# 7 Implementowanie zapytań w architekturze mikroserwisowej

**Niniejszy rozdział dotyczy**

- wyzwań związanych z tworzeniem zapytań w architekturze mikroserwisowej;
- kiedy i jak implementować zapytania z użyciem wzorca kompozycji API;
- kiedy i jak implementować zapytania za pomocą wzorca Command Query Responsibility Segregation (CQRS).

Mary i jej zespół już zaczynali czuć się komfortowo z myślą o wykorzystaniu sag do zachowania spójności danych. Później jednak odkryli, że zarządzanie transakcjami nie było jedynym wyzwaniem związanym z rozproszonymi danymi, o które musieli się martwić podczas migracji aplikacji FTGO do mikrousług. Musieli także dowiedzieć się, jak zaimplementować zapytania.

W celu obsługi interfejsu użytkownika aplikacja FTGO realizuje wiele operacji zapytań. Implementowanie tych zapytań w istniejącej aplikacji monolitycznej jest proste, ponieważ dotyczy jednej bazy danych. W większości przypadków programiści FTGO musieli napisać instrukcje SELECT i zdefiniować niezbędne indeksy. Jak odkryła Mary, pisanie zapytań w architekturze mikroserwisowej jest trudnym zadaniem. Zapytania

często muszą wyszukiwać dane rozproszone w bazach danych należących do wielu usług. Nie można jednak użyć tradycyjnego mechanizmu zapytań rozproszonych, ponieważ nawet gdyby było to technicznie możliwe, narusza to enkapsulację.

Rozważmy na przykład zapytania dla aplikacji FTGO opisane w rozdziale 2. Niektóre zapytania pobierają dane, które są własnością tylko jednej usługi. Na przykład zapytanie `findConsumerProfile()` zwraca dane z `Consumer Service`. Ale inne zapytania FTGO, takie jak `findOrder()` i `findOrderHistory()`, zwracają dane należące do wielu usług. Implementowanie tych zapytań już nie jest takie proste.

Istnieją dwa różne wzorce wdrażania operacji zapytań w architekturze mikroserwisowej:

- *Wzorzec kompozycji API* – jest to najprostsze podejście i powinno być stosowane w miarę istniejących możliwości. Polega ono na tym, że klienci usług, będących właścicielami danych, są odpowiedzialni za wywoływanie usług i łączenie wyników.
- *Wzorzec Command Query Responsibility Segregation (CQRS)* – ma on większy potencjał niż wzorzec kompozycji API, ale za to jest również bardziej złożony. Utrzymuje on jedną lub więcej baz danych widoków, których jedynym celem jest obsługa zapytań.

Po przedyskutowaniu tych dwóch wzorców omówię sposób projektowania widoków CQRS, a następnie implementację przykładowego widoku. Zacznijmy od przyjrzenia się wzorcowi kompozycji API.

## 7.1. Tworzenie zapytań z użyciem wzorca kompozycji API

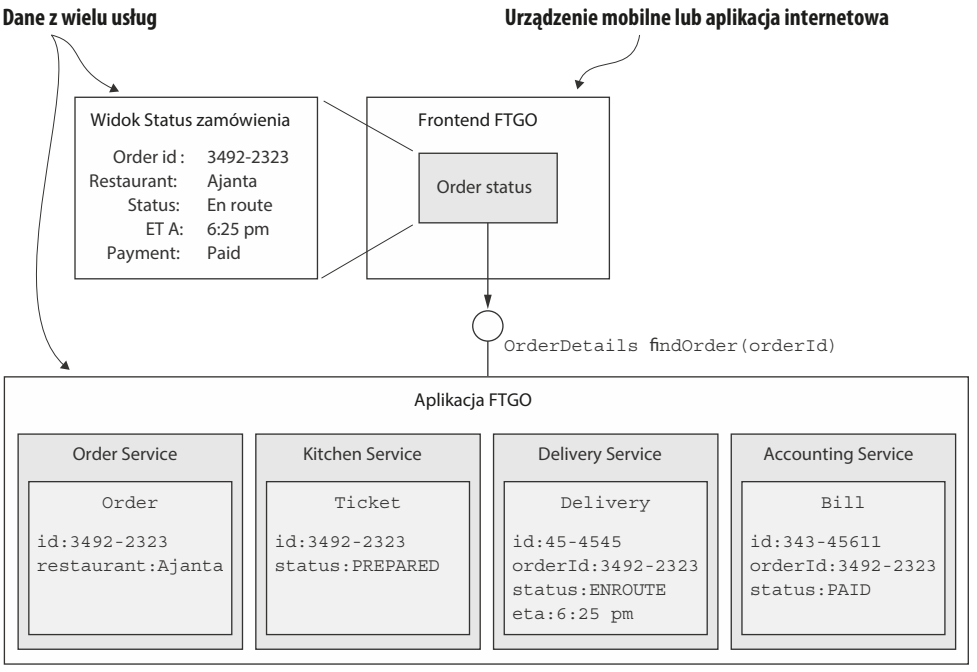
Aplikacja FTGO wykonuje wiele zapytań. Niektóre zapytania, jak wspomniano wcześniej, pobierają dane z jednej usługi. Wdrażanie tych zapytań jest zwykle proste – chociaż w dalszej części tego rozdziału, kiedy omówię wzorzec CQRS, zobaczymy przykłady zapytań dotyczących pojedynczej usługi, których wdrożenie jest trudne.

Istnieją również zapytania, które pobierają dane z wielu usług. W tym podrozdziale opiszę zapytanie `findOrder()`, które jest przykładem zapytania pobierającym dane z wielu usług. Wyjaśnię wyzwania pojawiające się często podczas implementacji tego typu zapytań w architekturze mikroserwisowej. Następnie opiszę wzorzec kompozycji API i pokażę, jak można go użyć do zaimplementowania zapytań, takich jak `findOrder()`.

### 7.1.1. Operacja zapytania findOrder()

Operacja `findOrder()` pobiera zamówienie według klucza głównego. Przyjmuje parametr `orderId` i zwraca obiekt `OrderDetails`, który zawiera informacje o zamówieniu. Jak pokazano na rysunku 7.1, ta operacja jest wywoływana przez moduł interfejsu użytkownika, taki jak urządzenie mobilne lub aplikacja internetowa, który implementuje widok *Status zamówienia*.

Informacje wyświetlane w widoku *Status zamówienia* obejmują podstawowe informacje o zamówieniu, w tym jego status, status płatności, status zamówienia z perspektywy restauracji oraz status dostawy, w tym jej lokalizację i przewidywany czas dostawy w przypadku konieczności transportu.




**Rysunek 7.1.** Operacja `findOrder()` jest wywoływana przez moduł frontendu FTGO i zwraca szczegóły `Order`

Ponieważ poszukiwane dane znajdują się w jednej bazie danych, monolityczna aplikacja FTGO może łatwo pobrać szczegóły zamówienia, wykonując pojedynczą instrukcję SELECT, która łączy różne tabele. Z kolei w wersji FTGO opartej na mikroserwisach dane są rozproszone w następujących usługach:

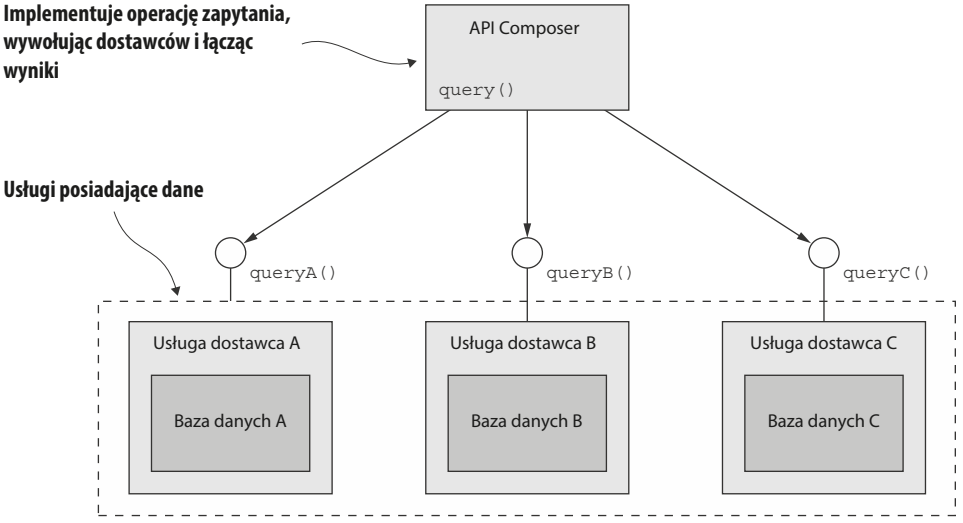
- `Order Service` – podstawowe informacje o zamówieniu, w tym szczegóły i status;
- `Kitchen Service` – status zamówienia z perspektywy restauracji i szacowany czas, kiedy będzie gotowe do odbioru;
- `Delivery Service` – status dostawy zamówienia, szacunkowe informacje o dostawie i jego bieżąca lokalizacja;
- `Accounting Service` – status płatności za zamówienie.

Każdy klient, który potrzebuje szczegółów zamówienia, musi zapytać o nie wszystkie te wymienione usługi.

### 7.1.2. Wzorzec kompozycji API

Jednym ze sposobów implementacji operacji zapytań, takich jak `findOrder()`, które pobierają dane należące do wielu usług, jest użycie wzorca kompozycji API. Ten wzorzec realizuje zapytanie przez wywołanie usług, które są właścicielami danych, i połączenie wyników. Rysunek 7.2 pokazuje strukturę tego wzorca. Ma on dwa typy uczestników:

- *API Composer* – realizuje operację zapytania przez wykonywanie zapytań do usług dostawców.
- *usługa dostawca* – jest to usługa, która jest właścicielem niektórych danych zwracanych przez zapytanie




**Rysunek 7.2.** Wzorzec kompozycji API składa się z API Composer i dwóch lub więcej usług dostawców. API Composer realizuje zapytanie, wysyłając zapytania do dostawców i łącząc wyniki

Rysunek 7.2 pokazuje trzy usługi dostawcy. API Composer realizuje zapytanie, pobierając dane z usług dostawców i łącząc wyniki. API Composer może być klientem, takim jak aplikacja internetowa, która potrzebuje danych do renderowania strony internetowej. Alternatywnie może to być usługa, taka jak brama API i jej wariant Backends for Frontends opisany w rozdziale 8, który wystawia operację zapytania jako punkt końcowy API.

**Wzorzec: Kompozycja API**

Realizuje zapytanie, które pobiera dane z wielu usług, wysyłając zapytanie do każdej usługi za pośrednictwem API i łącząc wyniki. Zobacz http://microservices.io/patterns/data/api-composition.html.

To, czy można użyć tego wzorca do implementacji określonej operacji zapytania, zależy od kilku czynników, w tym od sposobu partycjonowania danych, możliwości API udostępnianych przez usługi będące właścicielami danych oraz możliwości baz danych używanych przez te usługi. Na przykład, nawet jeśli *usługi dostawcy* mają API do pobierania wymaganych danych, agregator może wymagać wykonania nieefektywnego łączenia dużych zbiorów danych w pamięci. Później zobaczymy przykłady zapytań, których nie można zaimplementować z wykorzystaniem tego wzorca. Na szczęście istnieje wiele scenariuszy, w których ten wzorzec ma zastosowanie. Aby zobaczyć go w działaniu, przyjrzymy się przykładowi.