Искусство тестирования GraphQL

Константин Жуков

бизнес вконтакте

ВКонтакте для бизнеса — ваш гид в мир бизнеса в крупнейшей российской социальной сети

Стек

- Java
- Kotlin
- Swift
- Selenide
- Espresso
- Gradle
- TeamCity
- Allure
- TestOps
- Xcode
- XCUI Test
- Android Studio
- Charles
- Postman
- Swagger
- TestNG
- AssertJ
- IntelliJ IDEA

Сообщество ВКонтакте: основы работы

План доклада

Цель — дать набор знаний, достаточный для тестирования GraphQL на стенде

1. Подходы в построении бекенда, особенности тестирования GraphQL
2. Инструменты и стенды для тестирования GraphQL

Виды проектирования backend'а

REST

gRPC

SOAP

GraphQL

Особенности REST

REST — ресурсно-ориентированный стиль архитектуры, основанный на работе протокола HTTP

Плюсы REST

- Простота и легкость в понимании
- Масштабируемость
- Гибкость и независимость от платформы
- Кэширование

Минусы REST

- Ограниченность операций
- Передача избыточных данных
- Недостаточная стандартизация
- Отсутствие полной поддержки RPC

Особенности gRPC

gRPC — технология разработки удаленного вызова процедур на основе HTTP/2 в сочетании с бинарной сериализацией (обычно Protobuf)

Плюсы gRPC

- Высокая производительность
- Строгая типизация и схемы данных
- Многоплатформенная поддержка
- Поддержка строгих контрактов и обновлений

Минусы gRPC

- Сложность в настройке
- Ограничения сетевых прокси
- Ограничения при работе с большими данными
- Отсутствие широкой поддержки сторонних инструментов

Особенности SOAP

SOAP — это протокол обмена структурированными сообщениями на основе XML

Плюсы SOAP

- Стандартизация
- Интеграция
- Безопасность
- Обработка ошибок

Минусы SOAP

- Сложность
- Использование ресурсов
- Медленная разработка
- Неудобочитаемость

Особенности GraphQL

GraphQL - это язык запросов и среда выполнения запросов для API, разработанная Facebook

Плюсы GraphQL

- Гибкость запросов
- Единая точка входа
- Гибкая эволюция API
- Инструменты разработчика

Минусы GraphQL

- Усложнение сервера
- Избыточность и нагрузка на сервер
- Кэширование
- Отсутствие стандарта

Выводы

Выбор конкретного подхода зависит от требований и контекста проекта

- REST является широко распространенным и простым для использования
- gRPC обеспечивает высокую производительность и эффективность
- SOAP предоставляет стандартизацию и совместимость
- GraphQL позволяет гибко управлять запросами клиента

О GraphQL замолвите слово

Историческая справка по GraphQL

2012

GraphQL был разработан Facebook и начал использоваться для удовлетворения возрастающих потребностей в клиент-серверном взаимодействии на сайте Facebook

2015

Официальное анонсирование GraphQL произошло, когда Facebook опубликовал его как открытую спецификацию

Сегодня

Со временем GraphQL стал популярным инструментом и находит применение не только в Facebook

Проблемы, решаемые GraphQL

- Проблема избыточности данных (Overfetching)
- Проблема недостатка данных (Underfetching)
- Проблема версионирования API
- Проблема недостатка гибкости в данных
- Проблема недостатка документации

Особенности тестирования GraphQL

- Достоверность ответов
- Структура ответа
- Обработка ошибок
- Производительность
- Безопасность

Какие основные виды операций есть?

HTTP

Create	POST или PUT
Read	GET
Update	PUT или PATCH
Delete	DELETE

Какие основные виды операций есть?

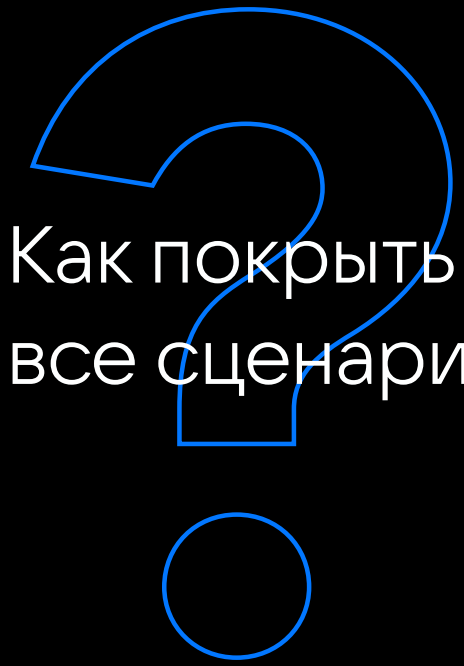
GraphQL

CRUD	HTTP	GQL request
Create	POST	Mutation
Read	POST	Query
Update	POST	Mutation
Delete	POST	Mutation

Тестирование подписок

```
1   subscription listenLikes {
2       listenLikes {
3           fname
4           likes
5       }
6   }
```

Приоритетность общих сценариев



Как покрыть все сценарии?

Полей же может быть бесконечно много!

Тестирование валидности запросов

Допустим у вас есть запрос для получения информации о пользователе.
Такой запрос будет выглядеть вот так:

QUERY

```
query GetUser ($userID: ID!)
{
    user (id: $userID)  {
        Id
        name
        mail
    }
}
```

GraphQL VARIABLES

```
{
    "userID" : 123
}
```

Обработка ошибок в ответе

```
{
 data: {}, // возврат данных
 errors: [...], // массив для ошибок
 extensions: {}, // объект для пользовательских данных
}
```

Тестирование производительности

QUERY

```
query GetHardRequest ($trackID: ID!) {
  track(id: $trackID) {
    artist {
      album {
        tracks {
          label {
            artists {
              id
            }
          }
        }
      }
    }
  }
}
```

GraphQL VARIABLES

```
{
    "trackID" : 123
}
```

Тестирование авторизации и доступа

Тестирование фрагментации запросов

QUERY

```
fragment UserInfo on User {
  name
  email
}

query GetUsers {
  user1: user(id: "1") {
    ...UserInfo
  }
  user2: user (id: "2") {
   ...UserInfo
  }
 }
```

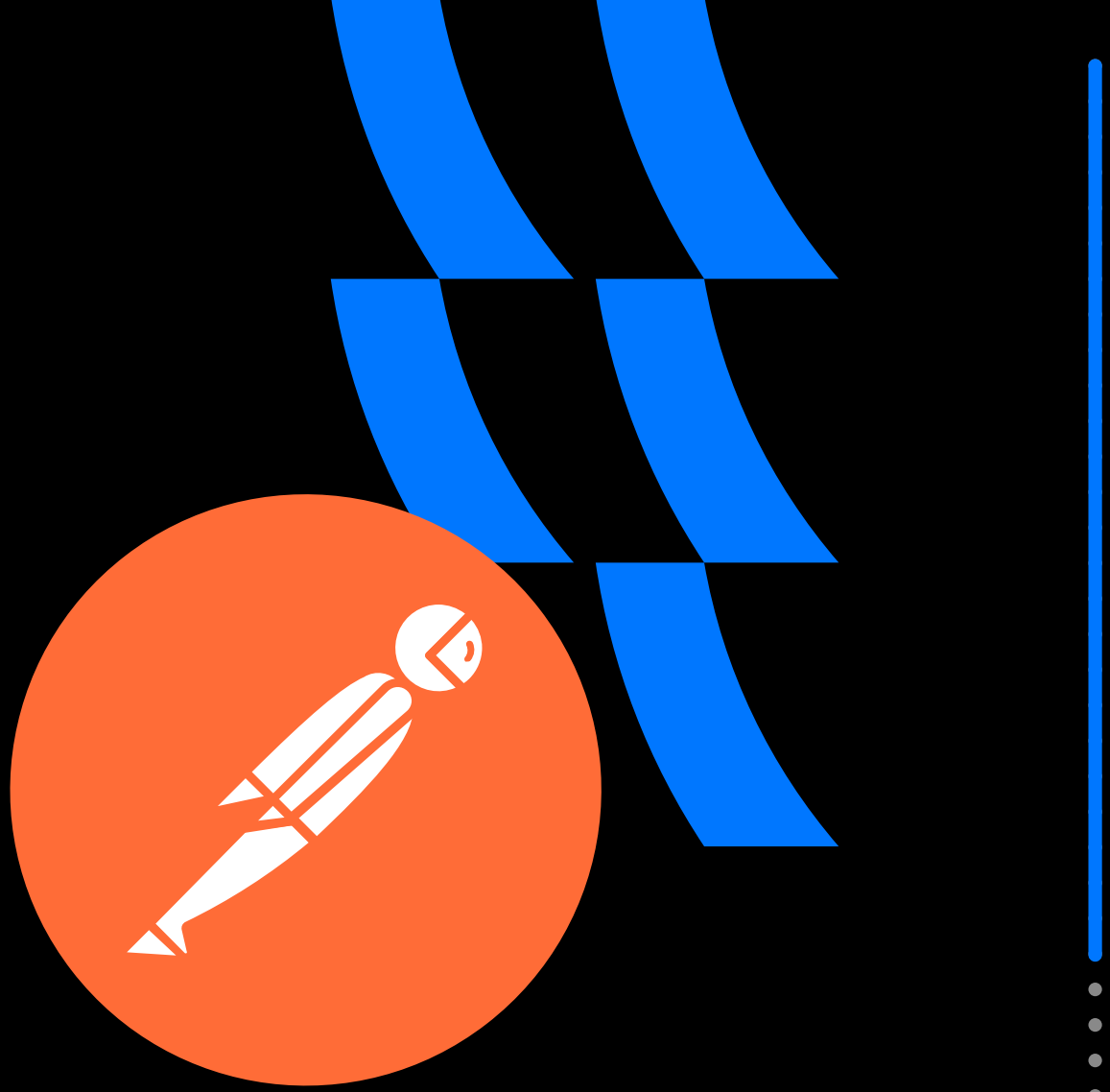
Заключение особенностей тестирования GraphQL

Инструменты для тестирования GraphQL

Сравнение инструментов для тестирования GraphQL

Postman

- Поддерживает GraphQL, REST и gRPC
- Удобный пользовательский интерфейс
- Возможность тестирования API без написания кода
- Поддержка автоматизации тестов с использованием сценариев Pre-request Scripts и Tests
- Встроенная поддержка переменных окружения



Insomnia

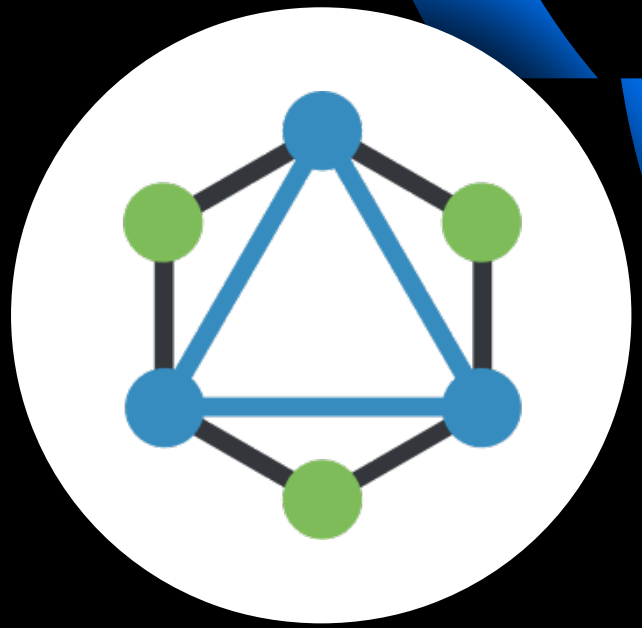
- Фокус на простоте и удобстве интерфейса
- Поддерживает GraphQL, REST и gRPC
- Возможность генерации кода для различных языков и фреймворков
- Поддержка переменных окружения и управление ими
- Плагины для расширения функциональности

GraphQL Playground

- Основан на GraphiQL с расширенным пользовательским интерфейсом
- Поддержка интерактивной документации (схемы)
- Возможности по работе с подписками
- История запросов и возможность их сохранения
- Встроенная поддержка HTTP-заголовков и переменных окружения

Altair GraphQL Client

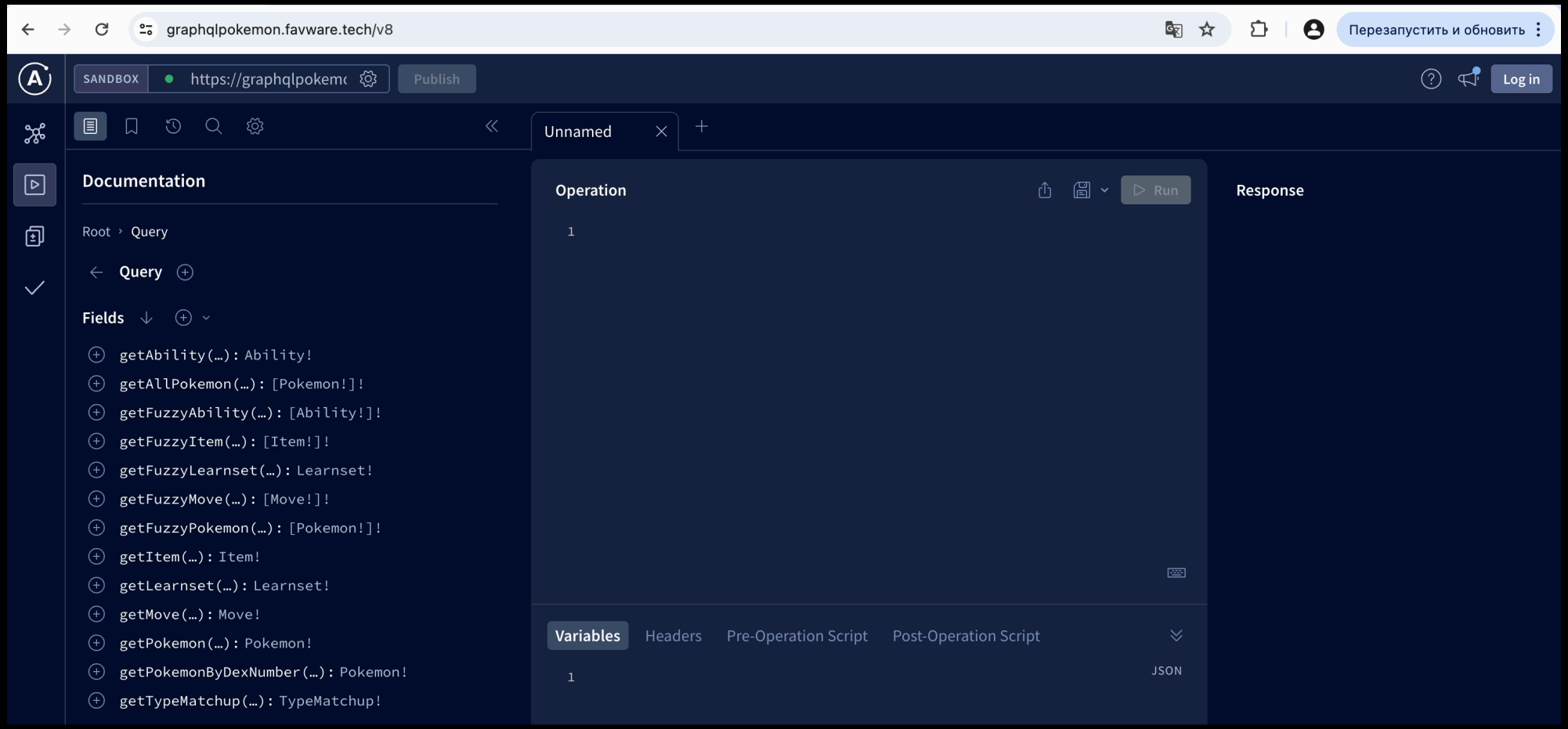
- Легко создавать и тестировать запросы
- Поддержка подписок
- Экспорт и импорт запросов
- Генерация документации из запросов
- Плагины и интеграция с другими инструментами



Итоговое сравнение инструментов для тестирования GraphQL

Инструмент	Поддержка запросов	Автоматизация	Доп. функции
Postman	REST, GraphQL, gRPC		Переменные окружения, сценарии
Insomnia	REST, GraphQL, gRPC		Переменные окружения, генерация кода
GraphQL Playground	GraphQL		Подписки, документация
Altair	GraphQL		Подписки, экспорт/импорт запросов

Практические стенды для тестирования GraphQL



Автоматизация

К применению GraphQL в контексте автоматизации не стоит относиться с опаской

В действительности, дело здесь сводится к использованию POST запроса с нужным body

Пример Kotlin

```kotlin
@Test
@TestOpsId(2211)
@DisplayName("Успешное получение информации по корзине пользователя")
fun getCartSuccessResponse() {
    val gqlQueryData = javaClass.classLoader.getResource(
        "graphQL/cart/getCart.query.graphql"
    )?.readText() ?: throw FileNotFoundException(
        "File with path graphQL/cart/getCart.query.graphql not found in Resources folder"
    )
    val response = apiRequest.post(
        headers = requestHeader(
            "Auth-Token" to user.token.id
        ),

        body = gqlQueryData.toRequestBody()
    )
    val responseJson = response.jsonBody()

    step("Проверяем что ответ приходит по треку что мы запросили") {
        assertThatJson(responseJson) {
            inPath("$.data.getCart[0].product.id").isString.isEqualTo(123)
        }
    }
```

Пример Python

```
query_carts = """
    {
      getCarts(){
          id,
          product {
              id,
              availability
          }
      }
    }
"""
def test_get_carts():
    response = requests.post("http://localhost:8888/graphql/", json={'query': query_carts})
    response_body = response.json()
    assert response_body['data']['product']['id'] == '123'
```

Выводы

GraphQL — это большое облегчение для команд разработчиков. Мы больше не заморачиваемся по поводу избыточности и недостатка данных. Мы просто получаем то, что нам нужно

Дэн Абрамов
разработчик в Facebook

GraphQL — просто фантастический! Он позволяет вам строить точно те API, которые вы хотите, а также получать все необходимые данные без избыточности

Ли Байрон
создатель Next.is

GraphQL привнес революцию в мир API. Он позволяет вам максимально удовлетворять потребности клиентов в данных, и я являюсь его большим сторонником

Эгон Ишматович
инженер по разработке в GitHub

Материалы

- https://ide.bitquery.io
- https://github.com/favware/graphql-pokemon?tab=readme-ov-file
- https://github.com/mazipan/graphql-pokeapi
- https://geographql.netlify.app
- https://graphql.org/swapi-graphql

Заключение

Telegram стикеры
VKMarketQA

Константин Жуков

VK

LinkedIn

Telegram