Hibernate в проекте "Автосалон"

Цель задания:

Разработать небольшой проект "Автосалон". Реализовать базовые CRUD-операции, оптимизировать запросы, использовать спецификации, Criteria API, а также решить проблему N+1. Опционально — реализовать кэш второго уровня.

Описание предметной области:

1. Автосалон продает автомобили различных брендов.
2. Каждый автомобиль может быть продан одному клиенту, но до продажи он привязан к автосалону.
3. Автомобили классифицируются по категориям (например, седан, внедорожник, грузовик).
4. У клиентов могут быть несколько контактов (телефон, email).
5. Клиент может участвовать в программе лояльности, если он купил более одного автомобиля.
6. Каждый клиент может оставить отзывы о купленных автомобилях.

Стэк:

- Java 17+
- Gradle

Требования:

1. Сущности проекта

- Car — автомобиль.
    - Поля: id, модель, марка, год выпуска, цена, категория.
    - Связи: ManyToOne с автосалоном, OneToMany с отзывами.
- CarShowroom — автосалон.
    - Поля: id, название, адрес, список автомобилей.
    - Связи: OneToMany с автомобилями.
- Client — клиент.
    - Поля: id, имя, контакты, дата регистрации.
    - Связи: ManyToMany с автомобилями, OneToMany с отзывами.

        - Использовать @ElementCollection для хранения контактов.
- Category — категория автомобиля.
    - Поля: id, название (седан, внедорожник и т.д.), список автомобилей.
    - Связи: OneToMany с автомобилями.
- Review — отзыв.
    - Поля: id, текст отзыва, рейтинг, клиент, автомобиль.
    - Связи: ManyToOne с клиентом, ManyToOne с автомобилем.

2. Методы для работы с данными

Реализовать сервисный слой с методами:

- Добавление, обновление и удаление данных (CRUD) для всех сущностей.
- Добавление автомобиля в автосалон.
- Привязка автомобиля к клиенту при покупке.
- Добавление отзыва клиента на автомобиль.

3. Запросы через Criteria API и спецификации

Реализовать следующие запросы:

- Поиск автомобилей по параметрам: марка, год выпуска, категория, диапазон цены.
- Сортировка автомобилей по цене (возрастание/убывание).
- Пагинация при запросе списка автомобилей.
- Полнотекстовый поиск отзывов по ключевым словам.

4. Решение проблемы N+1

При загрузке списка автомобилей из автосалона оптимизировать запрос с помощью:

- Entity Graph.
- JOIN FETCH в JPQL/Criteria API.

5.* Полнотекстовый поиск (опционально)

Настроить Hibernate Search для работы с отзывами:

- Поиск по тексту отзывов.
- Поиск отзывов с определенным рейтингом.

6.* Кэш второго уровня (опционально)

- Настроить кэш второго уровня для сущностей Car и Client.
- Проверить, что данные загружаются из кэша при повторных запросах.

Примерный функционал и запросы:

Методы для работы с сущностями

- Автомобиль:
  - Добавление автомобиля: addCar(Car car).
  - Поиск автомобилей по фильтрам: findCarsByFilters(String brand, String category, int year, double minPrice, double maxPrice).
  - Привязка автомобиля к автосалону: assignCarToShowroom(Car car, CarShowroom showroom).
- Клиент:
  - Регистрация клиента: addClient(Client client).
  - Привязка автомобиля к клиенту: buyCar(Client client, Car car).
- Отзыв:
  - Добавление отзыва клиента: addReview(Client client, Car car, String text, int rating).
  - Полнотекстовый поиск отзывов: searchReviews(String keyword).

Форма сдачи задания:

- Исходный код проекта на GitHub (сделать feature ветку с выполненным заданием и создать ПР в мастер).
- Файл README.md с инструкциями по запуску проекта.