ЗАДАНИЕ

1) Проектирование небольшой БД, в которой будут реализованы виды отношений one-to-one, one-to-many, many-to-many. БД должна быть нормализована по первым трем НФ.

* У всех жителей города есть паспорт.

* Жители города могут владеть одним/несколькими домами или не владеть вообще. У дома может быть один/несколько хозяев или не быть вообще.

* У жителя города может быть одна/несколько машин или не быть вообще, но у машины может быть ровно один хозяин.

2) Реализация взаимодействия с БД через ORM (Hibernate как самое популярное решение).

3) Реализовать основные CRUD-ы для созданных моделей с учетом ограничений БД.

* При создании жителя города создавать и паспорт (обернуть в транзакцию).

* Через API нельзя создать или удалить паспорт.

* При удалении жителя удаляются его паспорт и машины, обновляются списки хозяев домов (обернуть в транзакцию).

* При удалении дома обновляются списки хозяев домов (обернуть в транзакцию).

Формат ответов:

* получение объекта: объект(ы) в теле;

* добавление объекта: статус добавления и объект в теле;

* изменение объекта: статус изменения и объект в теле;

* удаление: статус удаления.

Также Реализовать обработку исключений:

Создать иерархию кастомных исключений (либо использовать существующие), покрывающих основные кейсы ошибок.

Создать обработчик исключений (RestControllerAdvice + ExceptionHandler)

И дополнить API кастомными методами:

Получить все машины, которыми владеет определенный житель города (через именованный запрос).

Получить всех хозяев домов, расположенных на определенной улице (через аннотацию @Query с помощью jpql).

Получить паспортные данные всех мужчин, чья фамилия начинается на определенную букву (через аннотацию @Query с помощью native sql).

Дополнительно: прочитать доку по ORM jOOQ, выделить преимущества/недостатки по сравнению с JPA. Добавить в проект реализацию всех методов, получающих работы с БД, через jOOQ. Добавить в проект профили, в зависимости от выбора которых будет работать либо JPA.

использовать технологии:

data jpa, mapStruct, liqudbase, dto