

Wydanie II

# Podręcznik architekta rozwiązań

Poznaj reguły oraz strategie  
projektu architektury  
i rozpocznij niezwykłą karierę

**Saurabh Shrivastava**  
**Neelanjali Srivastav**



Helion 

**Packt** >

Tytuł oryginału: Solutions Architect's Handbook: Kick-start your career as a solutions architect by learning architecture design principles and strategies,  
2nd Edition

Tłumaczenie: Robert Górczyński

ISBN: 978-83-8322-479-4

Copyright © Packt Publishing 2022. First published in the English language under the title 'Solution Architect's Handbook - Second Edition - (9781801816618)'

Polish edition copyright © 2023 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/podar2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

# Spis treści |

<b>O autorach</b> .....	<b>15</b>
<b>O korektorze merytorycznym</b> .....	<b>16</b>
<b>Przedmowa</b> .....	<b>17</b>
<b>Wprowadzenie</b> .....	<b>23</b>
<b>ROZDZIAŁ 1.</b>	
<b>Rola architekta rozwiązań</b> .....	<b>29</b>
Czym jest architektura rozwiązania? .....	30
Ewolucja architektury rozwiązania .....	34
Dlaczego architektura rozwiązania jest ważna? .....	36
Korzyści płynące z architektury rozwiązania .....	37
Odpowiedź na potrzeby biznesowe i jakość dostawy .....	38
Wybór najlepszej platformy technologicznej .....	39
Uwzględnianie ograniczeń rozwiązania i problemów .....	40
Pomoc w zarządzaniu zasobami i kosztami .....	41
Zarządzanie dostarczaniem rozwiązania i cyklu życiowego projektu .....	41
Uwzględnienie wymagań innych niż funkcjonalne .....	42
Architektura rozwiązania w publicznej chmurze .....	44
Czym jest publiczna chmura? .....	44
Chmura publiczna, prywatna i hybrydowa .....	45
Architektura chmury publicznej .....	46
Architektura dla natywnej chmury .....	48
Dostawcy chmury publicznej i oferowane przez nich usługi .....	49
Podsumowanie .....	50
<b>ROZDZIAŁ 2.</b>	
<b>Architekt rozwiązań w organizacji</b> .....	<b>51</b>
Rodzaje roli architekta rozwiązań .....	52
Role ogólnego architekta rozwiązań .....	52
Specjalistyczne role architekta rozwiązań .....	57

Obowiązki architekta rozwiązań .....	63
Analiza wymagań użytkownika .....	64
Definiowanie wymagań innych niż funkcjonalne .....	65
Zrozumienie i zaangażowanie interesariuszy .....	67
Poznanie różnych ograniczeń architektonicznych .....	68
Wybór technologii .....	70
Prototypowanie i utworzenie dowodu koncepcji .....	71
Zaprojektowanie i dostarczenie rozwiązania .....	71
Zapewnienie działania produktu po jego uruchomieniu i obsługa techniczna .....	74
Promowanie technologii .....	75
Architekt rozwiązań w organizacji stosującej metody zwinne .....	75
Dlaczego warto stosować metody zwinne? .....	75
Manifest metod zwinnych .....	77
Terminologia i proces metod zwinnych .....	78
Podsumowanie .....	83

### ROZDZIAŁ 3.

<b>Atrybuty architekta rozwiązań .....</b>	<b>84</b>
Skalowalność i elastyczność .....	85
Kwestie pojemności podczas skalowania .....	86
Skalowanie architektury .....	87
Skalowanie treści statycznej .....	88
Elastyczność floty serwerów .....	89
Skalowanie bazy danych .....	90
Wysoka dostępność i niezawodność .....	91
Odporność na awarie i nadmiarowość .....	93
Odzyskiwanie po awarii i kontynuowanie działalności biznesowej .....	95
Rozszerzalność i wielokrotne użycie .....	97
Użyteczność i dostępność .....	99
Przenośność i współdziałanie .....	101
Doskonałe działanie i łatwość obsługi technicznej rozwiązania .....	102
Zapewnianie bezpieczeństwa i zgodności .....	103
Uwierzytelnianie i autoryzacja .....	104
Bezpieczeństwo internetowe .....	105
Bezpieczeństwo sieci .....	105
Bezpieczeństwo infrastruktury .....	106
Bezpieczeństwo danych .....	106
Optymalizacja kosztów i budżetu .....	107
Podsumowanie .....	108

**ROZDZIAŁ 4.**

<b>Reguły projektu architektury rozwiązań .....</b>	<b>110</b>
Skalowanie obciążenia .....	111
Skalowanie prognostyczne .....	112
Skalowanie reakcyjne .....	114
Tworzenie niezawodnej architektury .....	115
Projektowanie pod kątem wydajności działania .....	118
Używanie zasobów możliwych do zastępowania .....	120
Tworzenie infrastruktury niemodyfikowalnej .....	120
Luźne powiązanie komponentów .....	122
Położenie nacisku na usługę zamiast na serwer .....	124
Używanie pamięci masowej odpowiedniej dla danych wymagań .....	126
Tworzenie projektu pod kątem danych .....	129
Pokonywanie ograniczeń architektonicznych .....	131
Produkt o minimalnej niezbędnej funkcjonalności .....	132
Zapewnianie bezpieczeństwa w każdym miejscu .....	133
Stosowanie automatyzacji, gdzie się da .....	134
Podsumowanie .....	136

**ROZDZIAŁ 5.**

<b>Migracja do chmury i projekt hybrydowej architektury chmury .....</b>	<b>137</b>
Korzyści oferowane przez architekturę natywnej chmury .....	138
Popularne opcje w zakresie dostawców publicznej chmury .....	141
Opracowanie strategii migracji do chmury .....	142
Migracja typu podniesienie i przesunięcie .....	144
Natywna chmura .....	146
Wycofanie lub pozostawienie .....	148
Wybór strategii dotyczącej chmury .....	150
Etapy migracji do chmury .....	152
Odkrywanie obciążenia .....	153
Analiza informacji .....	155
Tworzenie planu migracji .....	157
Projektowanie aplikacji .....	161
Migracja aplikacji do chmury .....	165
Integracja, weryfikacja i zakończenie procesu migracji .....	169
Działanie aplikacji w chmurze .....	171
Optymalizacja aplikacji w chmurze .....	173
Tworzenie architektury chmury hybrydowej .....	175
Podejście polegające na korzystaniu z usług wielu dostawców chmury .....	177

Projektowanie architektury natywnej chmury .....	178
Podsumowanie .....	180
Dalsza lektura .....	181

## ROZDZIAŁ 6.

### Wzorce projektowe architektury rozwiązań ..... 182

Tworzenie architektury n-warstwowej .....	183
Warstwa internetowa .....	184
Warstwa aplikacji .....	185
Warstwa bazy danych .....	185
Tworzenie architektury wielodostępnej na bazie SaaS .....	187
Tworzenie projektów architektury stanowej i bezstanowej .....	189
Architektura zorientowana na usługi .....	191
Usługa sieciowa w architekturze SOAP .....	192
Usługa sieciowa w architekturze RESTful .....	196
Tworzenie bazującej na architekturze SOAP witryny internetowej typu e-commerce .....	198
Budowanie architektury niewymagającej serwera .....	199
Tworzenie architektury mikrousług .....	202
Architektura aplikacji obsługującej głosowanie w czasie rzeczywistym .....	204
Budowanie architektury bazującej na kolejce .....	205
Wzorzec łańcucha kolejkowania .....	206
Wzorzec obserwacji zadania .....	208
Tworzenie architektury opartej na zdarzeniach .....	209
Model producenta i subskrybenta .....	210
Model strumienia zdarzeń .....	211
Budowanie architektury bazującej na buforze .....	212
Wzorce buforowania w architekturze trójwarstwowej .....	215
Wzorzec zmiany nazwy .....	216
Wzorzec serwera proxy .....	217
Wzorzec przepisanego proxy .....	218
Wzorzec buforowania aplikacji .....	219
Poznajemy wzorzec bezpiecznika .....	222
Implementacja wzorca grodziowego .....	222
Tworzenie wzorca pływających adresów IP .....	224
Wdrażanie aplikacji w kontenerze .....	225
Korzyści oferowane przez kontenery .....	226
Wdrażanie z użyciem kontenerów .....	228
Tworzenie architektury bazującej na kontenerach .....	230

Obsługa bazy danych w architekturze aplikacji .....	231
Wzorzec wysokiej dostępności bazy danych .....	233
Unikanie antywzorców w architekturze rozwiązania .....	235
Podsumowanie .....	237

## **ROZDZIAŁ 7.**

<b>Kwestie dotyczące wydajności działania .....</b>	<b>239</b>
Reguły projektowe związane z wydajnością działania architektury .....	240
Zmniejszenie opóźnienia .....	241
Poprawienie przepustowości .....	242
Obsługa współbieżności .....	244
Stosowanie buforowania .....	246
Technologia stosowana podczas optymalizacji wydajności działania .....	247
Wybór dotyczący obliczeń .....	248
Wybór pamięci masowej .....	255
Wybór bazy danych .....	260
Usprawnianie wydajności działania sieci .....	264
Monitorowanie wydajności działania .....	269
Podsumowanie .....	270

## **ROZDZIAŁ 8.**

<b>Kwestie dotyczące zapewnienia bezpieczeństwa .....</b>	<b>272</b>
Reguły projektowe dotyczące bezpieczeństwa architektury .....	273
Implementacja uwierzytelniania i autoryzacji .....	273
Stosowanie bezpieczeństwa wszędzie .....	274
Ograniczenie pola rażenia .....	274
Nieustanne monitorowanie i audyt wszystkiego .....	275
Automatyzacja na każdym kroku .....	275
Ochrona danych .....	276
Reakcja na incydenty dotyczące bezpieczeństwa .....	277
Wybór technologii podczas zabezpieczania architektury .....	277
Tożsamość użytkownika i zarządzanie dostępem .....	277
Bezpieczeństwo warstwy sieciowej .....	288
Zabezpieczenie aplikacji i jej infrastruktury .....	295
Bezpieczeństwo danych .....	301
Certyfikaty bezpieczeństwa i zgodności .....	309
Model współdzielonej odpowiedzialności za bezpieczeństwo w chmurze .....	310
Podsumowanie .....	312

**ROZDZIAŁ 9.**

<b>Kwestie dotyczące niezawodności architektury .....</b>	<b>314</b>
Reguły projektowe zapewniające niezawodność architektury .....	315
Automatyczna naprawa systemu po awarii .....	315
Stosowanie automatyzacji .....	316
Tworzenie systemu rozproszonego .....	317
Monitorowanie i zwiększanie pojemności .....	317
Weryfikacja procedury odzyskiwania po awarii .....	318
Technologia pomagająca zapewnić niezawodność architektury .....	319
Planowanie pod kątem docelowego punktu odzyskiwania i docelowego czasu odzyskiwania .....	319
Replikacja danych .....	321
Planowanie procedury odzyskiwania po awarii .....	323
Stosowanie najlepszych praktyk związanych z procedurą odzyskiwania po awarii .....	334
Poprawianie niezawodności za pomocą chmury .....	335
Podsumowanie .....	337

**ROZDZIAŁ 10.**

<b>Kwestie dotyczące zapewnienia sprawnego działania .....</b>	<b>338</b>
Reguły projektowe dotyczące zapewnienia sprawnego działania .....	339
Automatyzacja ręcznie wykonywanych zadań .....	339
Wprowadzanie zmian przyrostowo i z możliwością ich wycofania .....	340
Przewidywanie awarii i reagowanie na nie .....	341
Uczenie się na błędach .....	341
Uaktualnianie scenariusza działania .....	341
Wybrane technologie pozwalające zapewnić sprawne działanie .....	342
Planowanie w celu osiągnięcia sprawnego działania .....	343
Funkcjonowanie sprawności operacyjnej .....	349
Usprawnianie działania systemu .....	359
Osiągnięcie sprawnego działania w chmurze publicznej .....	363
Podsumowanie .....	365

**ROZDZIAŁ 11.**

<b>Kwestie dotyczące kosztów .....</b>	<b>366</b>
Reguły projektowe dotyczące optymalizacji kosztu .....	367
Obliczenie całkowitego kosztu własności .....	367
Planowanie przez oszacowanie budżetu i prognozowanie wydatków .....	369
Zarządzanie zapotrzebowaniem i katalogiem usług .....	371
Śledzenie nakładów kapitałowych .....	372
Nieustanna optymalizacja kosztu .....	373



Techniki optymalizacji kosztu .....	373
Zmniejszenie złożoności architekuralnej .....	374
Zwiększenie efektywności IT .....	376
Standaryzacja i zarządzanie .....	378
Monitorowanie kosztu użycia i raportowanie .....	381
Optymalizacja kosztu w publicznej chmurze .....	385
Podsumowanie .....	387

## **ROZDZIAŁ 12.**

<b>Framework DevOps i architektury rozwiązań .....</b>	<b>388</b>
Wprowadzenie do kultury DevOps .....	389
Omówienie korzyści oferowanych przez DevOps .....	390
Omówienie komponentów DevOps .....	392
Ciągła integracja i ciągłe wdrażanie .....	392
Ciągłe monitorowanie i usprawnianie .....	395
Infrastruktura jako kod .....	396
Zarządzanie konfiguracją .....	398
DevOps i zapewnienie bezpieczeństwa .....	401
Połączenie DevOps i CI/CD .....	402
Implementacja strategii CD .....	404
Wdrożenie w miejscu .....	405
Wdrożenie ciągle .....	405
Wdrożenie typu niebieski-zielony .....	405
Wdrożenie typu czerwony-czarny .....	407
Wdrożenie niemodyfikowalne .....	408
Implementacja ciągłego testowania w potoku CI/CD .....	408
Testy A/B .....	410
Używanie narzędzi DevOps w potoku CI/CD .....	411
Edytor kodu źródłowego .....	412
Zarządzanie kodem źródłowym .....	412
Serwer ciągłej integracji .....	413
Wdrożenie kodu .....	415
Potok kodu źródłowego .....	417
Implementacja najlepszych praktyk DevOps .....	418
Tworzenie w chmurze rozwiązań bazujących na DevOps i DevSecOps .....	420
Podsumowanie .....	422

**ROZDZIAŁ 13.**

<b>Inżynieria danych w architekturze rozwiązań .....</b>	<b>424</b>
Architektura big data .....	425
Projektowanie pod kątem potoku przetwarzania big data .....	428
Pobieranie danych .....	430
Technologie pobierania danych .....	431
Pobieranie danych do chmury .....	432
Przechowywanie danych .....	434
Technologia związana z przechowywaniem danych .....	436
Przetwarzanie danych i ich analizowanie .....	446
Rozwiązania technologiczne przeznaczone do przetwarzania i analizowania danych .....	447
Wizualizacja danych .....	451
Rozwiązania technologiczne na potrzeby wizualizacji danych .....	452
Projektowanie architektury big data .....	453
Architektura jeziora danych .....	455
Architektura repozytorium danych .....	459
Architektura mesh .....	462
Architektura danych strumieniowanych .....	464
Najlepsze praktyki dotyczące architektury big data .....	465
Podsumowanie .....	468

**ROZDZIAŁ 14.**

<b>Architektura uczenia maszynowego .....</b>	<b>470</b>
Czym jest uczenie maszynowe? .....	471
Nauka i uczenie maszynowe .....	473
Ocena modelu ML — nadmierne kontra niedostateczne wytrenowanie .....	476
Nadzorowane i nienadzorowane modele uczenia maszynowego .....	477
Uczenie maszynowe w chmurze .....	479
Tworzenie architektury uczenia maszynowego .....	480
Przygotowanie i nadawanie etykiet .....	481
Wybór algorytmu i budowa modelu .....	482
Trenowanie i dostrajanie .....	482
Wdrażanie i zarządzanie .....	483
Przykładowa architektura uczenia maszynowego .....	484
Podejście MLOps .....	487
Reguły MLOps .....	487
Najlepsze praktyki MLOps .....	488
Uczenie głębokie .....	490
Podsumowanie .....	493

**ROZDZIAŁ 15.**

<b>Architektura internetu rzeczy .....</b>	<b>495</b>
Czym jest internet rzeczy? .....	496
Komponenty architektury IoT .....	498
Zarządzanie urządzeniami IoT .....	499
Nawiązywanie połączenia i kontrolowanie urządzeń IoT .....	503
Przeprowadzanie analizy danych IoT .....	505
IoT w chmurze .....	507
Tworzenie przemysłowego rozwiązania IoT .....	509
Architektura IoT CF .....	511
Implementacja cyfrowego bliźniaka .....	513
Podsumowanie .....	515

**ROZDZIAŁ 16.**

<b>Obliczenia kwantowe .....</b>	<b>517</b>
Elementy konstrukcyjne komputera kwantowego .....	518
Kubit .....	518
Superpozycja .....	519
Splątanie .....	520
Mechanizm działania komputera kwantowego .....	521
Bramki kwantowe .....	522
Układy kwantowe .....	525
Typy komputerów kwantowych .....	528
Obliczenia kwantowe w rzeczywistych zastosowaniach .....	529
Obliczenia kwantowe w chmurze .....	530
Podsumowanie .....	531

**ROZDZIAŁ 17.**

<b>Przeprojektowywanie starych systemów .....</b>	<b>533</b>
Wyzwania związane ze starymi systemami .....	534
Trudność w sprostaniu oczekiwaniom użytkowników .....	536
Wyższy koszt obsługi technicznej i uaktualnień .....	537
Niedostateczne umiejętności i dokumentacja .....	537
Niedostosowanie do wymagań organizacji dotyczących zapewnienia bezpieczeństwa .....	538
Niezgodność z innymi systemami .....	539
Korzyści wynikające z modernizacji systemu .....	540
Definiowanie strategii modernizacji systemu .....	542
Analiza starej aplikacji .....	543
Modernizacja .....	544

Techniki modernizacji starych systemów .....	546
Hermetyzacja, rehosting i zmiana platformy .....	547
Refaktoryzacja i ponowne utworzenie architektury .....	548
Ponowne zaprojektowanie i zastąpienie .....	549
Definiowanie strategii migracji starych systemów do chmury .....	550
Dokumentacja i pomoc techniczna .....	551
Migracja rozwiązania typu mainframe do chmury .....	552
Migracja samodzielnej aplikacji .....	553
Migracja aplikacji zawierającej kod współdzielony .....	554
Podsumowanie .....	557

## ROZDZIAŁ 18.

<b>Dokumentowanie architektury rozwiązania .....</b>	<b>559</b>
Przeznaczenie dokumentu architektury rozwiązania .....	560
Widoki dokumentu architektury rozwiązania .....	561
Struktura dokumentu architektury rozwiązania .....	564
Ogólne omówienie rozwiązania .....	564
Kontekst biznesowy .....	566
Ogólne omówienie koncepcji rozwiązania .....	569
Architektura rozwiązania .....	570
Implementacja rozwiązania .....	574
Zarządzanie rozwiązaniem .....	575
Dodatek .....	575
Przygotowywanie dokumentacji IT na potrzeby architektury rozwiązania .....	576
Podsumowanie .....	577

## ROZDZIAŁ 19.

<b>Umiejętności pozwalające stać się lepszym architektem rozwiązań .....</b>	<b>578</b>
Umiejętności związane z sprzedażą .....	579
Komunikacja z kierownictwem najwyższego szczebla .....	581
Podejmowanie działań i ponoszenie odpowiedzialności za nie .....	583
Definiowanie strategii działania, celów oraz ważnych wyników .....	583
Myślenie szerszymi kategoriami .....	585
Elastyczność i umiejętność dostosowania się .....	586
Myślenie w kategoriach projektu .....	587
Zaangażowanie w tworzenie kodu .....	590
Nieustanne uczenie się .....	590
Bycie mentorem dla innych .....	593
Bycie propagatorem technologii i liderem .....	594
Podsumowanie .....	594
<b>Skorowidz .....</b>	<b>596</b>

# Rola architekta rozwiązań

Rozdział

1

Zadaniem tej książki jest ułatwienie Ci wykonania pierwszych kroków w świecie architektury rozwiązań. Niniejsza pozycja ma stać się wyczerpującym przewodnikiem, dzięki któremu dowiesz się wszystkiego o architekturze rozwiązań i staniesz się profesjonalnym architektem rozwiązań. W tym rozdziale zostanie ogólnie omówiona architektura rozwiązań oraz będą wyjaśnione podstawy opracowywania rozwiązań w organizacji. Niezawodny projekt architektury rozwiązań pomaga w przeprowadzeniu zakończonego sukcesem procesu tworzenia aplikacji oprogramowania w złożonej organizacji. Uwzględnia przy tym wszystkie aspekty, począwszy od infrastruktury IT, poprzez zapewnienie bezpieczeństwa aplikacji i niezawodności, po operacyjne aspekty w środowisku produkcyjnym.

Aby opracowanie aplikacji zakończyło się sukcesem, pierwszym krokiem powinno być zdefiniowanie architektury rozwiązania. To pozwoli przygotować podstawy i niezawodne elementy konstrukcyjne, które zostaną później użyte w implementacji. Architektura rozwiązania zajmuje się obsługą innych niż funkcjonalne wymagań o znaczeniu krytycznym, takich jak wysoka dostępność, łatwości późniejszej obsługi technicznej, wydajność działania i zapewnienie bezpieczeństwa, przy jednoczesnym uwzględnieniu wymagań biznesowych.

Architekt rozwiązań to osoba odpowiedzialna za opracowanie architektury rozwiązania we współpracy z innymi pracownikami i interesariuszami organizacji. Architekt rozwiązań analizuje funkcjonalne i definiuje inne niż funkcjonalne wymagania, aby uwzględnić wszystkie aspekty w rozwiązaniu oraz uniknąć wszelkich niespodzianek. Każde rozwiązanie ma wiele ograniczeń, takich jak koszt, budżet, ramy czasowe oraz ograniczenia prawne i regulacyjne. Dlatego też architekt rozwiązań musi je wszystkie rozważyć podczas wyboru technologii na etapie opracowywania projektu aplikacji, który ma być rozwiązaniem konkretnego problemu biznesowego.

Architekt rozwiązań opracowuje tzw. dowód koncepcji i prototyp, co pozwala mu na ocenę różnych platform technologicznych, a następnie wybór najlepszej strategii do zaimplementowania rozwiązania.

Opiekuje się on zespołem przez cały czas opracowywania rozwiązania, a po jego uruchomieniu dostarcza wskazówek dotyczących obsługi technicznej i skalowania ostatecznego produktu.

W rozdziale zostaną omówione następujące zagadnienia:

- Czym jest architektura rozwiązania?
- Ewolucja architektury rozwiązania.
- Dlaczego architektura rozwiązania jest ważna?
- Korzyści oferowane przez architekturę rozwiązania.
- Architektura rozwiązania w publicznej chmurze.

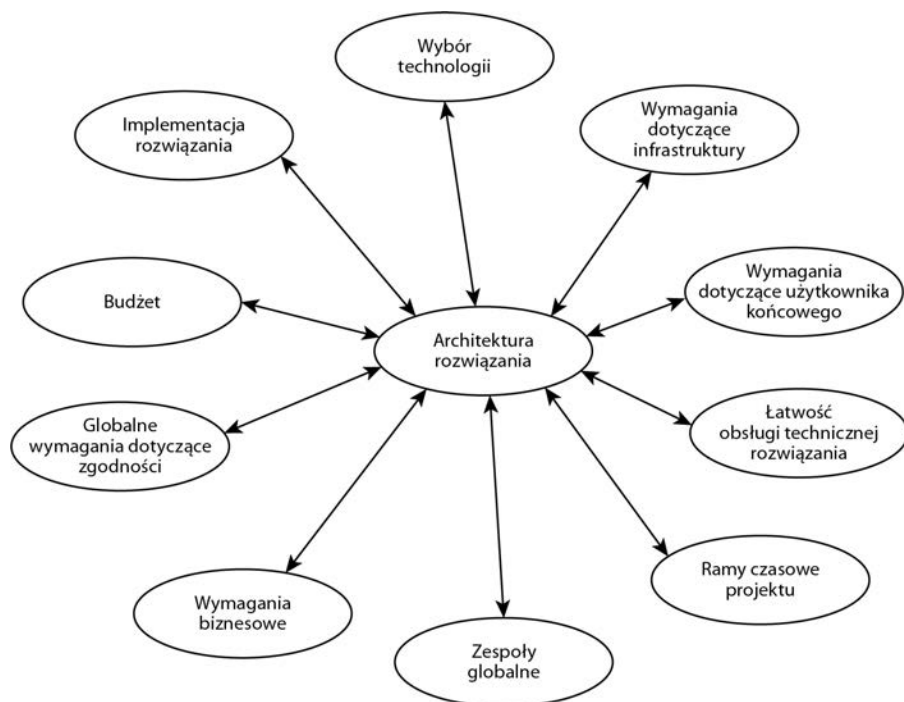
W rozdziale przedstawimy zalety architektury rozwiązania w każdym aspekcie aplikacji korporacyjnej. Dowiesz się, jak ocenić architekturę rozwiązania w publicznej chmurze oraz jak wypracować natywne podejście do projektowania architektury w chmurze.

## Czym jest architektura rozwiązania?

Jeżeli to pytanie zadasz różnym profesjonalistom, prawdopodobnie otrzymasz wiele odmiennych odpowiedzi definiujących architekturę rozwiązania. Tak naprawdę żadna z nich nie może być prawidłowa w kontekście struktury danej organizacji. Każda organizacja może postrzegać architekturę rozwiązania z zupełnie innej perspektywy, bazując na potrzebach biznesowych, hierarchii organizacyjnej oraz stopniu złożoności rozwiązania.

Ujmując rzecz najkrócej, architekturę rozwiązania można opisać jako definiowanie i przewidywanie wielu aspektów rozwiązania biznesowego z perspektywy zarówno strategicznej, jak i transakcyjnej. W tym kontekście „strategiczny” oznacza zdefiniowanie przez architekta rozwiązań długoterminowej wizji aplikacji oprogramowania, aby zapewnić jej aktualność niezależnie od przyszłych zmian oraz możliwość rozbudowy w celu obsługi coraz większego obciążenia powodowanego przez użytkowników i dodawania nowych funkcjonalności wymaganych przez użytkowników. Z kolei „transakcyjny” oznacza tutaj, że aplikacja powinna obsługiwać bieżące obciążenie powodowane przez klientów i bez żadnych problemów wykonywać codzienne zadania biznesowe.

Architektura rozwiązania nie ogranicza się jedynie do dostarczenia rozwiązania dotyczącego oprogramowania. Obejmuje wszystkie aspekty systemu, do których zaliczamy m.in. infrastrukturę systemu, sieć, zapewnienie bezpieczeństwa, wymagania dotyczące zgodności, działanie systemu, koszt i niezawodność. Jak możesz zobaczyć na rysunku 1.1, istnieje wiele aspektów, którymi architekt rozwiązań musi się zająć.



Rysunek 1.1. Różne aspekty architektury rozwiązania

Dobry architekt rozwiązań zajmuje się większością najczęściej pojawiających się aspektów wymagających uwzględnienia podczas tworzenia oprogramowania w organizacji.

- **Zespoły rozproszone globalnie.** W erze globalizacji niemal każdy produkt ma użytkowników na całym świecie, a grupy interesariuszy zajmują się potrzebami klientów. Zespoły odpowiedzialne za tworzenie oprogramowania bardzo często stosują model onshore-offshore, w którym zespół pracuje w różnych strefach czasowych, aby zwiększyć produktywność i zoptymalizować koszty projektu. Projekt rozwiązania musi uwzględniać strukturę globalnie rozproszonego zespołu. W takim przypadku opracowywanie rozwiązania i kwestie dotyczące działań operacyjnych nie powinny być zależne od pracowników, ale wykorzystywać narzędzia i możliwość współpracy niezależnie od miejsca i czasu pracy poszczególnych członków zespołu.
- **Globalne wymagania dotyczące zgodności.** Gdy rozwiązanie będzie wdrożone globalnie, trzeba pamiętać, że poszczególne kraje i terytoria mają własne normy prawne i regulacje, z którymi tworzone rozwiązanie musi być zgodne. Oto wybrane przykłady:

- W USA to **The Federal Risk and Authorization Management Program (FedRAMP)** i **Department of Defense Cloud Computing Security Requirements Guide (DoDSRG)**.
- W Europie to **Ogólne rozporządzenie o ochronie danych (RODO)** — **The General Data Protection Regulation (GDPR)**.
- W Australii to **The Information Security Registered Assessors Program (IRAP)**.
- W Japonii to **The Center for Financial Industry Information Systems (FISC)**.
- W Singapurze to standard **The Multi-Tier Cloud Security (MTCS)**.
- W Wielkiej Brytanii to **The G-Cloud**.
- W Niemczech to **The IT-Grundschutz**.
- W Chinach to **The Multi-Level Protection Scheme (MLPS) Level 3**.
- Wymagania dotyczące zgodności są różne w zależności od dziedziny. Na przykład **International Organization for Standardization (ISO) 9001** — wymagania przede wszystkim w dziedzinie ochrony zdrowia, nauk przyrodniczych, urzędów medycznych i przemysłu motoryzacyjnego, **Payment Card Industry Data Security Standard (PCI DSS)** — dziedzina finansów, **Health Insurance Portability and Accountability Act (HIPAA)** — dziedzina ochrony zdrowia. Wszystkie kwestie dotyczące zgodności muszą być uwzględnione już na etapie planowania architektury rozwiązania. W rozdziale 8. znajdziesz więcej informacji na temat zgodności.
- **Koszt i budżet.** Architektura rozwiązania pozwala w miarę wiarygodnie oszacować ogólny koszt projektu, co z kolei pomaga w zdefiniowaniu budżetu. To obejmuje **nakłady inwestycyjne** (ang. *capital expenditure*, CapEx) pomagające w pokryciu kosztów koniecznych do poniesienia już na samym początku projektu oraz **koszty operacyjne** (ang. *operational expenditure*, OpEx) pozwalające na pokrywanie bieżących kosztów projektu. Ogólnie rzecz biorąc, architektura rozwiązania pomaga w przygotowaniu budżetu związanego z kosztami zasobów ludzkich, zasobów infrastruktury i innych kosztów, np. dotyczących licencji.
- **Komponent implementacji rozwiązania.** Architektura rozwiązania zapewnia ogólny przegląd różnych komponentów implementacji produktu, co z kolei pomaga w przygotowaniu planu działania.
- **Wymagania biznesowe.** W architekturze rozwiązania są uwzględniane wszystkie wymagania biznesowe, zarówno funkcjonalne, jak i нефункционалне. Wymaganie funkcjonalne wiąże się z funkcjonalnością aplikacji, z której bezpośrednio będzie korzystał użytkownik. Z kolei wymaganie нефункционалне



nie jest bezpośrednio związane z funkcjonalnością używaną przez klienta, choć ma wpływ na aplikację w kategoriach czynników o znaczeniu krytycznym, takich jak wydajność działania, skalowalność i dostępność. Ten aspekt zapewnia, że wymagania biznesowe są ze sobą zgodne, mogą być skonwertowane na etap implementacji technicznej i zapewniają równowagę z oczekiwaniami interesariuszy.

- **Wymagania dotyczące infrastruktury IT.** Architektura rozwiązania określa rodzaj infrastruktury IT wymaganej do przeprowadzenia projektu. To obejmuje moc obliczeniową, pamięć masową, sieć itd., a także pomaga w znacznie efektywniejszym planowaniu zasobów IT.
- **Wybór technologii.** Na etapie projektowania rozwiązania jego architekt tworzy prototyp uwzględniający wymagania korporacyjne, a następnie proponuje odpowiednią technologię i narzędzia implementacji. Architektura rozwiązania ma pozwolić firmie na samodzielne przygotowanie rozwiązania zamiast korzystania z narzędzia opracowanego przez podmiot zewnętrzny, a jednocześnie umożliwia zdefiniowanie standardów oprogramowania w organizacji.
- **Wymagania dotyczące użytkownika końcowego.** W architekturze rozwiązania szczególną uwagę kładzie się na wymagania użytkownika końcowego, który będzie faktycznym konsumentem produktu. Ta architektura pomaga w odkryciu ukrytych wymagań, których nie udało się ustalić pomysłodawcy produktu bądź jego właścicielowi. Na etapie implementacji i uruchomienia architekt rozwiązań dostarcza standardowy dokument i standardową strukturę języka, aby upewnić się o spełnieniu wszystkich wymagań związanych z oczekiwaniami użytkownika.
- **Łatwość obsługi technicznej rozwiązania.** Architektura rozwiązania nie dotyczy jedynie projektu rozwiązania i jego implementacji. Obejmuje również działania prowadzone już po uruchomieniu projektu, takie jak skalowanie rozwiązania, odzyskiwanie po awarii i zapewnienie doskonałego działania.
- **Ramy czasowe projektu.** Jednym z celów architektury rozwiązania jest określenie szczegółów układu poszczególnych komponentów z uwzględnieniem ich poziomu złożoności. To pomaga później w definiowaniu kamieni milowych projektu i ram czasowych dzięki oszacowaniu niezbędnych zasobów i poziomu ryzyka związanego z projektem.

Będąca standardem przemysłowym i doskonale zdefiniowana architektura rozwiązania pozwala uwzględnić wszystkie wymagania biznesowe w rozwiązaniu technicznym. Ponadto gwarantuje dostarczenie zakładanego wyniku w celu zadowolenia interesariuszy, których oczekiwania można wyrazić w kategoriach jakości, dostępności i łatwości w późniejszej obsłudze technicznej rozwiązania oraz jego skalowalności.

Początkowy projekt architektury rozwiązania może powstać na bardzo wczesnym etapie cyklu przedsprzedażowego, np. **żądania propozycji** (ang. *request for proposal, RFP*) lub **żądania informacji** (ang. *request for information, RFI*). Następnie może powstać prototyp służący jako dowód koncepcji, a jego przeznaczeniem jest określenie wszelkiego ryzyka związanego z danym rozwiązaniem. Architekt rozwiązań ustala, czy będzie ono zbudowane samodzielnie, czy zostanie wykorzystane już istniejące. To pomaga w wyborze właściwej technologii przy jednoczesnym uwzględnieniu kwestii bezpieczeństwa organizacji o znaczeniu krytycznym i wymagań dotyczących zgodności.

Mamy dwie podstawowe sytuacje, w których jest tworzona architektura rozwiązania.

- Pierwsza to usprawnienie technologii istniejącej aplikacji. To może obejmować odświeżenie sprzętu komputerowego lub ponowne opracowanie oprogramowania.
- Druga to tworzenie nowego rozwiązania zupełnie od początku. Zapewnia się większą elastyczność w zakresie wyboru technologii, która ma być użyta do spełnienia wymagań biznesowych.

Jednak podczas ponownego opracowywania istniejącego rozwiązania konieczne jest uwzględnienie minimalnego wpływu na bieżące środowisko. Architekt rozwiązań może zdecydować się na całkowite przebudowanie rozwiązania, o ile modyfikacja istniejącego jest po prostu niewarta wysiłku. Efektem podejścia bazującego na pełnej przebudowie może być powstanie znacznie lepszego rozwiązania.

Ujmując rzecz najprościej: architektura rozwiązania to sprawdzenie wszystkich aspektów systemu w celu przygotowania wizji technicznej, która z kolei ujawni kroki pozwalające na implementację wymagań biznesowych. Architektura rozwiązania może definiować implementację projektu bądź grupę projektów w skomplikowanym środowisku, co odbywa się poprzez zebranie razem różnych fragmentów powiązanych ze sobą danych, infrastruktury, sieci i aplikacji oprogramowania. Dobra architektura rozwiązania nie tylko spełnia wymagania funkcjonalne i нефunkcjonalne, ale również wymagania dotyczące skalowalności systemu i łatwości jego obsługi technicznej w dłuższej perspektywie czasu.

W ten sposób pokrótce wyjaśniliśmy rolę architektury rozwiązania i jej różnych aspektów. W następnym podrozdziale poznasz ewolucję architektury rozwiązania.

## Ewolucja architektury rozwiązania

Architektura rozwiązania ewoluowała razem z modernizacją technologiczną. Obecny projekt architektury rozwiązania znacznie się różni od stosowanego kilka dekad temu,

co ma związek z coraz powszechniejszym wykorzystaniem internetu, dostępnością sieci o wysokiej przepustowości, spadkiem kosztu pamięci masowej oraz dostępnością sprzętu komputerowego.

W erze przed upowszechnieniem się internetu większość projektów rozwiązań koncentrowała się na dostarczeniu rozbudowanego klienta biurowego, który był w stanie działać w warunkach połączenia o małej przepustowości, a także działać w trybie offline, gdy system nie mógł nawiązać połączenia z internetem.

W ciągu dwóch dekad ta technologia ewoluowała. W projektowaniu rozwiązań rozproszonych zaczęła być stosowana **architektura zorientowana na usługi** (ang. *service-oriented architecture*, **SOA**). Z kolei w aplikacjach zaczęto przechodzić od architektury monolitycznej do **nowoczesnej architektury n-warstwowej**, w której serwery frontendu, aplikacji i bazy danych wykorzystują oddzielne warstwy obliczeniowe i pamięci masowej. Rozwój architektury SOA stał się możliwy głównie dzięki bazującemu na XML protokołowi komunikatów, **SOAP** (ang. *simple object access protocol*). Najważniejszą cechą w tym podejściu jest możliwość stosowania modelu klient-serwer, pozwalającego na tworzenie usług.

W erze cyfryzacji rozwiązania projektowe bazujące na mikrousługach zyskują coraz większą popularność. Najczęściej opierają się one na komunikatach **JSON** (ang. *javascript object notation*) i usłudze **REST** (ang. *representational state transfer*). Dostępne są API sieciowe, niewymagające bazujących na XML protokołów sieciowych (SOAP), wspomagające ich interfejsy. Wspomniane API opierają się na protokołach sieciowych bazujących na HTTP, np. POST, GET, UPDATE, DELETE itd. W rozdziale 6. znacznie dokładniej poznasz różne wzorce architektoniczne.

Architektura mikrousług odpowiada na potrzebę zmiany wymagań w środowisku zwinnym, w którym wszelkie zmiany rozwiązania muszą być szybko dostosowywane i wdrażane. Organizacje muszą działać elastycznie, aby wyprzedzać konkurencję. Dlatego też architektura rozwiązania powinna być elastyczna w porównaniu z modelem kaskadowym, który charakteryzował się długimi cyklami między wydaniem projektu.

Bazująca na sieci architektura mikrousług jest napędzana przez niemal niewyczerpalne źródło możliwości oferowanych przez dostawców chmury, a jej skalowanie może odbywać się w ciągu zaledwie minut bądź nawet sekund. Zatem coraz łatwiejsza stała się innowacyjność, eksperymentowanie i modyfikowanie, ponieważ architekti rozwiązań i programiści mogą podjąć ryzyko bez obaw, że niepowodzenie będzie miało wpływ na działalność biznesową organizacji.

# Dlaczego architektura rozwiązania jest ważna?

Architektura rozwiązania to komponent o fundamentalnym znaczeniu dla całego korporacyjnego rozwiązania w zakresie oprogramowania, odpowiadającego na konkretne problemy i wymagania. Gdy projekt staje się coraz większy, tworzący go zespół zaczyna być rozproszony globalnie. Niezbędne jest wówczas posiadanie architektury rozwiązania, aby zapewnić w dłuższej perspektywie czasu stabilność i solidne fundamenty projektu.

Architektura rozwiązania odpowiada na różne potrzeby i nie ingeruje w kontekst biznesowy. Określa i dokumentuje platformy technologiczne, komponenty aplikacji, wymagania dotyczące danych i zasobów, a także wiele ważnych wymagań нефunkcjonalnych, takich jak skalowalność, niezawodność, wydajność działania, przepustowość, dostępność, zapewnienie bezpieczeństwa i łatwość obsługi technicznej rozwiązania.

Architektura rozwiązania jest istotna w każdej dziedzinie, w której problemy biznesowe są rozwiązywane za pomocą aplikacji oprogramowania. W przypadku braku architektury rozwiązania istnieje niebezpieczeństwo, że prace programistyczne mogą zakończyć się niepowodzeniem: projekt będzie miał opóźnienie, nastąpi przekroczenie jego budżetu i nie zaoferuje odpowiedniej funkcjonalności. Ryzyko wystąpienia takiego niepożądanego scenariusza można znacznie ograniczyć dzięki utworzeniu architektury rozwiązania oraz wykorzystaniu wiedzy i doświadczenia — to wszystko może zaoferować architekt rozwiązań. Takie podejście pozwala zostawić po tej samej stronie wszystkich interesariuszy, począwszy od spraw nietechnicznych, po związane z pracami technicznymi. Dzięki temu unika się nieporozumień, zachowuje harmonogram pracy nad projektem oraz pomaga w osiągnięciu maksymalnego zwrotu z inwestycji (ang. *return on investment*, ROI).

Bardzo często architekt rozwiązań musi współpracować z klientem w celu pełnego zrozumienia wymagań. Od osoby w roli architekta rozwiązań wymaga się wielu umiejętności, począwszy od wiedzy technicznej, poprzez doświadczenie w biznesie, aż po zarządzanie projektem. Więcej informacji na temat roli architekta rozwiązań znajdziesz w rozdziale 2.

W dobrej architekturze rozwiązania specyfikacja znajduje się na swoim miejscu, razem z doskonale zdefiniowanym rozwiązaniem, co pomaga w ukończeniu produktu końcowego i jego dostarczeniu, a także w zapewnieniu płynności jego działania po uruchomieniu.

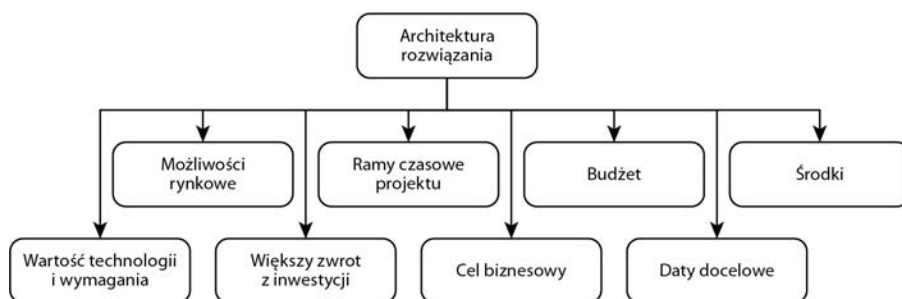
Problem może mieć wiele rozwiązań, z których każde może charakteryzować się pewnymi ograniczeniami. Należy przeanalizować wszystkie rozwiązania i wybrać najlepsze

z nich poprzez utworzenie dowodu koncepcji, w którym pod uwagę zostały wzięte wszystkie ograniczenia biznesowe i techniczne.

W następnym podrozdziale dokładnie przedstawię różne korzyści płynące z architektury rozwiązania.

## Korzyści płynące z architektury rozwiązania

Po omówieniu wagi architektury rozwiązania, przechodzimy do dokładniejszego przedstawienia korzyści oferowanych przez architekturę rozwiązania w różnych aspektach organizacji. Na rysunku 1.2 pokazaliśmy zestawienie potencjalnych korzyści, jakie może osiągnąć organizacja dzięki wykorzystaniu usług architekta rozwiązań.



Rysunek 1.2. Korzystne atrybuty architektury rozwiązania

Diagram zamieszczony na rysunku 1.2 wskazuje kilka atrybutów, którymi charakteryzuje się dobra architektura rozwiązania.

- **Wartość technologii i wymagania.** Architektura rozwiązania ma wpływ na stopę zwrotu z inwestycji, na podstawie wybranej technologii określa, które rozwiązanie może być użyte, a także wskazuje trendy na rynku. Architekt rozwiązań ocenia, którą technologię powinna wykorzystać organizacja lub projekt w celu zapewnienia długoterminowej stabilności, łatwości obsługi technicznej rozwiązania oraz komfortu zespołu.
- **Cel biznesowy.** W projekcie architektury rozwiązania jest konieczne uwzględnienie potrzeb interesariuszy oraz dostosowanie się do nich. Architektura rozwiązania konwertuje cele biznesowe na wizję techniczną przez analizowanie trendów rynkowych i implementację najlepszych praktyk. Architektura rozwiązania musi być na tyle elastyczna, aby sprostać nowym, ambitnym, trudnym i szybko się zmieniającym wymaganiom biznesowym.

- **Daty docelowe.** Architekt rozwiązań nieustannie współpracuje z wszystkimi interesariuszami, zespołem biznesowym, klientami oraz zespołem programistycznym. Jego zadaniem jest zdefiniowanie standardu procesu i wyznaczenie reguł stosowanych podczas tworzenia rozwiązania. Architekt rozwiązań upewnia się również, że ogólne rozwiązanie pozostaje w zgodzie z celami biznesowymi i ramami czasowymi jego uruchomienia, aby w ten sposób do minimum ograniczyć niebezpieczeństwo poślizgu czasowego.
- **Większy zwrot z inwestycji.** Architekt rozwiązań ustala stopę zwrotu z inwestycji i pomaga w określeniu miary sukcesu projektu. Wymusza na biznesie rozważenie sposobów na zmniejszenie kosztów i eliminację marnotrawstwa w trakcie samego procesu, co jest możliwe dzięki zastosowaniu automatyzacji. W ten sposób można zwiększyć ogólną stopę zwrotu z inwestycji.
- **Możliwości rynkowe.** Architektura rozwiązania obejmuje proces analizowania i nieustannego oceniania najnowszych trendów na rynku. Pomaga również w trakcie wspierania i promowania nowych produktów.
- **Budżet i środki.** Aby przygotować lepszy budżet, zawsze zaleca się przeprowadzenie dokładnego oszacowania. Świetnie zdefiniowana architektura rozwiązania pomaga w ustaleniu ilości zasobów niezbędnych do ukończenia projektu. To z kolei pomaga w sformułowaniu lepszych prognoz budżetowych i zaplanowaniu zasobów.
- **Ramy czasowe projektu.** Zdefiniowanie właściwych ram czasowych projektu ma znaczenie krytyczne dla implementacji rozwiązania. Architekt rozwiązań ustala zasoby i ilość pracy koniecznej do włożenia na etapie planowania, co powinno pomóc w przygotowaniu harmonogramu.

W ten sposób przedstawiliśmy ogólne omówienie architektury rozwiązania i oferowanych przez nią korzyści. Przechodzimy teraz do znacznie dokładniejszego omówienia codziennych aspektów architektury rozwiązania.

## Odpowiedź na potrzeby biznesowe i jakość dostawy

W cyklu życiowym opracowania produktu najbardziej wymagającym etapem jest ustalenie natury wymagań, zwłaszcza wtedy, gdy wiele elementów jest uznawanych za te o wysokim priorytecie i szybko ulega zmianom. Zadanie staje się jeszcze trudniejsze, gdy poszczególni interesariusze mają zupełnie odmiennie poglądy na te same wymagania. Na przykład użytkownik biznesowy analizuje projekt strony z perspektywy użytkownika, podczas gdy programista patrzy na nią z perspektywy możliwości implementacji i czasu wczytywania. To może prowadzić do konfliktów i niezrozumienia wymagań między

funkcjonalnymi i technicznymi członkami zespołu. W takich sytuacjach architektura rozwiązania pomaga w zbliżeniu stanowisk oraz definiuje standard zrozumiały i akceptowany przez wszystkich członków zespołu.

Wymagania funkcjonalne to odpowiednie działanie produktu mające spełnić oczekiwania użytkownika i pomóc w rozwiązaniu podstawowego problemu biznesowego. Gdy użytkownik pracuje z aplikacją oprogramowania, ma bezpośredni kontakt z wymaganiami funkcjonalnymi. Na przykład w aplikacji typu e-commerce do wymagań funkcjonalnych zaliczamy możliwość wyświetlania historii zamówień, szukania produktu w sklepie, dodawania produktu do koszyka na zakupy, dokonywania płatności za pomocą preferowanej metody itd. Wprawdzie ustalenie wymagań funkcjonalnych spoczywa przede wszystkim na właścicielu produktu, ale architekt rozwiązań zapewnia, że projekt i implementacja zostały przygotowane w sposób zapewniający możliwość skalowania, gdy zwiększą się oczekiwania użytkowników i pojawi się potrzeba późniejszej rozbudowy rozwiązania.

Architektura rozwiązania definiuje standard tworzenia dokumentacji, w której aspekty techniczne są wyjaśniane nietechnicznym interesariuszom, a także reguły jej regularnego uaktualniania. Skoro projekt architektury rozwiązania obejmuje całą organizację i jej różne zespoły, może pomóc w odkryciu ukrytych wymagań. Architekt rozwiązań upewnia się, że zespół programistyczny zna wymagania i stosuje się do cyklu postępu.

Dobra architektura rozwiązania definiuje nie tylko projekt rozwiązania, ale również kryteria sukcesu w postaci danych jakościowych i ilościowych, aby w ten sposób zapewnić odpowiednią jakość dostawy. Dane jakościowe mogą zostać zebrane na podstawie informacji dostarczanych przez użytkowników, np. poprzez analizę opinii krytycznych. Z kolei dane ilościowe mogą obejmować informacje zebrane po stronie technicznej (np. dotyczące opóźnienia, wydajności działania i czasu wczytywania) i po stronie biznesowej (np. wielkość sprzedaży). Nieustanne pobieranie takich danych i ich analizowanie ma znaczenie kluczowe dla zapewnienia wysokiej jakości dostawy, więc powinno odbywać się na wszystkich etapach projektowania rozwiązania i jego implementacji.

## Wybór najlepszej platformy technologicznej

rynku największym wyzwaniem jest zagwarantowanie używania najlepszych technologii. Obecnie, gdy mamy wiele zasobów rozsianych na całym świecie, konkretną technologię należy wybierać bardzo ostrożnie. Rozwiązaniem jest takie zaprojektowanie architektury, aby pozwalała na efektywne radzenie sobie z tym problemem.

Wybór stosu technologicznego odgrywa ważną rolę na etapie tworzenia przez zespół efektywnego rozwiązania. W architekturze rozwiązania należy wykorzystać odmienne strategie w celu zaadaptowania różnych platform, technologii i narzędzi. Architekt rozwiązań powinien ostrożnie zweryfikować wszystkie potrzeby, a następnie ocenić

je i przeanalizować wynik, używając do tego wielu parametrów, aby znaleźć najlepsze rozwiązanie dla procesu utworzenia produktu. To odbywa się przez utworzenie działającego modelu produktu w postaci prototypu.

Dobra architektura rozwiązania uwzględnia dostępność wielu różnych narzędzi i technologii poprzez przeanalizowanie wszystkich możliwych strategii architektonicznych na podstawie przypadków użycia, technik, narzędzi i wielokrotnego używania kodu — to wszystko jest możliwe dzięki wieloletniemu doświadczeniu. Wprawdzie najlepsza platforma upraszcza proces implementacji, ale wybór odpowiedniej technologii ma znaczenie krytyczne. To jest możliwe dzięki zbudowaniu prototypu zgodnie z przeanalizowanymi wcześniej wymaganiami biznesowymi oraz dzięki uwzględnieniu zwinności, szybkości i bezpieczeństwa aplikacji.

## Uwzględnianie ograniczeń rozwiązania i problemów

Każde rozwiązanie może być ograniczone przez różne czynniki oraz może napotykać problemy wynikające z poziomu złożoności lub nieprzewidzianego ryzyka. Architektura rozwiązania musi zapewniać równowagę między wieloma ograniczeniami, takimi jak dostępne zasoby, technologie, koszt, jakość, czas, w jakim produkt ma się pojawić na rynku, a także często zmieniające się wymagania.

Każdy projekt ma swój konkretny cel, wymagania, budżet i ramy czasowe. Architektura rozwiązania ma za zadanie ocenić wszystkie ścieżki o znaczeniu krytycznym oraz współdzielić najlepsze praktyki w celu umożliwienia osiągnięcia celu projektu w podanych ramach czasowych i budżecie. To jest podejście systematyczne, w którym wszystkie zadania są niezależne od wcześniejszych. Aby można było osiągnąć sukces w projekcie, wszystkie zadania muszą być wykonane w sekwencji. Opóźnienie podczas wykonywania jednego z nich może mieć wpływ na ramy czasowe projektu i doprowadzić do przegapienia przez organizację najlepszego momentu na wprowadzenie produktu na rynek.

W przypadku pojawienia się problemu związanego z procesem tworzenia projektu rośnie niebezpieczeństwo jego opóźnienia. Czasami napotkane problemy wynikają z ograniczeń wybranej technologii bądź nawet rozwiązania środowiskowego. Jeżeli masz dobrze przemyślaną architekturę rozwiązania, wówczas najczęściej pojawiające się problemy są związane z wymaganiami niefunkcjonalnymi: kwestie dotyczące zasobów i budżetowania mogą złagodzić problemy napotkane w cyklu życiowym tworzenia produktu.

Architekt rozwiązań pomaga w kierowaniu projektem przez dokładne przeanalizowanie jego wszystkich komponentów. Zastanawia się nad nieszablonowymi rozwiązaniami, aby



uchronić projekt przed nieprzewidzianymi problemami, takimi jak podczas odzyskiwania po wystąpieniu awarii, oraz przygotowuje plan awaryjny na wypadek niepowodzenia wykonania planu głównego. Architekt rozwiązań szuka najlepszego możliwego sposobu wykonania projektu, decydując się na stosowanie najlepszych praktyk i równoważąc ograniczenia.

## Pomoc w zarządzaniu zasobami i kosztami

Zawsze istnieje ryzyko związane z czynnikiem nieprzewidywalności na etapie implementacji rozwiązania. To może stać się niezwykle uciążliwe, gdy programista musi poświęcać czas np. na usuwanie znalezionych błędów. Dobra architektura rozwiązania pozwala na kontrolowanie kosztów i budżetu, a także zmniejsza niepewność, zapewniając programistom niezbędne wsparcie w kategoriach ustalania priorytetów, różnych usług komunikacyjnych oraz szczegółów związanych z poszczególnymi komponentami.

Architektura rozwiązania tworzy również dokumentację, która będzie używana do zapewnienia aktualności systemu, zawiera diagram wdrożenia, informacje o poprawkach oprogramowania i numerze wydania. Ponadto w dokumentacji znajduje się omówienie najczęściej pojawiających się problemów i procesów pozwalających na kontynuowanie działalności biznesowej. Dokumentacja powinna również uwzględniać pośredni wpływ kosztu budowy rozwiązania poprzez analizę możliwości jego rozbudowy, skalowalności oraz innych czynników zewnętrznych, które mają wpływ na środowisko programistyczne.

## Zarządzanie dostarczaniem rozwiązania i cyklu życiowego projektu

Na początkowym etapie tworzenia architektury rozwiązania mamy do czynienia z ogromną ilością planowania. Architektura rozwiązania rozpoczyna się od nakreślenia strategicznych celów, a następnie w miarę poruszania się w kierunku implementacji dostarcza coraz większej ilości technicznych szczegółów implementacji.

Architektura rozwiązania gwarantuje dostarczenie w pełni opracowanego projektu, na którego cykl życiowy ma ogromny wpływ. Definiuje standard procesów na różnych etapach cyklu życiowego projektu oraz gwarantuje stosowanie tego standardu w organizacji. Dzięki temu inne zależności będą mogły być rozwiązane podczas implementacji projektu.

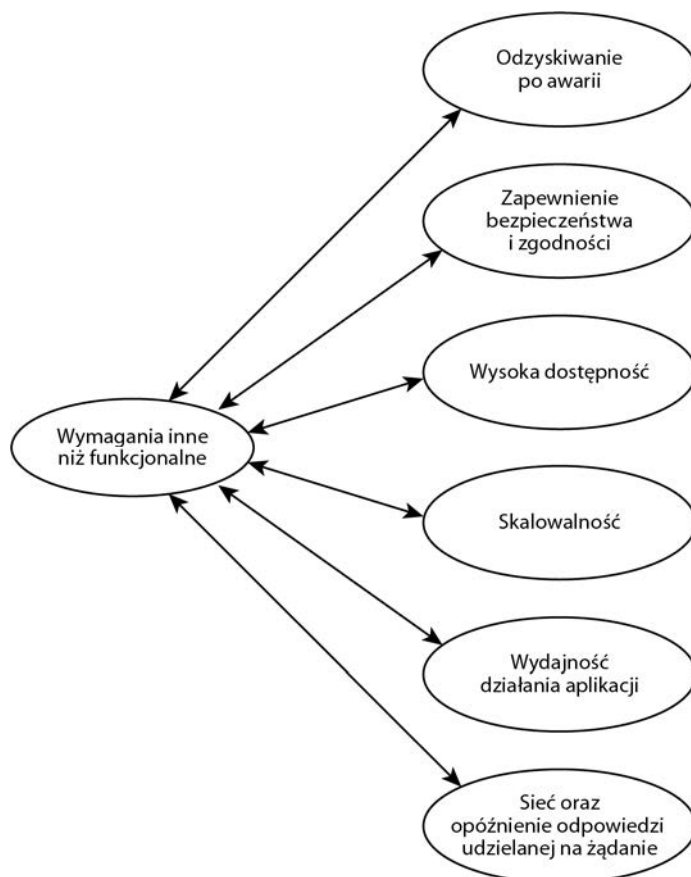
W architekturze rozwiązania stosowane jest holistyczne podejście do projektu. To pozwala zsynchronizować inne grupy zależne, takie jak dotyczące zapewnienia bezpieczeństwa, zgodności, infrastruktury, zarządzania projektem i pomocy technicznej, co ma na celu utrzymanie zaangażowania tych grup na różnych etapach cyklu życiowego projektu.

## Uwzględnienie wymagań innych niż funkcjonalne

Architekt rozwiązań często musi zajmować się także **wymaganiami innymi niż funkcjonalne** w aplikacji (ang. *non-functional requirements*, **NFR**). Aby projekt odniósł sukces, konieczne jest uwzględnienie tych wymagań, ponieważ mają one ogromny wpływ ogólnie na projekt i na rozwiązanie. Te wymagania mogą przyczynić się do zwiększenia lub zmniejszenia bazy użytkowników oraz wpływać na aspekty rozwiązania o znaczeniu krytycznym, takie jak zapewnienie bezpieczeństwa, kwestie dostępności, obawy związane z opóźnieniem, późniejsza konserwacja rozwiązania, rejestrowanie danych, ochrona informacji wrażliwych, kwestie związane z wydajnością działania, niezawodność, łatwość obsługi technicznej rozwiązania, skalowalność i użyteczność. Jeżeli wymienionymi kwestiami nie zajmiesz się na czas, mogą mieć wpływ na dostarczenie projektu. Na rysunku 1.3 pokazaliśmy najczęściej spotykane wymagania inne niż funkcjonalne.

Jak możesz zobaczyć na rysunku 1.3, wymagania inne niż funkcjonalne wiążą się z wymienionymi tutaj atrybutami architektury rozwiązania. Pamiętaj, że w zależności od typu projektu może być więcej tego rodzaju wymagań.

- **Odzyskiwanie po awarii.** Ten atrybut gwarantuje, że rozwiązanie działa i będzie funkcjonowało w przypadku nieprzewidzianych zdarzeń.
- **Zapewnienie bezpieczeństwa i zgodności.** Konieczne jest zastosowanie środków bezpieczeństwa, aby w ten sposób chronić rozwiązanie przed zagrożeniami zewnętrznymi, takimi jak wirusy, złośliwe oprogramowanie itd. Należy się również upewnić o zgodności rozwiązania z lokalnymi przepisami i przyjętymi normami.
- **Wysoka dostępność.** Dzięki temu atrybutowi rozwiązanie zawsze działa.
- **Skalowalność.** Ten atrybut ma zapewnić, że rozwiązanie poradzi sobie z obsługą dodatkowego obciążenia, gdy zajdzie taka potrzeba.
- **Wydajność działania aplikacji.** Ma na celu zagwarantowanie, że szybkość działania aplikacji spełnia oczekiwania użytkowników i w trakcie jej działania nie występuje zbyt wiele opóźnień.
- **Sieć oraz opóźnienie odpowiedzi udzielanej na żądanie.** Każda operacja przeprowadzana w aplikacji powinna być zakończona w odpowiednim czasie i nie powinno dochodzić do sytuacji przekroczenia czasu oczekiwania.



**Rysunek 1.3. Inne niż funkcjonalne atrybuty architektury rozwiązania**

Więcej informacji na temat tych atrybutów znajdziesz w rozdziale 3. Architektura rozwiązania definiuje początkowy framework przeznaczony na potrzeby opracowania produktu i elementów konstrukcyjnych rozwiązania. Przygotowując architekturę rozwiązania, pod uwagę zawsze trzeba brać jakość i satysfakcję klienta. Budowa architektury rozwiązania to proces zajmujący nieco czasu, w trakcie którego zaczyna się od utworzenia dowodu koncepcji, a następnie eksperymentuje się i testuje rozwiązanie aż do chwili osiągnięcia żądanej jakości.

# Architektura rozwiązania w publicznej chmurze

Obecnie architektura rozwiązania w publicznej chmurze przyciąga coraz większą uwagę i powoli staje się „nową normą”, ponieważ coraz więcej firm zaczyna korzystać z usług chmury, przenosząc do niej coraz więcej swoich zadań. Publiczna chmura stała się czynnikiem o znaczeniu krytycznym, który przyczynił się do gwałtownego wzrostu liczby start-upów, ponieważ użycie chmury nie wiąże się z ogromnymi kosztami koniecznymi do poniesienia na początku działalności biznesowej. Jednocześnie chmura oferuje organizacjom elastyczność pozwalającą na eksperymentowanie i innowacyjność.

Doskonałą cechą architektury chmury jest to, że masz całkowity podgląd wszystkich komponentów chmury, m.in. platform frontendu, platform tworzenia aplikacji, serwerów, pamięci masowej, baz danych, automatyzacji, dostarczania i sieci niezbędnych do zarządzania całym rozwiązaniem.

Zanim jednak przejdziesz do architektury rozwiązania w chmurze, najpierw musisz dowiedzieć się nieco więcej na temat samej chmury i tego, jak stała się ważną i napędzającą rozwój platformą technologiczną dla firm.

## Czym jest publiczna chmura?

Publiczna chmura bazuje na standardowym modelu obliczeniowym, w którym dostawca usługi poprzez internet zapewnia klientom pewne zasoby, takie jak maszyny wirtualne, aplikacje i pamięć masowa. W tego rodzaju usługach jest stosowany tzw. model *pay-as-you-go*, w którym płaci się za faktycznie wykorzystane zasoby.

W modelu przetwarzania w chmurze dostawca usług zapewnia zasoby na żądanie — np. serwer, bazę danych, sieć, pamięć masową — organizacjom, które następnie mogą z nich korzystać za pomocą bazujących na przeglądarce WWW bezpiecznych interfejsów bądź poprzez aplikacje w internecie. W większości przypadków klient płaci jedynie za faktycznie wykorzystane zasoby, co pozwala zmniejszyć koszty działalności przez optymalizację zasobów IT oraz skrócić czas bezczynności.

W dużym uproszczeniu publiczną chmurę można porównać do modelu sieci energetycznej — używasz przełącznika do np. zapalenia światła i płacisz jedynie za faktycznie zużytą ilość energii. Po wyłączeniu światła nie płacisz za energię elektryczną. Dzięki takiemu modelowi nie musisz przejmować się złożonością procesu generowania prądu za pomocą turbin, zasobami niezbędnymi do utrzymania elektrowni, ogromną infrastrukturą i konfiguracją całości — w uproszczony sposób korzystasz z całej usługi.

Poza tym, że oferują korzyści dotyczące kosztów, najważniejsi dostawcy usług publicznej chmury, np. **Amazon Web Services (AWS)**, **Google Cloud Platform (GCP)** i **Microsoft Azure**, przyczyniają się również do większej innowacyjności, ponieważ rozszerzają swoje technologie poprzez chmurę. Ci dostawcy w doskonałym stopniu opanowali tworzenie skalowalnej i przyszłościowej architektury, wykorzystując do tego skomplikowane modele uczenia maszynowego i analizę. Dzięki publicznej chmurze otrzymujesz dostęp do najnowszych technologii; masz też możliwość wykorzystać ją w opracowywanej samodzielnie architekturze.

## Chmura publiczna, prywatna i hybrydowa

W tym punkcie przedstawimy ogólne omówienie poszczególnych typów modeli wdrożeń przetwarzania w chmurze. Dokładne informacje na ten temat znajdziesz w rozdziale 3.

**Chmura prywatna**, inaczej **chmura lokalna**, jest przeznaczona dla pojedynczej organizacji, która jest jej właścicielem i z niej korzysta. Taka chmura działa jako replika bądź rozszerzenie istniejącego centrum danych firmy. Często zdarza się, że **chmura publiczna** współdzieli infrastrukturę IT, a tym samym serwery wirtualne wielu klientów są uruchomione w tym samym serwerze fizycznym. Jednak często dostawcy chmury publicznej oferują klientom tzw. serwery dedykowane, jeśli klient potrzebuje takiego rozwiązania ze względów licencyjnych bądź w celu zapewnienia zgodności. Chmura publiczna, taka jak AWS, Microsoft Azure lub GCP, wykorzystuje ogromną infrastrukturę IT, do której dostęp można uzyskać za pomocą internetu, z użyciem wspomnianego wcześniej modelu *pay-as-you-go*.

Trzecim modelem jest **chmura hybrydowa**, używana w ogromnych przedsiębiorstwach, które przenoszą swoją działalność z sieci lokalnej do chmury. Często zdarza się, że wciąż mają wiele starszych aplikacji, których nie można bezpośrednio przenieść do chmury, bądź też posiadają licencjonowane aplikacje wymagające używania ich w środowisku lokalnym. W jeszcze innych przypadkach te organizacje muszą zapewnić bezpieczeństwo danych, przechowując je lokalnie, ze względu na kwestie dotyczące zgodności. W takiej sytuacji model hybrydowy pomaga, gdy organizacja musi zachować częściowe środowisko lokalne, zaś pozostałe aplikacje przenieść do chmury publicznej. Czasami organizacja przenosi do chmury publicznej środowiska związane z testowaniem i opracowywaniem, natomiast lokalnie pozostawia środowisko produkcyjne. Model hybrydowy będzie się różnił w zależności od przyjętej przez daną organizację strategii związanej z chmurą.

Ponieważ na rynku istnieje wielu dostawców chmury publicznej, można zauważyć pojawiający się trend wykorzystania wielu chmur. Korporacje decydują się na podział zadań między różne chmury publiczne, aby w maksymalnym stopniu wykorzystać

poszczególne technologie chmury. Ewentualnie oferują zespołom różne możliwości wyboru, w zależności od umiejętności członków zespołu.

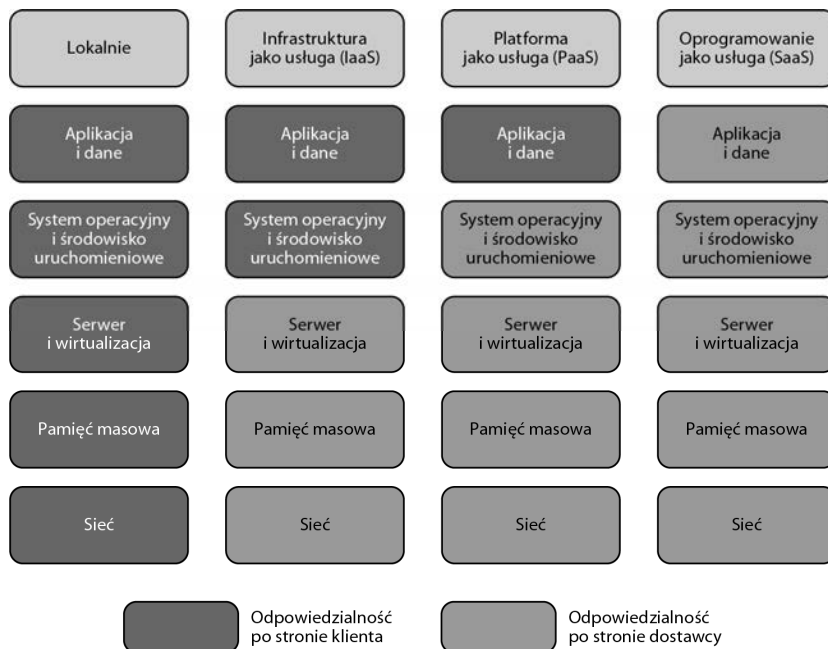
## Architektura chmury publicznej

Typowa definicja chmury publicznej przedstawia się następująco: w pełni zwirtualizowane środowisko dostępne zarówno poprzez internet, jak i sieć prywatną. Jednak ostatnio dostawcy chmury publicznej zaczęli oferować lokalną infrastrukturę fizyczną, aby zapewniać jeszcze lepsze rozwiązania hybrydowe. Chmura publiczna oferuje model nazywany *multitenancy*, w którym infrastruktura IT, taka jak pamięć masowa i moc obliczeniowa, jest współdzielona między wielu klientów. Mimo tego pozostają oni odizolowani na poziomie oprogramowania i sieci logicznej oraz nie przeszkadzają sobie w pracy. W chmurze publicznej dzięki zapewnieniu izolacji na poziomie sieci organizacje mogą mieć wirtualną chmurę prywatną, która jest odpowiednikiem logicznego centrum danych. Analizując wymagania organizacji dotyczące regulacji, dostawca chmury publicznej oferuje dedykowane egzemplarze fizyczne — są one dostępne przez sieć, choć to jest rzadziej spotykane rozwiązanie.

Pamięć masowa chmury publicznej zapewnia wysoką trwałość i dostępność dzięki wykorzystaniu modelu redundancji. Do jego utworzenia jest używanych wiele centrów danych i niezawodna replikacja danych. W ten sposób powstaje architektura odporna na awarie i łatwa do skalowania. Mamy trzy podstawowe modele przetwarzania w chmurze — zostały one pokazane na rysunku 1.4.

Na rysunku 1.4 możesz zobaczyć zestawienie kwestii, za które klient odpowiada w środowiskach lokalnym i modelu usługi chmury zarządzanej. W środowisku lokalnym klient musi zająć się wszystkim, podczas gdy w przypadku modelu usługi chmury zarządzanej odpowiedzialność za wszystkie zadania spada na dostawcę chmury, a klientowi pozostaje zajęcie się jedynie potrzebami biznesowymi. W zamieszczonych tutaj punktach nieco dokładniej omówiliśmy usługi oferowane w ramach różnych modeli chmury zarządzanej.

- **Infrastruktura jako usługa** (ang. *infrastructure as a service, IaaS*). W tym modelu dostawca chmury oferuje zasoby infrastruktury — takie jak serwer, komponenty sieciowe i pamięć masowa — w postaci usług zarządzanych. To pomaga klientom w używaniu zasobów IT bez konieczności przejmowania się kwestiami związanymi z obsługą centrum danych, takimi jak nagrzewanie się sprzętu, zapewnienie mu odpowiedniego chłodzenia, rozmieszczenie sprzętu, zapewnienie fizycznej ochrony centrum danych itd.



Rysunek 1.4. Typy modeli przetwarzania w chmurze

- **Platforma jako usługa** (ang. *platform as a service, PaaS*). W tym modelu została dodana warstwa usługi, na której dostawca chmury zajmuje się zasobami wymaganymi przez platformę programistyczną, takimi jak system operacyjny, obsługa oprogramowania i stosowanie poprawek, oraz zapewnia zasoby infrastruktury. Dzięki modelowi PaaS zespół może skoncentrować się na tworzeniu logiki biznesowej i obsłudze danych, ponieważ jest zwolniony z zajmowania się wszelkimi zadaniami związanymi z obsługą techniczną platformy.
- **Oprogramowanie jako usługa** (ang. *software as a service, SaaS*). Model SaaS dodaje jedną lub więcej warstw abstrakcji na bazie modeli PaaS i IaaS. Dostawca chmury bądź oprogramowania zapewnia oprogramowanie gotowe do użycia, a klient płaci za usługę. Na przykład możesz korzystać z poczty elektronicznej, takiej jak Gmail, Yahoo! Mail, AOL itd., w której w ramach usługi otrzymujesz pewną ilość miejsca na swoje wiadomości e-mail i nie musisz się przejmować aplikacjami lub infrastrukturą zapewniającą poprawne działanie poczty elektronicznej.

Pojawia się jeszcze czwarty model — **funkcja jako usługa** (ang. *function as a service, FaaS*) — który zyskuje popularność w dziedzinie tworzenia architektury niewymagającej serwera, z wykorzystaniem usług obejmujących np. AWS Lambda. Więcej informacji na temat takiej architektury znajdziesz w rozdziale 6.

Ponieważ funkcjonalność i modele kosztów dostawców publicznej chmury są bardzo odmienne, warto dowiedzieć się, jak opracować projekt architektoniczny dla natywnej chmury.

## Architektura dla natywnej chmury

W miarę wzrostu popularności chmury zaczął pojawiać się trend natywnej chmury, w którym architektura systemu została zoptymalizowana pod kątem chmury. Typowa architektura lokalna jest zwykle tworzona dla określonej na stałe i niezmiennej infrastruktury. Dlatego też dodawanie do niej nowych zasobów IT, np. serwerów, zabiera ogromną ilość czasu oraz wiąże się ze znacznym kosztem i wysiłkiem. Natomiast w przypadku chmury płacisz jedynie za faktyczne użycie zasobów, dzięki automatyzacji otrzymujesz łatwe w użyciu rozwiązanie, więc możesz np. na żądanie zmieniać (skalować) liczbę serwerów w górę lub dół bez przejmowania się długim cyklem nabywania zasobów. W architekturze natywnej chmury nacisk kładzie się przede wszystkim na skalowanie na żądanie, projekt rozproszony oraz zastępowanie uszkodzonych komponentów zamiast ich naprawiania.

Chmura publiczna to nie tylko infrastruktura. Większość dostawców publicznej chmury oferuje szeroką gamę usług zarządzanych, pozwalających użytkownikowi na zignorowanie infrastruktury i zadań związanych z jej obsługą techniczną. Na przykład AWS Lambda oferuje niewymagającą serwera platformę przetwarzania, której można używać do uruchamiania sprzętu bez konieczności zarządzania serwerem lub środowiskiem uruchomieniowym. Podobnie baza danych Amazon DynamoDB charakteryzuje się wysoką skalowalnością, a tworzone w niej tabele i przechowywane w niej dane nie wymagają zarządzania serwerem bazy danych. Dzięki usługom zarządzanym można bardzo łatwo i szybko tworzyć skalowalne aplikacje.

W architekturze natywnej chmury nieustannie są przeprowadzane zautomatyzowane operacje na potrzeby odzyskiwania po awarii, skalowalności, zapewnienia prawidłowego stanu rozwiązania oraz wysokiej dostępności. W tym celu są wykorzystywane możliwości ciągłej integracji, wdrożenia i automatyzacji infrastruktury. To zachęca do nieustannej optymalizacji aplikacji w kategoriach kosztów i wydajności działania przy użyciu w tym celu nowych możliwości chmury, udostępnianych i usprawnianych każdego dnia.

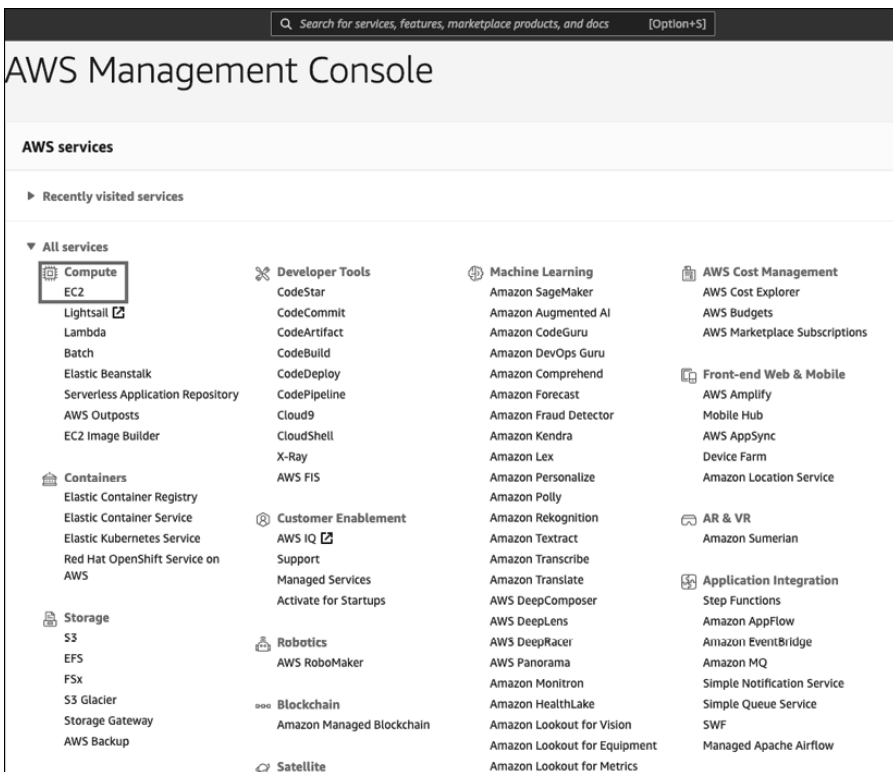
Znacznie dokładniejsze omówienie architektury natywnej chmury znajdziesz w następnym rozdziale.



## Dostawcy chmury publicznej i oferowane przez nich usługi

W branży IT mamy wielu dostawców publicznej chmury, wśród nich kluczowymi graczami są AWS, GCP, Microsoft Azure i Alibaba Cloud. Ci dostawcy oferują wiele usług: przetwarzanie w chmurze, pamięć masową, sieć, bazy danych, tworzenie aplikacji, analizę danych i uczenie maszynowe.

Na rysunku 1.5 pokazaliśmy konsolę AWS — zwróć uwagę na istnienie ogromnej liczby usług w wielu dziedzinach. Ramką została oznaczona usługa Amazon Elastic Compute Cloud (EC2), pozwalająca na tworzenie i uruchamianie w ciągu zaledwie minut nowych maszyn wirtualnych w chmurze AWS.



Rysunek 1.5. Konsola AWS i usługi oferowane przez tego dostawcę chmury

Dostawcy publicznej chmury zapewniają infrastrukturę i ogromną liczbę usług w różnych dziedzinach, takich jak analiza, uczenie maszynowe, blockchain (łańcuch bloków), robotyka, tworzenie aplikacji, poczta elektroniczna, zapewnianie bezpieczeństwa, monitorowanie i ostrzeganie. W publicznej chmurze różne możliwości techniczne stały się

łatwiej dostępne dla zespołu programistycznego, co pomaga w innowacji oraz skraca czas potrzebny na opracowanie produktu i jego wydanie na rynek.

Dostawcy publicznej chmury zapewniają infrastrukturę globalną na całym świecie, co ułatwia globalne skalowanie aplikacji, która tym samym może znaleźć się w pobliżu bazy jej użytkowników. Wszyscy dostawcy mają bezpłatny plan z wieloma zasobami dydaktycznymi. To pozwala zachęcić klientów do korzystania z usług chmury oraz umożliwia samodzielne wypróbowanie usług i ich poznanie.

## Podsumowanie

W rozdziale przedstawiliśmy w uproszczonej postaci definicję architektury rozwiązania stosowanej w standardach branżowych. Wyjaśniliśmy wagę architektury rozwiązania i to, jak może pomóc organizacji w osiągnięciu jeszcze lepszych wyników w kategoriach maksymalizacji zwrotu z inwestycji. Informacje zamieszczone w rozdziale pomogły Ci w zrozumieniu korzyści płynących ze stosowania architektury rozwiązania oraz wyjaśniły, jak taka architektura pomaga w różnych aspektach projektowania rozwiązania i podczas jego implementacji.

Podsumowując, architektura rozwiązania stanowi element konstrukcyjny w skomplikowanej organizacji i jest używana do uwzględnienia potrzeb wszystkich interesariuszy, a także do wypracowania standardu w celu wypełnienia luki między wymaganiami biznesowymi i rozwiązaniami technicznymi. Dobra architektura rozwiązania będzie uwzględniała nie tylko wymagania funkcjonalne, ale również przeanalizuje je w dłuższej perspektywie oraz weźmie pod uwagę wymagania inne niż funkcjonalne, takie jak skalowalność, wydajność działania, odporność na awarie, wysoka dostępność i odzyskiwanie po awarii. Architektura rozwiązania pomaga znaleźć optymalne rozwiązanie uwzględniające ograniczenia związane z kosztami, zasobami, ramami czasowymi, zapewnieniem bezpieczeństwa i zgodnością.

W rozdziale zostały przedstawione także podstawy przetwarzania w chmurze i architektury rozwiązania dla środowiska chmury, a także najważniejsi dostawcy publicznej chmury i wybrane z oferowanych przez nich usług. Przedstawiliśmy także ogólne omówienie różnych modeli przetwarzania w chmurze, np. IaaS, PaaS i SaaS, oraz modele wdrożenia w chmurze publicznej, prywatnej i hybrydowej. Ponadto ten rozdział rzucił nieco światła na ewolucję projektu architektury rozwiązania.

W następnym rozdziale skoncentrujemy się na roli architekta rozwiązań — przedstawimy różne typy architektów rozwiązań, omówimy odpowiedzialność w kategoriach architektury rozwiązania, a także wyjaśnimy, jak to wszystko mieści się w strukturze organizacji i środowisku zwinnym.



# Skorowidz

## A

- Active Directory, AD, 274, 282
  - biblioteka uwierzytelnienia, ADAL, 282
  - Domain Services, AD DS, 282
- ADFS, active directory federation services, 283
- adres
  - URI, 197, 292
  - URL, 193
- ADST, dynamic application security testing, 404
- AI, artificial intelligence, 60
- algorytmy
  - klastrowe, 478
  - uczenia maszynowego, 477
- Amazon
  - Athena, 450
  - CloudWatch, 364
  - Elastic MapReduce, 450
  - Neptune, 382
  - OpenSearch, 364
  - QuickSight, 452
  - Web Services Directory Service, 283
- AMI, amazon machine image, 326
- analiza
  - bezpieczeństwa, 403
  - danych, 446, 447
  - danych IoT, 505
  - głównej przyczyny problemu, 361
  - informacji, 155
  - operacji IT, ITOA, 359
  - RCA, 277
  - starej aplikacji, 543
  - wymagań użytkownika, 64
- antywzorce w architekturze rozwiązania, 235
- Apache
  - Flink, 445
  - Hadoop, 448
  - Kafka, 445
  - Spark, 448
  - Spark Streaming, 445
  - Zeppelin, 449
- API, application programming interface, 193, 554
- aplikacje
  - modernizacja, *Patrz* stare systemy
  - typu OLAP, 438
- AR/VR, augmented reality/virtual reality, 511
  - cyfrowy bliźniak, 513
- architekt
  - bezpieczeństwa, 61
  - biznesowy, 54
  - chmury, 55
  - danych, 59
  - DevOps, 62
  - infrastruktury, 57
  - promotor, 56
  - rozwiązań, 29, 54
    - atrybuty, 84
    - bycie mentorem, 593
    - bycie propagatorem technologii, 594
    - definiowanie celów i ważnych wyników, 583
    - definiowanie strategii działania, 583
    - elastyczność, 586
    - komunikacja z kierownictwem, 581
    - korporacyjny, 53
    - model obowiązków, 65
    - myślenie szerszymi kategoriami, 585
    - myślenie w kategoriach projektu, 587
    - nieustanne uczenie się, 590
    - obowiązki, 63
    - otwartość, 586
    - podejmowanie działań, 583
    - ponoszenie odpowiedzialności, 583
    - umiejętności związane z przedsięwzięciem, 579

- w organizacji stosującej metody
  - zwinne, 51, 75
  - zaangażowanie, 590
- sieci, 58
- techniczny, 55
- uczenia maszynowego, 60
- architektura
  - aplikacji, 571
    - głosowanie w czasie rzeczywistym, 204
    - obsługa bazy danych, 231
  - bazująca
    - na kontenerach, 230
    - na buforze, 212
    - na kolejce, 205
    - na zdarzeniach, 209
  - bezpieczeństwa, 573
  - bezstanowa, 189, 190
  - big data, 425, 453
    - najlepsze praktyki, 465
    - narzędzia i procesy, 429
    - pobieranie danych, 430
  - chmury, 137
    - hybrydowej, 176
    - natywnej, 48, 138, 178, 179
    - publicznej, 46
  - cyfrowego bliźniaka, 515
  - danych, 571
  - danych strumieniowanych, 464
  - informacji, 570
  - infrastruktury, 573
  - integracji, 572
  - internetu rzeczy, IoT, 495
    - na platformie AWS, 507
  - jeziora danych, 455, 458
  - kanapkowa WAF, 293
  - luźno powiązanych komponentów, 123
  - mesh, 462
  - metod zwinnych, 82
  - mikrouslug, 125, 202
  - monolityczna, 125
  - niewymagająca serwera, 199
  - niezawodność, 91, 115, 314, 319, 335
  - n-warstwowa, 183
  - potoku DevSecOps CI/CD, 421
  - repozytorium danych, 459, 461
  - RESTful
    - usługa sieciowa, 125, 196
  - rozwiązania, 30, *Patrz także* dokument architektury rozwiązania
    - atrybuty, 37
    - buforowanie, 215
    - buforowanie na warstwach, 213
    - unikanie antywzorców, 235
    - w publicznej chmurze, 44
    - wzorce projektowe, 182
    - wzorec buforowania aplikacji, 220
    - wzorec przepisano proxy, 219
    - wzorec serwera proxy, 218
    - wzorec wysokiej dostępności, 233
  - SOAP, 125, 192
    - usługa sieciowa, 196
    - witryna typu e-commerce, 198
  - stanowa, 189
  - strumienia zdarzeń, 211
  - ściśle powiązanych serwerów, 122
  - trójwarstwowa
    - wzorce buforowania, 215
  - typu CF, 512
  - uczenia maszynowego, 470, 484
  - wdrożenia z użyciem kontenerów, 229
  - wielodostępna, 188
  - wydajność działania, 240
  - zapewnianie niezawodności, 314, 319
  - zorientowana na usługi, SOA, 35, 124, 191
    - SOAP, 124
    - RESTful, 124, 196
- ASIC, application-specific integrated circuit, 249
- AST, application security testing, 403
- atak typu
  - CSRF, 291
  - DDoS, 116, 267, 289, 293
  - DoS, 289
  - eavesdropping, 307
  - MITM, 307
  - przepełnienie bufora, 291
  - SQL injection, 290
  - uszkodzenie pamięci, 291
  - XSS, 105, 290
- atomy Rydberga, Rydberg atoms, 528
- audyt, 275, 362
- automatyczna naprawa systemu, 315
- automatyczne
  - skalowanie, 200, 266
  - uaktualnienia, 296
- automatycznie skalowana grupa, 111
- automatyzacja, 134, 275, 316, 339
  - serwera ciągłej integracji, 413
  - wdrożenia, 135
  - zapewnienia bezpieczeństwa, 135
- autoryzacja, 104, 273

awarie, 341  
 kaskadowe, 118  
 AWS, Amazon Web Services, 137, 141  
 CloudFormation, 327, 363  
 CodeCommit, 364  
 Glue, 450  
 IoT Analytics, 506  
 IoT SiteWise, 510  
 Lambda, 364  
 System Manager, 363  
 Trusted Advisor, 363

## B

baza danych NoSQL, 261  
 bazująca na  
 dokumentach, 441  
 elementach typu klucz-wartość, 441  
 grafice, 441  
 kolumnach, 440  
 HBase, 449  
 bazy danych  
 nierelacyjne, 261  
 OLTP, 262  
 QLDB, 444  
 relacyjne, 231, 437  
 SQL, 439  
 typu klucz-wartość, 90  
 wybór, 260  
 zarządzania konfiguracją, CMDB, 346  
 bezpieczeństwo, 103, 272, 538  
 aplikacji, 355  
 centrum danych, 133  
 danych, 106, 133, 301, 355  
 infrastruktury, 106, 297  
 internetowe, 105  
 serwera, 355  
 sieci, 133, 355  
 w chmurze, 310  
 w każdym miejscu, 133  
 warstwy sieciowej, 288  
 biblioteka  
 MXNet, 492  
 TensorFlow, 492  
 biblioteki współdzielone, 555  
 big data  
 projektowanie architektury, 453  
 bit, 520  
 bity na sekundę, bps, 243  
 blokady, 244  
 BRA, role-based authentication, 278

brama internetowa, 298  
 broker komunikatów, 206  
 budżet, 38, 370  
 bufor  
 aplikacji, 214  
 bazy danych, 214  
 DNS, 214  
 internetowy, 214  
 przeglądarki WWW, 246  
 strony, 246  
 buforowanie, 212, 246  
 po stronie klienta, 213  
 w architekturze rozwiązania, 215

## C

CapEx, capital expenditure, 32  
 CDN, content delivery network, 89, 116,  
 214, 293  
 cel biznesowy, 37  
 centrum dystrybucji klucza, KDC, 280  
 certyfikaty  
 bezpieczeństwa, 309  
 zgodności, 309  
 CF, connected factory, 511  
 chmura  
 korzyści, 138  
 migracja  
 aplikacji z kodem współdzielonym,  
 554  
 rozwiązania typu mainframe, 552  
 samodzielnej aplikacji, 553  
 starych systemów, 550  
 chmury  
 hybrydowe, 45, 175, 176  
 natywne, 143, 146, 178  
 ponowny zakup, 147  
 refaktoryzacja, 147  
 publiczne, 44, 137  
 AWS, 141  
 GCP, 141  
 Microsoft Azure, 141  
 prywatne, 45  
 ciągła integracja i ciągłe wdrażanie, CI/CD,  
 62, 103, 296, 390, 392  
 ciągłe monitorowanie i usprawnianie, 395  
 ciągłe testowanie  
 implementacja, 408  
 ciągłe wdrażanie, CD  
 implementacja strategii, 404  
 CM, configuration management, 398

CNN, convolution neural networks, 492  
 CRM, customer relationship management, 169, 549  
 CSRF, cross-site request forgery, 291  
 cyfrowy bliźniak, 513  
 cykl życiowy  
   architektury IoT, 498  
   operacji dostarczenia rozwiązania, 72  
   projektu, 41  
   sprintu, 79  
   zasobu IT, 347  
   zdarzenia wdrożenia, 416

## D

dane  
   o ograniczonym dostępie, 302  
   osobowe, 107, 277  
   prywatne, 302  
   publiczne, 302  
   strumieniowane, 464  
 DAS, direct-attached storage, 258  
 daty docelowe, 38  
 DDoS, distributed denial of service, 289  
 definicja ukończenia, 80  
 DevOps, development and operations, 389  
   bezpieczeństwo, 401  
   bezpieczeństwo potoku CI/CD, 402  
   ciągła integracja i ciągłe wdrażanie, CI/CD, 392  
   ciągłe monitorowanie i usprawnianie, 395  
   edytor kodu źródłowego, 412  
   infrastruktura jako kod, 396  
   komponenty, 392  
   korzyści, 390  
   najlepsze praktyki, 418  
   serwer ciągłej integracji, 413  
   w chmurze, 420  
   zarządzanie kodem źródłowym, 412  
   zarządzanie konfiguracją, CM, 398  
 DevSecOps, 420  
 DiD, defence-in-depth, 274  
 DMS, Data Migration Service, 466  
 DNS, domain name service, 116  
 docelowy  
   czas odzyskiwania, RTO, 74, 95, 115, 234, 318, 341  
   punkt odzyskiwania, RPO, 74, 95, 115, 234, 318, 341  
 Docker, 251

dokument architektury rozwiązania, SAD, 559  
   dodatek, 575  
   implementacja rozwiązania, 574  
   kontekst biznesowy, 566  
   ogólne omówienie, 564  
   omówienie architektury, 570  
   omówienie koncepcji rozwiązania, 569  
   przeznaczenie, 560  
   widoki, 561  
   zarządzanie rozwiązaniem, 575  
 dokumentacja, 551, 576  
 dokumenty RFX, 576  
   prośba o informację, RFI, 34, 576  
   prośba o odpowiedź, RFR, 64  
   prośba o propozycje, RFP, 34, 576  
   prośba o wycenę, RFQ, 576  
 DoS, denial of service, 289  
 dostarczenie rozwiązania, 71  
 dostawcy chmury, 49, 177  
 dostęp użytkowników, 355  
 dostępność, 91  
 dowód koncepcji, POC, 71  
 drzewo decyzyjne, 478  
 dysk półprzewodnikowy, SSD, 246  
 dzienniki zdarzeń, 299

## E

EC2, Elastic Compute Cloud, 250, 414  
 e-commerce, 198  
 EDGE, 274  
 edytor kodu źródłowego, 412  
 egzemplarz serwera, 248  
 Elasticsearch, 263, 441  
 elastyczność floty serwerów, 89  
 ELB, Elastic Load Balancing, 219  
 ETL, extract, transform, load, 175

## F

FaaS, function as a service, 47, 200, 254  
 FIM, federated identity management, 279  
 projektowanie architektury danych, 428  
 FPGA, field-programmable gate array, 249  
 framework DevOps, 388  
 funkcja jako usługa, FaaS, 47, 200, 254  
 funkcjonowanie w chmurze, 363

**G**

GCP, Google Cloud Platform, 137, 141  
 globalne wymagania dotyczące zgodności, 31  
 gwarantowany poziom świadczenia usług, SLA, 319, 338

**H**

Hadoop User Experience, 448  
 HBase, 449  
 HCI, human/computer interaction, 410  
 HIPAA, 355  
 Hive, 449  
 HPC, high-performance computing, 267  
 HSM, hardware security module, 306  
 hurtownia danych, 438, 449

**I**

IaaS, infrastructure as a code, 63, 394, 396  
 IaaS, infrastructure as a service, 46, 318  
 IAM, identity and access management, 133, 280, 467  
 IAST, interactive application security testing, 404  
 ICMP, internet control message protocol, 298  
 IDS, intrusion detection system, 58, 274  
 implementacja  
   ciągłego testowania, 408  
   cyfrowego bliźniaka, 513  
   rozwiązania, 574  
   strategii CD, 404  
 incydenty  
   dotyczące bezpieczeństwa, 277  
   kategoryzacja priorytetów, 356  
 infrastruktura  
   IT, 135  
   jako kod, IaaS, 63, 394, 396  
   jako usługa, IaaS, 46, 318  
   klucza publicznego, PKI, 307  
   niemodyfikowalna, 120  
 interesariusz, 67  
 internet rzeczy, IoT, 59, 495  
   analizowanie danych, 505  
   architektura CF, 511  
   komponenty, 507  
   komponenty architektury, 498  
   kontrolowanie urządzeń, 503

  nawiązywanie połączenia, 503  
   rozwiązania przemysłowe, 509  
   urządzenia, 496  
   w chmurze, 507  
   zarządzanie urządzeniami, 499  
 inżynieria danych, 424  
 IOPS, 373  
 IoT, internet of things, 59, 495  
 IPS, intrusion prevention system, 58, 274  
 IPsec, internet protocol security, 308  
 ITAM, it asset management, 344  
 ITOA, it operations analytics, 359

**J**

jakość dostawy, 38  
 JDBC, java database connectivity, 410  
 jednostka  
   mikrokontrolera, MCU, 499  
   mikroprocesora, MPU, 499  
 jezioro danych, 447, 455  
 język  
   SQL, 290  
   YAML, 201  
 JSON, javascript object notation, 35, 197  
 JWT, JSON Web Token, 287

**K**

k-najbliższych sąsiadów, 478  
 KDC, key distribution center, 280  
 Kerberos, 280  
 klonowanie systemu operacyjnego, 167  
 klucz sesji TGS, 281  
 kluczowe wskaźniki wydajności działania, KPI, 59, 159, 315  
 KMS, key management service, 306, 467  
 kolejka, 206  
   komunikatów, 556  
 komponent implementacji rozwiązania, 32  
 komputery kwantowe  
   bramki kwantowe, 522, 525  
   Hadamarda, 524  
   Pauliego, 523  
 D-Wave, 528  
 kubit, 518  
 kwantowe wyzarzanie, quantum annealing, 527  
 mechanizm działania, 521  
 obliczenia kwantowe, 529  
 splątanie, 520, 522



superpozycja, 519, 522  
typy, 528  
układy kwantowe, 525  
w chmurze, 530  
komunikat, 206  
konfiguracja automatycznego skalowania,  
268  
konsola AWS, 49  
konsument, 206  
kontenery, 226, 251  
wdrażanie aplikacji, 225, 228  
kontrola dostępu, 298  
kontroler Jenkins, 415  
koperta  
SOAP, 192  
szyfrowania, 305  
kopia zapasowa, 325  
przywracanie systemu, 326  
kopiowanie  
danych użytkownika, 168  
dysku, 168  
maszyny wirtualnej, 168  
systemu operacyjnego, 168  
koszt, 107, 366, 381 *Patrz także*  
optymalizacja kosztów  
całkowity własności, TCO, 367  
obsługi technicznej, 368  
zasobów ludzkich i szkoleń, 368  
koszty operacyjne, OpEx, 32, 367  
KPI, key performance indicators, 59, 159,  
315  
kryptografia klucza publicznego, 299  
Kubernetes, 252  
kubity, 518, 520  
nadprzewodnikowe, 528

## L

lampka sygnalizacyjna, 327  
LAN, local area network, 58  
las losowy, 478  
LDAP, lightweight directory access  
protocol, 282  
liczba operacji wejścia-wyjścia na  
sekundę, IOPS, 243, 373  
logowanie pojedyncze, SSO, 273, 279  
luki w zabezpieczeniach, 288, 295, 403  
luźne powiązanie komponentów, 122

## Ł

łańcuch bloków, blockchain, 443  
łatwość obsługi technicznej rozwiązania,  
33, 102

## M

macierz  
arkuszy RACI, 158  
bramek bezpośrednio programowalna,  
249  
RAID, 259  
magazyn danych, 436  
łańcucha bloków, 443  
niestrukturalnych, 442  
obiektowy, 456  
strumieniowany, 444  
z wyszukiwaniem, 441  
maksymalny zwrot z inwestycji, ROI, 36,  
366  
mapowanie architektury lokalnej, 163,  
164  
masowe przetwarzanie równoległe, MPP,  
249, 438  
maszyna wektora nośnego, support vector  
machine, SVM, 478  
maszyny wirtualne, 226  
medium transmisyjne sieci, 241  
megabajty na sekundę, Mbps, 243  
mesh, 462  
metoda MoSCoW, 132  
metody  
zespołowe, 478  
zwinne, 75  
manifest, 77  
proces, 78  
terminologia, 78  
MFA, multi-factor authentication, 275  
Microsoft Azure, 141  
migawka bazy danych, 321  
migracja do chmury, 137  
analiza informacji, 155  
aplikacji, 165  
danych, 166  
działanie aplikacji, 171  
etapy, 152  
integracja, 169  
obciążenia, 153  
optymalizacja aplikacji, 173  
planowanie, 157

- migracja do chmury
    - pozostawienie niektórych aplikacji, 148
    - projektowanie aplikacji, 161
    - serwera, 167
    - strategia, 142
    - typu live, 170
    - typu podniesienie i przesunięcie, 144
    - weryfikacja, 169
    - wybór strategii, 150
    - wycofanie, 149
    - zakończenie procesu, 169
  - mikrotesty wydajności, 411
  - mikrouługi, 202
  - minimalna niezbędna funkcjonalność, 132
  - MITM, man-in-the-middle, 307
  - ML, machine learning, 60, 470
  - MLOps, machine learning operations, 487
    - najlepsze praktyki, 488
    - reguły, 487
  - model
    - kaskadowy, 81
    - producenta i subskrybenta, 210
    - strumienia zdarzeń, 211
  - modele uczenia maszynowego
    - nadzorowane, 477
    - nienadzorowane, 477
  - monitorowanie
    - aplikacji, 351
    - bezpieczeństwa, 134, 353
    - dzienników zdarzeń, 353
    - infrastruktury, 350
    - kosztu, 381
    - nieustanne, 275
    - platformy, 352
    - pojemności, 317
    - stanu systemu, 349
    - wydajności, 269
  - możliwości rynkowe, 38
  - MPP, massively parallel processing, 249, 438
- N**
- nadmiarowość, 93
  - nakłady
    - kapitałowe, CapEx, 367, 372
    - operacyjne, OpEx, 32, 367
  - narzędzia
    - do zarządzania konfiguracją, 399
    - programowania zwinnego, 80
    - typu DAST, 404
    - typu IAST, 404
    - typu SAST, 403
  - narzędzie
    - Amazon GuardDuty, 355
    - Amazon Kinesis, 445
    - BI, business intelligence, 427
    - Chef, 63, 400
    - CloudWatch, 384
    - DAEMON Tools, 295
    - Docker Swarm, 254
    - Ganglia, 450
    - Jaspersoft, 452
    - Kibana, 452
    - Power BI, 452
    - Puppet, 63, 400
    - Splunk, 384
    - Spotfire, 452
    - Supervisord, 295
    - System V init, 295
    - systemd, 295
    - Tableau, 452
  - NAS, network-attached storage, 257
  - nastawienie na usługi, 124
  - NAT, network address translation, 297
  - negocjacja SSL, 290
  - NFR, non-functional requirements, 42, 65
  - niezawodność architektury, 91, 115, 314, 319, 335
  - notacja
    - CIDR, 297
    - Diraca, 518
  - notatnik Jupyter, 450
- O**
- OAuth, 285
  - obiektowa pamięć masowa, 442
  - obliczenia kwantowe, QC, 517
    - w chmurze, 530
    - w rzeczywistych zastosowaniach, 529
  - obraz AMI, 326
  - obsługa
    - powiadomień, 356
    - techniczna, 74
  - ochrona danych, 276
  - odporność na awarie, 93, 319
  - odpowiedzialność współdzielona, 310
  - odzyskiwanie po awarii, 95, 168, 318, 323
    - kopia zapasowa, 325
    - lampka sygnalizacyjna, 327

- najlepsze praktyki, 334
- Warm Standby, 330
- wiele lokalizacji, 332
- ograniczenia
  - architekturalne, 68, 131
  - rozwiązań, 40
- ograniczone konteksty, 202
- OLAP, online analytical processing, 262, 438
- OLTP, online transaction processing, 260
- OpenID Connect, 285
- operacja
  - AST, 403
  - SCA, 403
- operacje
  - pobierania danych, 431
  - wejścia-wyjścia, 154
  - wejścia-wyjścia na sekundę, IOPS, 119
- OpEx, operational expenditure, 32, 367
- opóźnienie, 241
  - dyskowe, 243
- oprogramowanie jako usługa, SaaS, 47, 187
- optymalizacja kosztów, 107, 366
  - monitorowanie, 381
  - nadawanie tagów zasobom, 379
  - optymalizacja wydajności działania, 247
  - raportowanie, 381
  - reguły projektowe, 367
  - standaryzacja i zarządzanie, 378
  - techniki optymalizacji, 373
  - w publicznej chmurze, 385
  - zmniejszenie złożoności architekturalnej, 374
  - zwiększenie efektywności IT, 376
- organizacja grup użytkowników, 279
- oszacowanie budżetu, 369
- OWASP, 295

## P

- PaaS, platform as a service, 47
- pamięć masowa, 126–128, 255, 434
  - blokowa, 256
  - DAS, 258
  - obiekтовая, 257, 442
  - plikowa, 257
  - w chmurze, 257
- pamięć operacyjna, RAM, 212, 315
- Pig, 448

- Planning poker, 80
- planowanie, 344
  - pojemności, 113
  - procedur odzyskiwania po awarii, 96, 323
  - w chmurze, 363
- platforma
  - APU, 250
  - jako usługa, PaaS, 47
- płatności za faktycznie wykorzystane zasoby, 414
- pobieranie danych, 430
  - do chmury, 432
  - strumieniowanych, 444
  - technologie, 431
- POC, proof of concept, 71
- podejście
  - big data, 360
  - DevOps, 173
  - MVP, 132
  - niebieski-zielony, 171
- podniesienie i przesunięcie, 144
- pogłębiona ochrona, DiD, 274
- pomoc techniczna, 551
- PoP, point of presence, 117
- potok
  - analizy danych strumieniowanych, 464
  - big data, 427
  - ETL, 361, 447
  - kodu źródłowego, 417
  - uczenia maszynowego, 475
- potoki CI/CD, 393
  - narzędzia, 411
  - nieustanne testowanie, 409
  - podejście DevSecOps, 402
  - w DevOps, 394, 411, 421
- potrzeby biznesowe, 38
- Power BI, 452
- powiadomienia alarmowe, 357
- powtarzalny sposób działania, 118
- Presto, 449
- procedura odzyskiwania, *Patrz* odzyskiwanie po awarii
- proces
  - przełączenia, 170
  - zarządzania zasobami IT, 345
- procesor
  - CPU, central processing unit, 248
  - GPU, graphical processing unit, 249
  - sygnałowy, DSP, digital signal processor, 250

- producent, 206
  - prognozowanie wydatków, 369, 370
  - programowanie i operacje, DevOps, 389
  - projektowanie
    - aplikacji, 161
    - architektury natywnej chmury, 178
    - architektury hybrydowej chmury, 137
    - rozwiązań, 71
  - promowanie technologii, 75
  - propagacja sieci, 241
  - protokół
    - HTTP, 193
    - ICMP, 298
    - IMAP, 308
    - IPsec, 308
    - Kerberos, 280
    - LDAP, 282
    - SFTP, 308
    - SMTP, 308
    - SOAP, 35, 125, 192
    - SSH, 308
    - SSL, 106
    - TCP, 298
    - TLS, 106, 307
    - UDP, 298
  - przechowywanie danych, 434
    - strumieniowanych, 444
    - technologia, 436
  - przeniesienie, 145
  - przenośność i współdziałanie, 101
  - przepustowość
    - łącza, 242
    - sieci, 242
  - przeskoki sieci, 241
  - przetwarzanie danych, 446, 447
    - analityczne online, OLAP, 262
    - transakcyjne online, OLTP, 260
  - przewidywanie awarii, 341
  - punkt obecności, PoP, 117
- Q**
- QA, quality assurance, 316, 388
  - QC, quantum computing, 517
  - QoS, quality of service, 503
- R**
- RAID, redundant array of independent disks, 259
  - ramy czasowe projektu, 33, 38
  - raportowanie, 362, 381
  - RCA, root cause analysis, 277
  - reakcja na incydenty, 356
  - Redshift Spectrum, 461
  - refleksja UDP, 290
  - regresja
    - liniowa, 477
    - logistyczna, 478
  - reguła
    - ACID, 437
    - FLAIR, 428
    - najmniejszych uprawnień, 273
  - reguły projektu, 110
    - MLOps, 487
    - SOA, 198
  - rejestr wymagań, 80
  - rejestrwanie danych, 135
  - relacyjne bazy danych, 231, 437
  - replikacja danych, 321
    - asynchroniczna, 321
    - bazująca na
      - hipernadzorczy, 323
      - hoście, 323
      - macierzy dyskowej, 322
      - sieci, 322
    - synchroniczna, 321
  - repozytorium danych, 459, 461
    - warstwy architektury, 460
  - REST, representational state transfer, 35, 307
  - RESTful, representational state transfer, 125, 196
  - RFI, request for information, 34
  - RFP, request for proposal, 34
  - RFR, request for response, 64
  - RNN, recurrent neural networks, 492
  - RODO, 539
  - ROI, return on investment, 36
  - role architekta rozwiązań, 52, 57
  - routing DNS, 264, 274
  - rozproszona odmowa usług, DDoS, 105
  - rozszerzalność, 97
  - równoległość, 245
  - równoważenie obciążenia, 266
  - RPO, recovery point objective, 63, 74, 95, 234, 318, 341
  - RTO, recovery time objective, 63, 74, 95, 234, 318, 341
  - rzeczywistość wirtualna i rozszerzona, AR/VR, 511

**S**

- S3, Simple Storage Service, 200, 258
- SaaS, software as a service, 47, 187
- SAD, solution architecture document, 559
- SAML, Security Assertion Markup Language, 284
- SAMM Serverless Application Model, 201
- SAN, storage area network, 166, 256
- SAST, static application security testing, 403
- SCA, software composite analysis, 403
- scenariusz działania, 341
- schwyte jony, trapped ions, 528
- Scrum Master, 78
- semafory, 244
- serwer
  - ciągłej integracji, 413
  - DNS, 116, 214
  - przydzielający bilet, TGS, 280
  - uwierzytelniania, AS, 280
- sfera Blocha, 518
- sharding, 232
- sieciowa lista kontroli dostępu, NACL, 298
- sieć
  - CDN, 89, 116, 214, 293
  - LAN, 58
  - VPN, 58, 308
  - WAN, 58
- sieć neuronowa, 478
  - CNN, 492
  - RNN, 492
- silnik
  - buforowania
    - Memcached, 214, 221
    - Redis, 214, 221
  - wyszukiwania, 263
- skalowalność, 85
- skalowanie
  - architektury, 87
  - automatyczne, 266
  - bazy danych, 90
  - obciążenia, 111
  - operacji odczytu, 261
  - operacji zapisu, 261
  - pionowe, 86
  - poziome, 85
  - prognostyczne, 112
  - reakcyjne, 114
  - treści statycznej, 88
  - skanowanie
    - pakietów, 155
    - portów, 155
  - SLA, service-level agreement, 319, 338
  - SOA, service-oriented architecture, 35, 191, 198
  - SOAP, simple object access protocol, 35, 125, 192, 307
  - specjalizowany układ scalony, 249
  - sprawne działanie, 338
    - planowanie, 343
    - reguły projektowe, 339
    - w chmurze publicznej, 363
    - wyбір technologii, 342
  - sprawność operacyjna, 349
  - sprzętowy moduł bezpieczeństwa, HSM, 306
  - SQL, structured query language, 290
  - SSD, solid state drive, 246
  - SSH, secure shell, 308
  - SSL, secure socket layer, 106, 290
  - standard PCI DSS, 107
  - standaryzacja architekuralna, 375
  - stare systemy, 533
    - analiza, 543
    - brak zgodności, 539
    - dokumentacja, 537
    - koszt obsługi technicznej, 537
    - koszt uaktualnień, 537
    - modernizacja, 537, 540, 544
      - analiza kosztów i korzyści, 549
      - definiowanie strategii, 542
      - hermetyzacja, 547
      - korzyści, 540
      - migracja do chmury, 550, 551
      - migracja i usprawnienia, 546
      - ponowne opracowanie systemu, 545
      - ponowne utworzenie architektury, 548
      - ponowne zaprojektowanie, 549
      - refaktoryzacja, 548
      - rehosting, 547
      - sterowana architekturą, 545
      - techniki modernizacji, 546
      - zmiana platformy, 548
    - niedostateczne umiejętności, 537
    - oczekiwania użytkowników, 536
    - zapewnienie bezpieczeństwa, 538
  - strategia
    - CD, 404
    - migracji, 143, 150, 151
    - routingu DNS, 264
  - strefa dostępności, 92, 233

SYN Flood, 290

system

wykrywania włamań, IDS, 105, 274, 299

bazujący na goście, 300

bazujący na sieci, 301

operacyjiny FreeRTOS, 500

rozproszony, 317

zapobiegania włamaniom, IPS, 105, 274, 299

pamięci masowej, 258

sztuczna inteligencja, AI, 60

szyfrowanie danych

AES, 303

DES, 303

w trakcie transmisji, 307

## Ś

śledzenie nakładów kapitałowych, 372

środowisko QA, 316

## T

tablica sprintu, 80

taśmy magnetyczne, 258

technologie pobierania danych

Apache Flume, 432

Apache Sqoop, 432

testowanie aplikacji, 135

kanarkowe, 121

testy

A/B, 410

akceptacji, 169

akceptacyjne użytkownika, UAT, 409

TFLOP, tera floating-point operation, 249

TGS, ticket-granting server, 280

TGT, ticket-granting ticket, 281

TLS, transport layer security, 106, 307

token JWT, 287

tożsamość użytkownika, 277

TPU, tensor processing unit, 249

trasy, 298

TTL, time to live, 213

tworzenie

architektury

bazującej na kontenerach, 230

chmury hybrydowej, 175

mikrouslug, 202

n-warstwowej, 183

niezawodnej, 91, 115, 314, 319, 335

opartej na zdarzeniach, 209

stanowej i bezstanowej, 189

uczenia maszynowego, 480

wielodostępnej, 187

infrastruktury niemodyfikowalnej, 120

kopii zapasowej, 325

planu migracji, 157

projektu pod kątem danych, 129

systemu rozproszonego, 317

typy danych, 127, 128

## U

uaktualnienia nienadzorowane, 296

uczenie

głębokie, 490

model sieci neuronowej, 492

układ warstw, 491

maszynowe, ML, 60, 470

algorytmy, 477

architektura, 484

biblioteka MXNet, 492

biblioteka TensorFlow, 492

budowa modelu, 482

nadawanie etykiet, 481

podejście MLOps, 487

przygotowanie, 481

trenowanie i dostrajanie, 482

tworzenie architektury, 480

w chmurze, 479

wdrażanie i zarządzanie, 483

wybór algorytmu, 482

zastosowania, 472

przez wzmacnianie, 477

UDP, user datagram protocol, 290

URI, uniform resource identifier, 197, 292

URL, uniform resource locator, 193

urząd certyfikacji, CA, 307

urządzenia

NAS, 257

SAN, 166, 256

usługa, 124

Amazon MSK, 445

AWS IoT Greengrass, 502

AWS Trusted Advisor, 348

EKS, Elastic Kubernetes Service, 229

Elasticsearch, 441

REST, 35

sieciowa, 192

w architekturze RESTful, 196

w architekturze SOAP, 196

usługi chmury AWS, 49, 363

usprawnianie działania systemu, 359  
w chmurze, 364  
uwierzytelnianie, 104, 273  
Kerberos, 281  
na bazie roli, RBA, 278  
OAuth, 285  
SAML, 284  
wielopoziomowe, MFA, 275  
użyteczność i dostępność, 99

## V

VMware Cloud, 146  
VPC Flow Logs, 299  
VPC, virtual private cloud, 296  
VPN, virtual private network, 58, 308

## W

WAF, web application firewall, 105, 116,  
274, 292  
WAN, wide area network, 58  
Warm Standby, 330  
warstwa  
aplikacji, 185  
bazy danych, 185  
danych, 185  
internetowa, 184  
kontenera, 227  
logiki, 185  
prezentacji, 184  
warstwy architektury repozytorium  
danych, 460  
wartość  
bieżąca netto, NPV, 154  
RPO, 63  
RTO, 63  
SPN, 281  
technologii, 37  
wdrażanie ciągłe, CD, 405  
niemodyfikowalne, 408  
typu czerwony-czarny, 407  
typu niebieski-zielony, 405  
w miejscu, 405  
wdrożenie kodu, 225  
opcje, 415  
widok  
biznesowy, 561  
danych, 563  
implementacji, 563  
logiczny, 562  
operacyjny, 563

procesu, 562  
wdrożenia, 563  
wielokrotne użycie, 97  
wirtualna  
chmura prywatna, VPC, 296  
sieć prywatna, VPN, 58, 308  
witryna internetowa typu e-commerce,  
198  
wizualizacja danych, 451  
technologie, 452  
właściciel produktu, 79  
wprowadzanie zmian, 340  
współbieżność, 244  
wstrzyknięcie kodu SQL, 105  
wybór  
bazy danych, 260  
dotyczący obliczeń, 248  
pamięci masowej, 255  
platformy technologicznej, 39  
technologii, 33, 70  
wycofanie zmian, 340  
wydajność, 247  
działania, 118, 240  
monitorowanie, 269  
usprawnianie, 264  
wykres typu burndown, 80  
wykrywanie  
portfolio, 154  
włamań, IDS, 58  
wymagania  
biznesowe, 32  
dotyczące infrastruktury IT, 33  
dotyczące użytkownika końcowego, 33  
inne niż funkcjonalne, NFR, 42, 65  
niefunkcjonalne, 84  
użytkownika, 64  
wyszukiwanie danych, 263  
wytrenowanie, 476  
nadmierne, 476  
niedostateczne, 476  
wzorce buforowania, 215  
wzorzec projektowy, 182  
bezpiecznika, 222  
buforowania aplikacji, 219  
grodziowy, 222  
łańcucha kolejkwania, 206  
obserwacji zadania, 208  
pływających adresów IP, 224  
serwera proxy, 217  
wysokiej dostępności bazy danych, 233  
zmiany nazwy, 216

**X**

XSS, cross-site scripting, 290

**Y**

YAML, yet another markup language, 201

**Z**

zabezpieczanie

aplikacji, 295

architektury, 277

systemu operacyjnego, 295

zapewnienie

działania produktu, 74

jakości, QA, 388

zgodności, 355

zapobieganie włamaniom, IPS, 58

zapora sieciowa aplikacji internetowej,

WAF, 105, 116, 274, 292

zarządzanie

dostarczaniem rozwiązań, 41

dostępem, 277, 280

federacyjne tożsamością, FIM, 279

katalogiem usług, 371

kluczami, 306

kodek źródłowym, 412

konfiguracją, CM, 346, 347, 398

rozwiązaniem, 575

tożsamością i dostępem, IAM, 280

urządzeniami IoT, 499

zapotrzebowaniem, 371

zasobami i kosztami, 41, 344

integracja, 345

konserwacja, 345

nabycie, 344

planowanie, 344

wycofanie, 345

zasoby możliwe do zastępowania, 120

zdarzenia, 209

zespoły rozproszone globalnie, 31

zespół

programistyczny, 79

Scrum, 78

zgodność z SOC, 103

zmiana

hosta, 144

platformy, 145

zwrot z inwestycji, 38

**Ź**

źródła danych, 425

**Ż**

żądania

aktualizacji, pull request, 413

informacji, RFI, 34, 576

odpowiedzi, RFR, 64

propozycji, RFP, 34, 576

wyceny, RFQ, 576



# PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion** 

# Jeśli chcesz mieć łatwe życie, nie zostawaj architektem

Zaha Hadid, wielokrotnie nagradzana architektka

Usługi natywnej chmury pozwalają na uzyskiwanie imponującej wydajności i skalowalności przy niskim koszcie. Świadome tego przedsiębiorstwa poszukują architektów rozwiązań chmurowych, którzy spełniają wysokie wymagania. Taka osoba musi posiadać rozległą znajomość technologii i umiejętność wiązania tej wiedzy z wymaganiami biznesu w sposób zapewniający maksimum korzyści.

Dzięki tej książce dowiesz się, jak tworzyć niezawodne, skalowalne i odporne rozwiązania, a także jak projektować systemy następnej generacji przeznaczone dla środowiska chmury. Poznasz efektywne strategie dla produktu i nauczysz się je w pełni implementować w swojej organizacji. Zrozumiesz też, w jaki sposób architekt rozwiązań wpisuje się w środowisko elastycznie działającej firmy. W tym wydaniu pojawiły się również nowe rozdziały, poświęcone takim technologiom jak internet rzeczy, obliczenia kwantowe, inżynieria danych i uczenie maszynowe. Znajdziesz tu uaktualnione informacje dotyczące architektury natywnej chmury i magazynu danych łańcucha bloków. W efekcie lektury nauczysz się tworzyć projekty skutecznych rozwiązań, spełniających zdefiniowane wymagania biznesowe.

## Najciekawsze zagadnienia:

- zadania architekta rozwiązań w korporacji
- implementacja reguł i wzorców projektowych
- strategie zabezpieczania architektury
- modernizacja starych aplikacji za pomocą chmury
- big data, uczenie maszynowe, IoT i obliczenia kwantowe w nowoczesnej architekturze

**Saurabh Shrivastava** jest architektem i wynalazcą. Obecnie pracuje w Amazon Web Services jako lider architektów rozwiązań globalnych. Pasjonuje się najnowszymi technologiami i ich wpływem na społeczeństwo.

**Neelanjali Srivastav** jest liderką technologiczną, menedżerką produktu i trenerką metodyk zwinnych. Obecnie pracuje w Amazon Web Services jako starszy menedżer produktu. Opracowuje i rozwija nowe produkty.

	<b>KOD KORZYŚCI</b> Sięgnij po więcej! ▶	
 <a href="http://helion.pl">helion.pl</a>	ISBN 978-83-8322-479-4	
 <b>HELION SA</b> ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 918 63 helion@helion.pl	 9 788383 224794	
Cena: 129,00 zł		

**Packty**