

Wykorzystaj potencjał architektury usług!

Architektura mikrousług to sposób na odejście od dużych, monolitycznych aplikacji. Wspecjalizowane usługi realizujące konkretne zadania i komunikujące się z otoczeniem pozwalają na lepsze zapanowanie nad kodem, są łatwiejsze do przetestowania oraz bardziej elastyczne. Jednak oprócz zalet mają też wady. Sięgnij po tę książkę i dowiedz się, jak najlepiej radzić sobie z architekturą mikrousług!

Autor przedstawia w książce skuteczne techniki projektowania i korzystania z architektury mikrousług. W trakcie lektury kolejnych rozdziałów poznasz w szczególności ideę mikrousług, korzyści ich stosowania, sposoby modelowania usług oraz skuteczne techniki dzielenia dużej aplikacji na mikrousługi. Ponadto zapoznasz się z możliwymi sposobami integracji: zdalne wywołanie procedur, **REST** i zdarzenia — to tylko niektóre z poruszanych kwestii. Na sam koniec zaznajomisz się z najlepszymi metodami testowania i monitorowania usług, zapewnisz im bezpieczeństwo dzięki kluczem **API** oraz innym technikom. Ta książka jest obowiązkową lekturą dla wszystkich osób chcących tworzyć nowoczesne systemy bazujące na architekturze mikrousług.

- Odkryj, jak można dostosować projekt systemu do celów Twojej organizacji stosując architekturę mikrousług

- Zapoznaj się z możliwościami integracji usług z pozostałą częścią systemu

- Zastosuj przyrostowe podejście do podziału monolitycznych baz kodu

- Wdrażaj pojedyncze mikrousługi korzystając z techniki ciągłej integracji

- Zbadaj złożoność testowania i monitorowania rozproszonych usług

- Zarządzaj zabezpieczeniami stosując modele użytkownik-usługa oraz usługa-usługa

- Zapoznaj się z wyzwaniem skalowania architektury mikrousług

Przekonaj się, jak architektura mikrousług zmieni Twoje spojrzenie na aplikacje!

Sam Newman — technolog w firmie ThoughtWorks odpowiedzialny za wspomaganie klientów oraz architekturę wewnętrznych systemów. Prelegent, autor artykułów dla wydawnictwa O'Reilly. Programista języków Java oraz Python.

Przedmowa (13)

1. Mikrousługi (19)

- Czym są mikrousługi? (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)
- Pryncypialne 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)
 - Luźne 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)
 - Dbłość 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)
- Zdalne wywołania procedur (64)
 - Sprzężenia technologii (64)
 - Wywołania lokalne różnią się od zdalnych (65)
 - Kruczość (65)
 - Czy wywołania RPC są złym rozwiązaniem? (67)

- REST (67)
 - REST a HTTP (68)
 - Hipermedium jako silnik stanu aplikacji (68)
 - JSON, XML czy coś innego? (70)
 - Uważaj na zbyt wielkie wygody (71)
 - Wady interfejsu REST przez HTTP (72)
- Implementacja współpracy asynchronicznej, bazującej na zdarzeniach (73)
 - Opcje wyboru technologii (73)
 - Zawiłości architektur asynchronicznych (74)
- Usługi jako maszyny stanów (76)
- Rozszerzenia reaktywne (76)
- DRY i perypetie wielokrotnego wykorzystania kodu w świecie mikrousług (77)
 - Biblioteki klienckie (77)
- Dostęp przez referencję (78)
- Zarządzanie wersjami (80)
 - Odkładaj modyfikowanie interfejsu tak długo, jak to możliwe (80)
 - Wczesne wychwytywanie zmian naruszających zgodność interfejsu (81)
 - Zastosowanie semantycznej kontroli wersji (81)
 - Współlistnienie różnych punktów końcowych (82)
 - Korzystanie z wielu równoległych wersji usługi (83)
- Interfejsy użytkownika (84)
 - W stronę środowiska cyfrowego (85)
 - Ograniczenia (85)
 - Kompozycja interfejsów API (86)
 - Kompozycja fragmentu interfejsu użytkownika (87)
 - Zaplecza dla frontonów (89)
 - Podjęcie hybrydowe (90)
- Integracja z oprogramowaniem zewnętrznych producentów (91)
 - Brak kontroli (92)
 - Personalizacja (92)
 - Makaron integracji (92)
 - Personalizacja na własnych warunkach (93)
 - Wzorzec Dusiciel (95)
- 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 ciągłe 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 gości (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)
- Testuj ścieżki, a nie historie (161)
- Testy sterowane potrzebami konsumentów (162)
 - Pact (163)
 - Konwersacje (165)
- Czy należy używać testów od końca do końca? (165)
- Testowanie po opublikowaniu systemu do produkcji (166)
 - Oddzielenie wdrożenia od publikacji (166)
 - Publikacje kanarkowe (167)
 - Średni czas do naprawy kontra średni czas między awariami (168)
- Testy współzależności funkcjonalnych (169)
 - Testy wydajności (170)
- Podsumowanie (171)

8. Monitorowanie (173)

- Jedna usługa, jeden serwer (174)
- Jedna usługa, wiele serwerów (174)
- Wiele usług, wiele serwerów (175)
- Logi, logi i jeszcze raz logi... (176)
- Śledzenie metryk dotyczących wielu usług (177)
- Metryki usług (178)
- Monitorowanie syntetyczne (178)
 - Implementacja monitorowania semantycznego (179)
- Identyfikatory korelacji (180)
- Kaskada (182)
- Standaryzacja (182)
- Weź pod uwagę użytkowników (183)
- Przyszłość (184)
- Podsumowanie (185)

9. Bezpieczeństwo (187)

- Uwierzytelnianie i autoryzacja (187)
 - Popularne implementacje pojedynczego logowania (188)
 - Brama pojedynczego logowania (189)
 - Szczegółowa autoryzacja (190)

- Uwierzytelnianie i autoryzacja w trybie usługa-usługa (191)
 - Zezwalaj na wszystko wewnątrz obszaru (191)
 - Podstawowe uwierzytelnianie HTTP(S) (192)
 - Korzystanie z SAML lub OpenID Connect (192)
 - Certyfikaty klienta (193)
 - HMAC przez HTTP (194)
 - Klucze API (195)
 - Problem zastępcy (195)
- Zabezpieczanie danych w spoczynku (197)
 - Korzystaj ze sprawdzonych sposobów (198)
 - Wszystko dotyczy kluczy (198)
 - Wybierz swoje cele (199)
 - Odszyfruj dane na żądanie (199)
 - Szyfruj kopie zapasowe (199)
- Obrona wielostrefowa (199)
 - Zapory firewall (199)
 - Rejestrowanie (200)
 - Systemy wykrywania włamań (i zapobiegania im) (200)
 - Podział sieci (200)
 - System operacyjny (201)
- Praktyczny przykład (201)
- Bądź oszczędny (204)
- Czynnik ludzki (204)
- Złota reguła (204)
- Wdrażanie zabezpieczeń (205)
- Zewnętrzna weryfikacja (206)
- Podsumowanie (206)

10. Prawo Conwaya a projektowanie systemów (207)

- Dowody (207)
 - Organizacje sprzężone luźno i ściśle (208)
 - Windows Vista (208)
- Netflix i Amazon (208)
- Co można z tym zrobić? (209)
- Dostosowanie się do ścieżek komunikacyjnych (209)
- Własność usługi (210)
- Powody współdzielenia usług (211)
 - Zbyt trudne do rozdzielenia (211)
 - Zespoły funkcyjne (211)
 - Wąskie gardła dostaw (212)
- Wewnętrzne Open Source (212)
 - Rola opiekunów (213)
 - Dojrzałość (213)
 - Narzędzia (214)
- Konteksty ograniczone a struktura zespołów (214)
- Usługa osierocona? (214)
- Studium przypadku: RealEstate.com.au (215)
- Odwrócone prawo Conwaya (216)
- Ludzie (217)
- Podsumowanie (218)

11. Mikrouslugi w projektach duzej skali (219)

- Awarie zdarzaja sie wszedzie (219)
- Jak wiele jest zbyt wiele? (220)
- Degradowanie funkcjonalnosci (221)
- Srodki bezpieczenstwa architektury (222)
- Antykrucha organizacja (224)
 - Limity czasu (225)
 - Bezpieczniki (225)
 - Grodzie (227)
 - Izolacja (229)
- Idempotencja (229)
- Skalowanie (230)
 - Zwiekszenie rozmiarow (230)
 - Podzial obciazen (231)
 - Rozlozenie ryzyka (231)
 - Rownowazenie obciazenia (232)
 - Systemy bazujace na watkach roboczych (234)
 - Zaczynanie od nowa (235)
- Skalowanie baz danych (236)
 - Dostepnosc uslugi kontra trwaosc danych (236)
 - Skalowanie do obslugi operacji odczytu (236)
 - Skalowanie do obslugi operacji zapisu (237)
 - Wspolna 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 niezawodnosci (242)
 - Ukrywanie zrodla (242)
 - Zachowaj prostote (243)
 - Zatrucie pamiecia 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 uslug (250)
 - DNS (251)
- Dynamiczne rejestry uslug (252)
 - Zookeeper (252)
 - Consul (253)
 - Eureka (254)
 - Tworzenie własnych rozwiązań (254)
 - Nie zapomnij o ludziach! (255)
- Dokumentowanie uslug (255)

- Swagger (256)
- HAL i przeglądarka HAL (256)
- System samoopisujący się (257)
- Podsumowanie (258)

12. Podsumowanie (259)

- Zasady dotyczące mikrouslug (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 mikrouslug? (263)
- Ostatnie słowo (264)

Skorowidz (265)