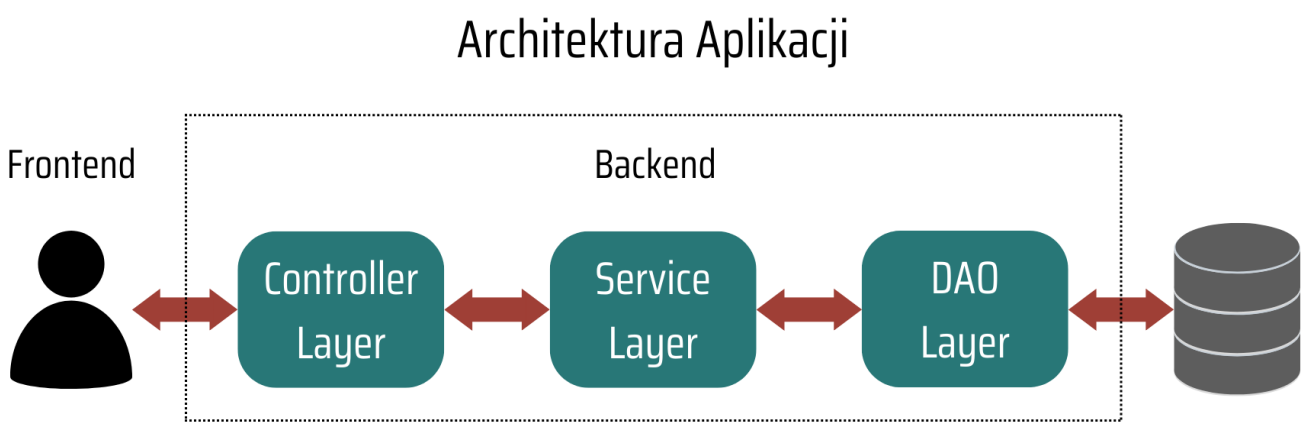
Car dealership - Projekt

Spis treści

Opis zagadnienia

W ramach tego zagadnienia oprzemy się na napisanym wcześniej projekcie **Car Dealership**. Wykorzystamy go, żeby móc napisać to samo przy wykorzystaniu Spring Data JPA. Wykorzystamy go do tego, żeby mieć odniesienie i być w stanie zauważyć jak dużo kwestii zostanie uproszczonych po dodaniu do projektu Spring Data Jpa.

Przypomnij sobie, że rozmawialiśmy wcześniej o tematyce architektury aplikacji i wspomnieliśmy wtedy o warstwowej architekturze aplikacji:



Obraz 1. Warstwowa Architektura Aplikacji

- Frontend - aplikacja, która uruchamia się użytkownikowi w przeglądarce internetowej,
- Controller Layer (*warstwa kontrolerów*) - ta część aplikacji będzie służyła do wystawiania backendu na świat i do tego, żeby frontend mógł z naszym backendem rozmawiać. Jeszcze tego nie umiemy, ale spokojnie, przejdziemy do tego,
- Service Layer (*warstwa usług*, *warstwa logiki biznesowej*) - ta część aplikacji będzie służyła do realizacji logiki biznesowej. Cały czas uczymy się technik, które pozwalają nam na realizację jakiejś logiki biznesowej,
- DAO Layer (*warstwa dostępu do danych*) - ta część aplikacji będzie odpowiedzialna za dostęp do szeroko rozumianych danych. Dane takie mogą być przetrzymywane w bazie danych lub nawet w pliku. W obrębie tego warsztatu oraz innych, gdzie poruszamy się w tematyce baz danych, tak na prawdę uczymy się cały czas o warstwie DAO oraz komunikacji z bazą danych.

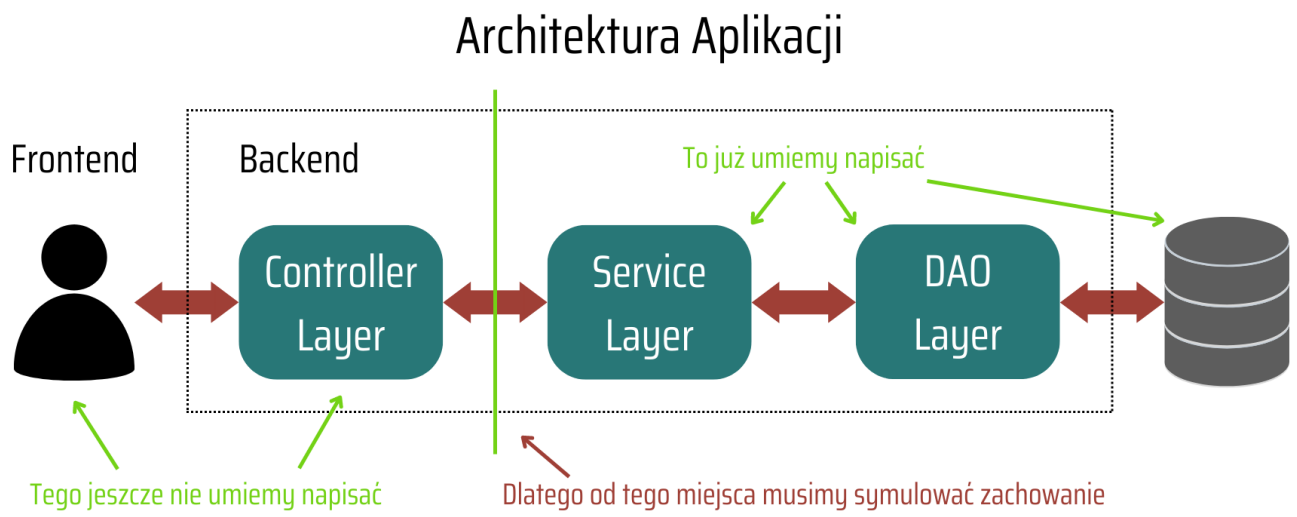
Chciałbym, żebyśmy w ramach tego zagadnienia przygotowali warstwy **DAO layer** oraz **Service Layer**, ale tym razem w oparciu oSpring Data JPA. Czyli napiszemy część aplikacji, która będzie odpowiedzialna za logikę biznesową takiej aplikacji oraz za zapisywanie danych w bazie danych. Opis logiki biznesowej był dostarczony wcześniej, czyli w obrębie tego projektu powinniśmy przepisać wypracowany wcześniej

kod, tak żeby korzystał ze Spring Data JPA i poznanych technik.

W praktyce użytkownicy takiej aplikacji podczas wykonywania swoich akcji na stronie (pobieranie danych, zapisywanie danych itp.) doprowadzaliby do modyfikacji danych w bazie danych. Użytkownik taki korzystałby z aplikacji frontendowej, a jego ruchy byłyby przekazywane do backendu. Backend przy wykorzystaniu warstwy **Controller Layer** odbierałby żądania od aplikacji frontendowej i realizował pewną logikę biznesową, która opierałaby się również o warstwę **DAO**.

Oczywiście cały ten opis to wielkie uproszczenie i o tym wszystkim będziemy się jeszcze uczyć. Szkopuł tego projektu polega na tym, że chcemy przygotować warstwę logiki biznesowej oraz warstwę repozytoriów bazodanowych, ale musimy zasymulować w jakiś sposób ruch użytkowników, żeby móc napisać tylko część prawdziwego backendu, bo całości jeszcze nie umiemy.

Nazywam to częścią, bo cały backend składałby się z kontrolerów, serwisów oraz repozytoriów. W ramach tego projektu napiszemy tylko serwisy i repozytoria, czyli musimy w jakiś sposób nagiąć wywołania serwisów, które byłyby wołane przez kontrolery - bo kontrolerów jeszcze nie znamy i ich nie napiszemy. Dlatego właśnie część aplikacji będziemy symulować. Mam nadzieję, że dobrze wyjaśni to poniższa grafika.



Obraz 2. Warstwowa Architektura Aplikacji

Dlatego właśnie został przygotowany plik `car-dealership-traffic-simulation.md`, który zawiera "zrzut" ruchu użytkowników w aplikacji. Napisz warstwy logiki biznesowej oraz repozytorium w taki sposób, żeby cały plik `car-dealership-traffic-simulation.md` mógł zostać wczytany i przetworzony poprawnie. Zwróć uwagę, że plik `car-dealership-traffic-simulation.md` jest mniejszy niż w poprzednim warsztacie. Wynika to z tego, że dane rozruchowe aplikacji mogą być dodane przy wykorzystaniu innego mechanizmu (pomyśl o czym mowa).

Czyli cały projekt będzie polegał na napisaniu aplikacji w oparciu o Spring Data JPA, tak żeby była ona w stanie wczytać i przetworzyć plik `car-dealership-traffic-simulation.md` i zapisać dane do bazy danych, a w efekcie na końcu mamy mieć uzupełnioną bazę danych, której projekt stworzyliśmy wcześniej.

Z racji, że w tym warsztacie rozmawialiśmy również o testach baz danych, to chciałbym, żeby wszystkie przypadki w aplikacji były uruchamiane z poziomu testów. Czyli nie napiszemy metody `main()`, tylko całość aplikacji ma być uruchamiana jako testy integracyjne. Poznaliśmy już techniki takie jak Flyway oraz testcontainers, zatem one również mają być przez nas wykorzystane.