

Spis treści (podsumowanie)

Wprowadzenie	xxi
1 Architektura oprogramowania bez tajemnic. <i>Zaczynamy!</i>	1
2 Cechy architektury. <i>Znaj swoje właściwości</i>	39
3 Dwa prawa architektury oprogramowania. <i>Kompromisy... nic, tylko kompromisy!</i>	77
4 Komponenty logiczne. <i>Elementy konstrukcyjne</i>	117
5 Style architektury. <i>Kategoryzacja i filozofie</i>	159
6 Architektura warstwowa. <i>Separując zagadnienia</i>	175
7 Modularne monolity. <i>Bazujące na dziedziniu</i>	203
8 Architektura mikrojądra. <i>Konstruowanie dostosowań</i>	233
9 Zrób to sam. <i>Aplikacja podróżnicza LuzTravel</i>	261
10 Architektura mikrousług. <i>Kawałek po kawałku</i>	287
11 Architektura sterowana zdarzeniami. <i>Asynchroniczne przygody</i>	331
12 Zrób to sam. <i>Sprawdzanie swojej wiedzy</i>	383
Dodatek A: pozostałości. <i>Sześć najważniejszych zagadnień, których nie opisaliśmy</i>	407
Skorowidz	421

Spis treści (ten właściwy)

Wprowadzenie

Architektura oprogramowania jest trudnym zagadnieniem, więc Twój mózg będzie się starał oszukać Cię i przekonać, że nie jesteś w stanie się jej nauczyć. Będzie Ci podsuwał takie myśli: „Lepiej skoncentrować się na czymś ważnym, takim jak zjedzenie lanczu lub rozważania o tym, czy świnię umieją latać”. Na szczęście to Ty MOŻESZ oszukać swój mózg i przekonać go, że opanowanie architektury oprogramowania jest niezwykle ważne, a ten rozdział pokaże Ci, jak to zrobić.

Dla kogo jest ta książka?	xxii
Wiemy, co myślisz	xxiii
Wiemy, co myśli Twój mózg	xxiii
Metapoznanie — myślenie o myśleniu	xxv
Oto co MY zrobiliśmy	xxvi
To, co TY możesz zrobić, aby zmusić swój mózg do posłuszeństwa	xxvii
Przeczytaj to	xxviii
Rozdziały „zrób to sam”	xxx
Zespół recenzentów technicznych	xxxi
Wspólne podziękowania	xxxii
Indywidualne podziękowania	xxxiii

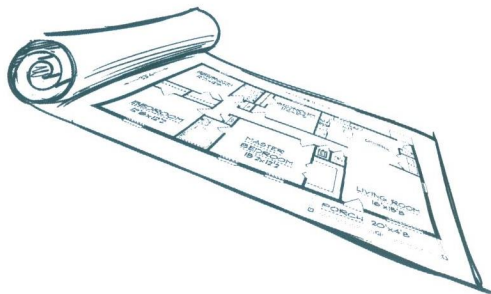
1

Architektura oprogramowania bez tajemnic

Zaczynamy!

Architektura oprogramowania ma fundamentalne znaczenie dla sukcesu systemu. Ten rozdział jest poświęcony omówieniu architektury oprogramowania. Czytając go, zrozumiesz, czym są wymiary architektury, i poznasz różnice pomiędzy architekturą oprogramowania a projektowaniem. Dlaczego jest to ważne? Ponieważ zrozumienie i stosowanie praktyk architektonicznych pomaga budować bardziej efektywne i poprawne oprogramowanie — oprogramowanie, które nie tylko lepiej funkcjonuje, ale także spełnia potrzeby i łagodzi obawy biznesu, a co więcej, zapewnia ciągłe poprawne działanie mimo bezustannych zmian, którym podlegają środowiska biznesowe i techniczne. Tak więc, bez dalszej zwłoki, zaczynamy.

Budowanie zrozumienia architektury oprogramowania	2
Plan budynku i architektura oprogramowania	3
Wymiary architektury oprogramowania	4
Rozgryzanie wymiarów	5
Pierwszy wymiar: cechy architektury	6
Drugi wymiar: decyzje architektoniczne	8
Trzeci wymiar: komponenty logiczne	10
Czwarty wymiar: style architektury	12
Z punktu widzenia projektu	16
Z punktu widzenia architektury	17
Spektrum pomiędzy architekturą a projektem	18
W którym miejscu spektrum wypada Twoja decyzja?	19
Strategiczna czy taktyczna	20
Wysoki czy niski nakład pracy	22
Kompromisy znaczące i mniej znaczące	24
Łączenie wszystkiego w całość	26
Udało Ci się!	27

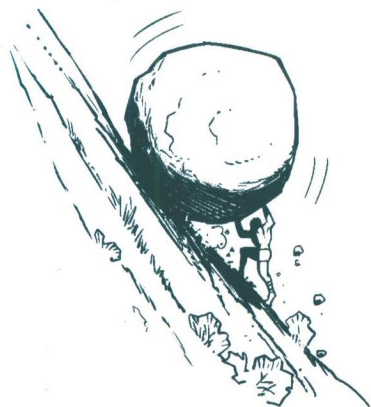


Cechy architektury

2

Znaj swoje właściwości

Co musi obsługiwać Twoja architektura? Cechy architektury (możliwości architektury) są podstawowymi elementami składowymi każdego systemu. Bez nich nie można podejmować decyzji architektonicznych, wybierać stylu architektury, a w wielu przypadkach nawet tworzyć logicznej architektury. W tym rozdziale dowiesz się, jak zdefiniować niektóre z bardziej powszechnych cech (takich jak skalowalność, niezawodność i testowalność), jak wpływają one na architekturę oprogramowania, jak pomagają w podejmowaniu decyzji architektonicznych i jak określić, które z nich są ważne w konkretnej sytuacji. Gotowy na dodanie pewnych możliwości do swojej architektury oprogramowania?



Cieszymy się razem!	40
Rozmowy w boksie	41
Czym są cechy architektury?	42
Definiowanie cech architektury	43
Cechy są aspektami projektu niezwiązanymi z dziedzina	44
Cechy mają wpływ na strukturę architektury	45
Ograniczaj cechy, by zapobiec nadmiernemu projektowaniu	46
Rozważ możliwości jawne i niejawne	48
Międzynarodowe Zoo Możliwości	49
Cechy architektury dotyczące procesu	50
Strukturalne cechy architektury	51
Operacyjne cechy architektury	52
Przekrojowe cechy architektury	53
Wyodrębnianie cech architektury z dziedziny problemu	58
Wyodrębnianie cech architektury ze świadomości środowiskowej	59
Określanie cech architektury na podstawie holistycznej wiedzy o dziedzinie	59
Złożone cechy architektury	61
Priorytety mają swój kontekst	62
Zagubiony w przekładzie	64
Cechy architektury i komponenty logiczne	66
Równoważenie kwestii związanych z dziedzina i cech architektury	67
Ograniczanie cech architektury	68

3

Dwa prawa architektury oprogramowania

Kompromisy... nic, tylko kompromisy!**Co się dzieje, gdy nie istnieje coś, co można by uznać za „najlepsze praktyki”?**

Tym, co w najlepszych praktykach jest najlepsze, jest to, że są one sposobami osiągnięcia określonych celów, które w znacznym stopniu nie są obciążone żadnym ryzykiem. Są one określane „najlepszymi” (a nie „lepszymi” lub „dobrymi”) nie bez powodu — wiadomo, że działają, więc dlaczego by ich nie użyć? Ale jedną rzeczą, której szybko nauczysz się o architekturze oprogramowania, jest to, że w jej przypadku nie ma czegoś takiego jak najlepsze praktyki. Będziesz musiał dokładnie przeanalizować każdą sytuację, aby podjąć decyzję, a w odniesieniu do tych decyzji komunikować nie tylko „co”, ale także „dlaczego”.

Jak więc poruszać się po tym nowym obszarze? Na szczęście masz do dyspozycji prawa architektury oprogramowania. Ten rozdział pokazuje, jak podczas podejmowania decyzji analizować kompromisy. Pokażemy również, jak tworzyć dokument ADR — zapisy decyzji architektonicznych — i w nim utrwalać „jak” oraz „dlaczego” decyzji. Pod koniec tego rozdziału będziesz dysponował narzędziami pozwalającymi poruszać się po niepewnym terytorium, jakim jest architektura oprogramowania.



Wszystko zaczyna się od aplikacji ze sneakersami	78
Co już wiemy?	80
Komunikacja z usługami serwerowymi	82
Analiza kompromisów	83
Analiza kompromisów: wersja z kolejkami	84
Analiza kompromisów: wersja z tematami	85
Pierwsze prawo architektury oprogramowania	86
Zawsze wszystko sprowadza się do kompromisów	88
Podejmowanie decyzji architektonicznych	89
Co jeszcze sprawia, że decyzja ma charakter architektoniczny?	90
Drugie prawo architektury oprogramowania	92
Dokumenty ADR — zapisy decyzji architektonicznych	93
Pisanie ADR: wybór odpowiedniego tytułu	95
Pisanie dokumentów ADR: jaki masz status?	96
Pisanie dokumentów ADR: określanie kontekstu	100
Pisanie dokumentów ADR: przedstawienie decyzji	101
Pisanie dokumentów ADR: rozważenie konsekwencji	103
Pisanie dokumentów ADR: zapewnianie nadzoru	105
Pisanie dokumentów ADR: uwagi końcowe	105
Korzyści ze stosowania dokumentów ADR	108
Sneakersowy Sezam okazał się sukcesem	109

4

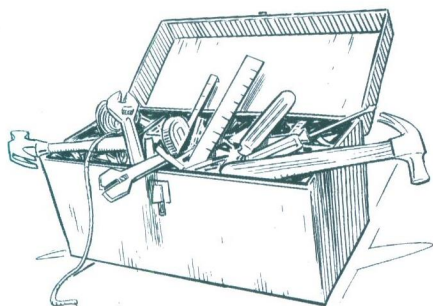
Komponenty logiczne

Elementy konstrukcyjne

Gotowy, by rozpocząć tworzenie architektury? Nie jest to tak łatwe, jak by się mogło wydawać, a jeśli nie zrobisz tego poprawnie, Twój system może się „rozsytać” — podobnie jak źle zaprojektowany wieżowiec lub most.

W tym rozdziale pokażemy kilka różnych rozwiązań problemu identyfikowania i tworzenia *komponentów logicznych* — funkcjonalnych elementów składowych systemów oprogramowania, opisujących sposób, w jaki poszczególne fragmenty systemu pasują do siebie nawzajem. Stosowanie opisanych tu technik ułatwi Ci stworzenie solidnej architektury — fundamentu, na którym będziesz w stanie zbudować udany system oprogramowania.

A zatem... załóż kask i rękawice, przygotuj narzędzia i zaczynajmy.



Ponowna prezentacja komponentów logicznych	118
Nazwij ten komponent	119
Aukcje na przygody online	120
Architektura logiczna a fizyczna	121
Tworzenie architektury logicznej	123
Krok 1. Określenie początkowych podstawowych komponentów	124
Podejście bazujące na przepływie pracy	126
Podejście bazujące na aktorach i akcjach	128
Pułapka encji	130
Krok 2. Przypisanie wymagań do komponentów	132
Krok 3. Analiza ról i odpowiedzialności	134
Dbanie o spójność	135
Krok 4. Analiza cech	136
Komponent Rejestracja oferty	138
Powiązania komponentów	139
Powiązania doprowadzające	140
Powiązania odprowadzające	141
Pomiar powiązań	142
System o ścisłych powiązaniach	144
Stosowanie prawa Demeter	145
Akt równoważenia	147
Kilka ostatnich słów o komponentach	148
Nazwij ten komponent	149

5

Style architektury

Kategoryzacja i filozofie

Istnieje wiele różnych stylów architektury. Każdy z nich powstał nie bez powodu i ma swoją własną filozofię dotyczącą tego, jak i kiedy powinien być używany. Zrozumienie filozofii danego stylu pomoże Ci ocenić, czy jest on odpowiedni dla danej dziedziny. Informacje zamieszczone w tym rozdziale stanowią fundament solidnej wiedzy o różnych rodzajach stylów architektury oprogramowania (które opiszemy znacznie dokładniej w dalszej części niniejszej książki). Jego lektura pomoże Ci zrozumieć wszelkie style architektury oprogramowania, z którymi będziesz miał okazję się zetknąć.

Dołóżmy zatem ten ostatni element układanki, co Ty na to?

Istnieje wiele stylów architektury	160
Świat stylów architektury	161
Podział: techniczny kontra dziedzinowy	162
Model wdrożenia: monolityczny kontra rozproszony	164
Monolityczne modele wdrażania: zalety	166
Monolityczne modele wdrażania: wady	167
Rozproszone modele wdrażania: zalety	168
Rozproszone modele wdrażania: wady	169
A oto i podsumowanie!	172



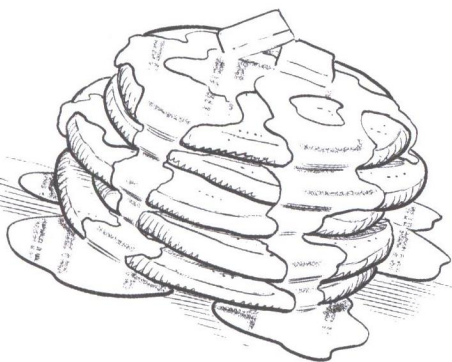
6

Architektura warstwowa

Separując zagadnienia**Co zrobić, jeśli problem jest prosty, a kluczowe znaczenie ma czas?**

Czy w ogóle warto zawracać sobie głowę architekturą? To zależy od tego, jak długo to, co aktualnie budujesz, ma działać. Jeśli jest to produkt jednorazowego użytku, w ogóle się nie przejmuj. Jeśli natomiast ma być używany przez dłuższy czas, to wybierz możliwie jak najprostszą architekturę, która zapewni jakąś organizację i korzyści, lecz nie będzie wywierać dużego wpływu na szybkość dostarczania produktu na rynek. Takie możliwości zapewnia *architektura warstwowa* — jest ona łatwa do zrozumienia oraz wdrożenia i pozwala na korzystanie z wzorców projektowych, które programiści już znają. Przyjrzyjmy się jej zatem warstwa po warstwie.

Naan Mniaam!: zbieranie wymagań	176
Wzorce projektowe — przypomnienie	178
Warstwy wzorca MVC	179
Warstwa po warstwie	182
Przekształcanie warstw na kod	183
Dziedziny, komponenty i warstwy	185
Zalety architektury warstwowej	188
Warstwy w realu: architektury fizycznej	189
Kompromisy architektury fizycznej	190
Ostatnie ostrzeżenie dotyczące zmian dziedziny	193
Supermoce architektury warstwowej	194
Kryptonit architektury warstwowej	195
Architektura warstwowa w ocenach	196
Podsumowanie	198



7

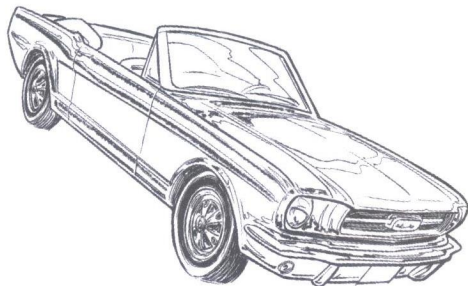
Modularne monolity

Bazujące na dziedzinie

Monolity można budować na więcej niż jeden sposób. Do tej pory spotkałeś się z architekturą warstwową, która zwraca uwagę na aspekty *techniczne*. Twoja przygoda z architekturą warstwowego monolitu może być długa i owocna, ale kiedy zmiany zaczną dotyczyć komunikacji i koordynacji pomiędzy różnymi zespołami, może się okazać, że będziesz potrzebował nieco więcej mocy i możliwości — a może nawet zupełnie innego stylu architektury.

W tym rozdziale przyjrzymy się stylowi architektury nazywanemu *modularnym monolitem*. Charakteryzuje się on tym, że aplikacje są dzielone w oparciu o *kwestie biznesowe*, a nie techniczne. Podczas lektury tego rozdziału dowiesz się, co oznacza taki sposób podziału, na co zwrócić uwagę w przypadku stosowania tego stylu architektury oraz jakie kompromisy się z nim wiążą. A zatem wypróbujmy ten modularny monolit, co Ty na to?

Modularny monolit?	207
Dziedzinowe problemy zmiany	209
Dlaczego modularne monolity?	210
Pokażcie mi kod!	212
Dbanie o modularność modułów	215
Rozszerzanie modularyzacji na bazy danych	219
Zwracaj uwagę na złączenia	221
Supermoce modularnych monolitów	222
Kryptonit modularnych monolitów	223
Modularny monolit w ocenach	224
Naan Mniaam! dostarcza pizze!	226



8

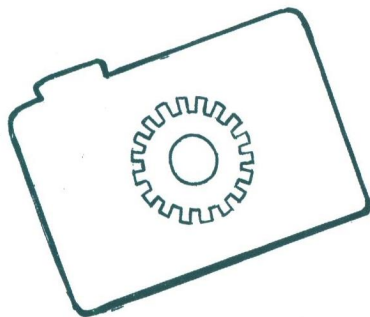
Architektura mikrojądra

Konstruowanie dostosowań**Możesz tworzyć niestandardowe doświadczenia, możliwość po możliwości.**

Niektóre style architektury są szczególnie dobrze dostosowane do niektórych możliwości, a jeśli chodzi o dostosowywanie, to prawdziwym mistrzem świata jest architektura mikrojądra. Oprócz tego świetnie się ona sprawdza w bardzo wielu różnych aplikacjach. Gdy zrozumiesz ten styl architektury, zaczniesz go widzieć wszędzie!

Przyjrzyjmy się zatem architekturze, która pozwala użytkownikom działać *po swojemu*.

Korzyści zapewniane przez RecyklKings	234
Dwie części architektury mikrojądra	237
Spektrum „mikrojądrowości”	239
Jądro usługi oceny urządzeń	241
Wtyczki hermetyzowane kontra wtyczki rozproszone	243
Komunikacja z wtyczkami	245
Kontrakty wtyczek	250
RecyklKings stawiają na ekologię	251
Supermoce mikrojądra	252
Kryptonit mikrojądra	253
Mikrojądro w ocenach	254
Podsumowanie	256



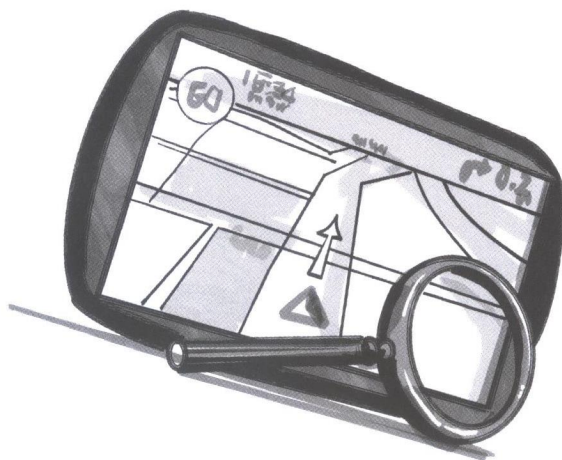
Zrób to sam

9

Aplikacja podróżnicza LuzTravel**Gotowy na rozszerzenie swojej podróży ku poznaniu architektury oprogramowania?**

W tym rozdziale wcielisz się w rolę architekta oprogramowania. Będziesz określać cechy architektury, budować architekturę logiczną, podejmować decyzje architektoniczne i decydować, czy użyć architektury warstwowej, modułowej, czy mikrojądra. Ćwiczenia zamieszczone w tym rozdziale dadzą Ci kompleksowy obraz tego, co robi architekt oprogramowania, i pokażą, jak wiele już się nauczyłeś. Przygotuj się do stworzenia architektury dla start-upu budującego witrynę przeznaczoną do integracji i organizacji podróży. *Bon voyage* — życzymy udanej podróży podczas tworzenia architektury.

Ułatwianie podróżowania	262
Przepływ pracy użytkownika LuzTravel	263
Planowanie architektury	264
Przewodnik architekta	265
Krok 1. Określenie cech architektury	266
Krok 2. Identyfikacja komponentów logicznych	268
Krok 3. Wybór stylu architektury	270
Krok 4. Dokumentowanie podjętych decyzji	272
Krok 5. Rysowanie diagramu architektury	274
Nie ma dobrych (ani złych) odpowiedzi	276

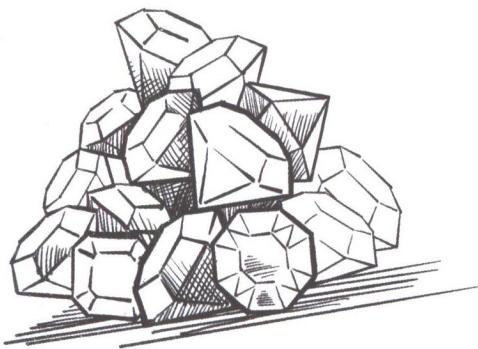


10

Architektura mikrousług

Kawałek po kawałku

Jak ułatwić sobie zmienianie architektury? Obecnie biznes zmienia się szybciej niż kiedykolwiek, a architektury oprogramowania muszą za nim nadążyć. W tym rozdziale dowiesz się, jak stworzyć elastyczną architekturę, która może być modyfikowana wraz ze zmianami biznesowymi, zapewni możliwość skalowania systemu wraz z rozwojem firmy i pozwoli mu działać nawet w przypadku awarii jego części. Zaintrygowany? Mamy nadzieję, że tak, ponieważ w tym rozdziale pokażemy Ci *mikrousługi* — styl architektury, który zapewni wszystkie te możliwości i parę innych. Naszą podróż przez krainę mikrousług odbędziemy... cóż... kawałek po kawałku.



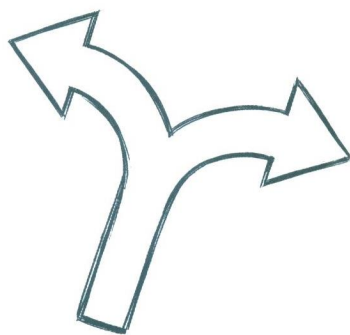
Jak się dzisiaj czujemy?	288
Czym jest mikrousługa?	291
To są moje dane, nie Twoje	292
Jak małe jest „mikro”?	294
Siły dezintegrujące	296
Dlaczego mielibyśmy zmniejszać mikrousługi?	297
Siły integrujące	298
Dlaczego mielibyśmy powiększać mikrousługi?	299
Wszystko sprowadza się do równowagi	300
Współdzielenie funkcjonalności	303
Wielokrotne użycie kodu dzięki zastosowaniu współdzielonej usługi	304
Wielokrotne użycie kodu dzięki zastosowaniu współdzielonej biblioteki	305
Zarządzanie przepływami pracy	309
Orkiestracja: dyrygowanie mikrousługami	310
Choreografia: zatańczmy	312
Supermoce architektury mikrousług	316
Kryptonit architektury mikrousług	317
Mikrousługi w ocenach	318
Podsumowanie	320

11

Architektura sterowana zdarzeniami

Asynchroniczne przygody

Co by było, gdyby Twoja architektura mogła robić wiele rzeczy w tym samym czasie? W miarę jak firmy rozwijają się i odnoszą coraz większe sukcesy, muszą także zapewniać możliwość obsługi coraz większej liczby użytkowników, i to bez spowalniania lub zawieszania się systemów. W tym rozdziale dowiesz się, jak projektować wydajne systemy, które można skalować wraz z rozwojem firmy. Przygotuj się na *architekturę sterowaną zdarzeniami*, bardzo popularny rozproszony styl architektury. Jest ona bardzo szybka, wysoce skalowalna i łatwa do rozbudowy, choć z drugiej strony jest także dość złożona. W tym rozdziale poznasz wiele nowych pojęć, w tym takie rzeczy jak zdarzenia, komunikaty i komunikacja asynchroniczna — ich znajomość jest niezbędna, jeśli chcesz nauczyć się tworzenia architektury, która może robić wiele rzeczy naraz. A więc zapnij pasy: wyruszmy na asynchroniczną przygodę z architekturą sterowaną zdarzeniami.



Zbyt wolno	332
Przyspieszanie obsługi	333
Der Nile rozwija się szybciej niż kiedykolwiek	334
Czym jest zdarzenie?	336
Zdarzenia a komunikaty	338
Zdarzenia inicjujące i pochodne	340
Czy ktokolwiek mnie słucha?	342
Komunikacja asynchroniczna	343
Odpal i zapomnij	345
Asynchroniczna rządzi	347
Synchroniczna rządzi	349
Topologie baz danych	351
Monolityczna baza danych	352
Bazy danych podzielone w oparciu o dziedzinę	354
Po bazie na usługę	356
Architektura EDA kontra mikrousługi	360
Hybrydy: mikrousługi sterowane zdarzeniami	364
Supermoce architektury sterowanej zdarzeniami	366
Kryptonit architektury sterowanej zdarzeniami	367
Architektura sterowana zdarzeniami w ocenach	368
Scalenie tego wszystkiego w całość	370
Podsumowanie	371

12

Zrób to sam

Sprawdzanie swojej wiedzy

Czy jesteś gotowy, by przetestować swoją wiedzę i umiejętności w zakresie tworzenia architektury rozproszonej? W tym rozdziale ponownie wcielisz się w rolę architekta oprogramowania. Będziesz określać cechy architektury, budować architekturę logiczną, podejmować decyzje architektoniczne i decydować, czy użyć mikrouslug, czy architektury sterowanej zdarzeniami. Ćwiczenia zamieszczone w tym rozdziale dadzą Ci kompleksowy obraz tego, co robi architekt oprogramowania, i pokażą, jak wiele się nauczyłeś. Przygotuj się do stworzenia architektury dla systemu przeprowadzania standaryzowanych testów dla uczniów o nazwie Testorama. Powodzenia — mamy nadzieję, że Twoja architektura zostanie oceniona na szóstkę!

Witamy w Testoramie	384
Przebieg realizacji testów	385
Planowanie architektury	386
Przewodnik architekta	387
Krok 1. Określenie cech architektury	388
Krok 2. Identyfikacja komponentów logicznych	390
Krok 3. Wybór stylu architektury	392
Krok 4. Dokumentowanie podjętych decyzji	394
Krok 5. Narysowanie diagramu architektury	396
Nie ma dobrych (ani złych) odpowiedzi	398

