Spis treści (podsumowanie)

Spis treści (ten właściwy)

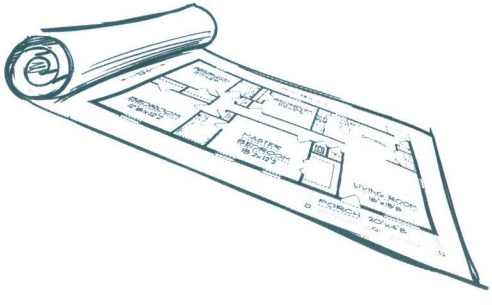
Wprowadzenie

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 świnie 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.

Architektura oprogramowania bez tajemnic

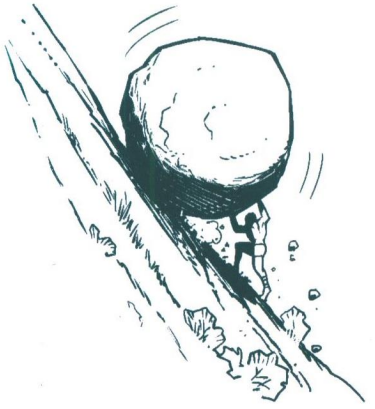
1 Zaczynajmy!

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, zaczynajmy.



2 Cechy architektury
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?



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 osiągania określonych celów, które w znacznym stopniu nie są obarczone 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".

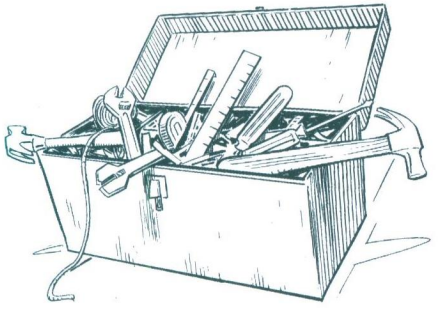
Jak więc poruszać się po tym nowym obszarze? Na szczęście masz do dyspozycji prawa architektury oprogramowania. Ten rozdział pokazuje, jak podczas podejmowania decyzji analizować kompromisy. Pokażemy również, jak tworzyć dokument ADR — zapisy decyzji architektonicznych — i w nim utrwalać „jak" oraz „dlaczego"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

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 „rozsypać" — 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.



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?

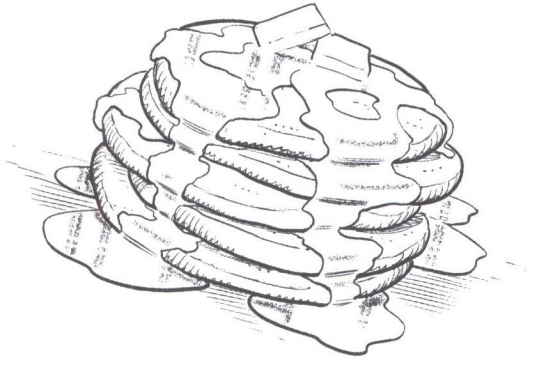
6

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.

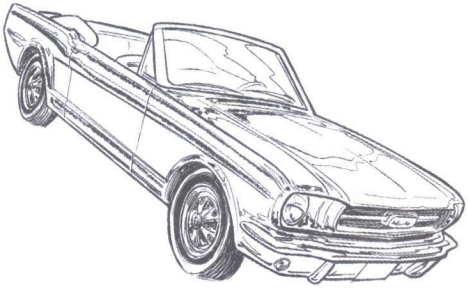


7

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?



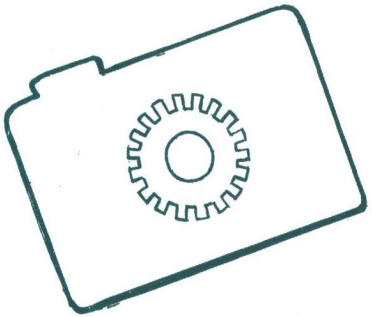
8

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*.

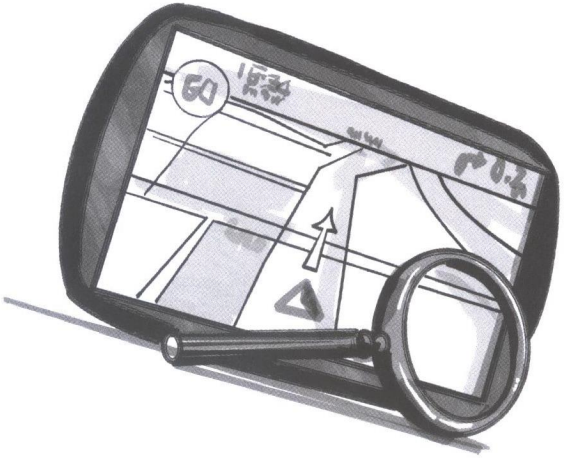


9

Zrób to sam

Aplikacja podróżnicza LuzTravel

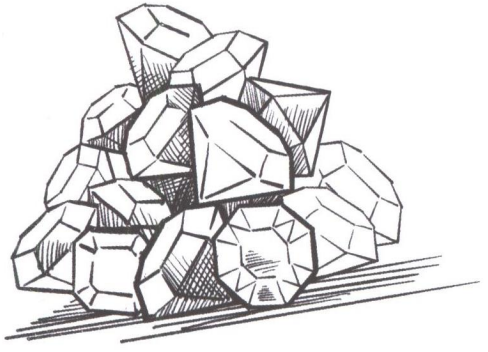
Gotowy na rozszerzenie swojej podróży ku poznaniu architektury oprogramowania? W tym rozdziale wcielisz się w rolę architekta oprogramowania. Będziesz określać cechy architektury, budować architekturę logiczną, podejmować decyzje architektoniczne i decydować, czy użyć architektury warstwowej, modułowej, 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.



10

Architektura mikrousług
Kawałek po kawałku

Jak ułatwić sobie zmienianie architektury? Obecnie biznes zmienia się szybciej niż kiedykolwiek, a architektury oprogramowania muszą za nim nadążać. 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 zapewnia wszystkie te możliwości i parę innych. Naszą podróż przez krainę mikrousług odbędziemy... cóż... kawałek po kawałku.

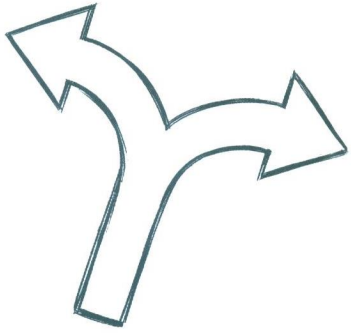


11

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.



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 mikrousług, 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!