

GraphQL:

- Просто
- Четко
- Оптимально

Что такое GraphQL

GraphQL – 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 данные из одной точки. GraphQL описывает типы, чтобы быть уверенным, что приложения запрашивают только то, что возможно и предоставляют ясные и понятные сообщения об ошибках.

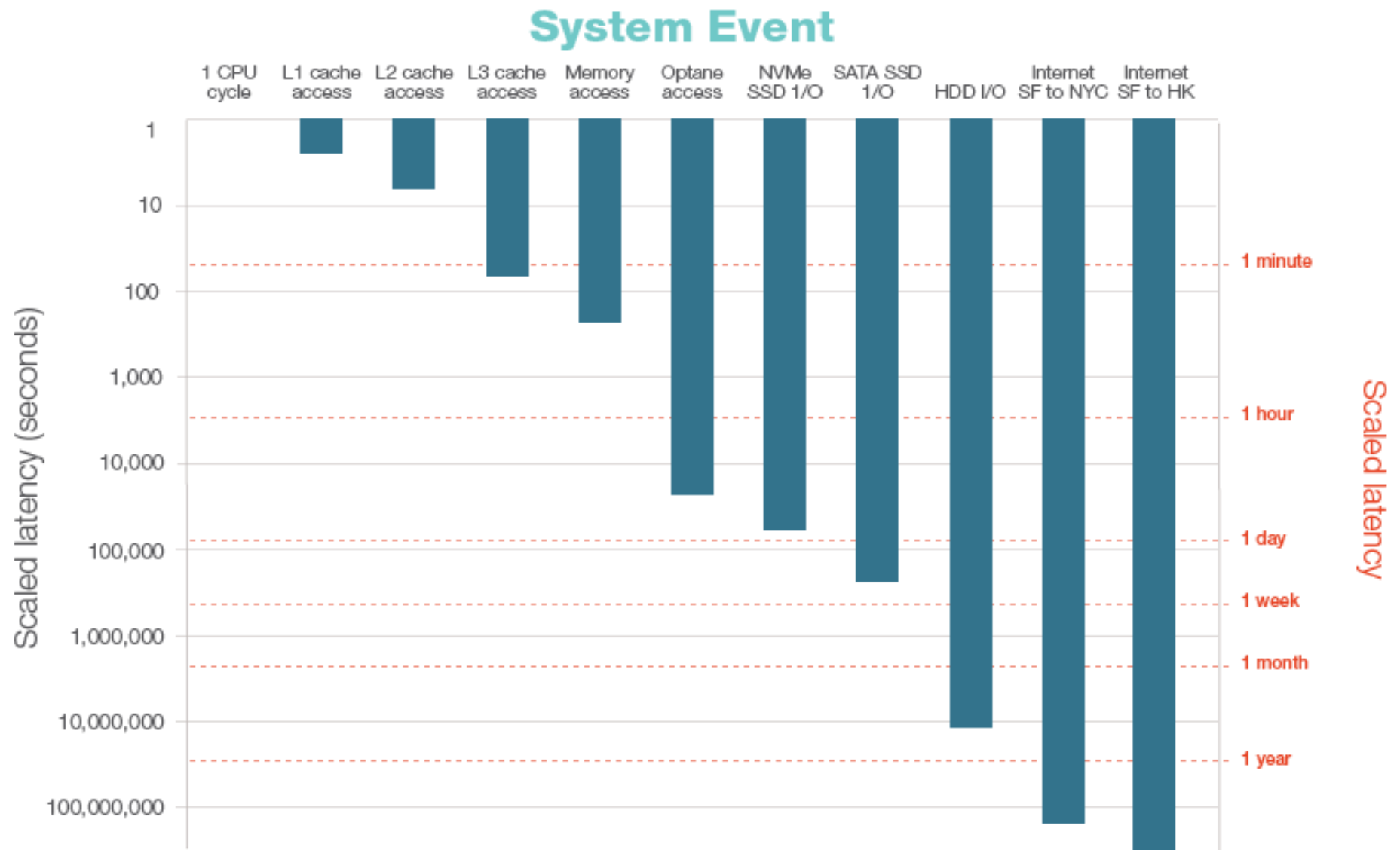
История

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

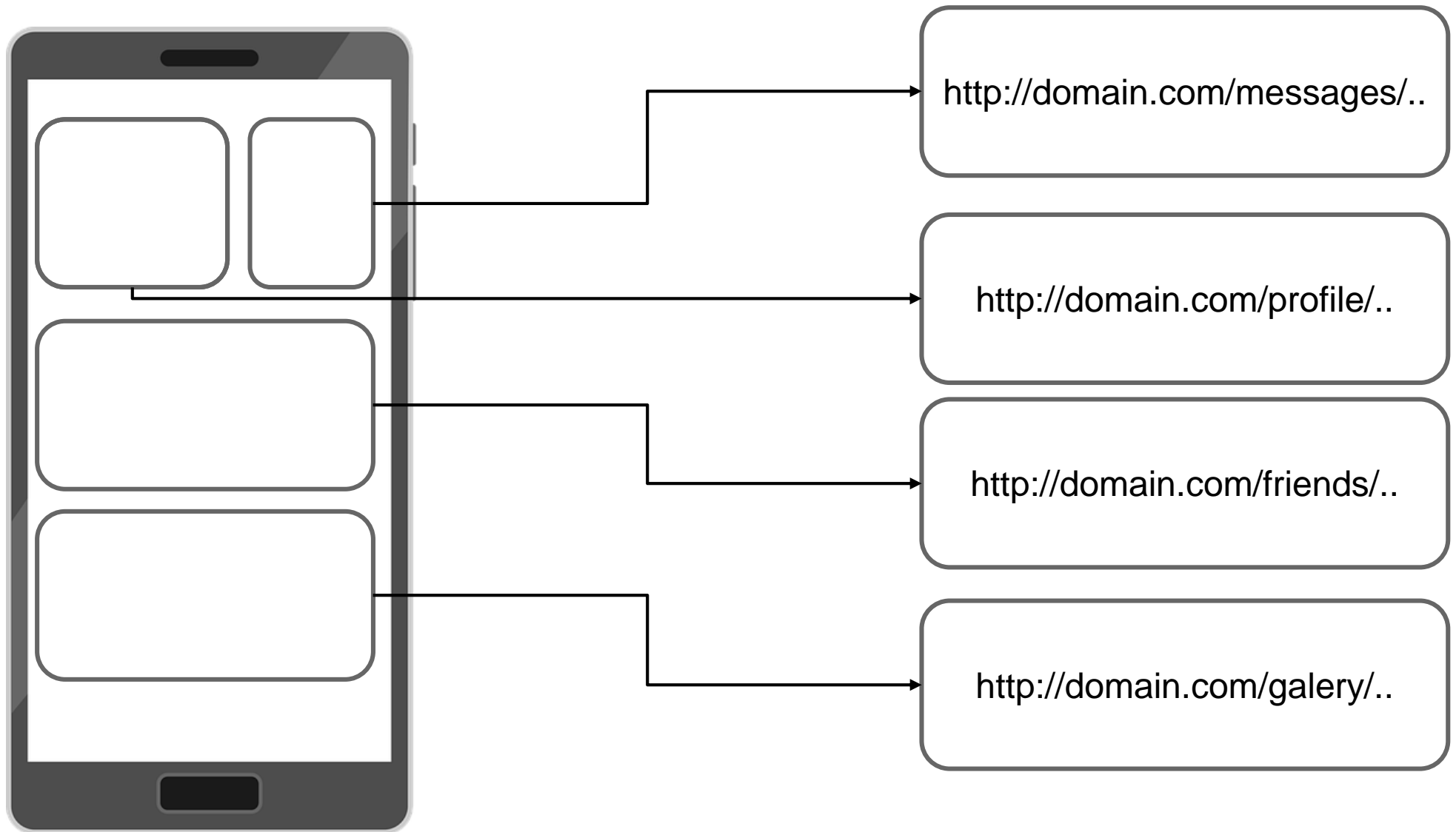
Предпосылки

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

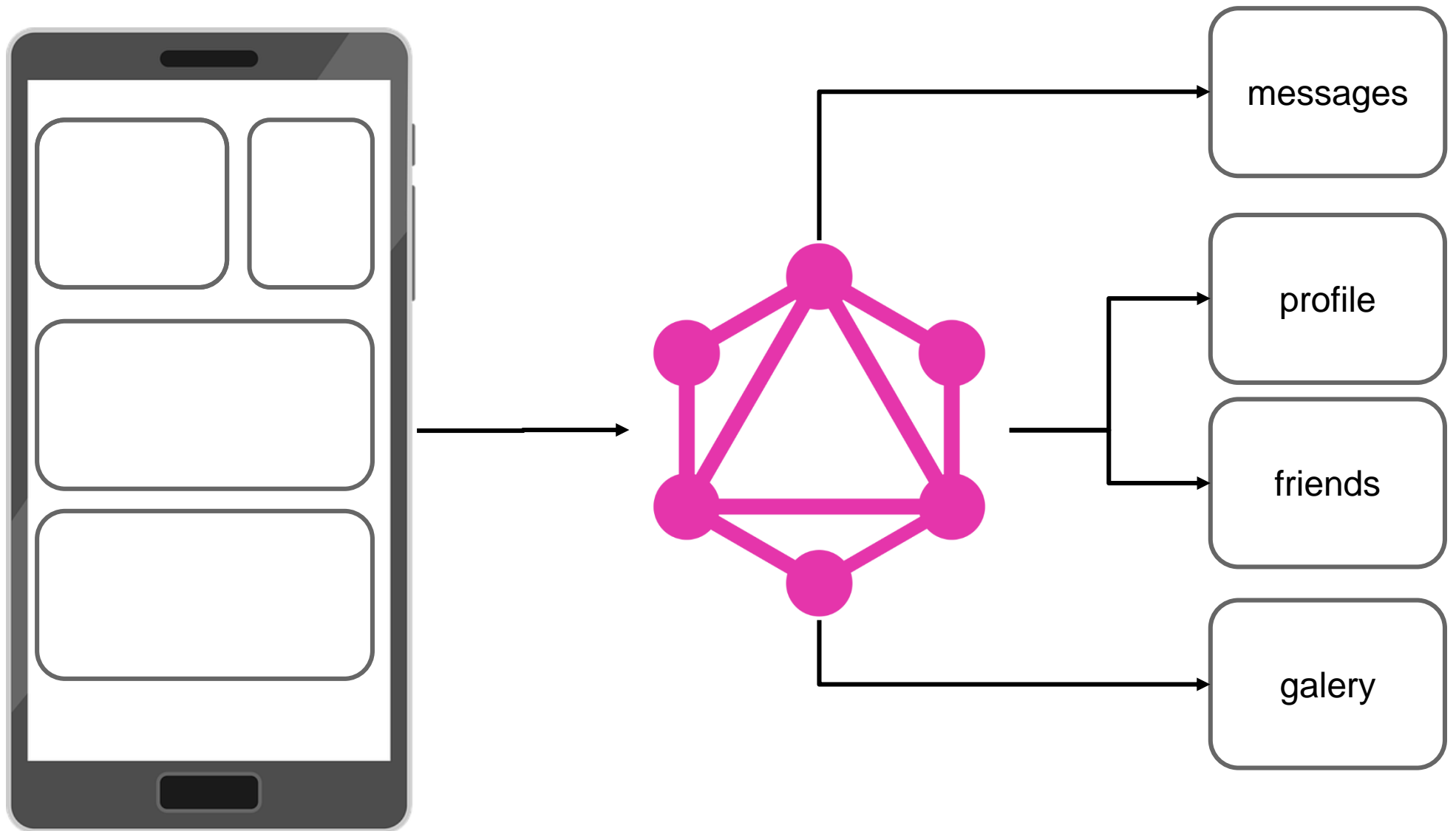
В наглядном масштабе



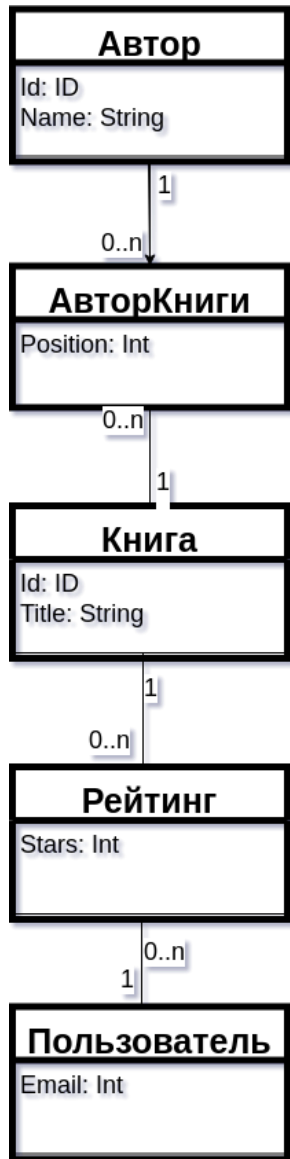
REST



GraphQL



GraphQL Schema



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

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

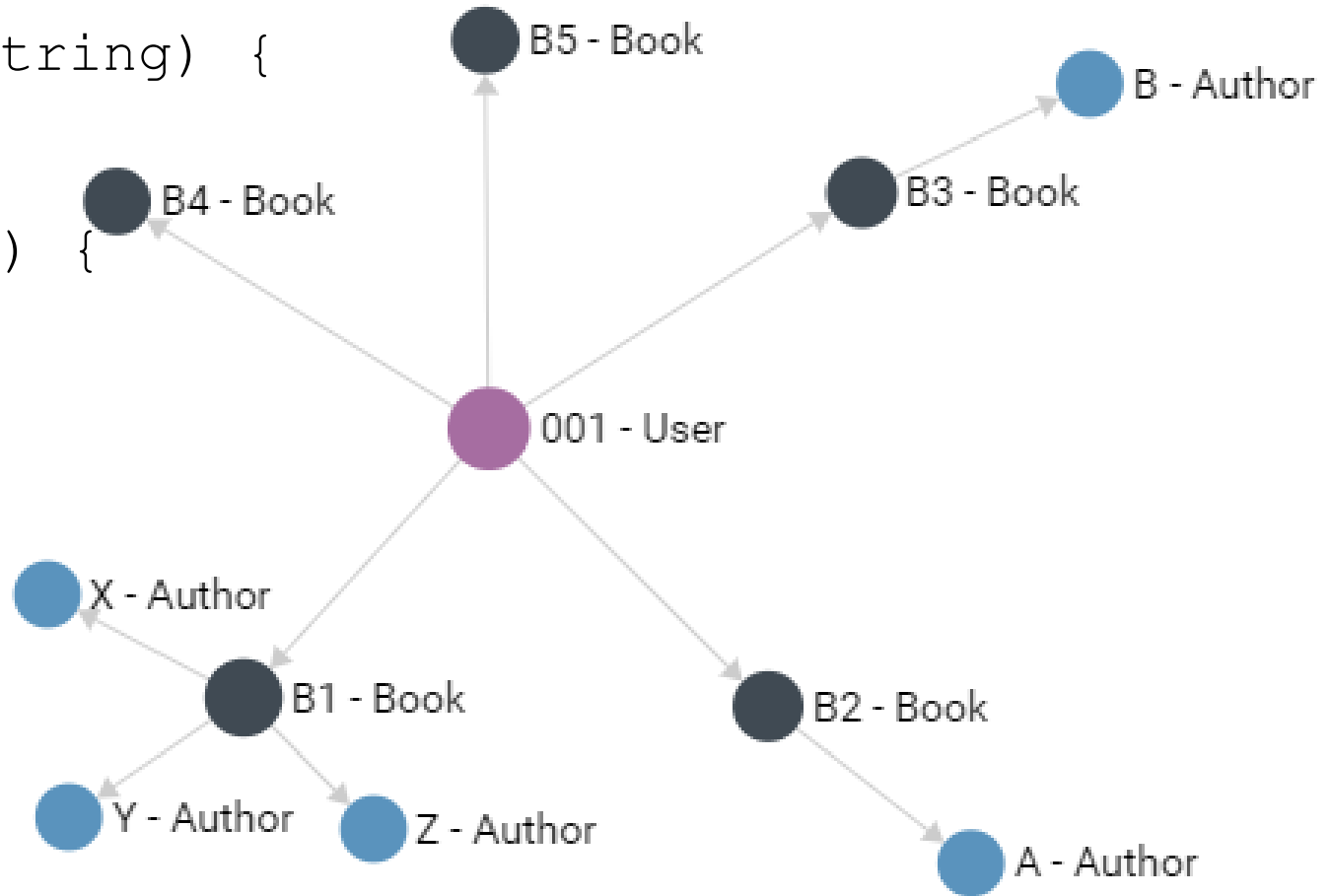
```
type BookAuthor {  
  book: Book!  
  position: Int!  
  author: Author!  
}
```

```
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

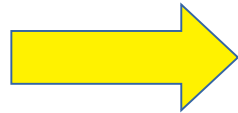
```
input UpdateBookTitleInput {  
  bookId: ID!  
  newTitle: String!  
}
```

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

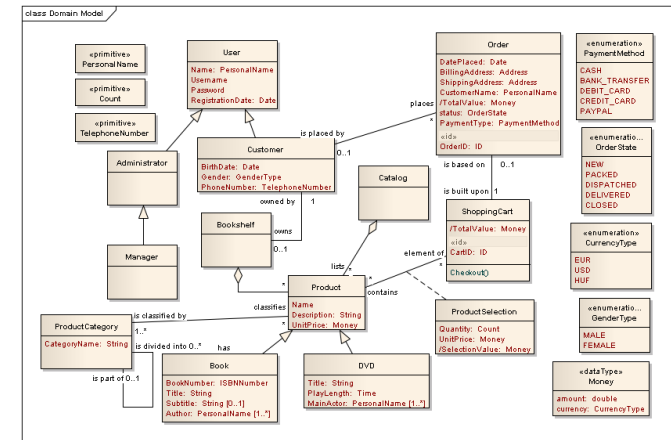
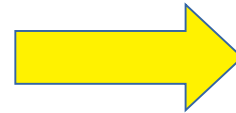
Слои приложений



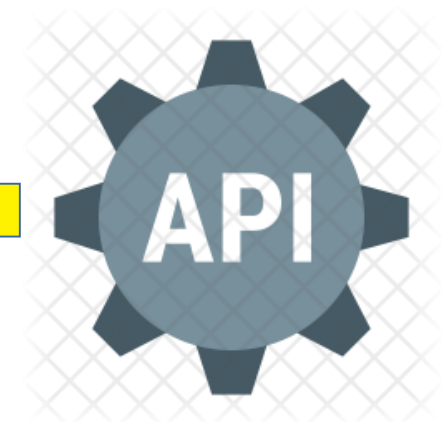
DB



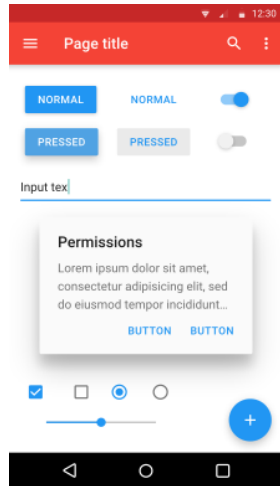
ORM



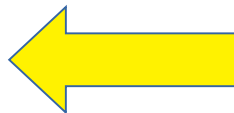
Domain Model



API-facade



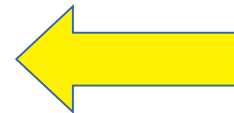
UI Toolkit



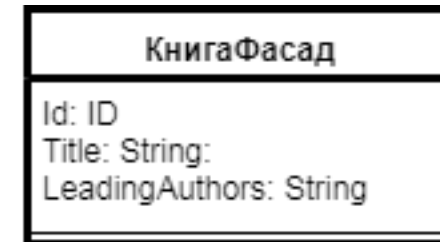
Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	
003	Rabin R	31	Marketing	
004	Jons	26	Security	

Assets	
AssetID	343453
AssetType	Tablet
AssignedTo	Barath
DeviceName	Surface PRO 3 128GB
ImageThumbnailURL	http://compass.surface.com/assets/09/73
ImageURL	https://www.microsoft.com/global/en-us/
SecurityCode	

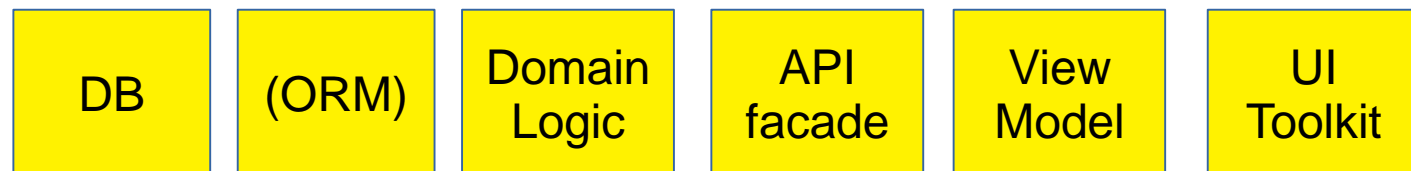
View Model



Преобразования в API-фасаде



История API-facade



Давно (< 1990)



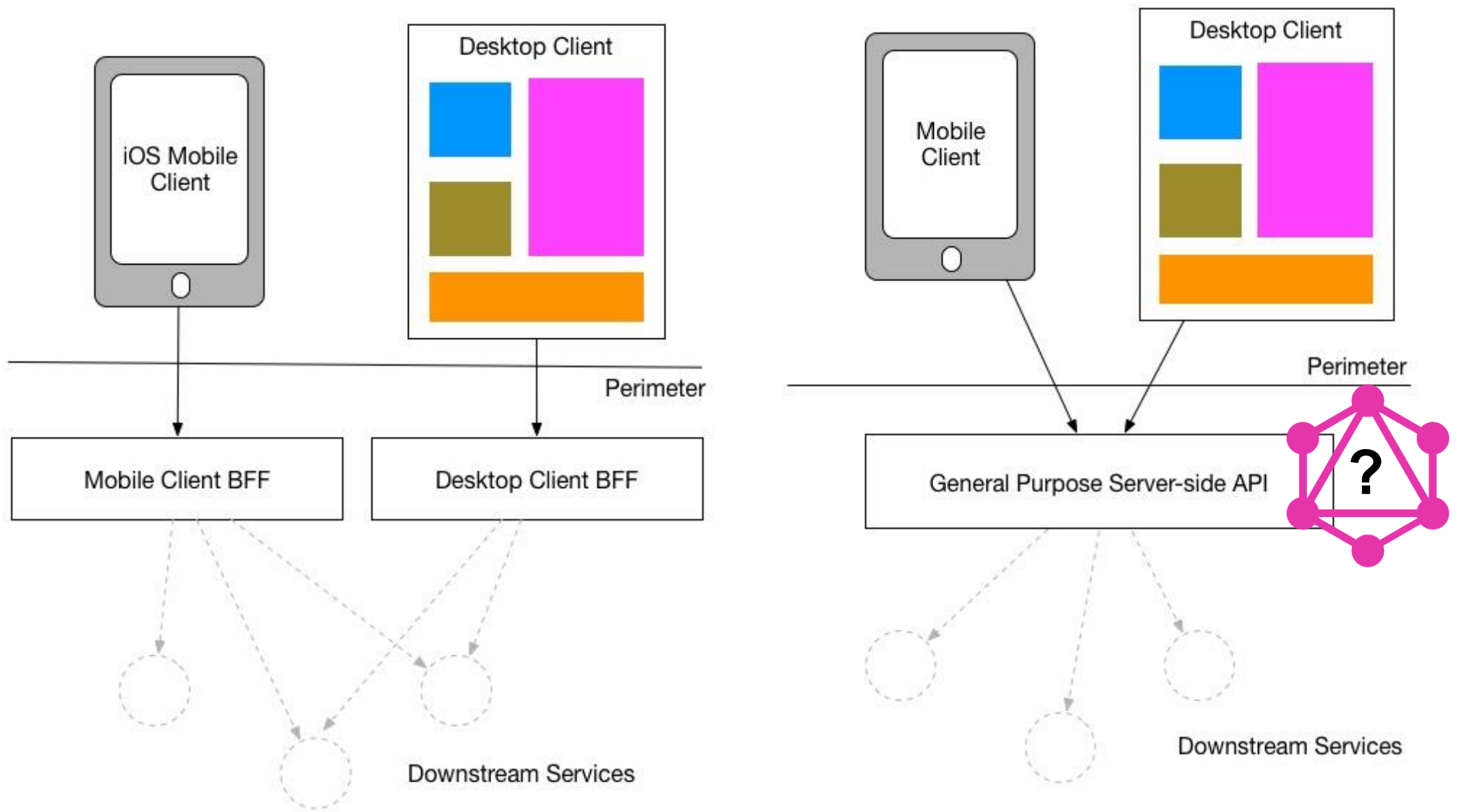
Web (2000)



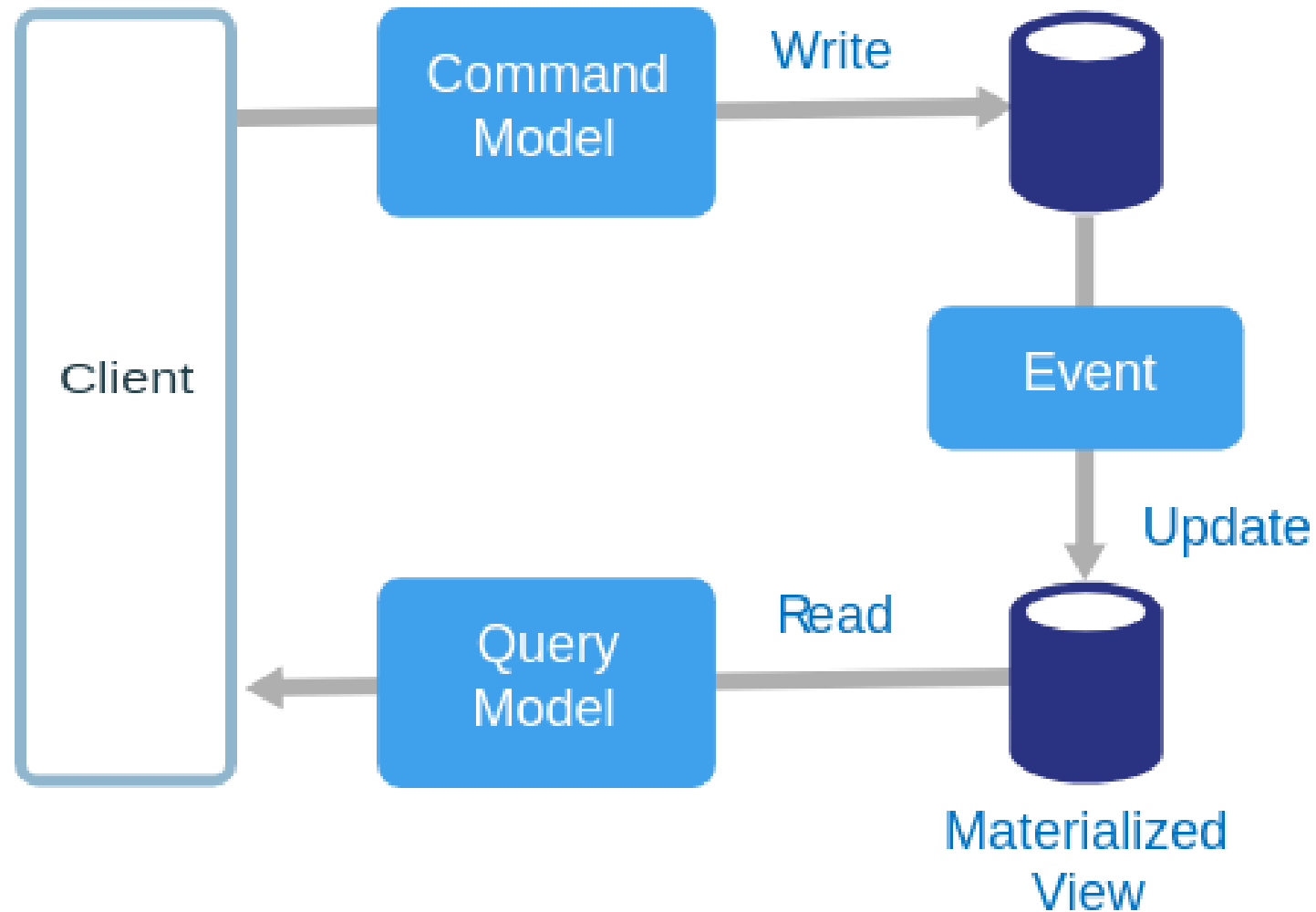
?????



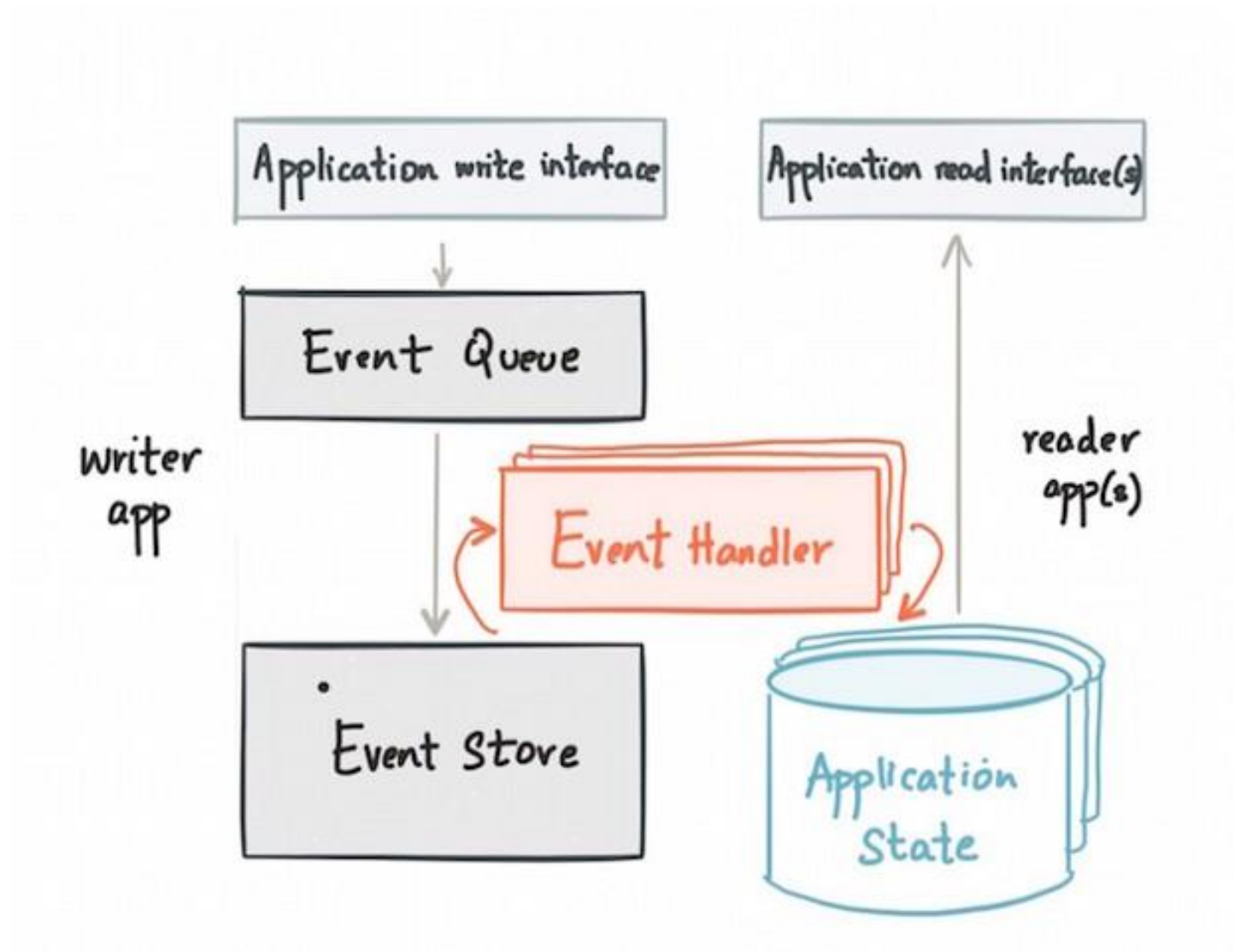
BFF



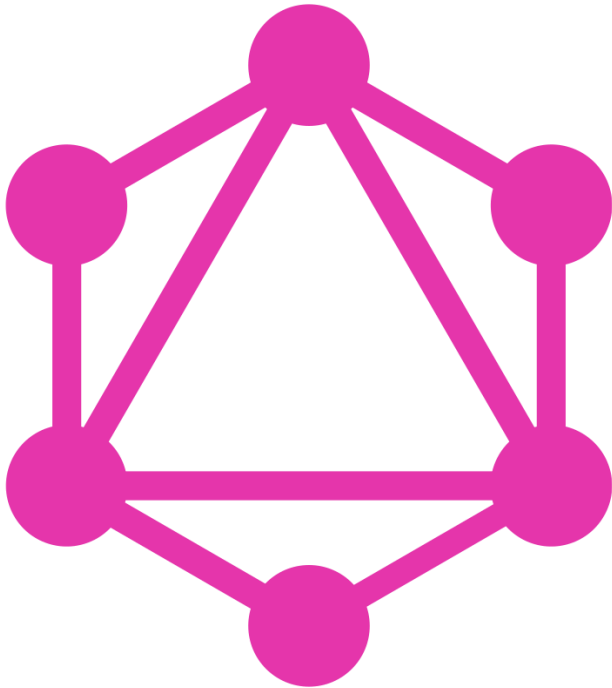
CQRS



Event Sourcing



GraphQL



- Model
- Query
- Mutation
- Subscription

Библиотеки

Server-side

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

Client-side

- Apollo Client
- React Relay
- ...

Инструментарий

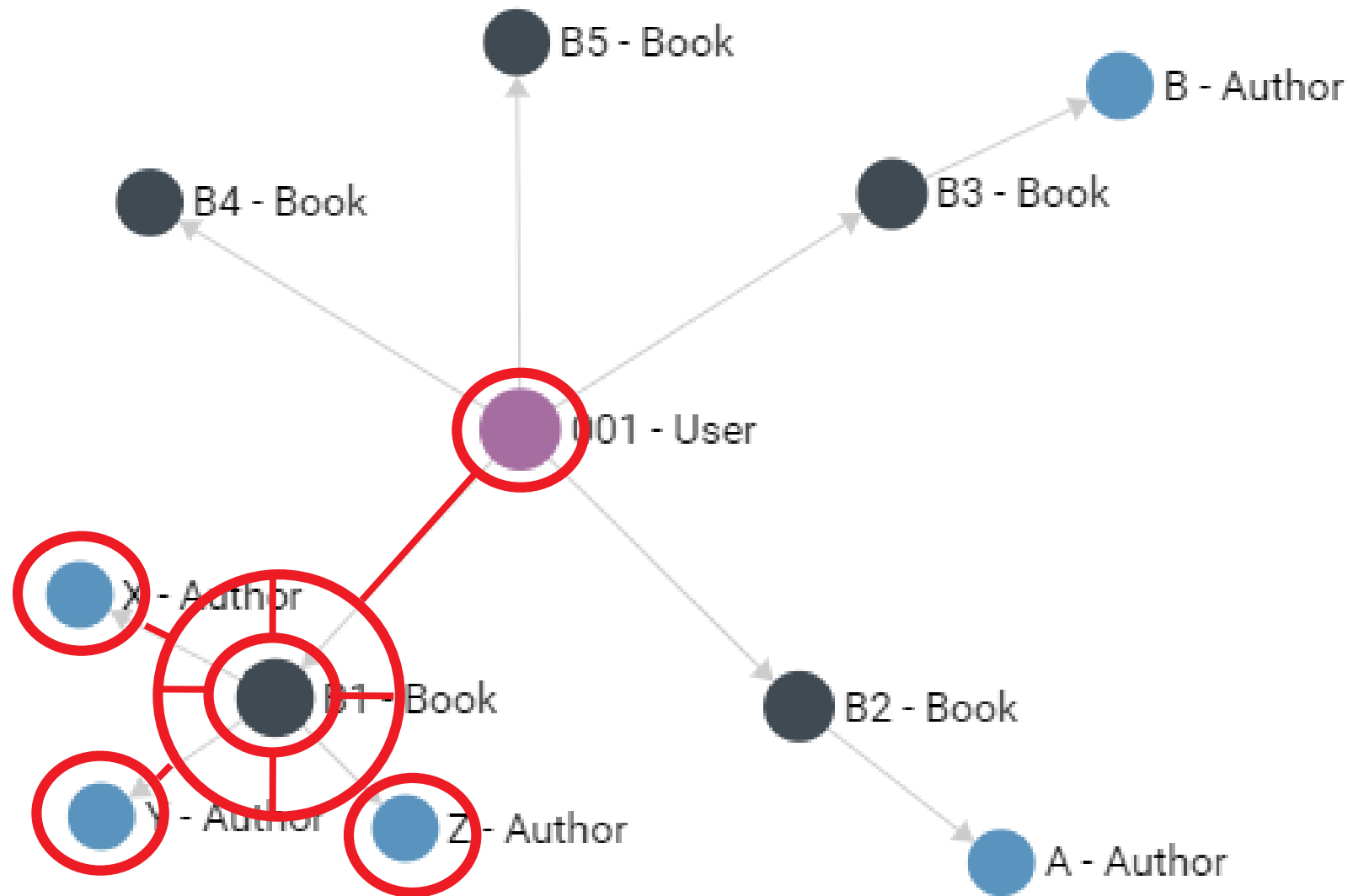
- Отладчик для Chrome
- Мэппинг для Swagger
- Мэппинг для OData
- ...
- graphql-code-generator (серверная и клиентская части)
- GraphQL плагины для VSCode
- GraphQL Playground
- ...

Инструментарий

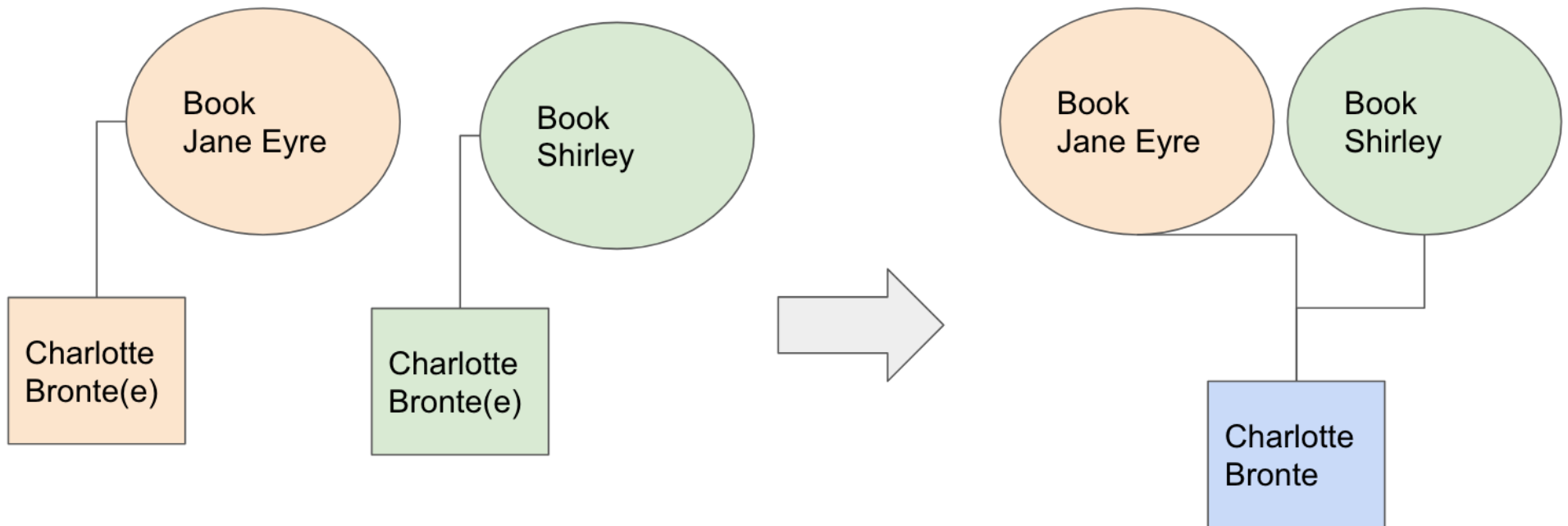
The screenshot displays the Apollo Optics web interface in a browser window. The browser's address bar shows the URL `localhost:3000/service/galaxy-eu-west-1`. The interface is divided into several sections:

- Header:** Includes the Apollo logo, the name "Optics", and a user profile "Dani - Meteor".
- Left Sidebar:** Contains the application name "acme-production", a time range selector set to "Last five minutes", and a list of queries under the heading "QUERIES".
- Main Content Area:**
 - Summary Cards:** Displays "2.1k Requests", "429rpm Frequency", and "96.8ms Average Duration".
 - SCHEMA:** Shows a dropdown for "App" and a metric "metrics: [Metric]" with a corresponding bar chart.
 - Queries List:** A table listing various queries, including "apollo", "Account_gh.apollostack", "ROOT_MUTATION", "ROOT_QUERY", and several "Account_gh.apollostack.services" and "Account_gh.apollostack.subscriptions" entries.
- Bottom Panel:** Features a tabbed interface with "Apollo" selected. It shows the "ROOT_QUERY" expanded, displaying a JSON-like structure of the query results, including fields like "account", "me", and "service".

Server — Resolvers



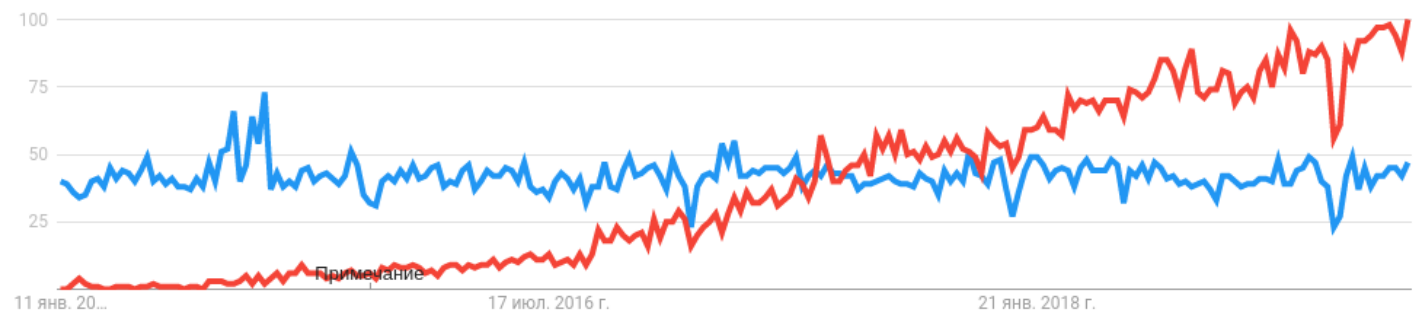
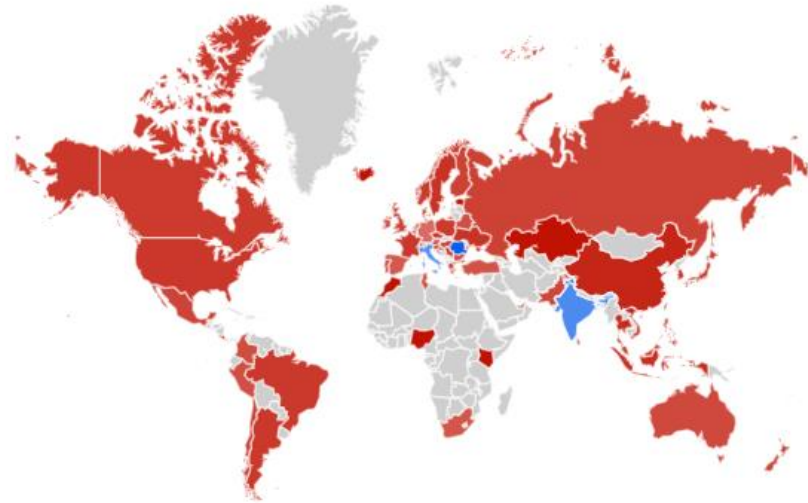
Client — Normalized Cache



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

Talk presented at
ViennaJS

Video recorded by
PUSHER

GRAPHQL IN THE ENTERPRISE FROM LEGACY TO BLEEDING EDGE

v1.2.8



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

▶ ▶| 🔊 0:00 / 37:16

CC 🔧 📺 📱 🖥️

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