1. Graphql 101

1.1. From the get go

- graphql is a **communication standard**
- graphql is not a programing language
- objective:

> "...for **describing** the capabilities and requirements of data models for client-server applications"

- **self-documented**:

> Ensure that all of your data is statically typed and these types inform what queries the schema supports.

- **included deprecation mechanism**

> Reduce the need for breaking changes, but utilize a built-in mechanism for deprecations when you need to.

- **data source Agnostic** "GraphQL does not mandate a particular programming language or storage system for application services that implement it"
- **you get what you ask for**:
  - GraphQL queries are **Field Sets**
  - field -> function **field resolver**

GraphQL **principles**:

1. Product-centric: ***GraphQL is unapologetically driven by the requirements of views and the front-end engineers that write them***.
   - "Client First", me, 2023
   - > "designed to build client applications by providing an intuitive and flexible syntax and system for describing their data requirements and interactions." GraphQL Spec, 2021
2. Hierarchical

3. Strong-typing
4. Client-specified response
5. Introspective

1.2. Overview

Its just HTTPS, auth whatever you like, client and server interact throught POST json body:

- query: a read-only fetch.

```
type Query {
    books: [Book!]!
}
query GetBooks {
  books {
      title
      author
  }
}
```

- mutation: a write followed by a fetch.

```
mutation {
    likeStory(storyID: 12345) {
        story {
            likeCount
        }
    }
}
```

- subscription: a long-lived request that fetches data in response to source events.
    - web sockets generally used
    - supports EventDriven archs

1.2.1. (Partial) Language Overview

For completeness, check the GraphQL Spec Document.

- type

```
type Person {
    name: String
    birthdate: Date
    picture: Url
}
```

- interface

```
interface Person {
    name: String!
    birthdate: Date!
    picture: Url
}
type Book {
    title: String!
    author: Author!
    publication_date: Date!
}
type Author implements Person {
    name: String!
    birthdate: Date!
    picture: Url
    books: [Book]
}
```

- union

```
union SearchResult = Book | Author
```

- enum

```
enum CardinalDirection {
  NORTH
  EAST
  SOUTH
  WEST
}
```

- non-null: `name: String!`
- Field Arguments

```
type Person {
    name: String
    picture(size: Int):
}
{
    name
    picture(size: 600)
}
```

- input objects:

```
  input Point2D {
    x: Float
    y: Float
  }

  {
    closestBathrooms(from: Point2D): [Bathroom]
  }
```

1.2.2. Fields and Field Resolvers

- Selection Set
- Field Alias

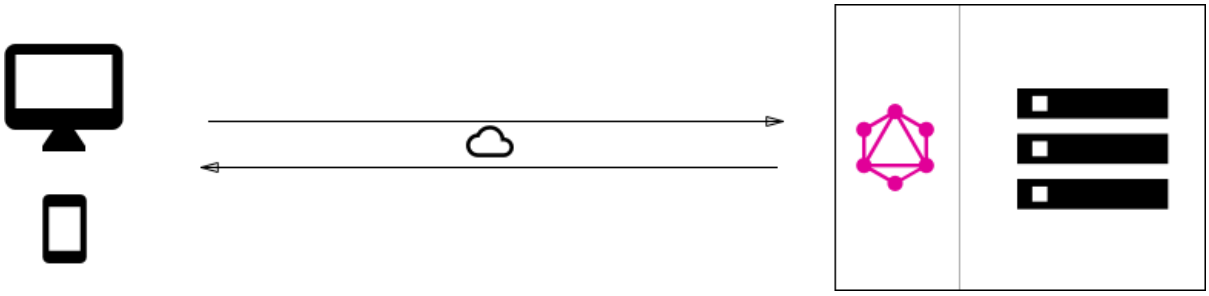
1.2.3. Fragments

- primary unit of composition
- recycle and reuse common pieces of queries
- inline fragments ???

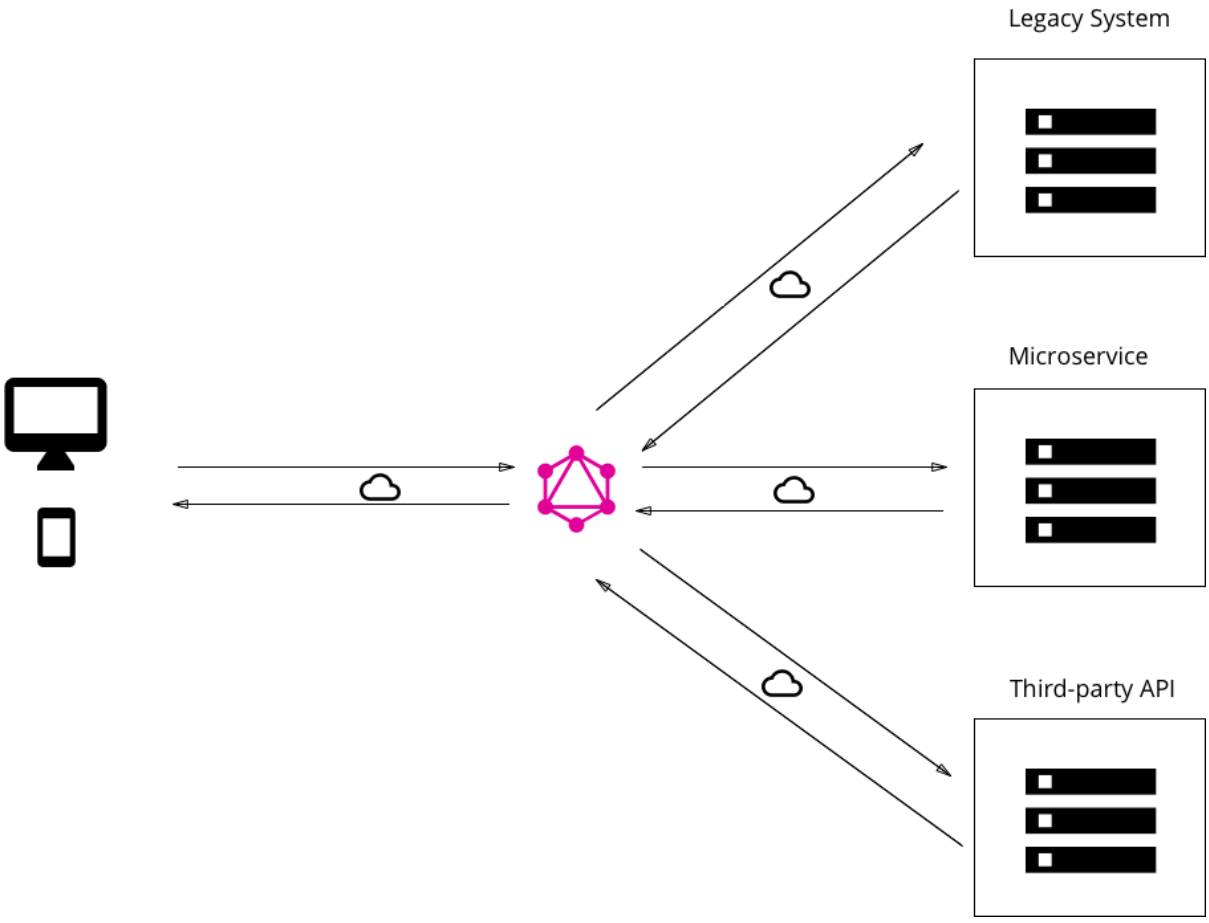
```
type User {
  # a bunch of fields...
}
type Adress {
  # like 100 fields, i know, crazy.
}

fragment friendFields on User {
  id
  name
  profilePic(size: 50)
}

fragment simpleAddress on Address {
  line1
  line2
  city
  country
}

# QUERY:
{
  user(id: "4") {
```

```
    friends(first: 10) {
      ...friendFields
      address {
        ...simpleAddress
      }
    }
    mutualFriends(first: 10) {
      ...friendFields
      address {
        ...simpleAddress
      }
    }
  }
}
```

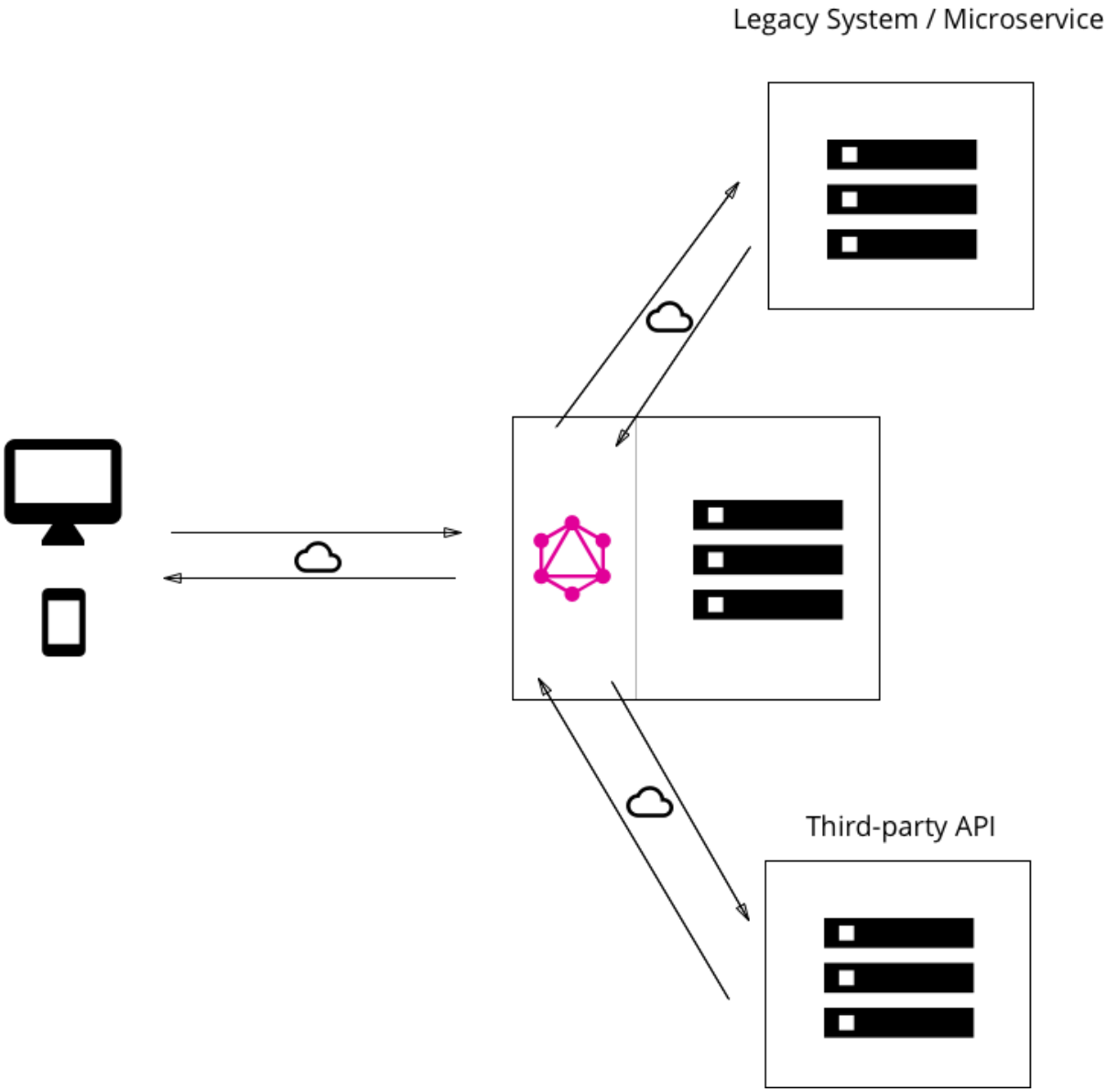
1.3. Architecture Examples



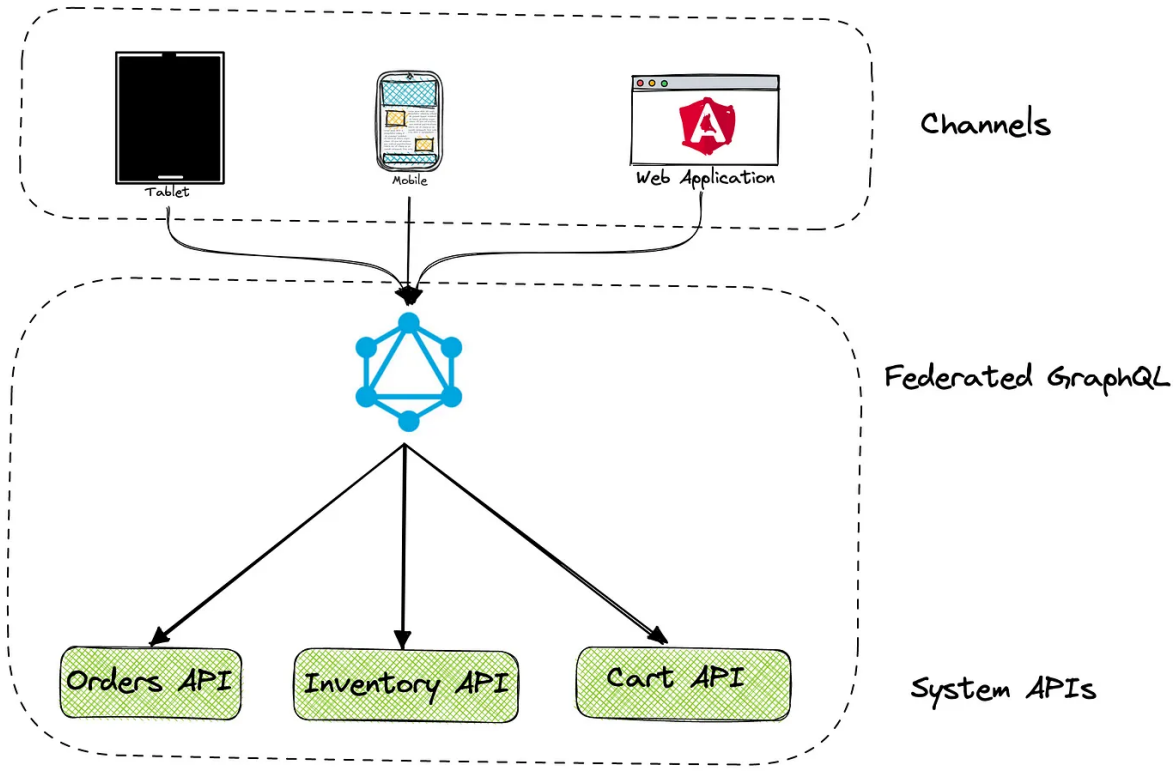
GraphQL server and db in same vm serving mobile and web clients [1]



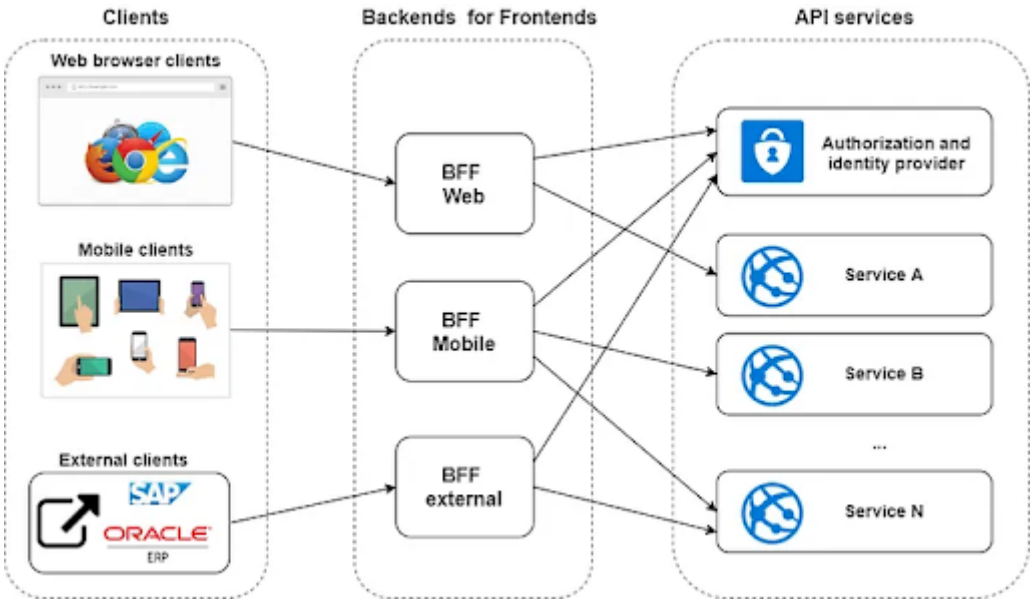
graphql server in dedicated 'orchestator' node in microservice arch with 3 different data sources [3]



graphql server with db in same vm while also orchestrating with two external data sources [3]



GraphQL Api Gateway example[3]

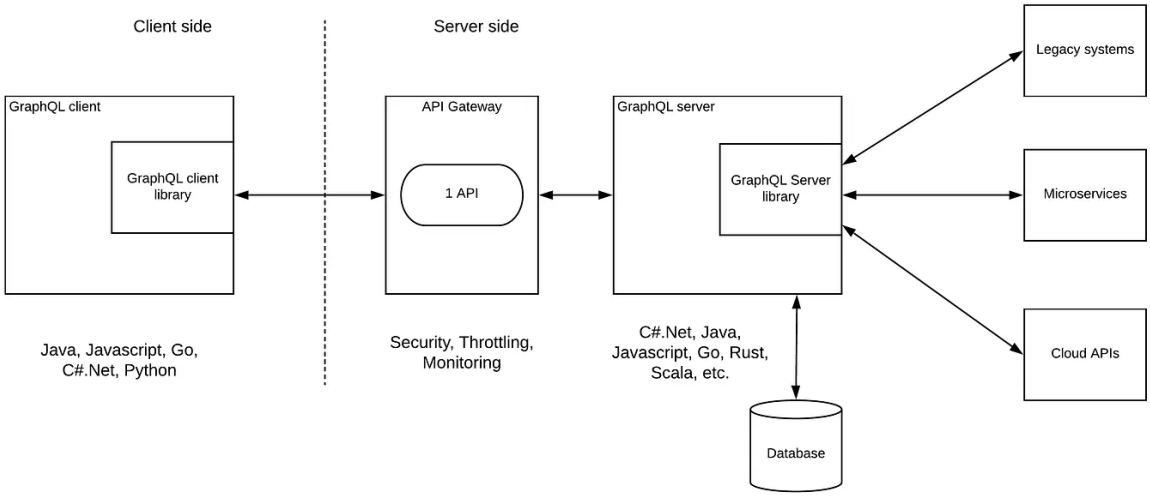


Backend For Frontend

1.3.1. Authentication

A secure Server has some sort of Authentication:

- Basic Auth: base64(user:password) (just dont use this) `curl --header "Authorization: Basic am9objpzZWNyZXQ=" my-website.com`
- Bearer Tokens:
  - JSON Web Tokens (JWT, normal RSA in payload + signature), header , payload + signature
  - OAuth 2.0: 1 Authorization (email+pass, 3rdP Identity Provider) then Bearer is Session token.

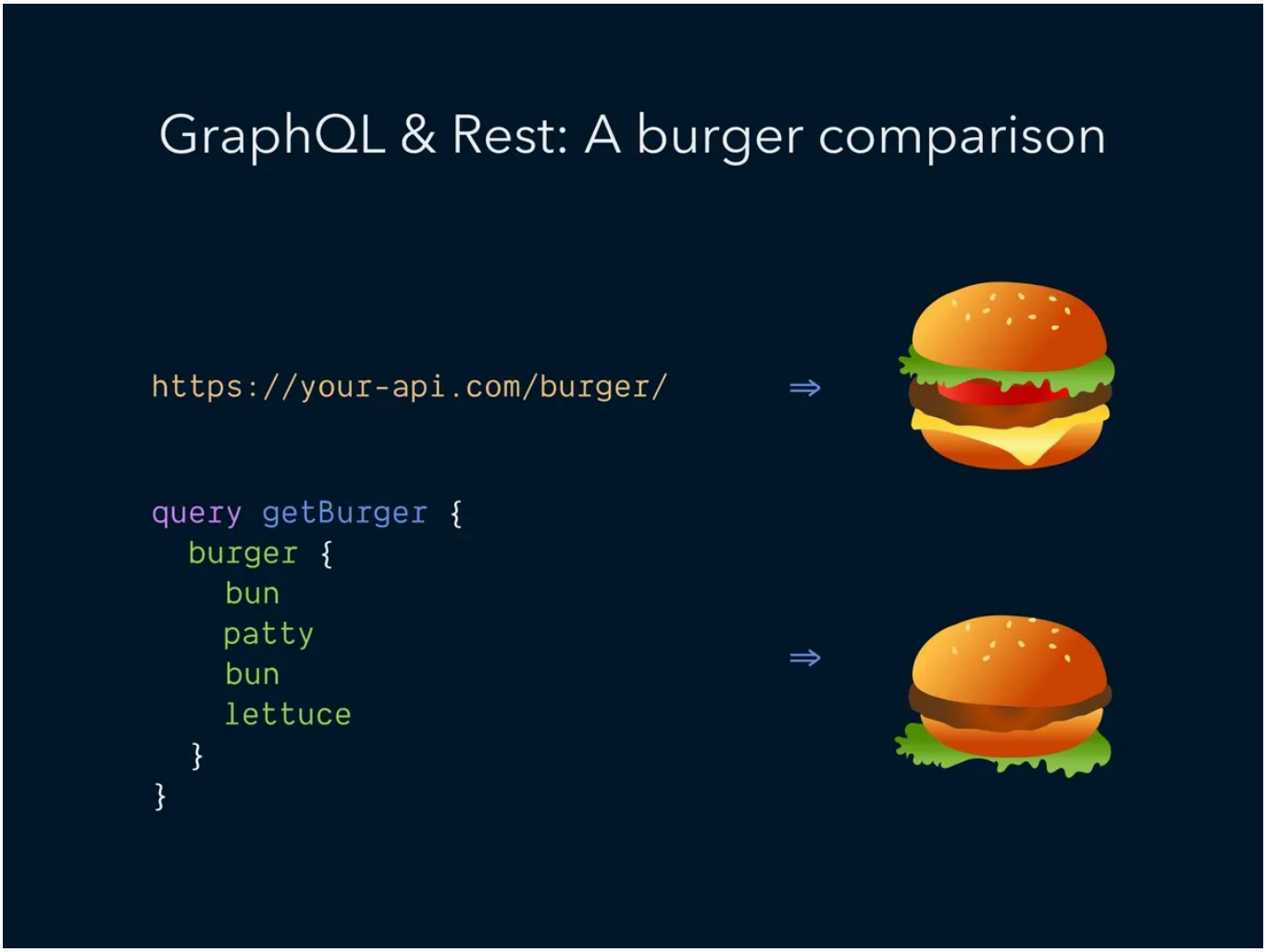


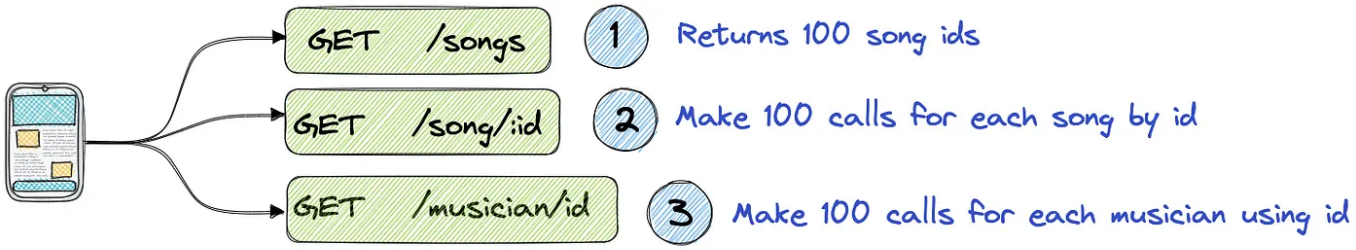
An Authentication Layer in front of the GraphQL service.

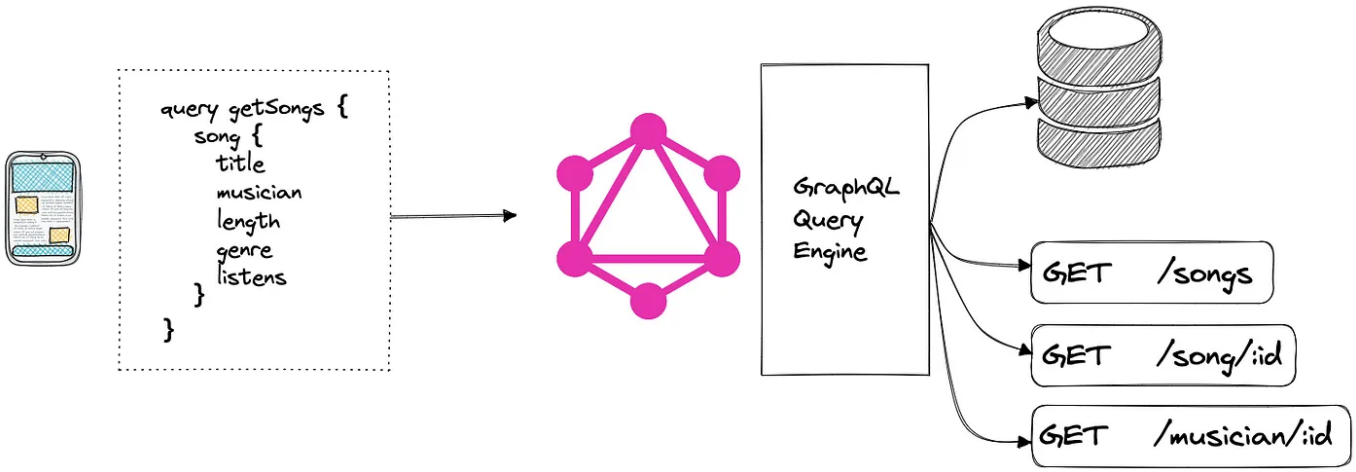
1.3.2. Performance vs REST

- 1 field -> 1 resolver function
- data batching on the server in stead of client -> less http calls for same data
- catered query for client -> allows for mutiple different clients, same endpoint fullfills different needs

performance improvements in frontend:







1.4. Instrospection

```
{
  __type(name: "Book") {
    name
    fields {
      name
      type {
        name
        kind
      }
    }
  }
}
```

1.5. Sources

- GraphQL Spec October2021
- graphql.org
- howtographql.com: Big Picture (Architecture)
- Solution Architects Guide to GraphQL
- Introduction to GraphQL
- GraphQL-based Architecture Patterns
- GraphQL.org: Instrospection

- Apollo Server: Union and Interfaces
- 12 Microservices Patterns I Wish I Knew Before the System Design Interview