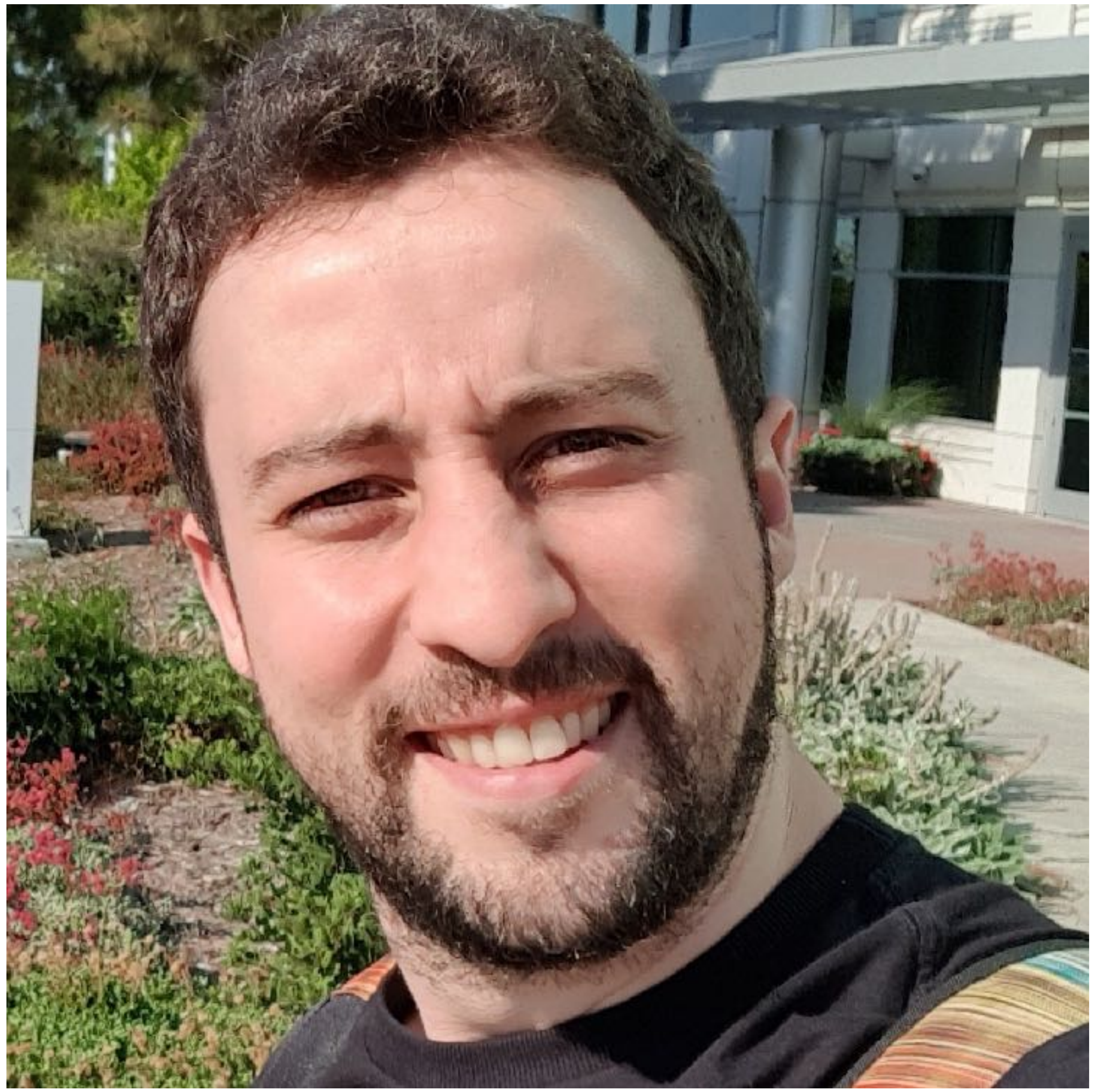
Jarvis

O gateway em graphql para as APIs do Globoplay

Marcelo Nalon

marcelo.nalon@engenharia.ufjf.br

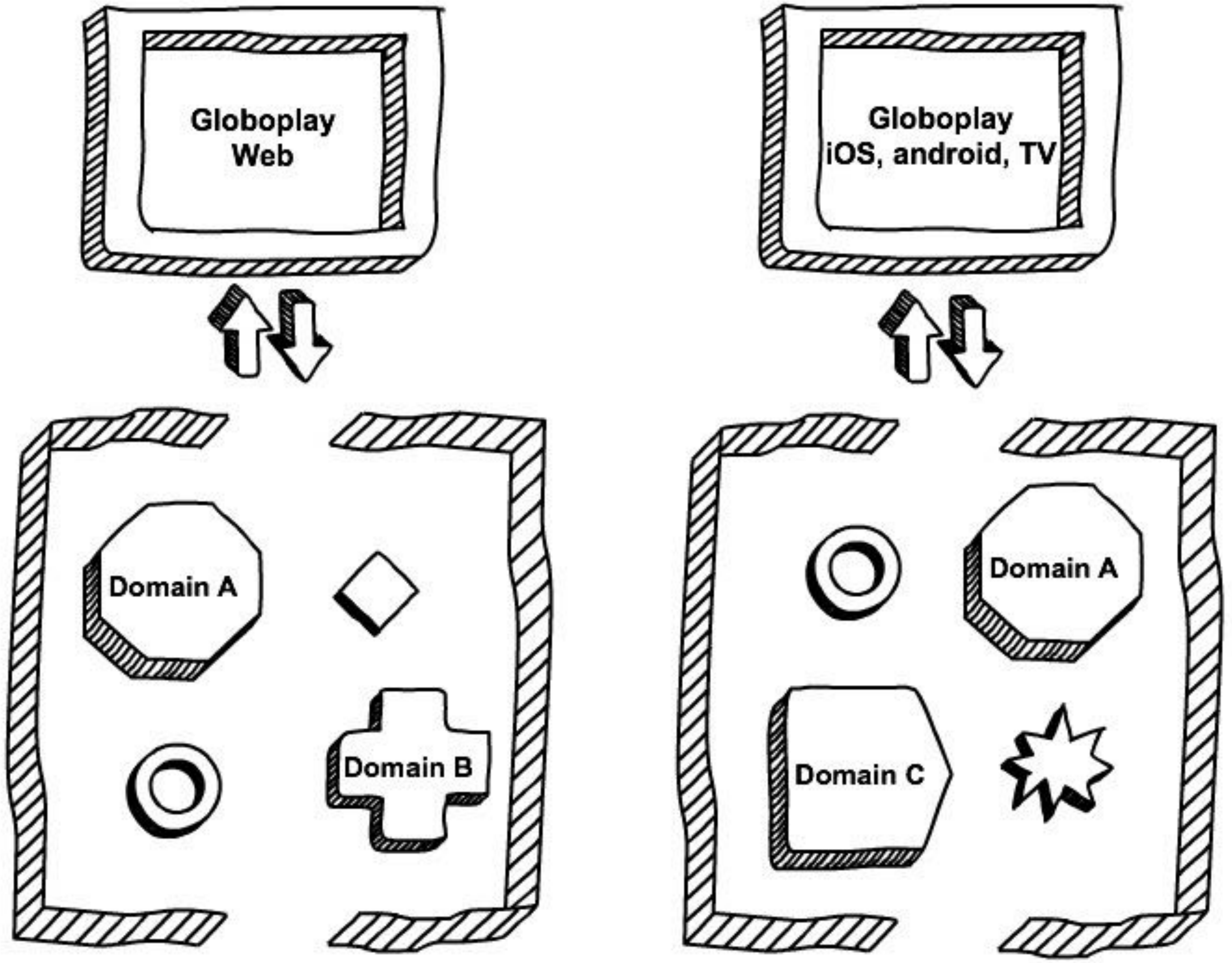


AGENDA

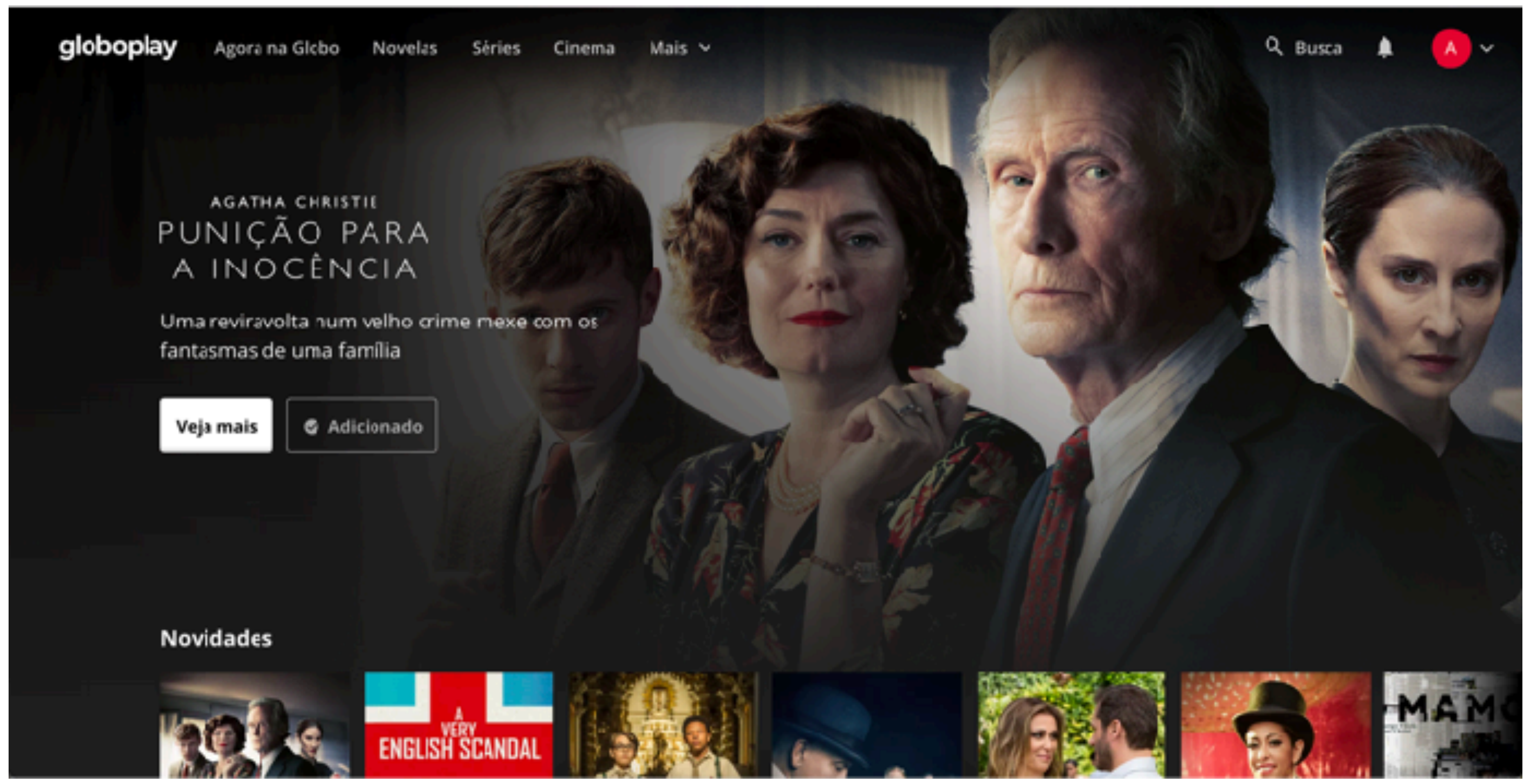
- Breve contexto e motivações para evoluir a arquitetura do BE
- Por que GRAPHQL?
- Principais features do Apollo Server

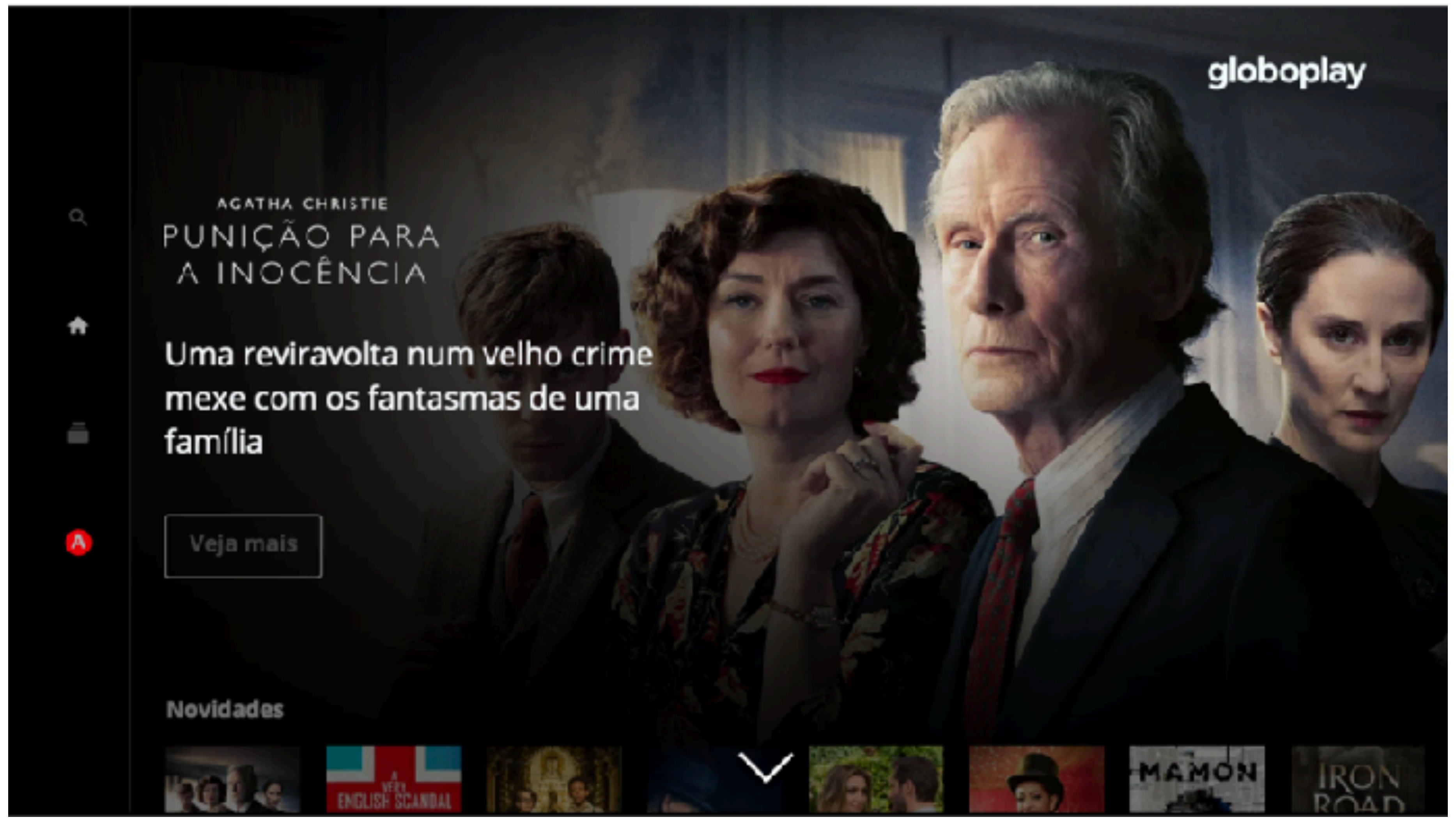
globoplay
the Voice Brasil
Meu Passado me Condena 2
O Filme
Segundo Sol
Tropa de Elite 2
O inimigo agora é outro
JN
The Good Doctor
No Ar
O Palhaço

Alguns meses atrás...

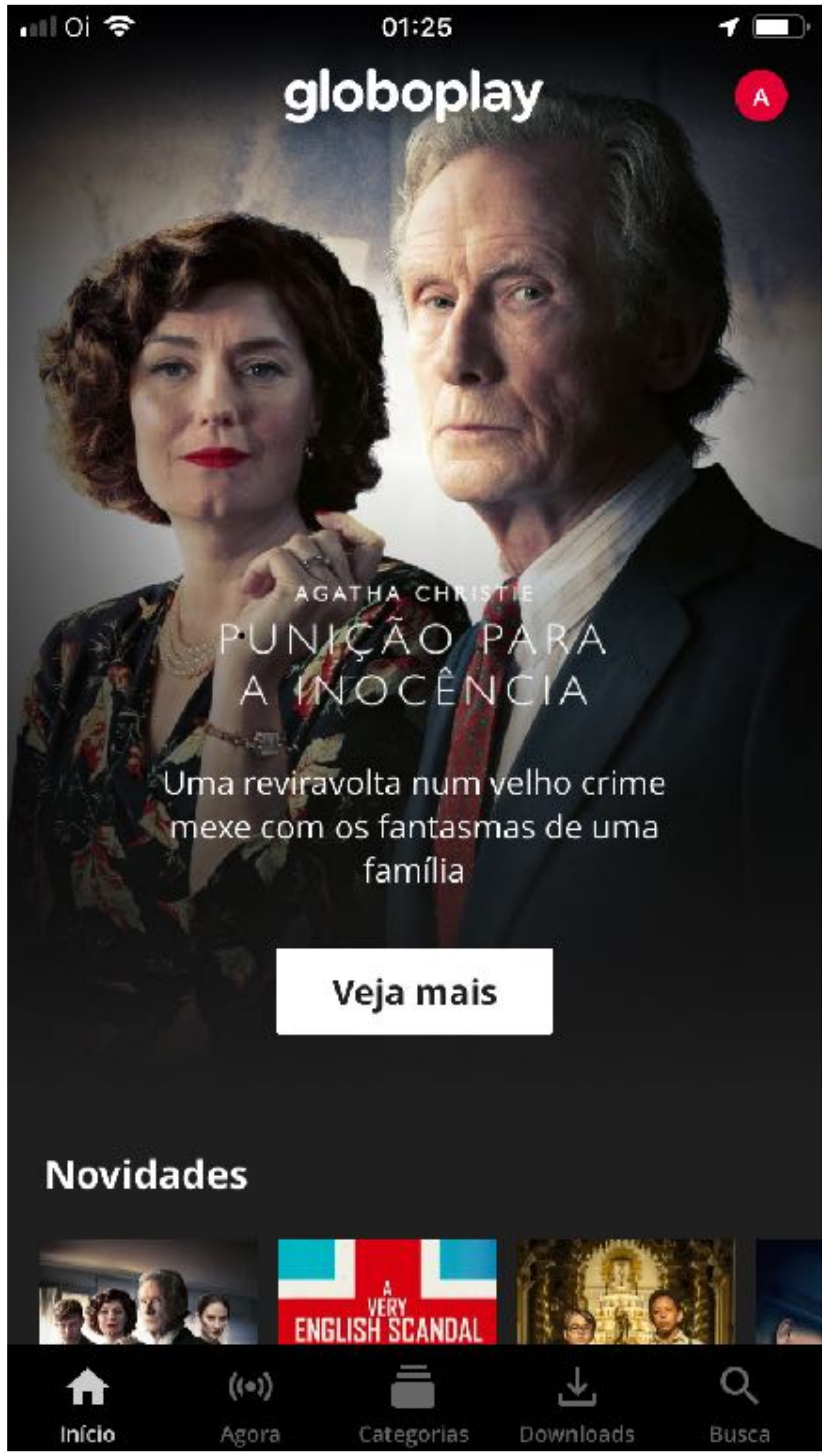


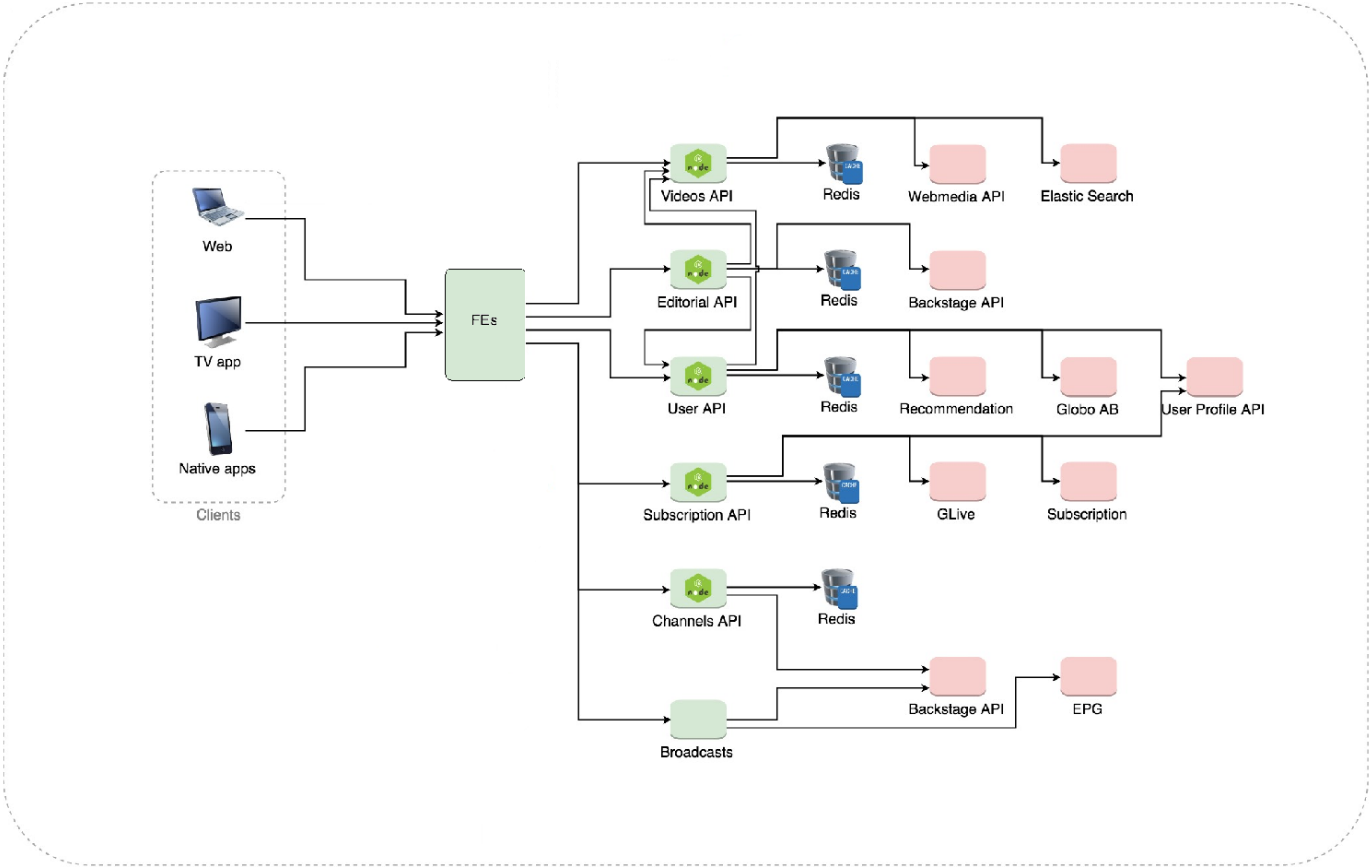
Ubiquidade!




globoplay
AGATHA CHRISTIE
PUNIÇÃO PARA A INOCÊNCIA
Uma reviravolta num velho crime mexe com os fantasmas de uma família
Veja mais
Novidades

SMART TV





🟩 Produto 🟥 Other teams

GraphQL

https://graphql.org/

ATLASSIAN

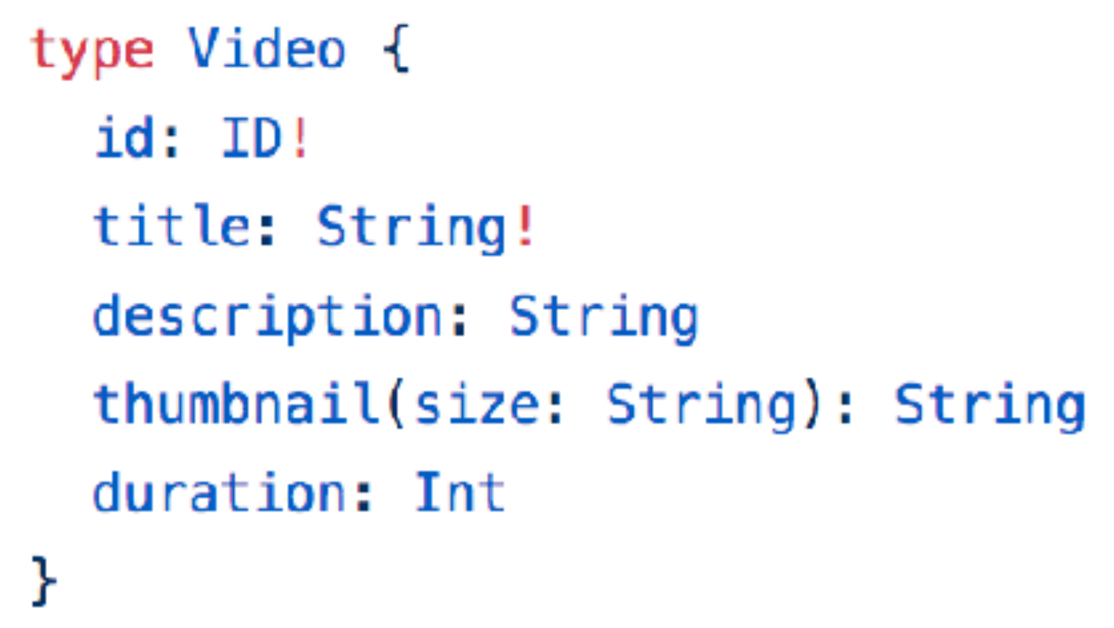
sky

GetNinjas

PayPal

coursera

- Descrição clara dos dados https://graphql.github.io/learn/schema/



```
type Video {
  id: ID!
  title: String!
  description: String
  thumbnail(size: String): String
  duration: Int
}
```

- Descrição clara dos dados https://graphql.github.io/learn/schema/

```
type Video {
  id: ID!
  title: String!
  description: String
  thumbnail(size: String): String
  duration: Int
}
```

- Fácil evolução da API (sem necessidade de versionamento)

```
type Video {
  id: ID!
  title: String!
  description: String
  thumbnail(size: String): String
  duration: Int
  program: String
}
```

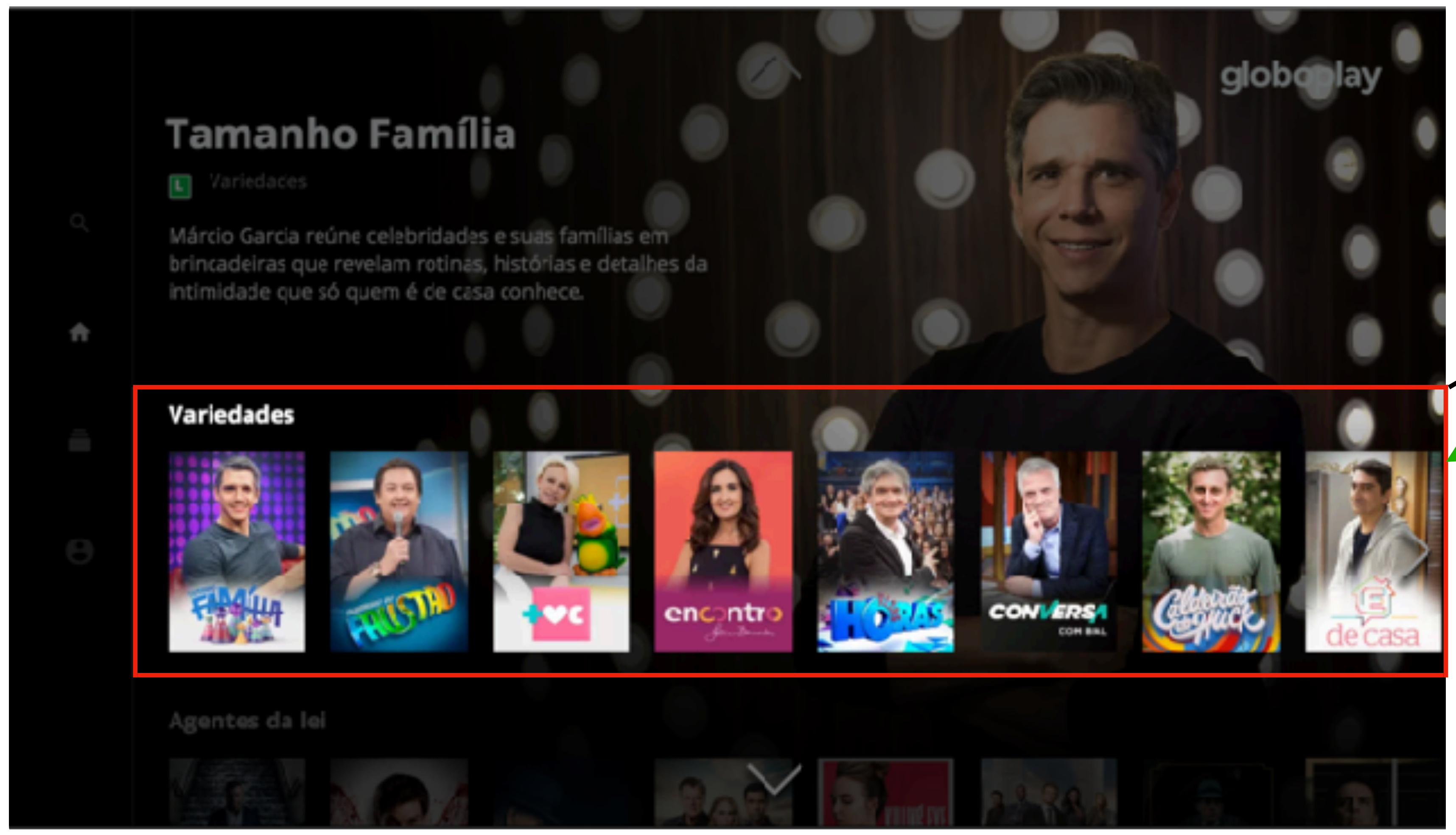
```
type Program {
  id: ID!
  title: String
}

type Video {
  id: ID!
  title: String!
  description: String
  thumbnail(size: String): String
  duration: Int
  program: String @deprecated
  relatedProgram: Program
}
```

- Ask for what you want

GraphQL

```
query getOffer{
  offer(id:"xyz"){
    headline
    items{
        poster{
           tv
        }
    }
  }
}
```

```
{
  "data": {
    "offer": {
      "headline": "Variedades",
      "items": [
        {
          "poster": {
            "tv": "http://foo.bar/tamanhofamilia.jpg"
          }
        },
        ...
      ]
    }
  }
}
```

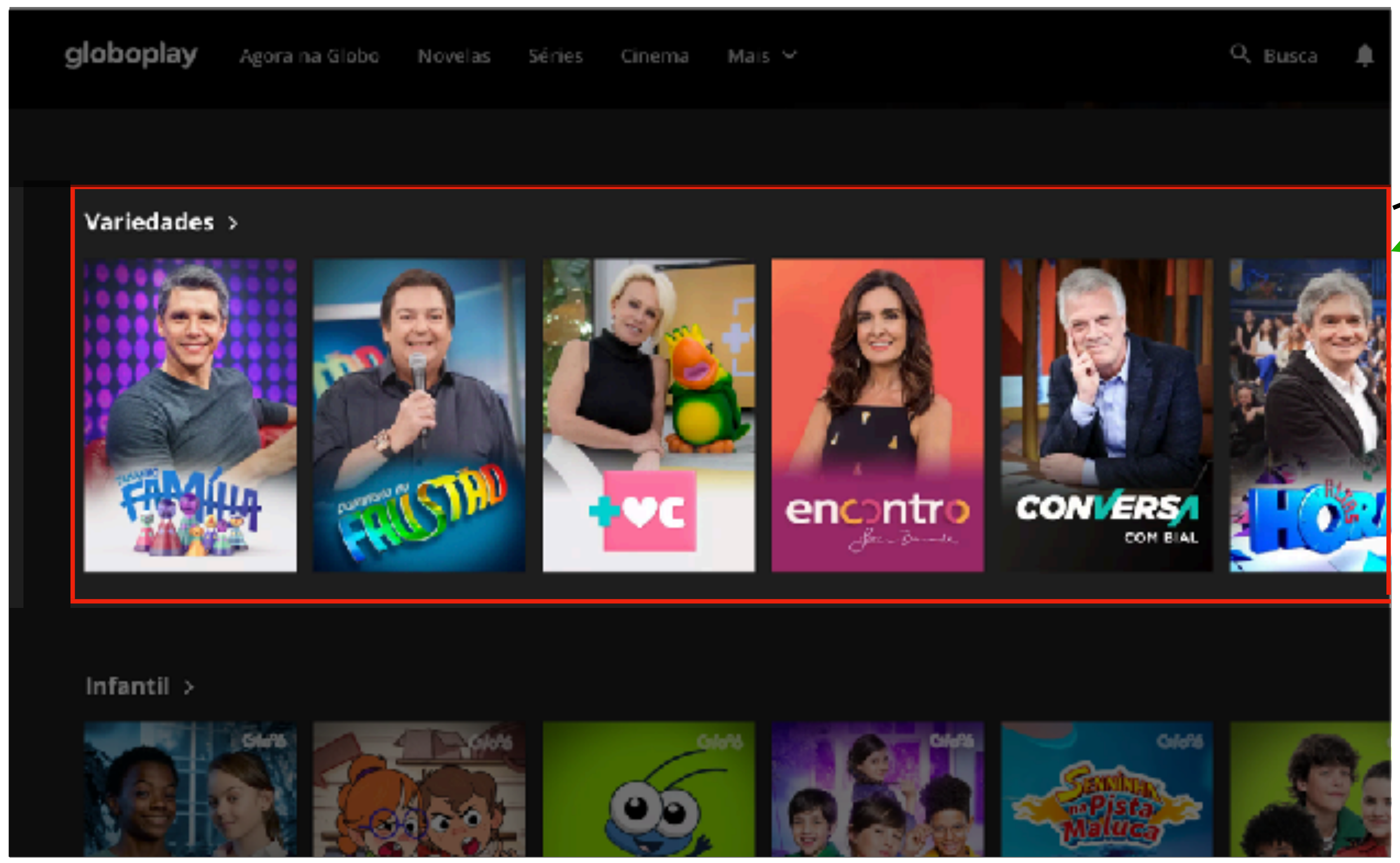


Márcio Garcia reúne celebridades e suas famílias em brincadeiras que revelam rotinas, histórias e detalhes da intimidade que só quem é de casa conhece.

- Ask for what you want

```
query getOffer{
    offer(id:"xyz"){
        headline
        url
        items{
            poster{
                web
            }
        }
    }
}
```

GraphQL



```
{
  "data": {
    "offer": {
      "headline": "Variedades",
      "url": "http://globoplay.com.br/variedades",
      "items": [
        {
          "id": "9529",
          "poster": {
            "web": "http://foo.bar/tamanhofamilia.jpg"
          }
        },
        ...
      ]
    }
  }
}
```

- Ask for what you want

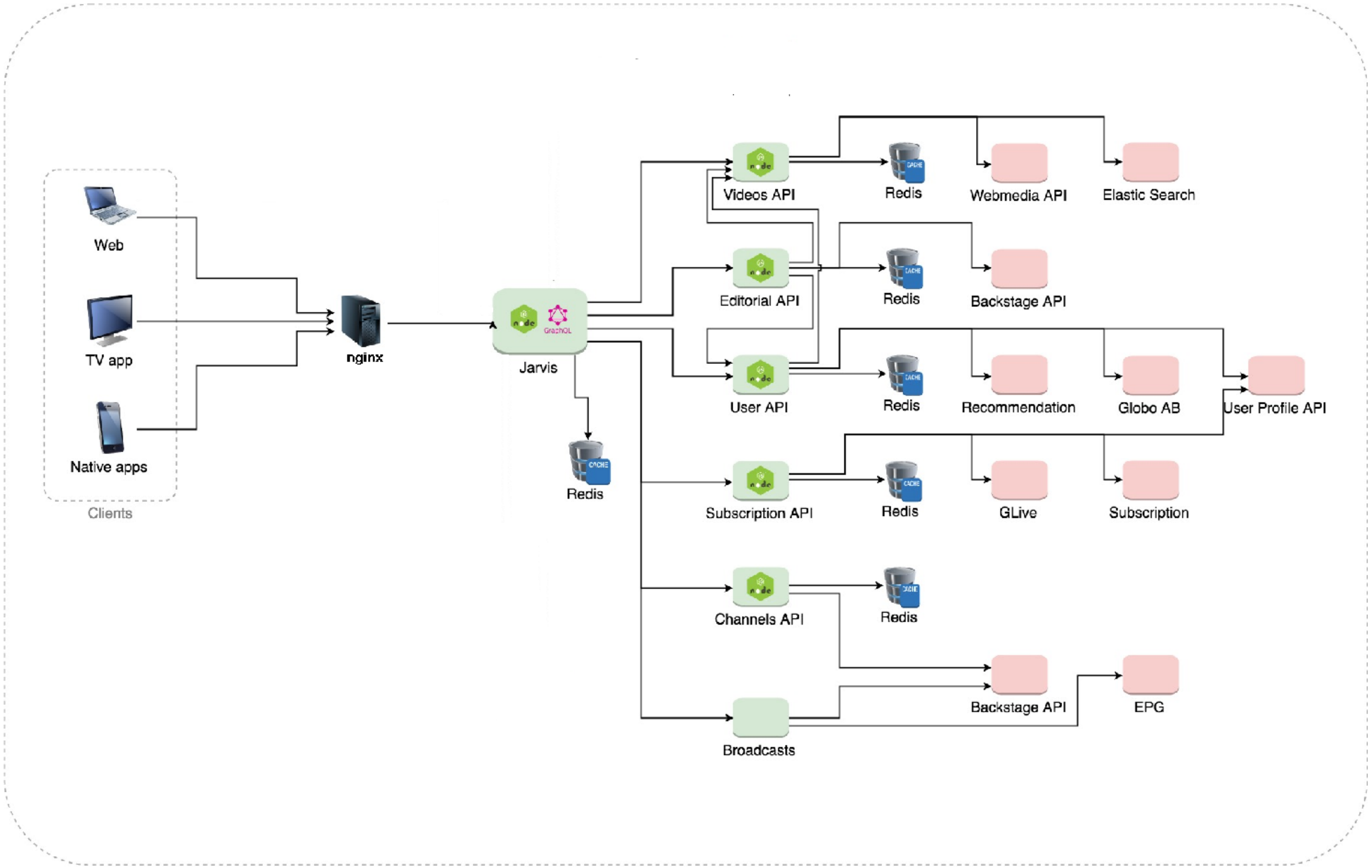
✓ Redução do payload (remoção de dados desnecessários na resposta)

✓ Resposta altamente previsível

✓ Possibilidade de obter vários recursos em uma única requisição

```
query getOfferAndHighlight{
  offer(id:"xyz"){
     items{
       poster{
         tv
       }
     }
  }
  highlight(id:"abc"){
     headline
     cover{
        tv
     }
  }
}
```

alright, what's our next move?

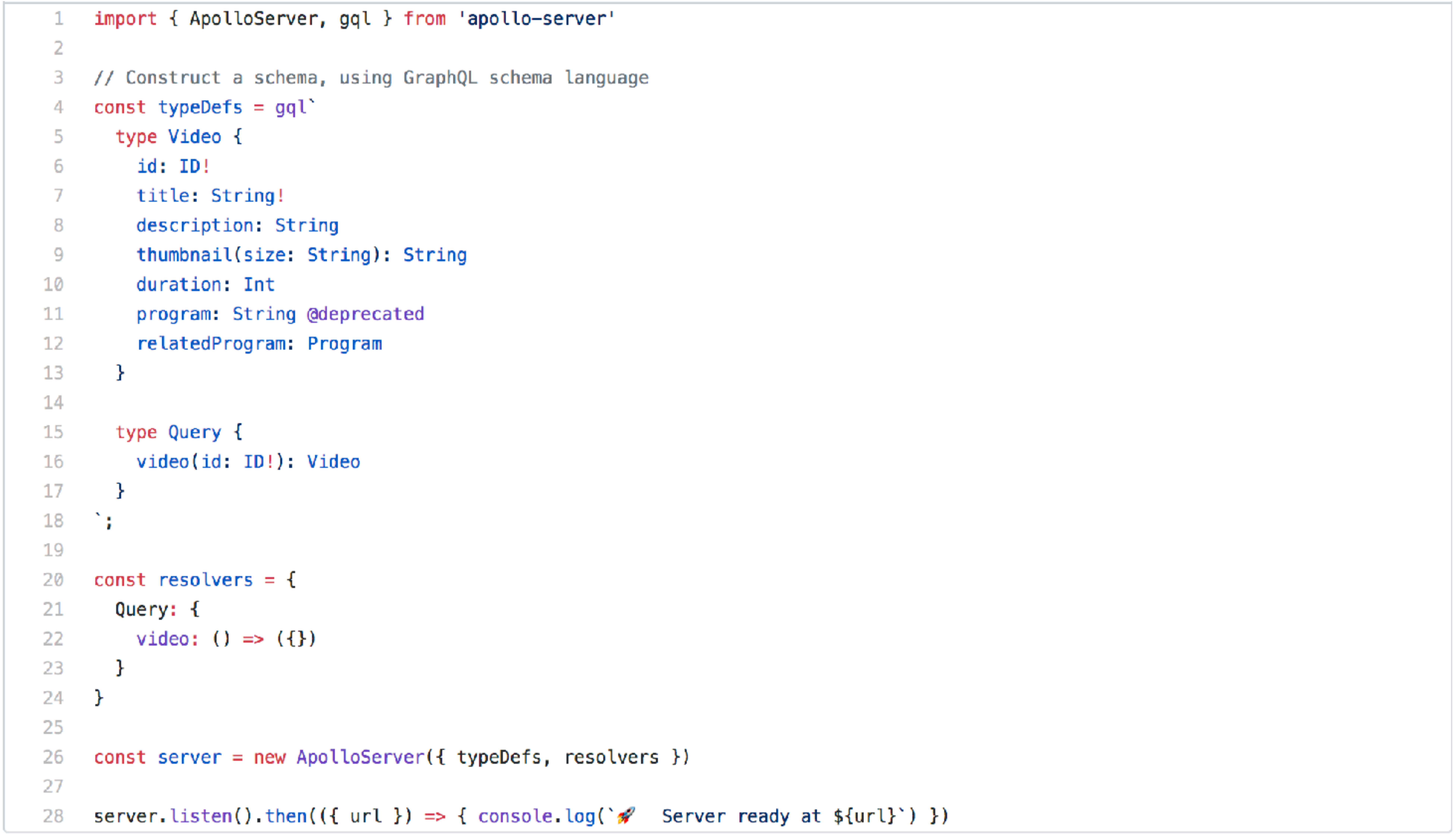


🟩 Produto 🟥 Other teams

APOLLO
PLATFORM
APOLLO SERVER
SCHEMA REGISTRY
OPERATION REGISTRY
EXECUTION GATEWAY
IDE PLUGINS
APOLLO CLIENT
TRACE WAREHOUSE

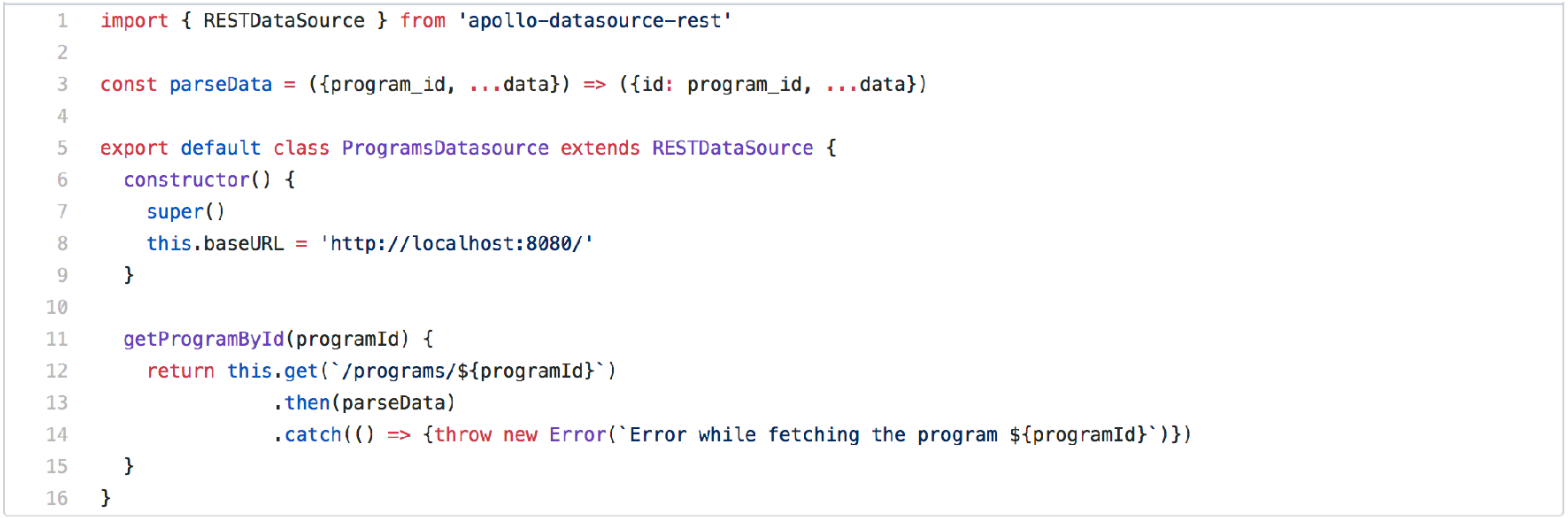
EXECUTION GATEWAY
APOLLO SERVER
OPERATION REGISTRY
APOLLO PLATFORM
IDE PLUGINS
APOLLO CLIENT
SCHEMA REGISTRY
TRACE WAREHOUSE

apollo-server



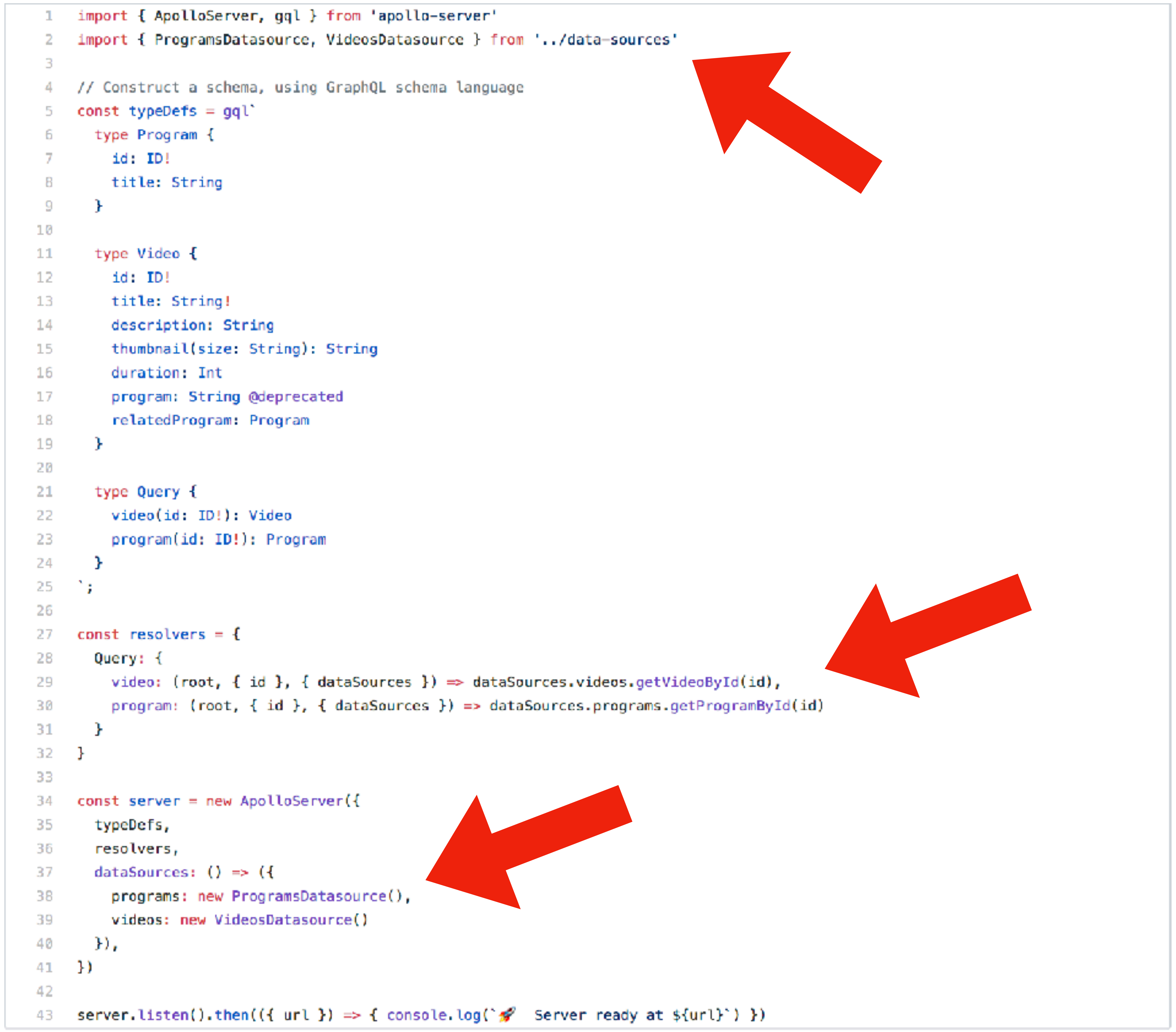
```
import { ApolloServer, gql } from 'apollo-server'

// Construct a schema, using GraphQL schema language
const typeDefs = gql`
  type Video {
    id: ID!
    title: String!
    description: String
    thumbnail(size: String): String
    duration: Int
    program: String @deprecated
    relatedProgram: Program
  }

  type Query {
    video(id: ID!): Video
  }
`;

const resolvers = {
  Query: {
    video: () => ({})
  }
}

const server = new ApolloServer({ typeDefs, resolvers })

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

apollo-datasource-rest



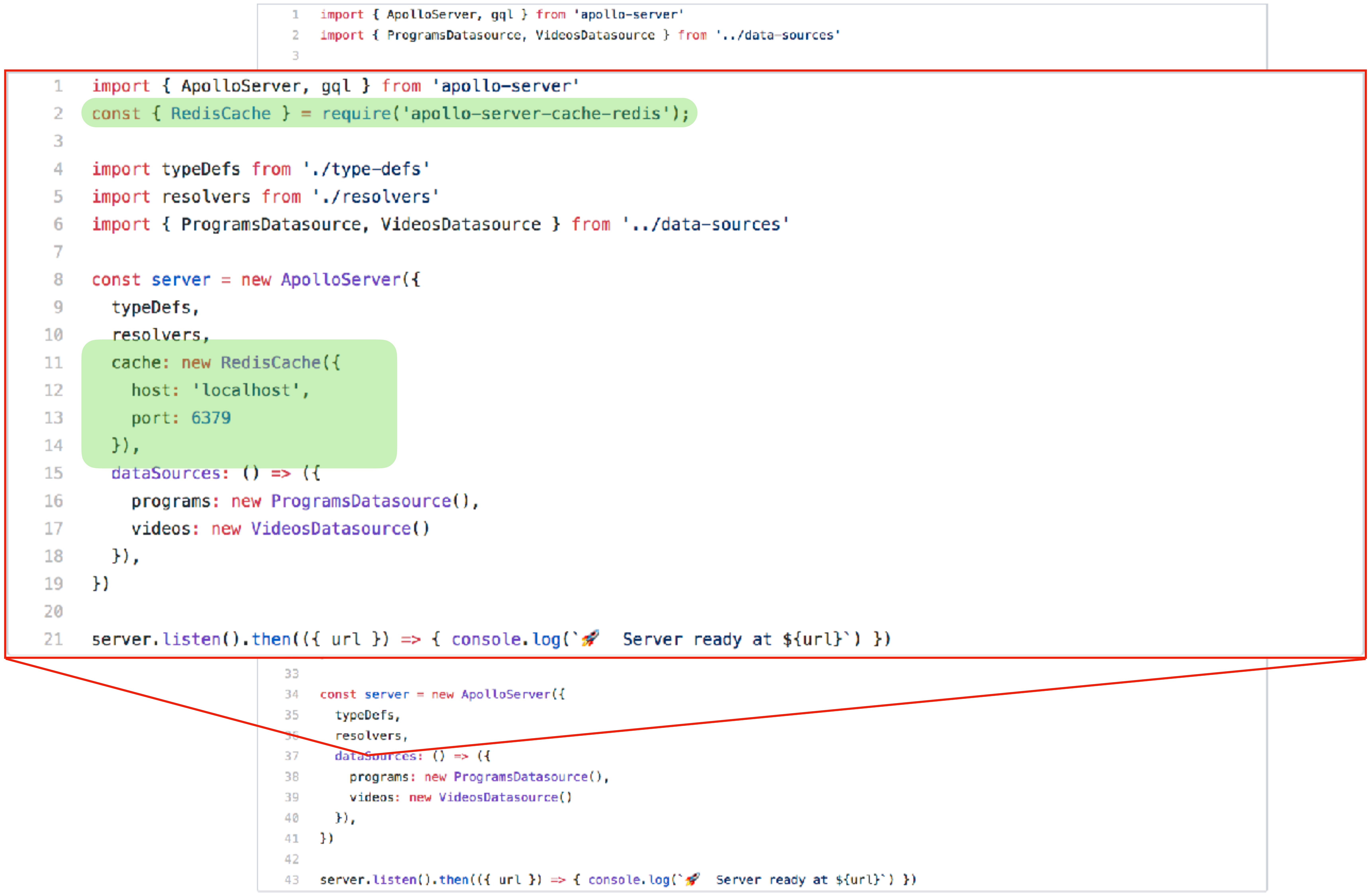
```
import { RESTDataSource } from 'apollo-datasource-rest'

const parseData = ({program_id, ...data}) => ({id: program_id, ...data})

export default class ProgramsDatasource extends RESTDataSource {
  constructor() {
    super()
    this.baseURL = 'http://localhost:8080/'
  }

  getProgramById(programId) {
    return this.get(`/programs/${programId}`)
             .then(parseData)
             .catch(() => {throw new Error(`Error while fetching the program ${programId}`)})
  }
}
```

apollo-datasource-rest



```
import { ApolloServer, gql } from 'apollo-server'
import { ProgramsDatasource, VideosDatasource } from '../data-sources'

// Construct a schema, using GraphQL schema language
const typeDefs = gql`
  type Program {
    id: ID!
    title: String
  }

  type Video {
    id: ID!
    title: String!
    description: String
    thumbnail(size: String): String
    duration: Int
    program: String @deprecated
    relatedProgram: Program
  }

  type Query {
    video(id: ID!): Video
    program(id: ID!): Program
  }
`;

const resolvers = {
  Query: {
    video: (root, { id }, { dataSources }) => dataSources.videos.getVideoById(id),
    program: (root, { id }, { dataSources }) => dataSources.programs.getProgramById(id)
  }
}

const server = new ApolloServer({
  typeDefs,
  resolvers,
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

apollo-datasource-rest

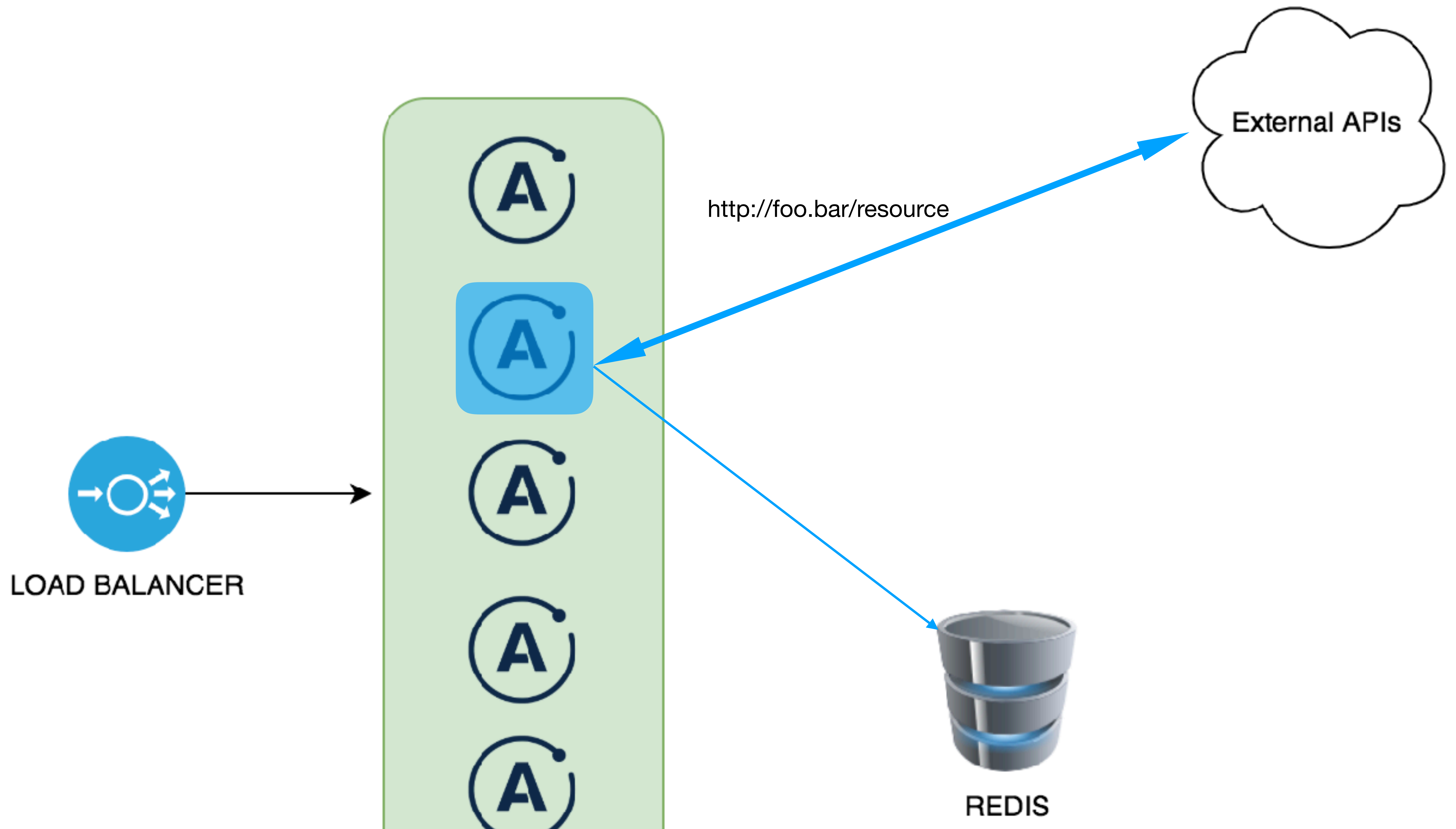


```
import { ApolloServer, gql } from 'apollo-server'
import { ProgramsDatasource, VideosDatasource } from '../data-sources'

```

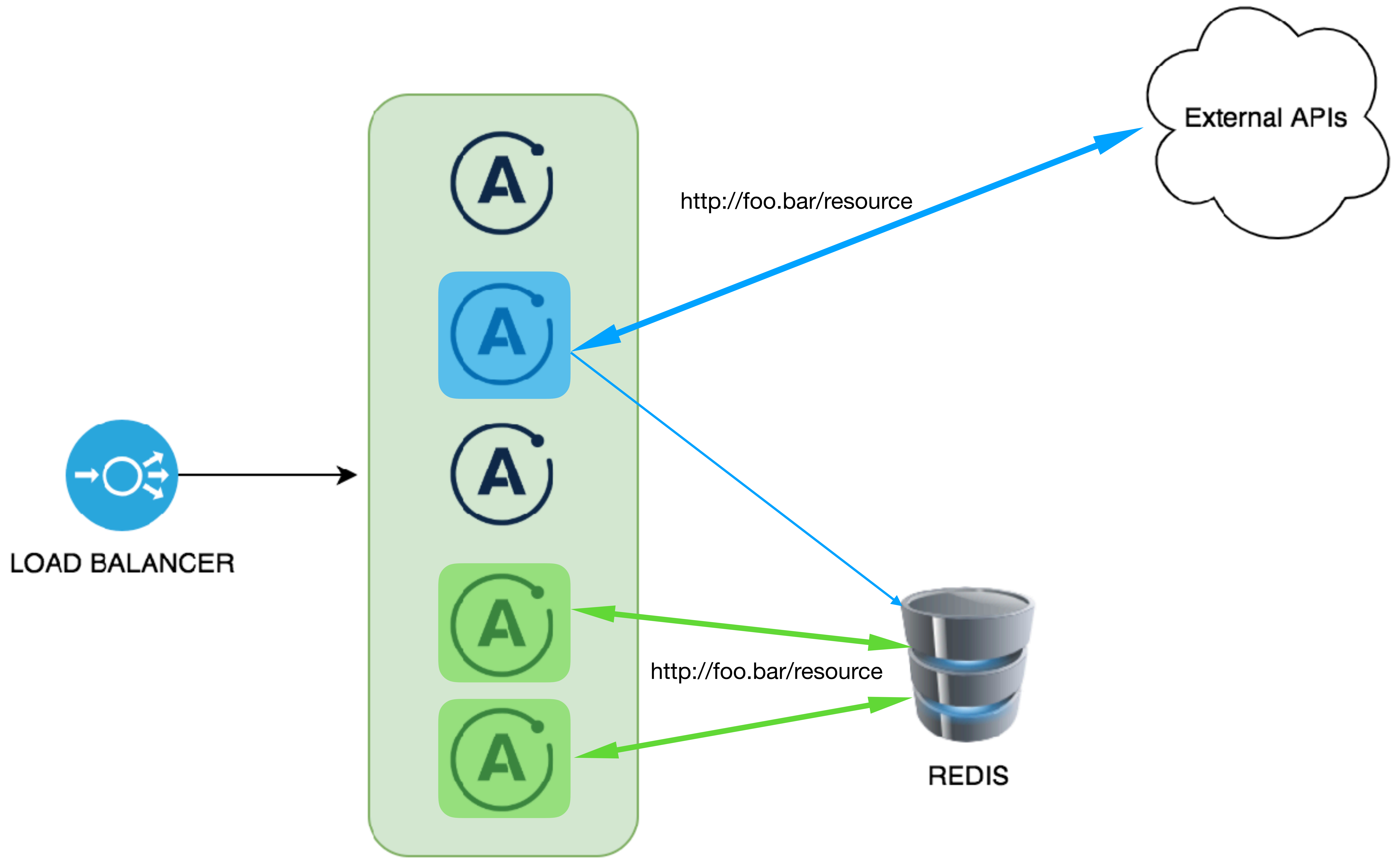
```
import { ApolloServer, gql } from 'apollo-server'
const { RedisCache } = require('apollo-server-cache-redis');

import typeDefs from './type-defs'
import resolvers from './resolvers'
import { ProgramsDatasource, VideosDatasource } from '../data-sources'

const server = new ApolloServer({
  typeDefs,
  resolvers,
  cache: new RedisCache({
    host: 'localhost',
    port: 6379
  }),
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

```

const server = new ApolloServer({
  typeDefs,
  resolvers,
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

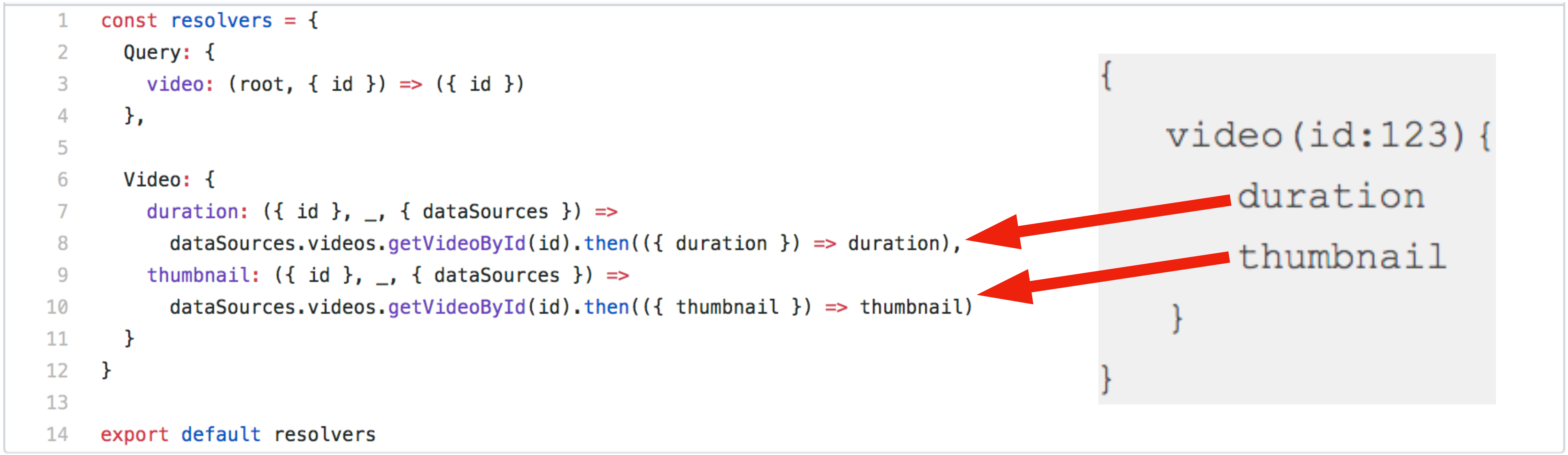
apollo-datasource-rest (cache)



apollo-datasource-rest (cache)



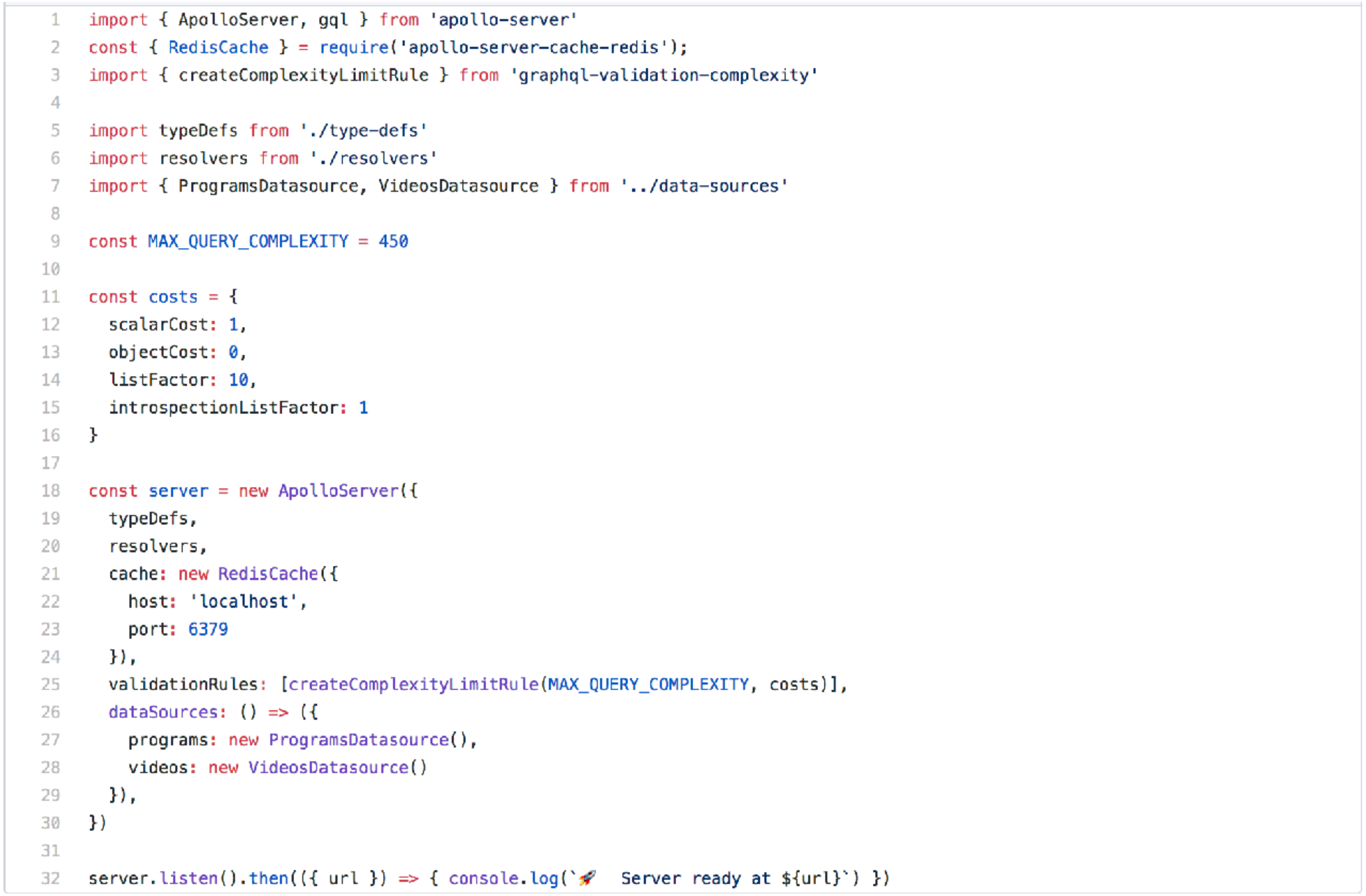
apollo-datasource-rest (memoization)



```
const resolvers = {
  Query: {
    video: (root, { id }) => ({ id })
  },

  Video: {
    duration: ({ id }, _, { dataSources }) =>
      dataSources.videos.getVideoById(id).then(({ duration }) => duration),
    thumbnail: ({ id }, _, { dataSources }) =>
      dataSources.videos.getVideoById(id).then(({ thumbnail }) => thumbnail)
  }
}

export default resolvers
```

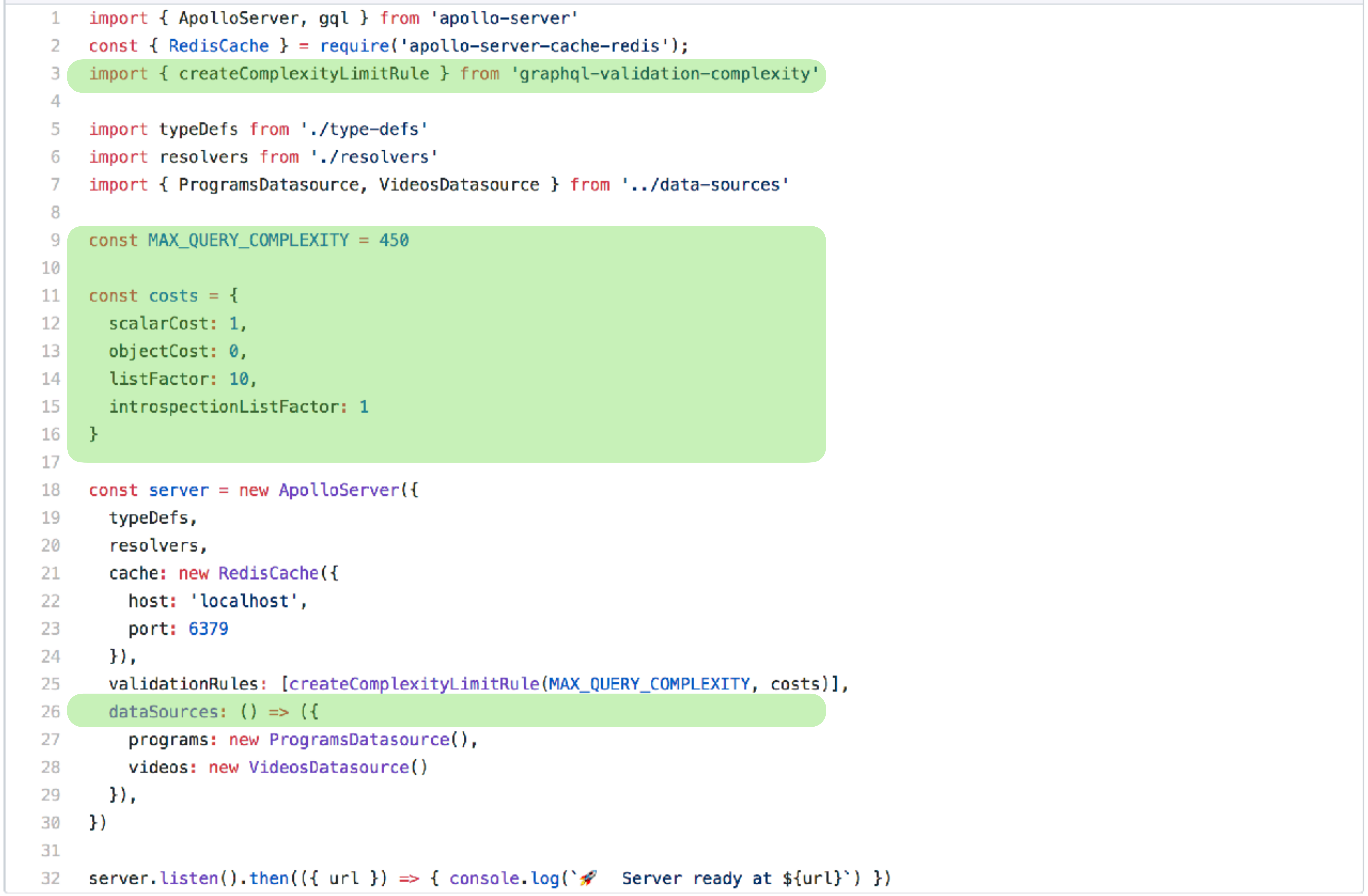
```
{
  video(id:123){
    duration
    thumbnail
  }
}
```

análise de custo da query de consulta



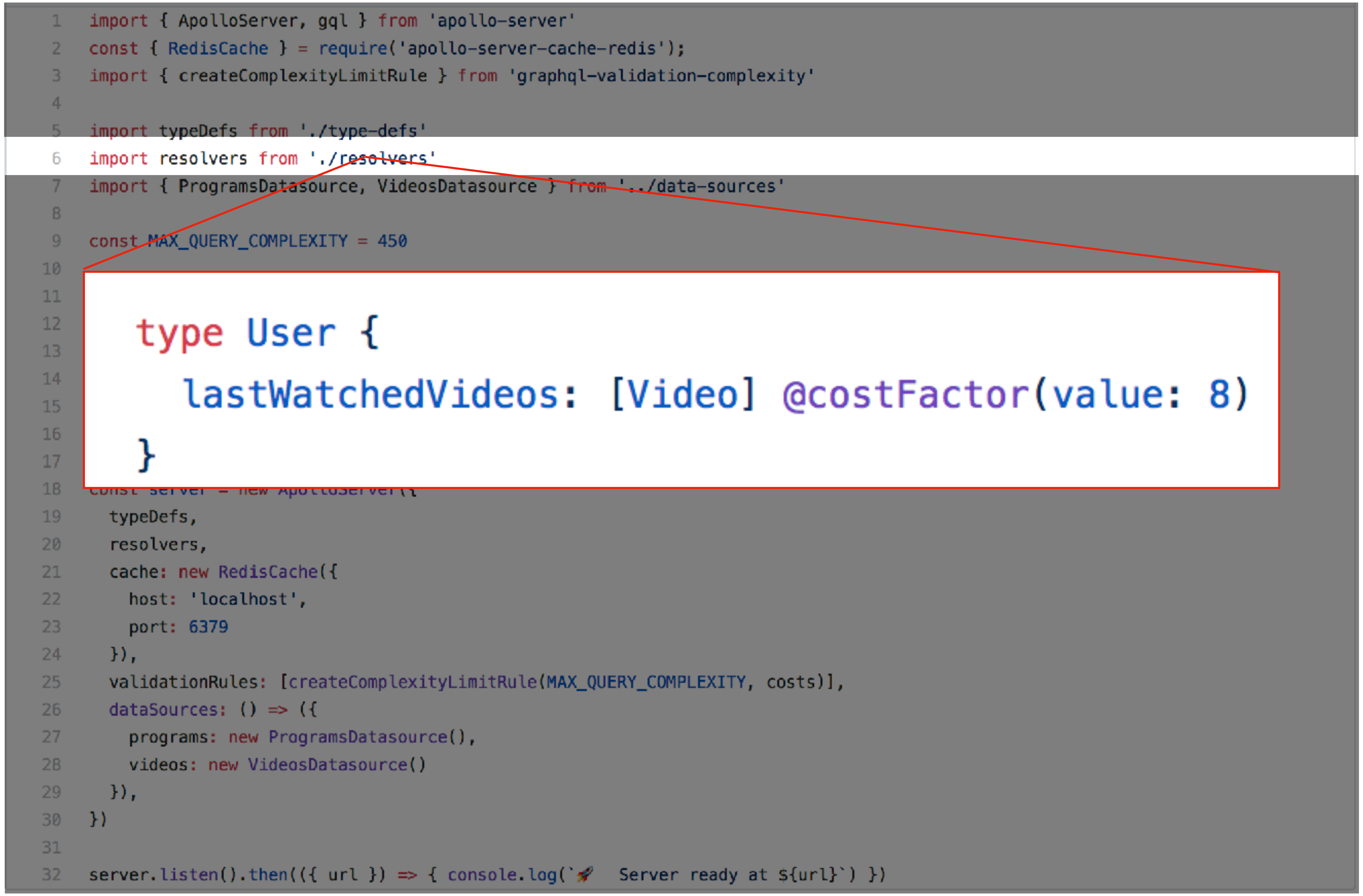
```javascript
import { ApolloServer, gql } from 'apollo-server'
const { RedisCache } = require('apollo-server-cache-redis');
import { createComplexityLimitRule } from 'graphql-validation-complexity'

import typeDefs from './type-defs'
import resolvers from './resolvers'
import { ProgramsDatasource, VideosDatasource } from '../data-sources'

const MAX_QUERY_COMPLEXITY = 450

const costs = {
  scalarCost: 1,
  objectCost: 0,
  listFactor: 10,
  introspectionListFactor: 1
}

const server = new ApolloServer({
  typeDefs,
  resolvers,
  cache: new RedisCache({
    host: 'localhost',
    port: 6379
  }),
  validationRules: [createComplexityLimitRule(MAX_QUERY_COMPLEXITY, costs)],
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

análise de custo da query de consulta



```
import { ApolloServer, gql } from 'apollo-server'
const { RedisCache } = require('apollo-server-cache-redis');
import { createComplexityLimitRule } from 'graphql-validation-complexity'

import typeDefs from './type-defs'
import resolvers from './resolvers'
import { ProgramsDatasource, VideosDatasource } from '../data-sources'

const MAX_QUERY_COMPLEXITY = 450

const costs = {
  scalarCost: 1,
  objectCost: 0,
  listFactor: 10,
  introspectionListFactor: 1
}

const server = new ApolloServer({
  typeDefs,
  resolvers,
  cache: new RedisCache({
    host: 'localhost',
    port: 6379
  }),
  validationRules: [createComplexityLimitRule(MAX_QUERY_COMPLEXITY, costs)],
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

análise de custo da query de consulta

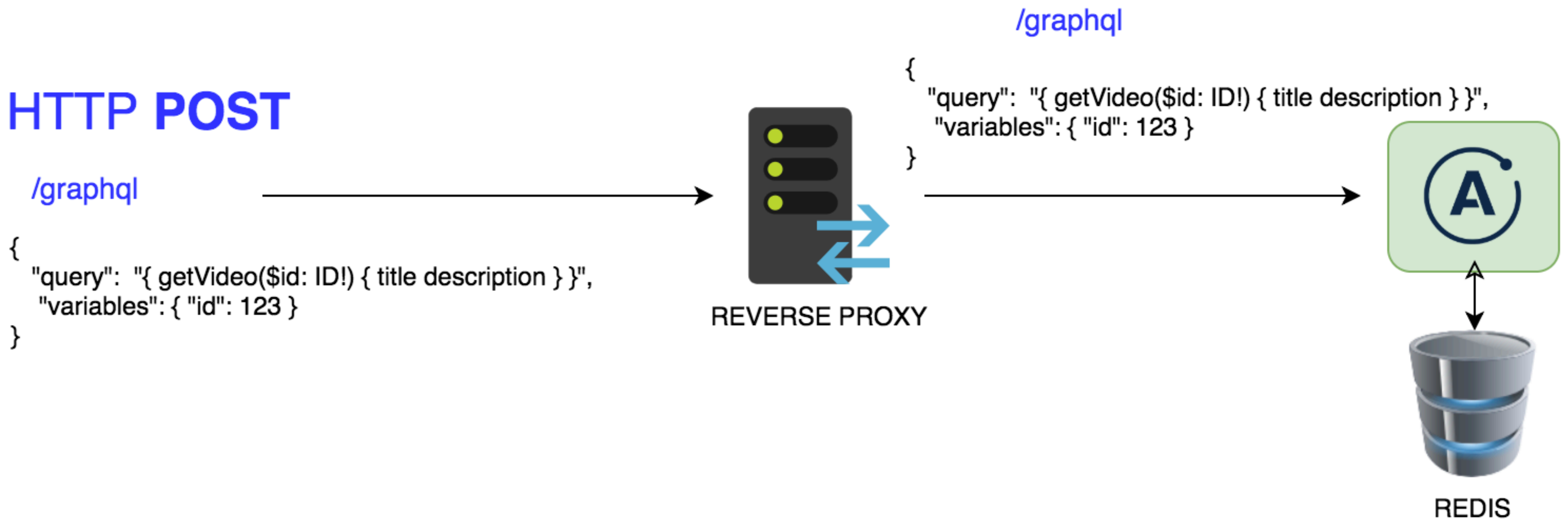


```
import { ApolloServer, gql } from 'apollo-server'
const { RedisCache } = require('apollo-server-cache-redis');
import { createComplexityLimitRule } from 'graphql-validation-complexity'

import typeDefs from './type-defs'
import resolvers from './resolvers'
import { ProgramsDatasource, VideosDatasource } from '../data-sources'

const MAX_QUERY_COMPLEXITY = 450
```

```
type User {
  lastWatchedVideos: [Video] @costFactor(value: 8)
}
```

```
const server = new ApolloServer({
  typeDefs,
  resolvers,
  cache: new RedisCache({
    host: 'localhost',
    port: 6379
  }),
  validationRules: [createComplexityLimitRule(MAX_QUERY_COMPLEXITY, costs)],
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})

server.listen().then(({ url }) => { console.log(`🚀  Server ready at ${url}`) })
```

cache http



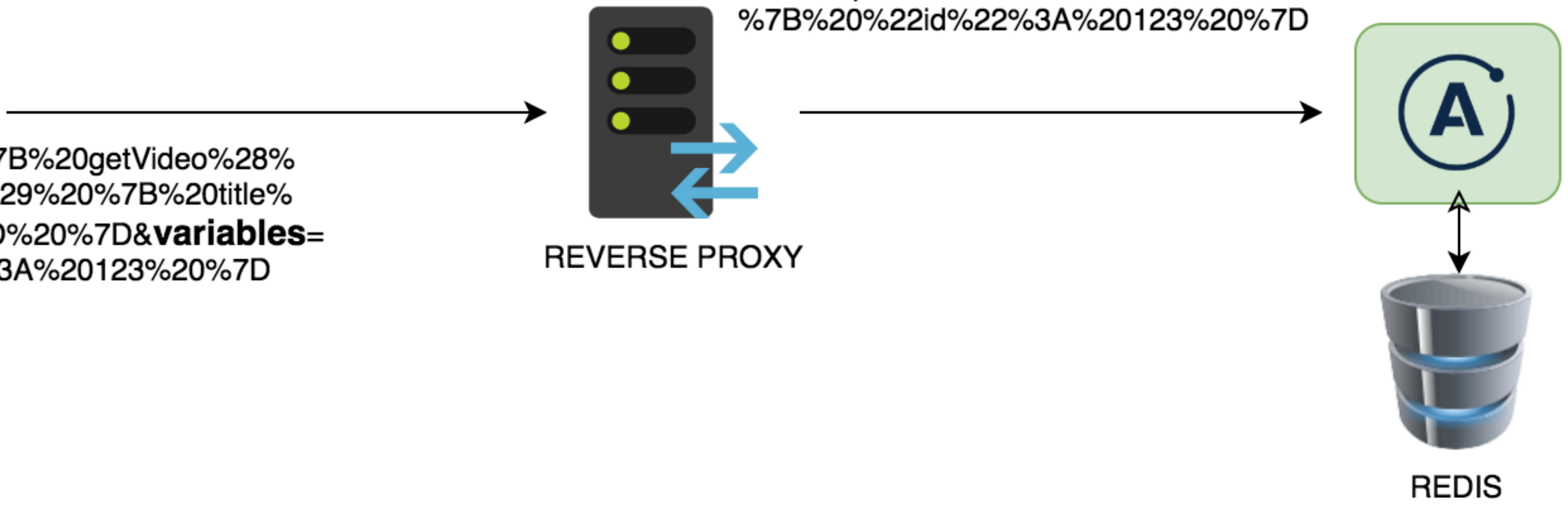
```
{
  "query":  "{ getVideo($id: ID!) { title description } }",
   "variables": { "id": 123 }
}
```

```
{
  "query":  "{ getVideo($id: ID!) { title description } }",
   "variables": { "id": 123 }
}
```

cache http

HTTP **GET**

/graphql?**query**=%7B%20getVideo%28%
24id%3A%20ID%21%29%20%7B%20title%
20description%20%7D%20%7D&**variables**=
%7B%20%22id%22%3A%20123%20%7D



/graphql?**query**=%7B%20getVideo%28%
24id%3A%20ID%21%29%20%7B%20title%
20description%20%7D%20%7D&**variables**=
%7B%20%22id%22%3A%20123%20%7D

cache http (cache-control header)

1. **default max-age**

cache http (cache-control header)

1. default max-age

```
const server = new ApolloServer({
  typeDefs,
  resolvers,
  cache: new RedisCache({
    host: 'localhost',
    port: 6379
  }),
  cacheControl: {
    defaultMaxAge: 300
  },
  validationRules: [createComplexityLimitRule(MAX_QUERY_COMPLEXITY, costs)],
  dataSources: () => ({
    programs: new ProgramsDatasource(),
    videos: new VideosDatasource()
  }),
})
```

cache http (cache-control header)

1. **default max-age**
2. **cache hints**

cache http (cache-control header)

1. **default max-age**

2. **cache hints**

```
type User {
  lastWatchedVideos: [Video] @costFactor(value: 8) @cacheControl(maxAge: 5, scope: PRIVATE)
}
```

cache http (cache-control header)

1. default max-age
2. cache hints
3. cache hint dinâmico

cache http (cache-control header)

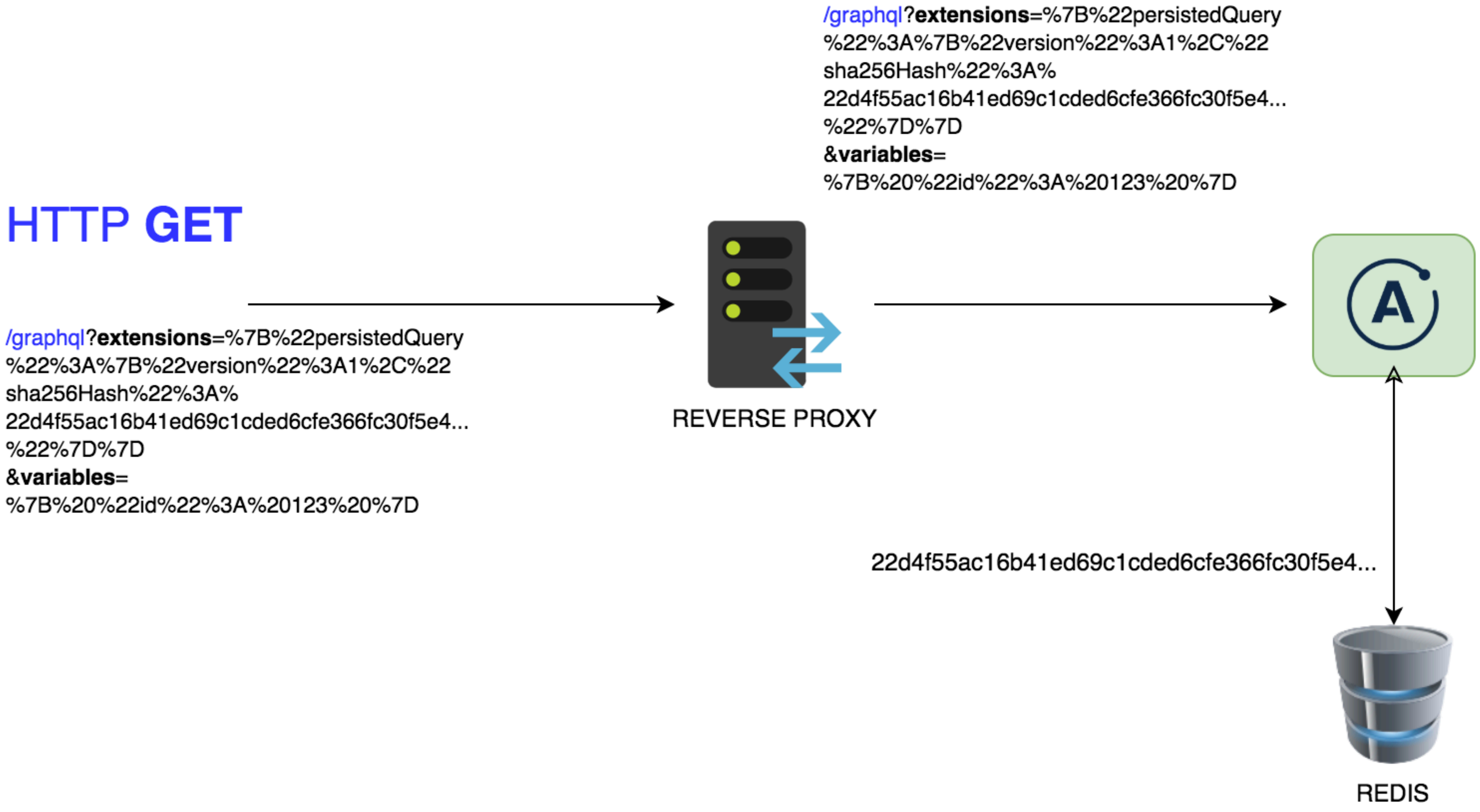
1. default max-age

2. cache hints

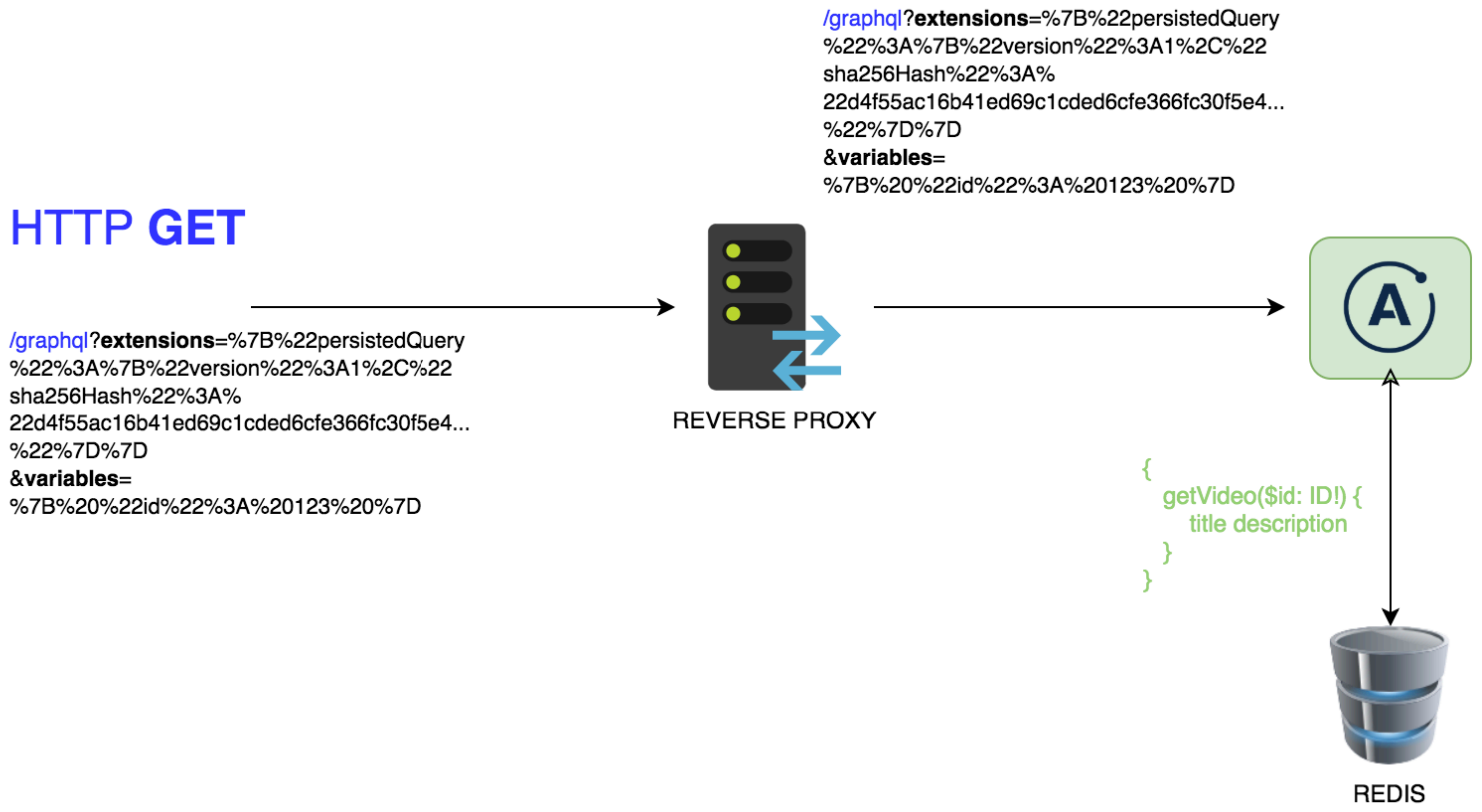
3. cache hint dinâmico

```
User: {
  lastWatchedVideos: (videos, variables, context, { cacheControl }) => {
    if('something happens...') cacheControl.setCacheHint({ maxAge: 300 })
    return videos
  }
}
```

persisted query


HTTP GET
/graphql?extensions=%7B%22persistedQuery%22%3A%7B%22version%22%3A1%2C%22sha256Hash%22%3A%22d4f55ac16b41ed69c1cded6cfe366fc30f5e4...%22%7D%7D&variables=%7B%20%22id%22%3A%20123%20%7D
REVERSE PROXY
/graphql?extensions=%7B%22persistedQuery%22%3A%7B%22version%22%3A1%2C%22sha256Hash%22%3A%22d4f55ac16b41ed69c1cded6cfe366fc30f5e4...%22%7D%7D&variables=%7B%20%22id%22%3A%20123%20%7D
22d4f55ac16b41ed69c1cded6cfe366fc30f5e4...
REDIS

persisted query



HTTP **GET**

```
/graphql?extensions=%7B%22persistedQuery
%22%3A%7B%22version%22%3A1%2C%22
sha256Hash%22%3A%
22d4f55ac16b41ed69c1cded6cfe366fc30f5e4...
%22%7D%7D
&variables=
%7B%20%22id%22%3A%20123%20%7D
```

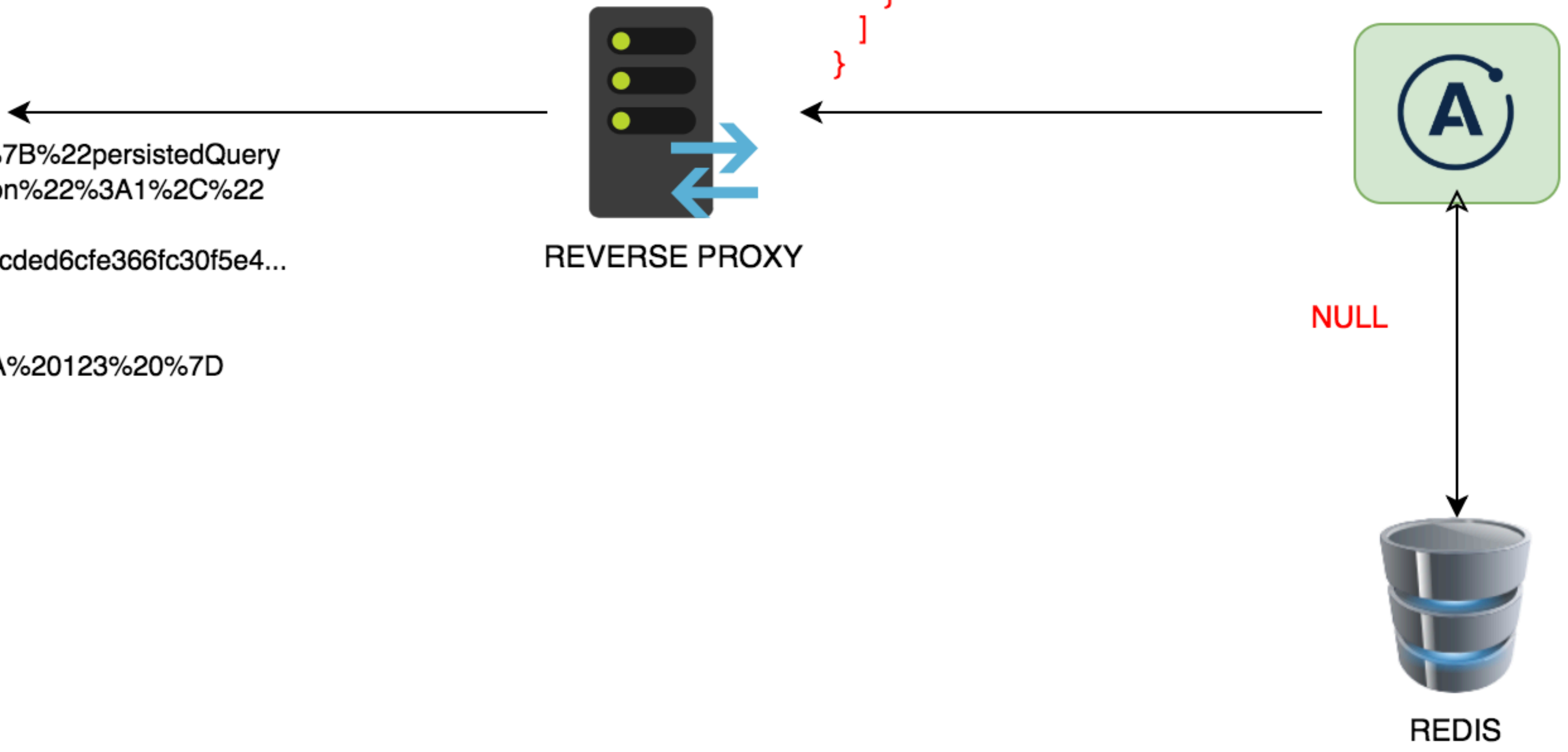
```
/graphql?extensions=%7B%22persistedQuery
%22%3A%7B%22version%22%3A1%2C%22
sha256Hash%22%3A%
22d4f55ac16b41ed69c1cded6cfe366fc30f5e4...
%22%7D%7D
&variables=
%7B%20%22id%22%3A%20123%20%7D
```

```
{
  getVideo($id: ID!) {
    title description
  }
}
```

persisted query

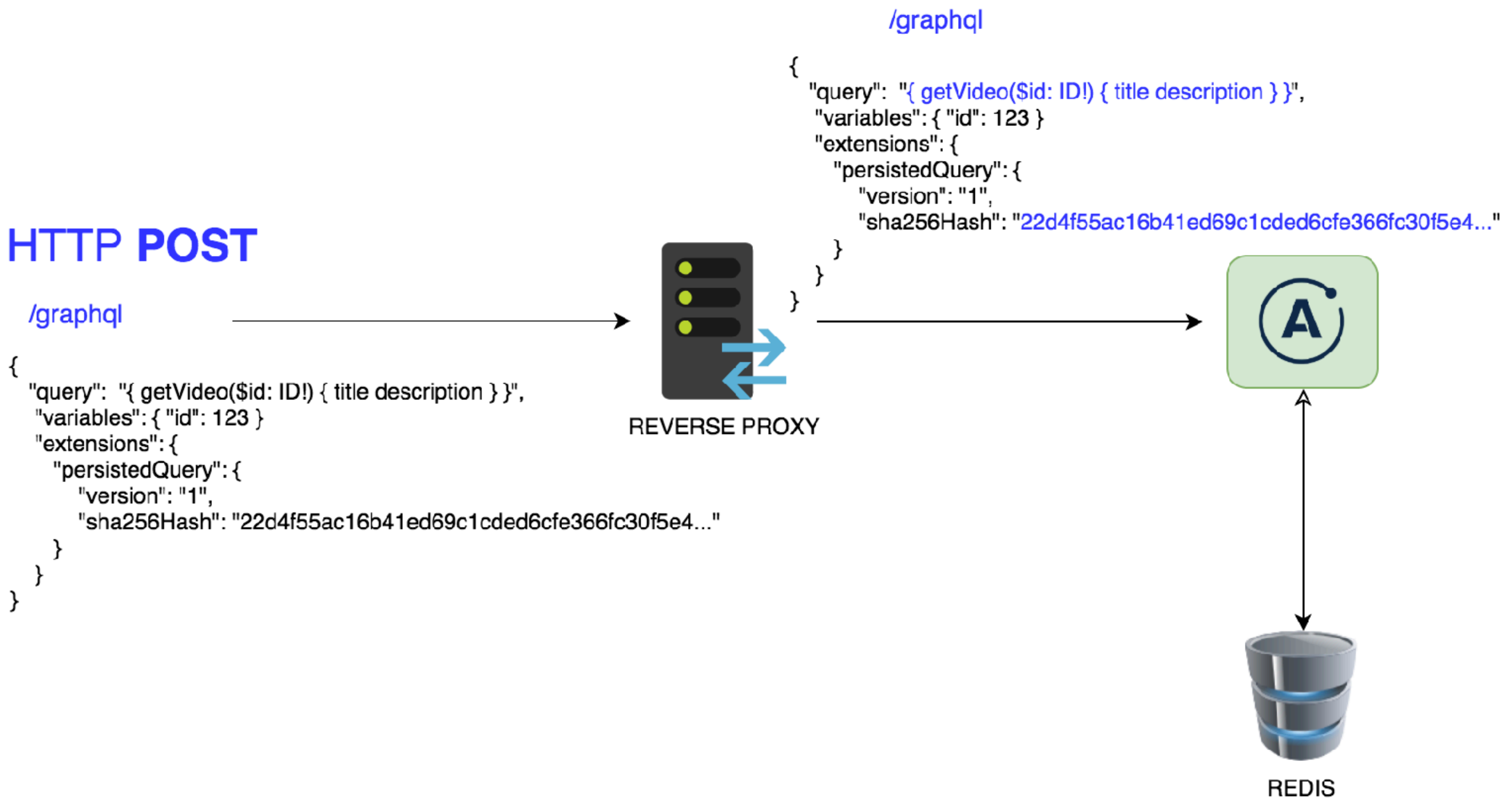
```
{
  "errors":[
    {
      "message": "PersistedQueryNotFound",
      "extensions": {
        "code": "PERSISTED_QUERY_NOT_FOUND"
      }
    }
  ]
}
```

HTTP **GET**



/graphql?**extensions**=%7B%22persistedQuery
%22%3A%7B%22version%22%3A1%2C%22
sha256Hash%22%3A%
22d4f55ac16b41ed69c1cded6cfe366fc30f5e4...
%22%7D%7D
&variables=
%7B%20%22id%22%3A%20123%20%7D

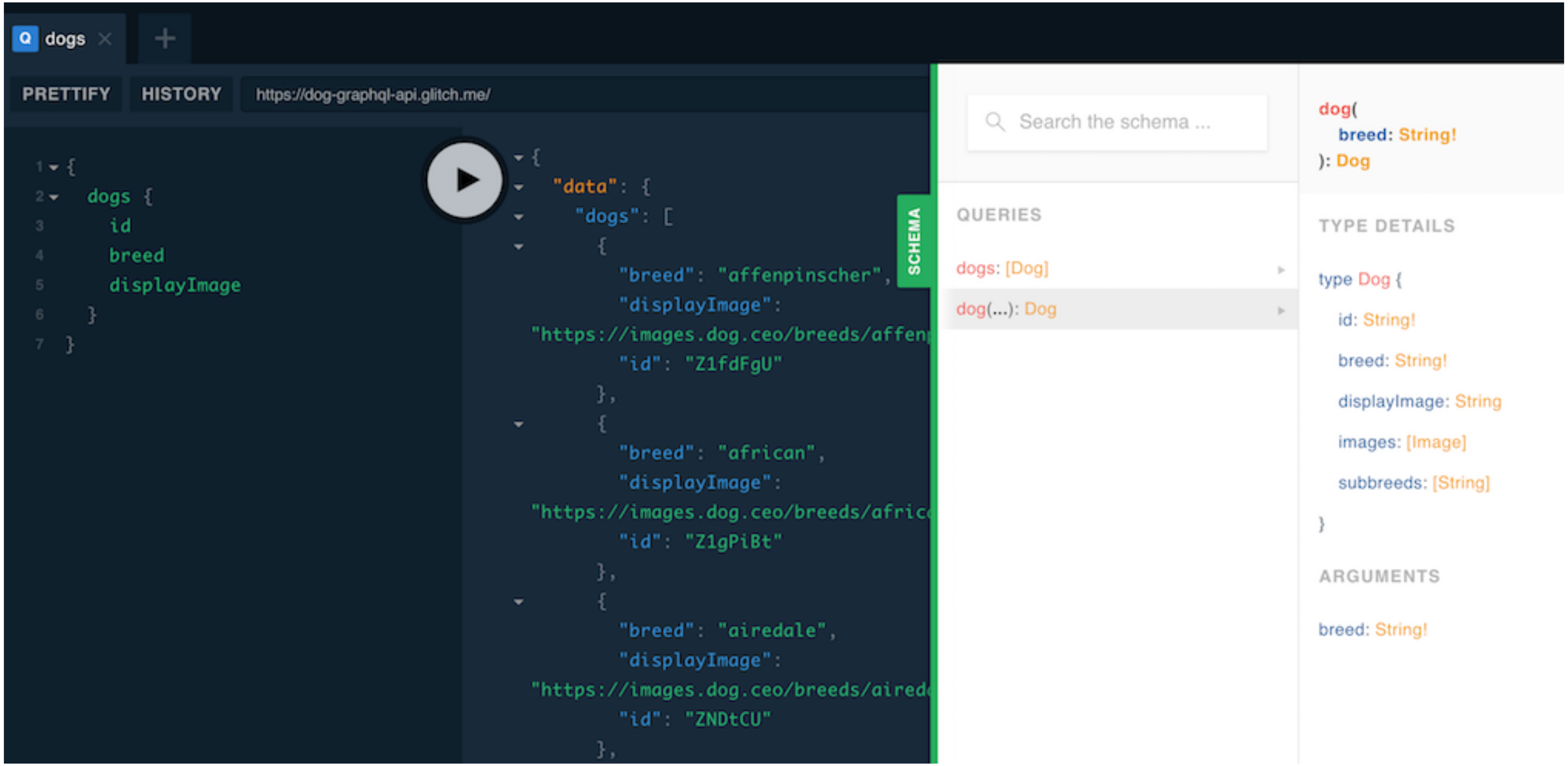
persisted query



```
{
  "query":  "{ getVideo($id: ID!) { title description } }",
   "variables": { "id": 123 }
   "extensions": {
     "persistedQuery": {
        "version": "1",
        "sha256Hash": "22d4f55ac16b41ed69c1cded6cfe366fc30f5e4..."
     }
   }
}
```

```
{
  "query":  "{ getVideo($id: ID!) { title description } }",
   "variables": { "id": 123 }
   "extensions": {
     "persistedQuery": {
        "version": "1",
        "sha256Hash": "22d4f55ac16b41ed69c1cded6cfe366fc30f5e4..."
     }
   }
}
```

tooling - Playground



```
{
  dogs {
    id
    breed
    displayImage
  }
}
```

```
{
  "data": {
    "dogs": [
      {
        "breed": "affenpinscher",
        "displayImage":
"https://images.dog.ceo/breeds/affen
        "id": "Z1fdFgU"
      },
      {
        "breed": "african",
        "displayImage":
"https://images.dog.ceo/breeds/afric
        "id": "Z1gPiBt"
      },
      {
        "breed": "airedale",
        "displayImage":
"https://images.dog.ceo/breeds/aired
        "id": "ZNDtCU"
      },
```

QUERIES

dogs: [Dog]

dog(...): Dog

```
dog(
  breed: String!
): Dog
```

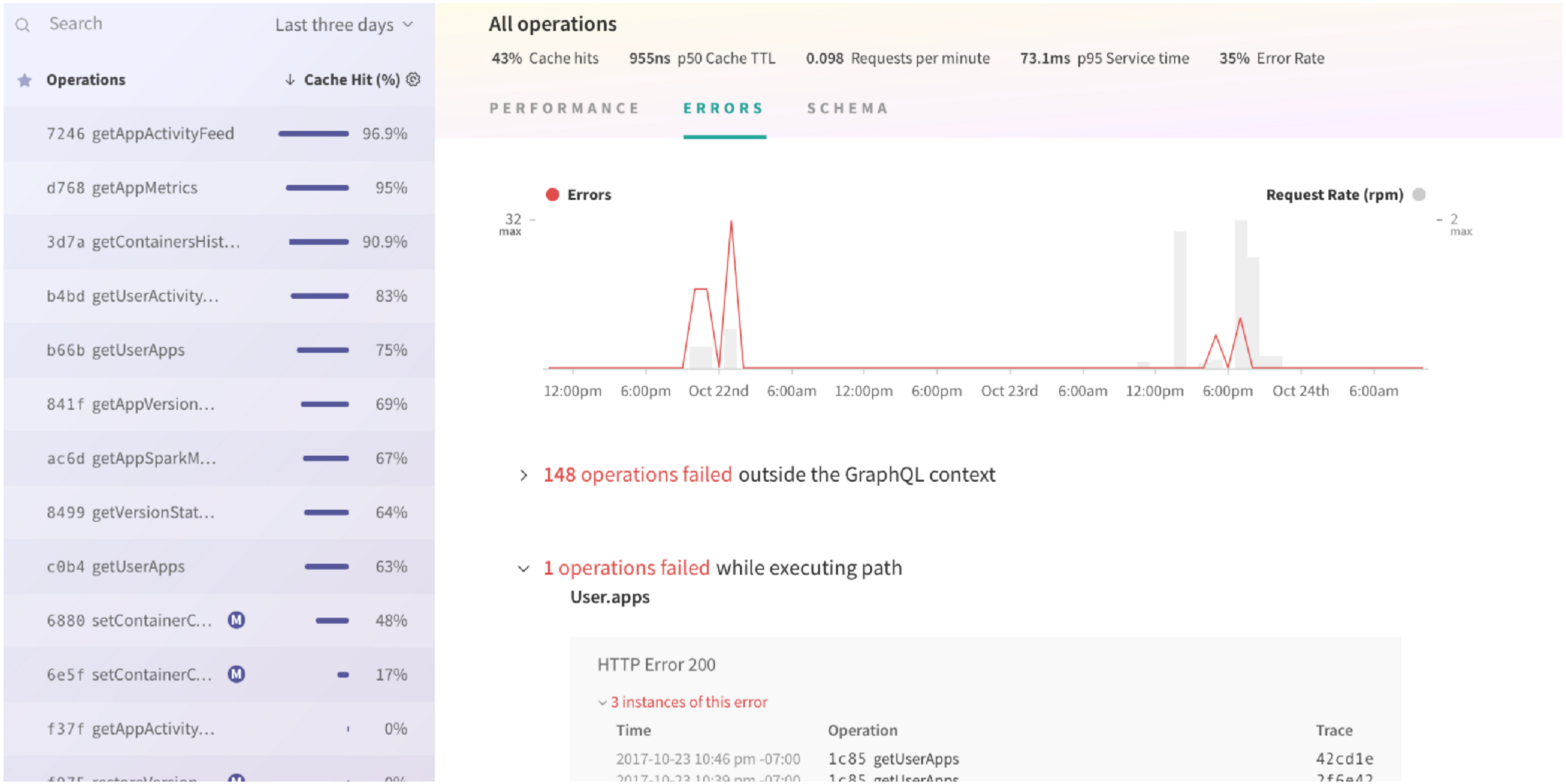
TYPE DETAILS

```
type Dog {
  id: String!
  breed: String!
  displayImage: String
  images: [Image]
  subbreeds: [String]
}
```

ARGUMENTS

breed: String!

tooling - Apollo Engine



Search | Last three days

Operations	Cache Hit (%)
7246 getActivityFeed	96.9%
d768 getAppMetrics	95%
3d7a getContainersHist...	90.9%
b4bd getUserActivity...	83%
b66b getUserApps	75%
841f getAppVersion...	69%
ac6d getAppSparkM...	67%
8499 getVersionStat...	64%
c0b4 getUserApps	63%
6880 setContainerC... (M)	48%
6e5f setContainerC... (M)	17%
f37f getAppActivity...	0%

All operations

43% Cache hits | 955ns p50 Cache TTL | 0.098 Requests per minute | 73.1ms p95 Service time | 35% Error Rate

PERFORMANCE | ERRORS | SCHEMA

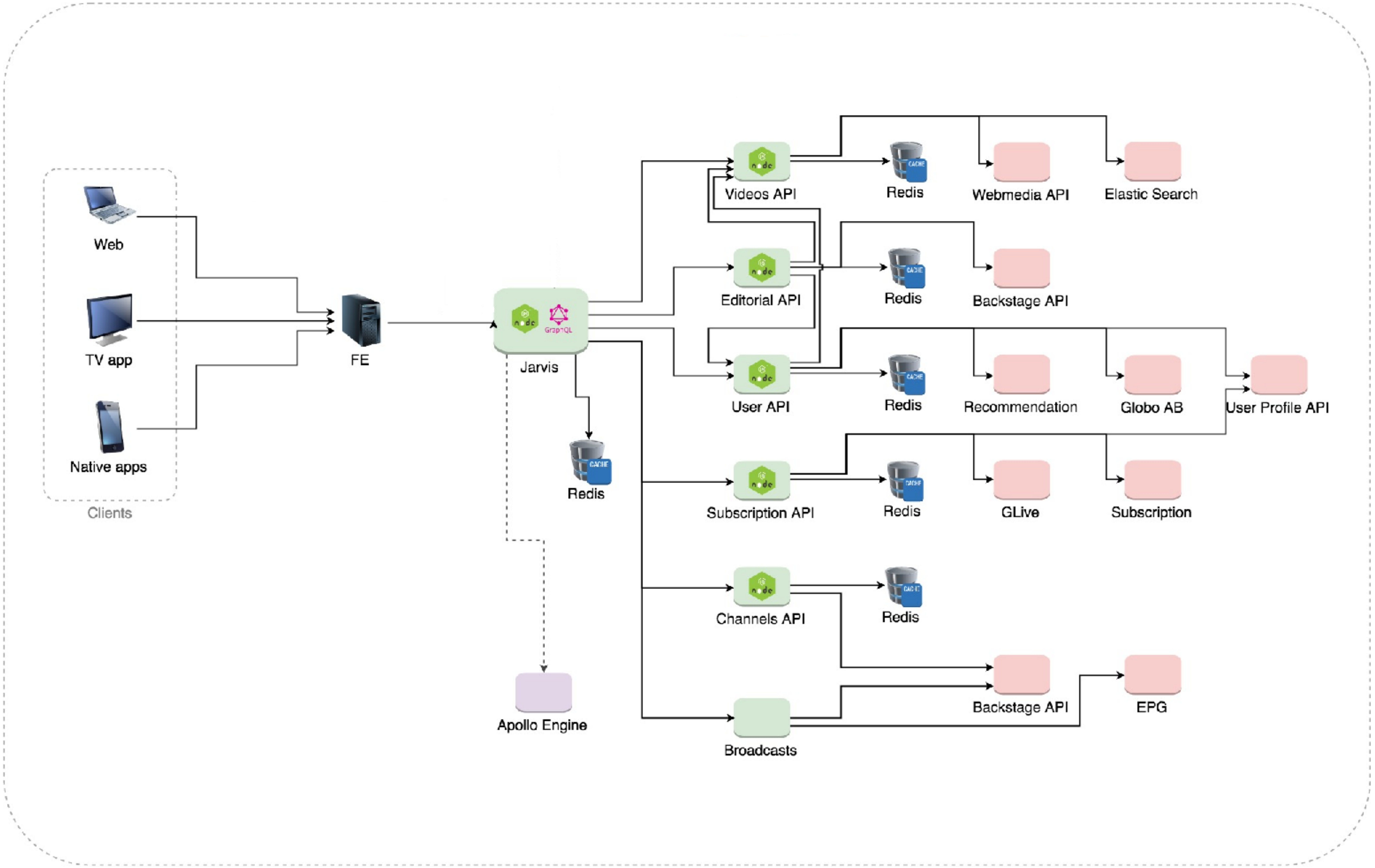
148 operations failed outside the GraphQL context

1 operations failed while executing path
User.apps

HTTP Error 200

3 instances of this error

Time	Operation	Trace
2017-10-23 10:46 pm -07:00	1c85 getUserApps	42cd1e



🟩 Produto 🟥 Other teams 🟪 External

CONSIDERAÇÕES FINAIS

OBRIGADO!

We are hiring talentos.globo.com globo.com