

Dokumentacja

Spis treści:

1) Opis Serwera

1.1 Klasy i interfejsy w serwerze

1.2 Opis działania i funkcje

2) Opis klienta

2.1 Serwisy

2.2 Komponenty

3) Możliwości rozwoju

4) Napotkane problemy

4.1 Serwer

4.2 Klient

5) Podsumowanie

1) Opis Serwera

1.1) Klasy i interfejsy w serwerze:

Serwer został w całości napisany w technologii Spring Boot

Klasy i interfejsy zawierają się w paczce `com.backend.springboot`

Interfejsy w tej paczce (repozytoria), dziedziczą po interfejsie `JpaRepository`

Służą do przechowywania obiektów klas z adnotacjami `@Entity`, służą do komunikacji z bazą danych: usuwają, dodają oraz odczytują obiekty.

Obiekty: *Admin*, *Auction*, *User*, są odpowiednikami tabel używanych w aplikacji.

Obiekt *AuctionAdder* – służy do dodawania aukcji, pobierając dane do logowania, w celu weryfikacji użytkownika.

Obiekt *DataController* – jest obiektem mapującym funkcje na określone kanały „localhost/api”, przechowuje repozytoria, zapewnia obejście CORS policy.

1.2) Opis działania i funkcje:

Połączenie z bazą danych: zdefiniowane jest w pliku `application.properties` w folderze „src/main/resources”: login, hasło, link i port na którym działa aplikacja

Obiekty odzwierciedlające tabele bazy danych posiadają gettersy, settersy, metodę `toString`, konstruktor domyślny, oraz przypisujący dane.

Funkcje zawarte w *DataController*:

Mapping klasy *DataController*: „api”

CORS Origin mapping - „<http://localhost:3000>”

public List<User> getUsers() - zwraca listę użytkowników, mapping „users”

public List<Admin> getAdmins() - zwraca listę administratorów, mapping „admins”

public List<Auction> getAuctions() - zwraca listę aukcji, mapping „auctions”

public List<Auction> getAuctionsOfUser(@PathVariable String owner_login) - zwraca listę aukcji danego użytkownika (nieużywane w kliencie), mapping „auctions/ofuser/{owner_login}”

public List<Auction> getAuctionsWithName(@PathVariable String name) - zwraca listę aukcji z daną nazwą (nieużywane w kliencie), mapping „auctions/withname/{name}”

public String addUser(@RequestBody User u) - dodaje użytkownika, mapping „addUser”

public int LoginAdmin(String AdLogin, String AdPass) - sprawdza logowanie administratora

public int Login(@RequestBody User u) - sprawdza dane logowania użytkownika mapping „login”

public String BuyProduct(@RequestBody Auction a) - kupuje produkt (usunięcie z DB) mapping „buyProduct”

public String Deleter(@RequestBody User u) - Kasuje użytkownika z bazy danych, oraz wszystkie jego aukcje, mapping „userRemove”

public String AddAuction(@RequestBody AuctionAdder Ad) - dodaje aukcję od użytkownika podającego dane logowania, mapping „addAuction”

Zmienne:

private UserRepository userRepo; - repozytorium użytkowników

private AdminRepository adminRepo; - repozytorium administratorów

private AuctionRepository auctionRepo; - repozytorium aukcji

2) Opis Klienta

Klient został napisany w technologii React.

Wykorzystuje biblioteki: *axios*, *react-routing-dom*, *react-bootstrap*, *history*

2.1 Serwisy

AuctionService i *UserService* służą do wypisywania tabel użytkowników i aukcji.

Wykorzystują bibliotekę *axios* i funkcję *get()* do wyciągnięcia rekordów z bazy

2.2 Komponenty:

AddAuction – dodaje aukcję, routing „/addAuction”

AddOrder – zamawia po ID, „/BuyProduct”

AddUser – rejestrowanie użytkownika w bazie, „/addUser”

AdminBase – strona administratora „/adminBase”

AuctionComponent - wypisanie aukcji „/auctions”

BaseComponent – strona użytkownika, „/base”

history – komponent *history*

Login – logowanie „/login”

LoginProtector – routing wymagający autoryzacji

RemoveUserAndAuctions – usuwanie użytkownika i jego aukcji po ID, „/removeUser”

UserComponent – wypisanie użytkowników, „/users”

3) Możliwości rozwoju

Mimo, że aplikacja ze strony serwera działa w sposób bardzo zadowalający, jest sporo miejsca na rozwój w kliencie.

Wynika to z braku wiedzy w technologii React

1) Autoryzacja / logowanie – dodanie mechaniki zapisu tokena autoryzacyjnego po stronie klienta (np. JWT), używając przygotowanego przeze mnie komponentu ProtectedRoute (w komponencie LoginProtector)

2) Wypisywanie z wyszukiwaniem – przygotowane przeze mnie 2 funkcje *getAuctionsWithName* i *getAuctionsOfUser* są nieużywane

3) Wykorzystanie tabeli „banned” - administrator mógłby tylko blokować od dodawania aukcji, zamiast usuwać

4) Napotkane Problemy

Podczas pisania serwera nie napotkałem problemów – technologia Spring Boot działa bardzo dobrze w kwestiach backendu

Podczas pisania klienta napotkałem się na kilka problemów:

- łamanie CORS policy, przy dodaniu przycisku zakupu, w tabeli aukcji, rozwiązanie tymczasowe – zakup ręczny, po dodaniu ID
- problemy z funkcją autoryzacji
- problemy z wyświetleniem danych aukcji względem nazwy/użytkownika – działa na serwerze, na kliencie nie.

5) Podsumowanie

Spring Boot jest bardzo łatwą technologią do pisania serwera – posiada dużo dodatków usprawniających przetwarzanie danych.

Technologia React również posiada bardzo wiele przydatnych bibliotek i rozwiązań, niestety jest ona znacznie bardziej skomplikowana.