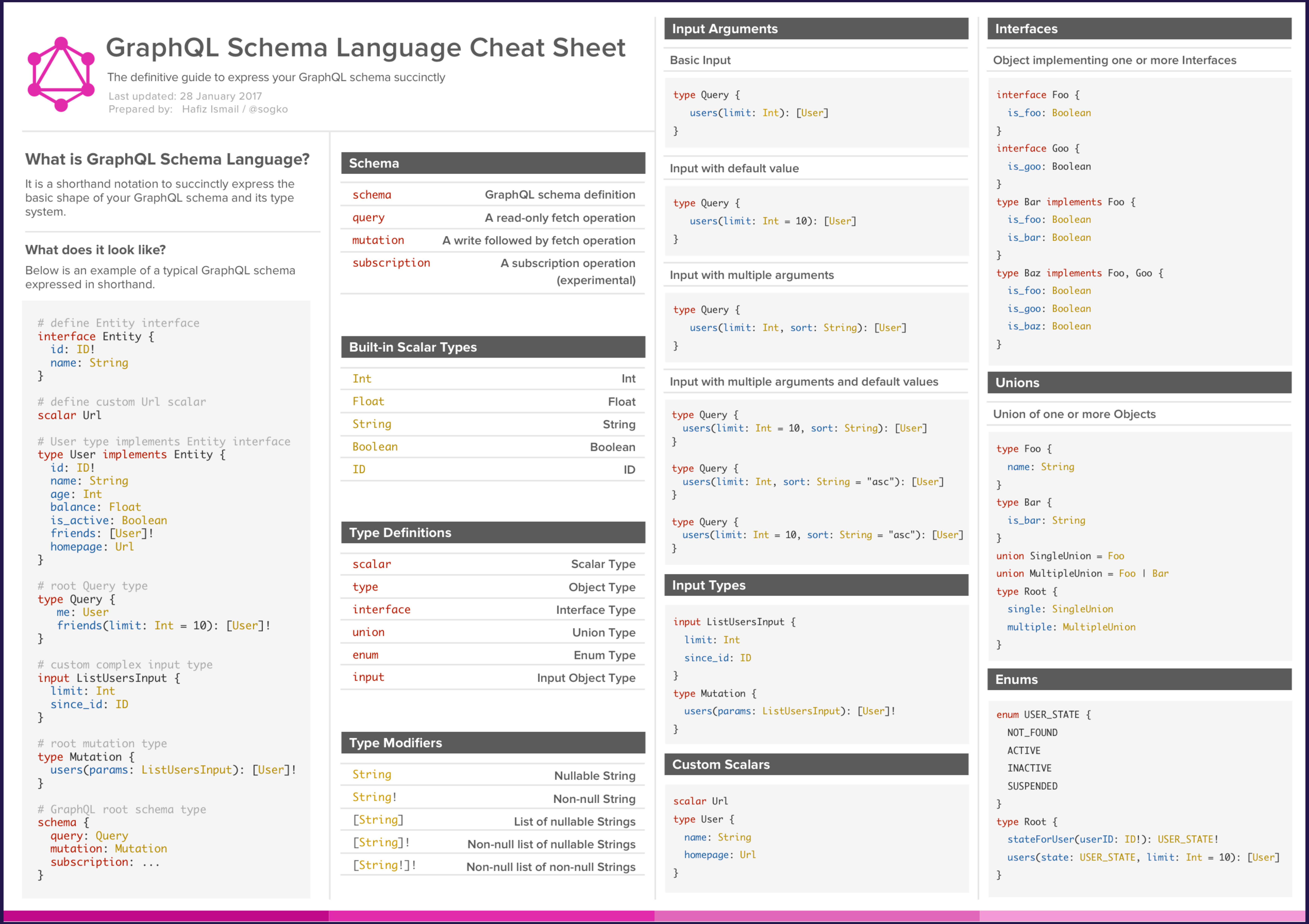
A collective list of public GraphQL APIs. PRs are welcome 😄 If you are interested in GraphQL in general, check out awesome-graphql.

Official APIs

API	Description	Graph*i*QL	Docs/Repo
Brandfolder	Digital asset management platform	Try it!	Repo
Buildkite	Continuous integration and deployments	Try it!	Docs
Deutsche Bahn	Infrastructure Data, like realtime facility status, stations, timetables and more	Try it!	Docs Repo
Digitransit HSL	Transit routes and realtime schedules from Helsinki Regional Transport Authority, Finland	Try it!	Docs
EHRI	Holocaust-related archival materials	Try it!	Docs
EtMDB	Ethopian Movie Database	Try it!	Docs
Gdom	DOM Traversing and Scraping using GraphQL	Try it!	Repo

HTTPS://GITHUB.COM/APIS-GURU/GRAPHQL-APIS

@BRWNGRLDEV



GraphQL Schema Language Cheat Sheet

The definitive guide to express your GraphQL schema succinctly

Last updated: 28 January 2017
Prepared by: Hafiz Ismail / @sogko

What is GraphQL Schema Language?

It is a shorthand notation to succinctly express the basic shape of your GraphQL schema and its type system.

What does it look like?

Below is an example of a typical GraphQL schema expressed in shorthand.

```
# define Entity interface
interface Entity {
  id: ID!
  name: String
}

# define custom Url scalar
scalar Url

# User type implements Entity interface
type User implements Entity {
  id: ID!
  name: String
  age: Int
  balance: Float
  is_active: Boolean
  friends: [User]!
  homepage: Url
}

# root Query type
type Query {
  me: User
  friends(limit: Int = 10): [User]!
}

# custom complex input type
input ListUsersInput {
  limit: Int
  since_id: ID
}

# root mutation type
type Mutation {
  users(params: ListUsersInput): [User]!
}

# GraphQL root schema type
schema {
  query: Query
  mutation: Mutation
  subscription: ...
}
```

Schema

schema	GraphQL schema definition
query	A read-only fetch operation
mutation	A write followed by fetch operation
subscription	A subscription operation (experimental)

Built-in Scalar Types

Int	Int
Float	Float
String	String
Boolean	Boolean
ID	ID

Type Definitions

scalar	Scalar Type
type	Object Type
interface	Interface Type
union	Union Type
enum	Enum Type
input	Input Object Type

Type Modifiers

String	Nullable String
String!	Non-null String
[String]	List of nullable Strings
[String]!	Non-null list of nullable Strings
[String!]!	Non-null list of non-null Strings

Input Arguments

Basic Input

```
type Query {
  users(limit: Int): [User]
}
```

Input with default value

```
type Query {
  users(limit: Int = 10): [User]
}
```

Input with multiple arguments

```
type Query {
  users(limit: Int, sort: String): [User]
}
```

Input with multiple arguments and default values

```
type Query {
  users(limit: Int = 10, sort: String): [User]
}

type Query {
  users(limit: Int, sort: String = "asc"): [User]
}

type Query {
  users(limit: Int = 10, sort: String = "asc"): [User]
}
```

Input Types

```
input ListUsersInput {
  limit: Int
  since_id: ID
}
type Mutation {
  users(params: ListUsersInput): [User]!
}
```

Custom Scalars

```
scalar Url
type User {
  name: String
  homepage: Url
}
```

Interfaces

Object implementing one or more Interfaces

```
interface Foo {
  is_foo: Boolean
}
interface Goo {
  is_goo: Boolean
}
type Bar implements Foo {
  is_foo: Boolean
  is_bar: Boolean
}
type Baz implements Foo, Goo {
  is_foo: Boolean
  is_goo: Boolean
  is_baz: Boolean
}
```

Unions

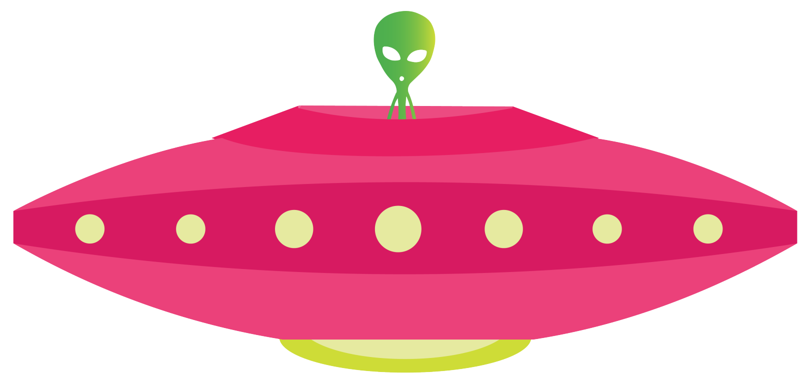
Union of one or more Objects

```
type Foo {
  name: String
}
type Bar {
  is_bar: String
}
union SingleUnion = Foo
union MultipleUnion = Foo | Bar
type Root {
  single: SingleUnion
  multiple: MultipleUnion
}
```

Enums

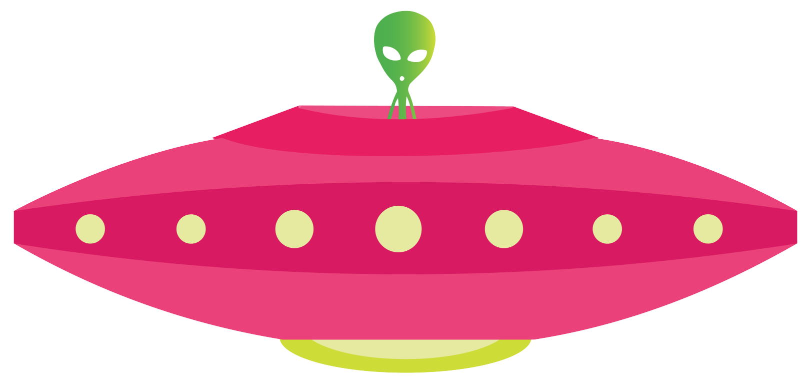
```
enum USER_STATE {
  NOT_FOUND
  ACTIVE
  INACTIVE
  SUSPENDED
}
type Root {
  stateForUser(userID: ID!): USER_STATE!
  users(state: USER_STATE, limit: Int = 10): [User]
}
```

HTTPS://GITHUB.COM/SOGKO/GRAPHQL-SCHEMA-LANGUAGE-CHEAT-SHEET

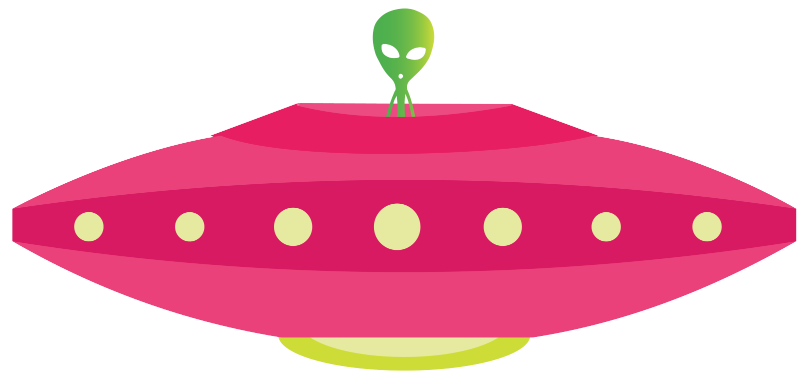


TRUE/FALSE

RAPID FIRE

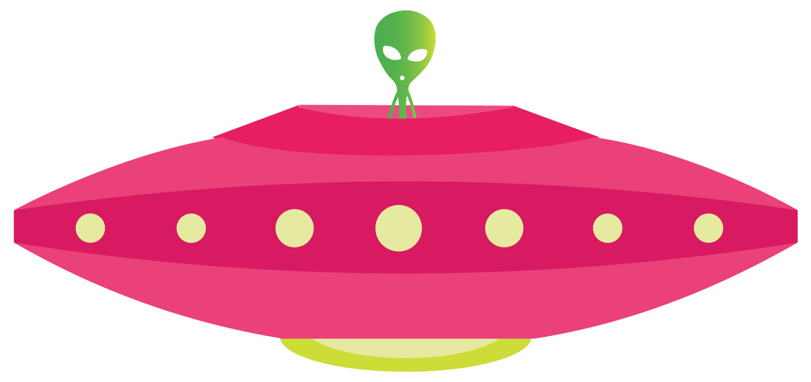


GRAPHQL WAS DESIGNED FOR GRAPH DATABASES.

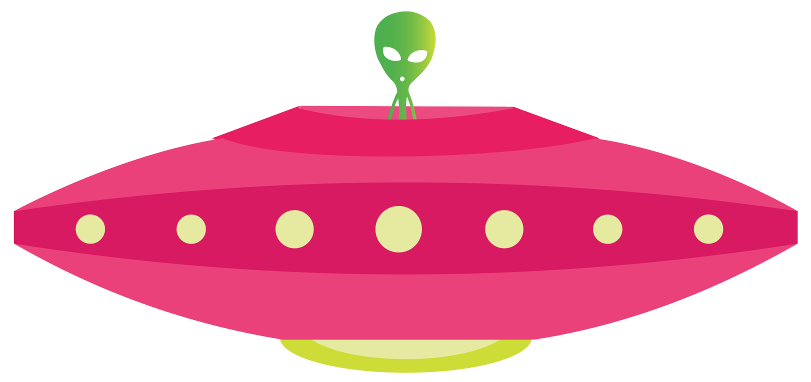


FALSE

GRAPHQL WAS DESIGNED FOR GRAPH DATABASES.

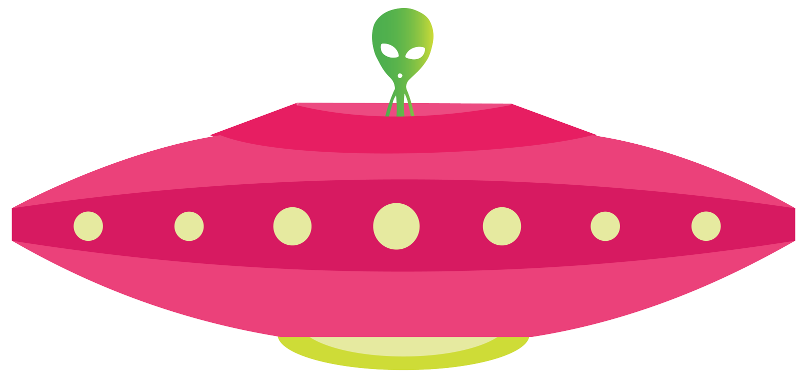


GRAPHQL IS LANGUAGE AGNOSTIC.

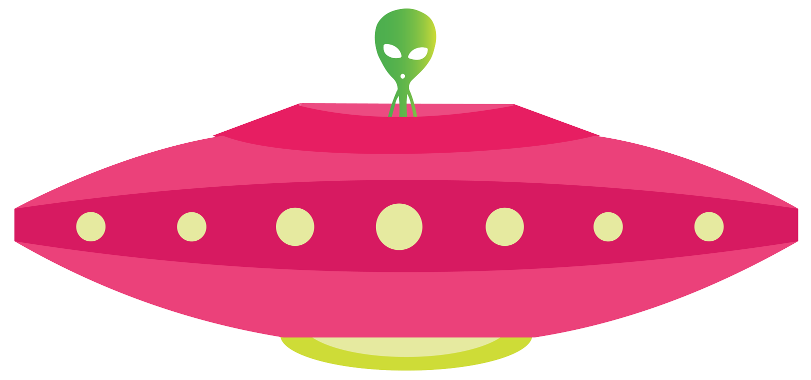


TRUE

GRAPHQL IS LANGUAGE AGNOSTIC.

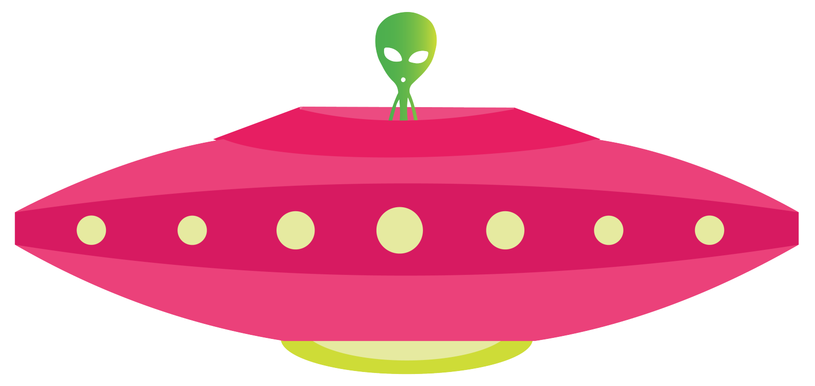


APOLLO ANDROID IS THE ONLY GRAPHQL CLIENT.



FALSE

APOLLO ANDROID IS THE ONLY GRAPHQL CLIENT.



REST IS DEAD.

YOU DECIDE

GETTING A GRIP ON GRAPHQL

@BRWNGRLDEV