Дополнительные советы ментора

Взаимодействие сервисов через REST

Для реализации взаимодействия сервисов через REST мы будем использовать `RestTemplate` — это самый простой и не требующий дополнительных библиотек способ отправить REST-запрос из одного сервиса в другой.

В шаблонном проекте мы подготовили для вас класс `BaseClient`, который реализует основные REST-методы с использованием `RestTemplate` — советуем вам внимательно изучить его код. Обратите внимание, что при отправке каждого запроса мы добавляем в него заголовки: `contentType`, `accept` и `X-Sharer-User-Id`, а также передаём параметры оригинального запроса пользователя.

Также в качестве примера мы создали класс `BookingClient`, наследующий от `BaseClient` и содержащий операции для работы с бронированиями. И класс `BookingCotroller` — он содержит пример реализации некоторых конечных эндпоинтов, с которыми будут работать пользователи.

Вам нужно ознакомиться с этими классами и при необходимости доработать их. Например, добавить недостающие эндпоинты в `BookingCotroller`, а также по аналогии реализовать контроллеры и REST-клиенты для остальных сущностей.

Подробнее про Docker

Продумайте зависимости между сервисами `shareIt-server`, `shareIt-gateway` и `postgresql` и укажите их в разделе `depends on` файла `docker-compose.yaml`.

Также продумайте нюансы развёртывания. Все сервисы будут запускаться на одном IP-адресе — соответственно, они не могут занимать одинаковые сетевые порты.

Обратите внимание: так как PostgreSQL теперь запускается в контейнере, мы связали в `docker-compose.yaml` переменную окружения `SPRING_DATASOURCE_URL` с адресом БД, работающей в контейнере.

Аналогично укажите в `application.properties` для `shareIt-gateway` параметры доступа к сервису `shareIt-server` — это нужно для работы через REST.

Чтобы тестировать работу базы данных, запущенной в Docker, вы можете обращаться к ней как к обычной БД. Используйте порт `6541` — ведь в `docker-compose.yaml` мы связали этот порт с портом `5432` внутри контейнера.