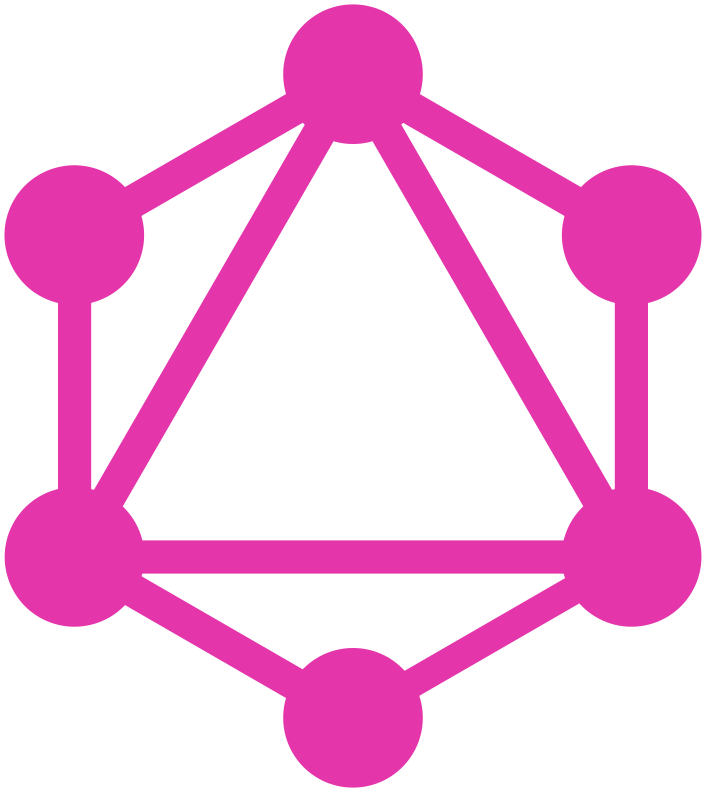


GraphQL:

- Просто
- Четко
- Оптимально

Что такое GraphQL

GraphQ – это Язык запросов для API и среда выполнения для выполнения запросов к вашим данным. GraphQL обеспечивает полное и понятное описание данных в вашем API, предоставляет клиенту возможность запрашивать в точности то, что ему нужно и ничего лишнего, а также предоставляет поддержку для продвинутых средств разработки.

Что такое GraphQL

Запрашиваете что вам нужно – получаете в точности это.

Отправьте GraphQL запрос к вашему API и получите то, что запросили: не больше и не меньше… Приложения, использующие GraphQL быстры и стабильны, т.к. не сервер, а именно они управляют данными.

Получаете много ресурсов за один запрос

Запросы GraphQL получают доступ не только к свойствам одного ресурса, но и следуют по ссылкам между ними…. Приложения, использующие GraphQL могут быть быстрыми на медленных соединениях.

Описываете, что умеет делать ваша система

API GraphQL описываются в терминах типов и полей, а не точек доступа. Вы получаете все ваши данных из одной точки. GraphQL описывает типы, чтобы быть уверенным, что приложения запрашивают только то, что возможно и предоставляют ясные и понятные сообщения об ошибках.

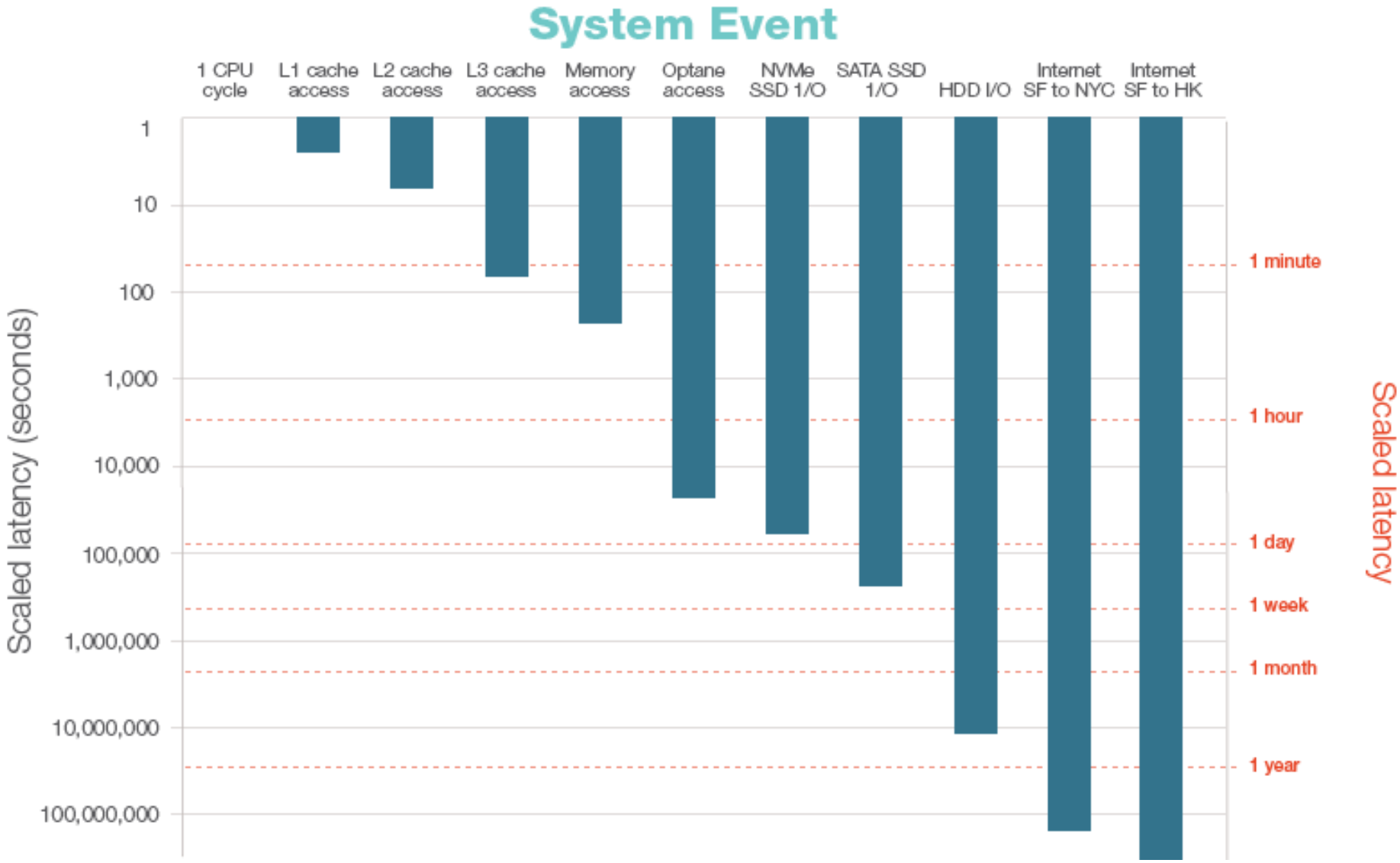
История

- 2012 — Начало разработки
- 2015 — Open Source
- 2017 — GraphQL Foundation
- ☺ *2018 – Запущен в production мобильный клиент CRM одного из крупнейших Москвских агентств элитной недвижимости.*

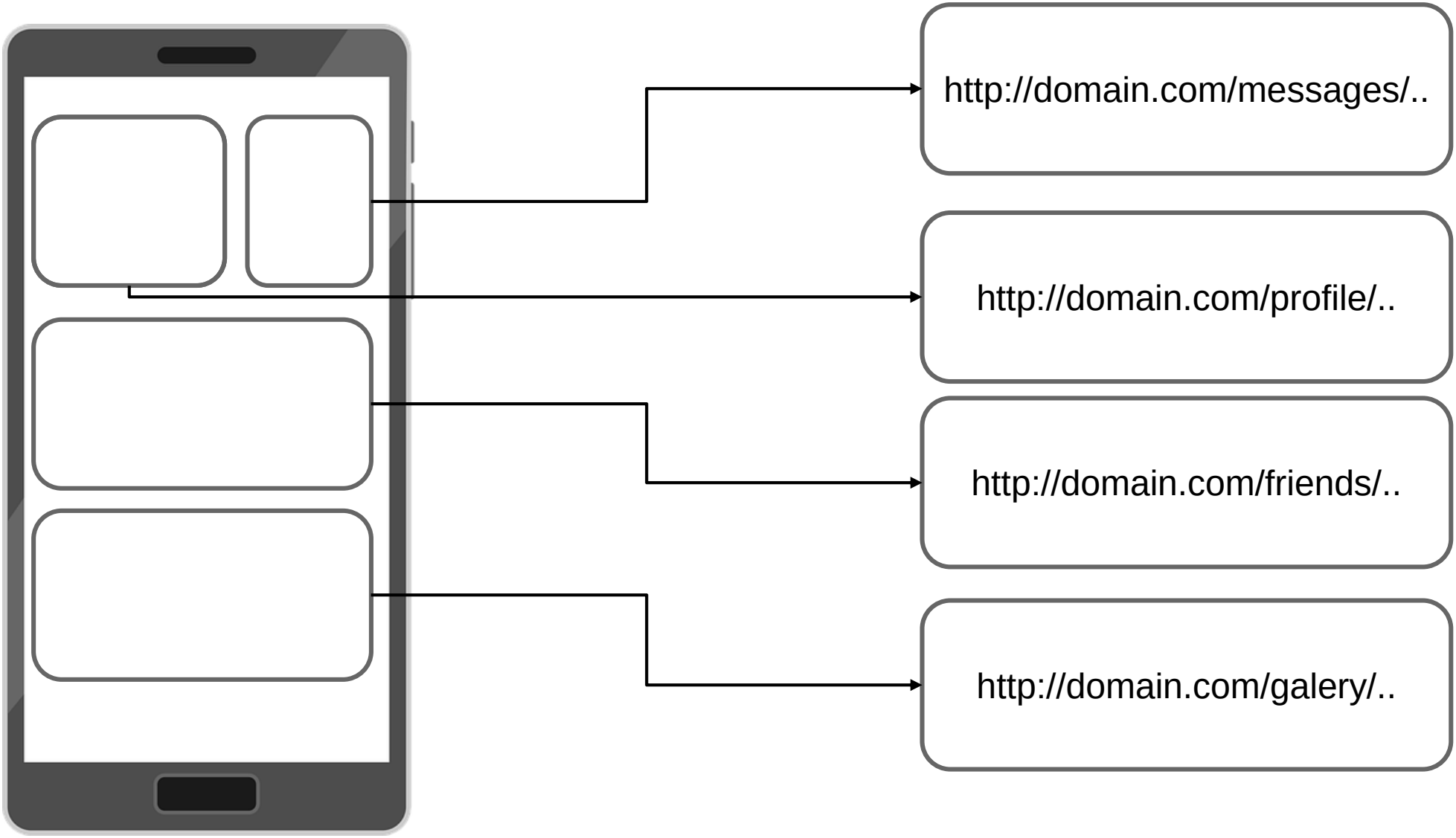
Предпосылки

- Усложнение приложений
- Увеличение производительности мобильных устройств
- Латентность сетевого подключения

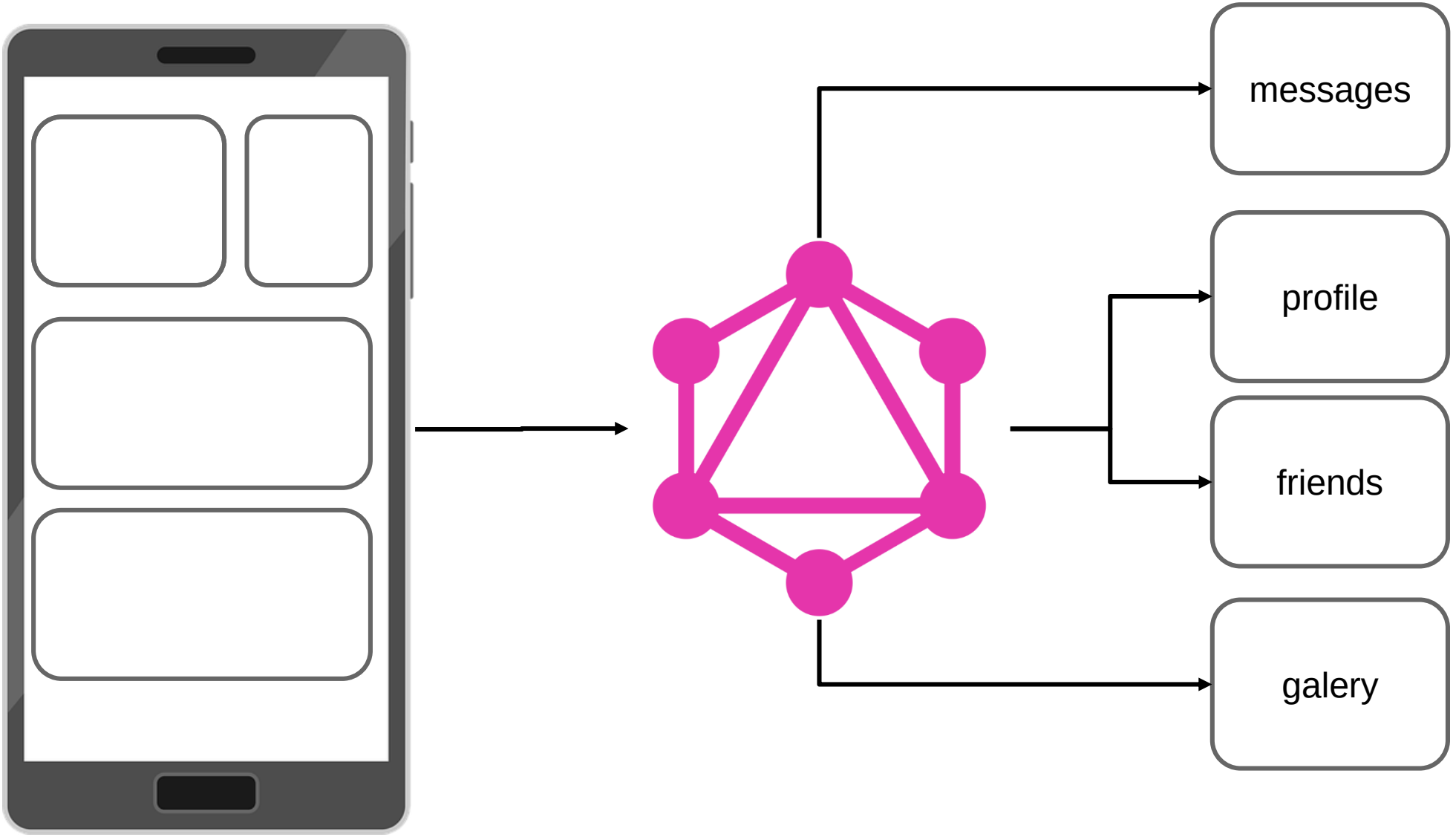
В наглядном масштабе



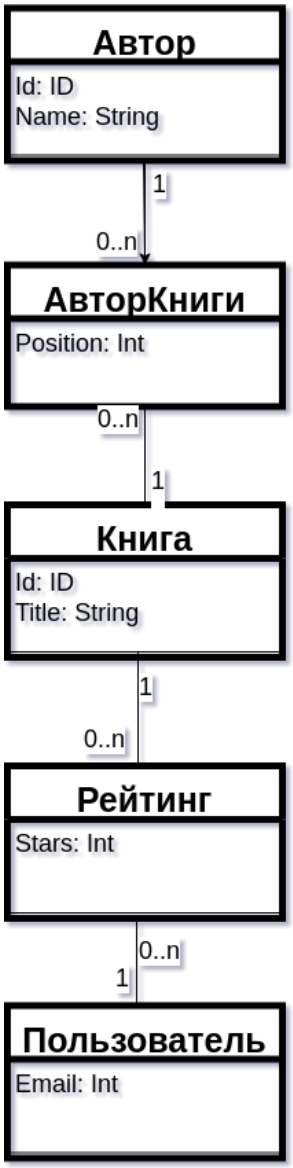
REST


http://domain.com/messages/..
http://domain.com/profile/..
http://domain.com/friends/..
http://domain.com/galery/..

GraphQL



GraphQL Schema




```
type Book {
  id: ID!
  title: String!
  authors: [BookAuthor!]
}

type Author {
  id: ID!
  name: String!
  books: [BookAuthor!]
}
```

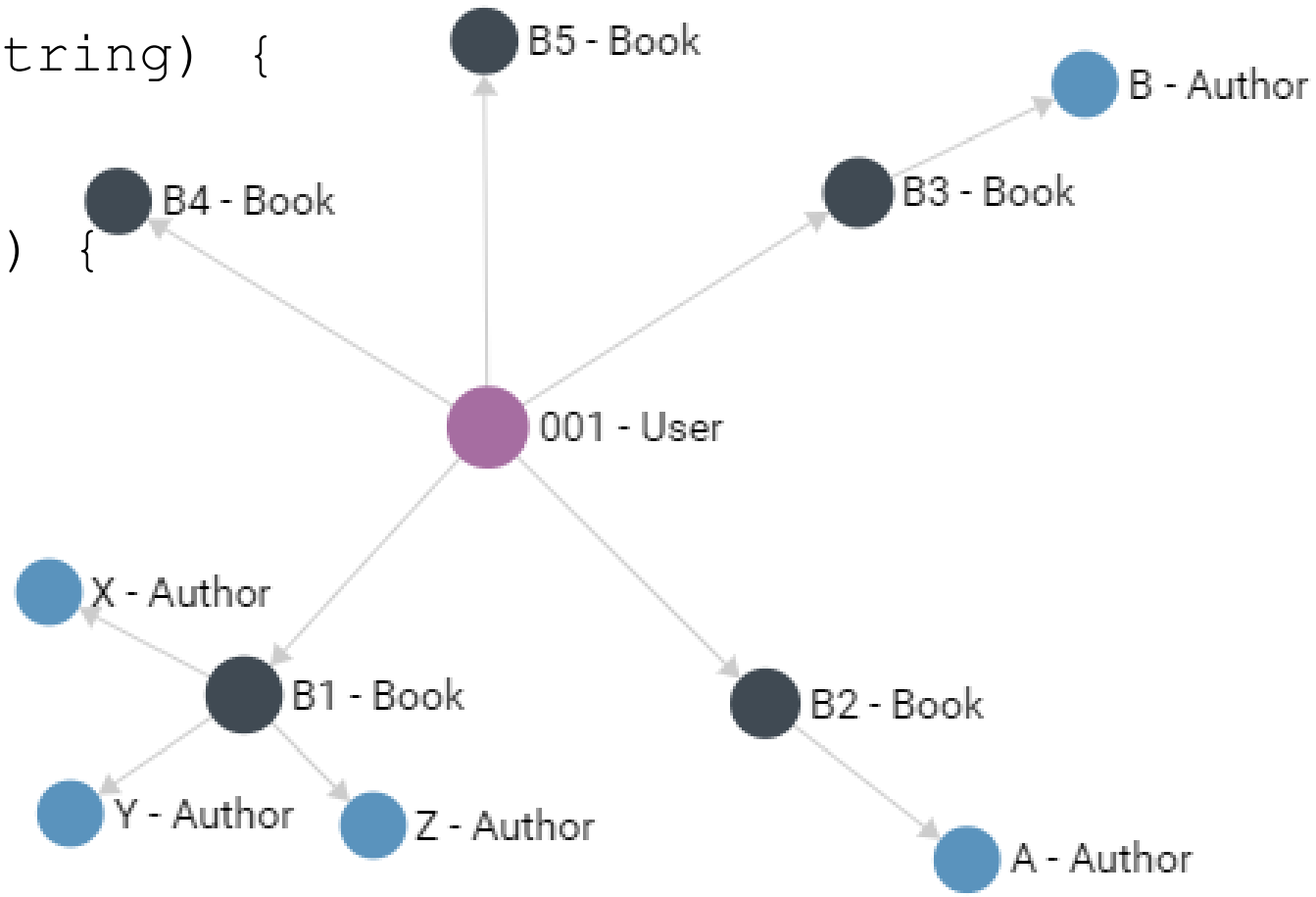
```
type BookAuthor {
  book: Book!
  position: Int!
  author: Author!
}

type Rating {
  book: Book!
  stars: Int!
  user: User!
}

type User {
  email: String
}
```

GraphQL Query

```
query books(title: String) {
  id
  title
  authors(first: Int) {
    position
    author {
      id
      name
      books {
        id
        title
      }
    }
  }
}
```



GraphQL Mutation

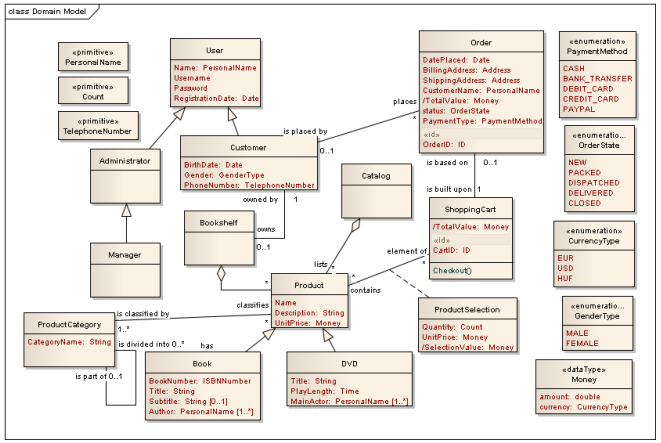
```
input UpdateBookTitleInput {
  bookId: ID!
  newTitle: String!
}

mutation updateBookTitle(value: UpdateBookTitleInput){
  book: Book
}
```

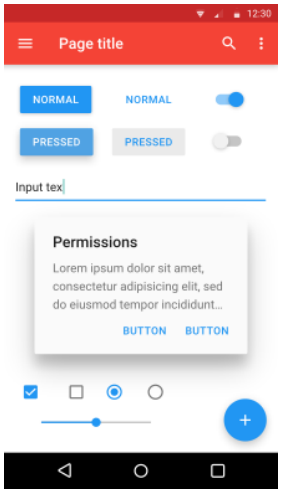
Слои приложений

DB

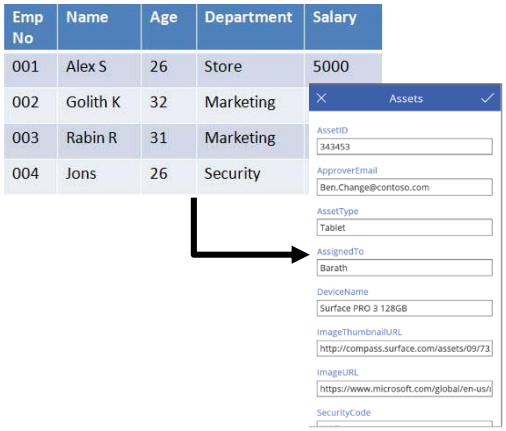
ORM



Domain Model



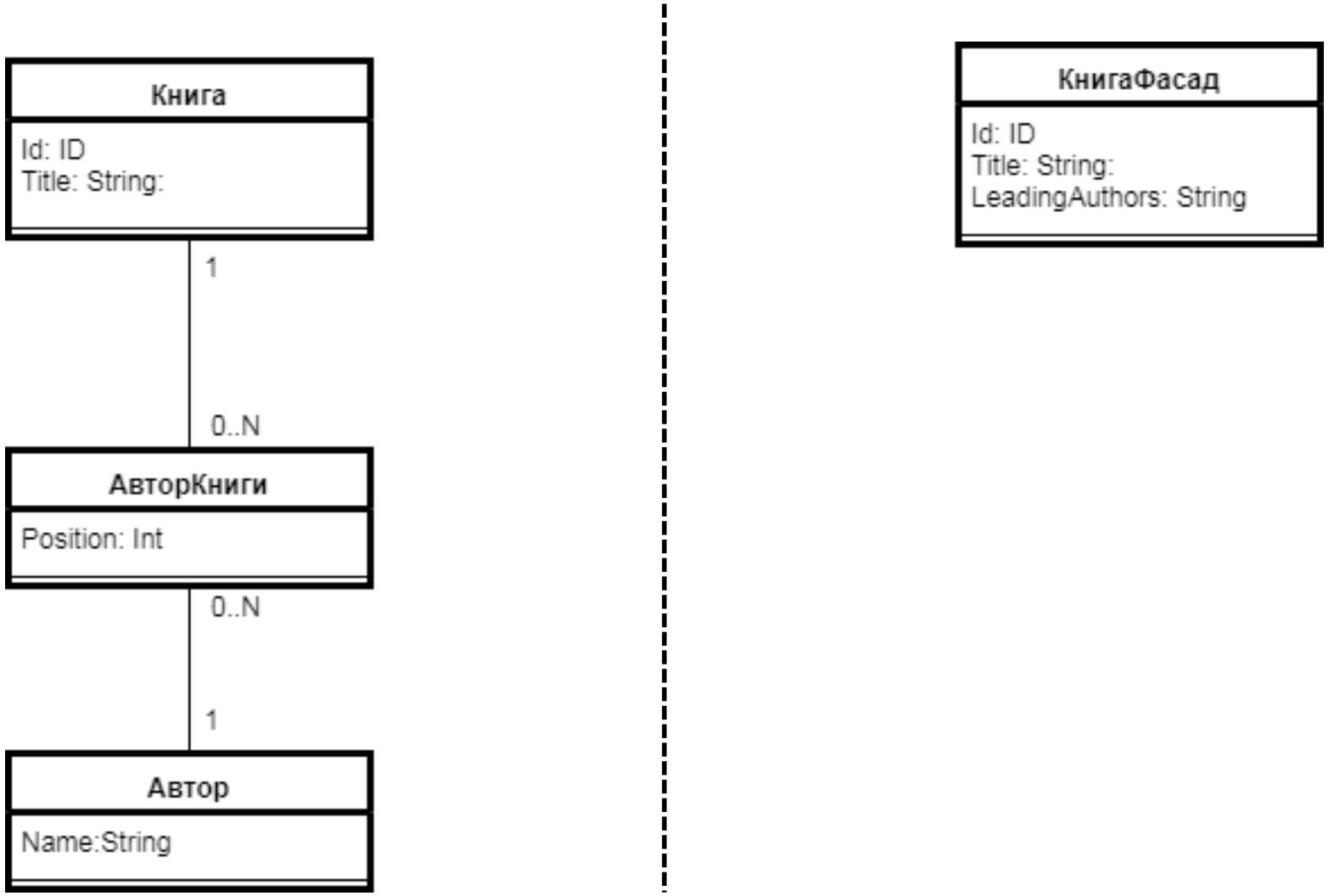
UI Toolkit



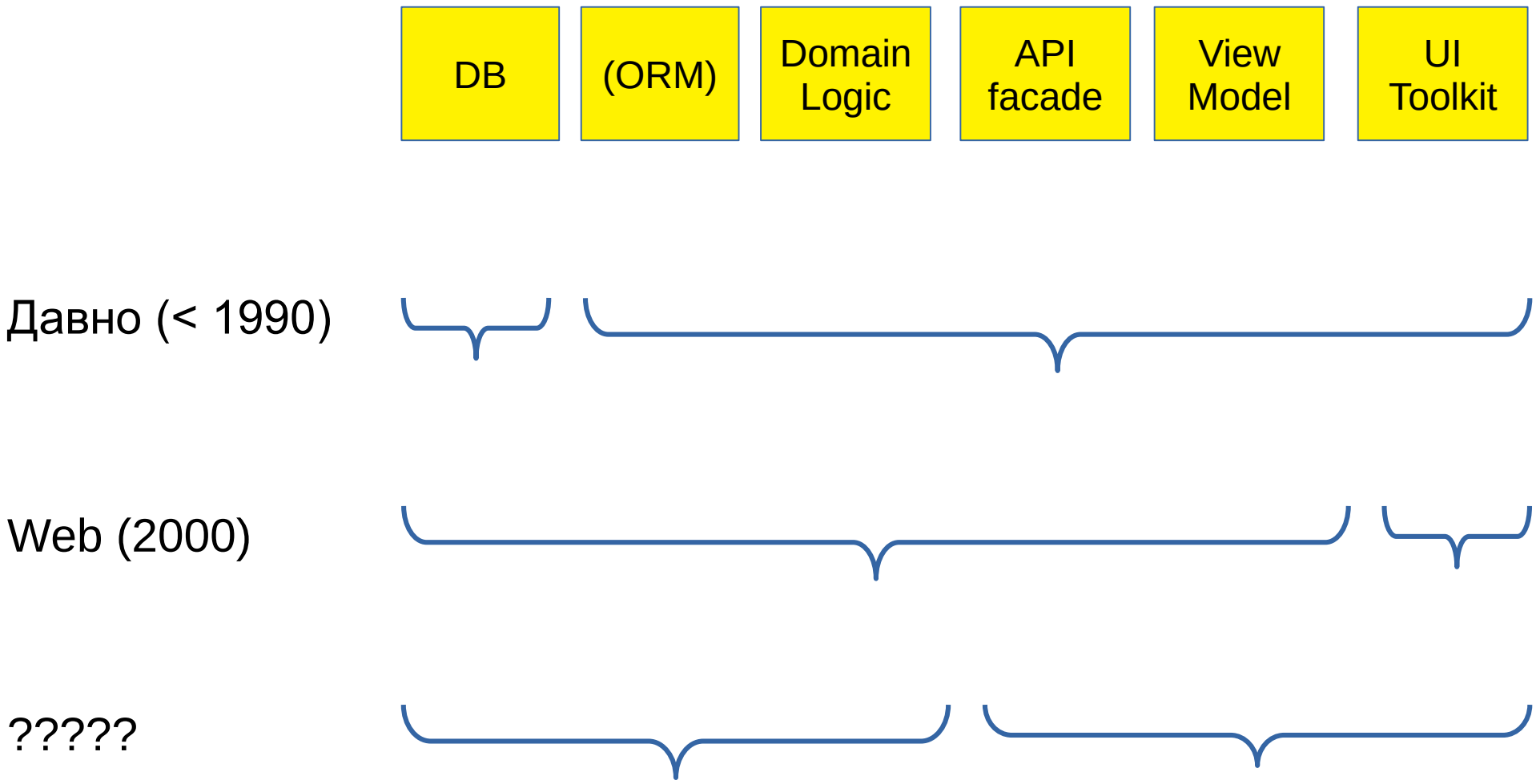
View Model

API-facade

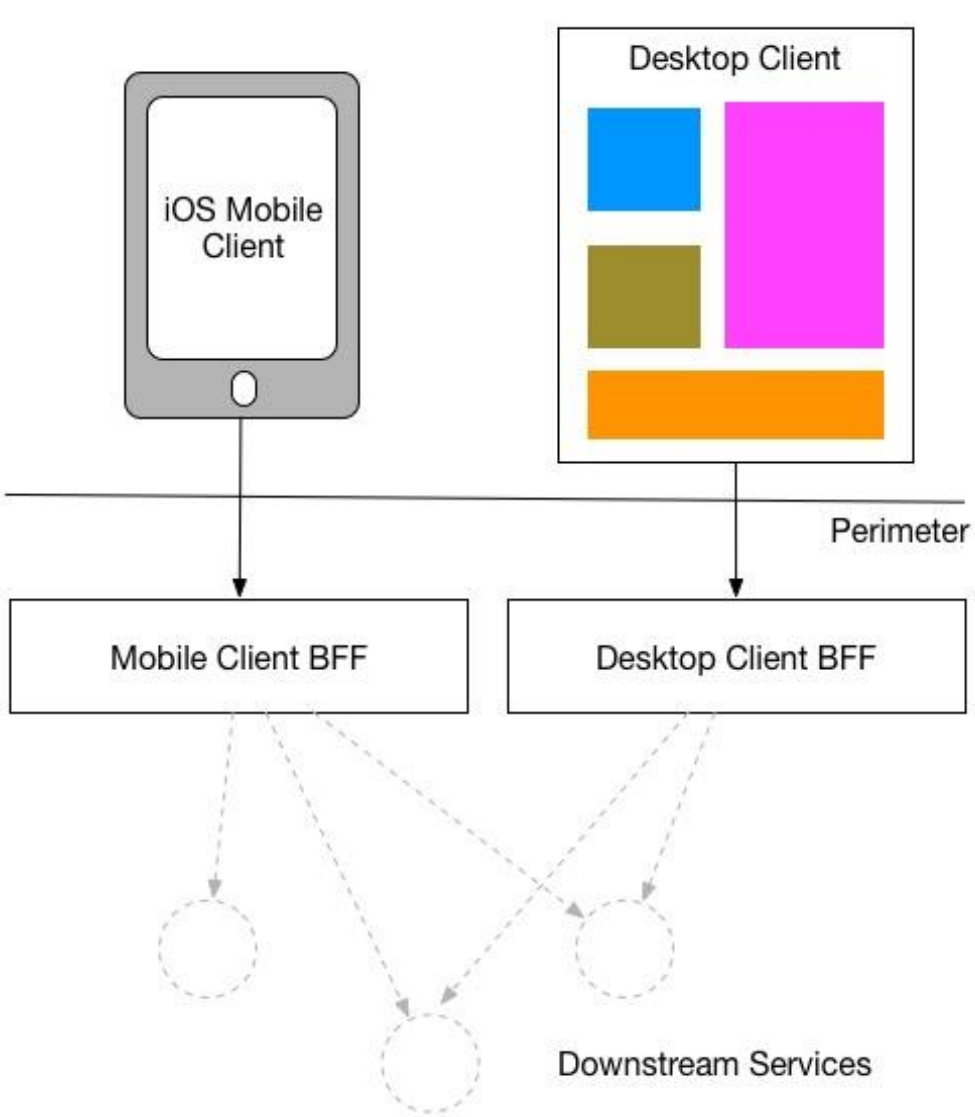
Преобразования в API-фасаде

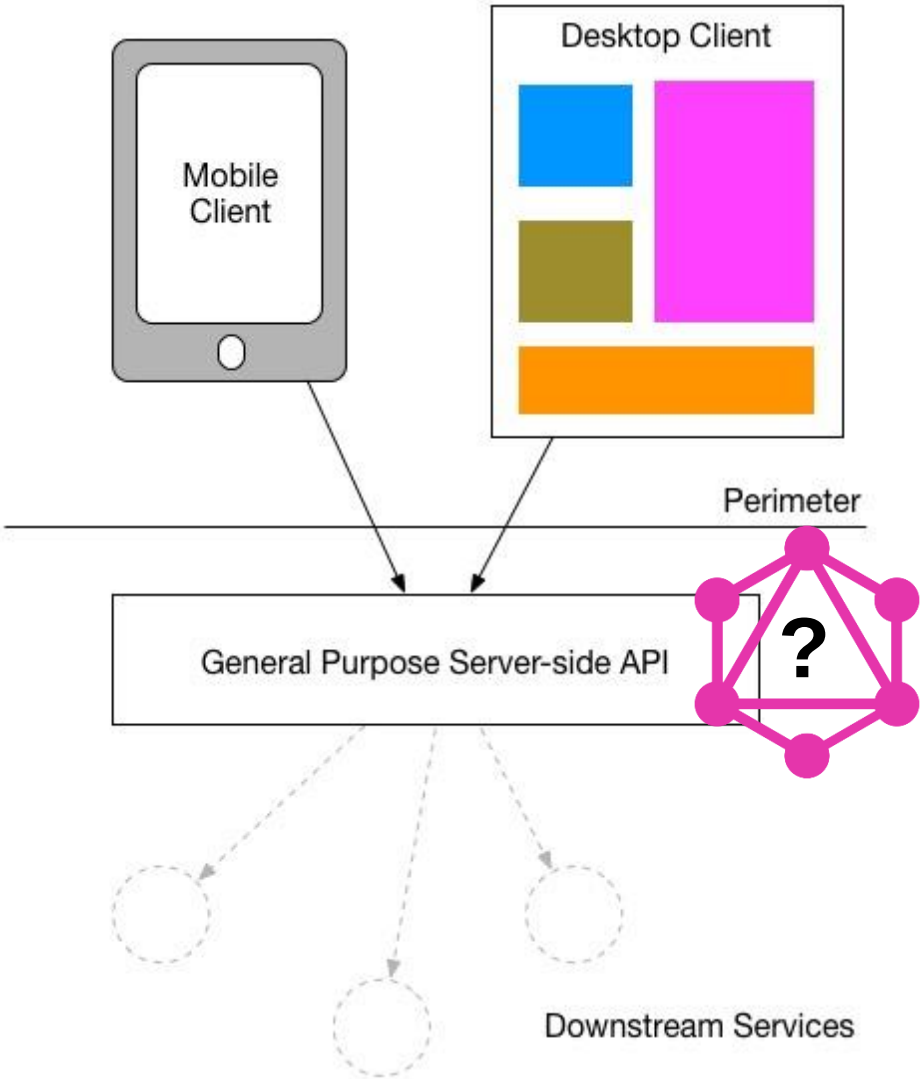


История API-facade

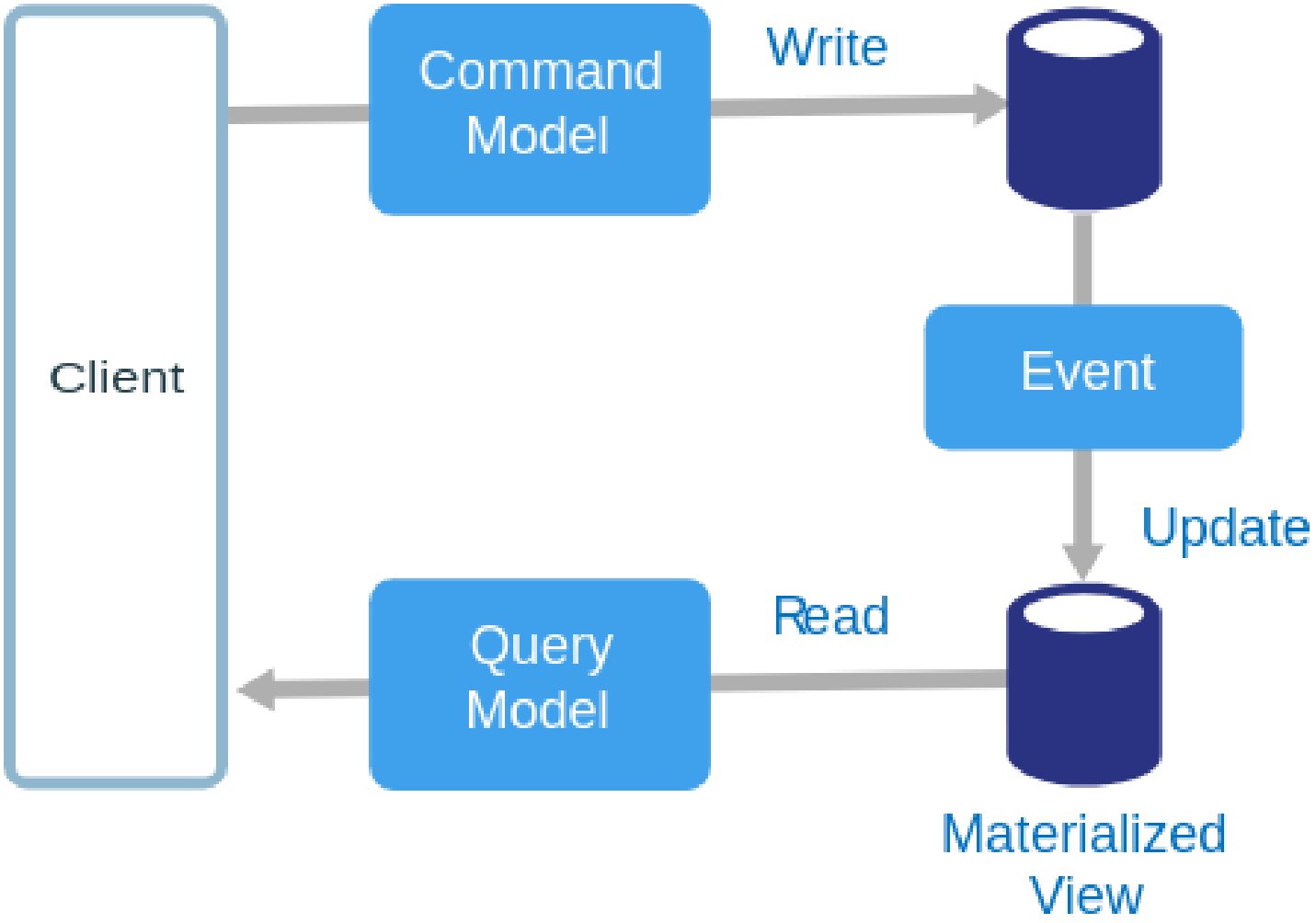


BFF

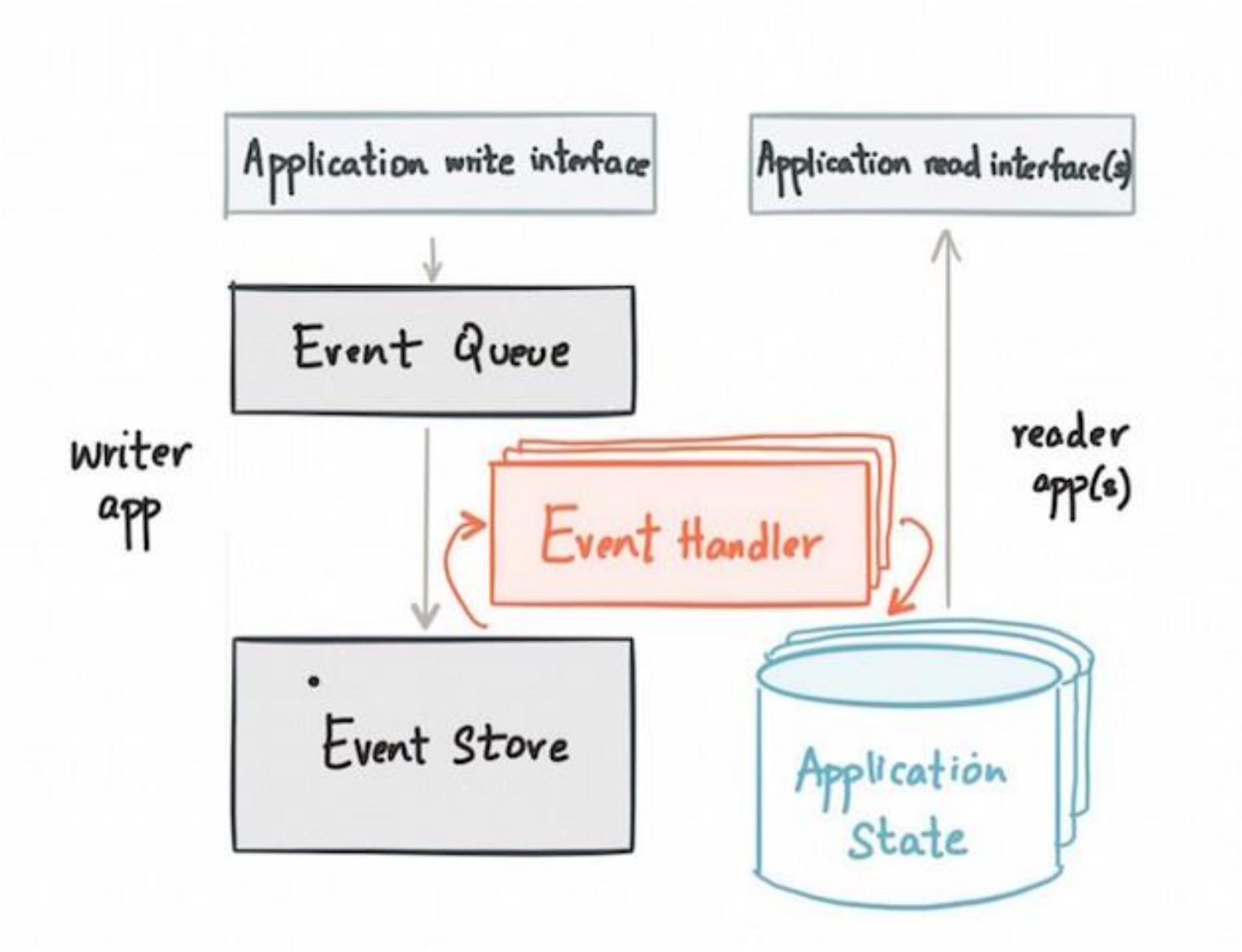

iOS Mobile Client
Desktop Client
Perimeter
Mobile Client BFF
Desktop Client BFF
Downstream Services


Mobile Client
Desktop Client
Perimeter
General Purpose Server-side API
?
Downstream Services

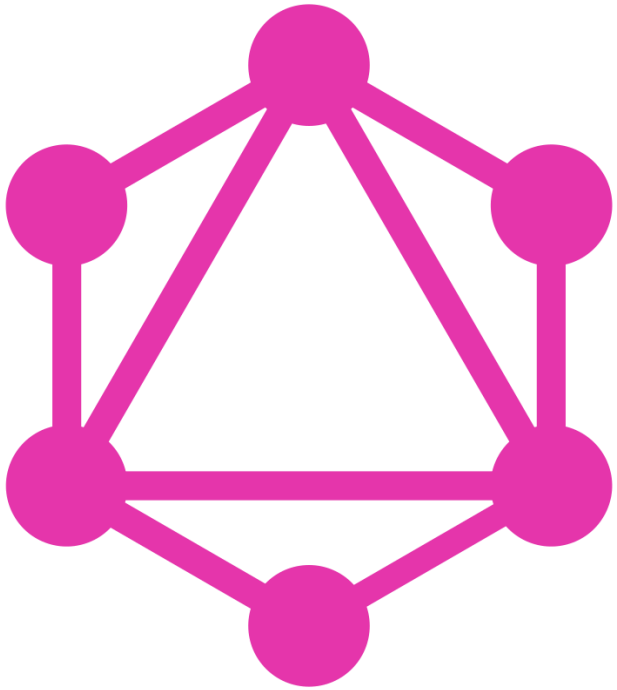
CQRS


Client
Command Model
Write
Event
Update
Query Model
Read
Materialized View

Event Sourcing



GraphQL



- Model
- Query
- Mutation
- Subscription

Библиотеки

Server-side

- Apollo Server
- graphql-dotnet
- EntityGraphQL
- Sangria (Scala)
- ….

Client-side

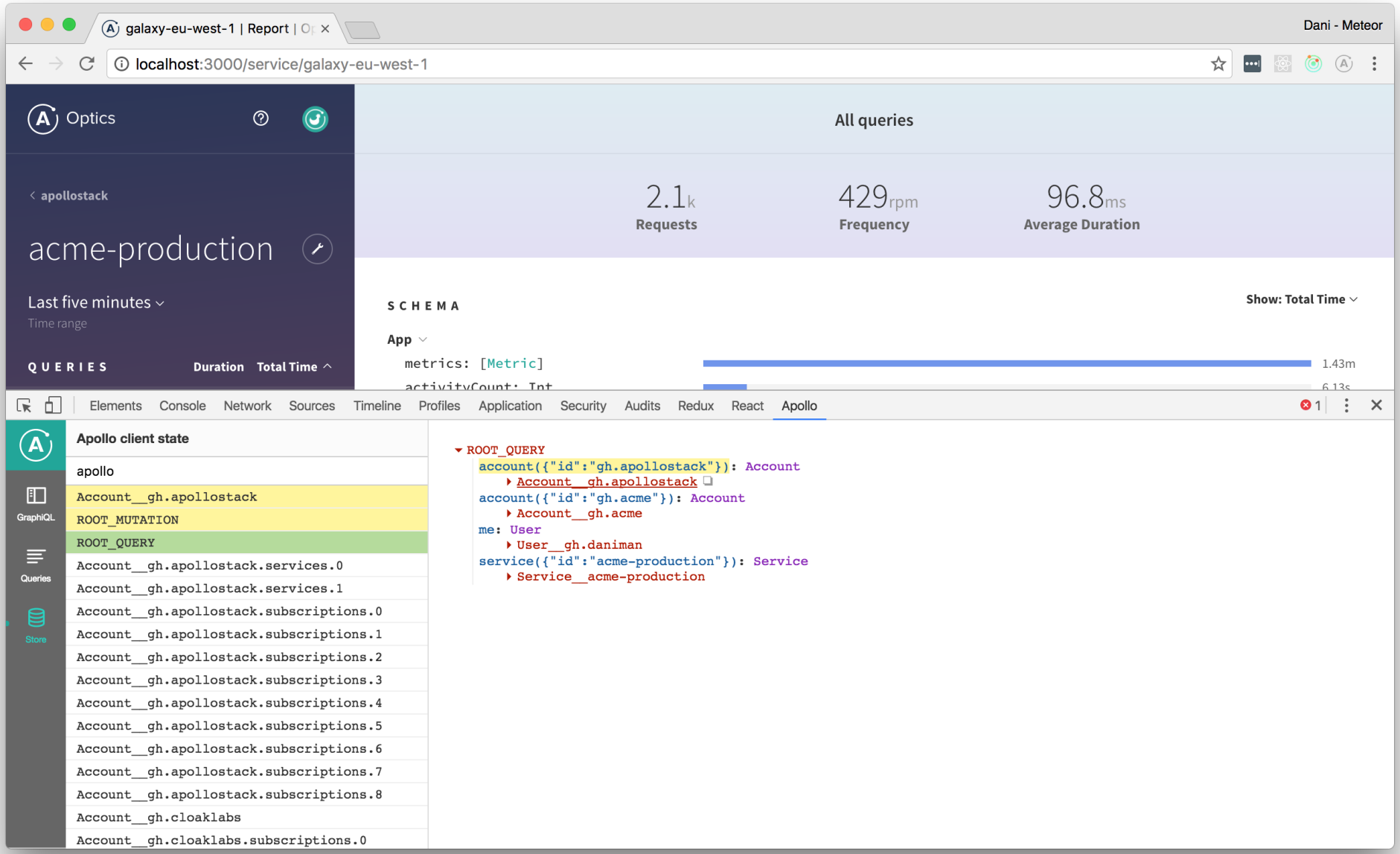
- Apollo Client
- React Relay
- …

Инструментарий

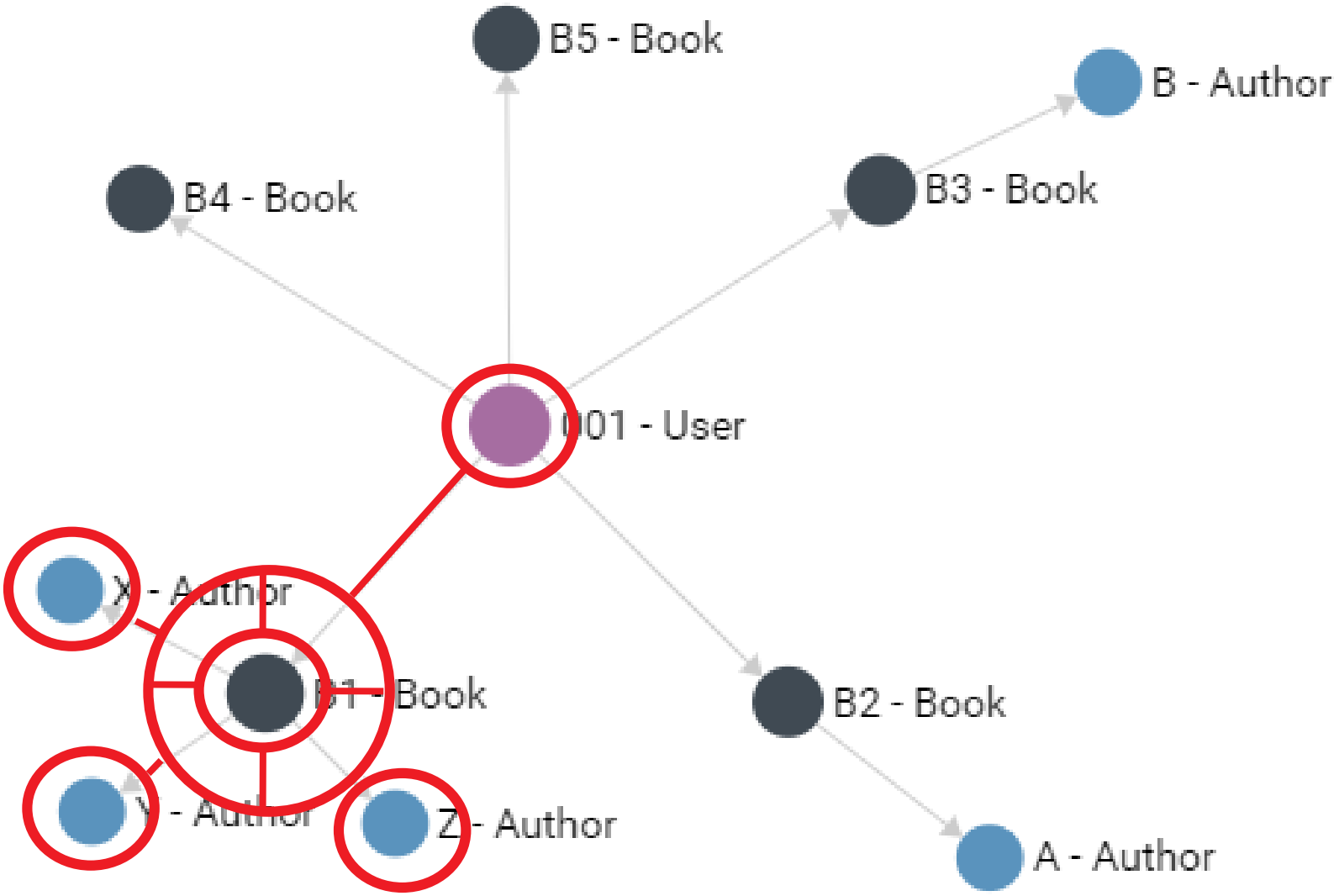
- Отладчик для Chrome
- Мэппинг для Swagger
- Мэппинг для OData
- …

- graphql-code-generator (серверная и клиентская части)
- GraphQL плагины для VSCode
- GraphQL Playground
- …

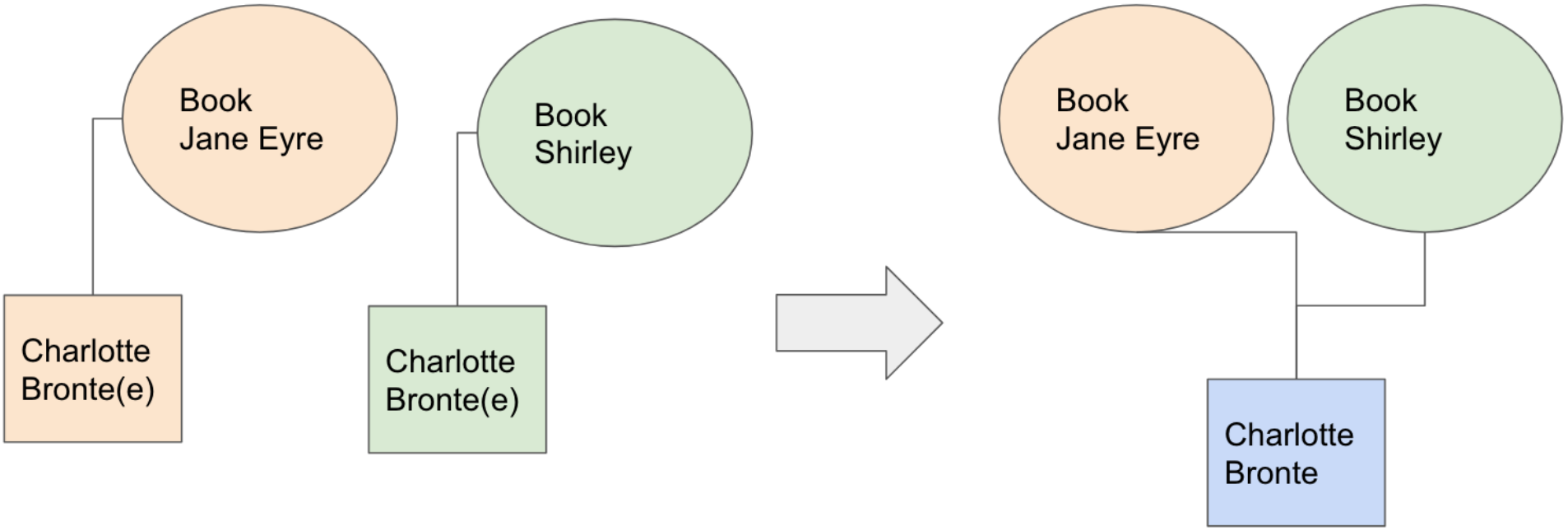
Инструментарий



Server — Resolvers



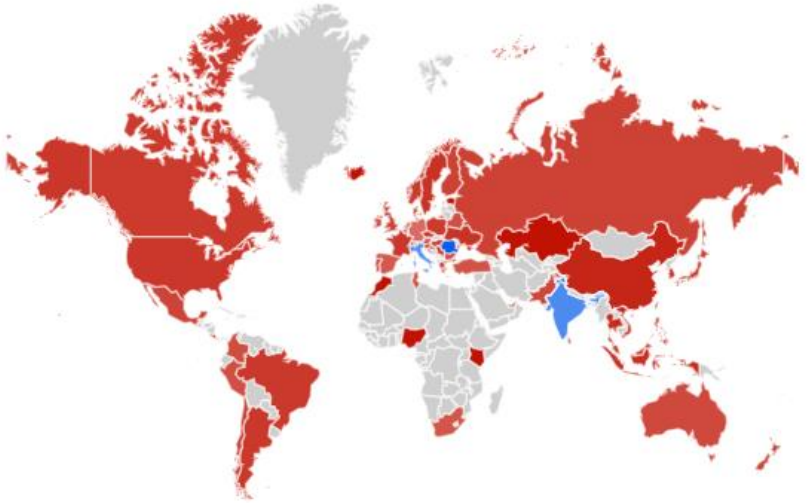
Client — Normalized Cache

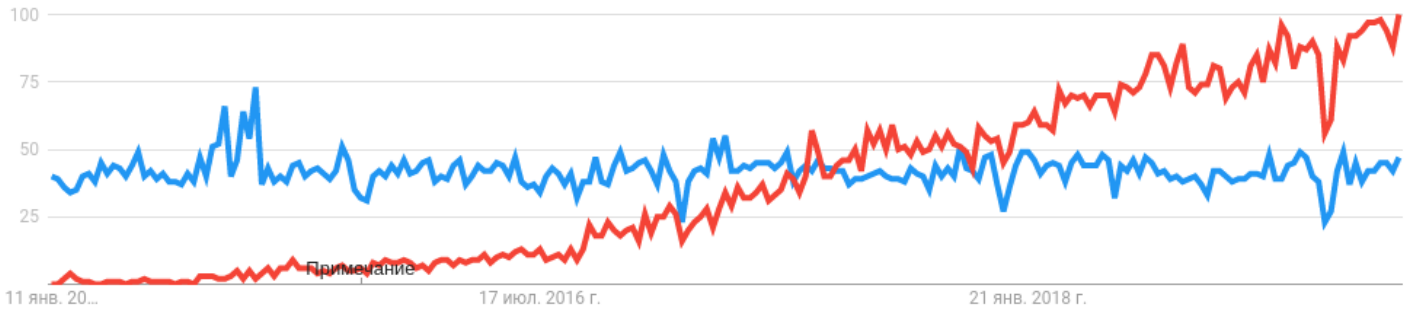


OData

GraphQL:

```
human(id: ""1"") {
    name
    friends {
      name
      appearsIn {
        name
      }
    }
}
```





OData:

Human?$filter=Id eq '1'&$select=Name&$expand=Friends($select=Name;$expand=AppearsIn($select=Name))

Производительность

- DataLoader
- Анализ сложности запросов
- Предопределенные запросы
- Ограничение времени выполнения

GraphQL в Enterprise

Vienna.js, GraphQL in the Enterprise: From Legacy to Bleeding Edge, January 2019

Проблемы

- Использование не по назначению

  Альтернативы:

  - gRPC
  - REST
  - OData
  - …
- Выбор технологии определяется контекстом использования

Варианты резолверов

- Прокси-объекты, полностью прячущие за собой доменные объекты
- Подклассы
- C# extensions