

Краткое содержание

Предисловие	20
Вступление	23
Благодарности	25
О книге.	27
Об авторе	34
Иллюстрация на обложке	35
От издательства	36

Часть I. Введение

Глава 1. Знакомство с API.	38
Глава 2. Знакомство с паттернами проектирования API.	51

Часть II. Принципы проектирования

Глава 3. Именованье	70
Глава 4. Состав и иерархия ресурсов	86
Глава 5. Типы данных и значения по умолчанию	106

Часть III. Основы

Глава 6. Идентификация ресурсов	130
Глава 7. Стандартные методы	150
Глава 8. Частичное обновление и извлечение	170

6 Краткое содержание

Глава 9. Пользовательские методы	191
Глава 10. Длительные операции	205
Глава 11. Перезапускаемые задачи	229

Часть IV. Связи ресурсов

Глава 12. Подресурсы-одиночки.	244
Глава 13. Перекрестные ссылки	256
Глава 14. Ассоциирующие ресурсы	264
Глава 15. Пользовательские методы Add и Remove.	276
Глава 16. Полиморфизм	284

Часть V. Коллективные операции

Глава 17. Копирование и перемещение	298
Глава 18. Пакетные операции	315
Глава 19. Удаление на основе критерия	333
Глава 20. Анонимная запись	343
Глава 21. Пагинация	351
Глава 22. Фильтрация	367
Глава 23. Импорт и экспорт.	382

Часть VI. Безопасность

Глава 24. Версионирование и совместимость.	404
Глава 25. Мягкое удаление	429
Глава 26. Повтор запросов	444
Глава 27. Валидация запросов	457
Глава 28. Ревизии ресурсов	465
Глава 29. Повтор запросов	482
Глава 30. Аутентификация запросов	494

Оглавление

Предисловие	20
Вступление	23
Благодарности	25
О книге.	27
Для кого эта книга	27
Структура издания	27
О коде	31
Онлайн-ресурсы.	33
Об авторе	34
Иллюстрация на обложке	35
От издательства	36

Часть I. Введение

Глава 1. Знакомство с API.	38
1.1. Что такое веб-API	38
1.2. Почему API важны.	40
1.3. Что значит ресурсно-ориентированные API.	41
1.4. Что делает API «хорошим»	44
1.4.1. Функциональность	45
1.4.2. Выразительность.	45
1.4.3. Простота	46
1.4.4. Предсказуемость	48
Резюме.	49

8 Оглавление

Глава 2. Знакомство с паттернами проектирования API	51
2.1. Что такое паттерны проектирования API	51
2.2. Чем важны паттерны проектирования API	54
2.3. Анатомия паттерна проектирования API.	57
2.3.1. Имя и аннотация	57
2.3.2. Зачем он нужен.	57
2.3.3. Обзор	58
2.3.4. Реализация	59
2.3.5. Компромиссы	60
2.4. Практическое исследование: Twapi, Twitter-подобный API.	60
2.4.1. Обзор	61
2.4.2. Создание списка сообщений.	61
2.4.3. Экспорт данных	64
Резюме.	68

Часть II. Принципы проектирования

Глава 3. Именованье	70
3.1. Почему важны имена	71
3.2. Что делает имя «хорошим»	72
3.2.1. Выразительность.	72
3.2.2. Простота	72
3.2.3. Предсказуемость	73
3.3. Язык, грамматика и синтаксис	74
3.3.1. Язык.	75
3.3.2. Грамматика	75
3.3.3. Синтаксис	78
3.4. Контекст	79
3.5. Типы данных и единицы измерения.	80
3.6. Практическое исследование: что происходит при выборе плохих имен.	82
3.7. Упражнения	84
Резюме.	85
Глава 4. Состав и иерархия ресурсов	86
4.1. Что такое схема ресурсов.	87
4.1.1. Типы связей	88
4.1.2. Блок-схемы связей между сущностями.	91
4.2. Выбор правильных связей	93
4.2.1. Нужна ли вообще связь.	93
4.2.2. Ссылки или встроенные данные	95
4.2.3. Иерархия.	97

4.3. Антипаттерны	99
4.3.1. Ресурсы для всего	99
4.3.2. Глубокие иерархии	102
4.3.3. Встраивание всего	104
4.4. Упражнения	105
Резюме	105
Глава 5. Типы данных и значения по умолчанию	106
5.1. Знакомство с типами данных	107
5.1.1. Отсутствующее значение и значение null	108
5.2. Логический тип	109
5.3. Числа	111
5.3.1. Границы	112
5.3.2. Значения по умолчанию	113
5.3.3. Сериализация	114
5.4. Строки	115
5.4.1. Границы	116
5.4.2. Значения по умолчанию	117
5.4.3. Сериализация	118
5.5. Перечисления	119
5.6. Списки	120
5.6.1. Атомарность	121
5.6.2. Границы	122
5.6.3. Значения по умолчанию	123
5.7. Карты	123
5.7.1. Границы	126
5.7.2. Предустановленные значения	127
5.8. Упражнения	127
Резюме	127

Часть III. Основы

Глава 6. Идентификация ресурсов	130
6.1. Что такое идентификатор	130
6.2. Что делает идентификатор хорошим	131
6.2.1. Простота использования	131
6.2.2. Уникальность	132
6.2.3. Постоянство	132
6.2.4. Быстрая и легкая генерация	133
6.2.5. Непредсказуемость	133
6.2.6. Читательность, передача и возможность проверки	134
6.2.7. Информационная плотность	134

10 Оглавление

6.3. Как выглядит хороший идентификатор	135
6.3.1. Тип данных	135
6.3.2. Набор символов	135
6.3.3. Формат идентификатора.	136
6.3.4. Контрольные суммы.	137
6.3.5. Тип ресурса	138
6.3.6. Иерархия и область уникальности.	138
6.4. Реализация.	140
6.4.1. Размер	140
6.4.2. Генерация	141
6.4.3. Отметка об удалении (tombstoning).	143
6.4.4. Контрольная сумма	144
6.4.5. Хранилище базы данных.	145
6.5. UUID	147
6.6. Упражнения	148
Резюме.	149
Глава 7. Стандартные методы	150
7.1. Зачем это нужно	150
7.2. Обзор	151
7.3. Реализация.	152
7.3.1. Какие методы должны поддерживаться	153
7.3.2. Идемпотентность и побочные эффекты	154
7.3.3. Метод Get	155
7.3.4. Метод List	156
7.3.5. Метод Create	159
7.3.6. Метод Update	161
7.3.7. Метод Delete	162
7.3.8. Метод Replace.	164
7.3.9. Итоговое определение API.	166
7.4. Компромиссы	168
7.5. Упражнения	168
Резюме.	169
Глава 8. Частичное обновление и извлечение	170
8.1. Зачем это нужно	171
8.1.1. Частичное извлечение	171
8.1.2. Частичное обновление	171
8.2. Обзор	174
8.3. Реализация.	176
8.3.1. Передача	176

8.3.2. Карты и вложенные интерфейсы	178
8.3.3. Повторяющиеся поля	180
8.3.4. Значения по умолчанию	183
8.3.5. Неявные маски полей	184
8.3.6. Обновление динамических структур данных	185
8.3.7. Недействительные поля	187
8.3.8. Итоговое определение API	187
8.4. Компромиссы	188
8.4.1. Универсальная поддержка	188
8.4.2. Альтернативные реализации	189
8.5. Упражнения	190
Резюме	190
Глава 9. Пользовательские методы	191
9.1. Зачем это нужно	192
9.1.1. Почему бы просто не задействовать стандартные методы	192
9.2. Обзор	195
9.3. Реализация	196
9.3.1. Побочные эффекты	197
9.3.2. Ресурсы или коллекции	199
9.3.3. Пользовательские методы без сохранения состояния	200
9.3.4. Итоговое определение API	202
9.4. Компромиссы	203
9.5. Упражнения	204
Резюме	204
Глава 10. Длительные операции	205
10.1. Зачем это нужно	206
10.2. Обзор	207
10.3. Реализация	210
10.3.1. Как выглядит длительная операция	210
10.3.2. Иерархия ресурсов	212
10.3.3. Разрешение	212
10.3.4. Обработка ошибок	216
10.3.5. Отслеживание прогресса	218
10.3.6. Отмена операций	219
10.3.7. Приостановка и возобновление операций	221
10.3.8. Инспектирование операций	222
10.3.9. Продолжительность хранения	224
10.3.10. Итоговое определение API	225

12 Оглавление

10.4. Компромиссы	227
10.5. Упражнения	228
Резюме	228
Глава 11. Перезапускаемые задачи	229
11.1. Зачем это нужно	230
11.2. Обзор	231
11.3. Реализация	232
11.3.1. Ресурсы Job	232
11.3.2. Пользовательский метод Run	235
11.3.3. Ресурсы выполнения задачи	236
11.3.4. Итоговое определение API	239
11.4. Компромиссы	241
11.5. Упражнения	241
Резюме	242
Часть IV. Связи ресурсов	
Глава 12. Подресурсы-одиночки	244
12.1. Зачем это нужно	244
12.1.1. Почему нужно использовать подресурс-одиночку?	245
12.2. Обзор	246
12.3. Реализация	247
12.3.1. Стандартные методы	248
12.3.2. Сброс	251
12.3.3. Иерархия	251
12.3.4. Итоговое определение API	252
12.4. Компромиссы	253
12.4.1. Атомарность выполнения	254
12.4.2. Ровно один подресурс	254
12.5. Упражнения	254
Резюме	255
Глава 13. Перекрестные ссылки	256
13.1. Зачем это нужно	256
13.2. Обзор	257
13.3. Реализация	258
13.3.1. Имя ссылочного поля	258
13.3.2. Целостность данных	259
13.3.3. Значение или ссылка	260
13.3.4. Итоговое определение API	262
13.4. Компромиссы	263
13.5. Упражнения	263
Резюме	263

Глава 14. Ассоциирующие ресурсы	264
14.1. Зачем это нужно	264
14.2. Обзор	265
14.2.1. Ассоциирующие методы-псевдонимы.	267
14.3. Реализация.	267
14.3.1. Выбор имени для ассоциирующего ресурса	267
14.3.2. Поведение стандартных методов	268
14.3.3. Уникальность	268
14.3.4. Поля только для чтения.	269
14.3.5. Ассоциирующие методы-псевдонимы.	270
14.3.6. Ссылочная целостность.	271
14.3.7. Итоговое определение API.	272
14.4. Компромиссы	274
14.4.1. Сложность	274
14.4.2. Отделенность ассоциирующих ресурсов	274
14.5. Упражнения	275
Резюме.	275
Глава 15. Пользовательские методы Add и Remove.	276
15.1. Зачем это нужно	276
15.2. Обзор	277
15.3. Реализация.	278
15.3.1. Вывод списка связанных ресурсов	279
15.3.2. Целостность данных.	279
15.3.3. Итоговое определение API.	280
15.4. Компромиссы	282
15.4.1. Подчинительная связь	282
15.4.2. Метаданные о связи	282
15.5. Упражнения	282
Резюме.	283
Глава 16. Полиморфизм	284
16.1. Зачем это нужно	284
16.2. Обзор	285
16.3. Реализация.	286
16.3.1. Когда использовать полиморфные ресурсы	286
16.3.2. Полиморфная структура	288
16.3.3. Полиморфное поведение	291
16.3.4. Почему не стоит использовать полиморфные методы.	292
16.3.5. Итоговое определение API.	294
16.4. Компромиссы	295
16.5. Упражнения	295
Резюме.	296

Часть V. Коллективные операции

Глава 17. Копирование и перемещение	298
17.1. Зачем это нужно	299
17.2. Обзор	299
17.3. Реализация.	300
17.3.1. Идентификаторы	300
17.3.2. Дочерние ресурсы	303
17.3.3. Связанные ресурсы	304
17.3.4. Внешние данные	307
17.3.5. Наследуемые метаданные	309
17.3.6. Атомарность	310
17.3.7. Итоговое определение API	312
17.4. Компромиссы	313
17.5. Упражнения	314
Резюме.	314
Глава 18. Пакетные операции	315
18.1. Зачем это нужно	315
18.2. Обзор	316
18.3. Реализация.	317
18.3.1. Атомарность	318
18.3.2. Операции над коллекцией	318
18.3.3. Упорядочивание результатов	319
18.3.4. Общие поля	320
18.3.5. Оперирование с разными родителями.	321
18.3.6. Пакетный метод Get	322
18.3.7. Пакетный метод Delete	324
18.3.8. Пакетный метод Create	326
18.3.9. Пакетный метод Update.	327
18.3.10. Итоговое определение API	329
18.4. Компромиссы	331
18.5. Упражнения	332
Резюме.	332
Глава 19. Удаление на основе критерия	333
19.1. Зачем это нужно	334
19.2. Обзор	335
19.3. Реализация.	335
19.3.1. Фильтрация результатов	337
19.3.2. По умолчанию только проверка.	337
19.3.3. Подсчет результатов.	338

19.3.4. Выборка результатов	339
19.3.5. Согласованность	340
19.3.6. Итоговое определение API	341
19.4. Компромиссы	341
19.5. Упражнения	341
Резюме	342
Глава 20. Анонимная запись	343
20.1. Зачем это нужно	343
20.2. Обзор	344
20.3. Реализация	346
20.3.1. Согласованность	347
20.3.2. Итоговое определение API	348
20.4. Компромиссы	349
20.5. Упражнения	350
Резюме	350
Глава 21. Пагинация	351
21.1. Зачем это нужно	352
21.2. Обзор	352
21.3. Реализация	353
21.3.1. Размер страницы	354
21.3.2. Маркеры страниц	357
21.3.3. Общее количество	360
21.3.4. Пагинация внутри ресурсов	361
21.3.5. Итоговое определение API	363
21.4. Компромиссы	363
21.4.1. Двухнаправленная пагинация	363
21.4.2. Произвольный выбор окна	364
21.5. Антипаттерн: смещения и границы	364
21.6. Упражнения	366
Резюме	366
Глава 22. Фильтрация	367
22.1. Зачем это нужно	367
22.2. Обзор	368
22.3. Реализация	369
22.3.1. Структура	370
22.3.2. Синтаксис фильтра и поведение	373
22.3.3. Итоговое определение API	380
22.4. Компромиссы	380
22.5. Упражнения	381
Резюме	381

Глава 23. Импорт и экспорт	382
23.1. Зачем это нужно	383
23.2. Обзор	384
23.3. Реализация	386
23.3.1. Методы Import и Export	386
23.3.2. Взаимодействие с системами хранения данных	387
23.3.3. Преобразование между ресурсами и байтами	389
23.3.4. Согласованность	390
23.3.5. Идентификаторы и коллизии	392
23.3.6. Обработка связанных ресурсов	393
23.3.7. Сбои и повторы	394
23.3.8. Фильтрация и маски полей	397
23.3.9. Итоговое определение API	399
23.4. Компромиссы	401
23.5. Упражнения	402
Резюме	402

Часть VI. Безопасность

Глава 24. Версионирование и совместимость	404
24.1. Зачем это нужно	404
24.2. Обзор	405
24.2.1. Что такое совместимость	406
24.2.2. Определение обратной совместимости	407
24.3. Реализация	415
24.3.1. Вечная стабильность	415
24.3.2. Подвижная нестабильность	417
24.3.3. Семантическое версионирование	420
24.4. Компромиссы	423
24.4.1. Детализация или простота?	423
24.4.2. Стабильность или новая функциональность?	424
24.4.3. Удовлетворенность или универсальность?	425
24.5. Упражнения	427
Резюме	428
Глава 25. Мягкое удаление	429
25.1. Зачем это нужно	430
25.2. Обзор	430
25.3. Реализация	431
25.3.1. Обозначение удаления	432
25.3.2. Изменение стандартных методов	434
25.3.3. Метод Undelete	436

25.3.4. Метод Exprunge	437
25.3.5. Сроки хранения	438
25.3.6. Ссылочная целостность	439
25.3.7. Влияние на другие методы	440
25.3.8. Добавление мягкого удаления для разных версий	441
25.3.9. Итоговое определение API	441
25.4. Компромиссы	442
25.5. Упражнения	443
Резюме	443
Глава 26. Повтор запросов	444
26.1. Зачем это нужно	444
26.2. Обзор	446
26.3. Реализация	447
26.3.1. Идентификатор запроса	448
26.3.2. Кэширование ответа	449
26.3.3. Согласованность	450
26.3.4. Коллизии ID запросов	452
26.3.5. Срок действия кэша	454
26.3.6. Итоговое определение API	454
26.4. Компромиссы	456
26.5. Упражнения	456
Резюме	456
Глава 27. Валидация запросов	457
27.1. Зачем это нужно	457
27.2. Обзор	459
27.3. Реализация	459
27.3.1. Внешние зависимости	461
27.3.2. Особые побочные эффекты	462
27.3.3. Итоговое определение API	463
27.4. Компромиссы	464
27.5. Упражнения	464
Резюме	464
Глава 28. Ревизии ресурсов	465
28.1. Зачем это нужно	466
28.2. Обзор	466
28.3. Реализация	467
28.3.1. Идентификаторы ревизий	468
28.3.2. Создание ревизий	469
28.3.3. Извлечение конкретных ревизий	473
28.3.4. Перечисление ревизий	474

- 28.3.5. Восстановление прежней ревизии 475
- 28.3.6. Удаление ревизий 476
- 28.3.7. Обработка дочерних ресурсов 478
- 28.3.8. Итоговое определение API 479
- 28.4. Компромиссы 480
- 28.5. Упражнения 480
- Резюме 481
- Глава 29. Повтор запросов 482**
 - 29.1. Зачем это нужно 482
 - 29.2. Обзор 483
 - 29.2.1. Интервал повторов на стороне клиента 484
 - 29.2.2. Интервал повторов, определяемый сервером 484
 - 29.3. Реализация 485
 - 29.3.1. Допустимость повтора 485
 - 29.3.2. Экспоненциальная выдержка 487
 - 29.3.3. Retry-After 490
 - 29.3.4. Итоговое определение API 492
 - 29.4. Компромиссы 493
 - 29.5. Упражнения 493
 - Резюме 493
- Глава 30. Аутентификация запросов 494**
 - 30.1. Зачем это нужно 495
 - 30.1.1. Источник 495
 - 30.1.2. Целостность 495
 - 30.1.3. Невозможность отказа 496
 - 30.2. Обзор 497
 - 30.3. Реализация 498
 - 30.3.1. Генерация учетных данных 498
 - 30.3.2. Регистрация и обмен учетными данными 499
 - 30.3.3. Генерация и верификация сырых подписей 500
 - 30.3.4. Цифровой отпечаток для запроса 502
 - 30.3.5. Добавление подписи 504
 - 30.3.6. Аутентификация запросов 507
 - 30.3.7. Итоговое определение API 508
 - 30.4. Компромиссы 508
 - 30.5. Упражнения 509
 - Резюме 510

Посвящается Ка-эль и Луке. Вы потрясающие.

Предисловие

Все началось с электронной ударной установки. Летом 2019 года один мой друг увлек меня игрой на ней, и я погрузился в это занятие с головой. Иногда я действительно играл на барабанах, но большую часть времени все же проводил за написанием кода, который позволял управлять конфигурацией установки с помощью команд MIDI SysEx.

Когда нагрянула пандемия COVID-19, у меня внезапно появились иные приоритеты, направленные на настройку аудиовизуальной связи в нашей местной церкви, где мы перешли в формат удаленного проведения служб и рассматривали возможности возобновления личных встреч. Для этого мне пришлось изучать протоколы VISCA, NDI и OSC (для камер и аудиомикшеров), а также осваивать программно-ориентированную интеграцию с Zoom, VLC, PowerPoint, Stream Deck и др.

Эти проекты не несут в себе больших объемов бизнес-логики. Практически весь код относится к интеграции, что одновременно и раздражает, и дает большие возможности. Раздражение вызывали протоколы, которые либо были плохо задокументированы, либо не предназначались для тех задач, которые я пытался реализовать с их помощью, либо просто не согласовывались друг с другом. Положительная сторона всего этого заключается в том, что, как только вам удастся разобраться с интеграцией, вы сможете очень легко писать полезные приложения.

Несмотря на то что моя работа в последние несколько лет заключалась в основном в локальной интеграции, точно такие же раздражение и ощущение больших

возможностей возникают и при работе с веб-API. Каждый опыт выбора нового веб-API представляет собой кривую, состоящую из эмоциональных реакций в виде волнения, замешательства, раздражения, принятия и достигаемого в итоге умиротворения. Как только ты досконально осваиваешь мощный API, возникает чувство, что ты являешься дирижером величественного оркестра, готового играть любую предложенную тобой музыку, — даже если ноты скрипачей согласуются лишь под конец, а для группы медных духовых без каких-либо очевидных причин приходится использовать палочку другого цвета.

Данная книга сама по себе не изменит положение дел. Это всего лишь книга. Но если после прочтения вы будете следовать указаниям, изложенным в ней, то она поможет вам обеспечить пользователям более качественный опыт. Если же эту книгу прочтет множество людей, то вместе мы сможем добиться сдвига в направлении более согласованного и менее раздражающего опыта работы с API.

Важно понять, что сама эта книга в целом гораздо более ценна, чем сумма ее фрагментов. Относительно любого из разбираемых Джей Джейем паттернов любая команда может сделать свой обдуманный выбор (хотя кто-то может упустить некоторые из указанных здесь пограничных случаев). Для конкретной ситуации этот выбор может оказаться даже более удачным, чем предложенные в книге рекомендации, ввиду ограниченных требований контекста. Такой подход позволяет получить множество локальных оптимальных решений, но дает очень фрагментированную общую картину, поскольку в этом случае даже в API одной компании потенциально применяется несколько разрозненных техник.

Помимо согласованности для любой отдельно взятой задачи, книга также предлагает единый подход, охватывающий множество сфер проектирования API. Разработчикам программных интерфейсов редко предоставляется пространство для углубленного анализа этих аспектов, и я считаю, что мне очень повезло работать с Джей Джейем и другими (в частности, с Люком Спирингером (Luke Spengler)) над обсуждением многих тем книги. Я счастлив, что вклад, внесенный компанией Google в проектирование API, может быть весьма полезным многим разработчикам благодаря этой книге и системе AIP, расположенной по адресу <https://aip.dev>.

Я абсолютно уверен в ценности этой книги, но она не упростит разработку API. Однако она позволит вам избавиться от ненужных сложностей, которые возникают в ходе проектирования интерфейсов, давая возможность сосредоточиться только на том, что действительно уникально для ваших API. Будьте готовы думать, и думать усиленно, но также знайте, что результатом этих размышлений может стать API, работа с которым доставит лишь удовольствие. Ваши пользователи могут никогда не поблагодарить вас за это открыто. Ведь

22 Предисловие

хорошо спроектированный API зачастую воспринимается как само собой разумеющийся, хоть и является результатом огромных усилий. Но зато вы сможете спать спокойно, зная, что ваши пользователи не будут злиться при работе с API, который пусть и функционирует, но сделан на троечку.

Используйте эту книгу как основу, чтобы построить такой API, который сможет стать фундаментом для других.

*Джон Скит (Jon Skeet),
Staff Developer Relations Engineer, Google*

Вступление

Многие из нас изучали информатику. Мы анализировали среду выполнения и пространственную сложность алгоритмов с помощью нотации «О большое», знакомились со всевозможными алгоритмами сортировки и изучали различные способы обхода бинарных деревьев. Поэтому, окончив колледж, я планировал, что моя работа будет в первую очередь связана с математикой и точными науками. Но каким же было мое удивление, когда в действительности все оказалось совсем не так.

На деле бóльшая часть работы была связана с проектированием, структурированием и дизайном, а не математикой и алгоритмами. Мне никогда не приходилось думать о том, какой алгоритм сортировки использовать, поскольку для этого всегда имелась библиотека (обычно что-то вроде `array.sort()`). Однако мне все же доводилось долго и упорно размышлять над создаваемыми классами, над функциями для них и параметрами, которые должна принимать каждая функция. Все это оказалось намного более сложным, чем я ожидал.

В реальном мире я узнал, что идеально оптимизированный код далеко не столь же ценен, как хорошо спроектированный. И это оказалось вдвойне верным в отношении веб-API, поскольку они, как правило, имеют гораздо более обширную аудиторию и намного больший спектр случаев применения.

Но здесь напрашиваются вопросы: что значит «хорошо спроектированное ПО»? Что такое «хорошо спроектированный API»? Чтобы ответить на эти вопросы, довольно долго мне приходилось полагаться в основном на случайный набор ресурсов. Для одних тем я отыскивал интересные посты, которые раскрывали популярные современные альтернативы. Для других находил отдельные полезные ответы на Stack Overflow, которые подсказывали нужное направление.

Однако во многих случаях было довольно мало материалов по рассматриваемой теме, и мне приходилось самостоятельно придумывать ответы, надеясь, что мои коллеги не слишком их осудят.

Спустя многие годы подобных изысканий (и таскания повсюду блокнота с надписью «Пугающие проблемы API») я наконец-то решил, что пришло время записать всю информацию, которую я собрал и использовал в работе. Поначалу все это выглядело как набор правил для Google, которые мы с Люком Снирингером систематизировали и которые в конечном итоге стали AIP.dev. Но эти правила выглядели как свод законов. В них говорилось, что нужно делать, но не пояснялось, почему нужно делать именно так. После долгого анализа и исследований, сопровождавшихся непрерывным повторением в уме этого вопроса, я готов представить вам эти правила, а также пояснить их причины.

Несомненно, было бы замечательно, если бы эта книга стала абсолютным решением для всех проблем в мире проектирования API, но, к сожалению, этого не случится. И причина тому проста: как и в архитектуре, выбор любой модели проектирования обычно опирается на личное мнение. Это означает, что одни из вас могут посчитать мои указания удачными и элегантными и будут использовать их для всех последующих проектов. Другие же могут решить, что эта книга демонстрирует отвратительные и чрезмерно ограниченные модели дизайна, и в связи с этим, наоборот, воспринять их в качестве примера того, как не нужно проектировать API. Поскольку я не могу угодить всем, моей единственной целью было предоставить набор проверенных в деле рекомендаций, подкрепив их логическими обоснованиями, объясняющими, почему они выглядят именно так.

Будете ли вы использовать их в качестве примеров для подражания или избегать — дело ваше. По меньшей мере надеюсь, что рассмотренные в книге темы вызовут много дискуссий и инициируют дальнейшую масштабную работу в этом удивительном, сложном и запутанном мире проектирования API.