13 Refaktoryzacja do mikroserwisów

Niniejszy rozdział dotyczy

- kiedy migrować aplikację monolityczną do architektury mikroserwisowej;
- dlaczego zastosowanie metody przyrostowej jest niezbędne w przypadku refaktoryzacji aplikacji monolitycznej do mikroserwisów;
- wdrażania nowych funkcjonalności jako usług;
- wyodrębniania usług z monolitu;
- integracji usługi i monolitu.

Mam nadzieję, że ta książka dobrze przedstawiła architekturę mikroserwisową, jej zalety i wady oraz to, kiedy z niej korzystać. Istnieje jednak spora szansa, że już pracujesz nad dużą, złożoną aplikacją monolityczną. Twoje codzienne doświadczenie w tworzeniu i wdrażaniu aplikacji jest powolne i bolesne. Mikroserwisy, które wydają się dobrze pasować do twojej aplikacji, wyglądają jak odległa nirwana. Podobnie jak Mary i reszta zespołu programistów FTGO, zastanawiasz się nad tym, jak do licha możesz zastosować architekturę mikroserwisową.

Na szczęście istnieją strategie, których można użyć, aby uciec od monolitycznego piekła bez konieczności przepisywania aplikacji od zera. Można stopniowo przekształcić swój monolit w mikroserwisy, tworząc tak zwaną aplikację dusiciel. Idea *aplikacji dusiciela* jest podobna do tzw. drzew dusicieli rosnących w lasach tropikalnych i owijających się wokół innych drzew, a czasem nawet je zabijających. Aplikacja dusiciel to nowa aplikacja składająca się z mikroserwisów, które tworzymy, implementując nową funkcjonalność jako usługę, a także wyodrębniając usługi z monolitu. Z czasem, gdy aplikacja dusiciel będzie zawierać coraz więcej funkcjonalności, spowoduje skurczenie się i ostatecznie zabicie monolitu. Ważną zaletą stworzenia aplikacji dusiciela jest to, że w przeciwieństwie do przepisywania całościowego aplikacji zapewnia wartość biznesową już na wczesnym etapie i często.

Ten rozdział zacznę od opisu motywacji do refaktoryzacji monolitu do architektury mikroserwisowej. Następnie opiszę, jak utworzyć aplikację dusiciel, realizując nowe funkcjonalności jako usługi i wydobywając usługi z monolitu. Potem omówię różne tematy projektowe, w tym jak zintegrować monolit i usługi, jak zachować spójność bazy danych w monolicie i usługach oraz jak postępować z bezpieczeństwem. Rozdział zakończę opisaniem kilku przykładowych usług. Jedną z nich będzie `Delayed Order Service`, która będzie implementować zupełnie nową funkcjonalność. Drugą z nich będzie `Delivery Service`, która zostanie wyodrębniona z monolitu. Zacznijmy od przyjrzenia się koncepcji refaktoryzacji do architektury mikroserwisowej.

13.1. Refaktoryzacja do mikroserwisów

Postawmy się w sytuacji Mary. Jesteśmy odpowiedzialni za aplikację FTGO, dużą i starą aplikację monolityczną. Firma jest wyjątkowo sfrustrowana niezdolnością inżynierów do szybkiego i niezawodnego dostarczania funkcjonalności. Wydaje się, że FTGO cierpi na klasyczny przypadek monolitycznego piekła. Prawdopodobnie mikroserwisy, przynajmniej na pozór, są odpowiedzią. Czy powinniśmy zaproponować przekierowanie zasobów programistycznych z rozwijania funkcjonalności od migracji do architektury mikroserwisowej?

Ten podrozdział zacznę od omówienia, dlaczego warto rozważyć refaktoryzację do mikroserwisów. Przedyskutuję również to, dlaczego ważne jest, aby mieć pewność, że problemy związane z tworzeniem oprogramowania są spowodowane tym, że znajdujemy się w monolitycznym piekle, a nie na przykład w kiepskim procesie tworzenia oprogramowania. Następnie opiszę strategie stopniowej refaktoryzacji monolitu do architektury mikroserwisowej. Potem omówię znaczenie wcześniejszych i częstszych ulepszeń w celu utrzymania wsparcia dla biznesu. W dalszej kolejności opiszę, dlaczego należy unikać inwestowania w zaawansowaną infrastrukturę wdrożeniową, dopóki nie opracuje się kilku usług. Na koniec przedstawię różne strategie, których można użyć do wprowadzenia usług do swojej architektury, w tym wdrażania nowych funkcjonalności jako usług i wydobywania usług z monolitu.