

Крис Ричардсон

Микросервисы

**Паттерны разработки
и рефакторинга**



Санкт-Петербург • Москва • Минск

2025

Краткое содержание

Предисловие.....	16
Благодарности.....	19
О книге.....	21
Глава 1. Побег из монолитного ада	26
Глава 2. Стратегии декомпозиции	63
Глава 3. Межпроцессное взаимодействие в микросервисной архитектуре.....	97
Глава 4. Управление транзакциями с помощью повествований	146
Глава 5. Проектирование бизнес-логики в микросервисной архитектуре.....	185
Глава 6. Разработка бизнес-логики с порождением событий	223
Глава 7. Реализация запросов в микросервисной архитектуре	264
Глава 8. Шаблоны внешних API.....	301
Глава 9. Тестирование микросервисов, часть 1	343
Глава 10. Тестирование микросервисов, часть 2	373
Глава 11. Разработка сервисов, готовых к промышленному использованию.....	405
Глава 12. Развертывание микросервисов	446
Глава 13. Процесс перехода на микросервисы	495

Оглавление

Предисловие.....	16
Благодарности.....	19
О книге.....	21
Кому следует прочитать эту книгу	21
Структура издания.....	21
О коде	23
Онлайн-ресурсы	23
Об авторе.....	23
Об иллюстрации на обложке	24
От издательства	25
Глава 1. Побег из монолитного ада	26
1.1. Медленным шагом в монолитный ад	27
1.1.1. Архитектура приложения FTGO.....	28
1.1.2. Преимущества монолитной архитектуры.....	29
1.1.3. Жизнь в монолитном аду	29
1.2. Почему эта книга актуальна для вас	32

1.3. Чему вы научитесь, прочитав эту книгу.....	33
1.4. Микросервисная архитектура спешит на помощь	34
1.4.1. Куб масштабирования и микросервисы	34
1.4.2. Микросервисы как разновидность модульности	37
1.4.3. У каждого сервиса есть своя база данных.....	38
1.4.4. Микросервисная архитектура для FTGO	38
1.4.5. Сравнение микросервисной и сервис-ориентированной архитектур	40
1.5. Достоинства и недостатки микросервисной архитектуры.....	41
1.5.1. Достоинства микросервисной архитектуры	41
1.5.2. Недостатки микросервисной архитектуры	44
1.6. Язык шаблонов микросервисной архитектуры	46
1.6.1. Микросервисная архитектура не панацея	47
1.6.2. Шаблоны проектирования и языки шаблонов	48
1.6.3. Обзор языка шаблонов микросервисной архитектуры	51
1.7. Помимо микросервисов: процесс и организация.....	58
1.7.1. Организация разработки и доставки программного обеспечения	59
1.7.2. Процесс разработки и доставки программного обеспечения	60
1.7.3. Человеческий фактор при переходе на микросервисы	61
Резюме	62
Глава 2. Стратегии декомпозиции	63
2.1. Что представляет собой микросервисная архитектура.....	64
2.1.1. Что такое архитектура программного обеспечения и почему она важна	64
2.1.2. Обзор архитектурных стилей	67
2.1.3. Микросервисная архитектура как архитектурный стиль	70
2.2. Определение микросервисной архитектуры приложения.....	74
2.2.1. Определение системных операций.....	76
2.2.2. Разбиение на сервисы по бизнес-возможностям	82
2.2.3. Разбиение на сервисы по проблемным областям	85
2.2.4. Методические рекомендации по декомпозиции.....	87
2.2.5. Трудности при разбиении приложения на сервисы	88
2.2.6. Определение API сервисов	92
Резюме	95

Глава 3. Межпроцессное взаимодействие в микросервисной архитектуре	97
3.1. Обзор межпроцессного взаимодействия в микросервисной архитектуре	98
3.1.1. Стили взаимодействия	99
3.1.2. Описание API в микросервисной архитектуре	100
3.1.3. Развивающиеся API	101
3.1.4. Форматы сообщений	103
3.2. Взаимодействие на основе удаленного вызова процедур	105
3.2.1. Использование REST	106
3.2.2. Использование gRPC	109
3.2.3. Работа в условиях частичного отказа с применением шаблона «Предохранитель»	111
3.2.4. Обнаружение сервисов	114
3.3. Взаимодействие с помощью асинхронного обмена сообщениями	119
3.3.1. Обзор механизмов обмена сообщениями	119
3.3.2. Реализация стилей взаимодействия с помощью сообщений	122
3.3.3. Создание спецификации для API сервиса на основе сообщений	124
3.3.4. Использование брокера сообщений	125
3.3.5. Конкурирующие получатели и порядок следования сообщений	129
3.3.6. Дублирование сообщений	130
3.3.7. Транзакционный обмен сообщениями	132
3.3.8. Библиотеки и фреймворки для обмена сообщениями	136
3.4. Использование асинхронного обмена сообщениями для улучшения доступности	139
3.4.1. Синхронное взаимодействие снижает степень доступности	139
3.4.2. Избавление от синхронного взаимодействия	141
Резюме	144
Глава 4. Управление транзакциями с помощью повествований	146
4.1. Управление транзакциями в микросервисной архитектуре	147
4.1.1. Микросервисная архитектура и необходимость в распределенных транзакциях	148
4.1.2. Проблемы с распределенными транзакциями	148
4.1.3. Использование шаблона «Повествование» для сохранения согласованности данных	150
4.2. Координация повествований	154
4.2.1. Повествования, основанные на хореографии	154
4.2.2. Повествования на основе оркестрации	159

4.3.	Что делать с недостаточной изолированностью	164
4.3.1.	Обзор аномалий.....	165
4.3.2.	Контрмеры на случай нехватки изолированности	166
4.4.	Архитектура сервиса Order и повествования Create Order	170
4.4.1.	Класс OrderService	172
4.4.2.	Реализация повествования Create Order	173
4.4.3.	Класс OrderCommandHandlers	181
4.4.4.	Класс OrderServiceConfiguration.....	182
	Резюме	184
Глава 5.	Проектирование бизнес-логики в микросервисной архитектуре.....	185
5.1.	Шаблоны организации бизнес-логики	186
5.1.1.	Проектирование бизнес-логики с помощью шаблона «Сценарий транзакции»	188
5.1.2.	Проектирование бизнес-логики с помощью шаблона «Доменная модель»	189
5.1.3.	О предметно-ориентированном проектировании.....	190
5.2.	Проектирование доменной модели с помощью шаблона «Агрегат» из DDD....	191
5.2.1.	Проблемы с расплывчатыми границами.....	192
5.2.2.	Агрегаты имеют четкие границы.....	194
5.2.3.	Правила для агрегатов	195
5.2.4.	Размеры агрегатов.....	198
5.2.5.	Проектирование бизнес-логики с помощью агрегатов.....	199
5.3.	Публикация доменных событий.....	200
5.3.1.	Зачем публиковать события об изменениях	200
5.3.2.	Что такое доменное событие	201
5.3.3.	Обогащение события	202
5.3.4.	Определение доменных событий	202
5.3.5.	Генерация и публикация доменных событий.....	204
5.3.6.	Потребление доменных событий.....	207
5.4.	Бизнес-логика сервиса Kitchen	208
5.4.1.	Агрегат Ticket	210
5.5.	Бизнес-логика сервиса Order	214
5.5.1.	Агрегат Order.....	215
5.5.2.	Класс OrderService	220
	Резюме	222

Глава 6. Разработка бизнес-логики с порождением событий	223
6.1. Разработка бизнес-логики с использованием порождения событий	224
6.1.1. Проблемы традиционного сохранения данных	225
6.1.2. Обзор порождения событий	227
6.1.3. Обработка конкурентных обновлений с помощью оптимистичного блокирования	234
6.1.4. Порождение и публикация событий	235
6.1.5. Улучшение производительности с помощью снимков	236
6.1.6. Идемпотентная обработка сообщений	238
6.1.7. Развитие доменных событий	239
6.1.8. Преимущества порождения событий	241
6.1.9. Недостатки порождения событий	242
6.2. Реализация хранилища событий	244
6.2.1. Принцип работы хранилища событий Eventuate Local	245
6.2.2. Клиентский фреймворк Eventuate для Java	248
6.3. Совместное использование повествований и порождения событий	252
6.3.1. Реализация повествований на основе хореографии с помощью порождения событий	253
6.3.2. Создание повествования на основе оркестрации	254
6.3.3. Реализация участника повествования на основе порождения событий	256
6.3.4. Реализация оркестраторов повествований с помощью порождения событий	260
Резюме	262
Глава 7. Реализация запросов в микросервисной архитектуре	264
7.1. Выполнение запросов с помощью объединения API	265
7.1.1. Запрос findOrder()	265
7.1.2. Обзор шаблона «Объединение API»	266
7.1.3. Реализация запроса findOrder() путем объединения API	268
7.1.4. Архитектурные проблемы объединения API	269
7.1.5. Преимущества и недостатки объединения API	272
7.2. Применение шаблона CQRS	273
7.2.1. Потенциальные причины использования CQRS	274
7.2.2. Обзор CQRS	277
7.2.3. Преимущества CQRS	280
7.2.4. Недостатки CQRS	281

7.3. Проектирование CQRS-представлений	282
7.3.1. Выбор хранилища данных для представления	283
7.3.2. Структура модуля доступа к данным.....	285
7.3.3. Добавление и обновление CQRS-представлений.....	288
7.4. Реализация CQRS с использованием AWS DynamoDB.....	289
7.4.1. Модуль OrderHistoryEventHandlers.....	290
7.4.2. Моделирование данных и проектирование запросов с помощью DynamoDB	291
7.4.3. Класс OrderHistoryDaoDynamoDb.....	296
Резюме.....	299
Глава 8. Шаблоны внешних API.....	301
8.1. Проблемы с проектированием внешних API.....	302
8.1.1. Проблемы проектирования API для мобильного клиента FTGO	303
8.1.2. Проблемы с проектированием API для клиентов другого рода	306
8.2. Шаблон «API-шлюз»	307
8.2.1. Обзор шаблона «API-шлюз».....	308
8.2.2. Преимущества и недостатки API-шлюза	315
8.2.3. Netflix как пример использования API-шлюза.....	316
8.2.4. Трудности проектирования API-шлюза.....	316
8.3. Реализация API-шлюза	320
8.3.1. Использование готового API-шлюза	320
8.3.2. Разработка собственного API-шлюза.....	322
8.3.3. Реализация API-шлюза с помощью GraphQL.....	329
Резюме.....	341
Глава 9. Тестирование микросервисов, часть 1	343
9.1. Стратегии тестирования микросервисных архитектур.....	345
9.1.1. Обзор методик тестирования	345
9.1.2. Трудности тестирования микросервисов	352
9.1.3. Процесс развертывания	358
9.2. Написание модульных тестов для сервиса.....	360
9.2.1. Разработка модульных тестов для доменных сущностей.....	363
9.2.2. Написание модульных тестов для объектов значений	364
9.2.3. Разработка модульных тестов для повествований.....	364
9.2.4. Написание модульных тестов для доменных сервисов	366

9.2.5. Разработка модульных тестов для контроллеров	368
9.2.6. Написание модульных тестов для обработчиков событий и сообщений.....	370
Резюме.....	371
Глава 10. Тестирование микросервисов, часть 2	373
10.1. Написание интеграционных тестов.....	374
10.1.1. Интеграционные тесты с сохранением	376
10.1.2. Интеграционное тестирование взаимодействия в стиле «запрос/ответ» на основе REST	378
10.1.3. Интеграционное тестирование взаимодействия в стиле «издатель/подписчик»	382
10.1.4. Интеграционные тесты контрактов для взаимодействия на основе асинхронных запросов/ответов.....	386
10.2. Разработка компонентных тестов	391
10.2.1. Определение приемочных тестов.....	392
10.2.2. Написание приемочных тестов с помощью Gherkin	392
10.2.3. Проектирование компонентных тестов.....	395
10.2.4. Написание компонентных тестов для сервиса Order.....	396
10.3. Написание сквозных тестов.....	401
10.3.1. Проектирование сквозных тестов	402
10.3.2. Написание сквозных тестов	402
10.3.3. Выполнение сквозных тестов.....	403
Резюме.....	403
Глава 11. Разработка сервисов, готовых к промышленному использованию.....	405
11.1. Разработка безопасных сервисов	406
11.1.1. Обзор безопасности в традиционном монолитном приложении	407
11.1.2. Обеспечение безопасности в микросервисной архитектуре	411
11.2. Проектирование конфигурируемых сервисов	420
11.2.1. Вынесение конфигурации вовне с помощью пассивной модели	421
11.2.2. Вынесение конфигурации вовне с помощью активной модели	423
11.3. Проектирование наблюдаемых сервисов.....	424
11.3.1. Использование API проверки работоспособности.....	426
11.3.2. Применение шаблона агрегации журналов	428
11.3.3. Использование шаблона распределенной трассировки	430
11.3.4. Применение шаблона «Показатели приложения»	434

11.3.5. Шаблон отслеживания исключений	437
11.3.6. Применение шаблона «Ведение журнала аудита»	439
11.4. Разработка сервисов с помощью шаблона микросервисного шасси	440
11.4.1. Использование шасси микросервисов	441
11.4.2. От микросервисного шасси до сети сервисов	442
Резюме	444
Глава 12. Развертывание микросервисов	446
12.1. Развертывание сервисов с помощью пакетов для отдельных языков	449
12.1.1. Преимущества использования пакетов для конкретных языков	452
12.1.2. Недостатки применения пакетов для конкретных языков	452
12.2. Развертывание сервисов в виде виртуальных машин	454
12.2.1. Преимущества развертывания сервисов в виде VM	456
12.2.2. Недостатки развертывания сервисов в виде VM	457
12.3. Развертывание сервисов в виде контейнеров	458
12.3.1. Развертывание сервисов с помощью Docker	460
12.3.2. Преимущества развертывания сервисов в виде контейнеров	463
12.3.3. Недостатки развертывания сервисов в виде контейнеров	463
12.4. Развертывание приложения FTGO с помощью Kubernetes	463
12.4.1. Обзор Kubernetes	464
12.4.2. Развертывание сервиса Restaurant в Kubernetes	467
12.4.3. Развертывание API-шлюза	470
12.4.4. Развертывание без простоя	471
12.4.5. Использование сети сервисов для отделения развертывания от выпуска	472
12.5. Бессерверное развертывание сервисов	481
12.5.1. Обзор бессерверного развертывания с помощью AWS Lambda	482
12.5.2. Написание лямбда-функции	483
12.5.3. Вызов лямбда-функций	484
12.5.4. Преимущества использования лямбда-функций	485
12.5.5. Недостатки использования лямбда-функций	485
12.6. Развертывание RESTful-сервиса с помощью AWS Lambda и AWS Gateway	486
12.6.1. Архитектура сервиса Restaurant на основе AWS Lambda	487
12.6.2. Упаковывание сервиса в виде ZIP-файла	491
12.6.3. Развертывание лямбда-функций с помощью бессерверного фреймворка	492
Резюме	493

Глава 13. Процесс перехода на микросервисы	495
13.1. Переход на микросервисы	496
13.1.1. Зачем переходить с монолита на что-то другое	496
13.1.2. «Удушение» монолита	497
13.2. Стратегии перехода с монолита на микросервисы	501
13.2.1. Реализация новых возможностей в виде сервисов	501
13.2.2. Разделение уровня представления и внутренних компонентов	503
13.2.3. Извлечение бизнес-возможностей в сервисы	505
13.3. Проектирование взаимодействия между сервисом и монолитом	512
13.3.1. Проектирование интеграционного слоя	513
13.3.2. Обеспечение согласованности данных между сервисом и монолитом	518
13.3.3. Аутентификация и авторизация	523
13.4. Реализация новой возможности в виде сервиса	525
13.4.1. Архитектура сервиса Delayed Delivery	526
13.4.2. Проектирование интеграционного слоя для сервиса Delayed Order	528
13.5. Разбиение монолита на части: извлечение управления доставкой	530
13.5.1. Обзор возможностей существующего механизма управления доставкой	530
13.5.2. Обзор сервиса Delivery	532
13.5.3. Проектирование доменной модели сервиса Delivery	533
13.5.4. Структура интеграционного слоя для сервиса Delivery	536
13.5.5. Изменение монолита для взаимодействия с сервисом Delivery	538
Резюме	541