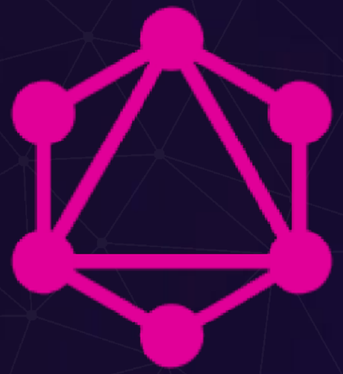
Scott

宋小菜前端技术专家

《GraphQL 数据聚合层解放前后端》



GraphQL 数据聚合层

宋小菜 | Scott 陈锦辉

GraphQL 与领域驱动带来的协同价值

协同 效率

领域驱动 前后端职能变化

不求高大上 只求接地气

无论我做多烂 都 Show 给大家看

GraphQL 的哲学理念是什么？什么是 SDL First？

什么样的业务场景和团队适合用 GraphQL？

GraphQL 对于前端和后端意味着什么？会带来什么职能变化？

什么时机适合从 RESTFul API 迁移到 GraphQL？

如何做前后端的技术选型/技术架构？

如何做 Schema Loader/鉴权/缓存/数据代理的模块设计？

如何针对关系型数据库做缓存优化分？

如何做嵌套 DDos 安全防控？

GraphQL 的性能问题和常见的坑有哪些及如何解决？

在复杂应用（如工业领域）如何应用 GraphQL？

GraphQL 的 Schema Type/Resolver 如何做可视化工程依赖管理？

如何对 GraphQL 的 Query/Request 做性能监控？

领域建模对技术的价值是什么？

架构模块边界如何划分？

…


收益
有多大
如果
在我的团队
推广
技术挑战/坑
有哪些

菜 菜 司机 菜 供应商 福 采 仓

小程序

ERP 系统

市调系统

报表系统

...

· 可以解决的 ·

大伯伯打包平台	√ 解决多人多 App 打包问题
大伯伯推包系统	√ 解决包脚本测试与推包流程
大表姐发包系统	√ 解决 App 热更新的热更新发布
大瓜子市调模板平台	√ 解决动态表单制作组装需求
110 日志报警平台	√ 解决端异常收集与报警需求
姑奶奶异常跟踪系统	√ 解决异常分析与故障定级指派
RGB 可视化平台	√ 解决端访问行为可视化需求
堂哥工作台	√ 解决资源分布与周报日报集成需求
ITMS App 预装平台	√ 解决企业版 App 预装需求

…

· 解决不了的 ·

多端之间的

类报表同步

多端之间的

多模块共享

单端内的

多变页面形态

宋小菜产品开发流程


前后端协同阻塞节点
需求讨论
产品 prd 评审
UI 评审
系统设计
接口设计
接口 mock
代码开发
单元测试
集成测试
产品验收
灰度发布
产品上线

API 受制于页面

服务端被动升级

Mock 职责重合

前后工作流钳制

前端报表输出慢

强依赖后端接口

宋小菜新架构

- GraphQL 和 DDD 结合
- 省去后端网关接口层的胶水代码开发
- 减少 App 端的页面开发，复用具有业务含义的页面组件
- 开发模式升级，前后端分离，特殊业务场景下前端页面开发不需要服务端开发接口支持

先看看收益

2 年 50 张报表

4 个月 200 张报表

大表哥报表系统

4 人日单页对接

3 人日单页对接

大舅子数据聚合服务

先看看收益

前端

页面中数据控制权

数据含义提前介入

业务流程更加了解

后端

从页面 API 中解放

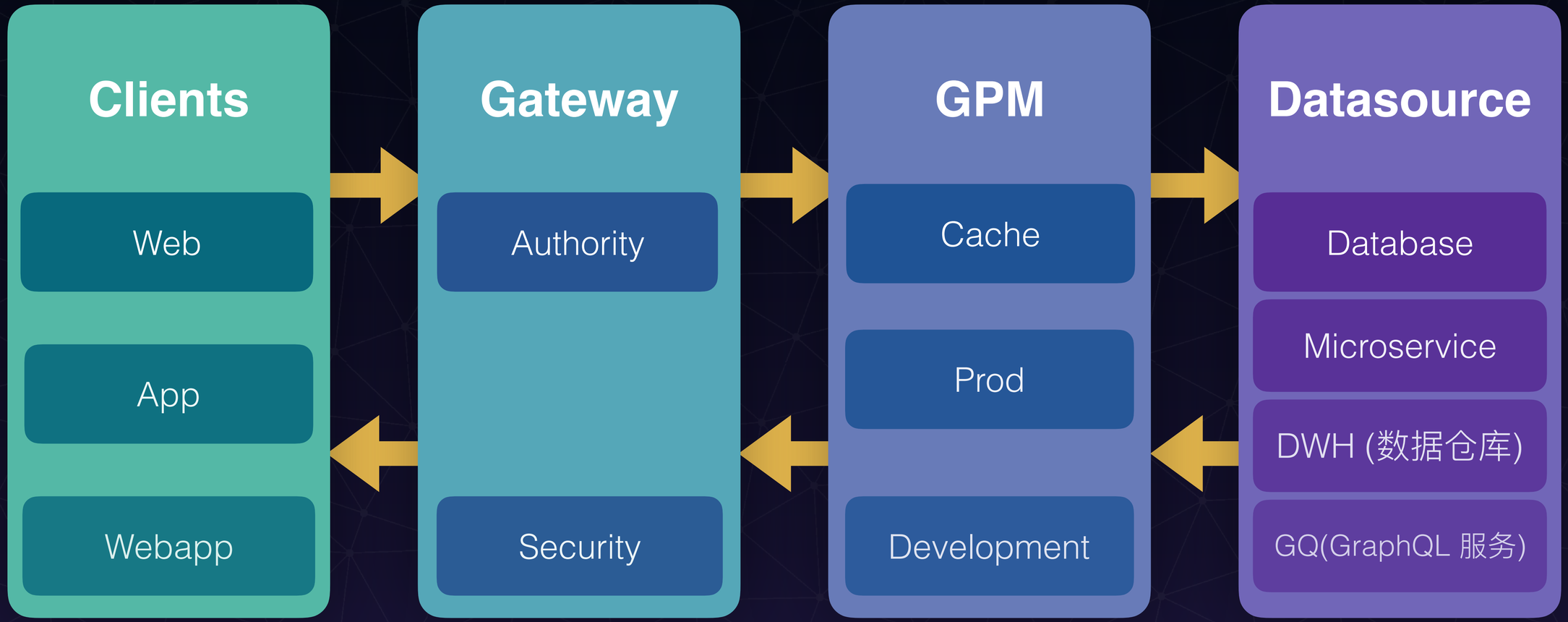
更通用的领域抽象

更稳定的数据服务

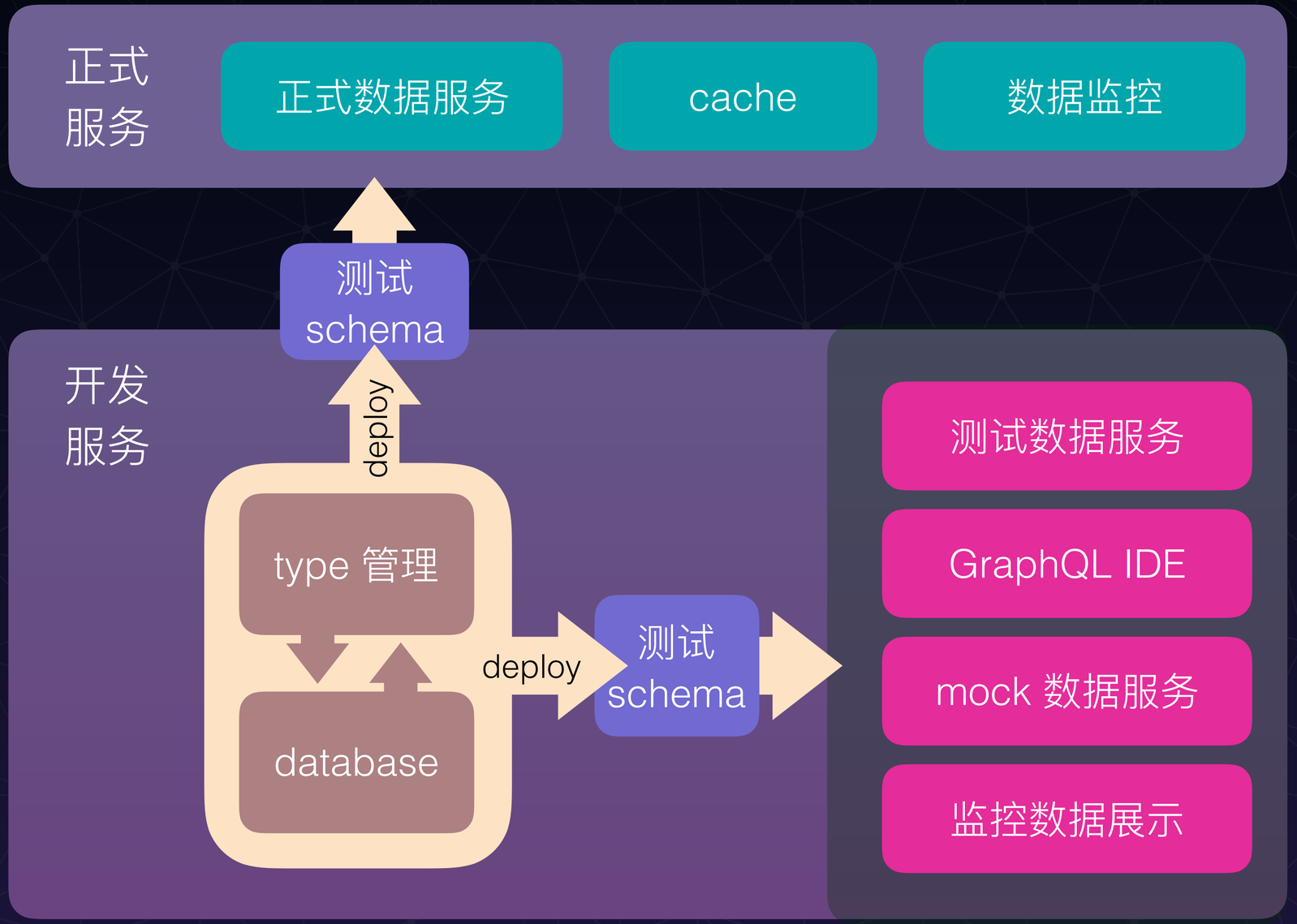
CONTENT

- 数据聚合服务架构
- GraphQL 有哪些便捷特点
- GraphQL 在宋小菜的实践 - 大舅子演示
- 安全、性能和数据收集等方面的实践建议

宋小菜数据聚合服务 GPm 架构关系



GPM
内部架构


正式
服务
正式数据服务
cache
数据监控
测试
schema
开发
服务
deploy
type 管理
database
deploy
测试
schema
测试数据服务
GraphQL IDE
mock 数据服务
监控数据展示

GraphQL = Graph Query Language = A query language for your API

使用 GraphQL 能带来哪些便捷

单一入口

便捷文档

避免
数据冗余

数据聚合

方便数据
MOCK

1

单一入口

RESTful API

- 多端点
- 版本管理

/v2/user
/v2/user/1
/v2/user/
/v1/article
/v2/article/comments
/v1/item
/v2/item/2
/v1/order
/v1/order/detail
...

/graphql

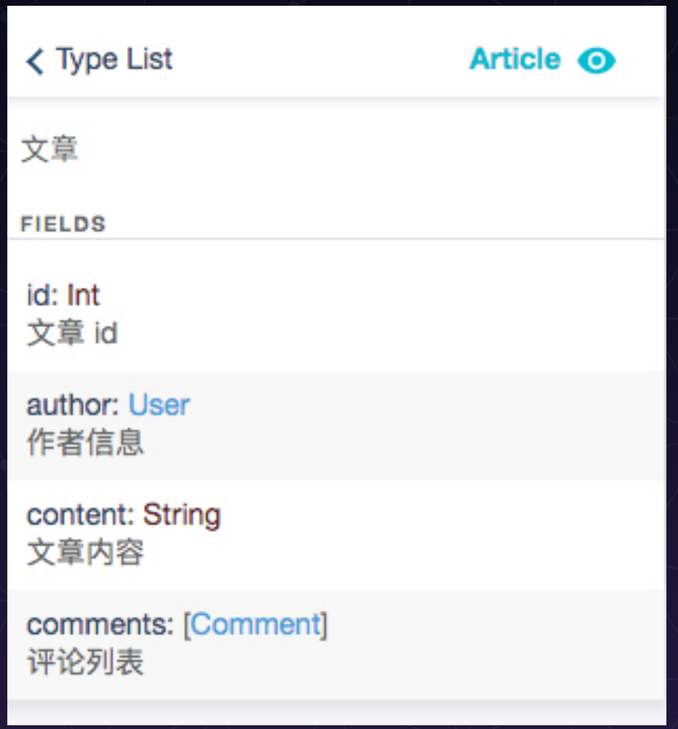
2

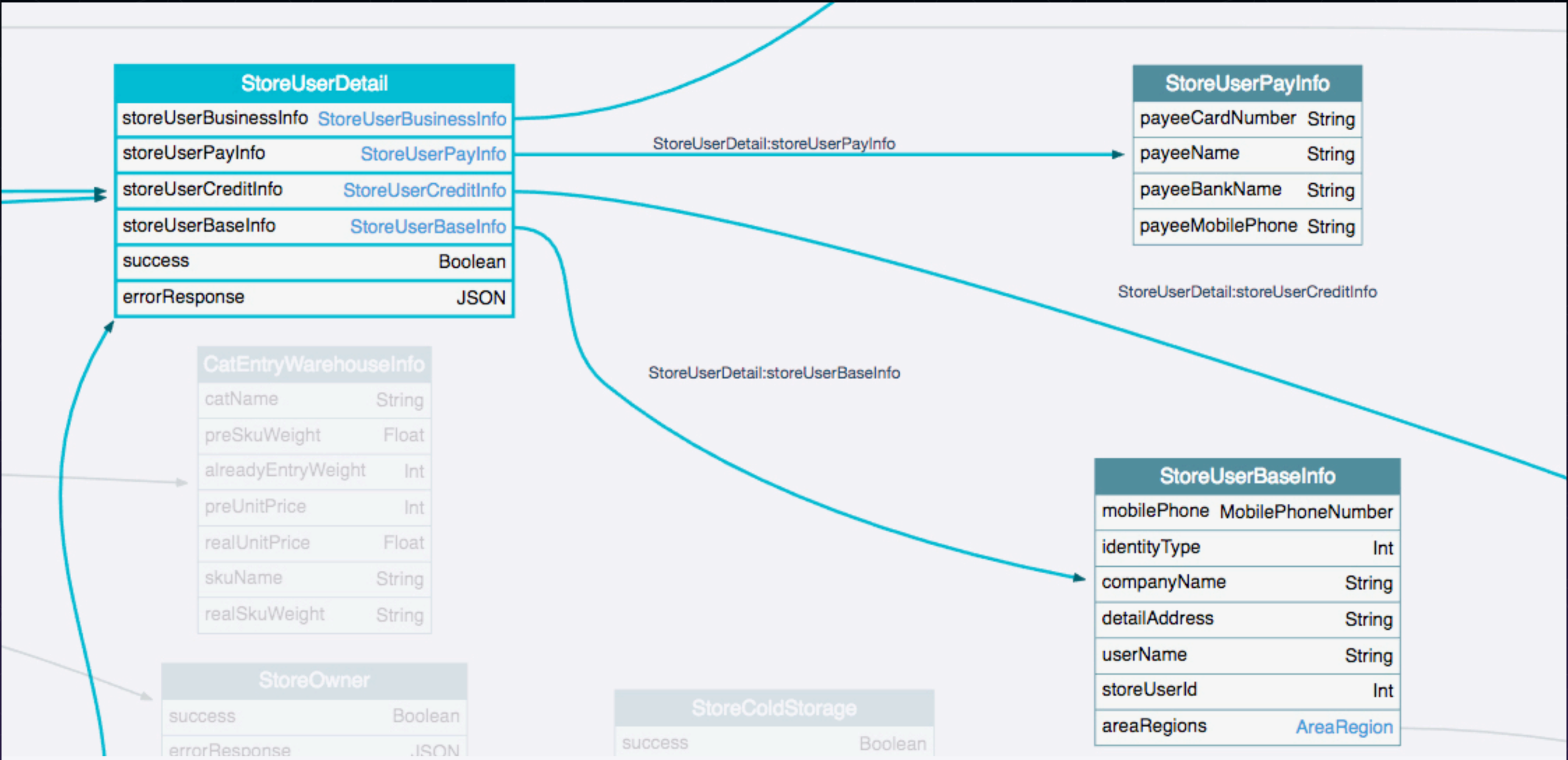
文档

RESTful API 文档管理

- 有学习成本
- 操作复杂
- API 与文档的同步问题

```graphql
# 文章
type Article @cache(expire: 180) {
  # 文章 id
  id: Int
  # 作者信息
  author: User
  # 文章内容
  content: String
  # 评论列表
  comments: [Comment]
}
```





github graphQL 传送门 →

模型关系更直观形象

3

避免数据冗余

多余的数据字段

```
{
  "status": 40,
  "statusStr": "已打款",
  "auditReason": null,
  "applierName": "陈炉",
  "appliedTime": "15283",
  "loanRate": 12,
}
```

RESTful

- 前端被动获取，选择是否使用
- 前端与后端约定，是否删除冗余字段，或者维护 “万能” 接口，增量维护
- 中间层筛选字段，返回给前端

GraphQL

前端自己决定返回数据的结构

```
{
  getArticle(id: 1){
    id
    content
  }
}
```

```
{
  "data": {
    "getArticle": {
      "id": 1,
      "content": "hello world!"
    }
  }
}
```

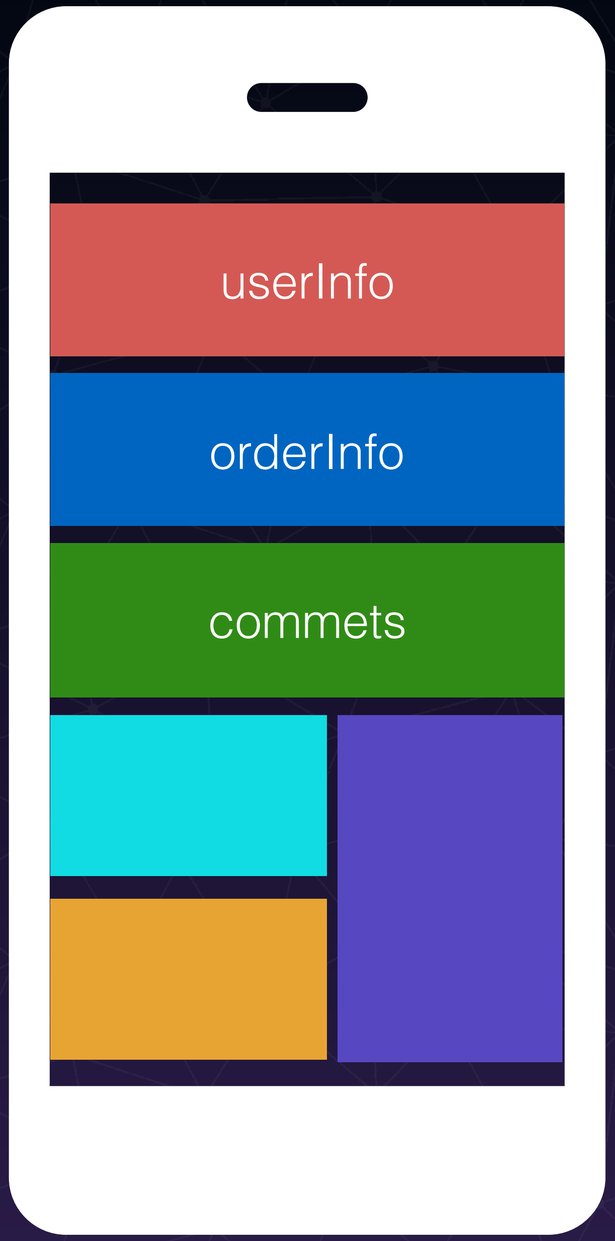
```
{
  getArticle(id: 1){
    id
    author {
      userId
      userName
    }
    content
  }
}
```

```
{
  "data": {
    "getArticle": {
      "id": 1,
      "author": {
        "userId": 1,
        "userName": "Caterina"
      },
      "content": "hello world!"
    }
  }
}
```

4

数据聚合

页面多数据源



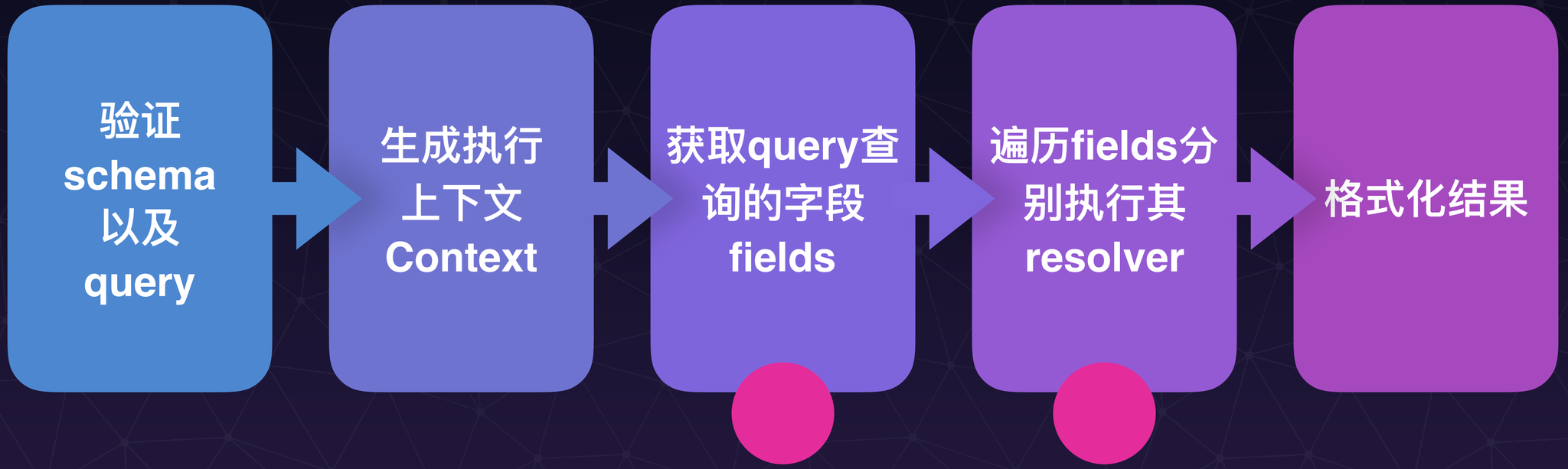
RESTful

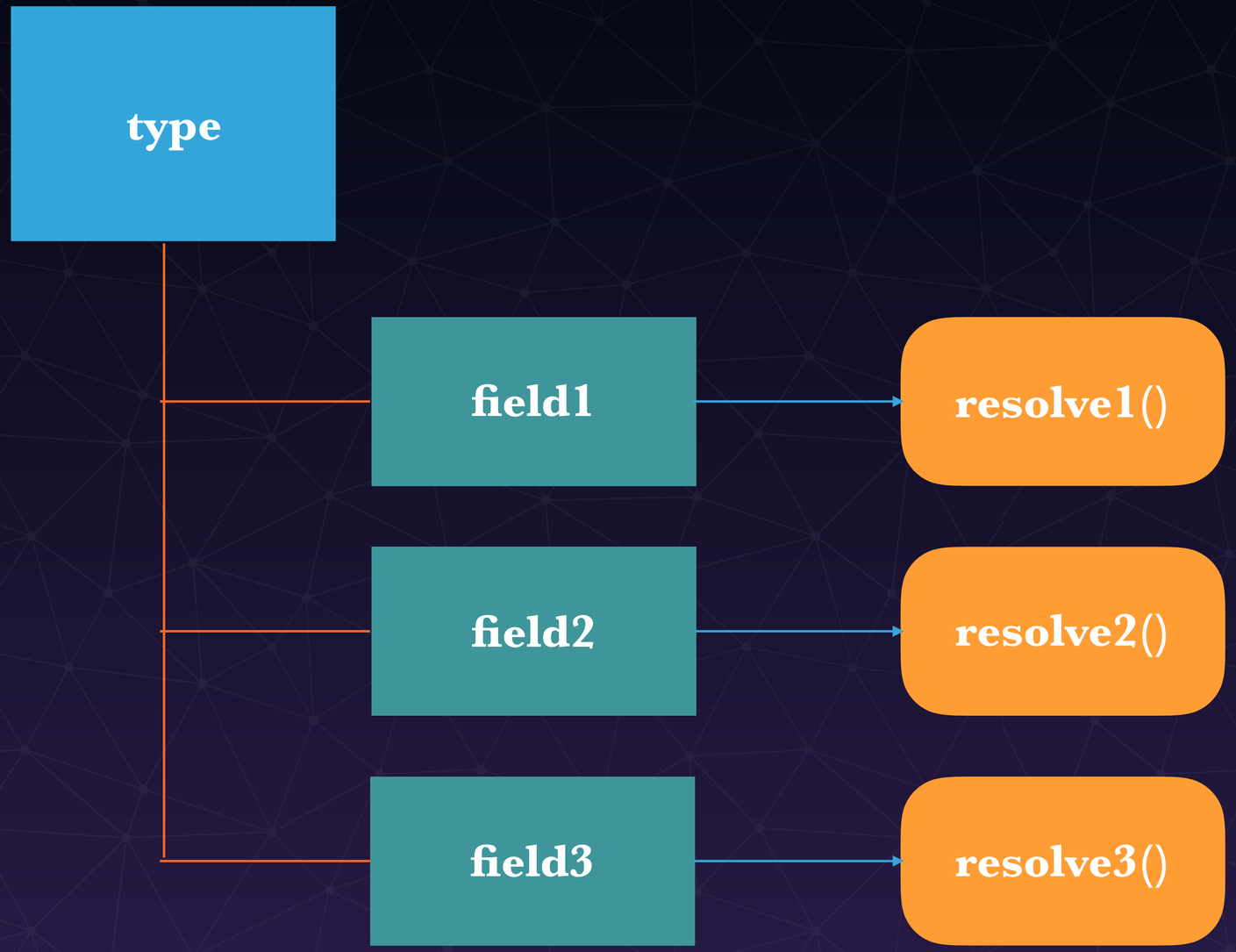
- 前端发起多请求获取数据
- 中间层拼装后端数据，然后一并返回
- 后端同学编写针对页面的 API 来拼接各个服务的数据，返回给前端

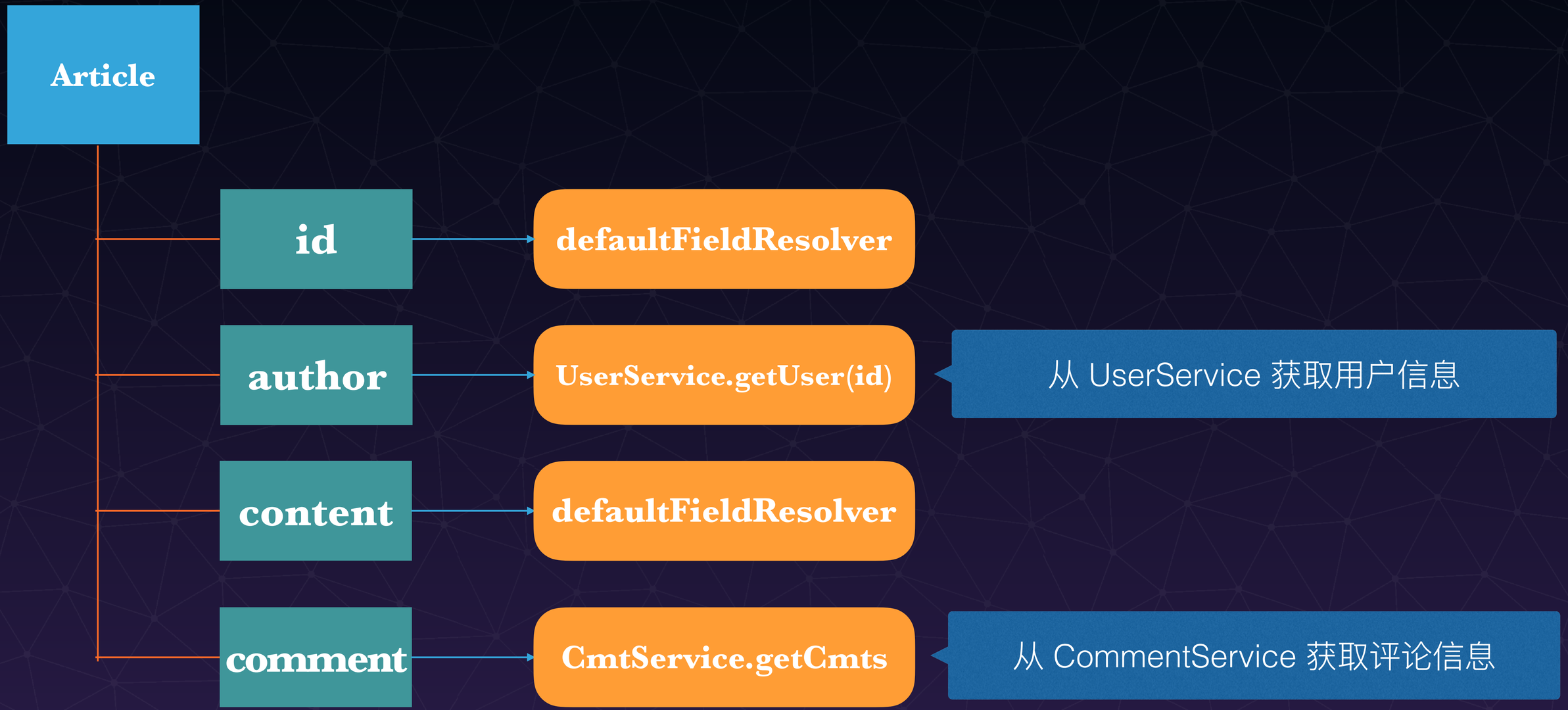
GraphQL

天生支持数据拼装

GraphQL Execute Process




type
field1
field2
field3
resolve1()
resolve2()
resolve3()


Article
id
defaultFieldResolver
author
UserService.getUser(id)
从 UserService 获取用户信息
content
defaultFieldResolver
comment
CmtService.getCmts
从 CommentService 获取评论信息

5

MOCK

使用 GQ 可以轻松实现 mock

- 标量类型(Int, Float，String 或者自定义标量类型
  如：手机号码，图片地址，身份证号码等) mock
- 普通类型 mock 数据自动生成
- 经典 mock 数据复用

GQ Mock 好处

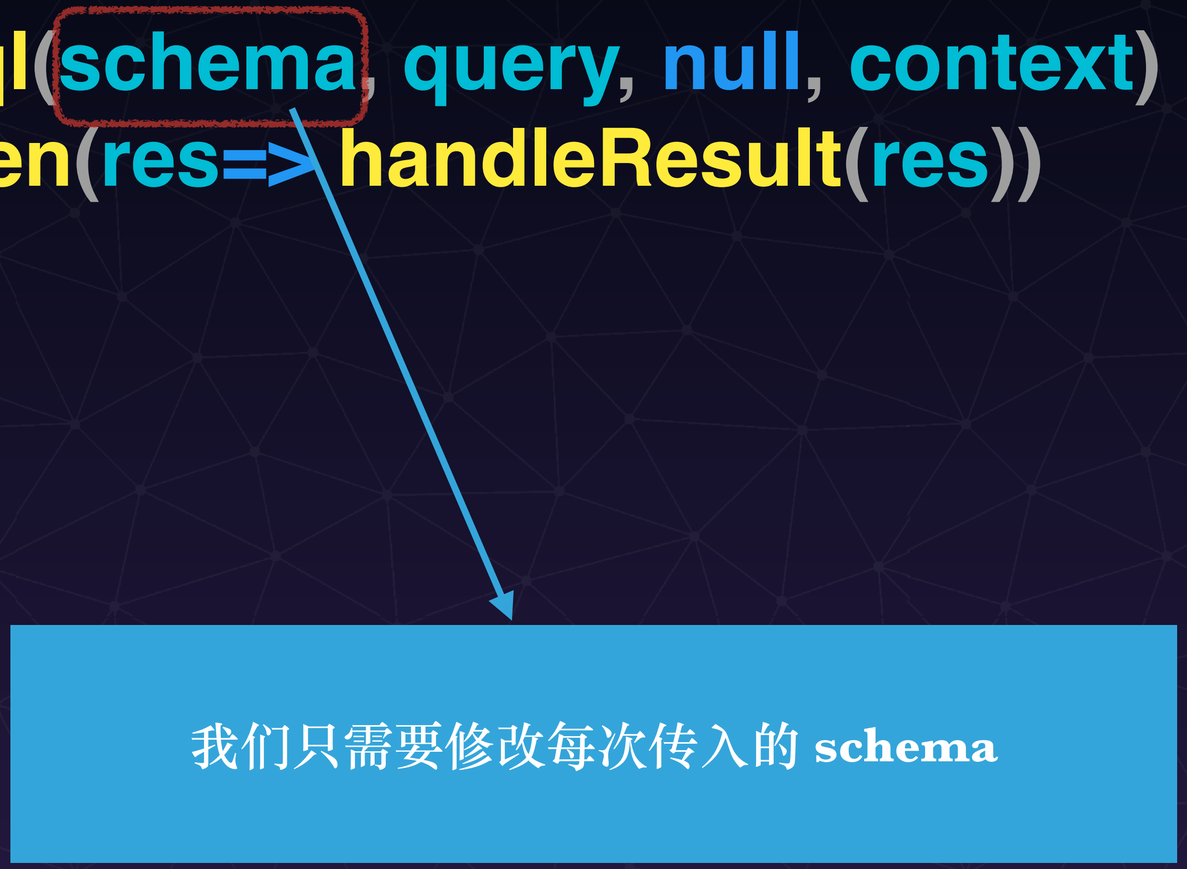
- mock 随 type 走，修改 type 时 mock 的数据类型也被同步修改，不会出现 mock 与 API 不同步的情况
- 很容易实现 mock 数据细粒度，基础数据类型自动 mock, 提高效率，便于开发调试
- 经典 mock 数据复用
- 前后端都可以进行 mock

演示

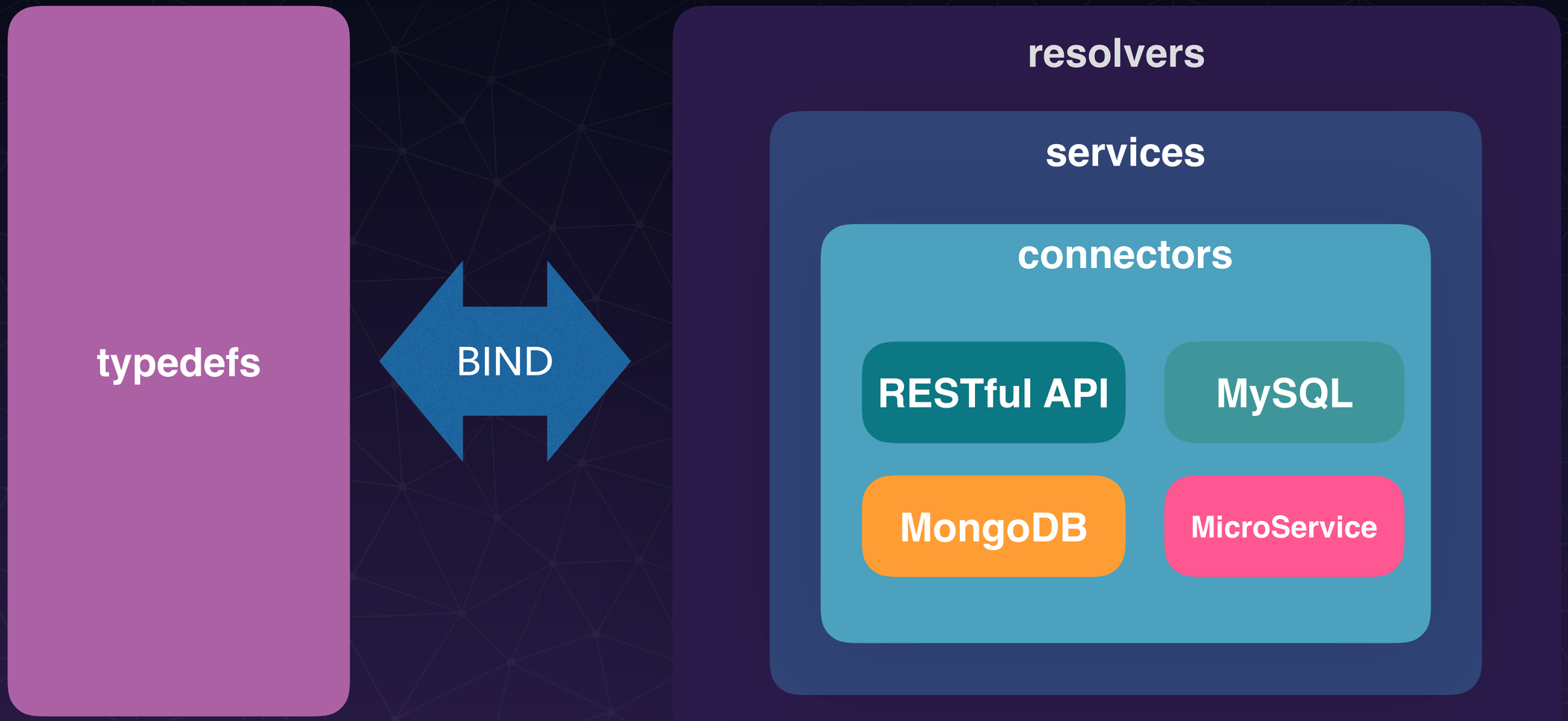
6

在线编辑应用 SCHEMA

```
graphql(schema, query, null, context)
  .then(res=> handleResult(res))
```



GraphQL Schema



typeDefs 本质是 String

关键在于如何动态生成 resolver

简化 resolver

```
fieldName(obj, args, context, info) { result }
```

resolver 函数签名

GPM 中的 resolver

```
fieldName(obj, args, context, info) {
  const param = mergeParams(obj, args)
  const result = context.service.XXService.getdata(param)
}
```

实现关键点

- 替换 schema
- 将 typeDefs 和 resolver 抽象为数据模型
- 充分利用 context, 简化 resolver，便于动态生成
- 参数提取

7

需要解决的问题

我们需要解决的问题

- 安全
- 慢查询
  - 终端做缓存（apollo、relay）
  - GQ 服务做缓存 (apollo-engine,合理利用 directives 和 resolver)
  - 批量处理查询 (dataloader)
  - …
- 其他

8

加分项

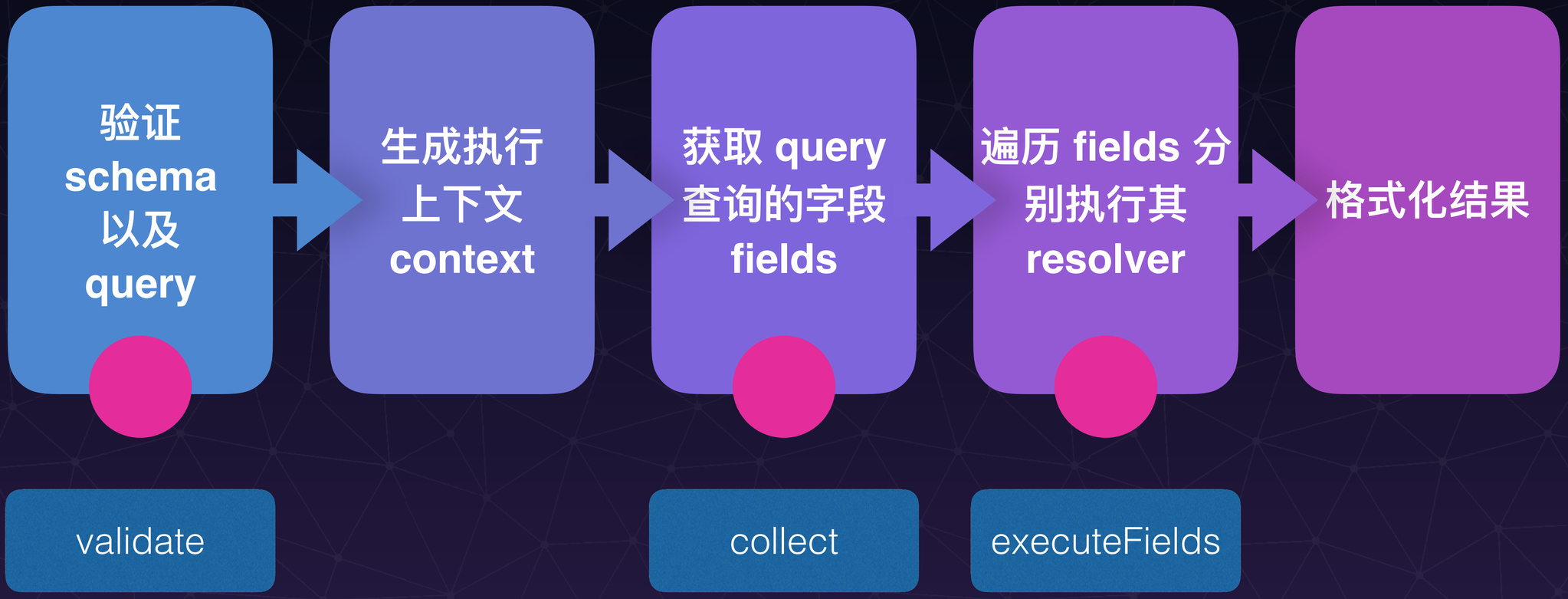
数据搜集

- 参照 graphql-extensions 编写自己的 extension 来追踪执行情况
- 使用第三方工具, 如 apollo

9

脑洞

GraphQL execute process



GraphQL 规范传送门

优化空间

- 相同的 query 是不是每次都需要 Validate ？
- 既然是相同的 query，它们的 fields 是否也不必重复 collect？
- executeFields 是否还有优化空间？

10

总结

单一入口

- 单端点入口方便前端工程管理
- 避免后端繁琐的 API 版本管理

文档

- 无需单独维护文档，Type 即文档
- 文档和类型能做到实时同步
- 倒逼前端贴近业务

数据冗余

- 降低前后端数据字段约定成本
- 后端从面向页面的 API 设计中解放
- 前端自由获取自己所需数据

数据聚合

- 天生的数据聚合支持
- 不同类型数据源的数据拼装

MOCK

- Mock 数据细粒度
- 前端可适当自产自销
- 一定程度解决同步问题
- 倒逼前端贴近业务

动态编辑

- 实时部署，敏捷开发

Thank you !