Wyższa Szkoła Technologii Informatycznych w Katowicach

Temat pracy: Analiza sposobów komunikacji z bazą danych w wybranym środowisku programistycznym

Autor: Szymon Derleta

Promotor: dr Jadwiga Bakonyi

Kategorie: porównanie wydajności, bazy danych

Słowa kluczowe: usługa rest api, bazy danych, Java, SQL, Spring Framework

Referat pracy dyplomowej

1. Cel i podstawowe założenia

Celem pracy jest analiza wybranych sposobów komunikacji z bazą danych na przykładzie aplikacji pracującej jako usługa REST API oraz zrealizowanej w środowisku Java. Na potrzeby pracy zostanie napisana aplikacja typu REST API z wykorzystaniem zestawu bibliotek Spring, a następnie zostaną wdrożone wybrane sposoby komunikowania się z bazą danych. Do analizy zostaną wybrane trzy spośród proponowanych niżej rozwiązań:

- za pomocą interfejsu programowania JDBC,
- mapowanie ORM z wykorzystaniem standardu JPA oraz biblioteki Hibernate,
- wykorzystanie implementacji sterownika R2DBC dla środowiska Java,
- mapowanie ORM z wykorzystaniem biblioteki JOOQ,
- rozwiązanie EclipseLink służące do obsługi usług bazodanowych.

Na analizę każdego z wybranych rozwiązań będzie się składać:

- charakterystyka możliwych technik komunikacji w danym rozwiązaniu,
- testowanie komunikacji na warstwie dostępu do danych,
- testowanie autoryzacji dostępu do zasobów na podstawie ról przypisanych do użytkowników,
- testowanie punktów końcowych w usłudze REST, dostosowując zapytania do standardu HATEOAS.

2. Realizacja projektu

W ramach pracy zaprezentowano zagadnienia teoretyczne oraz technologie wykorzystywane do celów badań, w tym przedstawiono pojęcia takie jak usługa RESTful oraz środowisko Java. Dokonano również wyboru modelu bazy danych uwzględniając rodzaj licencji, popularność modelu bazodanowego oraz jego funkcjonalność. Do celów badań wybrano relacyjną bazę danych MySQL w wersji 8. Do badań wybrana została testowa baza danych *employees*, której struktura posiada złożone klucze główne oraz tabele zawierające po kilkaset tysięcy wierszy danych. Baza *employees* dostępna jest na stronie producenta bazy danych MySQL. Napisana została również własna baza danych *accounts*, której celem jest obsługa autoryzacji opartej na tokenie JWT. Następnie scharakteryzowano każdy z proponowanych sposobów komunikowania się i wybrano trzy sposoby.

W kolejnym kroku scharakteryzowano platformę testową, przedstawiony został szablon aplikacji, która będzie udostępniona na serwerze Apache Tomcat w wersji 9 oraz opisano proces wdrażania bazy danych. Proces testowania został podzielony na etapy, a dla każdego z etapów opracowano klika scenariuszy testowych. Na potrzeby testowania aplikacji za pomocą testów jednostkowych w środowisku testowym opracowane zostały trzy scenariusze koncentrujące się na składniach języka SQL oraz realizujące złożone zapytania. W celu testowania aplikacji jako usługi REST API opracowano testowanie z wykorzystaniem dwóch narzędzi: Postman oraz Apache JMeter. Narzędzie Postman ma na celu przetestowania poprawności dostępu do zasobów oraz zmierzenia czasu realizacji w przypadku jednostkowego dostępu. Narzędzie Apache JMeter ma za zadanie sprawdzić jak aplikacja radzi sobie z wieloma żądaniami naraz.

Dla każdego z wybranych sposobów komunikacji opisano przebieg wdrożenia komunikacji, na który składają się:

- spis wykorzystanych bibliotek dla danego sposobu komunikacji z bazą danych,
- sposób konfiguracji komunikacji z bazą danych,
- opis problemów napotkanych podczas etapu wdrożenia danego sposobu,
- proces wdrożenia obsługi autoryzacji opartej na tokenie JWT,
- proces udostępnienia bazy *employees* pracującej pod usługą REST API,
- sposób realizacji scenariuszy testowych na warstwie dostępu do danych.

Po wykonaniu testów na warstwie dostępu do danych, aplikacje zostały wdrożone na serwerze Apache Tomcat, a następnie zrealizowano plany testowe dla aplikacji pod postacią usługi REST API. Dla każdego z etapów testowania zebrane zostały pomiary czasów realizacji zarówno dla pojedynczego testu jaki i całego scenariusza.

3. Analiza wyników

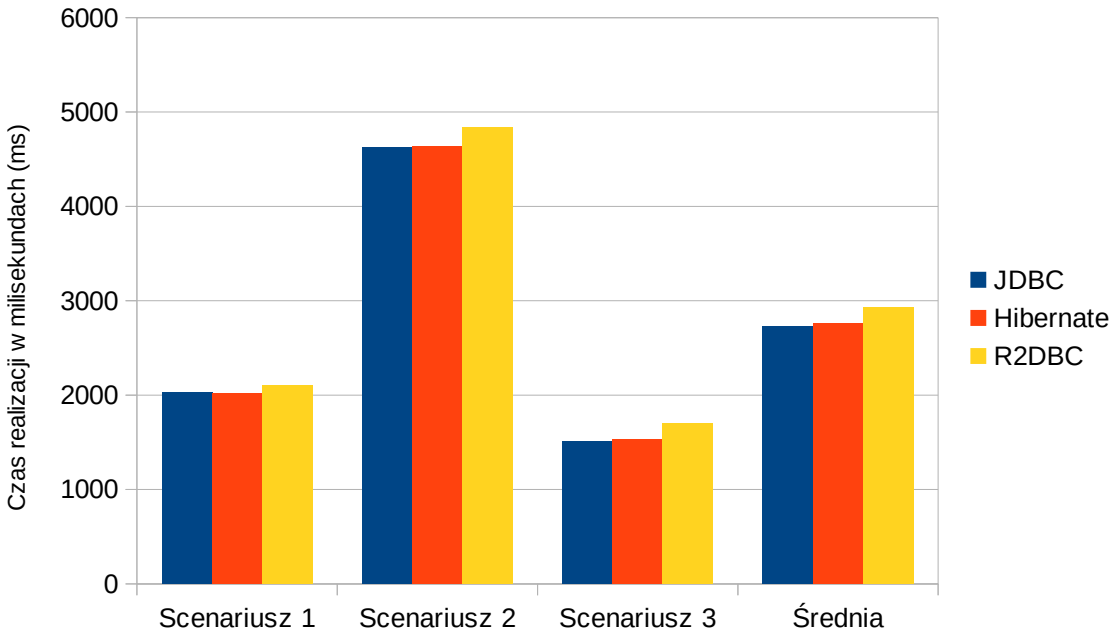
Etap testowania na warstwie dostępu do danych były podzielony na 3 scenariusze. Każdy scenariusz wykonany został 1000 krotnie. Z uwagi na pojawiające się cyklicznie odstające wartości, postanowiono odrzucić silnie odstające wartości. W tabeli 1 przedstawiono uzyskane średnie czasu realizacji całego scenariusza, liczbę wziętych pod uwagę testów oraz procent odrzuconych testów.

Tabela 1: Wyniki testów etapu pierwszego po odrzuceniu wartości odstających

	JDBC			Hibernate			R2DBC		
Nr Scenariusza	Średnia czasu (ms)	Liczba testów	Procent odrzuconych testów	Średnia czasu (ms)	Liczba testów	Procent odrzuconych testów	Średnia czasu (ms)	Liczba testów	Procent odrzuconych testów
1	2024,79	842	15,80%	2022,03	840	16,00%	2104,76	849	15,10%
2	4620,52	994	0,60%	4637,2	991	0,90%	4831,67	981	1,90%
3	1507,03	894	10,60%	1528,84	862	13,80%	1696,67	877	12,30%
Całość	**8193,65**	792	20,80%	8284,85	792	20,80%	**8778,55**	**808**	19,20%

Źródło: opracowanie własne

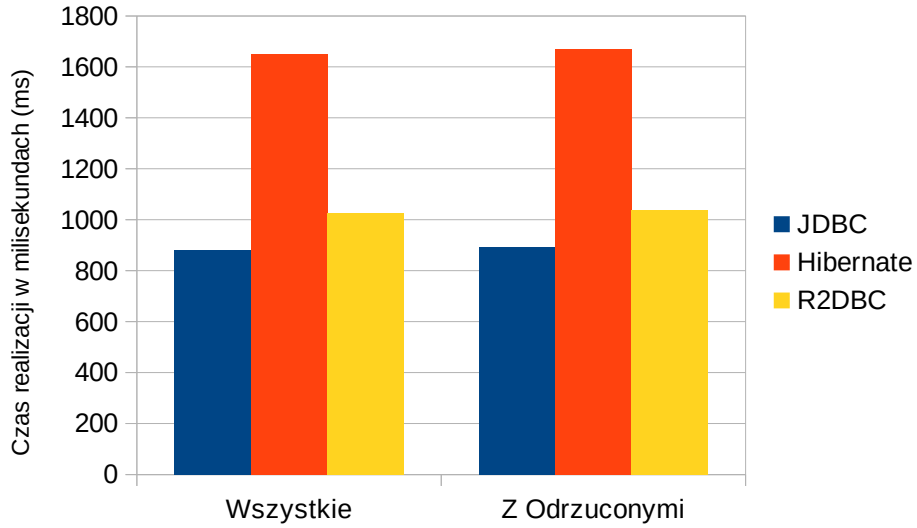
Analiza wyników zebranych podczas testowania na warstwie dostępu do danych pokazała, iż najszybciej realizowane były zapytania w sposobie opartym na sterowniku JDBC, a najwolniejszym okazało się rozwiązanie R2DBC (rys 1).



Rysunek 1: Etap pierwszy, wyniki po odrzuceniu odstających wartości

Źródło: opracowanie własne

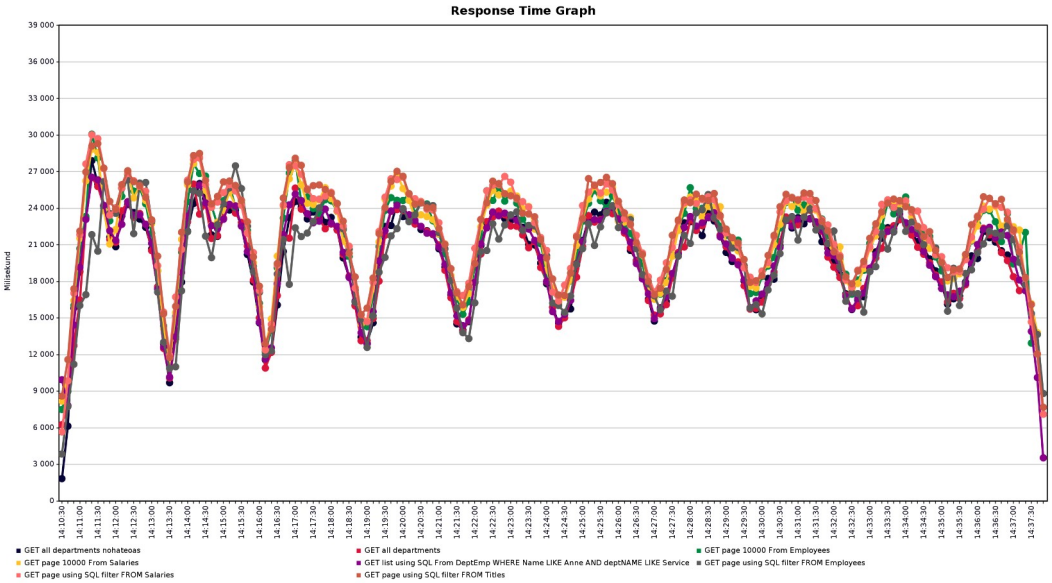
Testowanie narzędziem Postman nie wykazało żadnych nieprawidłowości. Tokeny JWT generowane były poprawnie, a zasoby były udostępniane zgodnie z przypisanymi rolami do użytkowników. Analiza wyników tego etapu wykazała, iż komunikacja JDBC posiada najniższą średnią czasu realizacji testu, a biblioteka Hibernate znacząco odstaje od rozwiązań komunikujących się bezpośrednio za pomocą sterowników (rys. 2).



Rysunek 2: Porównanie testów w aplikacji Postman, przed i po odrzuceniu wartości odstających

Źródło: opracowanie własne

W przypadku testowania za pomocą narzędzia Apache JMeter określono liczbę jednoczesnych zapytań HTTP na 1000. Komunikacja R2DBC nie potrafiła sobie poradzić z obsługą takiej liczby zapytań, a w rezultacie ponad 80% testów zakończyło się niepowodzeniem. Rozwiązanie Hibernate osiągnęło zaledwie 0,02% nieudanych testów względem 8000 pomiarów. W przypadku rozwiązania JDBC wszystkie testy zostały zrealizowane poprawnie, a czas realizacji całego etapu trwał około 27 minut (rys. 3), gdzie rozwiązanie Hibernate potrzebowało ponad 52 minut aby zrealizować cały etap. W przypadku komunikacji opartej o sterownik R2DBC, procent niepoprawnie wykonanych testów jest zbyt duży, aby wziąć jakiekolwiek czasy wykonania testów pod uwagę.



Rysunek 3: Wykres czasów odpowiedzi dla JDBC, z aplikacji Apache JMeter

Źródło: opracowanie własne

4. Podsumowanie wyników

Analiza porównawcza oraz interpretacja wyników pokazała, iż komunikacja za pomocą sterownika JDBC i natywnych zapytań była najszybszym rozwiązaniem, a także jako jedyna bezbłędnie zrealizowała wszystkie założenia testowe. Zaleca się JDBC jako podstawowy sposób komunikowania się z bazą danych. W przypadku komunikowania się z wykorzystaniem biblioteki Hibernate, procent błędów był marginalny i występował jedynie podczas testowania za pomocą narzędzia Apache JMeter. Biblioteka Hibernate umożliwia komunikowanie się na kilka sposobów lecz jest rozwiązaniem wolniejszym od sterownika JDBC. Komunikacja oparta o sterownik R2DBC nie nadaje się na główne rozwiązanie bazodanowe i analiza wykazała, iż w przypadku usługi REST API ma zastosowanie jedynie jeśli jest przeznaczona dla mniejszej liczby odbiorców.