**Практическая работа №5.**

**Теоретическое введение.**

**Почему использую Java в backend разработке.**

Java — это строго типизированный объектно-ориентированный язык программирования. Приложения на Java транслируются в байт-код, поэтому их можно запускать на любом устройстве, где установлена виртуальная машина Java (JVM).

Язык Java отличается завидной универсальностью. На нём можно писать что угодно: серверные и десктопные программы, мобильные и веб-приложения, игры любой сложности.

Чаще всего на Java пишут Backend (бэкенд). А вот теперь самое время узнать, что это такое и за что отвечает бэкенд-разработчик.

Благодаря своей надёжности и прекрасной обратной совместимости Java прекрасно подходит для корпоративных разработок. Например, на нём написана Jira — пожалуй, самая популярная платформа для управления проектами. По тем же причинам Java очень любят в финтехе, а такие банки, как «Сбер», «Альфа» и «Райффайзен» регулярно хантят джавистов.

У Java есть прекрасные технологии и в десктопе — например, OpenJFX для создания локальных приложений с графическим интерфейсом. Среды разработки ПО (о них — ниже) Eclipse IDE и IntelliJ IDEA тоже написаны на Java.

Бэкендер должен знать преимущества и недостатки технологий, чтобы выбрать ту, которая лучше всего подходит для текущего проекта. Он должен хорошо разбираться в выбранной технологии или иметь возможность быстро её изучить. Рассмотрим основные инструменты, с которыми работают бэкендеры на Java.

JDK, или Java Development Kit, — это набор инструментов для разработки ПО на языке Java. В него входят компилятор Javac, виртуальная машина Java, стандартная библиотека модулей, классов и функций и документация.

Стандартный дистрибутив JDK поддерживает компания Oracle, но помимо него существует и ряд других. Например, Red Hat, SAP, Google разрабатывают свои инструменты и вносят вклад в развитие свободной версии JDK — OpenJDK. В России, с учётом курса на замещение продуктов зарубежных вендоров, имеет смысл использовать отечественный рантайм Axiom JDK — это единственный JDK, внесённый в реестр российского ПО.

Фреймворк — это шаблон, на основе которого создаются приложения. Самый популярный фреймворк для Java — Spring, но недавно у него появились конкуренты Micronaut и Quarkus, заточенные под создание облачных приложений и микросервисов.

В повседневной работе любой разработчик активно пользуется библиотеками, которые содержат готовые решения для многих распространённых задач программирования. Помимо стандартной библиотеки Java, существует много дополнительных полезных проектов — например, Maven, JHipster, Apache Commons и так далее. Вы так или иначе познакомитесь с этими инструментами в процессе обучения и увидите, насколько они облегчают жизнь разработчика.

Все практики на Java в этом курсе будут выполняться с помощью библиотеки Spring. Поговорим про нее подробнее. По сути Spring Framework представляет собой просто контейнер внедрения зависимостей, с несколькими удобными слоями (например: доступ к базе данных, прокси, аспектно-ориентированное программирование, RPC, веб-инфраструктура MVC). Это все позволяет вам быстрее и удобнее создавать Java-приложения.

Вся работа со Spring ведется с помощью подключенных зависимостей, к которым программист может обращаться с помощью специальных аннотаций. В рамках этой практической задачи мы познакомимся с такими зависимостями как:

1. @Controller - эта аннотация создаст бин, с помощью которого Spring создаст класс контроллер, который будет принимать запросы;
2. @ResponseBody - аннотация, которая дает фреймворку понять, что результат метода - это неконечное представление, которое нужно отправить клиенту, нужно результат предварительно прогнать через класс HttpMessageConverter;
3. @RestController - это составная аннотация, которая сама мета-аннотируется аннотациями @Controller и @ResponseBody;
4. @GetMapping - обрабатывает get запросы;
5. @PostMapping - обрабатывает post запросы;
6. @PathVariable - позволяет получать значения параметров метода из запроса на сервер;
7. @RequestParam - это аннотация, которая перехватывает параметр запроса;
8. @RequestMapping - с помощью этой аннотации можно разделять запросы по путям;
9. . @SpringBootApplication - аннотация, которая создает начальную конфигурацию приложения (@EnableAutoConfiguration, @ComponentScan, @Configuration).

**CRUD приложения в Java.**

Несмотря на огромное число разнообразных сайтов, практически всю веб разработку можно свести к CRUD операциям. CRUD – широко распространенный термин, означающий 4 стандартные операции над любой сущностью (ресурсом): создание, чтение, обновление и удаление. Например, в случае с пользователем можно составить такое соответствие:

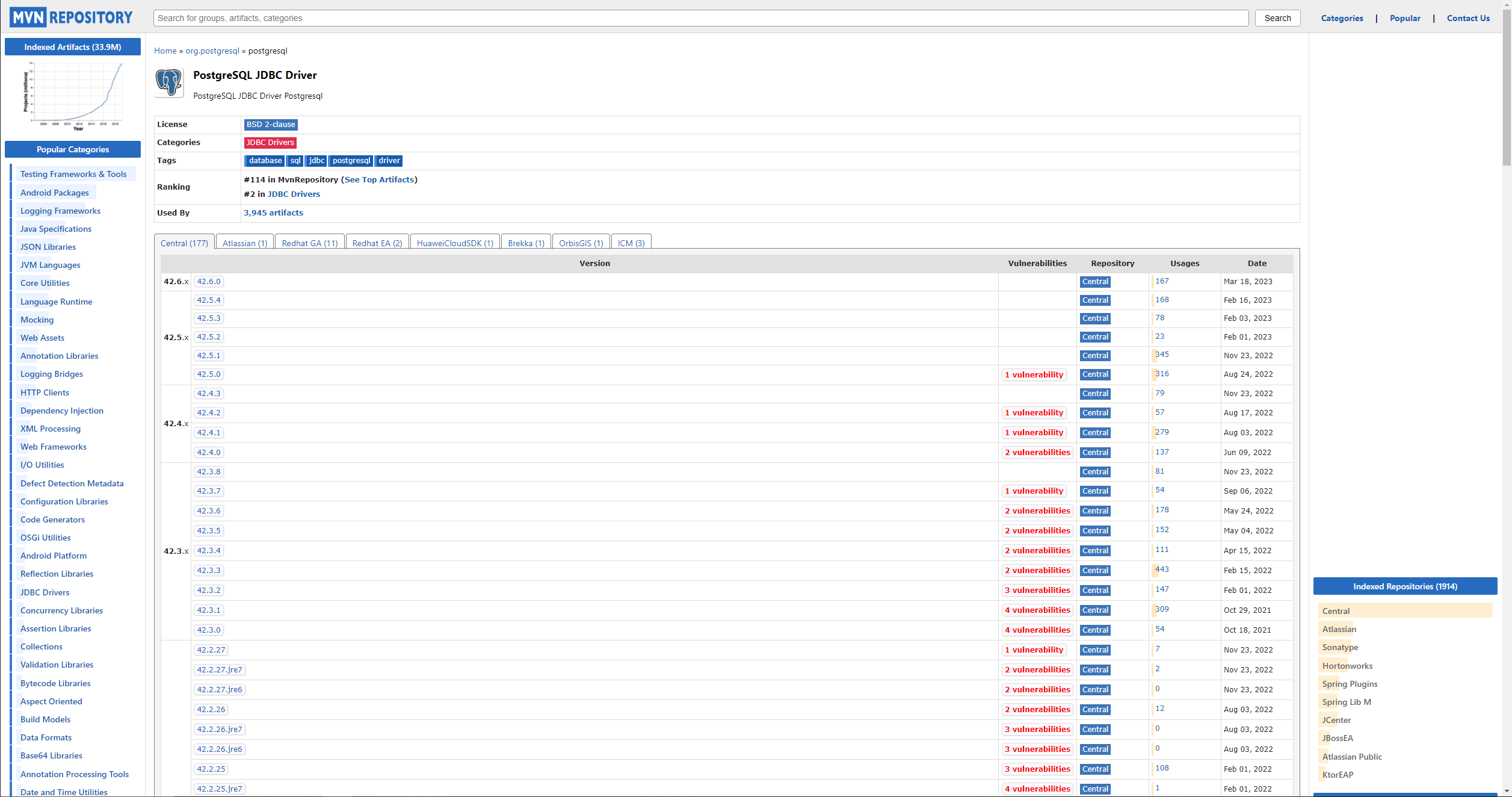
* Create – регистрация нового пользователя
* Read – Просмотр профиля пользователями сайта
* Update – Обновление личных данных пользователя (смена емейла, пароля)
* Delete – Удаление пользователя

Точно так же можно расписать действия над любыми другими ресурсами: фотографиями пользователя, его друзьями, сообщениями и т.п.

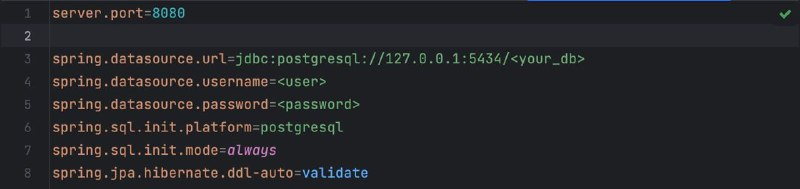
В данной практике будет использоваться Gradle, СУБД PostgreSQL и такие зависимости: Spring Data JDBC/JPA, Lombok, Gson, Flyway. Разберем каждый пункт по отдельности. Так как мы используем Gradle, то все зависимости подключаются в файле build.gradle.kts.

**PostgreSQL**

Для того, чтобы ваше приложение могло общаться с СУБД нужно подключить коннектор. Все нужные зависимости можно подключить с помощью сайта mvnrepository.com. Пишем в поисковую строку «postgresql» и ищем PostgreSQL JDBC Driver, дальше выбираем подходящую версию и подключаем в build.gradle.kts.

Рисунок 1 — версии драйвера (коннектора)

После подключение коннектора, нужно настроить подключение к СУБД, с помощью файла application.properties.

Рисунок 2 — пример настройки приложения

**Flyway**

Flyway обновляет версии баз данных с помощью миграций. Миграции можно писать на SQL (с синтаксисом, специфичным для конкретной СУБД) или на Java.Миграции могут быть версионными или повторяющимися. Первые имеют уникальную версию и применяются ровно один раз. У вторых номера версии нет, и они применяются, когда у них изменяется контрольная сумма.Повторяющиеся миграции в рамках одного запуска всегда применяются после выполнения версионных миграций. Повторяющиеся миграции применяются в порядке их описания. В одной миграции все операции выполняются в рамках одной транзакции базы данных.

**Spring Data JDBC**

Spring Data JDBC был анонсирован 2018 году. Целью было предоставить разработчикам более простую альтернативу JPA, продолжая при этом следовать принципам Spring Data. Подробнее узнать о мотивах, лежащих в основе проекта, вы можете в документации.

Spring Data — дополнительный удобный механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечение данных, изменение, в каких то случаях для этого будет достаточно объявить интерфейс и метод в нем, без имплементации.

Для работы со Spring Data студенту нужно будет изучить такие аннотации: @Repository, @Entity, @Table и все дополнительные аннотации для настройки моделей сущностей.

**Lombok**

Lombok – это библиотека Java, в которой предоставляется ряд аннотаций, направленных на исключение именно того кода Java, о котором известно, что он часто становится повторяющимся и/или шаблонным. Проект Lombok включается прямо в процесс сборки. Затем Lombok автоматически сгенерирует для Java байт-код, который вставляет в файлы .class, необходимые для реализации желаемого поведения, в зависимости от используемых вами аннотаций. Следовательно, каждая аннотация, предлагаемая в проекте Lombok, позволяет частично обойтись без написания методов и логики, без которых вы хотели бы обойтись. Речь о конструкторах, равенствах и функциях хеш-кода. Так вы сможете сэкономить массу времени и сосредоточиться на бизнес-логике вашего проекта. Кроме того, вы сможете держать базу кода сравнительно компактной, чистой, удобной для чтения и поддержки.

**Gson**

Gson – это библиотека, которая предназначенна для преобразования Java-объектов в текстовый формат JSON (сериализация) и обратного преобразования (десереализация). Часто при работе с Gson хватает стандартных настроек библиотеки, но бывают случаи, когда необходимо кастомизировать процессы преобразований.

В практической работе нельзя просто создать объект этого класса в каждом контроллере, нужно сделать bean объекта этого класса, который будет singleton.

**Полезные ссылки:**

1. Статья про rest приложения - https://habr.com/ru/articles/471140/
2. Статья для аннотирования контроллеров - https://javarush.com/quests/lectures/questspring.level05.lecture02#
3. Статья про аннотация @ResponseBody - https://www.baeldung.com/spring-request-response-body
4. Статья про аннотация @SpringBootApplication - https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-using-springbootapplication-annotation.html
5. Статья про аннотация @PathVariable - https://www.baeldung.com/spring-pathvariable
6. CRUD приложение на Java – https://ru.hexlet.io/courses/java-web/lessons/crud-in-db/theory\_unit
7. Первое CRUD приложение на Java — https://javarush.com/groups/posts/2253-znakomstvo-s-maven-spring-mysql-hibernate-i-pervoe-crud-prilozhenie-chastjh-1
8. PostgreSQL – https://ru.wikipedia.org/wiki/PostgreSQL
9. Знакомство с Flyway – https://habr.com/ru/companies/otus/articles/506788/
10. Введение в Spring Data JDBC – https://habr.com/ru/companies/otus/articles/531332/
11. Введение в Spring Data JPA – https://habr.com/ru/articles/435114/
12. Gson – https://habr.com/ru/companies/naumen/articles/228279/
13. Lombok – https://habr.com/ru/companies/piter/articles/676394/

**Задание**

Студенту предлагается создать свое серверное CRUD приложение. Приложение должно стать основанием для создание 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.

**Вопросы к практической работе**

1. Что такое JDBC и JPA, в чем отличия.
2. Основные интерфейсы для реализации Spring Data.
3. Аннотации @Entity, @Table. В чем разница.
4. Что такое Lombok. Приведите несколько примеров использования.
5. Назовите способы инициализации бинов в Java.
6. Что такое миграции и как они работаю в библиотеке Flyway.
7. Что такое сущность и модуль. В чем разница.