Zadanie ewaluacyjne na młodszego programistę JAVA

Spis treści

Wprowadzenie
Celem zadania jest stworzenie aplikacji w architekturze rozproszonej. Aplikacja powinna składać się z następujących komponentów:

- Komponent bazodanowy zawierający tabele do składowania danych.
- Komponenty z usługami napisanymi w SpringBoot

Opis aplikacji
Aplikacja służy do składowania oraz pobierania informacji o kredycie klienckim.

Przypadki użycia:

UC01: Złożenie informacji o kredycie

UC02: Pobranie informacji o kredycie

UC01: Złożenie informacji o kredycie
Dla tego przypadku użycia aplikacja wystawia usługę REST o nazwie: CreateCredit.

Na wejściu usługa przyjmuje następujące informacje:

Klient:

- Imię
- Nazwisko
- Pesel

Produkt:

- Nazwa produktu
- Wartość produktu

Kredyt:

- Nazwa kredytu

Na wyjściu usługa zwraca informację o numerze złożonego kredytu.

UC02: Pobranie informacji o kredycie
Dla tego przypadku użycia aplikacja wystawia usługę REST o nazwie: GetCredits.

Na wejściu usługa nie przyjmuje żadnych informacji.

Na wejściu usługa zwraca listę wszystkich złożonych kredytów, gdzie pojedynczy element listy zawiera następujące informacje:

Klient:

- Imię
- Nazwisko
- Pesel

Produkt:

- Nazwa produktu
- Wartość produktu

Kredyt:

- Nazwa kredytu

Architektura rozwiązania

Aplikacja powinna składać się z czterech komponentów:

- Bazy danych (trzy różne schematy)
- Komponent przechowujący kredyty
- Komponent przechowujący klientów
- Komponent przechowujący produkty

Poniższy diagram przedstawia architekturę rozwiązania. Komponenty powinny być zrealizowane z wykorzystaniem technologii SpringBoot.

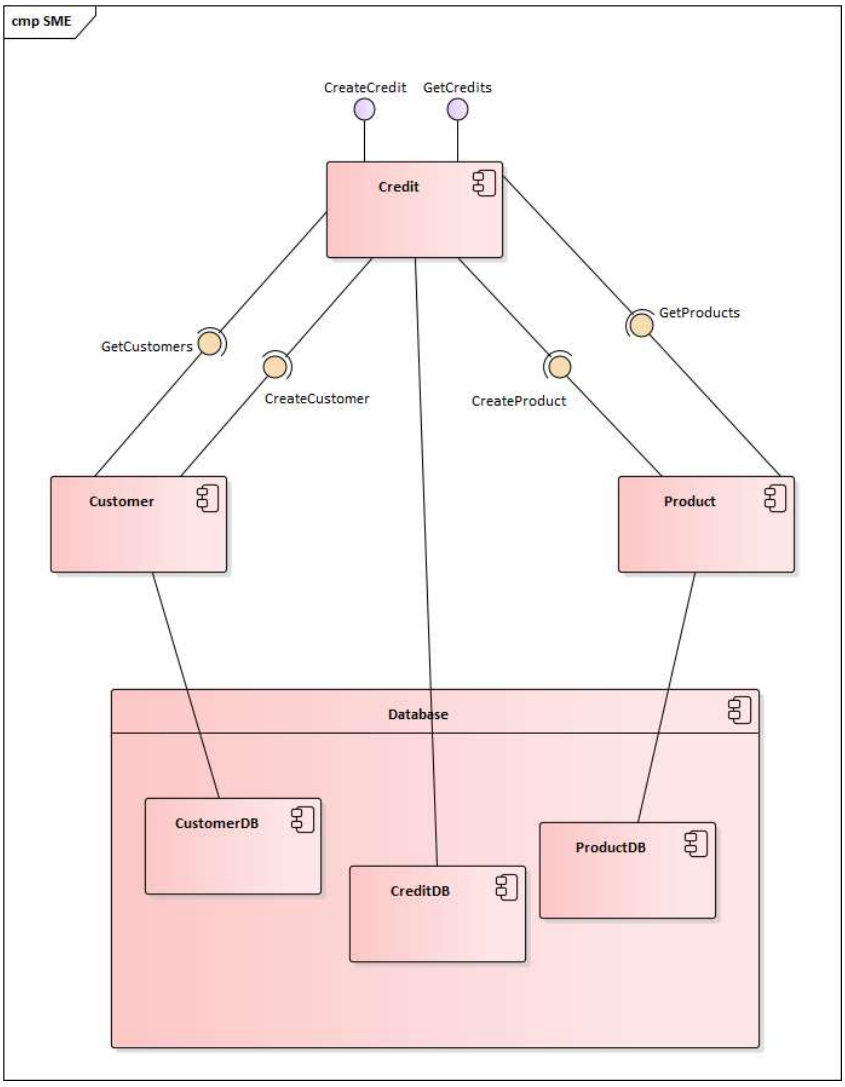
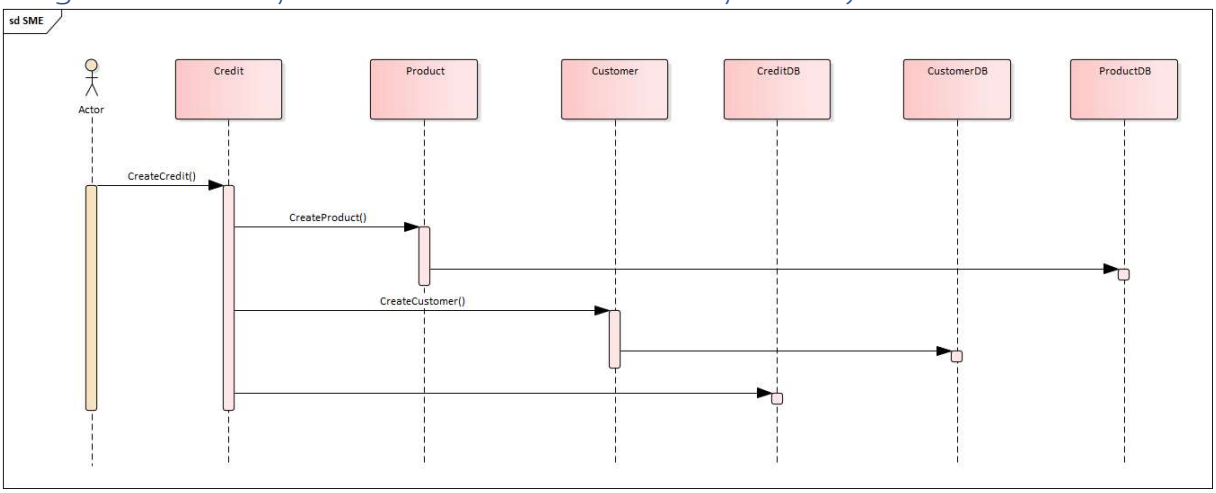
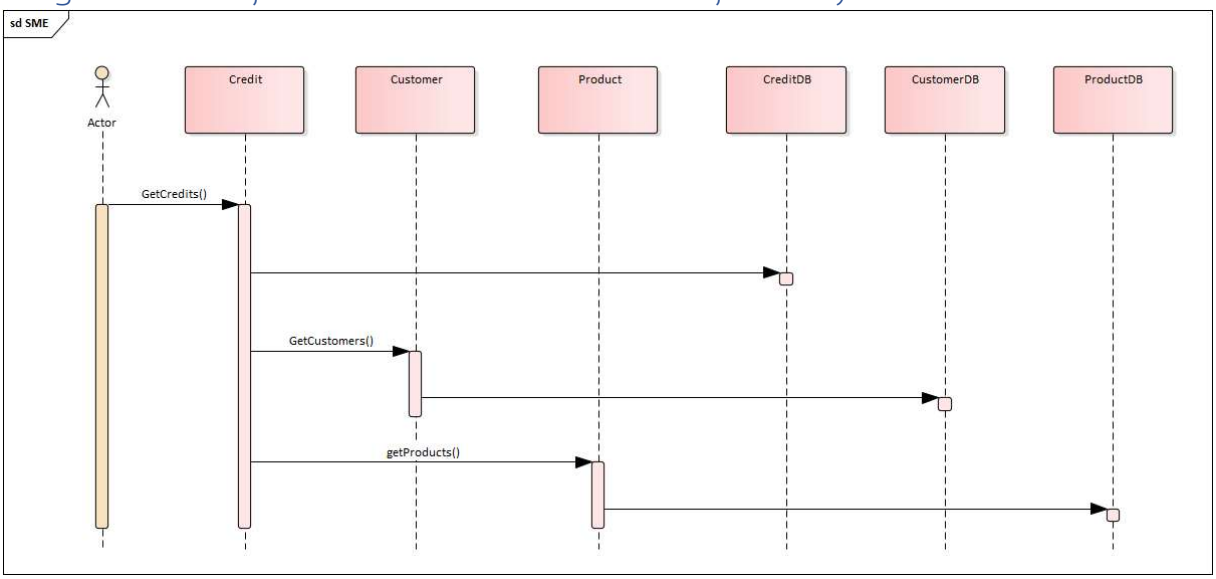


Diagram sekwencji dla UC01: Złożenie informacji o kredycie



1. Klient wywołuje usługę CreateCredit w celu złożenia informacji o nowym kredycie
2. Komponent Credit nadaje nowy numer kredytu
3. Komponent Credit wywołuje usługę CreateProduct komponentu Product i przekazuje informacje o produkcie oraz numer kredytu
4. Komponent Product składuje informacje o nowym produkcie w bazie danych
5. Komponent Credit wywołuje usługę CreateCustomer komponentu Customer i przekazuje informacje o kliencie oraz numer kredytu
6. Komponent Customer składuje informacje o kliencie w bazie danych
7. Komponent Credit składuje informacje o kredycie w bazie danych
8. Komponent Credit zwraca informacje o nadanym numerze kredytu klientowi.

Diagram sekencji dla UC02: Pobranie informacji o kredycie



1. Klient wywołuje usługę GetCredits
2. Komponent Credit pobiera informacje o wszystkich kredytach z bazy danych
3. Komponent Credit wywołuje usługę GetCustomers komponentu Customer i przekazuje numery kredytów, dla których należy zwrócić klientów
4. Komponent Customer pobiera informacje o klientach z bazy danych i zwraca
5. Komponent Credit wywołuje usługę GetProducts komponentu Product I przekazuje numery kredytów, dla których należy zwrócić produkty.
6. Komponent Product pobiera informacje o produktach z bazy danych i zwraca

7. Komponent Credit agreguje dane zwrócone z usług GetCustomers oraz GetProducts i zwraca listę kredytów Klientowi.

Uwagi do implementacji

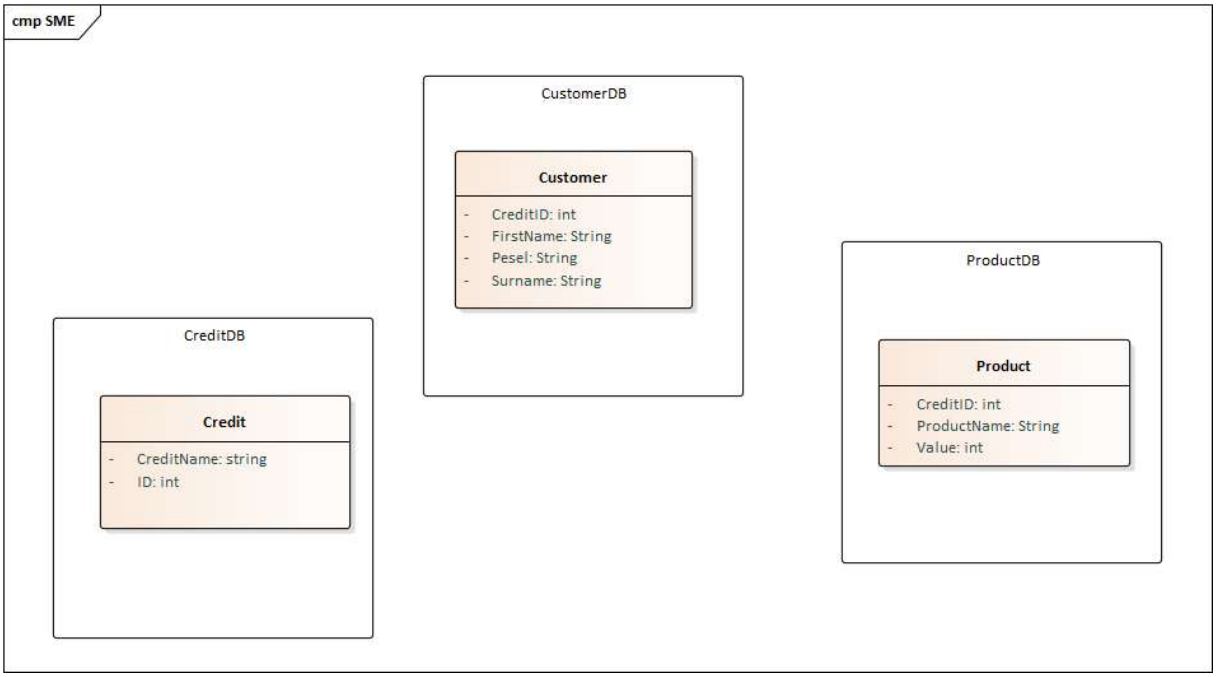
- Jest jedna baza danych, która zawiera trzy oddzielne schematy

Model danych oraz kontrakty usług

Poniżej opisano wymagania dotyczące modelu bazy

Model bazy danych

Poniżej przedstawiono poglądowy model bazy danych dla każdego z komponentów. Proponowany model można dowolnie rozszerzać. Zakładamy realizację w relacyjnej bazie danych. Komunikacja między SpringBoot a bazą danych odbywa się poprzez JDBC.



Kontrakty usług

Kontrakty usług realizowane przez SpringBoot należy dowolnie zaprojektować.

Dostawa rozwiązania i wymagania technologiczne

1. W projekcie wykorzystujemy narzędzie do budowania Apache Maven.
2. Stworzony kod należy złożyć w GitHub
3. Aplikacja powinna być rozmieszczona jako kontenery Docker. Realizacja w postaci czterech kontenerów dla poszczególnych aplikacji:
    a. SpringBoot Credit
    b. SpringBoot Customer
    c. SpringBoot Product
    d. Baza Danych

Kontenery nie mogą wymagać do działania użytkownika ROOT. Dokumentację do uruchomienia kontenerów należy umieścić na GitHub. Uruchomienie skryptu budującego Apache Maven powinno stworzyć wszystkie wymagane kontenery dockerowe.

4. Implementacja w Java 8 lub wyższej.
5. Wykorzystanie Kubernetes będzie dodatkowo punktowane.
6. Przykładowe wywołania usług należy opisać na GitHub.

Pozostałe kwestie implementacyjne

- Kwestie nieopisane/sprzeczne/niespójne/błędnie opisane/brakujące w dokumencie implementujący rozstrzyga według własnych upodobań.

Oceniane elementy

1. Kompletność rozwiązania
2. Jakość kodu: testy, komentarze, logiczny podział na pliki i elementy aplikacji.