

Az alkalmazásokat a `server.port` és a `mongodb.port` java paraméterek megadásával indítottam. Mindkettő saját `embedded mongodb`-vel rendelkezik, amelyekbe indításkor a működéshez szükséges szótárak beszállásra kerülnek a „dictionary” dokumentumba.

Kialakítottam egy `common` modult, amelybe az esetlegesen máshol is használható komponenseket szántam, mint a `dtok`, a validációk, áthíváshoz használatos `proxyk`, és a szótárakat használó szolgáltatások.

Az összetett validációkat annotációkkal szándékoztam megvalósítani, az esetleges hibaüzeneteket kigyűjtöttem a `messages.properties` fájlba. A hibaüzenet listát a `DTO`-k szerves részének gondoltam ezért azokban adom vissza. Az adott hibaüzenet tartalmazza, hogy melyik mezőre vonatkozik.

Az okmány feldolgozáshoz átadom az egész személy objektumot, ha abból a jövőben esetleg szükség lenne további adatra az okmányok feldolgozásához.

A `microservice` architektúrát `spring cloud` komponensekkel valósítottam meg. `Feign` által a `spring.application.name` alapján is beazonosíthatóak lettek a komponensek, amelyeket beregisztráltam egy `eureka` névszolgáltatásba, így a `document-service` áthívás megvalósulhat `ribbon` `load balancer` használatával választván egyet az éppen futó `document-service` példányok közül. A szolgáltatások elérése központosított, `zuul` gatewayen keresztül történik meg. A nyomon követhetőség megkönnyítése érdekében minden kérés kap egy egyedi azonosítót `sleuth` segítségével.

`Swagger`-t használtam dokumentáció generálásához.

Teszteseteknél felvettem egy ok példát, amelytől azt várnám el, hogy hiba nélkül lefusson, egy min példát, amely nem futat hiba nélkül, és minden mezőre legalább egy példát, amelyekben várok az adott mezőt érintő hibaüzenetet.

Fordításhoz `apache-maven-3.6.3`-t használtam, kódminőséget `sonarqube-7.8` segítségével ellenőriztem.