A GraphQL guide for backend developers

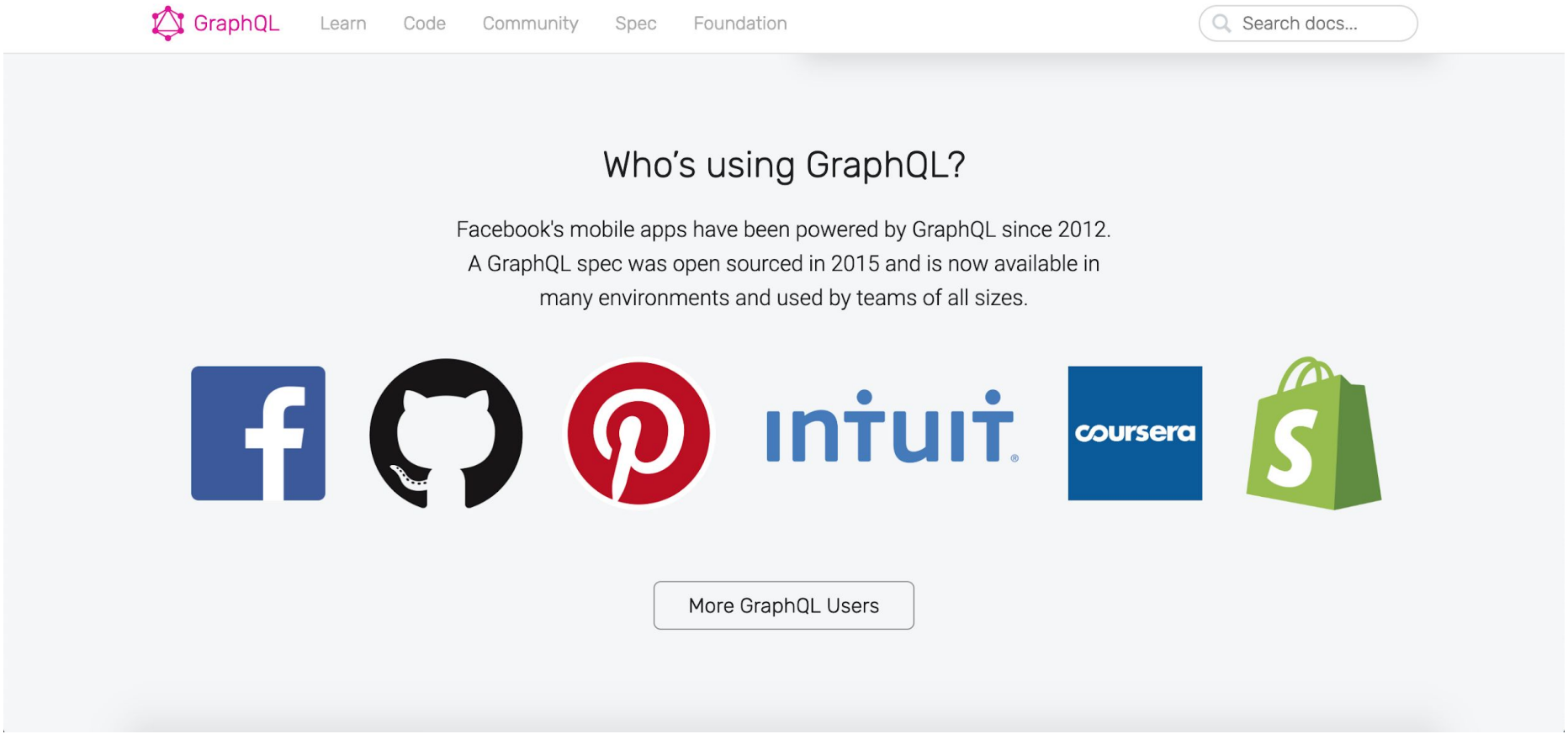

Backend * developer
GraphQL
this talk
• What’s GraphQL
• Benefits
• Best practices
• Challenges

Francesca Guiducci

Software Engineer
Pinterest, API team

@engfragui

GraphQL



Source: graphql.org/

“All the data you need, in one request.”

Source: graphql.com/

What is it

01 02 03 04

GraphQL is an alternative to REST

GraphQL is a query language for your API

Source: graphql.org/

GraphQL is a query language to get exactly the data you need

Source: myself

@eveporcello

GraphQL in action

Demo: snowtooth.moonhighway.com

Source: github.com/MoonHighway/snowtooth

```
query {
  liftCount (status:OPEN)
  allLifts {
    name
    status
    capacity
    elevationGain
  }
}
```


```
{
  "data": {
    "liftCount": 6,
    "allLifts": [
      {
        "name": "Astra Express",
        "status": "OPEN",
        "capacity": 3,
        "elevationGain": 899
      },
      {
        "name": "Jazz Cat",
        "status": "OPEN",
        "capacity": 2,
        "elevationGain": 1230
      },
      {
        "name": "Jolly Roger",
        "status": "OPEN",
        "capacity": 6,
```

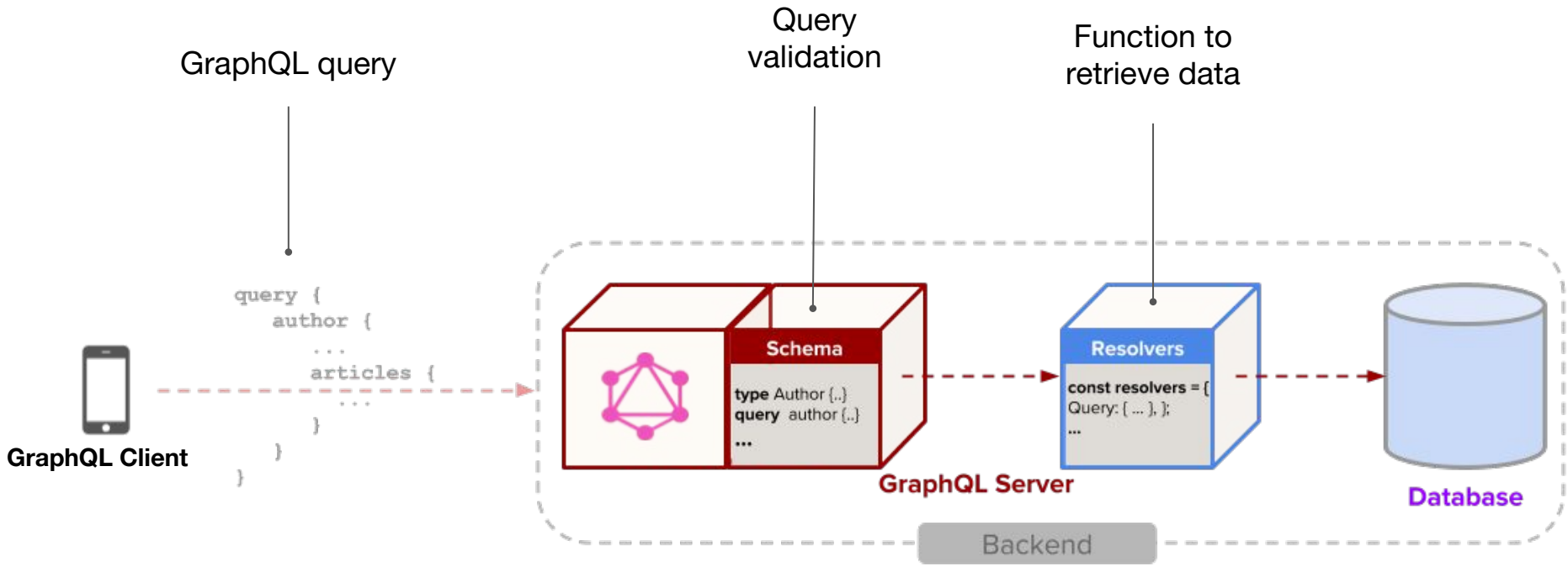
schema.graphql

```
type Lift {
    id: ID
    name: String
    status: LiftStatus
    capacity: Int
    night: Boolean
    elevationGain: Int
}

enum LiftStatus {
    OPEN
    CLOSED
    HOLD
}
```

```
type Query {
    allLifts(status: LiftStatus): [Lift]
    lift(id: ID): Lift
    liftCount(status: LiftStatus): Int
}
```

resolver.js

```
lift(obj, args, context, info) {
  return context.db.getLiftById(args.id).then(
    liftData => new Lift(liftData)
  )
}
```

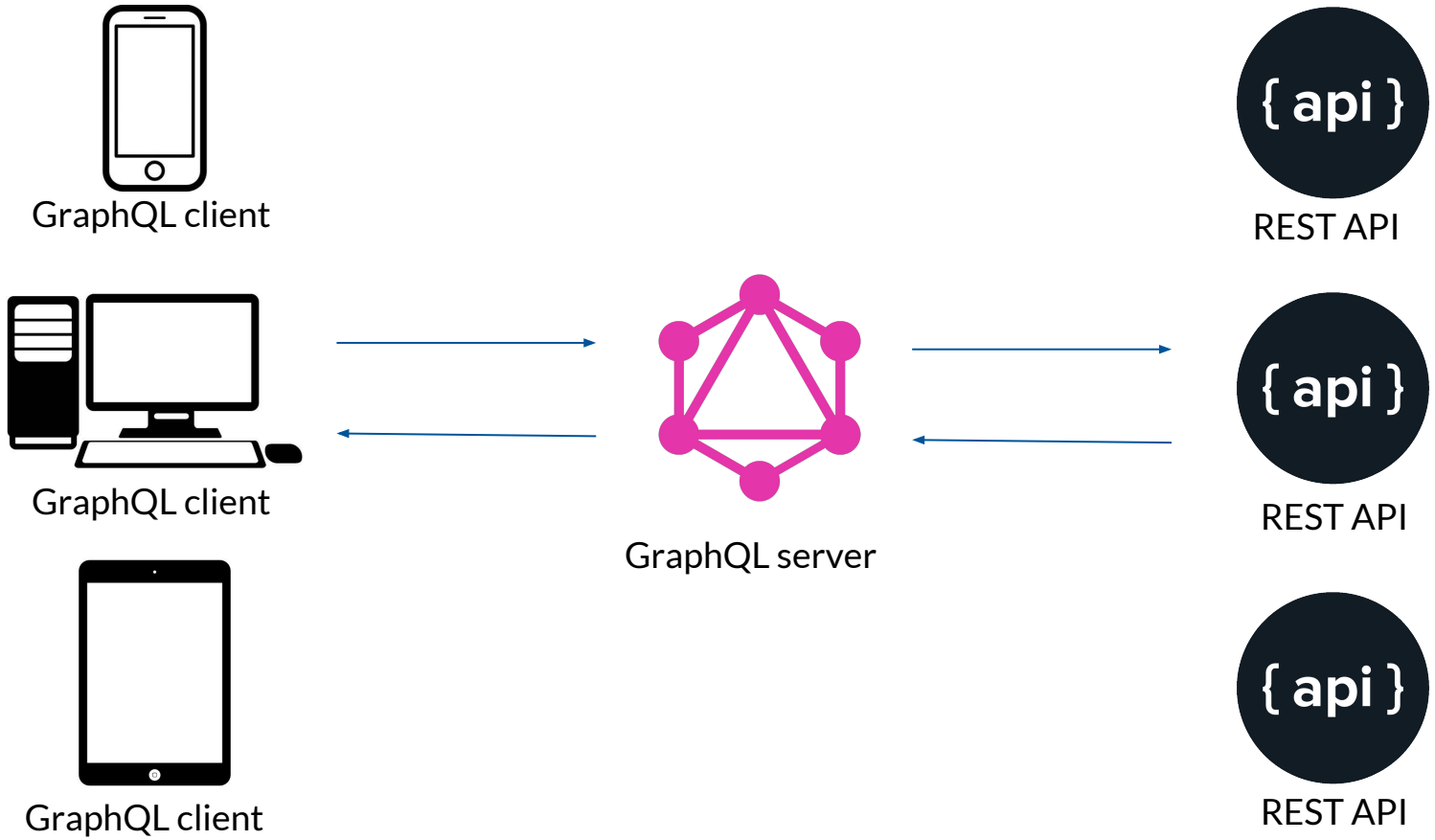

GraphQL query
Query validation
Function to retrieve data
query {
author {
...
articles {
...
}
}
}
GraphQL Client
Schema
type Author {..}
query author {..}
...
GraphQL Server
Resolvers
const resolvers = {
Query: { ... },
};
...
Database
Backend

Why I like it

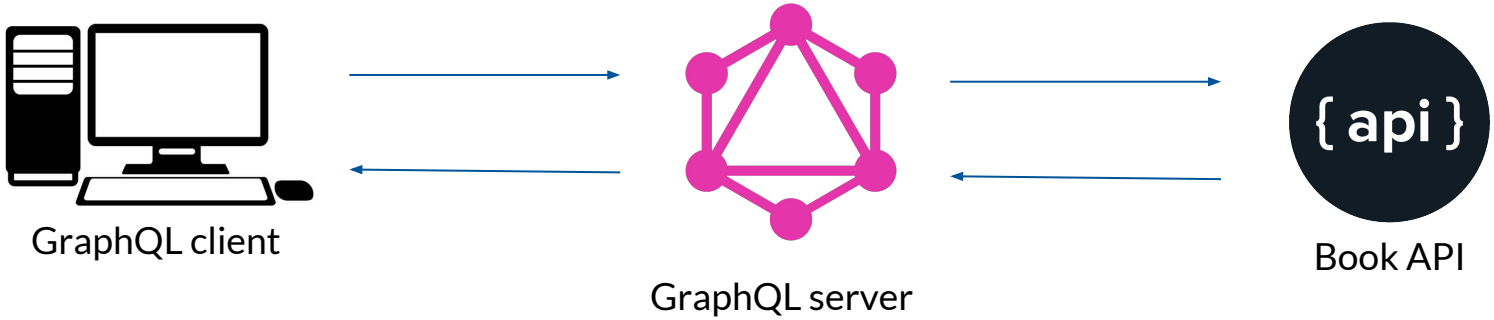
01 02 03 04

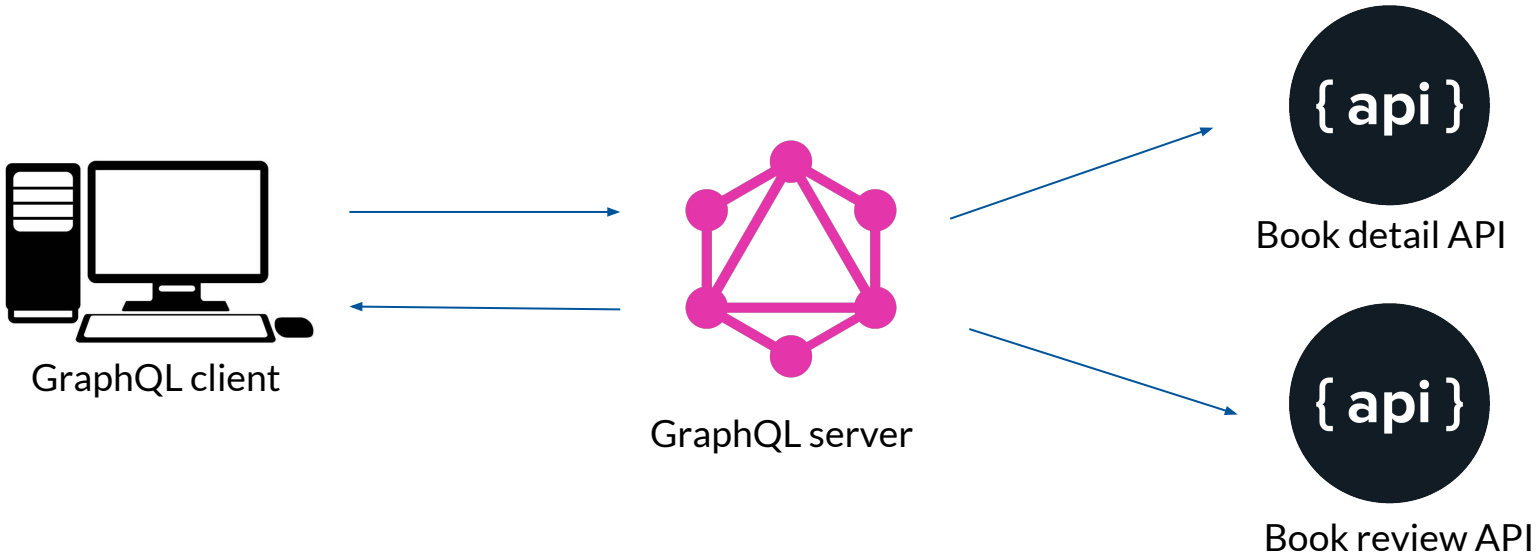
Can be integrated in your existing infra

01 02 03 04 05 06 07


GraphQL client
GraphQL client
GraphQL client
GraphQL server
{ api }
REST API
{ api }
REST API
{ api }
REST API

Decouples clients and servers


{ api }
GraphQL client
GraphQL server
Book API


{ api }
Book detail API
GraphQL client
GraphQL server
{ api }
Book review API

Encourages you to nail down your domain model


Impact
Full
Pow
Wow

No need for custom endpoints or parameters

01 02 03 **04** 05 06 07

/book/{id}

/book/{id}?fields={...}

/book/{id}/withpicture

/book/{id}?include=picture

Documentation comes for free and it’s “explorable”

01 02 03 04 **05** 06 07

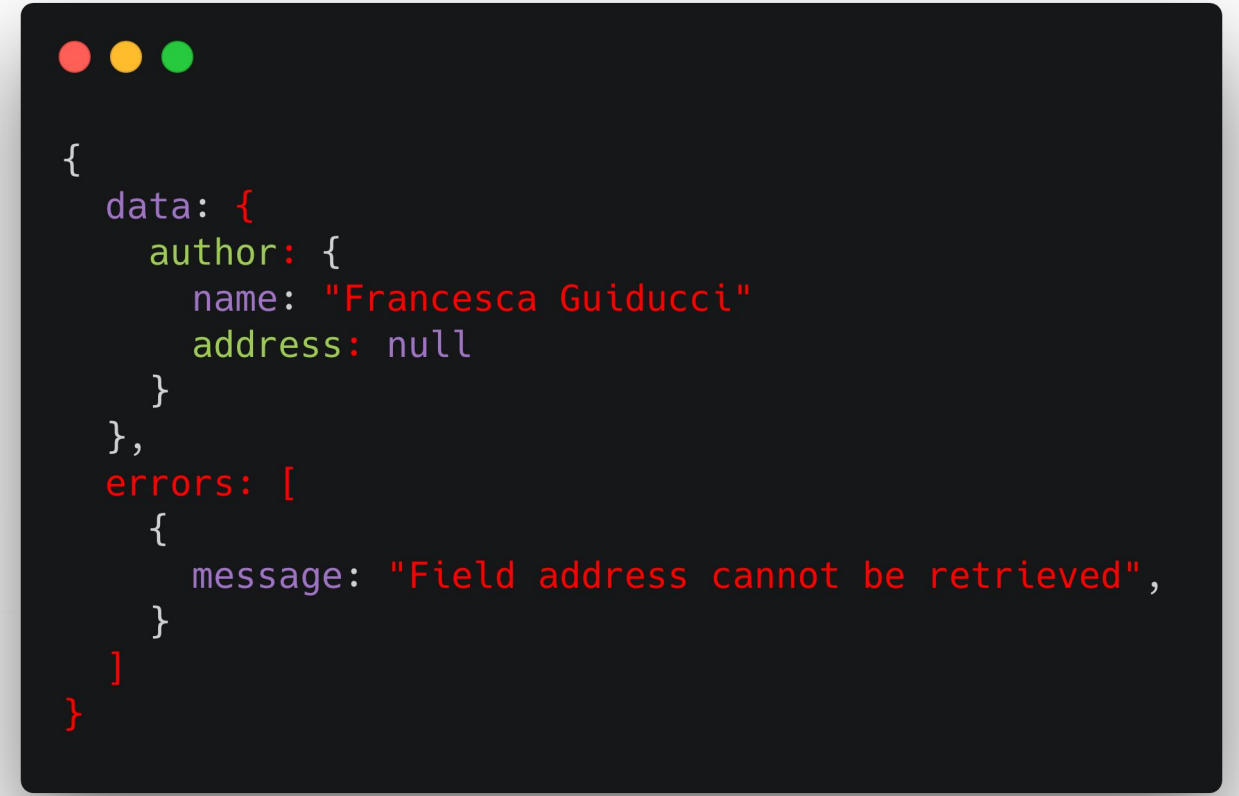
schema.graphql

```
"""
A `Lift` is a chairlift, gondola, tram, funicular,
pulley, rope tow, or other means of ascending a mountain.
"""
type Lift {

    """
    The unique identifier for a `Lift` (id: "panorama")
    """
    id: ID!

    """
    The name of a `Lift`
    """
    name: String!

    """
    The current status for a `Lift`: `OPEN`, `CLOSED`, `HOLD`
    """
    status: LiftStatus
}
```

Can return both data and errors

01 02 03 04 05 **06** 07

```
query {
  author {
    name
    address
  }
}
```



```
{
  data: {
    author: {
      name: "Francesca Guiducci"
      address: null
    }
  },
  errors: [
    {
      message: "Field address cannot be retrieved",
    }
  ]
}
```

Easy to mock the response

01 02 03 04 05 06 **07**


```
import { mockServer } from 'graphql-tools';

mockServer(schema, {
  Person: () => ({
    name: casual.name,
    age: () => casual.integer(0,120),
  })
});
```

Why I like it

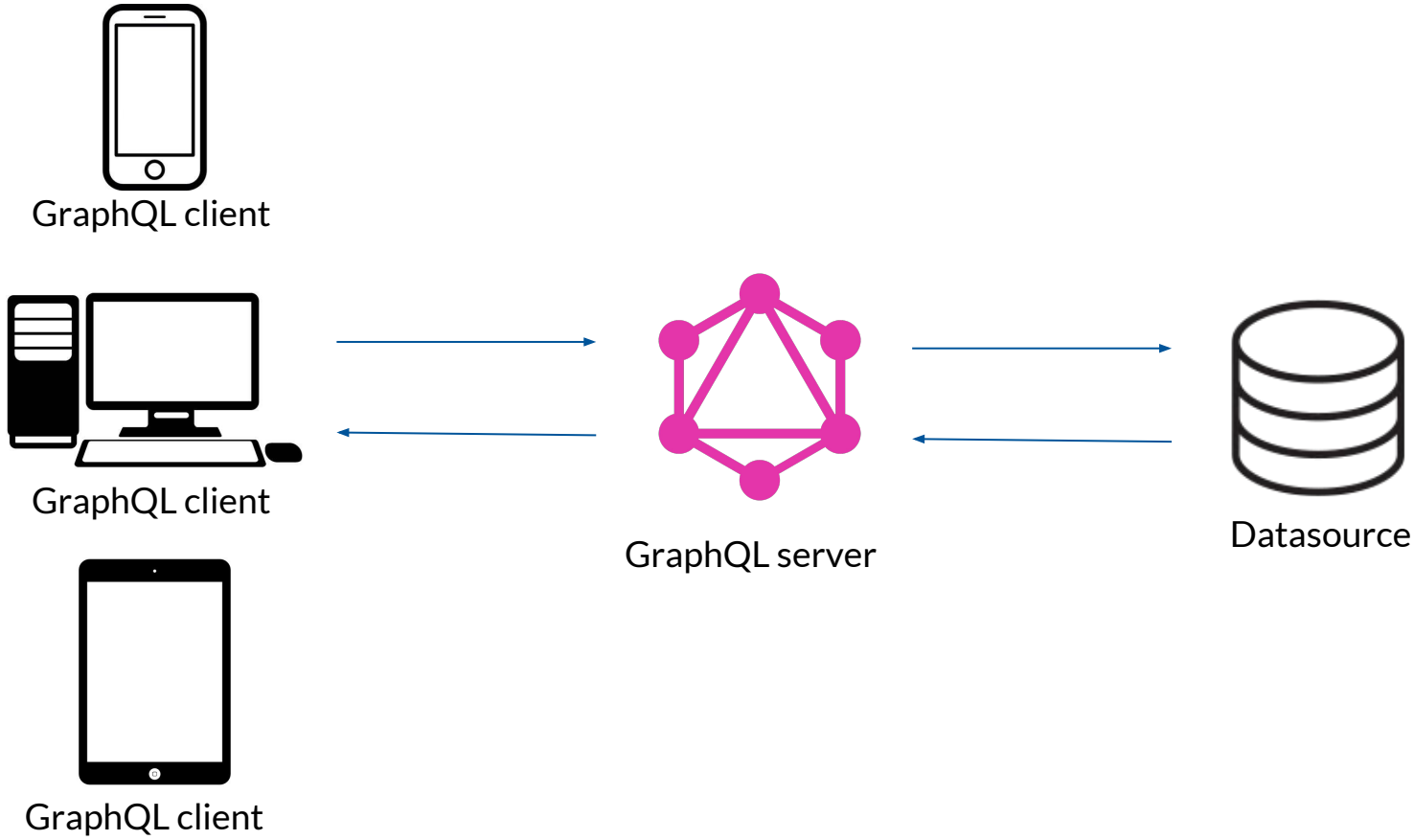
01 Can be integrated into your existing infra
02 Decouples clients and servers
03 Encourages you to nail down domain model
04 No need for custom endpoints or params
05 Free and “explorable” documentation
06 Can return both data and errors
07 Easy to mock the response

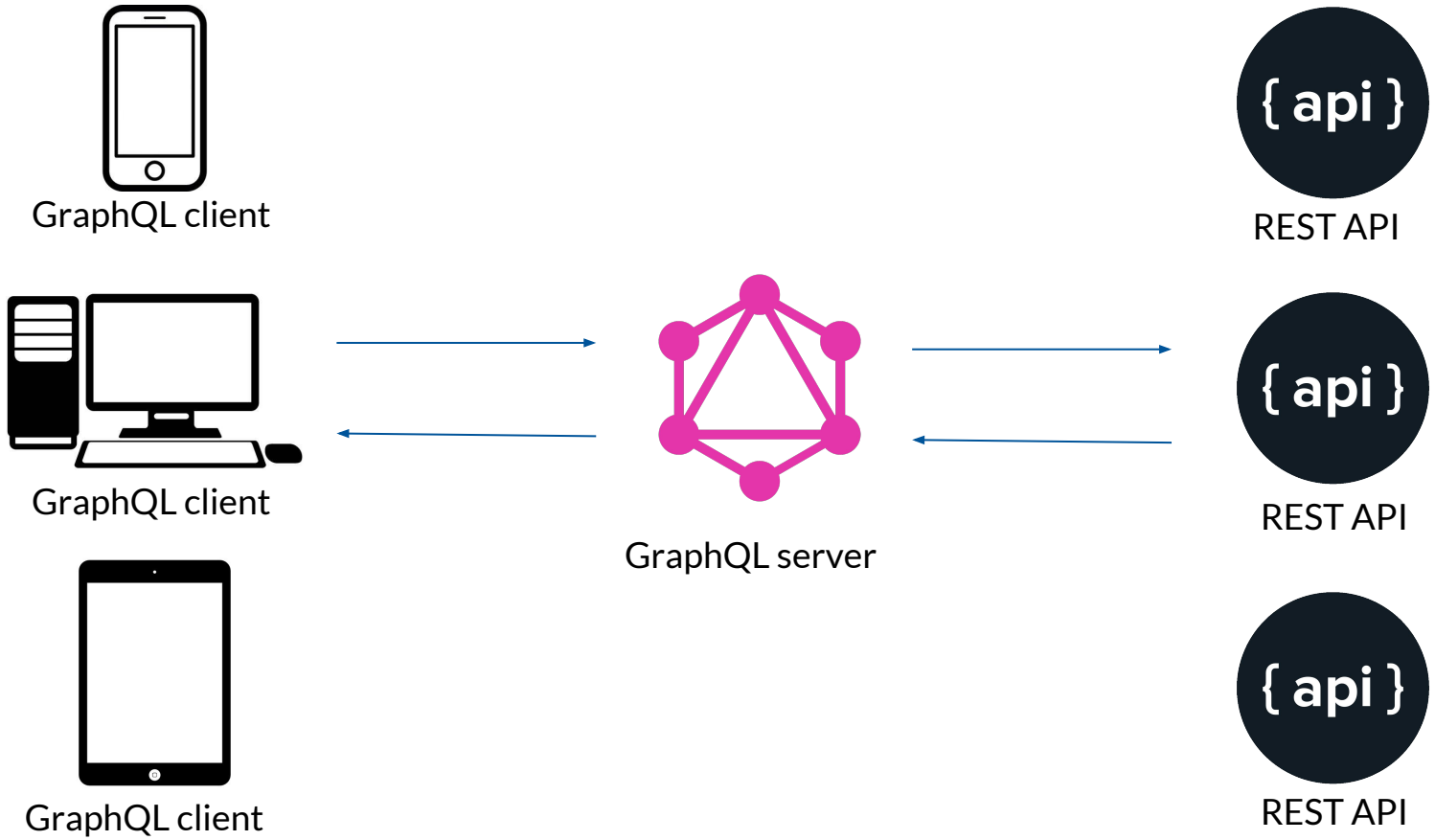
Advices

01 02 03 04

Figure out where GraphQL fits in your architecture

01 02 03 04 05 06


GraphQL client
GraphQL client
GraphQL client
GraphQL server
Datasource


GraphQL client
GraphQL client
GraphQL client
GraphQL server
{ api }
REST API
{ api }
REST API
{ api }
REST API

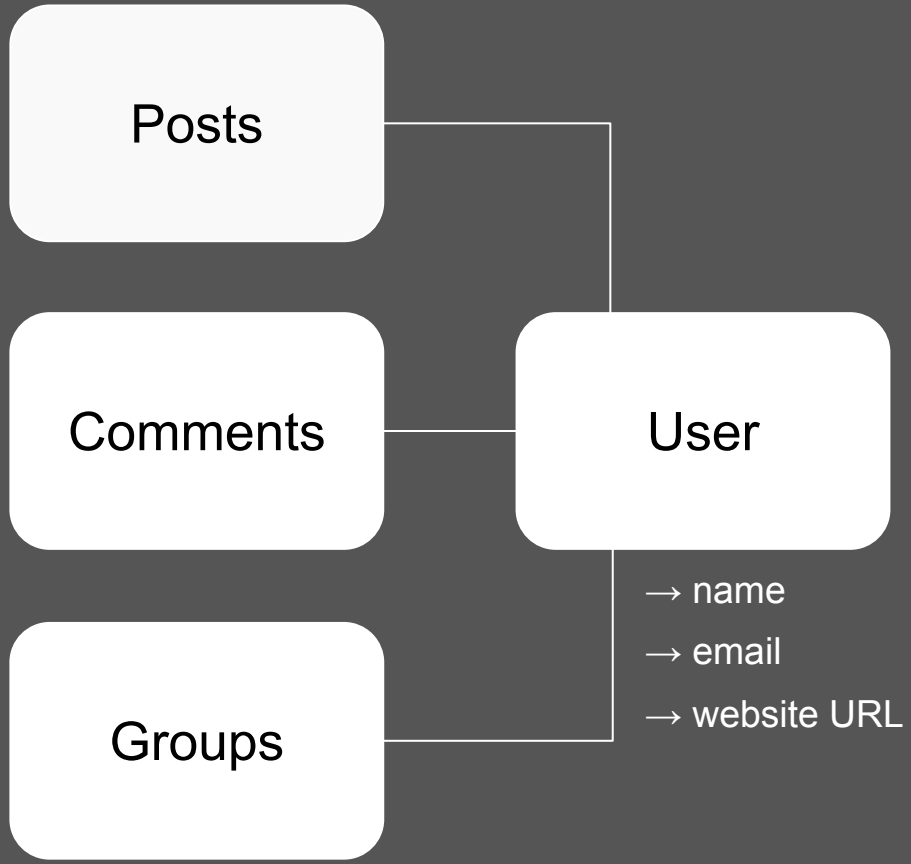
Involve others when designing the schema

01 **02** 03 04 05 06


Homepage
H1
H2
Download
H1-H6 Design
Video

Start thinking in graphs

01 02 **03** 04 05 06



schema.graphql

```graphql
type User {
  name: String
  email: GraphQLEmail
  websiteURL: GraphQLURL
  posts: [Post]
  comments: [Comment]
  groups: [Group]
}
type Post {
  id: ID!
  title: String
  author: User
}
```

Design a schema that is easy to evolve

01 02 03 **04** 05 06

schema.graphql

```
type Restaurant {
  name: String
  rating: Int
  address: String
}
```

alternative

schema-1.graphql

```
type Restaurant {
  name: String
  rating: Int
  location: Location
}
type Location {
  address: String
}
```

schema-1.graphql

```
type Restaurant {
  name: String
  rating: Int
  location: Location
}
type Location {
  address: String
  zipCode: Int
  country: String
  latLng: LatLng
}
type LatLng {
  lat: Float
  lng: Float
}
```

schema.graphql

```graphql
type Restaurant {
  name: String
  rating: Int
  address: String
}
```

evolve

schema.graphql

```graphql
type Restaurant {
  name: String
  rating: Int
  address: String @deprecated(reason: "Use `location.address`")
  location: Location
}
type Location {
  address: String
}
```

Be intentional when it comes to nullability

01 02 03 04 **05** 06

non-nullable

schema.graphql

```
type Lift {
    id: ID!
    name: String!
    status: LiftStatus!
}

type Query {
    lift(id: ID): Lift
}
```

nullable

no lift found

```
{ lift: null }
```

all data present and not null

```
{
  lift: {
    id: "astra-express",
    name: "Astra Express",
    status: "OPEN"
  }
}
```

Implement pagination if it might be useful

```
friends(first:2, offset:4)

friends(first:2, after:$friendId)

friends(first:2, after:$cursor)
```

Advices

01 Figure out where GraphQL fits
02 Involve others when designing the schema
03 Start thinking in graphs
04 Design a schema that is easy to evolve
05 Be intentional when it comes to nullability
06 Implement pagination if it might be useful

Challenges

01 02 03 04

Challenges

01 Versioning is frowned upon
02 No standard structure for pagination
03 Performance hit if naive implementation
04 Possible malicious queries
05 Testing needs to cover big surface area
06 Cannot rely on HTTP error codes


ТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

GraphQL guide

01 What is GraphQL
02 Why I like it
03 Advices
04 Challenges

Twitter shoutouts

Who's talking about GraphQL?

twitter.com/eveporcello
twitter.com/ xuorig
twitter.com/peggyrayzis
twitter.com/stubailo
twitter.com/jlengstorf
twitter.com/jnwng
twitter.com/leeb

Resources

For further reading/listening/watching

- GraphQL official documentation: graphql.org
- From REST to GraphQL: youtube.com/watch?v=eD7kLFGOgVw
- Everything You Need to Know About GraphQL in 3 Components: youtube.com/watch?v=F_M8v6MK0Sc
- Hard-Learned GraphQL Lessons: Based on a True Story: youtu.be/eUrtRzqN0h0
- Wrapping a REST API in GraphQL: graphql.org/blog/rest-api-graphql-wrapper/
- GraphQL best practices: graphql.org/learn/best-practices/
- Principled GraphQL: principledgraphql.com
- GraphQL at Trulia: medium.com/trulia-tech/graphql-one-endpoint-to-rule-them-all-4daaec273e5
- Evolvable GraphQL schemas: blog.apollographql.com/graphql-schema-design-building-evolvable-schemas-1501f3c59ed5
- Full stack tutorial: howtographql.com/
- Game of Thrones-themed tutorial: graphql-of-thrones.herokuapp.com

Thank you

Francesca Guiducci

Software Engineer
Pinterest, API team

@engfragui