В прошлом спринте вы приступили к проекту ShareIt и уже сделали немало — например, реализовали слой контроллеров для работы с вещами. В этот раз вы продолжите совершенствовать сервис, так что он станет по-настоящему полезным для пользователей.

Перед вами две большие задачи: добавить работу с базой данных в уже реализованную часть проекта, а также дать пользователям возможность бронировать вещи.

**Немного подготовки**

В этом спринте разработка будет вестись в ветке add-bookings. Создайте ветку с таким названием и переключитесь на неё.

Далее переходите к настройке базы данных. Пришло время использовать Hibernate и JPA самостоятельно. Для начала добавьте зависимость spring-boot-starter-data-jpa и драйвер postgresql в файл pom.xml.

**Создание базы данных**

Теперь поработайте над структурой базы данных. В ней будет по одной таблице для каждой из основных сущностей, а также таблица, где будут храниться отзывы.

Подумайте, какой тип данных PostgreSQL лучше подойдёт для каждого поля. В качестве подсказки проанализируйте таблицы, которые были использованы в приложении Later.

Напишите SQL-код для создания всех таблиц и сохраните его в файле resources/schema.sql — Spring Boot выполнит содержащийся в нём скрипт на старте проекта. На данный момент вам достаточно создать таблицы для двух сущностей, которые вы уже разработали — Item и User.

Важный момент: приложение будет запускаться много раз, и каждый раз Spring будет выполнять schema.sql. Чтобы ничего не сломать и не вызвать ошибок, все конструкции в этом файле должны поддерживать множественное выполнение. Это значит, что для создания таблиц следует использовать не просто конструкцию CREATE TABLE, но CREATE TABLE IF NOT EXIST — тогда таблица будет создана, только если её ещё не существует в базе данных.

Подсказка: пример кода для создания таблицы users

**Настройка JPA**

Пора подготовить сущности к работе с базой данных. Мы говорили, что для этого используют аннотации JPA: @Entity, @Table, @Column, @Id. Для поля status в классе Booking вам также пригодится @Enumerated. Добавьте соответствующие аннотации для сущностей.

Создайте репозитории для User и Item и доработайте сервисы, чтобы они работали с новыми репозиториями.

Подсказка: маппинг между столбцами БД и моделью данных

**Реализация функции бронирования**

Чтобы сделать приложение ещё более полезным и интересным, добавьте возможность брать вещи в аренду на определённые даты.

Вот основные сценарии и эндпоинты:

* Добавление нового запроса на бронирование. Запрос может быть создан любым пользователем, а затем подтверждён владельцем вещи. Эндпоинт — POST /bookings. После создания запрос находится в статусе WAITING — «ожидает подтверждения».
* Подтверждение или отклонение запроса на бронирование. Может быть выполнено только владельцем вещи. Затем статус бронирования становится либо APPROVED, либо REJECTED. Эндпоинт — PATCH /bookings/{bookingId}?approved={approved}, параметр approved может принимать значения true или false.
* Получение данных о конкретном бронировании (включая его статус). Может быть выполнено либо автором бронирования, либо владельцем вещи, к которой относится бронирование. Эндпоинт — GET /bookings/{bookingId}.
* Получение списка всех бронирований текущего пользователя. Эндпоинт — GET /bookings?state={state}. Параметр state необязательный и по умолчанию равен **ALL** (англ. «все»). Также он может принимать значения **CURRENT** (англ. «текущие»), \*\*PAST\*\* (англ. «завершённые»), **FUTURE** (англ. «будущие»), **WAITING** (англ. «ожидающие подтверждения»), **REJECTED** (англ. «отклонённые»). Бронирования должны возвращаться отсортированными по дате от более новых к более старым.
* Получение списка бронирований для всех вещей текущего пользователя. Эндпоинт — GET /bookings/owner?state={state}. Этот запрос имеет смысл для владельца хотя бы одной вещи. Работа параметра state аналогична его работе в предыдущем сценарии.

Для начала добавьте в модель данных сущность Booking и код для создания соответствующей таблицы в файл resources/schema.sql.

Создайте контроллер BookingController и методы для каждого из описанных сценариев. Подумайте, не нужно ли написать дополнительные DTO-классы для каких-то сценариев.

Кроме контроллеров, необходимо реализовать хранение данных — то есть сервисы и репозитории.

Подсказка: какие могут быть изменения в DTO

Например, может быть полезно создать отдельное перечисление для возможных методов параметра state, ведь задачи этого перечисления могут отличаться в слое представления (параметр для поиска) и в модели данных (состояние бронирования).

**Добавление дат бронирования при просмотре вещей**

Осталась пара штрихов. Итак, вы добавили возможность бронировать вещи. Теперь нужно, чтобы владелец видел даты последнего и ближайшего следующего бронирования для каждой вещи, когда просматривает список (GET /items).

**Добавление отзывов**

Мы обещали, что пользователи смогут оставлять отзывы на вещь после того, как взяли её в аренду. Пришло время добавить и эту функцию!

В базе данных уже есть таблица comments. Теперь создайте соответствующий класс модели данных Comment и добавьте необходимые аннотации JPA. Поскольку отзыв — вспомогательная сущность и по сути часть вещи, отдельный пакет для отзывов не нужен. Поместите класс в пакет item.

Комментарий можно добавить по эндпоинту POST /items/{itemId}/comment, создайте в контроллере метод для него.

Реализуйте логику по добавлению нового комментария к вещи в сервисе ItemServiceImpl. Для этого также понадобится создать интерфейс CommentRepository. Не забудьте добавить проверку, что пользователь, который пишет комментарий, действительно брал вещь в аренду.

Осталось разрешить пользователям просматривать комментарии других пользователей. Отзывы можно будет увидеть по двум эндпоинтам — по GET /items/{itemId} для одной конкретной вещи и по GET /items для всех вещей данного пользователя.

**Тестирование**

Для проверки всей функциональности, которую вы добавили в этом спринте, мы подготовили [Postman-коллекцию](https://github.com/yandex-praktikum/java-shareit/blob/add-bookings/postman/sprint.json) — используйте её для тестирования приложения.

**Дополнительные советы ментора**

Как и в прошлом задании спринта, более подробную информацию вы найдёте в файле: [Дополнительные советы ментора](https://code.s3.yandex.net/Java/14sprint/MentorsAdvice_05_04_23v4.pdf).

На этом пока всё: вы отлично потрудились! Как всегда, интересного вам программирования!