Documentació Microservei de dades

En el present document s'explicarà el funcionament i concepte del microservei. Aquest microservei està implementat en una estructura de tres capes: Controller, Service i Repository.

Controller: Encarregat de rebre les peticions REST i redirigir-les al servei
Servei: Encarregat de gestionar la lògica de la aplicació rebent la crida dels controller processant-la i cridant al Repository/s corresponent/s.
Repository: Els diferents repositoris es comuniquen amb la BDD. En aquest cas MongoDB

Estructura de dades

La següent representació s'empra tant com per model de dades a nivel de MongoDB com per DTOs en el controlador. Això és possible ja que mongo guarda la informació en JSON, llavors el mapeig típic de Entity -> DTO no és necessari.

Cal entendre el concepte darrera la estructura de dades. Per fer-ho s'han creat dos col·leccions a MongoDB, anomenades: expedient i dada. Totes dues dins una Database anomenada hel_dada.

Entendrem el conjunt de les dos col·leccions com les dades d'un expedient. Però cal deixar clar que **la col·lecció expedient només representa les dades de capçalera** d'aquest. Tot i això s'ha anomenat Expedient (tant la col·lecció com l'objecte) perquè son les mínimes dades a guardar per un expedient (la seva capçalera). Així es considera que si no existeix un expedientId en un document a la col·lecció expedient de MongoDB, no existeixen documents a la col·lecció dada. En altres paraules no existeix l'expedient amb expedientId.
Ambdues col·leccions conten una _id interna (usada per MongoDB) i un expedientId (per referenciar l'expedient dins Helium i aprofitar les id's ja existents).
S'ha volgut diferenciar la _id interna de mongo de la de l'expedient ja que les ids d'expedient ja estan al Helium 3.2.

Existeix una exentenció de MongoRepository per cada una de les col·leccions a nivell de Java

Estructura:

- **Expedient:**
    - expedientId: Long
    - entornId: Long
    - tipusId: Long
    - numero: String
    - titol: String
    - procesPrincipalId: Long
    - estat: String
    - dataInici: Date
    - dataFi: Date

- **Dada:**
    - expedientId: Long
    - procesId: Long
    - codi: String
    - tipus: Long / String / Date / Float / Termini / Preu / Integer / Boolean / Registre
    - multiple: boolean = false
    - valor: [ValorSimple|ValorRegistre]

- **ValorSimple:**
    - valor : String
    - valorText: String

- **ValorRegistre:**
    - camps: [Dada]

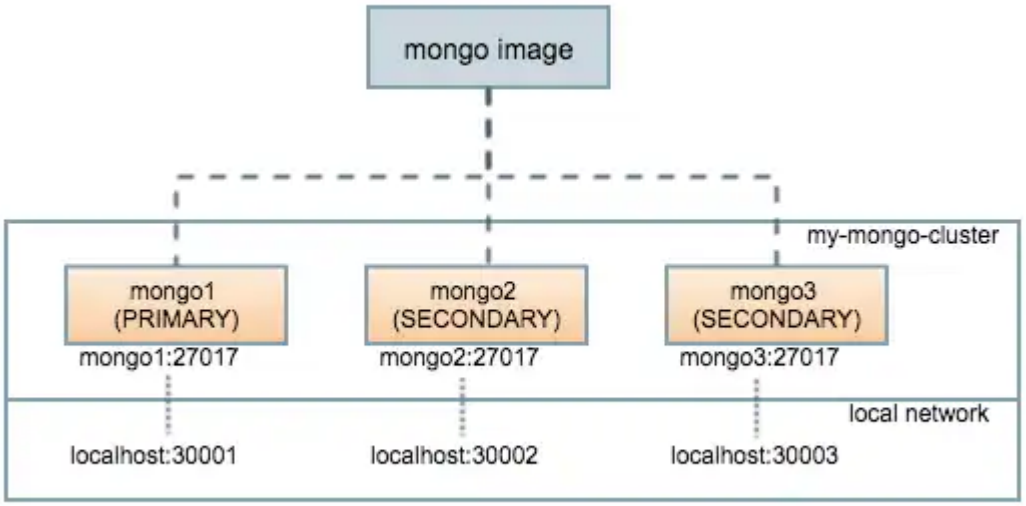
Transaccions MongoDB

A MongoDB les transaccions són atòmiques a nivell de document. Això implica que per fer operacions que modifiquin documents de varis documents d'un o varies col·leccions necessiten poder assegurar la coherència de les dades.

Per aconseguir-ho existeix la anotació @Transactional.

S'ha creat la interficia ExpedientRepositoryCustom i la seva implementació on s'hi encapsulen els mètodes que requereixen de la anotació.

D'altre banda es necessita crea un Replica Set per el MongoDB. Emprant Docker es s'ha creat la següent estructura:



Apart de permetre les transaccions, el Replica permet tenir una estructura de back-up de les dades.

Crides REST

Per fer les crides els objectes complexes afegeixen un paràmetre perquè es puguin deserialitzar correctament quan el controller rep la crida.

Tota la informació sobre com fer-ho està continguda en el Javadoc de cada classe. Al montar els objectes JSON per les crides que ho requereixin, hom ha de començar per la especificació de la API i veure quins paràmetres espera. Els objectes que requereixin d'un paràmetre adicional a la capçalera són els de tipus: Filtre, Valor. Aquest són objectes abstractes.