GraphQL 101

by R. Apú

June, 2023

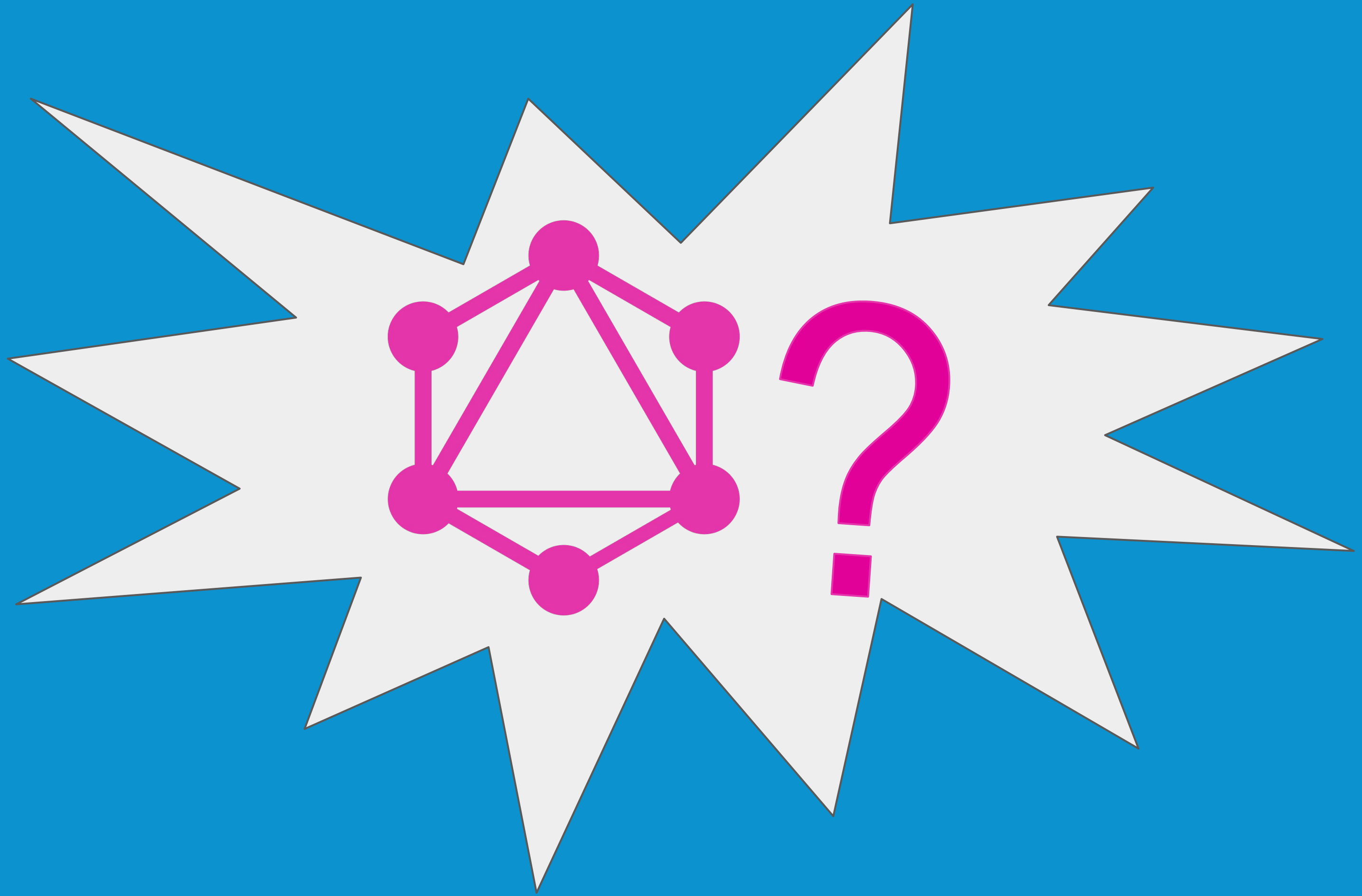
1 What?

2 Why?

GraphQL vs REST

3 How?

Architecture Examples



What is GraphQL?

“a query language and execution engine originally created at Facebook in 2012 for describing the capabilities and requirements of data models for client-server applications.”

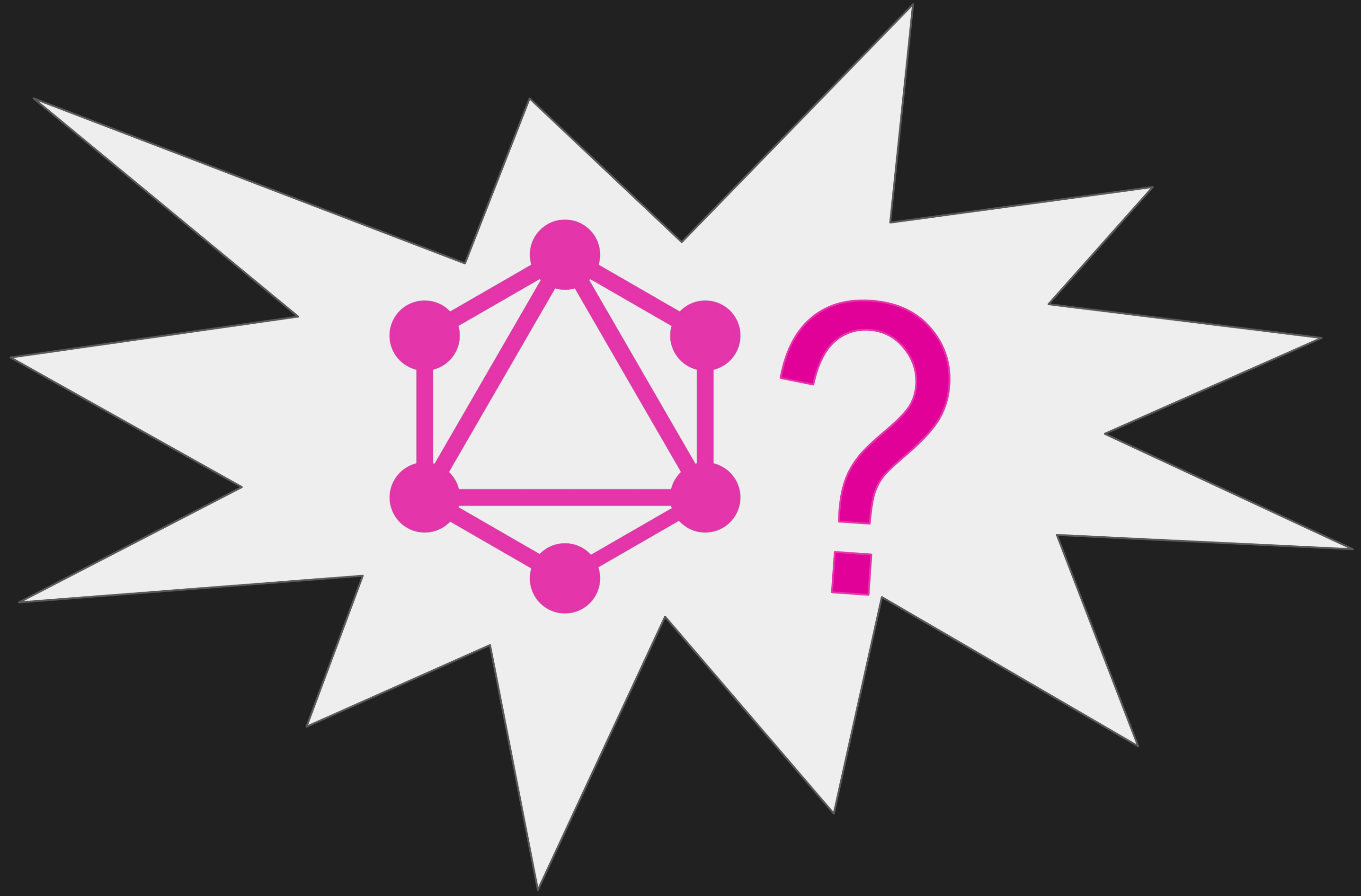
GraphQL Spec (October, 2021)

GraphQL is...

- ... a standard of a language to communicate between clients and servers.
- ... self documented once an engine is deployed
- ... data source agnostic
- ... used in gateways in microservices archs.
- ... declarative
- ... used for both queries and data manipulation

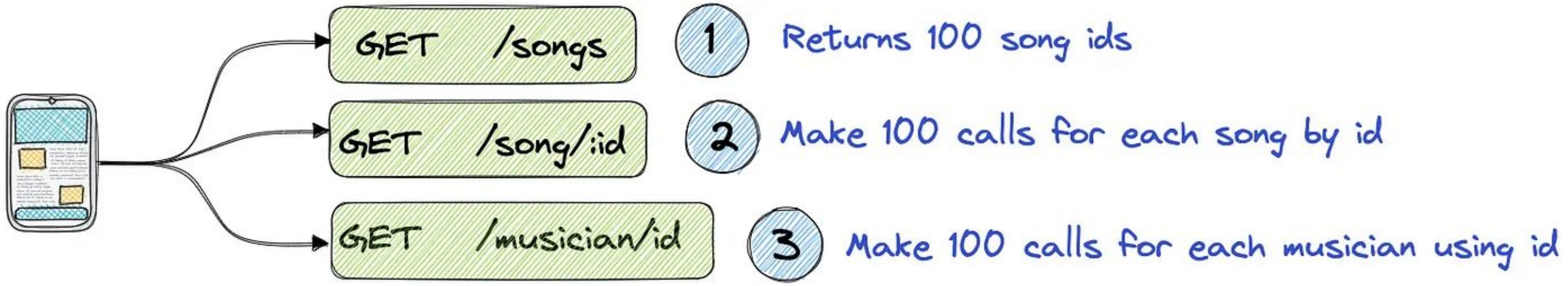
GraphQL *isn' t*...

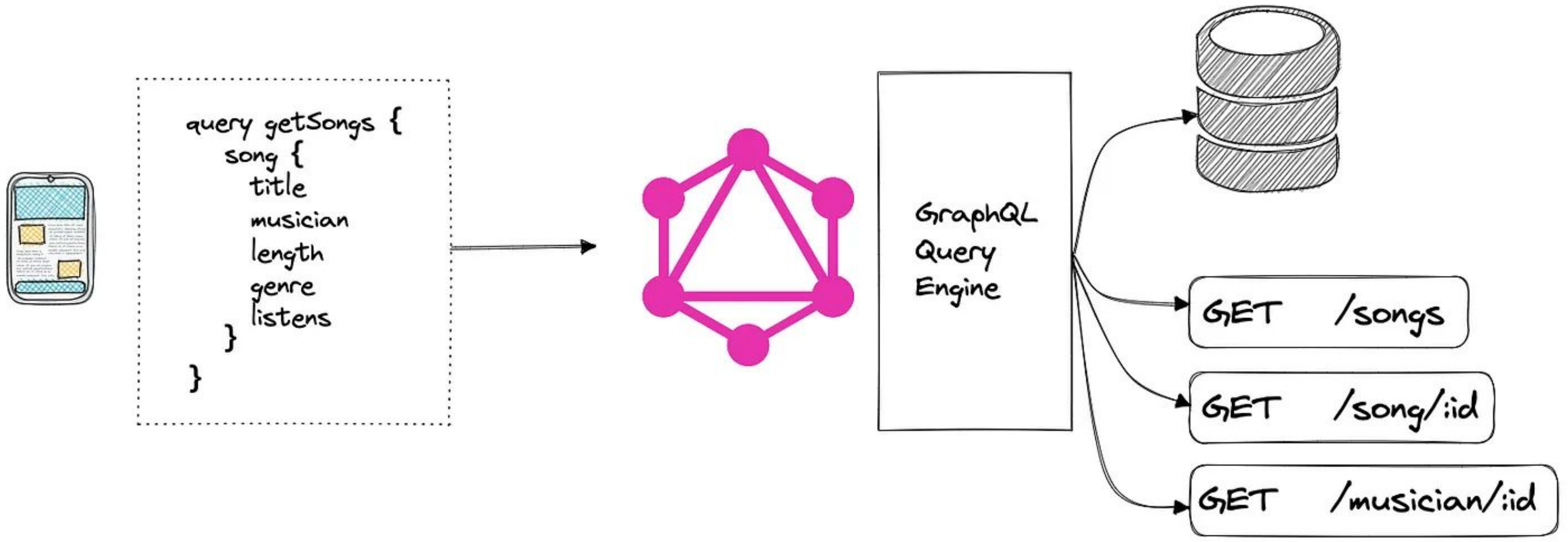
- ...a programming language
- ... a relational database management language like SQL.
- ... implemented in a central repository.



Why use GraphQL?

1. Catered Queries

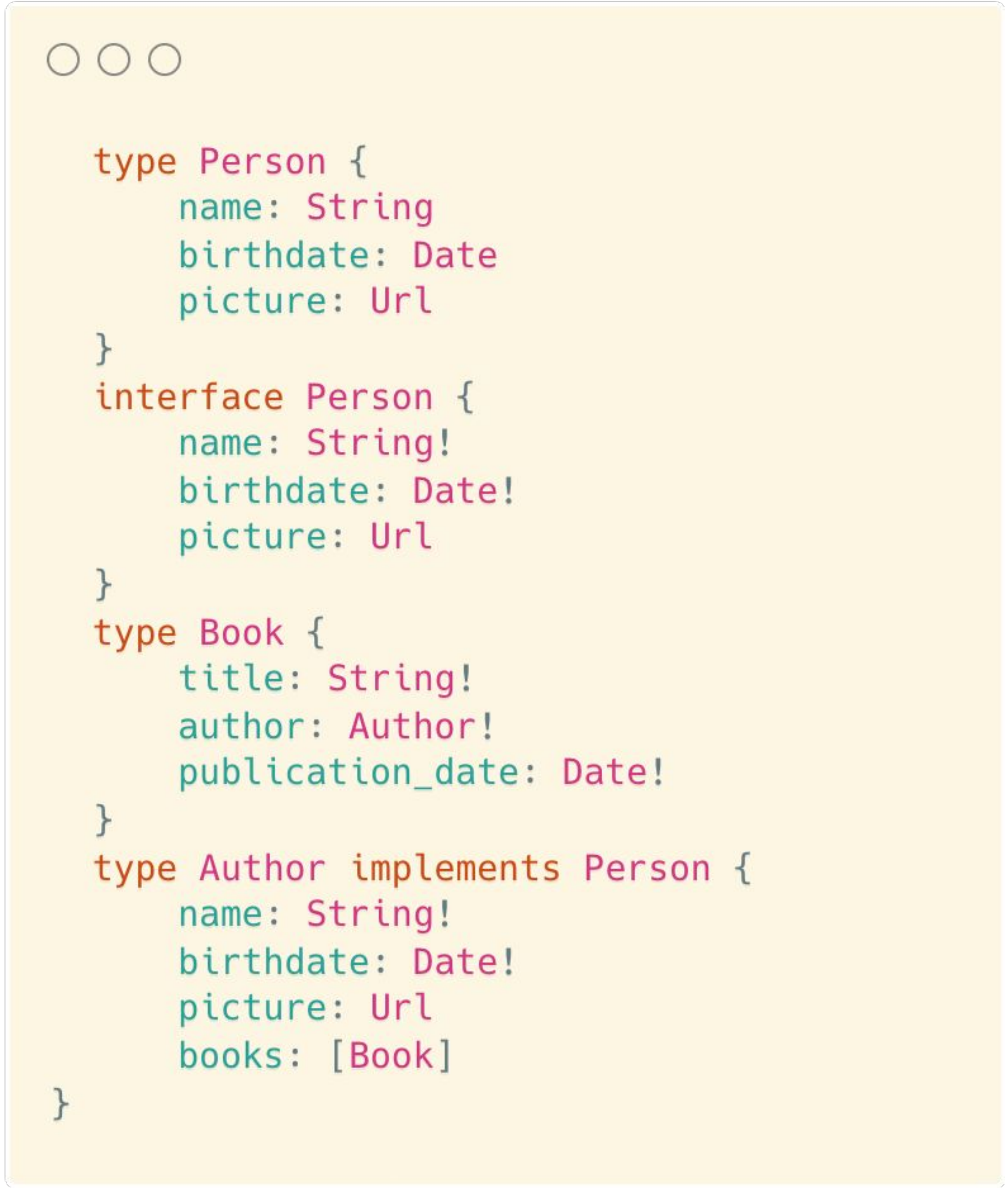

GET /songs
GET /song/:id
GET /musician/id
1
2
3
Returns 100 song ids
Make 100 calls for each song by id
Make 100 calls for each musician using id


```
query getSongs {
    song {
      title
      musician
      length
      genre
      listens
    }
}
```

2. Strongly typed schemas

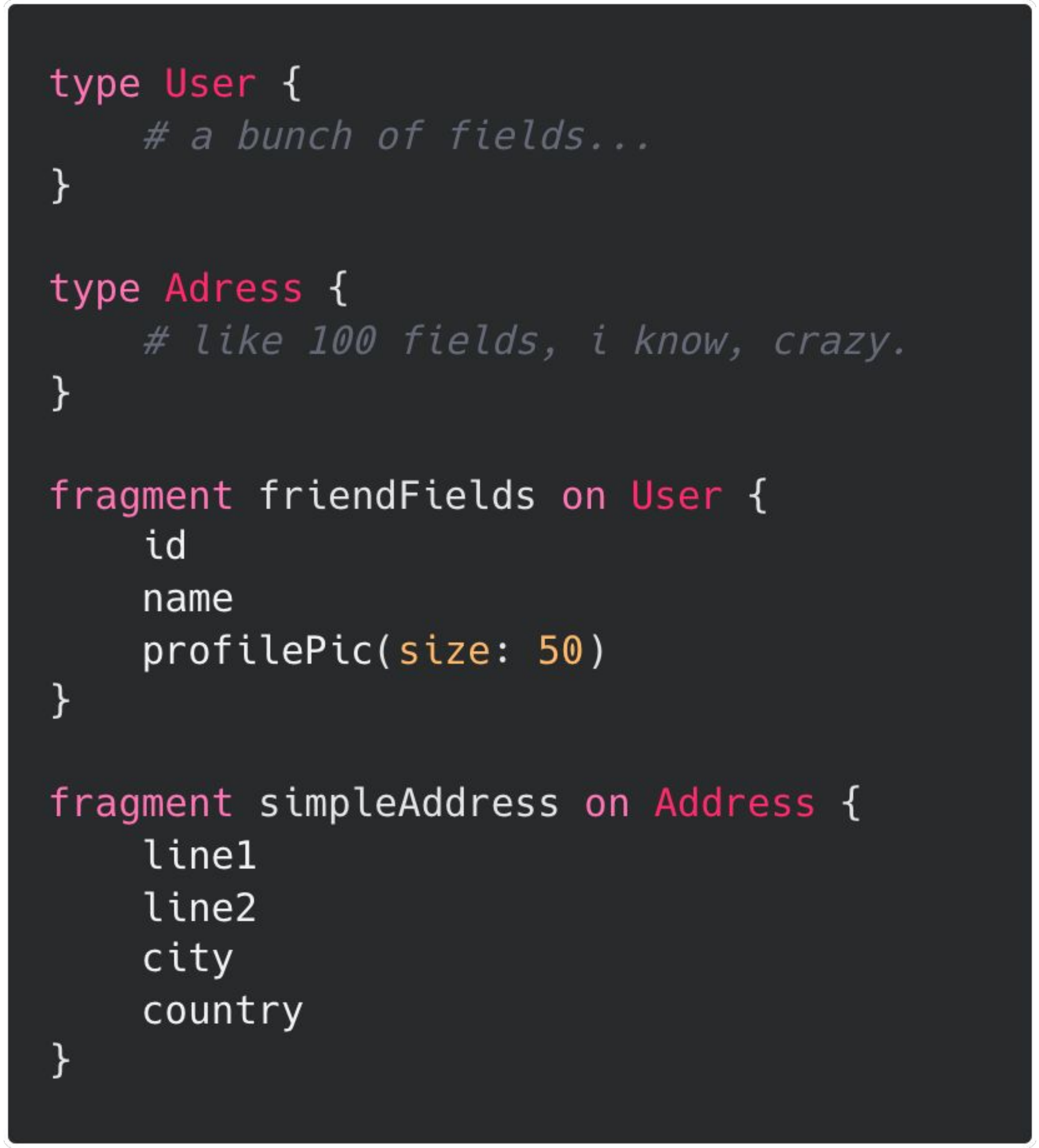
Language specifies:

- types,
- interfaces,
- unions,
- enums,
- field arguments,
- polymorphism,
- fragments

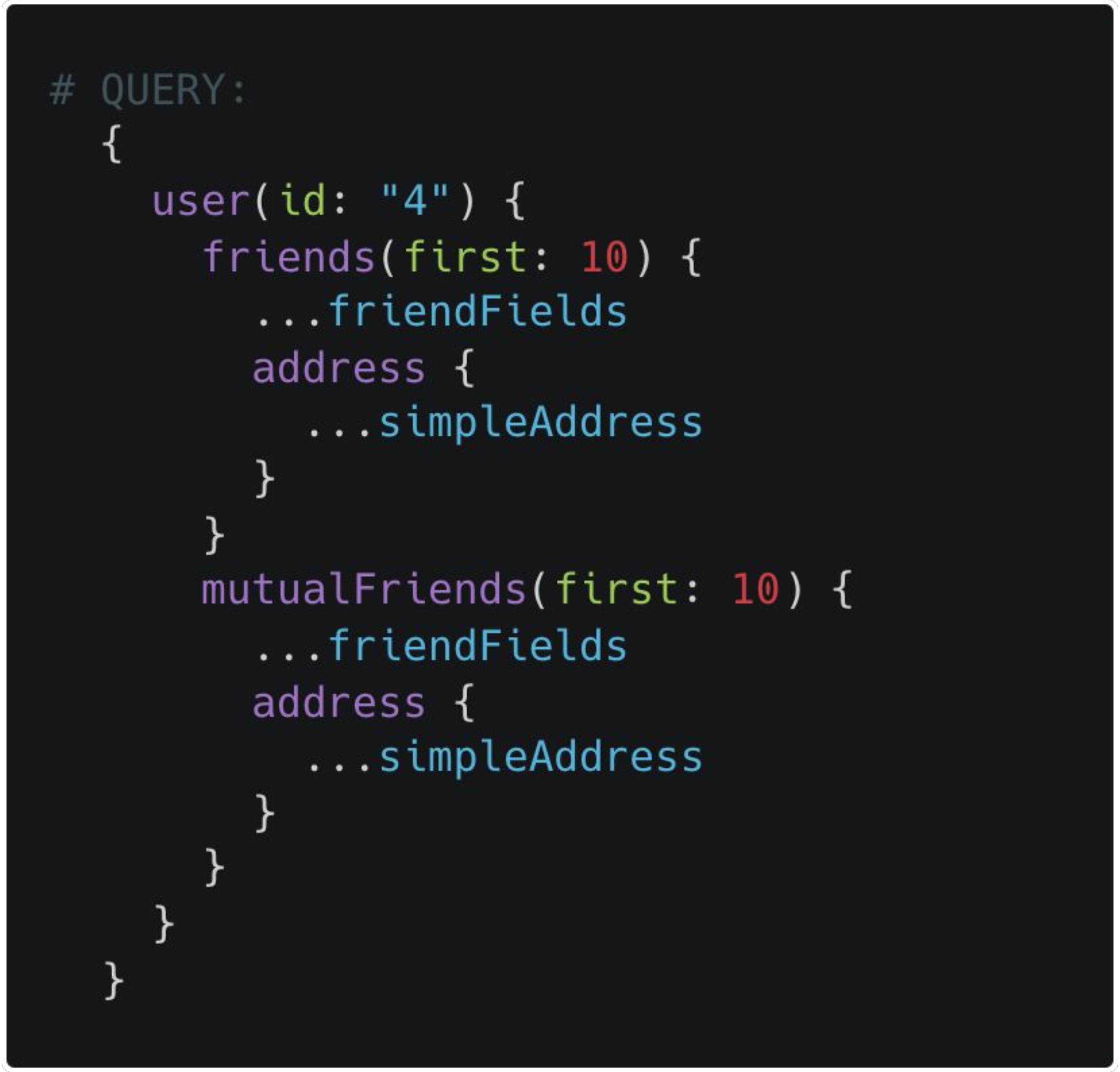


```
  type Person {
      name: String
      birthdate: Date
      picture: Url
  }
  interface Person {
      name: String!
      birthdate: Date!
      picture: Url
  }
  type Book {
      title: String!
      author: Author!
      publication_date: Date!
  }
  type Author implements Person {
    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

3. Composition tools: Fragments

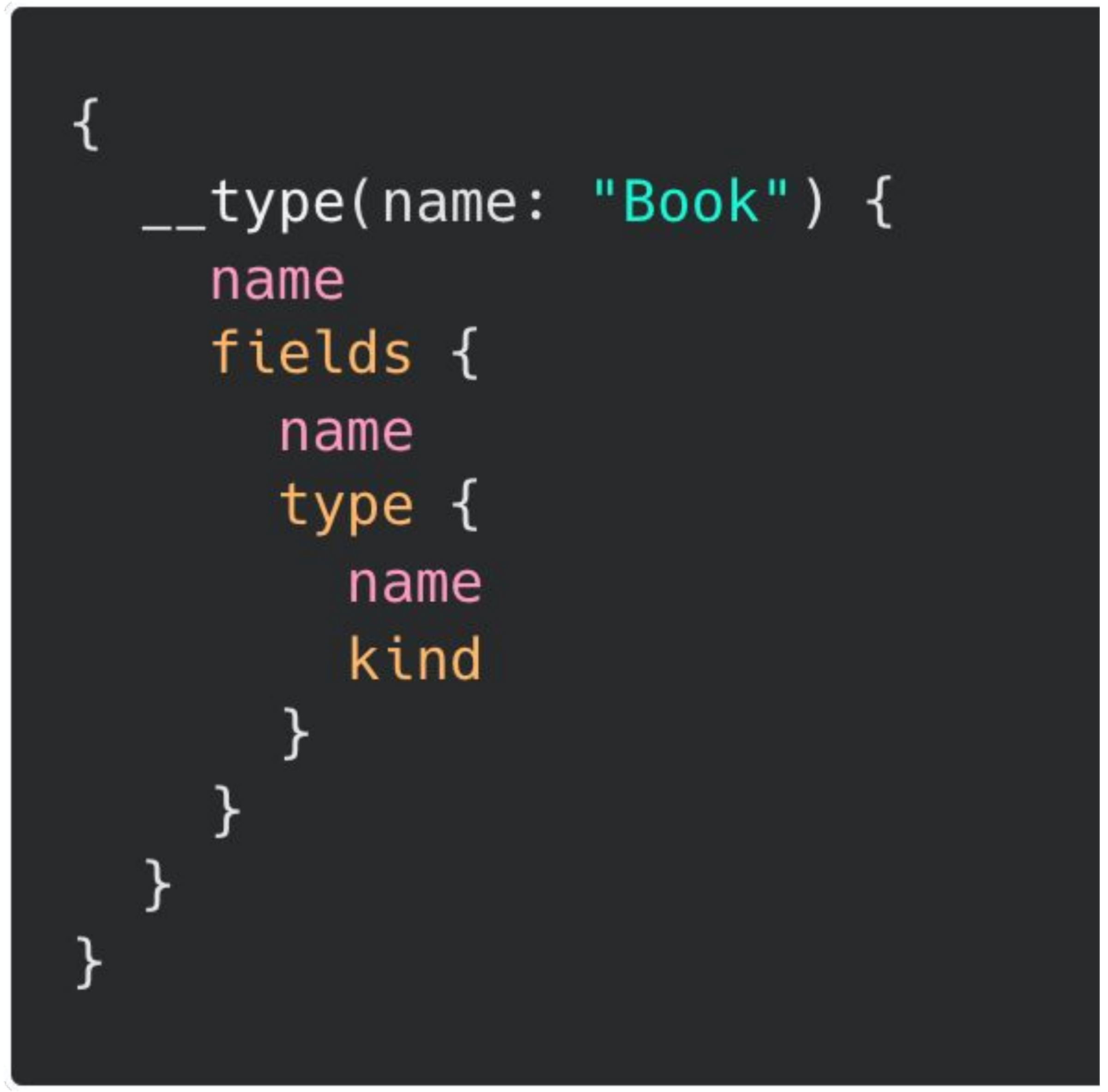


```
type User {
    # a bunch of fields...
}

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
}
```

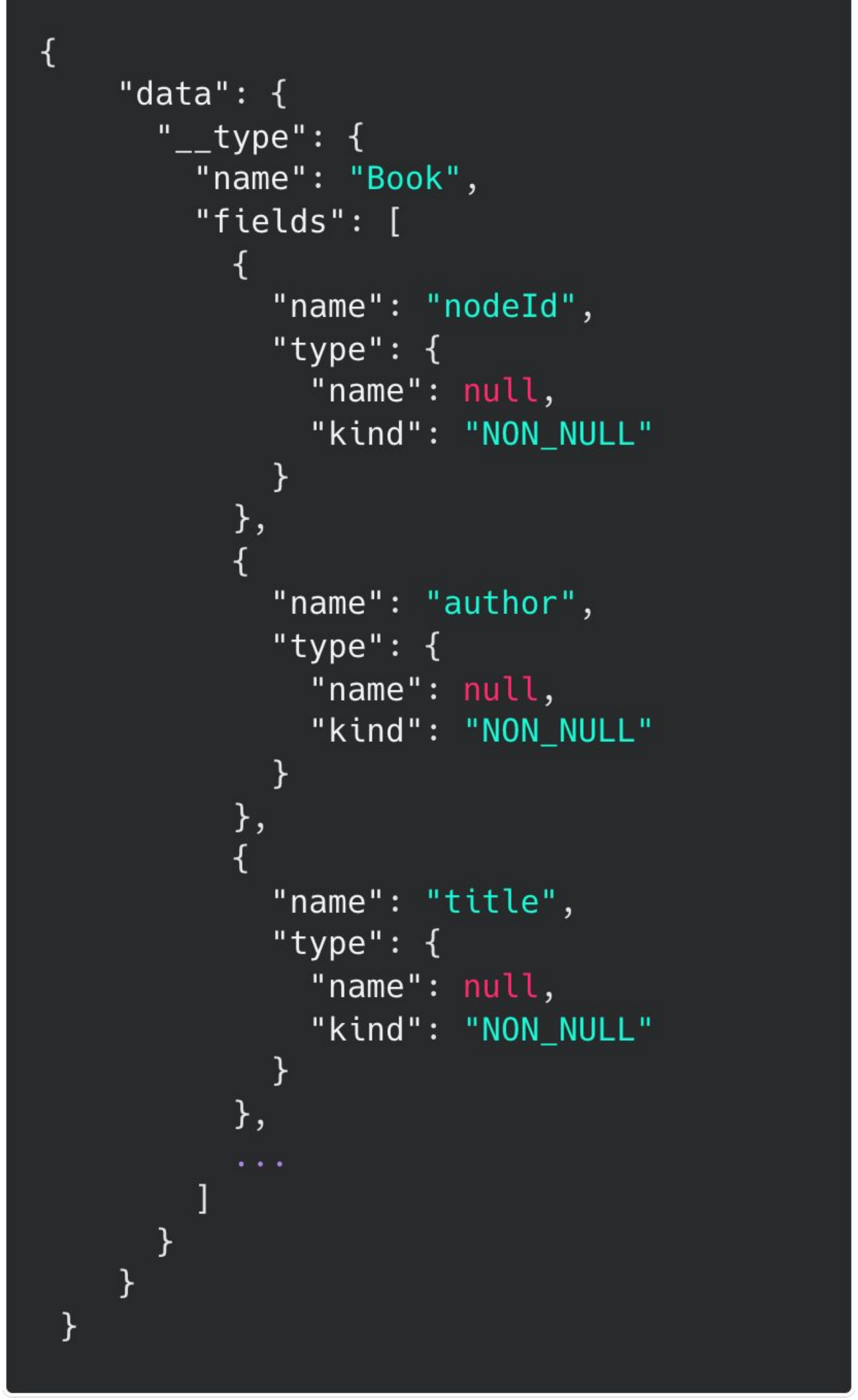


```
# QUERY:
  {
    user(id: "4") {
      friends(first: 10) {
        ...friendFields
        address {
          ...simpleAddress
        }
      }
      mutualFriends(first: 10) {
        ...friendFields
        address {
          ...simpleAddress
        }
      }
    }
  }
```

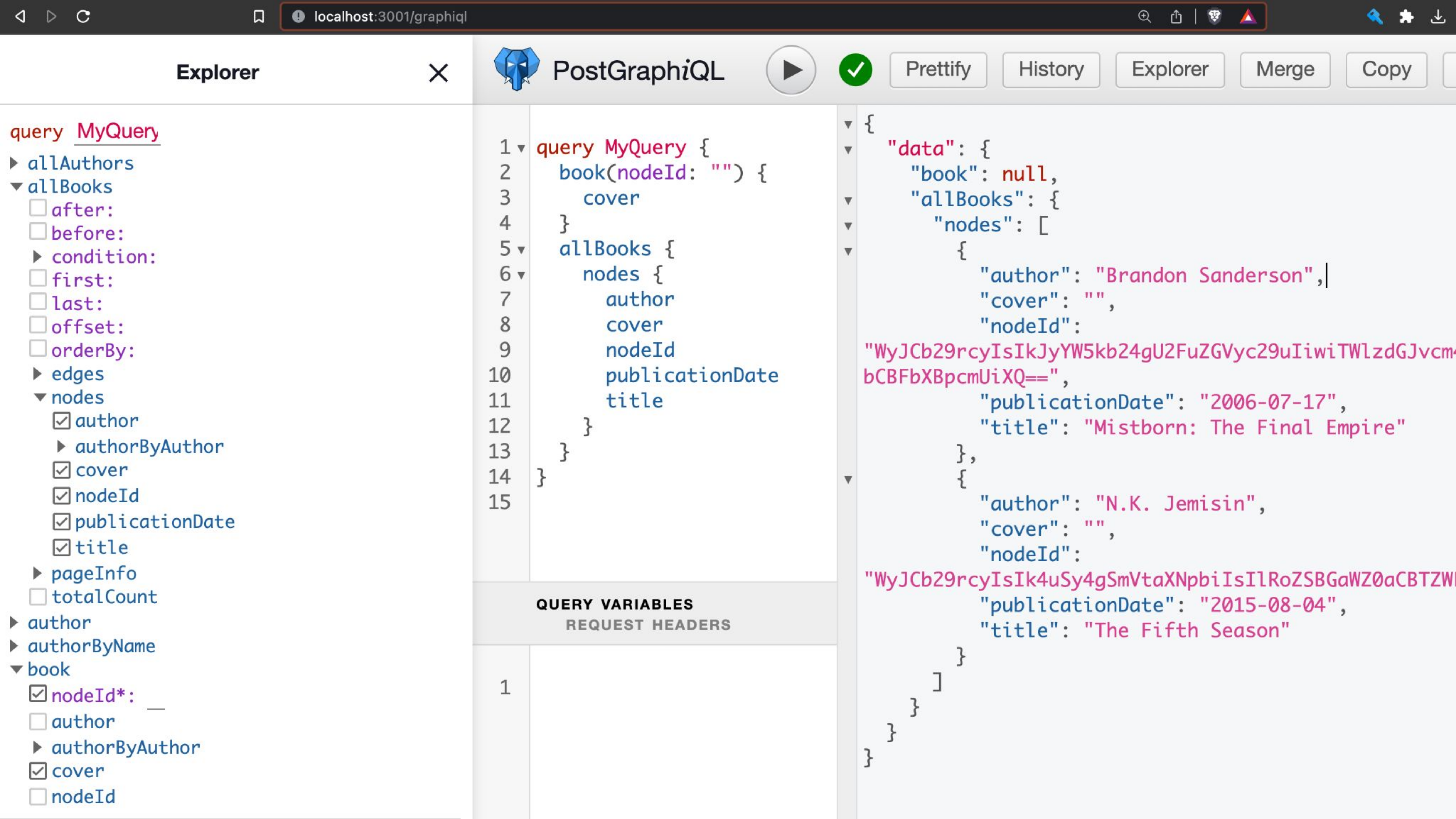
4. Self documented through introspection



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

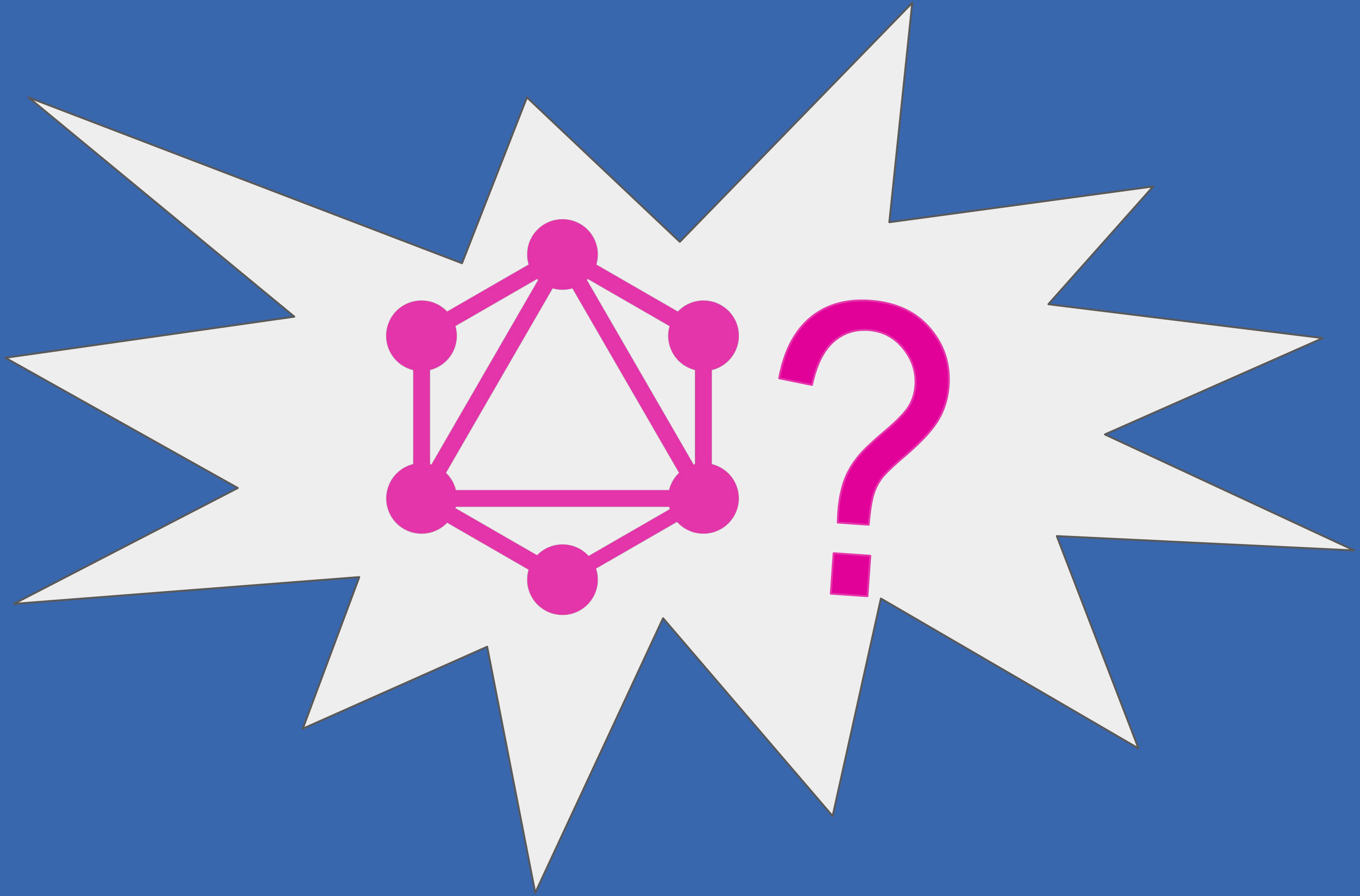


```
{
    "data": {
      "__type": {
        "name": "Book",
        "fields": [
          {
            "name": "nodeId",
            "type": {
              "name": null,
              "kind": "NON_NULL"
            }
          },
          {
            "name": "author",
            "type": {
              "name": null,
              "kind": "NON_NULL"
            }
          },
          {
            "name": "title",
            "type": {
              "name": null,
              "kind": "NON_NULL"
            }
          },
          ...
        ]
      }
    }
  }
```



```
query MyQuery {
  book(nodeId: "") {
    cover
  }
  allBooks {
    nodes {
      author
      cover
      nodeId
      publicationDate
      title
    }
  }
}
```

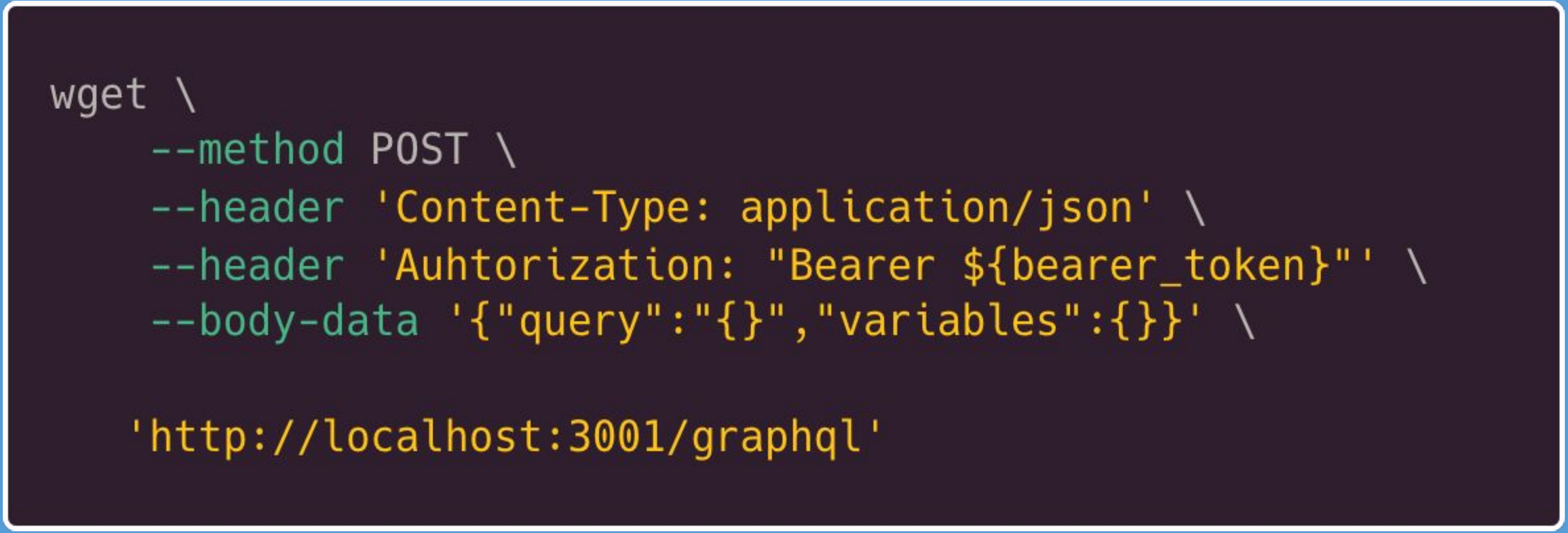
```
{
  "data": {
    "book": null,
    "allBooks": {
      "nodes": [
        {
          "author": "Brandon Sanderson",
          "cover": "",
          "nodeId":
"WyJCb29rcyIsIkJyYW5kb24gU2FuZGVyc29uIiwiTWlzdGJvcm
bCBFbXBpcmUiXQ==",
          "publicationDate": "2006-07-17",
          "title": "Mistborn: The Final Empire"
        },
        {
          "author": "N.K. Jemisin",
          "cover": "",
          "nodeId":
"WyJCb29rcyIsIk4uSy4gSmVtaXNpbiIsIlRoZSBGaWZ0aCBTZW
          "publicationDate": "2015-08-04",
          "title": "The Fifth Season"
        }
      ]
    }
  }
}
```



How does GraphQL Work?

It’s just a POST http request!

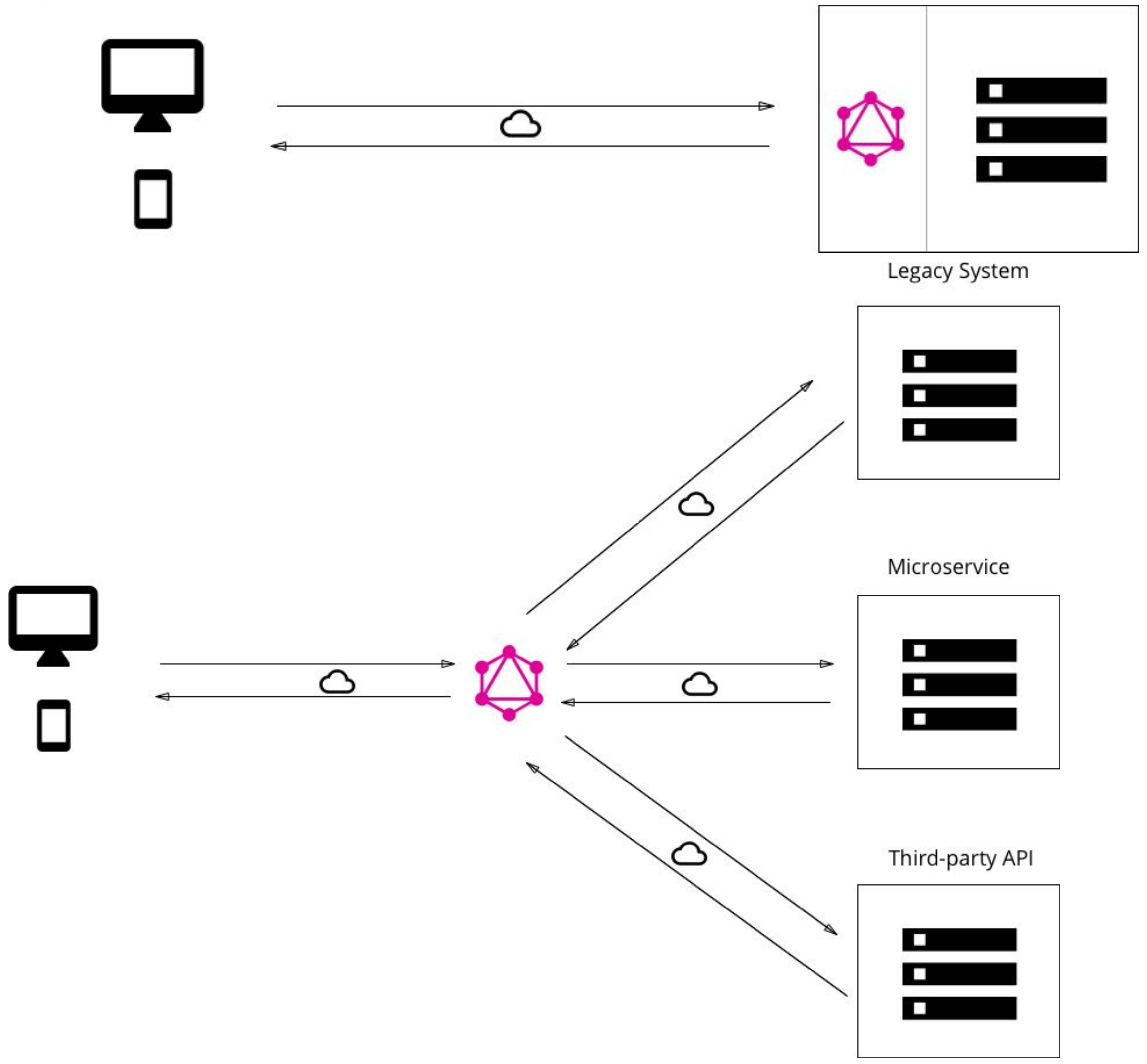
- Auth in Headers
- JSON Body is Operation (query, mutation, subscription)
- Variable support



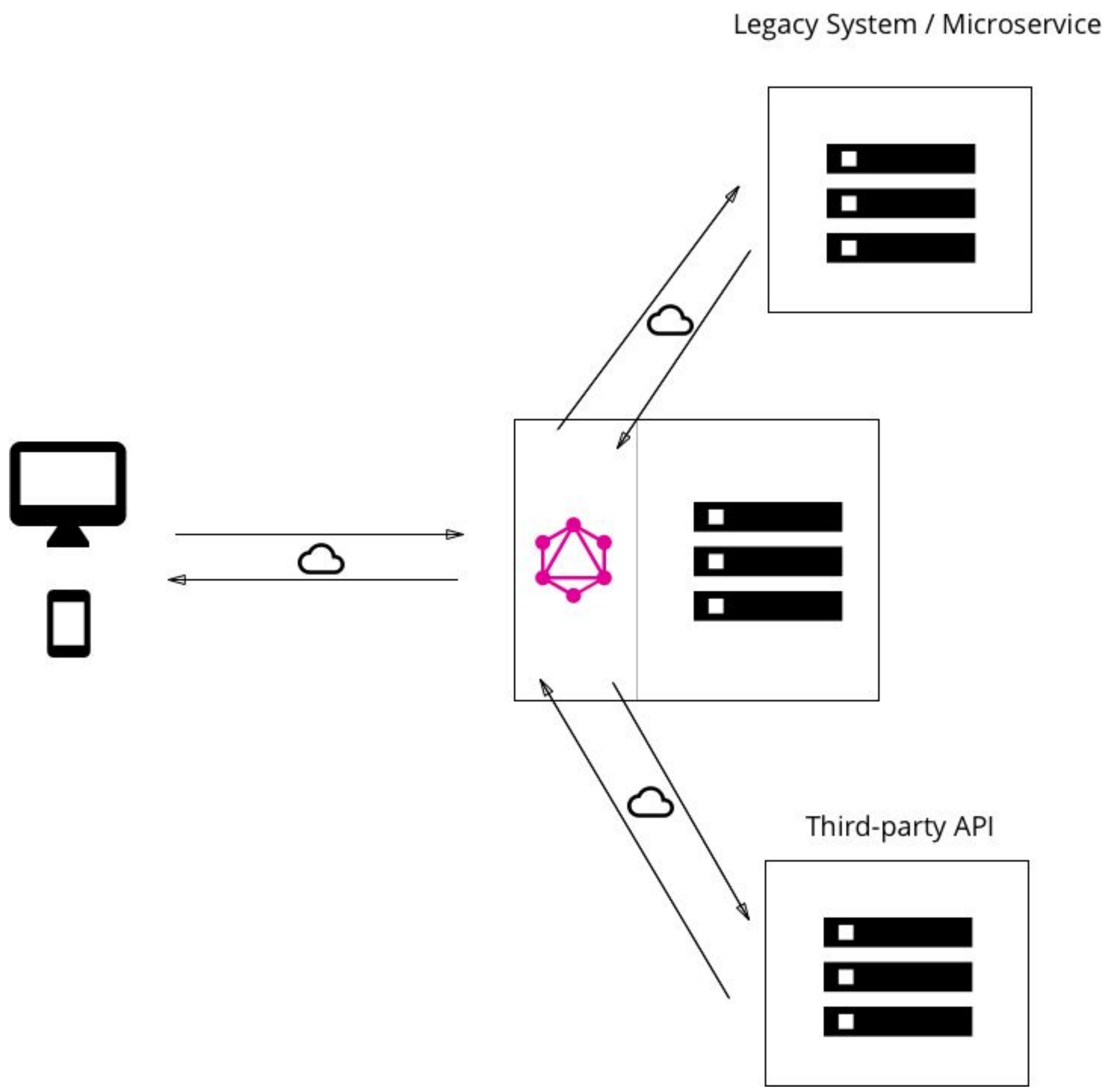
```
wget \
    --method POST \
    --header 'Content-Type: application/json' \
    --header 'Auhtorization: "Bearer ${bearer_token}"' \
    --body-data '{"query":"{}","variables":{}}' \

   'http://localhost:3001/graphql'
```

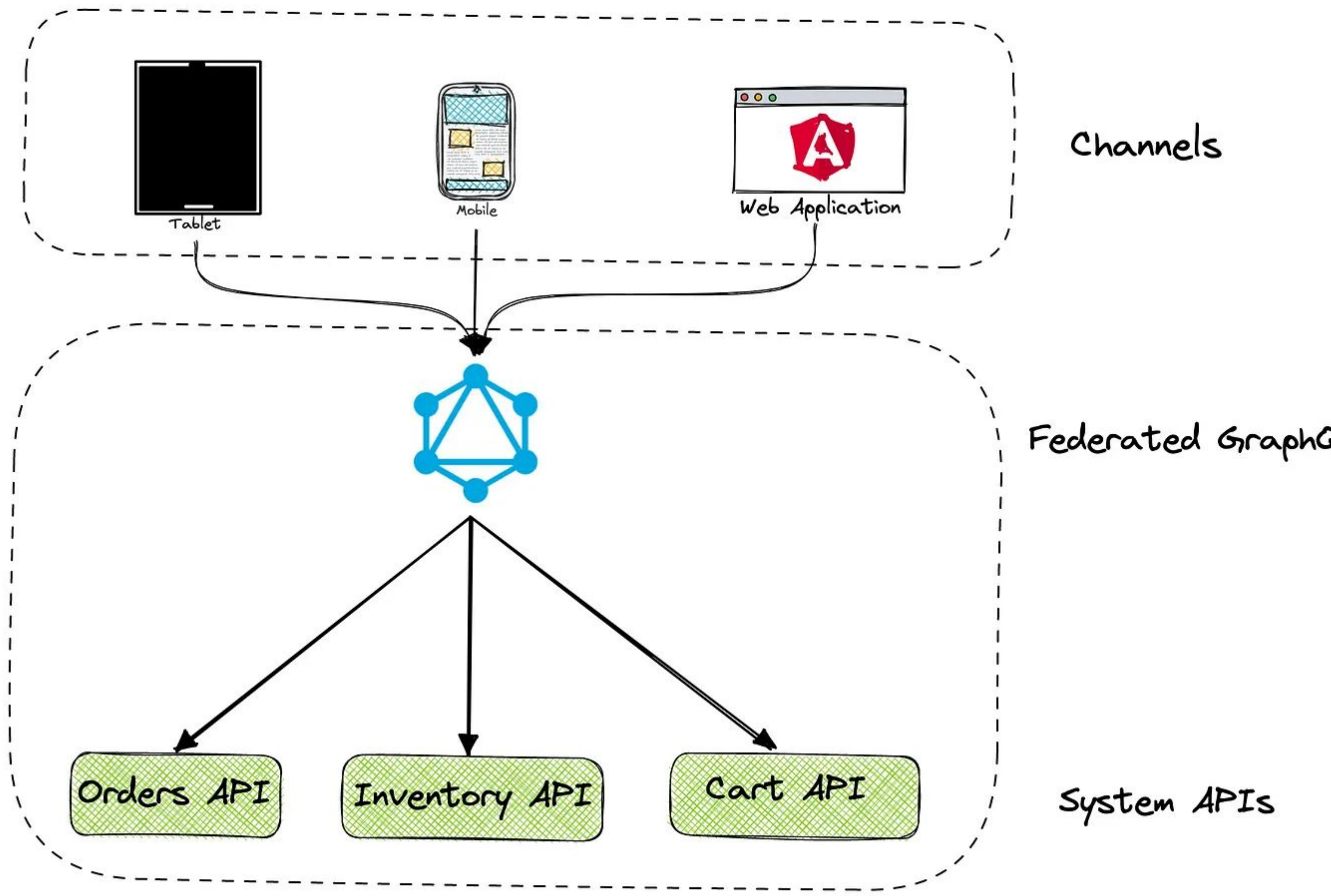
Architecture examples (1, 2)

1. Embedded alongside the db engine and data.
2. Standalone gateway
3. Hybrid of 1 and 2
4. Analog of 3.
5. BFF
6. Authentication is important

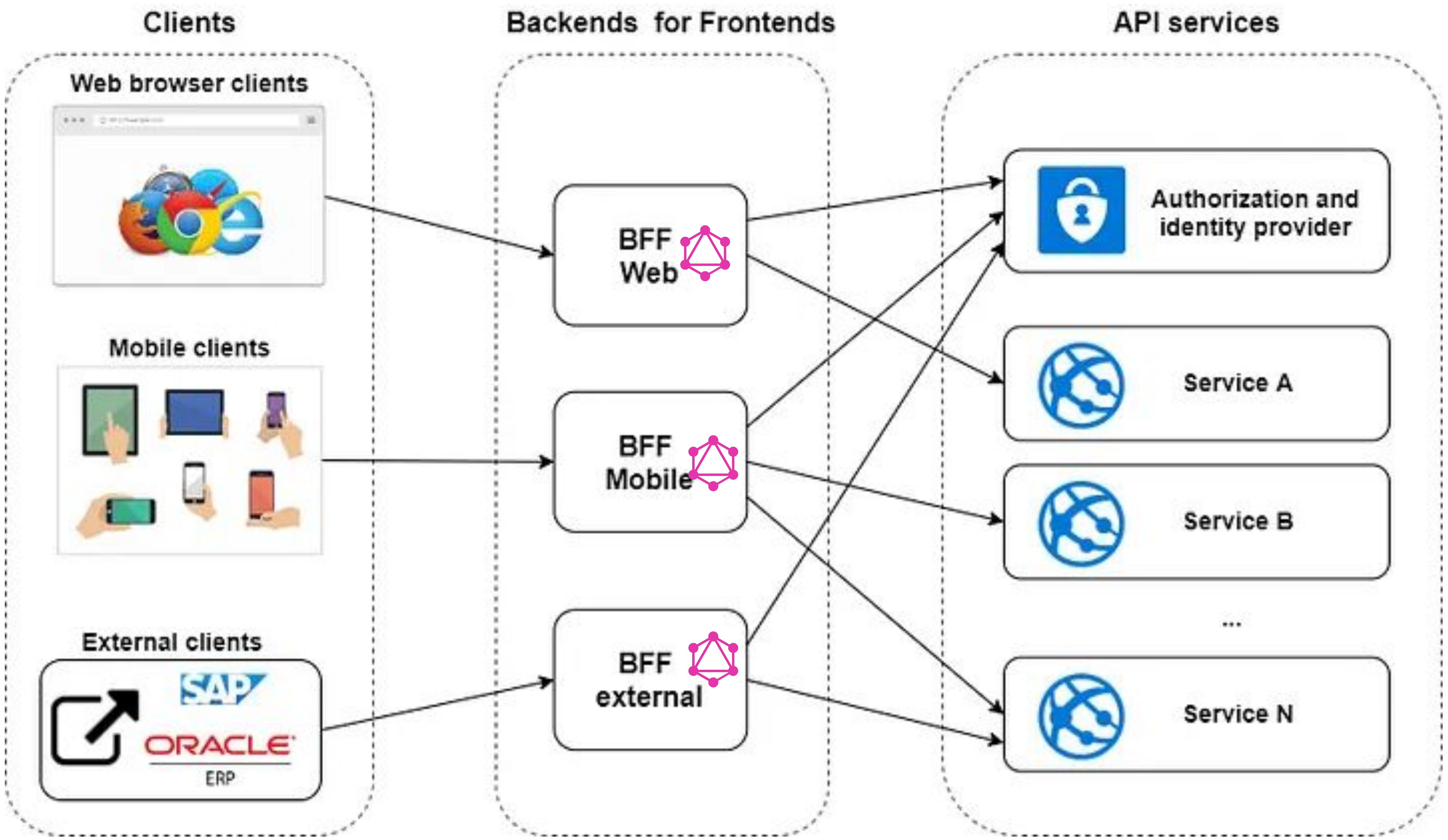


Architecture examples (3, 4)

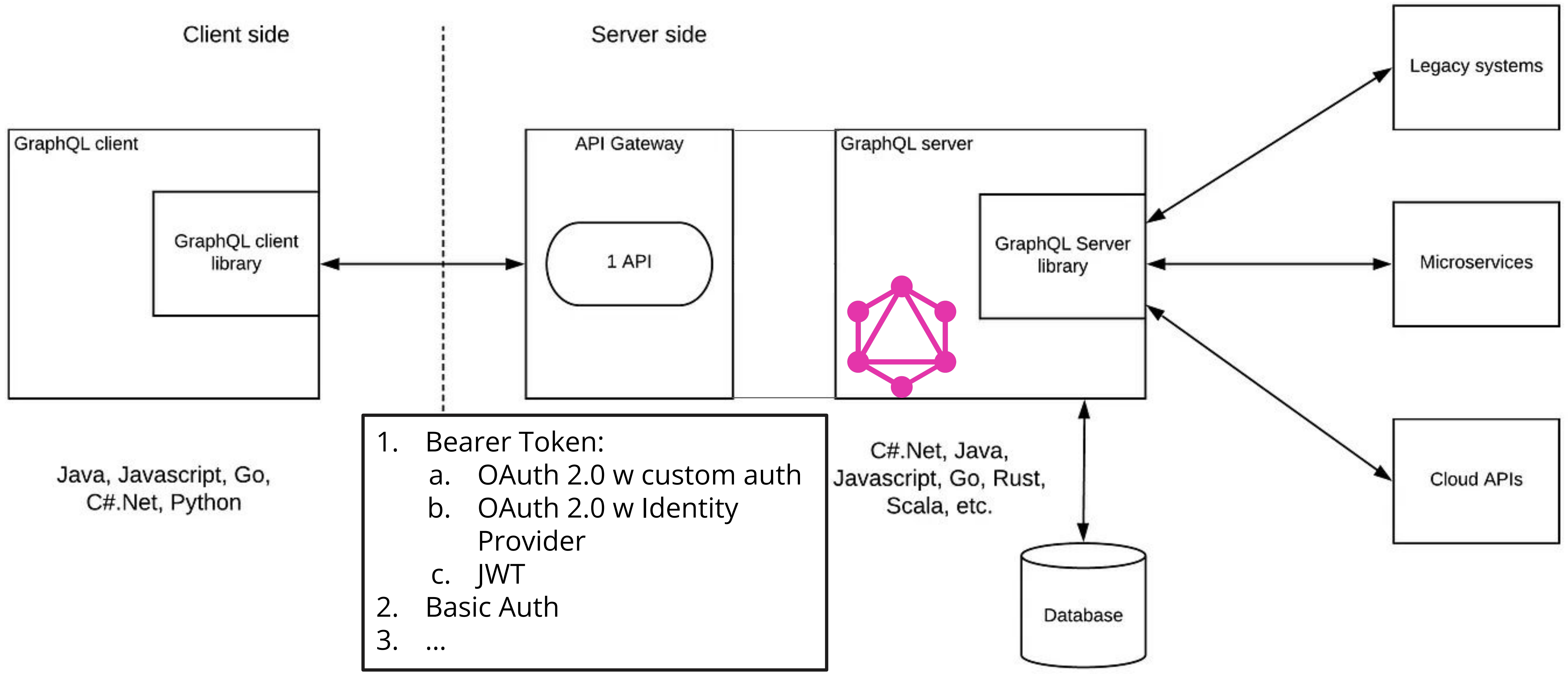




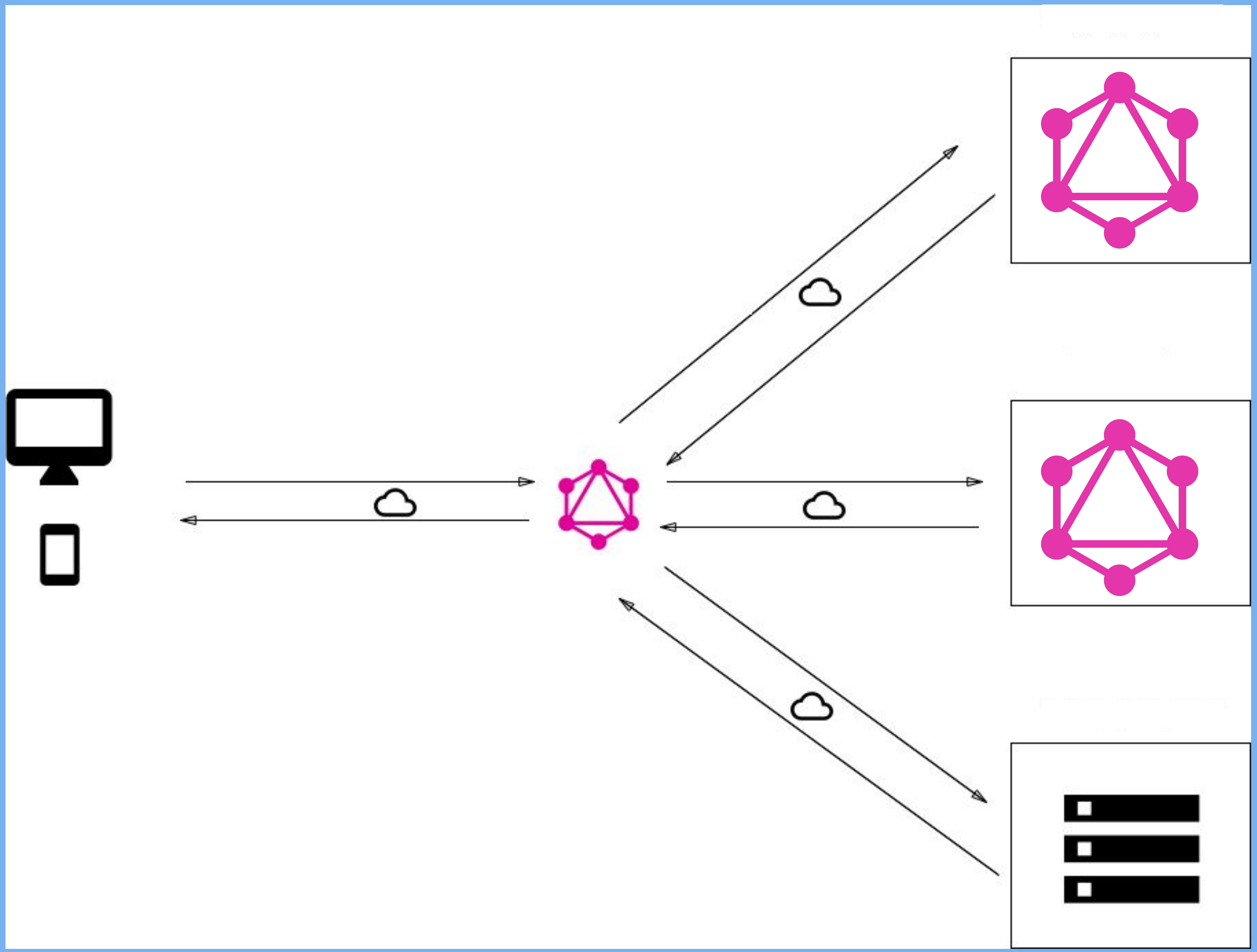
Architecture examples (5)



Architecture examples (6)



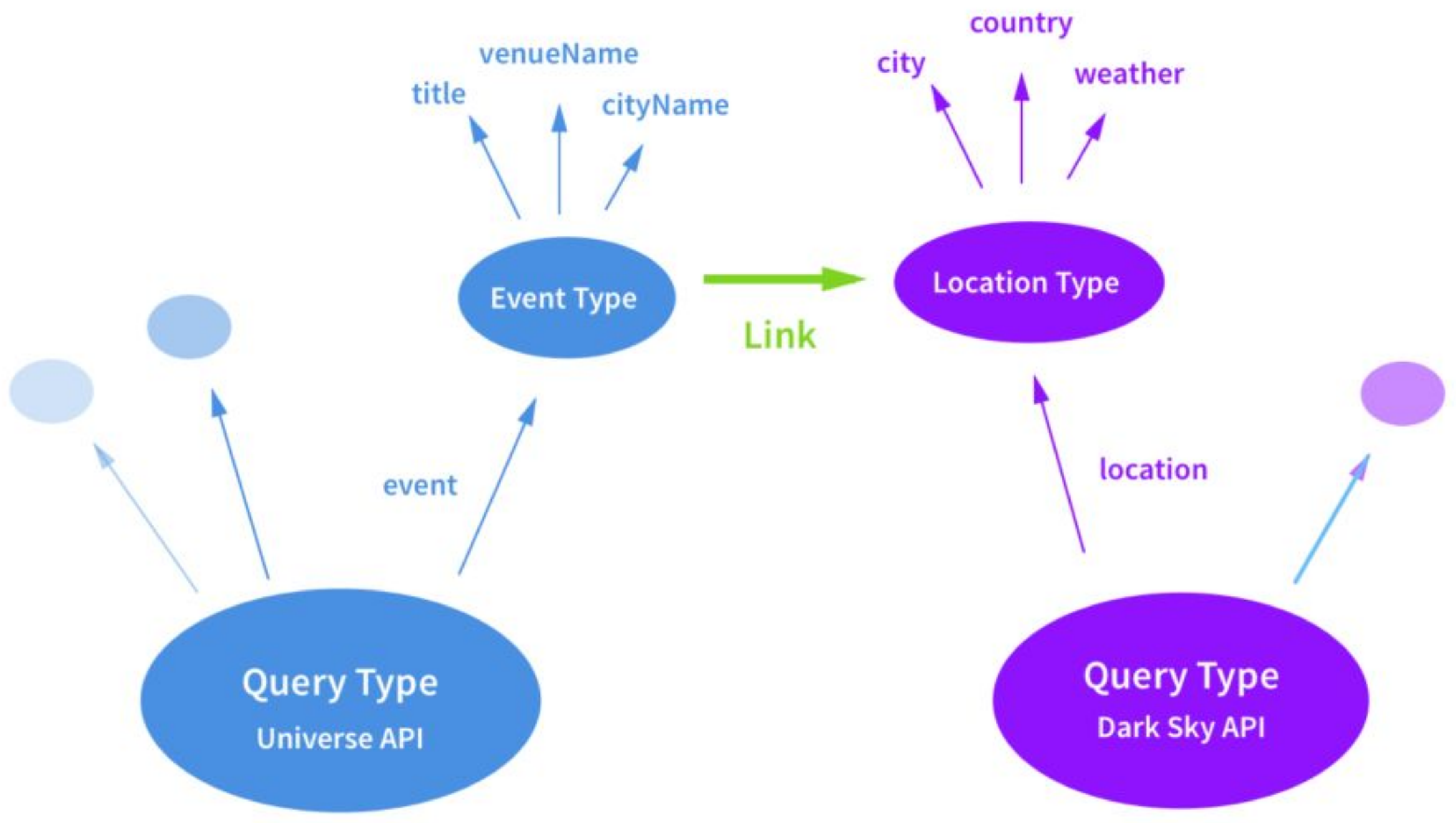
1. Bearer Token:
   a. OAuth 2.0 w custom auth
   b. OAuth 2.0 w Identity Provider
   c. JWT
2. Basic Auth
3. ...

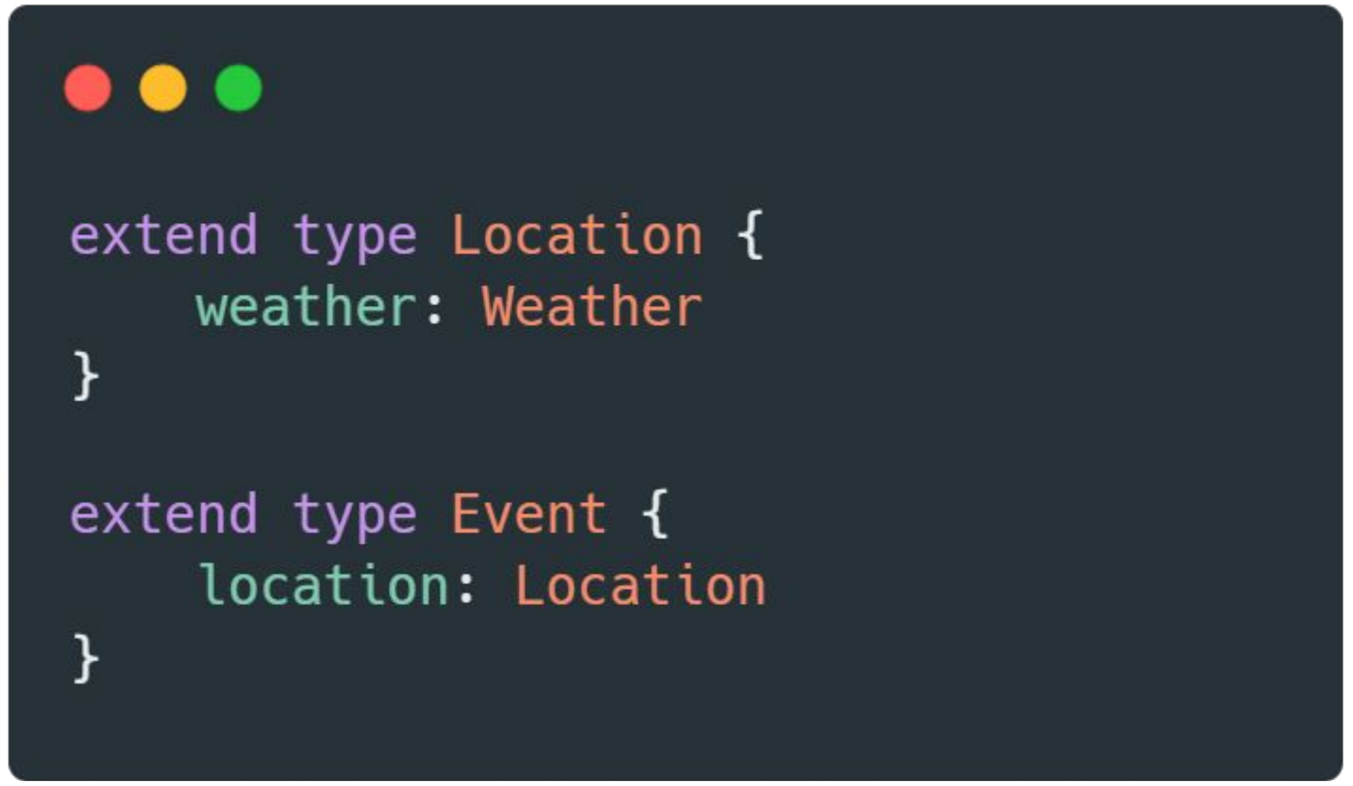


BONUS: Schema Stitching

Schema stitching (cont.)

- **Related models in different APIs can be linked in a server environment to save resources in client machines**
- **graphql-tools by The Guild**

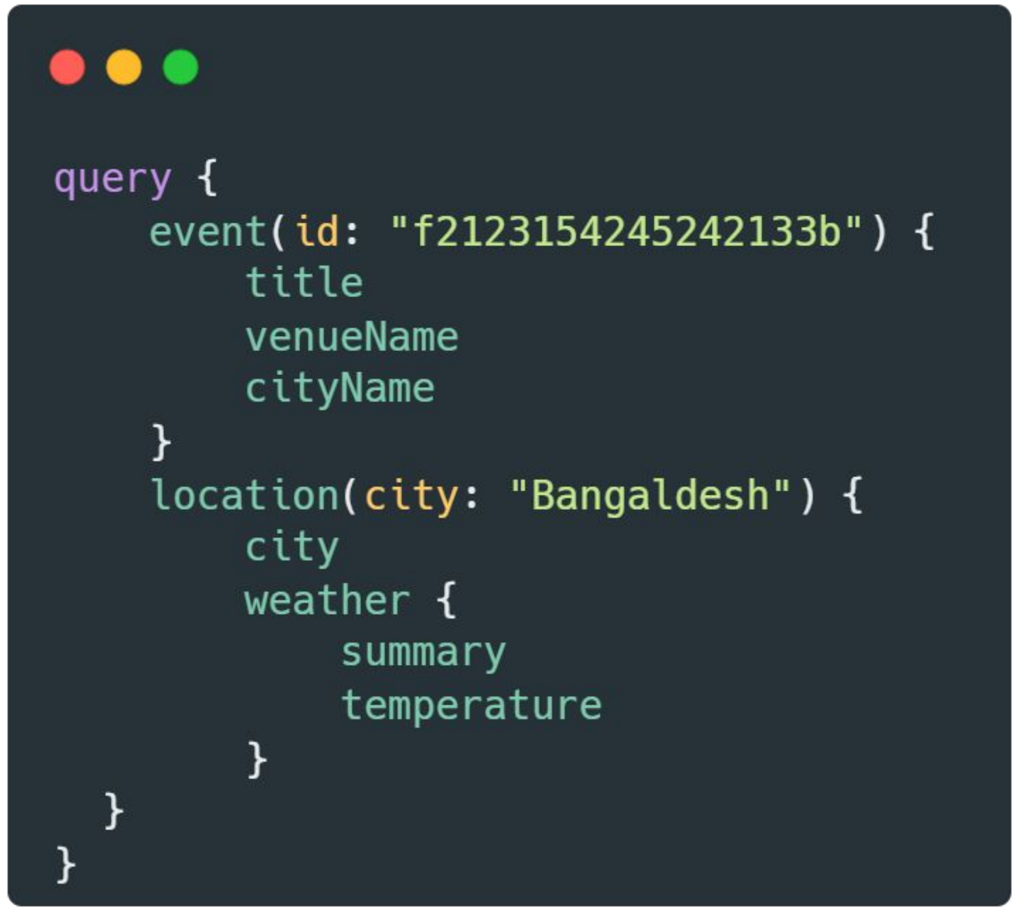




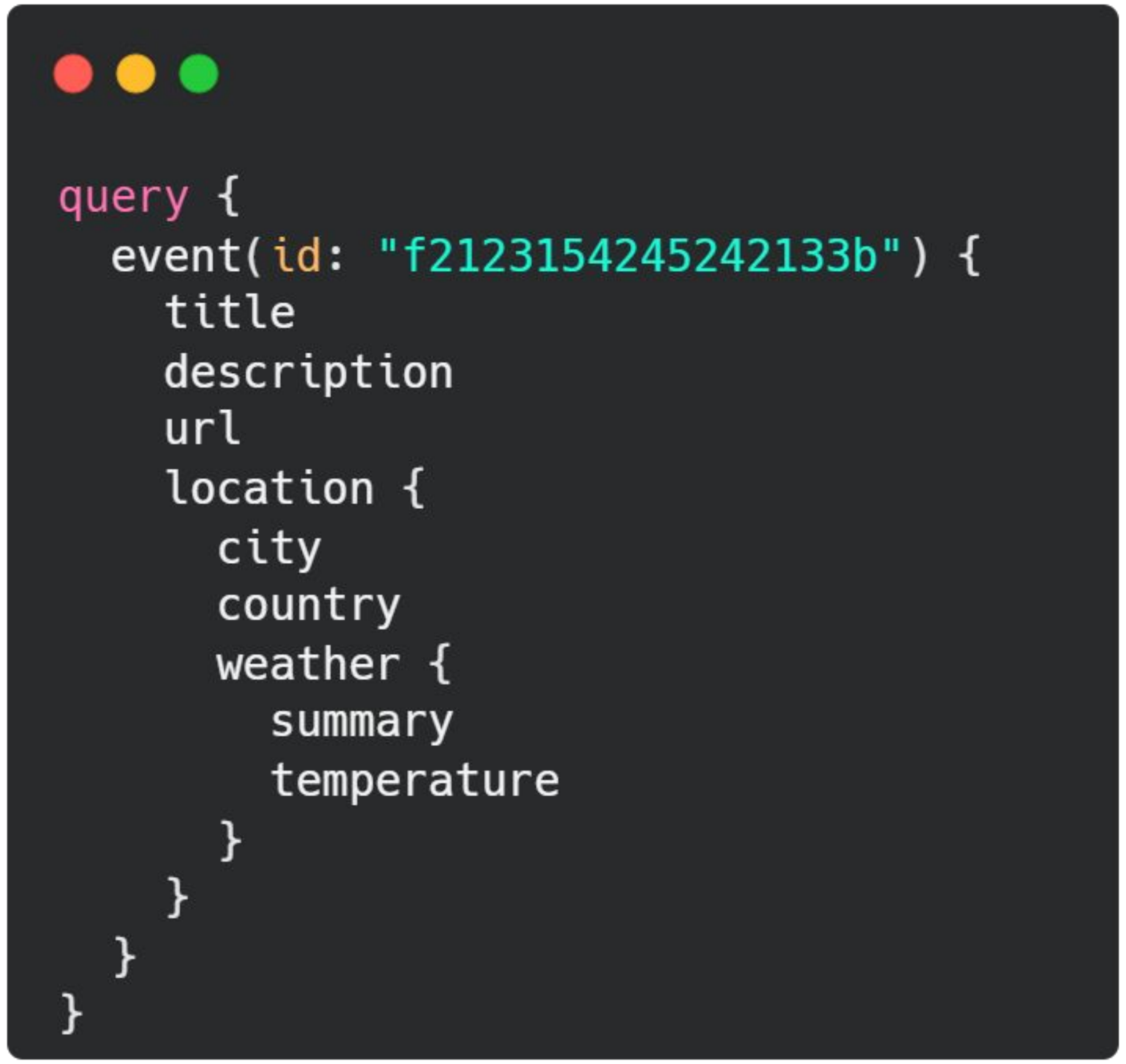
```
extend type Location {
    weather: Weather
}

extend type Event {
    location: Location
}
```

Schema stitching (cont.)

- Related models in different APIs can be linked in a server environment to save resources in client machines
- graphql-tools by The Guild



```
query {
    event(id: "f2123154245242133b") {
        title
        venueName
        cityName
    }
    location(city: "Bangaldesh") {
        city
        weather {
            summary
            temperature
        }
   }
}
```



```
query {
  event(id: "f2123154245242133b") {
    title
    description
    url
    location {
      city
      country
      weather {
        summary
        temperature
      }
    }
  }
}
```

Recap

- Catered queries are good
- Auth is Important

- GraphQL puts the client perspective first
- Not a Language, a Standard

- Abstract BE away at your own risk
- HTTPS FTW

the end.

Additional Resources:

- Research Notes
- Demo Repository