Schema Design

SPOILER ALERT

GraphQL doesn’t magically great awesome APIs for you

Exercise

Fetching my favourite locations

```
{
  me {
    favouriteRestaurantIds
  }
}
```

Fetching my favourite locations

```
{
  me {
    favouriteRestaurants {
      id
    }
  }
}
```

Exercise

Dashboard with multiple stats:

```
{
  restaurant {
    id
    name
  }
}
```

- visitors e.g. 304
- revenue e.g. EUR 1567.30
- reservedTables e.g. 120

Exercise

```
location {
  id: ID!,
  name: string!,
  description: ???
}
```

```
description(lang: string!)
description(lang: string)

# example
description(lang: "en")
```

```
description(lang: Lang!) # enum

# example
description(lang: Lang.EN)
```

```
description {
  en
  nl
}

# example
description {
  en
  nl
}
```

```
description(lang: Lang!) # enum

# example
enDescription: description(lang: Lang.EN)
deDescription: description(lang: Lang.DE)
```

Exercise

Relay style connections

Example: SWAPI

https://facebook.github.io/relay/graphql/connections.htm

Exercise

Error Handing for Mutations

Using the build in GraphQL errors for forms has couple of drawbacks.

- input errors and general errors are mixed
- can't establish mapping form errors to specific fields without establishing conventions

Exercise

GraphQL Schema Design: Building Evolvable Schemas

https://blog.apollographql.com/graphql-schema-design-building-evolvable-schemas-1501f3c59ed5

Tutorial: Designing a GraphQL API (Shopify)

https://gist.github.com/swalkinshaw/3a33e2d292b60e68fcebe12b62bbb3e2

```
interface Collection {
  id: ID!
  memberships: [CollectionMembership!]!
  title: String!
  imageId: ID
  bodyHtml: String
}

type AutomaticCollection implements Collection {
  id: ID!
  rules: [AutomaticCollectionRule!]!
  rulesApplyDisjunctively: Bool!
  memberships: [CollectionMembership!]!
  title: String!
  imageId: ID
  bodyHtml: String
}

type ManualCollection implements Collection {
  id: ID!
  memberships: [CollectionMembership!]!
  title: String!
  imageId: ID
  bodyHtml: String
}

type AutomaticCollectionRule {
  column: String!
  relation: String!
  condition: String!
}

type CollectionMembership {
  collectionId: ID!
  productId: ID!
}
```

The End