|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА № 5**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Тема практической работы:** Создание REST приложения с использованием Java-фреймворка Spring

|  |  |
| --- | --- |
| Студент группы ИКБО-32-21 | Быченков А.К. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

|  |  |
| --- | --- |
| **Руководитель практической работы** | преподаватель Волков М.Ю. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

**Цель работы:** получить навыки создания REST приложений с использованием Java-фреймворка Spring.

**Задание**

Студенту предлагается создать свое серверное CRUD приложение. Приложение должно стать основанием для создания маркетплейса. В работе должны быть реализованы такие модели:

1. Book – модель, которая описывает сущность книги, с полями: автор, номер продавца, тип продукта (подразумевается электроника, книги, сантехника и т. п.), стоимость, название.
2. Client – модель, которая описывает сущность клиента сайта. У модели должны быть поля: имя, электронная почта, логин, пароль.
3. Telephone – модель, описывающая сущность телефона. Обязательные поля: производитель, объем аккумулятора, номер продавца, тип продукта (подразумевается электроника, книги, сантехника и т. п.), стоимость, название.
4. WashingMachine – модель, которая описывает сущность стиральной машины, с такими полями: производитель, объем бака, номер продавца, тип продукта (подразумевается электроника, книги, сантехника и т. п.), стоимость, название.

Так же должны быть контроллеры, которые позволят манипулировать модели (CRUD). Вся настройка приложения должна быть реализована в директории configuration. Приложение должно запускаться с помощью docker-compose.

**Ход работы**

Были созданы сущности базы данных для книг, клиентов, телефонов и стиральных машин. Код для сущности книг показан на рисунке 1.

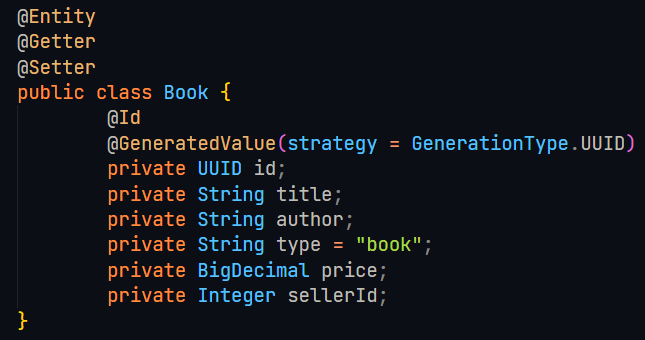


Рисунок 1 – Листинг Book.java

Для миграций структуры базы данных используется библиотека Flyway. Был написан файл миграции, показанный на рисунке 2.

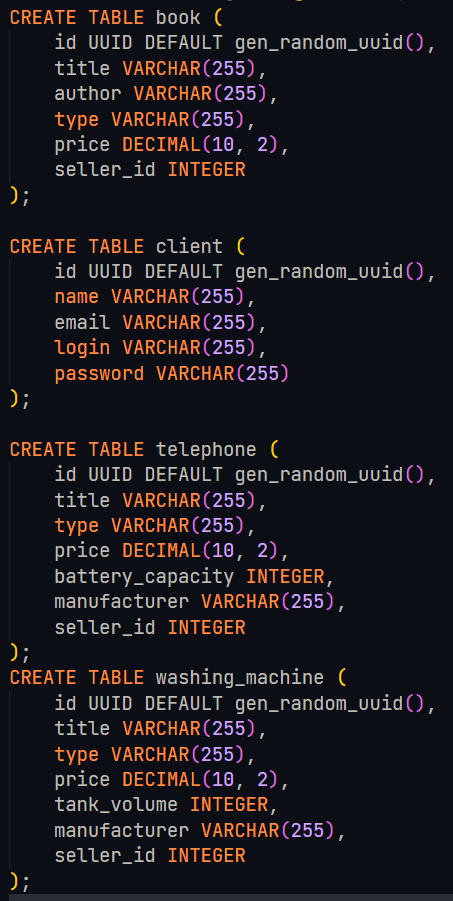


Рисунок 2 – Листинг V1\_\_init\_db.sql

Для работы с базой данных используются JPA репозитории. На рисунке 3 показан код создания репозитория для сущности книг.

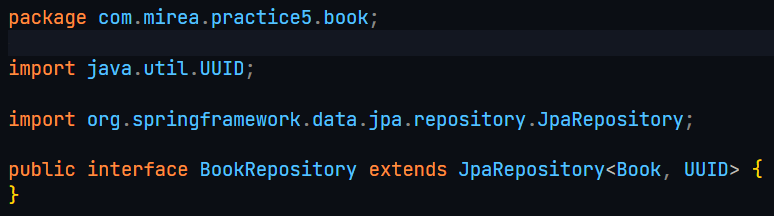


Рисунок 3 – Листинг BookRepository.java

На рисунке 4 показан код создания сервиса для сущности книг.



Рисунок 4 – Листинг BookService.java

На рисунке 5 показана реализация контроллера для сущности книг.



Рисунок 5 – Листинг BookController.java

Аналогичным образом был написан код и для других сущностей.

Для запуска приложения написан файл Docker Compose, показанный на рисунке 6.

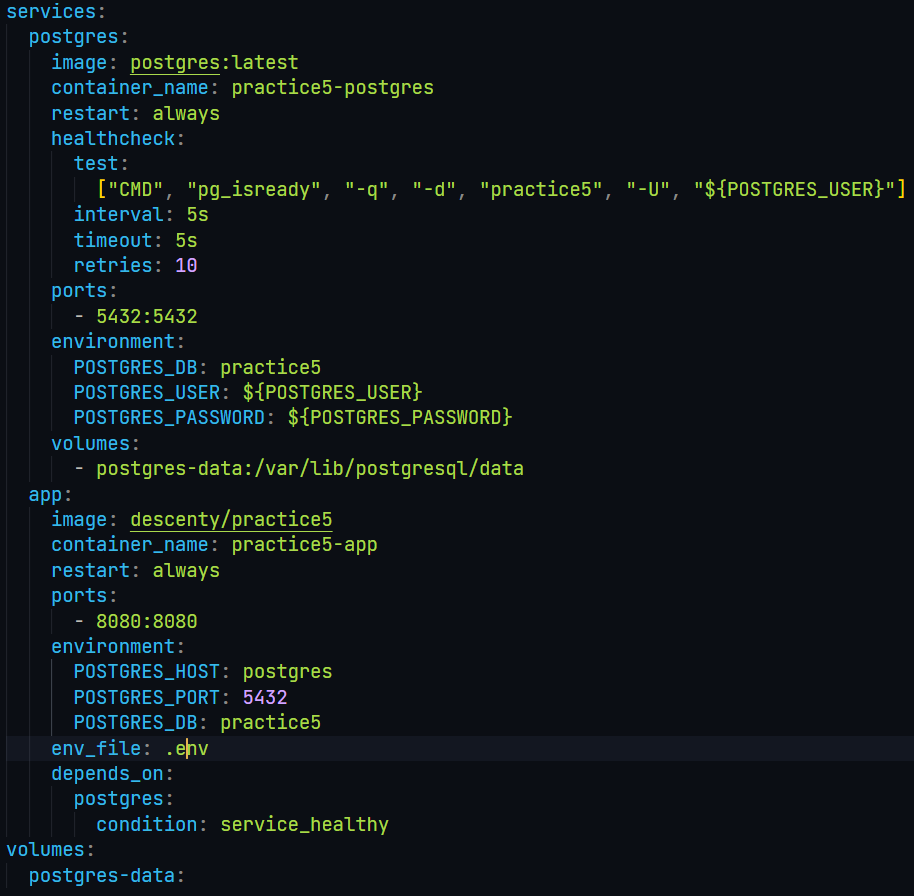


Рисунок 6 – Листинг compose.yaml

Для создания образа Spring Boot приложения используется популярный инструмент Jib от компании Google, который позволяет собирать максимально оптимизированные образы Java-приложений. Команда для запуска приложения показана на рисунке 7.



Рисунок 7 – Команда запуска приложения

Было проведено автоматизированное тестирование API с помощью инструмента Cypress. Пример тестов и результаты показаны на рисунках 8, 9, 10.



Рисунок 8 – Листинг book.spec.js



Рисунок 9 – Результаты тестов

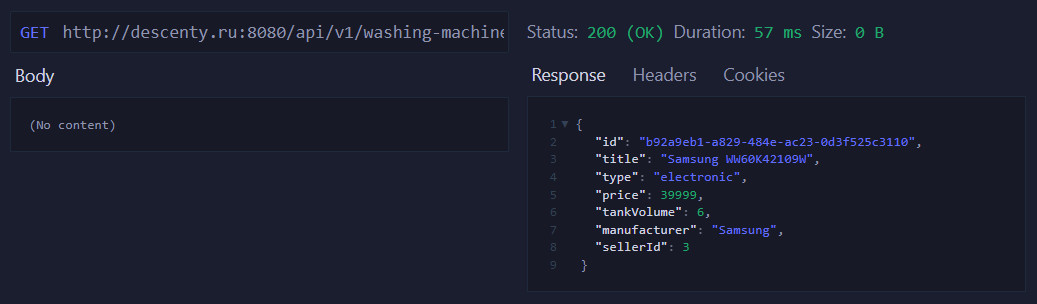


Рисунок 10 – Пример запроса для получения стиральной машины

**Вывод**

В ходе выполнения практической работы были получены навыки создания REST-приложений с Java-фреймворком Spring.

**Ответы на вопросы**

1. **Что такое JDBC и JPA, в чем отличия?**

JDBC (Java Database Connectivity) и JPA (Java Persistence API) — это два различных подхода к взаимодействию с базами данных в приложениях, разрабатываемых на языке Java. Они предоставляют разные уровни абстракции и функциональности для работы с базами данных. Вот их краткое описание и отличия:

JDBC (Java Database Connectivity):

* Является низкоуровневым API, предоставляющим Java-приложениям доступ к базам данных через SQL-запросы.
* При использовании разработчикам приходится самим писать SQL-запросы и управлять процессами, такими как установка соединения с базой данных, выполнение запросов, обработка результатов и управление транзакциями.
* Предоставляет прямой доступ к базе данных, что может быть полезно в случае, когда требуется максимальный контроль над запросами и производительностью.

JPA (Java Persistence API):

* Представляет собой более высокоуровневый и абстрактный способ работы с базами данных в Java-приложениях.
* Основан на объектно-реляционном отображении (ORM), что позволяет разработчикам работать с объектами Java, представляющими данные, вместо прямой работы с SQL-запросами и таблицами баз данных.
* Облегчает выполнение операций CRUD (Create, Read, Update, Delete) с помощью аннотаций и API для управления сущностями и их отношениями.
* Популярные реализации JPA включают в себя Hibernate, EclipseLink и др.

Отличия между JDBC и JPA:

* Уровень абстракции: JDBC — это низкоуровневый API, тогда как JPA предоставляет более высокоуровневую абстракцию.
* Работа с SQL: В JDBC SQL-запросы пишутся напрямую, в то время как в JPA происходит работы с объектами Java, не приходится писать SQL-запросы (или крайне редко).
* Производительность: JDBC может предоставить более точный контроль над производительностью, но требует больше кода. JPA обеспечивает более быстрое и простое взаимодействие с базой данных за счет абстракции, но может потребовать оптимизации для сложных случаев.
* Сложность: JDBC более сложен в использовании, особенно для новичков, тогда как JPA упрощает разработку благодаря аннотациям и объектно-реляционному отображению.

1. **Основные интерфейсы для реализации Spring Data**

Spring Data предоставляет ряд основных интерфейсов, которые можно реализовать для создания репозиториев и упрощения взаимодействия с базами данных. Вот некоторые из наиболее важных интерфейсов Spring Data:

* Repository. Это базовый интерфейс Spring Data, который предоставляет основные методы для работы с сущностями (CRUD операции). Репозитории, реализующие этот интерфейс, должны указывать тип сущности и тип её ID.
* CrudRepository. Расширяет Repository и предоставляет методы для выполнения CRUD (Create, Read, Update, Delete) операций.
* PagingAndSortingRepository. Расширяет CrudRepository и добавляет методы для пагинации и сортировки данных.

1. **Аннотации @Entity, @Table, в чем разница?**

Аннотации @Entity и @Table являются часто используемыми аннотациями в Java Persistence API (JPA), которые используются для описания сущностей и их отображения в базе данных. Они выполняют разные роли и имеют следующие различия:

@Entity – это JPA аннотация, которая помечает класс как сущность (entity) и указывает, что этот класс может быть сохранен в базе данных как запись в таблице. Класс, помеченный @Entity, обычно представляет объектную модель данных и является аналогом таблицы в базе данных. JPA использует аннотацию @Entity для сканирования классов и создания соответствующих таблиц в базе данных.

@Table – это также JPA аннотация, которая может использоваться вместе с @Entity или отдельно для настройки деталей отображения класса в таблицу базы данных. Аннотация @Table позволяет указать имя таблицы в базе данных, которая будет соответствовать классу. Таким образом, вы можете явно указать имя таблицы, если оно отличается от имени класса. Кроме указания имени таблицы, @Table позволяет настраивать и другие параметры таблицы, такие как схема (schema), каталог (catalog), уникальные ограничения (uniqueConstraints) и другие.

1. **Что такое Lombok? Приведите несколько примеров использования**

Lombok – это библиотека для языка Java, которая упрощает и автоматизирует процесс создания кода, такого как геттеры, сеттеры, конструкторы и другие стандартные методы. Lombok добавляет аннотации в код, которые сообщают компилятору, какие методы и функции должны быть автоматически сгенерированы. Это позволяет уменьшить объем шаблонного кода в проектах и улучшить читаемость кода.

Примеры использования Lombok и его аннотаций:

* Геттеры и сеттеры. С помощью аннотаций @Getter и @Setter Lombok автоматически создает геттеры и сеттеры. Это уменьшает необходимость писать эти методы вручную.
* Конструкторы. Аннотации @AllArgsConstructor и @NoArgsConstructor генерируют конструкторы с аргументами и без них соответственно. Это особенно полезно, когда есть много полей в классе.
* Преобразование в строку. Аннотация @ToString создает метод toString(), который автоматически возвращает строковое представление объекта с данными о его полях.
* Паттерн Строитель. Аннотация @Builder создает паттерн Строитель (Builder), который упрощает создание объектов с множеством параметров.

1. **Назовите способы инициализации бинов в Java.**

В Spring Framework, бины (компоненты) могут быть инициализированы различными способами. Основные способы инициализации бинов в Java Spring:

* Аннотация @Component и автоматическое сканирование. Пометка класса аннотацией @Component, такой как @Service, @Repository, или @Controller, и использование автоматического сканирования компонентов (@ComponentScan) для обнаружения и инициализации бинов.
* XML-конфигурация. Использование файлов конфигурации XML для определения бинов и их зависимостей. Это традиционный способ конфигурации Spring.
* Аннотация @Configuration и Java-конфигурация. Создание конфигурационного класса, помеченного аннотацией @Configuration, и определение бинов в этом классе с использованием методов, помеченных аннотацией @Bean.

1. **Что такое миграции и как они работают в библиотеке Flyway?**

Миграции (Database Migrations) – это процесс изменения структуры базы данных с течением времени, чтобы поддерживать схему базы данных в актуальном состоянии, соответствующем текущим требованиям приложения. Миграции включают в себя создание, изменение и удаление таблиц, добавление и удаление столбцов, а также другие операции с базой данных.

Библиотека Flyway – это инструмент для управления миграциями базы данных. Она обеспечивает автоматическую и контролируемую эволюцию схемы базы данных, что делает ее подходящей для проектов, где схема базы данных должна изменяться вместе с развитием приложения. Как работает Flyway:

* Создание миграций. Разработчик создает SQL-скрипты, которые описывают изменения схемы базы данных. Эти скрипты обычно содержат SQL-запросы для создания, изменения или удаления таблиц и столбцов.
* Именование миграций. Каждый миграционный скрипт должен иметь уникальное имя, обычно включающее временную метку, чтобы гарантировать порядок выполнения миграций.
* Хранение миграций. Скрипты миграций обычно хранятся в специальном каталоге внутри проекта или в отдельном репозитории. Flyway сканирует этот каталог для поиска миграций.
* Использование метаданных. Flyway хранит информацию о примененных миграциях в специальной таблице базы данных (обычно называемой `schema\_version` или подобным образом). Это позволяет Flyway отслеживать, какие миграции уже были применены, и применять только новые.
* Применение миграций. При запуске приложения Flyway автоматически обнаруживает новые миграции и применяет их к базе данных в порядке их временных меток. Если какие-либо миграции были пропущены, Flyway автоматически применит их.

Преимущества использования Flyway включают в себя:

* Управление версией схемы базы данных.
* Автоматическое применение миграций при запуске приложения.
* Возможность отката миграций (Down migrations) в случае необходимости.
* Поддержка различных баз данных, таких как PostgreSQL, MySQL, Oracle, SQL Server и других.
* Интеграция с системами сборки (например, Maven, Gradle) для автоматического выполнения миграций при сборке проекта.

1. **Что такое сущность и модуль, в чем разница?**

Различие между сущностью и модулем заключается в том, что сущность относится к объектам или классам, представляющим данные и функциональность, в то время как модуль описывает изолированный компонент системы, который может выполнять определенные функции или служить логически независимой частью кода.

**Ссылка на удаленный репозиторий проекта**

<https://github.com/descenty/server-side-5-sem>

**Список использованной литературы**

* 1. Статья про REST приложения – <https://habr.com/ru/articles/471140>
  2. Статья про аннотирование контроллеров – <https://javarush.com/quests/lectures/questspring.level05.lecture02#>
  3. Статья про аннотацию @PathVariable – <https://www.baeldung.com/spring-pathvariable>
  4. CRUD приложение на Java – <https://ru.hexlet.io/courses/java-web/lessons/crud-in-db/theory_unit>
  5. Знакомство с Flyway – <https://habr.com/ru/companies/otus/articles/506788>
  6. Введение в Spring Data JDBC – <https://habr.com/ru/companies/otus/articles/531332>
  7. Введение в Spring Data JPA – <https://habr.com/ru/articles/435114>
  8. Gson – <https://habr.com/ru/companies/naumen/articles/228279>
  9. Lombok – <https://habr.com/ru/companies/piter/articles/676394>