TURNING REST INTO GRAPHQL

Errol Hassall

CONTENTS

BACKGROUND

Why did I do this?

GRAPHQL

What is GraphQL anyway?

DEMO

Lets see how it works

WHAT WE DID

And why you should care

BACKGROUND

I work at a consulting company called Our Very Own.

We were contracted to build an internal application by the Starlight foundation.

We needed the app to talk to multiple REST services.

Most importantly the Microsoft Graph API.

REQUIREMENTS

- Make the calls to the backend easy
- Send as little data as possible
- Make it as fast as possible
- Create one unified interface to all of the external services
- Leverage our strengths of GraphQL

WHAT IS GRAPHQL?

- Only the best thing ever!
- No really its amazing
- It allows you to write a single query that can scale up or down based on the data you need
- Meaning it only sends what you ask for, nothing more, nothing less
- The API does the heavy lifting
- It uses JSON in its response
- Its highly reusable, write once it works for heaps of scenarios
- It doesn’t use XML
- Broken up into three types of queries: query, mutation and subscription

SAMPLE QUERY

```
query ($id: ID!) {
  student (id: $id) {
    id,
    lastName,
    gender,
    studentCode
  }
}
```

```
{
  "data": {
    "student": {
      "studentCode": "1000",
      "lastName": "Abulencia",
      "id": "1",
      "gender": "F"
    }
  }
}
```

SAMPLE MUTATION

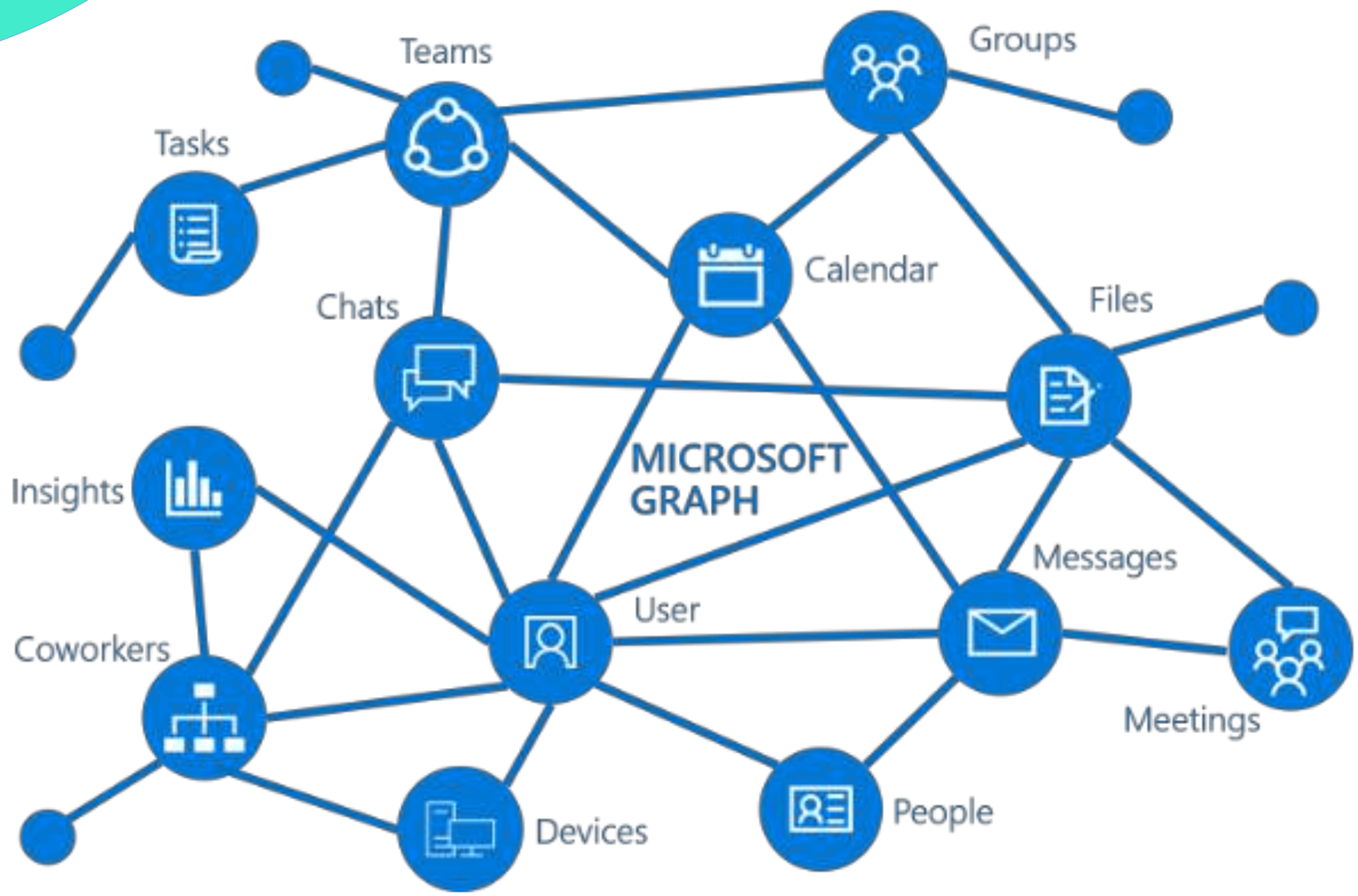
```
mutation CreateStudent(
  $id: ID,
  $studentCode: String!,
  $firstname: String!,
  $lastname: String!,
  $gender: String!,
  $currentgrade: String!,
  $currentclass: String!,
  $schoolId: ID!,
  $newGrade: String!)
{
  createOrUpdateStudent(
    student: {
      id: $id,
      firstName: $firstname,
      lastName: $lastname,
      gender: $gender,
      studentCode: $studentCode,
      active: true,
      currentGrade: $currentgrade,
      currentClass: $currentclass,
      schoolId: $schoolId,
    newGrade: $newGrade}) {
    id,
    firstName,
    lastName,
    gender,
    studentCode,
    schoolId
  }
}
```

```
{
  "studentCode": "9999",
  "firstname": "Errol",
  "lastname": "Hassall",
  "gender": "M",
  "currentgrade": "P",
  "currentclass": "PA",
  "schoolId": 2,
  "newGrade": "1"
}
```

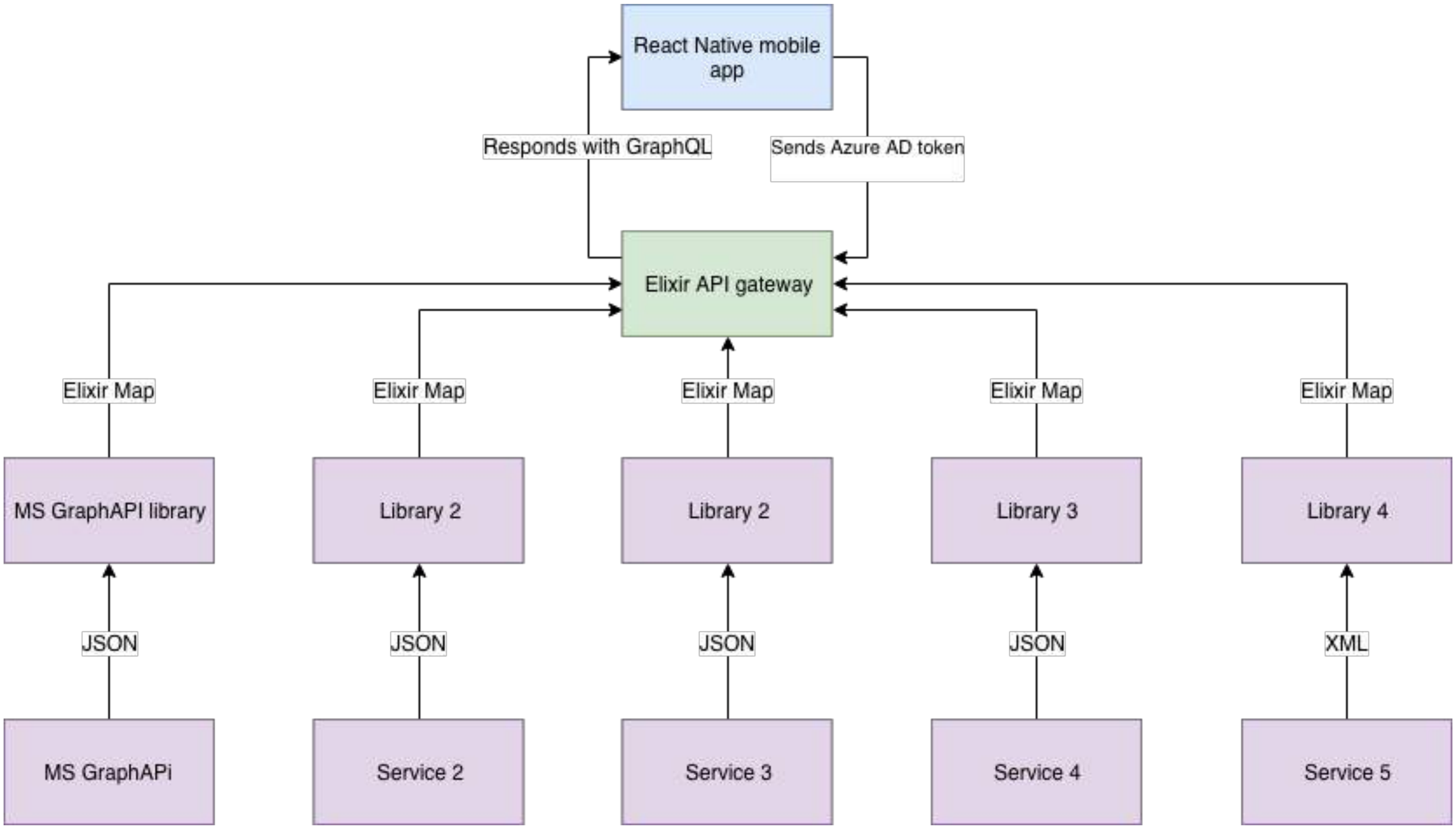
```
{
  "data": {
    "createOrUpdateStudent": {
      "studentCode": "9998",
      "schoolId": "2",
      "lastName": "Hassall",
      "id": "361",
      "gender": "M",
      "firstName": "Errol"
    }
  }
}
```

WHAT IS MS GRAPH API?

- It’s a way of connecting all of Microsofts products via one API with a single Azure Active Directory login
- It uses REST
- Responds with JSON
- Its okay



THE ARCHITECTURE



DEMO

MY PROFILE QUERY


```
{:ok,
 %HTTPoison.Response{
   body: "{\"@odata.context\":\"https://graph.microsoft.com/v1.0/$metadata#users/$entity\",\"id\":\"ff44e747-fc48-4292-8464-30a71749145f\",\"businessPhones\":[],\"displayName\":\"Capconnect test\",\"givenName\":\"Capconnect\",\"jobTitle\":null,\"mail\":\"Capconnecttest@
starlight.org.au\",\"mobilePhone\":null,\"officeLocation\":null,\"preferredLanguage\":null,\"surname\":\"test\",\"userPrincipalName\":\"Capconnecttest@starlight.org.au\"}",
   headers: [
     {"Cache-Control", "private"},
     {"Transfer-Encoding", "chunked"},
     {"Content-Type",
      "application/json;odata.metadata=minimal;odata.streaming=true;IEEE754Compatible=false;charset=utf-8"},
     {"request-id", "da2684d2-5196-4c1e-8aaf-ab4e7cb9c69b"},
     {"client-request-id", "da2684d2-5196-4c1e-8aaf-ab4e7cb9c69b"},
     {"x-ms-ags-diagnostic",
      "{\"ServerInfo\":{\"DataCenter\":\"Southeast Asia\",\"Slice\":\"SliceC\",\"Ring\":\"3\",\"ScaleUnit\":\"002\",\"Host\":\"AGSFE_IN_11\",\"ADSiteName\":\"SEA\"}}"},
     {"OData-Version", "4.0"},
     {"Duration", "104.4341"},
     {"Strict-Transport-Security", "max-age=31536000"},
     {"Date", "Sun, 18 Nov 2018 21:28:04 GMT"}
   ],
   request: %HTTPoison.Request{
     body: "",
     headers: [
       {"Authorization",
        "Bearer eyJ0eXAiOiJKV1QiLCJub25jZSI6IkFRQUJBQUFBQUFDNXVuYTBFVUZnVElGOEVsYXh0V2pUTzR3UjNMa0lVQW5jOVhaLUIwN0l6aUJiejdsYkM4cXhpRnFfYlBqczkySDU2ZzNYQ243SXRIaGVZRUUwWFl4UXNSRkM1NGIzR3FxX043YWFVRHdub3lBQSIsImFsZyI6IlJTMjU2IiwieDV0Ijoid1VMbVlmc3FkUXVXdFZfLWh4VnRESkpaTTRRIiwia2lkIjoid1VMbVlmc3FkUXVXdFZfLWh4VnRESkpaTTRRIn0.eyJhdWQiOiJodHRwczovL2dyYXBoLm1pY3Jvc29mdC5jb20iLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC8zNDFhM2IyZS0wNTM3LTQyNzctYmFlNS1mYzczNGNlMDNlOTYvIiwiaWF0IjoxNTQyNTc2MDc1LCJuYmYiOjE1NDI1NzYwNzUsImV4cCI6MTU0MjU3OTk3NSwiYWNyIjoiMSIsImFpbyI6IjQyUmdZREI4RjZvdVptL2g3eTdxV3VWNisydk1DdkgvNnd2ZU5RbDJKdDZ4MjIxUTBBQUEiLCJhbXIiOlsicHdkIl0sImFwcF9kaXNwbGF5bmFtZSI6IkNhcENvbm5lY3QgdGVzdCIsIm9pZCI6ImZmNDRlNzQ3LWZjNDgtNDI5Mi04NDY0LTMwYTcxNzQ5MTQ1ZiIsInBsYXRmIjoiMiIsInB1aWQiOiIxMDAzQkZGREFFQkJBQ0VCIiwic2NwIjoiVXNlci5SZWFkIiwic3ViIjoiamVrbGQ2QWhWejViZEtza2V5X0hZd0hXNGJscF9jRU9hcFAtQ3pVa3JRWSIsInRpZCI6IjM0MWEzYjJlLTA1MzctNDI3Ny1iYWU1LWZjNzM0Y2UwM2U5NiIsInVuaXF1ZV9uYW1lIjoiQ2FwY29ubmVjdHRlc3RAc3RhcmxpZ2h0Lm9yZy5hdSIsInVwbiI6IkNhcGNvbm5lY3R0ZXN0QHN0YXJsaWdodC5vcmcuYXUiLCJ1dGkiOiI5eENKSUtvcUwwV1BrS3lHbmpVcUFBIiwidmVyIjoiMS4wIiwieG1zX3RjZHQiOjE0MTg5NjAzOTl9.VFOgkqr3QhlyyRiAyYPsGEAweBIpAZ8_0iFSt9EnEiiy2iosQAHS0QJg5ODzjMyFGE5_vtK8vaPsrBpO2lT2bvFv4dmJJK1NxPXe_8BKACi29gUbjv4WR3BjtfBbRjxvQUF1XtSTLwpEB2IocJ0S_3chzrPnZlhxEZRPo8w7CVcIlSGFyKQ91HAX23KPC9brQHpLUxlSpQAY77vdRFgQWxfUrd-IlvCk1UBOhE96snt4-IgjKbz3E-4ltq5Uvmw0SZOn1EsaIFV7uNm6ILn03FMAMRnCnDb0-D4ywqEAw-IK_8pvgWYmWalQWOhg5IHFs11udRBRHTJvc9JtWy-HNw"}
     ],
     method: :get,
     options: [],
     params: %{},
     url: "https://graph.microsoft.com/v1.0/me"
   },
   request_url: "https://graph.microsoft.com/v1.0/me",
   status_code: 200
 }}
```

HOW WE GOT THAT MESS

```elixir
@spec call_graph_api(any(), any()) :: any()
def call_graph_api(azure_token, url \\ @graph_api_url) do
  retry with: exponential_backoff() |> randomize |> cap(1_000) |> expiry(5_000) do
    HTTPoison.get("#{url}me", [
      {"Authorization", "Bearer #{azure_token}"}
    ])
  after
    response -> response
  else
    _error -> {:error, "timeout"}
  end
end
```

MY PROFILE QUERY

```
{:ok,
 %MicrosoftGraphApi.Models.User{
   business_phones: [],
   display_name: "Capconnect test",
   given_name: "Capconnect",
   id: "ff44e747-fc48-4292-8464-30a71749145f",
   job_title: nil,
   mail: "Capconnecttest@starlight.org.au",
   mobile_phone: nil,
   office_location: nil,
   preferred_language: nil,
   surname: "test",
   user_principal_name: "Capconnecttest@starlight.org.au"
 }}
```

THE CODE

CONTEXT

```
def get_user_profile(azure_token) do
  MicrosoftGraphApi.Service.UserService.return_user(azure_token)
end
```

SERVICE

```
def get_user_profile(azure_token) do
  Users.get_user_profile(azure_token)
end
```

RESOLVER

```
def users_profile(_, %{context: context}) do
  UserService.get_user_profile(context.access_token)
end
```

TYPES

```
@desc "Returns a user's profile"
field :users_profile, :ms_user do
  resolve(&CapConnectApi.UserResolver.users_profile/2)
end
```

SCHEMA

```
@desc "MS user model"
object :ms_user do
  field(:id, :id)
  field(:business_phones, list_of(:string))
  field(:display_name, :string)
  field(:given_name, :string)
  field(:job_title, :string)
  field(:mail, :string)
  field(:mobile_phone, :string)
  field(:office_location, :string)
  field(:preferred_language, :string)
  field(:surname, :string)
  field(:user_principal_name, :string)
end
```

MY PROFILE QUERY

```
query {
  usersProfile{
    id
    businessPhones
    displayName
    givenName
    jobTitle
    mail
    mobilePhone
    officeLocation
    preferredLanguage
    surname
    userPrincipalName
  }
}
```


```
{
  "data": {
    "usersProfile": {
      "userPrincipalName": "Capconnecttest@starlight.org.au",
      "surname": "test",
      "preferredLanguage": null,
      "officeLocation": null,
      "mobilePhone": null,
      "mail": "Capconnecttest@starlight.org.au",
      "jobTitle": null,
      "id": "ff44e747-fc48-4292-8464-30a71749145f",
      "givenName": "Capconnect",
      "displayName": "Capconnect test",
      "businessPhones": []
    }
  }
}
```

WE TURNED REST INTO GRAPHQL NOW WHAT?

WE TURNED | REST INTO GRAPHQL NOW WHAT?

- We turned a REST call into a GraphQL call but it was only a small call, so who cares?
- When you have a small call say a single user for instance, its not a problem
- However when you start introducing nesting, it becomes a mess
- With REST if you wanted to grab all the students, then grab their class. You’re going to have a bad time.
- Why is that? Well if you only needed the name for each student, not the entire table, you're going to send lots of useless data.
- In comes nesting in GraphQL

SECOND DEMO

MORE COMPLEX QUERIES

```
query {
  students {
    id,
    active
    firstName,
    gender,
    studentCode,
    newClass{
      id
      label
    }
  }
}
```


```
{
  "data": {
    "students": [
      {
        "studentCode": "1056",
        "newClass": {
          "label": "2B",
          "id": "5"
        },
        "id": "57",
        "gender": "F",
        "firstName": "Christina",
        "active": true
      },
      {
        "studentCode": "1057",
        "newClass": {
          "label": "2C",
          "id": "6"
        },
        "id": "58",
        "gender": "F",
        "firstName": "Mary",
        "active": true
      },
      {
        "studentCode": "1058",
        "newClass": {
          "label": "2A",
          "id": "4"
        },
        "id": "59",
        "gender": "M",
        "firstName": "Aldo",
        "active": true
      },
      {
        "studentCode": "1059",
        "newClass": {
          "label": "2A",
          "id": "4"
        },
        "id": "60",
        "gender": "F",
        "firstName": "Sherlyn",
        "active": true
      },
```

THE CODE

CONTEXT

```
def list_students(school_id) do
  school_id
  |> query_all_by_school_id()
  |> Repo.all()
end
```

SERVICE

```
def all(school_id, context) do
  with :ok <- Bodyguard.permit(__MODULE__, :view_student, context, school_id: school_id) do
    {:ok, Students.list_students(school_id)}
  end
end
```

RESOLVER

```
def all(_, %{context: %{school_id: school_id} = context}) do
  StudentService.all(school_id, context)
end
```

TYPES

```
field :students, list_of(:student) do
  arg(:admin_only, :boolean)
  middleware(Middleware.Authorize, ["super", "admin", "teacher"])
  resolve(&ClasssolverApi.StudentResolver.all/2)
end
```

SCHEMA

```
object :student do
  field(:id, :id)
  field(:first_name, :string)
  field(:last_name, :string)
  field(:gender, :string)
  field(:student_code, :string)
  field(:school_id, :id)
  field(:current_class_id, :id)
  field(:current_grade_id, :id)
  field(:new_grade_id, :id)

  field(:active, :boolean)

  field(:school, non_null(:school)) do
    resolve(dataloader(ClasssolverApi.Students, :school))
  end

  field(:current_class, :current_class) do
    resolve(dataloader(ClasssolverApi.Students, :current_class))
  end

  field(:current_grade, non_null(:school_grade)) do
    resolve(dataloader(ClasssolverApi.Students, :current_grade))
  end

  field(:new_grade, non_null(:school_grade)) do
    resolve(dataloader(ClasssolverApi.Students, :new_grade))
  end

  field(:new_class, :new_class) do
    resolve(&ClasssolverApi.StudentResolver.get_new_class/3)
  end
```

NESTING

- One area that GraphQL really shines is its ability to nest calls as I showed in the student query
- You can keep drilling down to get more of the information you need, when you actually need it
- This provides many benefits to that of a REST endpoint
- We can make the calls much more flexible, meaning we can reuse the calls for different purposes

LET'S WRAP IT UP!

WHAT WE DID

- We turned a REST service that we didn’t own into a complete GraphQL service
- The frontend had no idea, only that they don’t have to make any more REST calls
- The API is much more flexible
- Most of all the mobile app consumes much less data

OUR VERY OWN