JAVA SPRING MICROSERVICES

3. forduló

A kategória támogatója: Vodafone Magyarország Zrt.

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

Tud még valakit, aki összekapcsol 8 hálózatot és kitísztít kétmillió kódot ennyiért? - Dennis Nedry

A csapatotok halad előre, de nem elég gyorsan, Mr. Hammond kezd aggódni, mert bár nem spórol semmit, de kezdi megkérdejelezni a csapat kézbesítőképességét. Sürgeti a park biztonsági rendszerének a bekötését, az eddigi tapasztalatok alapján mindig azzal volt a legtöbb baj.

Mire lesz szükséged a fordulóban?

- Internet kapcsolat
- Java 11
- Docker és/vagy docker-compose
- Maven telepítve (vagy Maven Wrapper használata)
- Fejlesztői környezet (ajánlott)

Ha egy adott feladat egy annotációt vár válaszként, akkor oda csak az annotáció nevét add meg @ előjellel, nem szükséges teljes minősített nevet megadni, például: @NotNull.

Felhasznált idő: 00:00/40:00 **Elért pontszám:** 0/11

1. feladat 0/2 pont

A következő felsorolásból mely technológiák használhatóak szolgáltatások közötti kommunikáció létesítésére?

Válaszok

- [ ] Jetty
- [x] gRPC
  Ez a válasz helyes, de nem jelölted meg.
- [ ] OpenAPI
- [x] GraphQL
  Ez a válasz helyes, de nem jelölted meg.
- [ ] AWS

Magyarázat

A felsorolásból a gRPC és a GraphQL tekinthető szolgáltatások közötti kommunikációt megvalósító technológiáknak. A gRPC a Google által fejleszett Protocoll Buffert használó eszköze, míg a GraphQL a Facebook által fejlesztett specifikáció, amely képes több protokollon keresztül megvalósítani a szolgáltatások közötti kommunikációt.

A Jetty egy Javas alkalmazás szerver, OpenAPI egy API dokumentáció specifikáció és az AWS pedig egy felhőszolgáltatás.

2. feladat 0/0 pont

Milyen előnyei vannak a CQRS minta alkalmazásának?

Válasz

- ( ) Segítségével a legtöbb inkonzisztenciát okozó probléma áthidalható
- ( ) NoSQL adatbázisra épül, így séma nélkül tudunk adatokat befogadni és tárolni használatakor
- (•) Hatékony lekérdezéseket definiálhatunk vele a szolgáltatások között
  Ez a válasz helyes, de nem jelölted meg.
- ( ) Az architektúránkat egyszerűsíthetjük vele

Magyarázat

Kedves Versenyzők!

A kérdést 0 pontosra állítottuk, mivel:

Nincs kontextusba helyezve, ezzel a megadott válasz sem fogadható el teljesen. A kérdés nem említi meg, hogy a többi szolgáltatás is hasonló mintával épült volna, így a minta magában nem ad elegendő információt arra, hogy a lekérdezések hatékonyabbak lennének a szolgáltatások között. Ha külön említésre került volna több szolgáltatás és köztük lévő interakciók számossága akkor lenne igaz. A minta önmagában nem oldja meg ezt a gyakorlati problémát.

Köszönjük megértésedet!

(2022.11.09.)

A **CQRS** minta (**Command Query Responsibility Segregation**) segítséget nyújt több szolgáltatáson átívelő lekérdezések implementálására. A mintánál szükségünk van egy tárra, amely több szolgáltatásból származó adatot képes tárolni, amelyekre pedig egy szolgáltatáson belül kell megírnunk a megfelelő lekérdezéseket és kiadnunk azokat. Az architektúránk és a szolgáltatásaink lehetnek komplexebbek lesznek ezáltal, mert szükséges a folyamatos adatszinkronizáció a kezelt adathalmazok között például **event sourcing** segítségével. Bármilyen típusú adatbázist használhatunk, nincs megkötve, hogy csak NoSQL-t, sőt igazából az sem, hogy adatbázisnak kell lennie, egyszerűn egy perzisztens egységre van szükségünk.

Cikkek, leírások a mintáról:

- https://microservices.io/patterns/data/cqrs.html

- https://www.eventstore.com/cqrs-pattern
- https://event-driven.io/en/cqrs_facts_and_myths_explained/

3. feladat 0/0 pont

Melyik mintát alkalmazhatjuk a következő felsorolásból, ha nem szeretnénk a CQRS mintát használni, de szeretnék hasonló működést elérni?

Válasz

- ○ SAGA
- ◉ API kompozíció
  Ez a válasz helyes, de nem jelölted meg.
- ○ Elosztott követés
- ○ API átjáró (API gateway)

Magyarázat

Kedves Versenyzők!

A kérdést 0 pontosra állítottuk, mivel:

A kérdés feltételezi, hogy az előző (2.) kérdésben a szolgáltatások egymás között sokat interaktálnak és sok adatot adatot kérdeznek le, így ha a második kérdés sem volt kontextusban, emiatt a 3. sem értelmezhető mert feltételezte, hogy a 2. kérdés válaszából oldható meg/következtethető rá.

Köszönjük megértésedet!

(2022.11.09.)

A **CQRS** minta különböző szolgáltatásokhoz tartozó adatokat gyűjt össze és aggregál, hogy könnyebb legyen ezeket összekötnünk és lekérdeznünk egy szolgáltatástól akár. A felsorolt minták közül hasonlót az **API kompozíció** próbál megvalósítani, viszont itt maga a szolgáltatás több szolgáltatást hív meg és úgymond a memóriában próbálja meg az adatokat aggregálni, amivel ha mondjuk bizonyos szolgáltatások interfészei nem biztosítanak operációkat a megfelelő mennyiségű adathalmaz lekérdezésére akkor ezáltal az **API kompozíciót** megvalósító szolgáltatás nem tud hatékonyan adatokat aggregálni.

A többi minta,az API átjáró, az elosztott követés nem tartozik az említett témakörbe, a Saga minta pedig szolgáltatások közötti tranzakciókat vezérlésére/lebonyolítására ad megoldásokat.

Cikkek, leírások:

- https://microservices.io/patterns/data/api-composition.html

4. feladat 0/3 pont

Nevezd meg azt a mintát (patternt), amelyiket a hívott felek (szolgáltatások) implementálhatják és a védelmüket szolgálhatják?

Válaszok

- ☐ **Retry** minta
- ☑ **Bulkhead** minta
  Ez a válasz helyes, de nem jelölted meg.
- ☐ **Circuit Breaker** minta
- ☐ **Timeout** minta
- ☑ **Rate Limiter** minta

Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A Bulkhead mintát alkalmazó fél, külső szolgáltatás törekszik arra, hogy egy benne kevésbé stabilban működő kódrészlet túlterhelése ne okozhassa az egész szolgáltatást "vesztét", bizonyos erőforrásokat korlátoz, hogy csak megadott számú fél férhessen hozzá, ezzel növelve a robosztusságát/stabilitását a további kódrészeknek.

A Rate Limiter minta használatával a szolgáltatás limitálni tudja a hozzá beérkező kérések számát egy bizonyos időintervallumon belül. Segítségével különböző nagyobb klienseknek biztosíthatunk több hívási lehetőséget, még kisebb kliensek számára ezzel csökkenthetjük az elérhetőségünket.

5. feladat 0/1 pont

Melyik HTTP fejlécet kell küldenünk minden kérésben, hogy működjenek a hívások?

A park külső áramforrását biztosító rendszere elvileg "biztonságos", Mr. Hammond nem szeretné ha újra probléma történne vele, de ezt egy külsős cég fejlesztette le, és náluk bizony sok spórolás volt a fejlesztések során. API dokumentációt nem igazán csatoltak az alkalmazásukhoz, csak kettő darab végpont érhető el. Ahhoz, hogy a külső áramforrásokat tudjuk vezérelni elvileg előbb meg kell tudnunk a szolgáltatáshoz egy kulcsot, amelyet az egyik végpont ad vissza!

WOW - Mr. Hammond, hatalmas a baj, ez semennyire sem biztonságos!

Miután megvan a kulcs csak azzal tudjuk vezérelni az áramforrást.

Készíts egy alkalmazást amely a spring-cloud-starter-openfeign függőséget használja és próbáld meghívni a következő szolgáltatást. (Gyors projekt összeállítás):

- Szolgáltatás elérhetősége: **https://power-supply.api.ingensol.co**
- Indítsd el lokálisan a szolgáltatást a következő Docker képfájllal: **ghcr.io/ingen-sol/ingen-external-power-supply-service -** a 8080-as porton keresztül érhető el. **- docker run -p 8080:8080 ghcr.io/ingen-sol/ingen-external-power-supply-service**

Kettő darab végpont áll rendelkezésre:

- /secret - Egy GET-es végpont amelyen keresztül a titkos kulcsot lehet lekérni, viszont biztosítani kell egy HTTP fejlécet ami az első hívás során ki is derül, értéke a következő legyen: **InGen-Sol** - A válaszban a titkos kulcs fog szerepelni ami a következő hívásban lesz kötelező.
- /power-supply - Egy PUT-os végpont amelyben
- két mezőt kell beküldeni (JSON vagy XML formátumban):
- territory
- areaId
- plusz a titkos értéket tartalmazó HTTP fejlécet: **x-secret**

Az 6.-9. kérdések megválaszolásához is a fentieket használd!

Válasz

A helyes válasz:

x-external-client

Magyarázat

A külső szolgáltatás csak akkor hívható meg, ha azt egy Java kliensből tesszük meg, konzolból illetve másik nyelveken is megoldható, viszont kell egy HTTP fejléc ami jelzi a meghívott szolgáltatás felé, hogy ténylegesen jó "kliensből" hívjuk meg. Ha a

spring-cloud-starter-openfeign függőséget használjuk, akkor ez a fejléc adott lesz, és a többi kiegészítő fejlécen kívül a hívások működni fognak.

Az első hívásból kiderül, hogy melyik az a fejléc amelyet minden kérésben küldenünk kell fixen, ez pedig az **x-external-client**.

6. feladat 0/2 pont

Mi a titkos kulcs - a **secret** nevű mezőben tér vissza a válaszban?

Válasz

A helyes válasz:

X-ISLA-NUBLAR-1993

Magyarázat

A következő hívásnál már láthatjuk is a titkos kulcsot, amely a másik végpont meghívásához szükséges, a titkos kulcs értéke pedig **X-ISLA-NUBLAR-1993**

7. feladat 0/1 pont

A PUT /power-supply kérésben milyen **territory** értéket kell megadnunk, hogy a hívás sikeres legyen (HTTP 200 OK) ?

Válasz

- COSTA_RICA
- SORNA
- ARTIC
- NUBLAR
  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

8. feladat 0/1 pont

A PUT /power-supply kérésben milyen **areaId** értéket kell megadnunk, hogy a hívás sikeres legyen (HTTP 200 OK)?

Válasz

- THEROPOD-12
- THEROPOD-03
  Ez a válasz helyes, de nem jelölted meg.

THEROPOD-11

THEROPOD-04

Magyarázat

9. feladat 0/1 pont

A sikeres PUT /power-supply hívás után milyen érték található a válasz **message** mezőjében?

Válasz

A helyes válasz:

POWER-SUPPLY-ACTIVE

Magyarázat

A többi híváshoz a kombinációkat kell végigpróbálni, így jön ki, hogy a két keresett érték a következők: **NUBLAR** és **THEROPOD-03** majd a válaszban az érték: **POWER-SUPPLY-ACTIVE**

© 2023 Human Priority Kft.