Zadanie rekrutacyjne

1) Treść

Stwórz serwis, który będzie udostępniał punkt końcowy (endpoint) REST API poprzez metodę HTTP typu GET z parametrami wskazanymi w punkcie 3. Celem samego serwisu jest prezentacja oraz obsługa informacji na temat klienta bankowego, posiadanych przez niego kont bankowych (rodzajów) oraz listy przelewów wykonanych z danego konta. Konsumentem tej usługi będzie pracownik bankowy, który będzie przeprowadzał audyt operacji danych klientów. Dla uproszczenia, użytkownik końcowy powinien autentykować się przy pomocy BASIC w każdym żądaniu HTTP (stateless). Lista użytkowników może być predefiniowana w wybrany sposób, np. w pliku typu properties z użytkownikiem jako klucz i hasłem jako wartość.

Na wejściu serwis przyjmuje dwie wartości: numer identyfikacyjny klienta oraz identyfikator rodzaju konta. W odpowiedzi użytkownik dostaje posortowaną rosnąco po kwocie listę transakcji (data, kwota, identyfikator) i rodzajem rachunku bankowego (nazwa), z którego wykonywana była transakcja, oraz imię, nazwisko zlecającego.

2) Zestawy danych wejściowych

plik transactions.csv: plik zawierający transakcje z datą oraz ich kwoty, identyfikatory zlecających transakcje oraz rodzaj rachunku (identyfikator), którego transakcja dotyczy
plik accounttypes.csv: plik zawierający rodzaje rachunków wraz z ich identyfikatorami
plik customers.csv: plik zawierający informacje o zlecających transakcje

3) Parametry wyszukiwania

a) Rodzaj konta (account_type): identyfikator rodzaju konta (np. 1) lub lista identyfikatorów rodzaju konta oddzielonych przecinkiem (np. 1, 3). W przypadku użycia liczby mnogiej identyfikatorów rodzajów kont, powinno to zostać traktowane jako wyrażenie logiczne „lub”, tj. „1” lub „3”. W przypadku braku podania parametru rodzaju konta lub wypełnienie go wartością „ALL” powinno przyjąć wszystkie rodzaje kont (1, 2, 3, 4).

b) Numer identyfikacyjny zlecającego tranzakcję-klienta(id): wyszukiwanie użytkownika po numerze identyfikacyjnym (np. 5) z możliwością podania kilku numerów identyfikacyjnych oddzielonych przecinkiem (np. 1, 5). Analogicznie do rodzaju identyfikatora rodzaju konta, wprowadzenie kilku identyfikatorów klienta powinno być traktowane jako wyrażenie logiczne „lub”. Brak podania wartości parametru identyfikatora klienta lub wypełnienie go wartością „ALL” powinno przyjąć wszystkie dostępne identyfikatory (1, 2, 3, 4, 5).

4) Przykładowe żądanie-odpowiedź

Uwaga! Przedstawione poniżej żądania i odpowiedzi nie pochodzą z danych dołączonych do zadania. Prezentują jedynie rezultaty logiki samego wyszukiwania.

Przykład:
Parametry żądania: account_type = 2, customer_id = 3
Odpowiedź: 4 transakcje →

1. (Data transakcji: 2020-06-04 12:31:00, Identyfikator transakcji: 135, Kwota transakcji: 1500,00 rodzaj rachunku: saving_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski
2. (Data transakcji: 2019-03-01 11:22:05, Identyfikator transakcji: 121, Kwota transakcji: 99,00 rodzaj rachunku: saving_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski
3. (Data transakcji: 2011-03-02 01:32:10, Identyfikator transakcji: 9, Kwota transakcji: 100,00 rodzaj rachunku: saving_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski
4. (Data transakcji: 2017-01-01 12:50:12, Identyfikator transakcji: 88, Kwota transakcji: 123,45 rodzaj rachunku: saving_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski

Przykład 2:
Parametry żądania: account_type = 4. customer_id = ALL
Odpowiedź: 5 transakcji →
1. (Data transakcji: 2020-06-04 12:31:00, Identyfikator transakcji: 11, Kwota transakcji: 27,31 rodzaj rachunku: trading_account, Imię zlecającego: Damian, Nazwisko Zlecającego: Damianowski
2. (Data transakcji: 2019-03-01 11:22:05, Identyfikator transakcji: 5522, Kwota transakcji: 31,00 rodzaj rachunku: trading_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski
3. (Data transakcji: 2011-03-02 01:32:10, Identyfikator transakcji: 9, Kwota transakcji: 121,55 rodzaj rachunku: trading_account, Imię zlecającego: Marek, Nazwisko Zlecającego: Marecki
4. (Data transakcji: 2017-01-01 12:50:12, Identyfikator transakcji: 3131, Kwota transakcji: 123,45 rodzaj rachunku: trading_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski
5. (Data transakcji: 2014-01-01 13:33:55, Identyfikator transakcji: 27721, Kwota transakcji: 999,21 rodzaj rachunku: trading_account, Imię zlecającego: Adam, Nazwisko Zlecającego: Adamowski

5) Wymagania techniczne:
Serwis powinien być aplikacją webową napisaną w Kotlinie wraz z użyciem Spring Framework (najchętniej aplikacja Spring Boot w wersji 2.x). Kod może być również napisany częściowo w Java (wersja 8 i wyższe), aczkolwiek zachowując proporcję napisanych co najmniej połowy klas w języku Kotlin.
Dane, które dostarczone są w postaci plików, powinny zostać załadowane do bazy danych MongoDB przy starcie aplikacji. Każdorazowe włączenie aplikacji powinno usuwać dane i kolekcje z bazy danych oraz ładować je na nowo do bazy danych. Kolekcje mogą być podzielone ze względu na rodzaj danych wraz z relacją między kolekcjami w każdym dokumencie lub może być to również jeden rodzaj kolekcji zawierający wszystkie dane w postaci dokumentów. Baza danych może być zarówno umieszczona w kontenerze Docker lub jako standardowa baza „standalone”. Obsługę mapowania obiektowo-relacyjnego

zaimplementuj w wybrany przez siebie sposób (możesz użyć zewnętrznych bibliotek np. Spring Data JPA). Przy projektowaniu obsługi zapytań do bazy danych weź pod uwagę wydajność odczytu. Wydajność zapisu do bazy nie jest najważniejsza, gdyż docelowo dane są ładowane z zewnętrznego źródła.
Preferowane narzędzia do budowania to: Gradle 5+ lub Maven.
Instrukcja włączania i budowania aplikacji powinna być umieszczona w projekcie.
Możesz również dołączyć obraz dockerowy aplikacji.
Punkt końcowy (endpoint), którego zadanie dotyczy powinien posiadać co najmniej trzy testy jednostkowe (z podanym customer_id oraz account_type, podanym jedynie customer_id oraz pustymi wartościami/wypełnionymi przez wartość ALL) napisane w języku Java (wersja 8 i wyższe). Narzędzie do włączania testów jednostkowych jest dowolone, jednakże proszę o zawarcie instrukcji, jak testy te należy włączyć.
Użycie dobrych praktyk typu „clean code” oraz wzorców projektowych tam, gdzie to ma szczególny sens będzie dodatkowym atutem.

Zadanie nie powinno zająć Ci dłużej niż 4 godziny.

Powodzenia!