

Микросервисы и API

ХОСЕ АРО ПЕРАЛЬТА

Выпущено
при поддержке
КРОК

 **ПИТЕР®**

Санкт-Петербург • Москва • Минск

2024

Краткое содержание

Предисловие	16
Благодарности	18
О книге	20
Об авторе	25
Иллюстрация на обложке	26
О научных редакторах русскоязычного издания	27
От издательства	28

Часть I. Представляем API микросервисов

Глава 1. Что такое API микросервисов	31
Глава 2. Разработка простого API	50
Глава 3. Проектирование микросервисов	76

Часть II. Проектирование и разработка REST API

Глава 4. Принципы проектирования REST API	93
Глава 5. Документирование REST API с помощью OpenAPI	125
Глава 6. Создание REST API на Python	147
Глава 7. Паттерны реализации сервисов	183

Часть III. Проектирование и реализация GraphQL API

Глава 8. Проектирование GraphQL API	225
Глава 9. Использование GraphQL API	253
Глава 10. Создание GraphQL API с помощью Python	277

Часть IV. Защита, тестирование и развертывание API микросервисов

Глава 11. Авторизация и аутентификация API.	315
Глава 12. Тестирование и валидация API	352
Глава 13. Контейнеризация API микросервисов.	384
Глава 14. Развертывание API микросервисов с помощью Kubernetes	396

Приложения

Приложение А. Типы веб-API и протоколов	434
Приложение Б. Управление жизненным циклом API	446
Приложение В. Авторизация API с использованием поставщика удостоверений	451

Оглавление

Предисловие	16
Благодарности	18
О книге	20
Кому стоит прочитать эту книгу	20
Структура издания	20
О коде	23
Другие онлайн-ресурсы	24
Об авторе	25
Иллюстрация на обложке	26
О научных редакторах русскоязычного издания	27
От издательства	28

Часть I. Представляем API микросервисов

Глава 1. Что такое API микросервисов	31
1.1. Что такое микросервисы	32
1.1.1. Определение микросервисов	32
1.1.2. Микросервисы против монолитов	33
1.1.3. История появления современных микросервисов	36
1.2. Что такое веб-API	37
1.2.1. API	37
1.2.2. Веб-API	38
1.2.3. Как API помогают нам управлять интеграцией микросервисов	38

1.3. Проблемы микросервисной архитектуры	40
1.3.1. Оптимальная декомпозиция сервиса	41
1.3.2. Интеграционные тесты микросервисов.	41
1.3.3. Обработка недоступности сервиса.	42
1.3.4. Трассировка распределенных транзакций	43
1.3.5. Повышение сложности эксплуатации и накладных расходов на инфраструктуру	44
1.4. Введение в разработку через документирование (DDD)	45
1.5. Знакомство с приложением CoffeeMesh	47
1.6. Для кого эта книга и чему вы научитесь	47
Резюме.	48
Глава 2. Разработка простого API	50
2.1. Знакомимся со спецификацией API сервиса заказов	51
2.2. Высокоуровневая архитектура приложения заказов	52
2.3. Реализация эндпоинтов API.	53
2.4. Реализация моделей валидации данных с помощью pydantic	60
2.5. Валидация полезной нагрузки запроса с помощью pydantic.	65
2.6. Маршалинг и валидация полезной нагрузки ответа с помощью pydantic.	69
2.7. Добавление в API сохраненного в памяти списка заказов	73
Резюме.	75
Глава 3. Проектирование микросервисов	76
3.1. Представляем CoffeeMesh	77
3.2. Принципы проектирования микросервисов.	77
3.2.1. Принцип «База данных для каждого сервиса»	77
3.2.2. Принцип слабой связанности.	79
3.2.3. Принцип единственной ответственности	80
3.3. Декомпозиция сервисов по бизнес-функциям	80
3.3.1. Анализ бизнес-структуры CoffeeMesh	81
3.3.2. Декомпозиция микросервисов по функциям	81
3.4. Декомпозиция сервисов по поддоменам	84
3.4.1. Что такое предметно-ориентированное проектирование.	84
3.4.2. Применение стратегического анализа в CoffeeMesh.	85
3.5. Декомпозиция по бизнес-функциям в сравнении с декомпозицией по поддоменам.	89
Резюме.	90

Часть II. Проектирование и разработка REST API

Глава 4. Принципы проектирования REST API	93
4.1. Что такое REST.	94
4.2. Архитектурные ограничения приложений REST.	95
4.2.1. Разделение задач: принцип клиент-серверной архитектуры.	96
4.2.2. Добавляем масштабирование: принцип отсутствия записи состояния	97
4.2.3. Оптимизация производительности: принцип кэшируемости	97
4.2.4. Упрощение для клиента: принцип многоуровневой системы	98
4.2.5. Расширяемые интерфейсы: принцип «код по запросу»	99
4.2.6. Сохранение согласованности: принцип единства интерфейса	100
4.3. Гипермедиа как двигатель состояния приложений	100
4.4. Анализ зрелости API с помощью модели Ричардсона.	103
4.4.1. Уровень 0. Веб-API в стиле RPC.	104
4.4.2. Уровень 1. Введение понятия ресурса.	105
4.4.3. Уровень 2. Использование методов HTTP и статус-кодов.	106
4.4.4. Уровень 3. Возможность обнаружения API	106
4.5. Структурированные URL-адреса ресурсов с методами HTTP	107
4.6. Использование статус-кодов HTTP для создания информативных HTTP-ответов	111
4.6.1. Что такое статус-коды HTTP	111
4.6.2. Использование статус-кодов HTTP для сообщения об ошибках клиента в запросе	112
4.6.3. Использование статус-кодов HTTP для сообщения об ошибках на сервере	116
4.7. Проектирование полезной нагрузки API.	117
4.7.1. Что такое полезная нагрузка HTTP и когда мы ее используем	118
4.7.2. Модели проектирования полезной нагрузки HTTP.	119
4.8. Проектирование параметров запроса URL.	123
Резюме.	124
Глава 5. Документирование REST API с помощью OpenAPI.	125
5.1. Использование JSON Schema для моделирования данных.	126
5.2. Структура спецификации OpenAPI.	130
5.3. Документирование эндпоинтов API.	132
5.4. Документирование параметров запроса URL.	133
5.5. Документирование полезной нагрузки запросов.	134
5.6. Рефакторинг определений схем во избежание повторов	136

5.7. Документирование ответов API	139
5.8. Создание общих ответов	142
5.9. Определение схемы аутентификации API	144
Резюме.	146
Глава 6. Разработка REST API на Python	147
6.1. Обзор API сервиса заказов.	148
6.2. Параметры URL-запроса для API сервиса заказов.	149
6.3. Валидация полезной нагрузки с неизвестными полями	152
6.4. Переопределение динамически генерируемой спецификации FastAPI.	156
6.5. Обзор API сервиса кухни.	157
6.6. Вводим в работу flask-smorest	159
6.7. Инициализация веб-приложения для API	161
6.8. Реализация эндпоинтов API.	163
6.9. Реализация моделей проверки полезной нагрузки с помощью marshmallow	167
6.10. Проверка параметров запроса URL	171
6.11. Проверка данных перед сериализацией ответа	174
6.12. Реализация списка расписаний в памяти	178
6.13. Переопределение динамически генерируемой спецификации API flask-smorest.	180
Резюме.	181
Глава 7. Паттерны реализации сервисов.	183
7.1. Гексагональные архитектуры для микросервисов	184
7.2. Настройка среды и структуры проекта	187
7.3. Реализация моделей баз данных	189
7.4. Реализация паттерна Репозиторий для доступа к данным	195
7.4.1. Паттерн Репозиторий: что это и чем он полезен	195
7.4.2. Реализация паттерна Репозиторий	197
7.5. Реализация уровня бизнес-логики.	202
7.6. Внедрение паттерна Единица работы	212
7.7. Интеграция уровня API и уровня сервиса	217
Резюме.	221
 Часть III. Проектирование и реализация GraphQL API	
Глава 8. Проектирование GraphQL API	225
8.1. Знакомство с GraphQL	226
8.2. Представляем API сервиса продукции	229

8.3. Знакомство с системой типов GraphQL	232
8.3.1. Создание определений свойств с помощью скалярных величин	232
8.3.2. Моделирование ресурсов с использованием объектных типов	233
8.3.3. Создание кастомных скалярных величин	235
8.4. Представление коллекций элементов с помощью списков	236
8.5. Мыслите графически: выстраивание значимых связей между объектными типами	237
8.5.1. Соединение типов с помощью свойств-ребер	237
8.5.2. Создание соединений со сквозными типами	239
8.6. Сочетание различных типов с помощью объединений и интерфейсов	241
8.7. Ограничение значений свойств с помощью перечислений	244
8.8. Определение запросов для получения данных из API	245
8.9. Изменение состояния сервера с помощью мутаций	248
Резюме	251
Глава 9. Использование GraphQL API	253
9.1. Запуск mock-сервера GraphQL	254
9.2. Запросы GraphQL	256
9.2.1. Выполнение простых запросов	256
9.2.2. Выполнение запросов с параметрами	257
9.2.3. Ошибки запросов	258
9.3. Использование фрагментов в запросах	260
9.4. Выполнение запросов с входными параметрами	262
9.5. Навигация по графу API	262
9.6. Выполнение нескольких запросов и алиасинг	264
9.6.1. Выполнение нескольких запросов за один раз	264
9.6.2. Алиасинг запросов	265
9.7. Выполнение мутаций GraphQL	268
9.8. Выполнение параметризованных запросов и мутаций	269
9.9. Раскрываем тайны запросов GraphQL	273
9.10. Вызов GraphQL API с помощью кода на Python	274
Резюме	276
Глава 10. Создание GraphQL API с помощью Python	277
10.1. Анализ требований к API	278
10.2. Технологический стек	278
10.3. Фреймворк Ariadne	280
10.4. Реализация API сервиса продукции	286
10.4.1. Разработка структуры проекта	286

10.4.2. Создание точки входа для сервера GraphQL	287
10.4.3. Реализация резольверов запросов	288
10.4.4. Реализация резольверов типов	291
10.4.5. Обработка параметров запроса	298
10.4.6. Реализация резольверов мутаций	301
10.4.7. Создание резольверов для кастомных скалярных типов	304
10.4.8. Реализация резольверов полей	308
Резюме	311

Часть IV. Защита, тестирование и развертывание API микросервисов

Глава 11. Авторизация и аутентификация API	315
11.1. Настройка среды для примеров этой главы	316
11.2. Протоколы аутентификации и авторизации	317
11.2.1. Что такое Open Authorization	317
11.2.2. Что такое OpenID Connect	323
11.3. Работа с JSON Web Tokens	325
11.3.1. Заголовок JWT	327
11.3.2. Утверждения JWT	328
11.3.3. Создание JWT	330
11.3.4. Проверка JWT	332
11.3.5. Валидация JWT	334
11.4. Добавление авторизации на API сервера	335
11.4.1. Создание модуля авторизации	337
11.4.2. Создание промежуточного программного обеспечения для авторизации	338
11.4.3. Добавление промежуточного программного обеспечения CORS	341
11.5. Авторизация доступа к ресурсам	343
11.5.1. Обновление базы данных для привязки пользователей и заказов	343
11.5.2. Ограничение доступа пользователей к их собственным ресурсам	346
Резюме	350
Глава 12. Тестирование и валидация API	352
12.1. Настройка среды для тестирования API	353
12.2. Тестирование REST API с помощью Dredd	354
12.2.1. Что такое Dredd	355
12.2.2. Установка и запуск набора тестов Dredd по умолчанию	356
12.2.3. Настройка набора тестов Dredd с помощью хуков	358
12.2.4. Использование Dredd для тестирования API	366

12.3. Введение в тестирование на основе свойств	366
12.3.1. Что такое тестирование на основе свойств	366
12.3.2. Традиционный подход к тестированию API	367
12.3.3. Тестирование на основе свойств с использованием Hypothesis	369
12.3.4. Использование Hypothesis для тестирования эндпоинта REST API	371
12.4. Тестирование REST API с помощью Schemathesis	374
12.4.1. Запуск набора тестов Schemathesis по умолчанию	375
12.4.2. Использование ссылок для расширения набора тестов Schemathesis	375
12.5. Тестирование GraphQL API	379
12.5.1. Тестирование GraphQL API с помощью Schemathesis	380
12.6. Разработка стратегии тестирования API	382
Резюме	382
Глава 13. Контейнеризация API микросервисов	384
13.1. Настройка среды для этой главы	385
13.2. Контейнеризация микросервиса	386
13.3. Запуск приложений с помощью Docker Compose	391
13.4. Публикация Docker-образов в реестре контейнеров	393
Резюме	395
Глава 14. Развертывание API микросервисов с помощью Kubernetes	396
14.1. Настройка среды для этой главы	398
14.2. Как работает Kubernetes: версия CliffsNotes	399
14.3. Создание кластера Kubernetes с помощью EKS	401
14.4. Использование ролей IAM для учетных записей сервисов Kubernetes	405
14.5. Развертывание балансировщика нагрузки Kubernetes	406
14.6. Развертывание микросервисов в кластере Kubernetes	409
14.6.1. Создание объекта развертывания	410
14.6.2. Создание объекта сервиса	413
14.6.3. Предоставление доступа к сервисам с помощью ingress-объектов	415
14.7. Настройка бессерверной базы данных с помощью AWS Aurora	417
14.7.1. Создание бессерверной базы данных Aurora	417
14.7.2. Управление секретами в Kubernetes	420
14.7.3. Запуск миграции базы данных и подключение сервиса к базе данных	424
14.8. Обновление спецификации OpenAPI с указанием имени хоста ALB	427
14.9. Удаление кластера Kubernetes	430
Резюме	432

Приложения

Приложение А. Типы веб-API и протоколов	434
А.1. Рассвет API: RPC, XML-RPC и JSON-RPC.	435
А.2. SOAP и появление стандартов API	437
А.3. RPC снова наносит удар: быстрый обмен данными через gRPC	439
А.4. API HTTP и REST	441
А.5. Детализированные запросы с использованием GraphQL	443
Приложение Б. Управление жизненным циклом API	446
Б.1. Стратегии управления версиями для развивающихся API	447
Б.2. Вывод API из эксплуатации.	449
Приложение В. Авторизация API с использованием поставщика удостоверений	451
В.1. Использование поставщика удостоверений как услуги	452
В.2. Использование схемы авторизации PKCE	458
В.3. Использование схемы учетных данных клиента	459
В.4. Авторизация запросов в пользовательском интерфейсе Swagger	461