Laboratorium 4 – bazy NoSQL

Grupa zadań G (1p):

1. Pobierz najnowszą wersję MongoDB dla Twojego systemu operacyjnego i rozpakuj do wybranego katalogu
   http://downloads.mongodb.org/win32/mongodb-win32-x86_64-latest.zip
   http://downloads.mongodb.org/osx/mongodb-osx-ssl-x86_64-latest.tgz
   http://dl.mongodb.org/dl/linux
   W katalogu MongoDB utwórz hierarchię folderów `data\db`
   Uruchom konsolę w głównym katalogu MongoDB i uruchom MongoDB na porcie 8004 poleceniem: `bin\mongod --dbpath ./data/db/ --port 8004`
2. Wykorzystaj projekt usługi zarządzania ocenami studentów z poprzednich zajęć i dodaj w nim zależności Maven do sterownika MongoDB oraz biblioteki Morphia służącej mapowaniu obiektów Javy na obiekty MongoDB. Możesz wykorzystać dokumentację sterownika MongoDB oraz Morphia.
   W wyniku powyższych operacji powinieneś otrzymać następujące zależności w projekcie (można wykorzystać nowsze wersje niż w przykładzie):
3. ```
   <dependency>
       <groupId>org.mongodb</groupId>
       <artifactId>mongodb-driver</artifactId>
       <version>3.10.1</version>
   </dependency>
   <dependency>
       <groupId>org.mongodb</groupId>
       <artifactId>mongodb-driver-core</artifactId>
       <version>3.10.1</version>
   </dependency>
   <dependency>
       <groupId>org.mongodb</groupId>
       <artifactId>bson</artifactId>
       <version>3.10.1</version>
   </dependency>
   <dependency>
       <groupId>org.mongodb.morphia</groupId>
       <artifactId>morphia</artifactId>
       <version>1.3.2</version>
   </dependency>
   ```
4. Zmodyfikuj model danych w projekcie tak, aby dane były odczytywane i zapisywane do bazy danych MongoDB oraz została zachowana funkcjonalność usługi z poprzednich zajęć. Wykorzystaj mechanizm mapowania obiektowego Morphia poprzez dodanie odpowiednich adnotacji do klas twojego modelu.
   a. Utwórz osobne kolekcje MongoDB (co najmniej dwie) dla dokumentów głównego poziomu (np.: student, przedmiot) - zależnie od struktury twojego modelu.
   b. Do dokumentów głównego poziomu dodaj pole `@Id ObjectId _id;` reprezentujące unikalny identyfikator obiektu w bazie danych. Typ `ObjectId` jest hierarchicznym identyfikatorem obiektu, który zawiera m.in. identyfikator partycji bazy danych, a więc może służyć efektywnemu wyszukiwaniu obiektów w rozproszonych bazach MongoDB.
   c. Dla obiektów posiadających swój naturalny identyfikator – np.: indeks dla studenta, zablokuj udostępnianie sztucznego identyfikatora bazodanowego w usłudze REST wykorzystując adnotacje `@XmlTransient` oraz załóż indeks unikalny na naturalnym identyfikatorze wykorzystując odpowiednie adnotacje Morphia.
   d. Zrealizuj funkcjonalność „auto-increment” dla indeksu studenta, która będzie atomowo tworzyć kolejne indeksy dla nowych studentów dodawanych metodą POST. W tym celu utwórz dodatkową kolekcję, która będzie zawierać jeden dokument z polem zawierającym aktualny
   ```

stan sekwencji. Wykorzystaj metodę `findAndModify` aby atomowo pobierać i inkrementować wartość sekwencji.

e. Dla prawidłowej (de-)serializacji typu `ObjectId`, na odpowiednim polu klasy dopisz adnotację `@XmlJavaTypeAdapter(ObjectIdJaxbAdapter.class)` korzystającą z klasy `ObjectIdJaxbAdapter`.

f. **Uwaga!** Stosując `ObjectId` jako identyfikator zasobów REST, powinieneś zastosować typ `String` jako typ identyfikatora w klasach reprezentujących zasoby JAX-RS. Konwersja pomiędzy `String` i `ObjectId` realizowana jest poprzez odpowiedni konstruktor. **Uwaga 2!** Ponieważ serializator JAXB w JAX-RS traktuje pary getter/setter jak pole klasy, może on stwierdzić istnienie w klasie dwóch pól o tej samej nazwie, ale innych adnotacjach. Rozwiązaniem problemu może być nieznaczna zmiana w nazwach gettera/setera względem nazwy pola (pominięcie _ poniżej) oraz dopisanie adnotacji `@XmlTransient` na polu, np.:

```
@Id
@XmlTransient
private ObjectId _id;

@XmlJavaTypeAdapter(ObjectIdJaxbAdapter.class)
public ObjectId getId() {
   return _id;
}

public void setId(ObjectId id) {
   this._id = id;
}
```

g. Aby nie duplikować danych, zastosuj referencje MongoDB (`DBRef`/`@Reference`) dla dokumentów głównego poziomu referowanych w innych dokumentach (np.: dokument typu przedmiot w dokumencie typu ocena).

h. Utwórz mechanizm, który przy starcie usługi wypełni bazę danych przykładowymi danymi, jeśli jest ona pusta.

5. Wykorzystaj aplikację Robo 3T, aby zweryfikować, czy uzyskana struktura dokumentów jest zgodna z założeniami. Możesz również wykorzystać ją na potrzeby debugowania.

Grupa zadań H (1p):

6. Rozszerz interfejs REST usługi o możliwość filtrowania zasobów reprezentujących kolekcje studentów, ocen i przedmiotów. Wykorzystaj część query w istniejących URL zasobów (przykład). Usługa powinna pozwalać na filtrowanie w celu zyskania:
   a. Listy studentów po fragmencie imienia, nazwiska (każdy atrybut osobno, oraz łączenie przez &)
   b. Listy studentów, urodzonych w/przed/po zadanej dacie
      Zarejestruj parser dat `DateParamConverterProvider` przez metodę `ResourceConfig.register()` podczas uruchamiania serwera.
      Możesz wykorzystać poniższą adnotację w celu poprawnego formatowania daty w odpowiedziach JSON
      ```
      @JsonFormat(shape=JsonFormat.Shape.STRING,
      pattern="yyyy-MM-dd", timezone="CET")
      private Date birthday;
      ```
   c. Listy ocen studenta z wybranego przedmiotu
   d. Listy ocen studenta wyższych/równych/niższych niż zadana wartość
   e. Listy przedmiotów po fragmencie nazwy przedmiotu i po fragmencie nazwy prowadzącego