## Задача

Нужно сделать простой сервис для управления номерами отелей и бронированиями.

* Методы реализуются в виде HTTP-хендлеров, на выходе возвращают JSON в теле ответа. В случае ошибок нужно вернуть текст ошибки (тоже в JSON).
* Сервис должен иметь хендлеры для работы с каталогом номеров отеля:
  + добавить номер отеля. Принимает на вход текстовое описание и цену за ночь. Возвращает ID номера отеля.
  + удалить номер отеля и все его брони. Принимает на вход ID номера отеля.
  + получить список номеров отеля. Должна быть возможность отсортировать по цене или по дате добавления (по возрастанию и убыванию).
* Сервис должен иметь хендлеры для работы со списком броней номеров:
  + добавить бронь. Принимает на вход существующий ID номера отеля, дату начала, дату окончания брони (проверять, свободен ли отель на эти даты, не нужно; даты должны быть в формате “год-месяц-день”, например: “2020-01-30”; даты должны быть валидными). Возвращает ID брони.
  + удалить бронь. Принимает на вход ID брони.
  + получить список броней номера отеля. Принимает на вход ID номера отеля. Возвращает список бронирований, каждое бронирование содержит ID, дату начала, дату окончания. Бронирования должны быть отсортированы по дате начала.
* Примеры запросов и ответов:
  + Запрос создания брони: curl -X POST -d "room\_id=24” -d "date\_start=2021-12-30" -d "date\_end=2022-01-02" http://localhost:9000/bookings/create
  + Ответ: {"booking\_id": 1444}
  + Запрос списка броней номера: curl -X GET "http://localhost:9000/bookings/list?room\_id=24"
  + Ответ: [{"booking\_id": 1444, "date\_start": "2021-12-30", "date\_end": "2022-01-02"}, ...]

## Требования

* HTTP JSON API (можно вызвать, например, через curl), без авторизации.
* Данные не теряются между перезагрузками сервиса.
* Язык программирования Go/PHP — очень желательно, можно на Python/JavaScript/Java/C#.
* База данных PostgreSQL/MySQL. Миграции не нужны, достаточно файла с запросами на создание всех таблиц в БД.
* Можно использовать любой фреймворк или обойтись без него.
* Графический интерфейс делать необязательно, но можно при желании.
* Стараться использовать оптимальные подходы (алгоритмы, структуры, индексы).
* Простая инструкция для запуска (в идеале — с возможностью запустить через docker-compose up, но это необязательно).
* Код на GitHub (просьба не делать форк этого репозитория, чтобы не плодить плагиат).
* Если есть сомнения по деталям — решение принять самостоятельно, но в своём README.md рекомендуем выписать вопросы и принятые решения по ним.

## Дополнительные усложнения

Это необязательно, но будет плюсом:

* Юнит-тесты.
* Документация (достаточно структурированного описания методов и примеров их вызова в README.md).
* Реализовать проверку в момент бронирования, свободен ли номер на даты новой брони. Например, при наличии брони на 20-25 июня не давать бронировать на 23-27 июня.