**Budowanie mikrousług : projektowanie drobnoziarnistych systemów / Sam Newman. – Wydanie II. – Gliwice, © 2022**

Spis treści


**Przedmowa** **15**

**CZĘŚĆ I. Podstawy** **23**

**1. Czym są mikrousługi?** **25**
Mikrousługi w skrócie 25
Kluczowe pojęcia dotyczące mikrousług 28
Możliwość niezależnego wdrażania 28
Zamodelowane wokół domeny biznesowej 28
Posiadanie własnego stanu 30
Rozmiar 30
Elastyczność 31
Dopasowanie architektury do organizacji zespołów 32
Monolit 35
Monolit jednoprocesowy 36
Monolit modułowy 37
Monolit rozproszony 38
Monolity i rywalizacja o dostawy 38
Zalety monolitów 39
Technologie pomocnicze 39
Agregacja logów i rozproszone śledzenie 40
Kontenery i Kubernetes 41
Przesyłanie strumieniowe 42
Chmura publiczna i platformy bezserwerowe 42
Najważniejsze korzyści 43
Niejednorodność technologii 43
Odporność na błędy 44
Skalowanie 44
Łatwość wdrażania 45
Dopasowanie do organizacji zespołów 46
Komponowalność 46
Niedogodności związane z architekturą mikrousług 47
Wrażenia programisty 47
Przeciążenie technologią 47
Koszty 48
Raportowanie 48
Monitorowanie i rozwiązywanie problemów 49
Bezpieczeństwo 50
Testowanie 50
Opóźnienia 50

Spójność danych 51
Czy powinienem korzystać z mikrousług? 51
Kiedy mikrousługi mogą się nie sprawdzić? 52
Gdzie mikrousługi działają dobrze? 53
Podsumowanie 54

**2. Jak modelować mikrousługi? 55**
Przedstawiamy firmę MusicCorp 55
Co decyduje o tym, że granice mikrousługi są dobre? 56
Ukrywanie informacji 56
Spójność 57
Sprzężenia 58
Wzajemne oddziaływanie pomiędzy sprzężeniami a spójnością 58
Rodzaje sprzężeń 59
Sprzężenie domen 60
Sprzężenia przelotowe 62
Sprzężenie wspólnych danych 65
Sprzężenia treści 68
Wprowadzenie do metodologii projektowania opartego na domenie (DDD) 70
Język wszechobecny 71
Agregat 71
Kontekst ograniczony 74
Mapowanie agregatów i kontekstów ograniczonych na mikrousługi 76
Event Storming 77
Projektowanie DDD w kontekście mikrousług 79
Alternatywy dla granic domen biznesowych 80
Ulotność 80
Dane 81
Technologia 83
Względy organizacyjne 83
Modele mieszane i wyjątki 85
Podsumowanie 86

**3. Dzielenie monolitu 87**
Określenie celu 87
Migracja przyrostowa 88
Monolit rzadko jest Twoim wrogiem 89
Niebezpieczeństwa przedwczesnej dekompozycji 89
Co podzielić najpierw? 90
Dekompozycja według warstwy 91
Najpierw kod 92
Najpierw dane 93
Przydatne wzorce dekompozycji 94
Wzorzec figowca-dusiciela 94
Uruchamianie równoległe 95
Przełącznik funkcji 95

Problemy z dekompozycją danych 95
Wydajność 95
Integralność danych 98
Transakcje 98
Narzędzia 99
Bazy danych raportowania 99
Podsumowanie 100

**4. Rodzaje komunikacji mikrousług 101**
Od komunikacji wewnątrz procesu do komunikacji między procesami 101
Wydajność 101
Modyfikacje interfejsów 102
Obsługa błędów 103
Technologia komunikacji między procesami: wiele możliwości do wyboru 104
Style komunikacji mikrousług 105
Łącz i dopasowuj 106
Wzorzec komunikacja synchroniczna blokująca 106
Zalety 107
Wady 107
Gdzie stosować wzorzec? 108
Wzorzec komunikacja asynchroniczna nieblokująca 109
Zalety 110
Wady 111
Gdzie stosować wzorzec? 112
Wzorzec komunikacja za pośrednictwem współdzielonych danych 112
Implementacja 113
Zalety 113
Wady 114
Gdzie stosować wzorzec? 115
Wzorzec komunikacja żądanie - odpowiedź 115
Implementacja: komunikacja synchroniczna kontra asynchroniczna 116
Gdzie stosować wzorzec? 118
Wzorzec komunikacja sterowana zdarzeniami 119
Implementacja 120
Co jest wewnątrz zdarzenia? 122
Gdzie stosować wzorzec? 125
Zachowaj ostrożność 125
Podsumowanie 127

**CZĘŚĆ II. Implementacja 129**

**5. Implementacja komunikacji mikrousług 131**
Poszukiwanie idealnej technologii 131
Łatwość zachowania zgodności wstecz 131
Zdefiniuj interfejs w sposób jawny 131
Zachowaj niezależność technologii interfejsów API 132

Spraw, aby Twoja usługa była prosta dla konsumentów 132
Ukryj szczegóły wewnętrznej implementacji 132
Wybór technologii 133
Zdalne wywołania procedur 133
REST 137
GraphQL 142
Brokery wiadomości 144
Formaty serializacji 149
Formaty tekstowe 149
Formaty binarne 150
Schematy 150
Strukturalne i semantyczne naruszenia kontraktu 151
Czy należy używać schematów? 151
Obsługa zmian między mikrousługami 152
Unikanie zmian naruszających kontrakt 152
Zmiany rozszerzające 153
Tolerancyjny konsument 153
Właściwa technologia 154
Jawny interfejs 155
Wczesne wykrywanie zmian naruszających kontrakt 156
Zarządzanie zmianami naruszającymi zgodność wstecz 157
Wdrażanie lockstep 157
Współistnienie niezgodnych ze sobą wersji mikrousług 157
Emulowanie starego interfejsu 158
Jakie podejście preferuję? 160
Umowa społeczna 160
Śledzenie użycia 161
Środki ekstremalne 162
Zasada DRY i niebezpieczeństwa wielokrotnego wykorzystywania kodu w świecie mikrousług 162
Udostępnianie kodu za pośrednictwem bibliotek 163
Wykrywanie usług 165
DNS 165
Dynamiczne rejestry usług 167
Nie zapomnij o ludziach 169
Siatki usług i bramy interfejsów API 170
Bramy API 171
Siatki usług 173
A co z innymi protokołami? 177
Dokumentowanie usług 177
Jawne schematy 177
System samoopisujący się 178
Podsumowanie 181

**6. Przepływy pracy 182**
Transakcje bazodanowe 182
Transakcje ACID 182

Nadal ACID, ale bez niepodzielności? 183
Transakcje rozproszone — dwufazowe zatwierdzanie 185
Transakcje rozproszone — po prostu powiedz „nie" 187
Sagi 188
Tryby awarii dla sag 189
Implementacja sag 194
Sagi a transakcje rozproszone 200
Podsumowanie 200

**7. Budowanie 202**
Krótkie wprowadzenie do ciągłej integracji 202
Czy rzeczywiście stosujesz mechanizmy CI? 203
Modele rozgałęziania 204
Potoki budowania a ciągłe dostawy 205
Narzędzia 208
Kompromisy i środowiska 208
Tworzenie artefaktów 209
Mapowanie kodu źródłowego i kompilacji na mikrousługi 210
Jedno gigantyczne repozytorium, jedna gigantyczna kompilacja 210
Wzorzec jedno repozytorium na mikrousługę (tzw. multirepo) 212
Wzorzec monorepo 215
Jakie podejście bym zastosował? 220
Podsumowanie 221

**8. Wdrażanie 222**
Od widoku logicznego do fizycznego 222
Wiele egzemplarzy 223
Baza danych 224
Środowiska 227
Zasady wdrażania mikrousług 230
Odizolowane uruchamianie 231
Koncentracja na automatyzacji 233
Infrastruktura jako kod (IaC) 234
Wdrażanie bez przestojów 235
Zarządzanie pożądanym stanem 236
Opcje wdrażania 239
Maszyny fizyczne 240
Maszyny wirtualne 240
Kontenery 243
Kontenery aplikacji 248
Platforma jako usługa (PaaS) 249
Funkcja jako usługa (FaaS) 251
Która opcja wdrażania jest dla Ciebie odpowiednia? 258
Kubernetes i orkiestracja kontenerów 260
Przypadek orkiestracji kontenerów 260
Uproszczony widok pojęć związanych z Kubernetes 261
Wielodostępność i federacja 262

Cloud Native Computing Federation (CNCF) 265
Platformy i przenośność 266
Heim, Operator, CRD. O mój Boże! 266
I jeszcze Knative 267
Przyszłość 268
Czy powinieneś korzystać z Kubernetes? 268
Dostawy progresywne 269
Oddzielenie wdrożenia od wydania 270
Na drodze do dostaw progresywnych 270
Przełączniki funkcji 271
Wydania kanarkowe 271
Uruchamianie równoległe 272
Podsumowanie 273

**9. Testowanie 275**
Rodzaje testów 275
Zakres testów 277
Testy jednostkowe 279
Testy usług 280
Testy od końca do końca 281
Kompromisy 281
Implementacja testów usług 282
Mocki czy namiastki usług 283
Inteligentniejsza namiastka usługi 284
Kłopotliwe testy od końca do końca 284
Testy kruche i łamliwe 286
Kto pisze testy od końca do końca? 287
Jak długo? 289
Piętrzące się zaległości 290
Metawersje 290
Brak niezależnej testowalności 291
Czy należy unikać testów od końca do końca? 291
Testy kontraktu oraz kontrakty konsumenckie 292
Czy należy używać testów od końca do końca? 295
Wygoda pracy programistów 295
Od fazy przedprodukcyjnej do testowania w produkcji 296
Rodzaje testów w produkcji 297
Bezpieczeństwo testowania w produkcji 298
Średni czas do naprawy kontra średni czas między awariami 298
Testy współzależności funkcjonalnych 299
Testy wydajności 300
Testy wytrzymałości 301
Podsumowanie 302

**10. Od monitorowania do obserwowalności 303**
Niepokój, panika i zamieszanie 303
Jedna usługa, jeden serwer 304

Jedna usługa, wiele serwerów 305
Wiele usług, wiele serwerów 306
Obserwowalność a monitorowanie 307
Filary obserwowalności? Nie tak szybko 307
Elementy składowe obserwowalności 308
Agregacja logów 309
Agregacja metryk 318
Rozproszone śledzenie 321
Czy postępujemy właściwie? 323
Ostrzeganie 325
Monitorowanie sentantyczne 329
Testowanie w produkcji 330
Standaryzacja 333
Wybór narzędzi 333
Wybór powinien być demokratyczny 334
Wybieraj narzędzia łatwe do integracji 334
Zapewniaj odpowiedni kontekst 334
Informacje w czasie rzeczywistym 335
Informacje odpowiednie dla Twojej skali 335
Maszynowy ekspert 336
Od czego zacząć? 337
Podsumowanie 337

**11. Bezpieczeństwo 339**
Podstawowe zasady 340
Zasada najmniejszych uprawnień 341
Obrona w głąb 341
Automatyzacja 342
Wbuduj zabezpieczenia w proces dostaw 343
Pięć funkcji cyberbezpieczeństwa 344
Identyfikacja 344
Ochrona 346
Wykrywanie 346
Reagowanie 346
Odtwarzanie 347
Podstawy zabezpieczeń aplikacji 347
Poświadczenia 347
Łatki bezpieczeństwa 353
Kopie zapasowe 355
Odbudowa 357
Zaufanie domyślne kontra zaufanie zerowe 358
Zaufanie domyślne 358
Zaufanie zerowe 358
To jest pasmo 359
Zabezpieczanie danych 361
Dane podczas przesyłania 361
Zabezpieczanie danych w spoczynku 364

Uwierzytelnianie i autoryzacja 367
Uwierzytelnianie między usługami 367
Uwierzytelnianie użytkowników 367
Popularne implementacje pojedynczego logowania 368
Brama pojedynczego logowania 369
Szczegółowa autoryzacja 370
Problem zdezorientowanego zastępcy 371
Scentralizowana autoryzacja w górze strumienia przetwarzania 373
Autoryzacja zdecentralizowana 373
Tokeny JWT 374
Podsumowanie 377

**12. Niezawodność 378**
Co to jest niezawodność? 378
Solidność 379
Zdolność do odtwarzania 380
Rozszerzalność z wdziękiem 380
Trwałe zdolności adaptacyjne 381
Architektura mikrousług 382
Awarie zdarzają się wszędzie 382
Jak wiele to zbyt wiele? 383
Degradowanie funkcjonalności 384
Wzorce stabilności 385
Limity czasu 388
Ponowienia prób 389
Grodzie 390
Bezpieczniki 391
Izolacja 394
Redundancja 395
Middleware 395
Idempotencja 396
Rozłożenie ryzyka 397
Twierdzenie CAP 398
Poświęcenie spójności 400
Poświęcenie dostępności 400
Poświęcenie tolerancji podziału? 401
AP czy CP? 401
To nie jest zasada „wszystko albo nic" 401
Świat rzeczywisty 402
Inżynieria chaosu 403
Dni ćwiczeń 404
Eksperymenty produkcyjne 404
Wykraczając poza solidność 405
Szukanie winnych 405
Podsumowanie 407

**13. Skalowanie** **408**
Cztery osie skalowania 408
Skalowanie pionowe 409
Implementacja 410
Najważniejsze korzyści 410
.Ograniczenia 411
Zwielokrotnianie w poziomie 411
Partycjonowanie danych 414
Dekompozycja funkcjonalna 418
Łączenie modeli 420
Zacznij od małych rozmiarów 422
Buforowanie 423
Buforowanie w celu poprawy wydajności 424
Buforowanie w celu skalowania 424
Buforowanie w celu poprawy niezawodności 424
Gdzie buforować 425
Unieważnianie 429
Złota zasada buforowania 434
Aktualność danych a optymalizacja 435
Zatrucie pamięcią podręczną — historia ku przestrodze 435
Autoskalowanie 436
Zaczynanie od nowa 438
Podsumowanie 439

**CZĘŚĆ III. Ludzie** **441**

**14. Interfejsy użytkownika** **443**
W stronę środowiska cyfrowego 444
Modele własności 444
Przesłanki dla tworzenia dedykowanych zespołów frontendowych 446
Zespoły dopasowane do strumienia przetwarzania 446
Współdzielenie specjalistów 447
Zapewnienie spójności 449
Pokonywanie technicznych wyzwań 450
Wzorzec monolityczny frontend 450
Kiedy należy korzystać ze wzorca? 451
Wzorzec mikrofrontend 452
Implementacja 452
Kiedy stosować wzorzec? 453
Wzorzec dekompozycja na bazie stron 454
Gdzie stosować wzorzec? 455
Współdzielenie specjalistów 447
Zapewnienie spójności 449
Pokonywanie technicznych wyzwań 450
Wzorzec monolityczny frontend 450
Kiedy należy korzystać ze wzorca? 451
Wzorzec mikrofrontend 452

Implementacja 452
Kiedy stosować wzorzec? 453
Wzorzec dekompozycja na bazie stron 454
Wzorzec dekompozycja oparta na widżetach 455
Implementacja 457
Kiedy korzystać ze wzorca? 459
Ograniczenia 460
Wzorzec centralna brama agregująca 461
Własność 462
Różne typy interfejsów użytkownika 463
Wiele obaw 464
Kiedy korzystać ze wzorca? 465
Wzorzec backend dla frontendu (BFF) 465
Ile komponentów BFF? 467
Wielokrotne użycie kodu a BFF 469
BFF dla desktopowego interfejsu webowego i nie tylko 472
Kiedy korzystać ze wzorca? 473
GraphQL 473
Podejście hybrydowe 475
Podsumowanie 475

**15. Struktury organizacyjne 476**
Organizacje luźno sprzężone 476
Prawo Conwaya 477
Dowody 478
Wielkość zespołu 479
Zrozumieć prawo Conwaya 480
Małe zespoły, duża organizacja 481
O autonomii 482
Własność silna kontra własność kolektywna 483
Własność silna 484
Własność kolektywna 485
Na poziomie zespołu kontra na poziomie organizacji 486
Równoważenie modeli 486
Zespoły wspomagające 487
Społeczności praktyków 489
Platforma 490
Mikrousługi współdzielone 492
Zbyt trudne do rozdzielenia 492
Przekrojowe zmiany 493
Wąskie gardła dostaw 493
Wewnętrzne open source 494
Rola opiekunów 495
Dojrzałość 495
Narzędzia 495
Mikrousługi modułowe 496
Przeglądy zmian 498

Usługa osierocona 501
Studium przypadku: RealEstate.com.au 502
Rozproszenie geograficzne 503
Odwrócone prawo Conwaya 504
Ludzie 505
Podsumowanie 506

**16. Ewolucyjny architekt 507**
Co oznacza ta nazwa? 507
Czym jest architektura oprogramowania? 509
Umożliwienie wprowadzania zmian 510
Ewolucyjna wizja architekta 511
Definiowanie granic systemowych 512
Konstrukt społeczny 514
Warunki do „zamieszkiwania” 515
Pryncypialne podejście 516
Cele strategiczne 517
Zasady 517
Praktyki 518
Łączenie zasad i praktyk 518
Praktyczny przykład 518
Kierowanie architekturą ewolucyjną 519
Architektura w organizacji dostosowanej do strumienia przetwarzania 520
Budowanie zespołu 523
Wymagane standardy 523
Monitorowanie 524
Interfejsy 524
Bezpieczeństwo architektury 524
Zarządzanie i droga utwardzona 525
Przykładowe egzemplarze 526
Spersonalizowany szablon usługi 526
Utwardzona droga na dużą skalę 527
Dług techniczny 528
Obsługa wyjątków 528
Podsumowanie 529

**Posłowie: mikrousługi w pigułce 531**

**Bibliografia 542**

**Glosariusz 547**

oprac. BPK