

Spis treści

Przedmowa	13
1. Mikrouslugi	19
Czym są mikrouslugi?	20
Niewielkie, skoncentrowane na dobrym wykonywaniu jednej rzeczy	20
Autonomiczne	21
Najważniejsze korzyści	22
Niejednorodność technologii	22
Odporność na błędy	23
Skalowanie	23
Łatwość wdrażania	24
Dopasowanie do organizacji zespołów	25
Interoperatywność	25
Optymalizacja w kierunku wymienności	25
Architektura zorientowana na usługi	26
Inne techniki dekompozycji	27
Biblioteki współdzielone	27
Moduły	28
Nie istnieje panaceum na wszystko	29
Podsumowanie	29
2. Ewolucyjny architekt	31
Niedokładne porównania	31
Ewolucyjna wizja architekta	33
Podział na strefy	34
Principjalne podejście	35
Cele strategiczne	36
Zasady	36
Praktyki	37

Łączenie zasad i praktyk	37
Praktyczny przykład	37
Wymagane standardy	38
Monitorowanie	39
Interfejsy	39
Bezpieczeństwo architektury	39
Zarządzanie za pośrednictwem kodu	40
Przykładowe egzemplarze	40
Spersonalizowany szablon usługi	40
Dług techniczny	42
Obsługa wyjątków	42
Zarządzanie i przewodnictwo od środka	43
Budowanie zespołu	44
Podsumowanie	45
3. Jak modelować usługi?	47
Przedstawiamy firmę MusicCorp	47
Co decyduje o tym, że usługa jest dobra?	48
Ładne sprzężenia	48
Wysoka spójność	48
Ograniczony kontekst	49
Modele współdzielone i ukryte	49
Moduły i usługi	51
Przedwczesna dekompozycja	51
Możliwości biznesowe	52
Żółwie aż do spodu	52
Komunikacja w kategoriach pojęć biznesowych	54
Granice techniczne	54
Podsumowanie	55
4. Integracja	57
Poszukiwanie idealnej technologii integracji	57
Unikanie wprowadzania przełomowych zmian	57
Dbanie o niezależność interfejsów API od technologii	57
Dbałość o zapewnienie prostoty usługi dla konsumentów	58
Ukrycie wewnętrznych szczegółów implementacji	58
Interfejs z klientami	58
Wspólna baza danych	59
Komunikacja synchroniczna kontra asynchroniczna	60
Aranżacja kontra choreografia	61

70	Zdalne wywołania procedur	64
71	Sprzężenia technologii	64
72	Wywołania lokalne różnią się od zdalnych	65
73	Kruczość	65
74	Czy wywołania RPC są złym rozwiązaniem?	67
75	REST	67
76	REST a HTTP	68
77	Hipermedium jako silnik stanu aplikacji	68
78	JSON, XML czy coś innego?	70
79	Uważaj na zbyt wielkie wygodę	71
80	Wady interfejsu REST przez HTTP	72
81	Implementacja współpracy asynchronicznej, bazującej na zdarzeniach	73
82	Opcje wyboru technologii	73
83	Zawiloci architektur asynchronicznych	74
84	Usługi jako maszyny stanów	76
85	Rozszerzenia reaktywne	76
86	DRY i perypetie wielokrotnego wykorzystania kodu w świecie mikrousług	77
87	Biblioteki klienckie	77
88	Dostęp przez referencję	78
89	Zarządzanie wersjami	80
90	Odkładaj modyfikowanie interfejsu tak długo, jak to możliwe	80
91	Wczesne wychwytywanie zmian naruszających zgodność interfejsu	81
92	Zastosowanie semantycznej kontroli wersji	81
93	Współlistnienie różnych punktów końcowych	82
94	Korzystanie z wielu równoległych wersji usługi	83
95	Interfejsy użytkownika	84
96	W stronę środowiska cyfrowego	85
97	Ograniczenia	85
98	Kompozycja interfejsów API	86
99	Kompozycja fragmentu interfejsu użytkownika	87
100	Zaplecza dla frontonów	89
101	Podejście hybrydowe	90
102	Integracja z oprogramowaniem zewnętrznych producentów	91
103	Brak kontroli	92
104	Personalizacja	92
105	Makaron integracji	92
106	Personalizacja na własnych warunkach	93
107	Wzorzec Dusiciel	95
108	Podsumowanie	96

5. Dzielenie monolitu	97
To wszystko są szwy	97
Podział systemu w firmie MusicCorp	98
Powody dzielenia monolitu	99
Tempo zmian	99
Struktura zespołu	99
Bezpieczeństwo	99
Technologia	100
Splątane zależności	100
Baza danych	100
Zlikwidowanie problemu	100
Przykład: eliminowanie relacji kluczy obcych	101
Przykład: wspólne statyczne dane	103
Przykład: współdzielone dane	104
Przykład: wspólne tabele	105
Refaktoryzacja baz danych	106
Podział na etapy	106
Granice transakcyjne	107
Spróbuj ponownie później	108
Odrzucenie całej operacji	109
Transakcje rozproszone	109
Jakie rozwiązanie wybrać?	110
Raportowanie	111
Bazy danych raportowania	111
Pobieranie danych za pośrednictwem wywołania usługi	112
Pompy danych	114
Alternatywne lokalizacje docelowe	115
Pompa danych sterowana zdarzeniami	116
Pompa danych z kopii zapasowej	117
W stronę czasu rzeczywistego	117
Koszty zmiany	117
Zrozumieć przyczyny	118
Podsumowanie	119
6. Wdrażanie	121
Krótkie wprowadzenie do ciągłej integracji	121
Czy rzeczywiście to robisz?	122
Mapowanie ciągłej integracji na mikrousługi	123
Potoki kompilacji a ciągle dostawy	125
Nieuniknione wyjątki	126

Artefakty specyficzne dla platformy	127
Artefakty systemu operacyjnego	128
Spersonalizowane obrazy	129
Obrazy jako artefakty	131
Serwery niezmiennie	131
Środowiska	131
Konfiguracja usługi	133
Odwzorowanie usługa-host	133
Wiele usług na hoście	134
Kontenery aplikacji	136
Jedna usługa na host	137
Platforma jako usługa	138
Automatyzacja	139
Dwa studia przypadków na potwierdzenie potęgi automatyzacji	139
Od świata fizycznego do wirtualnego	140
Tradycyjna wirtualizacja	140
Vagrant	141
Kontenery w Linuksie	142
Docker	144
Interfejs instalacji	145
Definicja środowiska	146
Podsumowanie	147
7. Testowanie	149
Rodzaje testów	149
Zakres testów	150
Testy jednostkowe	152
Testy usług	152
Testy od końca do końca	153
Kompromisy	153
Ile?	154
Implementacja testów usług	154
Makiety lub namiastki	155
Inteligentniejsza namiastka usługi	155
Kłopotliwe testy od końca do końca	156
Wady testowania od końca do końca	157
Testy kruche i łamliwe	158
Kto pisze te testy?	159
Jak długo?	159
Piętrzące się zaległości	160
Metawersje	161

131	Testuj ścieżki, a nie historie	161
132	Testy sterowane potrzebami konsumentów	162
133	Pact	163
134	Konwersacje	165
135	Czy należy używać testów od końca do końca?	165
136	Testowanie po opublikowaniu systemu do produkcji	166
137	Oddzielenie wdrożenia od publikacji	166
138	Publikacje kanarkowe	167
139	Średni czas do naprawy kontra średni czas między awariami	168
140	Testy współzależności funkcjonalnych	169
141	Testy wydajności	170
142	Podsumowanie	171
143	8. Monitorowanie	173
144	Jedna usługa, jeden serwer	174
145	Jedna usługa, wiele serwerów	174
146	Wiele usług, wiele serwerów	175
147	Logi, logi i jeszcze raz logi...	176
148	Śledzenie metryk dotyczących wielu usług	177
149	Metryki usług	178
150	Monitorowanie syntetyczne	178
151	Implementacja monitorowania semantycznego	179
152	Identyfikatory korelacji	180
153	Kaskada	182
154	Standaryzacja	182
155	Weź pod uwagę użytkowników	183
156	Przyszłość	184
157	Podsumowanie	185
158	9. Bezpieczeństwo	187
159	Uwierzytelnianie i autoryzacja	187
160	Popularne implementacje pojedynczego logowania	188
161	Brama pojedynczego logowania	189
162	Szczegółowa autoryzacja	190
163	Uwierzytelnianie i autoryzacja w trybie usługa-usługa	191
164	Zezwalaj na wszystko wewnątrz obszaru	191
165	Podstawowe uwierzytelnianie HTTP(S)	192
166	Korzystanie z SAML lub OpenID Connect	192
167	Certyfikaty klienta	193
168	HMAC przez HTTP	194
169	Klucze API	195
170	Problem zastępcy	195

015	Zabezpieczanie danych w spoczynku	197
016	Korzystaj ze sprawdzonych sposobów	198
020	Wszystko dotyczy kluczy	198
021	Wybierz swoje cele	199
022	Odszyfruj dane na żądanie	199
024	Szyfruj kopie zapasowe	199
025	Obrona wielostrefowa	199
026	Zapory firewall	199
027	Rejestrowanie	200
028	Systemy wykrywania włamań (i zapobiegania im)	200
029	Podział sieci	200
030	System operacyjny	201
031	Praktyczny przykład	201
032	Bądź oszczędny	204
033	Czynnik ludzki	204
034	Złota reguła	204
035	Wdrażanie zabezpieczeń	205
036	Zewnętrzna weryfikacja	206
037	Podsumowanie	206
040	10. Prawo Conwaya a projektowanie systemów	207
041	Dowody	207
042	Organizacje sprzężone luźno i ściśle	208
043	Windows Vista	208
044	Netflix i Amazon	208
045	Co można z tym zrobić?	209
046	Dostosowanie się do ścieżek komunikacyjnych	209
047	Własność usługi	210
048	Powody współdzielenia usług	211
049	Zbyt trudne do rozdzielenia	211
050	Zespoły funkcyjne	211
051	Wąskie gardła dostaw	212
052	Wewnętrzne Open Source	212
053	Rola opiekunów	213
054	Dojrzałość	213
055	Narzędzia	214
056	Konteksty ograniczone a struktura zespołów	214
057	Usługa osierocona?	214
058	Studium przypadku: RealEstate.com.au	215
059	Odwrócone prawo Conwaya	216
060	Ludzie	217
061	Podsumowanie	218

11. Mikrousługi w projektach dużej skali	219
Awarie zdarzają się wszędzie	219
Jak wiele jest zbyt wiele?	220
Degradowanie funkcjonalności	221
Środki bezpieczeństwa architektury	222
Antykrucha organizacja	224
Limity czasu	225
Bezpieczniki	225
Grodzie	227
Izolacja	229
Idempotencja	229
Skalowanie	230
Zwiększenie rozmiarów	230
Podział obciążeń	231
Rozłożenie ryzyka	231
Równoważenie obciążenia	232
Systemy bazujące na wątkach roboczych	234
Zaczynanie od nowa	235
Skalowanie baz danych	236
Dostępność usługi kontra trwałość danych	236
Skalowanie do obsługi operacji odczytu	236
Skalowanie do obsługi operacji zapisu	237
Wspólna infrastruktura bazy danych	238
CQRS	238
Buforowanie	239
Buforowanie po stronie klienta, na serwerze proxy i po stronie serwera	239
Buforowanie w HTTP	240
Buforowanie operacji zapisu	242
Buforowanie w celu poprawy niezawodności	242
Ukrywanie źródła	242
Zachowaj prostotę	243
Zatrucie pamięcią podręczną: historia ku przestrodze	244
Autoskalowanie	245
Twierdzenie CAP	246
Poświęcenie spójności	247
Poświęcenie dostępności	247
Poświęcenie tolerancji podziału?	248
AP czy CP?	249
To nie jest zasada „wszystko albo nic”	249
Świat rzeczywisty	250

Wykrywanie usług	250
DNS	251
Dynamiczne rejestry usług	252
Zookeeper	252
Consul	253
Eureka	254
Tworzenie własnych rozwiązań	254
Nie zapomnij o ludziach!	255
Dokumentowanie usług	255
Swagger	256
HAL i przeglądarka HAL	256
System samoopisujący się	257
Podsumowanie	258
12. Podsumowanie	259
Zasady dotyczące mikrousług	259
Wzorowanie na koncepcjach działania biznesu	260
Przyjęcie kultury automatyzacji	260
Ukrywanie wewnętrznych szczegółów implementacji	261
Decentralizacja wszystkich operacji	261
Możliwość niezależnej instalacji	262
Izolowanie awarii	262
Łatwe do obserwacji	263
Kiedy nie należy używać mikrousług?	263
Ostatnie słowo	264
Skorowidz	265

Dlaczego napisałem tę książkę?

O architekturach aplikacji zacząłem myśleć wiele lat temu, kiedy poddawałem pod wyśmianie firm w wyborach dostarczania oprogramowania. Zdałem sobie wtedy sprawę, że o ile automatyzacja infrastruktury, testowanie i technika ciągłych dostaw (continuous delivery) mogą okazać się pomocne, o tyle jeśli podstawowy projekt systemu nie pozwoli na łatwe wprowadzanie zmian, to łatwiej granice tego, co można osiągnąć.