Zadanie 1 – Uruchomienie projektu

Pobrać projekt szkoleniowy z https://github.com/mzubala/dmsCap i zaimportować do IDE.

Uruchomić bazę danych HSQL – polecenie i baza danych znajdują się w katalogu db projektu.

Uruchomić aplikację szkoleniową z IDE `pl.com.bottega.dms.App.`

Zadanie 2a – Mapowanie encji (mapowanie podstawowe)

Utworzyć encję Document o następujących atrybutach:

- id
- number
- title
- createdAt
- verifiedAt
- publishedAt
- content

Utworzyć encję User o następujących atrybutach

- id
- login
- password

Utworzyć encję Employee o następujących atrybutach:

- id

Napisz aplikację która zapisuje do bazy danych kilka egzemplarzy w.w. encji.

Zadanie 2b – Mapowanie encji (Agregacja)

Utworzyć następujący model:

- User ma jednego Employee. Employee nie musi mieć Usera (wykluczeni cyfrowo).
- Dokument ma jednego autora. Jeden pracownik może być autorem wielu Dokumentów (jeden do wielu – dwustronna).
- Dokumnet ma jednego weryfikatora. Jeden pracownik może być weryfikatorem wielu dokumentów.
- Dokument ma jednego publikatora. Jeden pracownik może opublikować wiele dokumentów.
- Dokument ma przypisanych wielu czytelników. Pracwonik może mieć przypisanych wiele Dokumentów (wiele do wieli - dwustronna)

Zmodyfikuj aplikację z poprzedniego zadania, tak żeby tworzyła również powiązania pomiędzy zapisanymi encjami.

Zadanie 2c – Mapowanie encji (Embeddable, Enum)

Stworzyć klasę PhoneNumber mającą dwa atrybuty: countryCode i number. Dodać w klasie Employee pola workPhone i privatePhone zmapowane jako obiekty zagnieżdżone.

Dodać do klasy Dokument atrybut Status będący typem wyliczeniowym

Zadanie 2d – Operacje kaskadowe

Ustawić mapowanie tak aby:

- dodanie nowego użytkownika powodowało zapis jego pracownika
- usunięcie użytkownika NIE powodowało usunięcia pracownika
- zapis pracownika powodował zapis dokumentu dodanego do dowolnej z jego kolekcji
- usunięcie pracownika powodowało usunięcie użytkownika
- usunięcie pracownika NIE powodowało usunięcia żadnych dokumentów

Zadanie 2e – Polityka pobierania danych

Ustawić mapowanie Dokumentu tak, aby nie był pobierany wraz z autorem. Dokumenty będą pobierane w celu wyświetlenia na listach w module administracyjnym zatem pobieranie autora będzie wówczas bezcelowe i nieoptymalne.

Zadanie 3 EntityManager (zarządzanie Encjami)

Stworzyć klasę DocumentRepository, która zarządza dokumentami – operacje CRUD

- Zapis dokumentu wraz ze statusem i przypisanym autorem
- Zmiana danych dokumentu
- Usunięcie dokumentu

Zadanie 4a EntityManager (kwerendy)

Stworzyć Servis, który wyszukuje dokumenty po następujących kryteriach:

- status
- tytuł (część)
- autor
- treść (część)

Co się stanie gdy po odebraniu wyniku wywołamy document.getReaders()?

Zadanie 4b EntityManager (wpływ na pobieranie danych)

Zmodyfikować zapytania tak aby pobierały dokumenty wraz z zagregowanymi encjami w jednym

zapytaniu SQL

Zadanie 4c EntityManager (pobieranie wybranych pól)
Do servisu wyszukującego dokumenty dodać metodę, która zwraca DocumentDTO (wzorzec Data Transfer Object) zawierający: nr, status, tytuł, treść.

Zadanie 4d (Criteria API)

Ponownie zaimplementować dynamicznie składane zapytania w Servisie wyszukiwania Dokumentów z wykorzystaniem Criteria API.

Zadanie 5 (Wspólne atrybuty)
Stworzyć klasę bazową dla wszystkich encji, która zawiera:

- datę utworzenia (ustawiana w konstruktorze albo poprzez mechanizm callback)
- datę ostatniej modyfikacji – mechanizm Optimistic Locking

Zadanie 6 (Optimistic locking)

Dodać ochronę Encji Document przed jednoczesnym modyfikowaniem (adnotacja @Version). Sprowokować sytuację, w której klient B modyfikuje encję podlegającą zmianom przez klienta A.

Zadanie 7 (L2 cache)

Dodać cache dla encju User. Pobrać wielokrotnie Encję. Zmodyfikować encję w trakcie trwania prób.