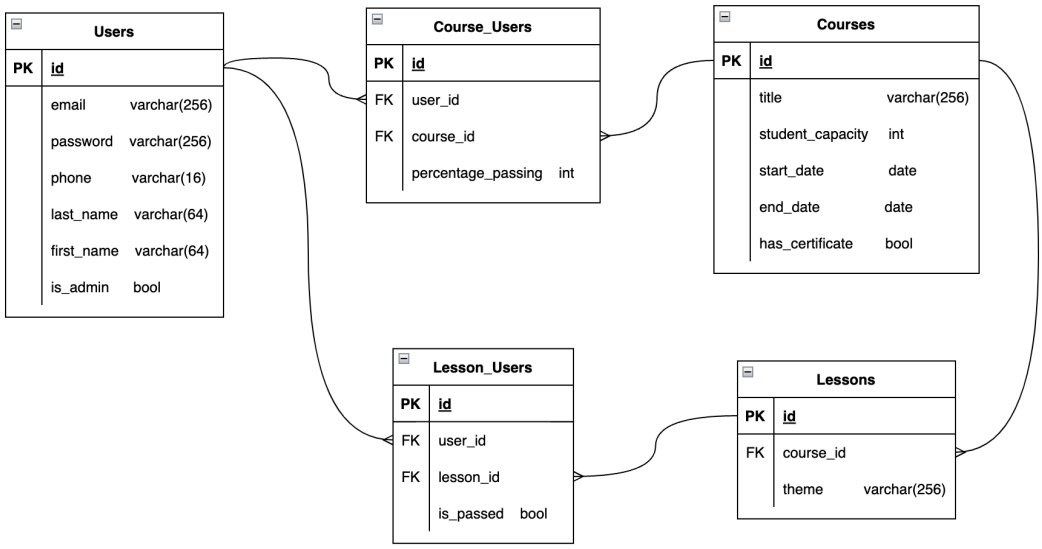
Задача

1. Развернуть docker контейнер с проектом на Lumen и контейнер для БД с СУБД PostgreSQL.
2. Указать в проекте конфигурации подключения к БД.
3. Реализовать следующую схему БД с помощью Laravel миграций:



4. Создать Eloquent модели для представленных в БД сущностей.
5. Заполнить таблицы `users`, `courses` и `course_lessons`, используя seeder классы и factory классы.
6. Реализовать следующий список API (ответ должен быть в формате json):

Адрес	Описание	Ролевой доступ
POST `/api/users/register`	Создание пользователя	Гость
POST `/api/users/login`	Авторизация. Получение jwt-токена авторизации	Гость
PUT `/api/users/1`	Редактирование пользователя	Авторизованный пользователь, владелец учетной записи
DELETE `/api/users/1`	Удаление пользователя	Авторизованный пользователь, владелец учетной записи
GET `/api/courses`	Получение списка всех курсов	Гость, авторизованный пользователь, администратор
POST `/api/courses`	Создание курса	Авторизованный администратор
POST `/api/course_users`	Запись пользователя на курс	Авторизованный пользователь, владелец учетной записи
GET `/api/course_lessons?course_id=1`	Получение списка всех уроков курса по id курса	Гость

GET `/api/users`	Получение списка всех пользователей	Авторизованный администратор
PUT `/api/course_lesson_users/1`	Завершение урока пользователем	Авторизованный пользователь, владелец учетной записи

7. Реализовать JWT авторизацию используя библиотеку tymondesigns/jwt-auth JWT токен передается в Headers - Authorization: JWT-auth каждого запроса
8. Реализовать валидацию клиентских запросов:
    - Валидация на типы данных и обязательность полей.
    - Дата окончания курса не может быть раньше даты начала.
    - Дата начала курса не может быть раньше сегодня.
9. С помощью Laravel Event/Listeners реализовать логику:
    - Один пользователь не может дважды записаться на один и тот же курс.
    - На курс можно записываться только при наличии свободных мест.
10. С помощью триггеров и триггерных функций необходимо реализовать следующее:
    - Прикреплять всех студентов, записавшихся на курс, к урокам курса.
    - При прохождении урока пересчитывать процент прохождения курса.
11. С помощью отношений расширить следующие запросы:
    - GET `/api/courses` - добавить к каждому курсу вывод всех уроков курса
    - GET `/api/students` - добавить к каждому пользователю вывод всех его курсов, отсортированных по проценту прохождения
12. Подготовить Postman коллекцию с примерами запросов реализованных в п.6 API. Экспортировать коллекцию в json файл

Проект должен содержать:

- Папку deploy с docker-compose file
- Папку project с Lumen проектом
- Папку collection с экспортированной Postman коллекцией

Проект разместить в github/gitlab репозитории