11.12.2022

PZSP2 – Konsultacje na temat architektury

Zespół 7 MHMD – Mateusz Krakowski, Hubert Soroka, Maciej Tymoftyjewicz, Daniel Kobiałka

1. Temat

Tematem naszego projektu jest platforma edukacyjna, pozwalająca na dodawanie pytań połączonych z odpowiedzią i źródłem wiedzy (taki zestaw będzie dalej zwany fiszką), przeglądanie fiszek oraz grupowanie ich z użyciem grup i tagów. Użytkownicy muszą się zalogować, aby przeglądać i dodawać nowe fiszki i grupy, które mają ustawienia publiczności.

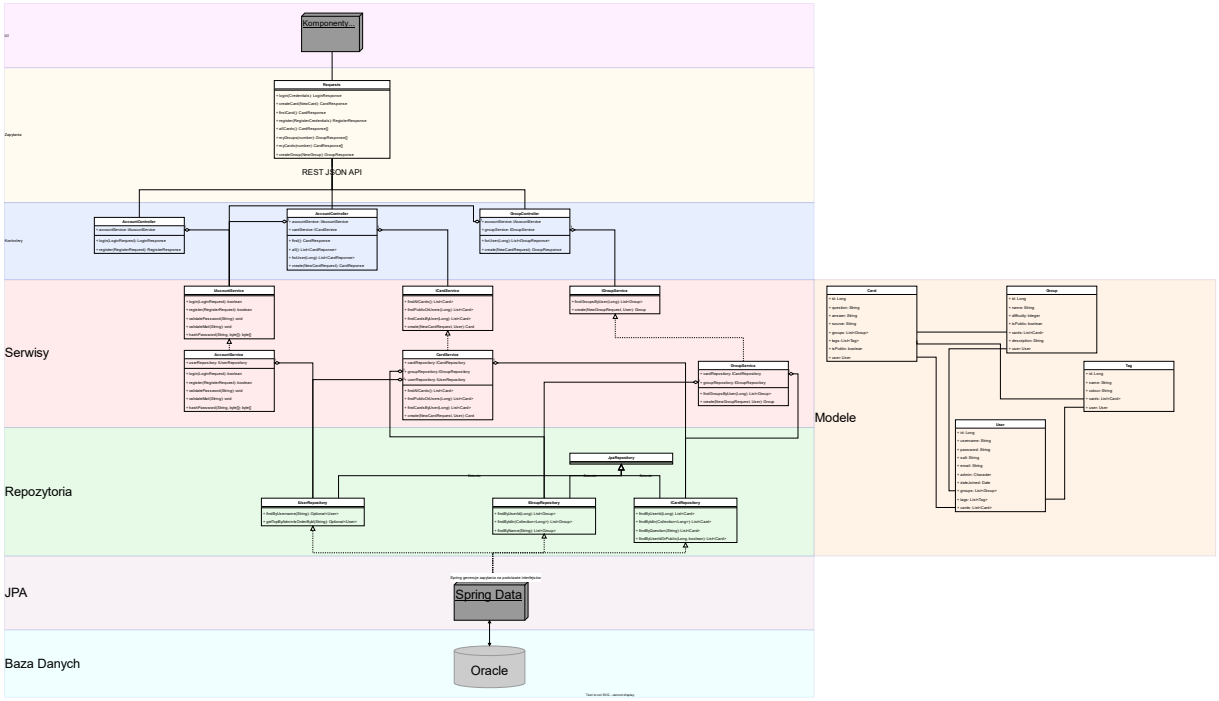
2. Ogólne założenia architektury

Projekt wykonywany z użyciem bazy danych Oracle (udostępnianej przez uczelnię), backendu w Javie z użyciem Spring oraz niezależnego frontendu w typescripcie z Reactem. Widać jasno trzy rozłączne części, każda z nich funkcjonująca niezależnie od pozostałych, każda uruchamiania oddzielnie.

Baza danych komunikuje się z backendem poprzez interfejs Spring Data JPA, który wykorzystuje driver JDBC. Backend wystawia Restowe API, przez które frontend nawiązuje połączenie. Dane przesyłane Jsonem, na ten moment protokół http, ale najpewniej nastąpi przejście na https, w celu zapewnienia większego bezpieczeństwa. W celach poglądowych, na moodle załączamy też diagram ERD bazy danych.

3. Szczegóły architektury backendu

Diagram przedstawiający architekturę backendu (w razie problemów, na moodlu także załączony plik svg w wersji ciemnej):



UI oraz ZAPYTANIA są realizowane po stronie aplikacji Reactowej. Nie stworzyliśmy diagramu dla komponentów frontendowych, ponieważ z punktu widzenia organizacji architektury, całość danych, które przetwarzają, będzie przechodzić przez klasę Requests z diagramu. Klasy wyżej niż ZAPYTANIA odpowiadają tylko prezentacji danych, a nie ich przetwarzaniu.

Klasy z grupy MODELE są zamieszczone na boku warstw SERWISÓW oraz REPOZYTORIÓW, ponieważ są często wykorzystywane przez obie z nich. Nie zamieszczono linii notacji UML przy każdym wykorzystaniu tych klas, ponieważ uczyniłoby to diagram nieczytelnym. Te klasy są bezpośrednio mapowane na tabele relacyjne w bazie danych, reprezentują wszystkie główne tabele, pozostałe służą tylko reprezentacji związków N-N pomiędzy nimi.

Nasz pomysł na architekturę najprościej można podsumować następująco:

I. Komponent frontendowy, potrzebując konkretnych danych, wywołuje jedną z funkcji z klasy statycznej Requests
II. Każda funkcja tej klasy odpowiada jednej akcji w jednym z kontrolerów
III. Kontrolery wywołują metodę z odpowiedniego serwisu, który jest polem kontrolera. Kontrolery zawierają zero lub minimum logiki „biznesowej" programu. Odsyłają dane w klasie z sufiksem Response, są to proste klasy bez metod, które zawierają tylko to, co będzie potrzebne na frontendzie w kontekście danej metody (np. zwracamy fiszkę bez całego powiązanego z nią obiektu User, ponieważ nie chcemy pokazywać uprawnień użytkownika przy wyświetlaniu jego fiszki, jest to nadmiarowe i niepotrzebne)
IV. Serwis wykonuje operacje na obiektach z modelu dostarczonych przez repozytoria tak, aby móc zwrócić wymagane dane lub rzucić wyjątek. (Kontrolery mają domyślny ExceptionHandler, który zwróci odpowiedź zawierającą treść wyjątku)
V. Repozytoria dziedziczą po JpaRepository, dostarczonym przez Spring. Dzięki czemu nie trzeba ich implementować, wystarczy stworzyć sam interfejs, nazywając metody zgodnie ze standardem Spring. Zapytania im odpowiadające wygenerują się automatycznie. W przypadku, gdy potrzeba wykonać trudniejsze zapytanie, wystarczy dodać adnotację @Query(…) i podać zapylanie SQLowe.
VI. Spring wykonuje zapytania i zwraca obiekty z modelu przedstawiające relacyjne dane z bazy danych.

Architekturę będziemy rozwijać i utrzymywać w oparciu o powyższe założenia, wydają się nam wartościową podstawą do dalszego rozwoju systemu.

4. Pytania i wątpliwości

W ramach konsultacji, chcielibyśmy też zapytać o system uwierzytelnienia w naszej aplikacji. Realizacja takiego systemu z użyciem Spring Security spędza nam sen z powiek (niestety dosłownie). Chcemy autoryzację zrealizować w ten sposób, że użytkownik po logowaniu otrzymywałby specjalny token (najpewniej JWT), który po stronie Reacta byłby zapisywany lokalnie. Przy każdym następnym requeście, token byłby doklejany do treści żądania, i na jego bazie byłaby potwierdzana tożsamość użytkownika. Idealnie byłoby, gdyby móc w kontrolerze lub serwisie zawołać statyczną metodę, która zwróci encję User (albo chociaż id lub nazwę użytkownika) z danymi użytkownika, na rzecz którego wykonujemy operację. Dodatkowo, chcielibyśmy, aby wszystkie endpointy poza logowaniem i rejestracją były niedostępne dla użytkowników bez tokena. Na ten moment dodanie dependency do Spring Security (tylko to, bez zmieniania ani linii kodu) powoduje niemożliwość wysłania ani jednego requestu do backendu. Problemem zawsze jest CORS i brak odpowiedniego nagłówka „Access-Control-Allow-Origin". W kontrolerach mamy adnotację @CrossOrigin, która blokuje requesty z adresu innego niż wpisany adres frontendu, ale po usunięciu tej adnotacji nic się nie zmienia. Bylibyśmy bardzo wdzięczni za wskazówki i porady.

5. Link do repozytorium

https://gitlab-stud.elka.pw.edu.pl/mkrakows/pzsp_poniedzialek_grupa_7