СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc157523995)

[Задание № 1 «В среде Intellij IDEA Ultimate и используя Spring Framework Java написать клиент серверное приложение с микросервисной архитектурой» 7](#_Toc157523996)

[Задание № 2 «Сервер должен принимать два аргумента и возвращать их сумму/разность» 14](#_Toc157523997)

[Задание № 3 «Сгенерировать веб-интерфейс для ваших SpringBoot API с помощью Swagger 3.0.0» 16](#_Toc157523998)

[Задание № 4 «Написать тесты на проект» 19](#_Toc157523999)

[Задание № 5 «Подключить in-memory БД и сохранять туда все результаты расчетов калькулятора» 22](#_Toc157524000)

[ЗАКЛЮЧЕНИЕ 31](#_Toc157524001)

ВВЕДЕНИЕ

В данной практической работе я буду заниматься разработкой клиент-серверного приложения на языке Java с использованием Spring Framework. Моя цель - создать микросервисную архитектуру, где сервер с применением Spring Framework, принимает два числа от клиента и возвращает их сумму или разность. Также познакомлюсь с веб-интерфейсом Swagger и тестами. Кроме того, планируется подключить внутреннюю базу данных, где будут сохраняться результаты вычислений.

Задание № 1 «В среде Intellij IDEA Ultimate и используя Spring Framework Java написать клиент серверное приложение с микросервисной архитектурой»

Начало установки программы «IntelliJ IDEA» для дальнейшей работы с кодом (Рисунок 1).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок – начало установка IntelliJ IDEA

Определение директории установки программы IntelliJ IDEA (Рисунок 2).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок – выбор папки установки

Установка опции ‘Создать ярлык на рабочем столе’, а также создания ассоциации .java (Рисунок 3).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок – конфигурация программы

Название папки, где будут располагаться файлы программы (Рисунок 4).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок – выбор названия папки

Процесс установки программы на компьютер (Рисунок 5).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок – установка программы

Конфигурация проекта через сайт <https://start.spring.io/>. Выбор проекта, языка проекта, версии Spring Bot (Рисунок 6).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – первая часть конфигурации

После выбора основных настроек проекта происходит заполнение описания проекта, а также выбор расширения и версии java (Рисунок 7).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – вторая часть конфигурации

Добавление зависимости Lombok для автоматизации методов, с целью упрощения процесса разработки и повышения читаемости кода. Также требуется добавление зависимости Spring Web для разработки веб-приложений с использованием фреймворка Spring (Рисунок 8).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок – добавление зависимостей

Генерация конфигурации проекта при нажатии на кнопку Generate (рисунок 9).

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок – генерация конфигурации проекта

Распаковка архива со сгенерированной конфигурации в папку проекта (Рисунок 10).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок – распаковка архива

Создание папки controller в директории src/main/java/ru/neoflex/practice (Рисунок 11).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок – результат метода сложения

Создание файла CalcController с расширением .java в папке controller (Рисунок 12).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок – создание файла CalcController.java

Установка SDK для обеспечения необходимых инструментов для разработки и компиляции Java-приложений (Рисунок 13).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок – установка SDK

Задание № 2 «Сервер должен принимать два аргумента и возвращать их сумму/разность»

Создание кода. Первым делом импортируются необходимые классы из библиотеки фреймворка Spring. Класс CalcController будет обрабатывать HTTP запросы. Метод ‘add’ выполняет операцию сложения указанных чисел в URL запросе. Метод принимает два числа и возвращает их сумму. Метод ‘subtract’ выполняет операцию вычитания указанных чисел в URL запросе. Метод принимает два числа и возвращает их разность (Рисунок 14).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 14 – код для калькулятора

Запуск приложения при нажатии на зелёную кнопку Run (Рисунок 15).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – запуск приложения

Для тестирования метода сложения двух чисел необходимо в адресной строки веб-браузера ввести <http://localhost:8080/add/10/7>, а для метода разности двух чисел ввести <http://localhost:8080/subtract/5/2> (Рисунок 16-17).

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок – результат метода сложения

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – результат метода разности

Задание № 3 «Сгенерировать веб-интерфейс для ваших SpringBoot API с помощью Swagger 3.0.0»

Для интеграции Swagger в проект, необходимо добавить соответствующую зависимость в файл pom.xml. В данном случае, выбрана версия 2.0.2 (Рисунок 18).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – код в pom.xml

Обновление изменений в файле pom.xml при нажатии на кнопку ‘Load Maven Changes’ (Рисунок 19).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – обновление изменений pom.xml

Для доступа к веб-интерфейсу Swagger необходимо открыть веб-браузер и перейти по адресу <http://localhost:8080/swagger-ui/index.html>. Дальше выбирается нужный метод из предоставленного списка, заполняются соответствующие поля для ввода чисел и выводится результат операции (Рисунок 20-22).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок – главный вид веб-страницы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Параллельный

Автоматически созданное описание

Рисунок 21 – результат метода разности

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок – результат метода сложения

Задание № 4 «Написать тесты на проект»

Создание пакета controller в директории src\test\java\ru.neoflex.practice (Рисунок 23)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – создание пакета controller

Создание файла CalcTest.java пакете controller (Рисунок 24).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – создание файла CalcTest.java

В файле pom.xml требуется внести зависимость на JUnit Jupiter API, предоставляющую доступ к библиотеке для создания и выполнения тестов. Выбрана версия 5.10.0 (Рисунок 25).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок – код в pom.xml

В представленном коде реализованы тесты для методов сложения и вычитания. Создается класс, содержащий тесты для методов сложения и вычитания. В каждом тесте задаются входные параметры и ожидаемые результат, затем запускаются тесты (Рисунок 26).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 26 – код для теста

Тест считается успешным, если полученный результат совпадает с ожидаемым, в ином случае тест провален. Результаты примерных тестов представлены на рисунках (Рисунок 27-28).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 27 – тест не пройден

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 28 – тесты пройдены успешно

Задание № 5 «Подключить in-memory БД и сохранять туда все результаты расчетов калькулятора»

Для сохранения расчетов калькулятора будет использоваться встроенная база данных H2, написанная на языке Java. Данная база данных довольна проста для использования и внедрения в приложение. База данных H2 предоставляет удобный веб-интерфейс, который позволяет легко и удобно взаимодействовать с базой данных. В ней можно выполнять SQL-запросы и следить за данными. Для начала в ‘ru.neoflex.practice’ необходимо создать пакеты ‘model’ и ‘repository’ (Рисунок 29).

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 29 – создание двух пакетов

Создание класса ‘CalculatingResults’ в пакете ‘model’ (Рисунок 30).

Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание

Рисунок – создание класса ‘CalculatingResults’

Создания кода, который служит для представления и хранения результатов вычислений в БД H2 с использованием Java Persistence API (JPA). В нашем случае поле ‘id’ будет использоваться как первичный ключ с автоматической генерацией значений. Более детальное и точное пояснение к каждой строчке кода представлено на рисунке кода (Рисунок 31).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок – код с комментариями

Создание класса ‘Repository’ в пакете ‘repository’ (Рисунок 32).



Рисунок – создание класса ‘Repository’

Создание кода, который создает интерфейс репозитория, предназначенного для взаимодействия с базой данных. Данный код облегчает выполнение основных операций работы с данными: сохранение, чтение, обновление и удаление (Рисунок 33).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – код для репозитория

Далее необходимо внести дополнения в код калькулятора с целью интеграции репозитория для сохранения результатов вычислений. Внесенные изменения в код позволяют не только выполнять операции сложения и вычитания через HTTP-запросы, но и сохранять полученные результаты в базе данных. Добавлена возможность получения списка всех предыдущих результатов. Таким образом, код обеспечивает взаимодействие с базой данных для хранения и извлечения результатов вычислений. Комментарии к каждой новой строчке представлены на рисунке с кодом (Рисунок 34).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок – обновленный код калькулятора

Перед началом работы с базой данных H2 следует настроить файл конфигурации ‘application.properties’. В этом файле указываются различные параметры, например порт, настройки подключения к БД и не только. В нашем случае нужно добавить только настройки подключения к БД. На рисунке с кодом представлены комментарии к каждой строчке настройки (Рисунок 35).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 35 – код в application.properties

После внесения всех настроек можно запустить приложение. Теперь база данных H2 работает в штатном режиме. Для подключения к БД нужно перейти по адресу /h2-console. Прежде, чем подключиться к БД будет сделана проверка получения списка всех результатов, которые сохраняются, пока приложение работает. Для проверки необходимо перейти по адресу <http://localhost:8080/calcList>. При этом никаких результатов на данных момент нет, так как еще не было выполнено никаких операций (Рисунок 36).

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок 36 – пустой список результатов

Для проверки будет выполнено несколько операций сложения и вычитания. После данных операций необходимо обновить страницу для получения списка предыдущих результатов (Рисунок 37).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 37 – заполненный список результатов

Проверка пройдена успешно. Теперь можно перейти к подключению БД H2 по адресу <http://localhost:8080/h2-console> (Рисунок 38).

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рисунок 38 – главный интерфейс H2

На главном интерфейсе базы данных H2 представлены поля для ввода данных, а также настройки интерфейса позволяют выбрать нужный язык интерфейса. Кроме того, можно найти справочную информацию про БД и другое. Для понимания текста будет выбран русский язык (Рисунок 39).

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 39 – выбор русского языка

Для подключения к БД необходимо ввести данные, которые были указаны в файле конфигурации (Рисунок 40).

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рисунок 40 – заполнение данных

Перед соединением к БД интерфейс предлагает возможность сделать тестовое соединение. Если данные введены верно без ошибок, то тест будет успешно пройден (Рисунок 41).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок – успешное тестовое соединение

После успешного соединения появится основной интерфейс базы данных H2, который обладаем разным полезным функционалом (Рисунок 42).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 42 – основной интерфейс БД

Для вывода списка предыдущих результатов вычислений необходимо написать следующий SQL запрос и нажать на кнопку ‘Выполнить’ (Рисунок 43).

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 43 – код запроса

После выполнения запроса будет выведен список предыдущих результатов вычислений в виде таблицы с колонками RESULT и ID (Рисунок 44).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 44 – результат выполнения запроса

ЗАКЛЮЧЕНИЕ

В ходе работы было успешно реализовано клиент-серверное приложение на языке программирования Java, которое принимает два числа от клиента и возвращает их сумму или разность, также был получен опыт работы с веб-интерфейсом Swagger для удобного взаимодействия с кодом. Были написаны тесты для надежности функционала приложения. Кроме того была выполнена интеграция внутренней базы данных H2. База данных дала возможность эффективно взаимодействовать с данными, обеспечивая легкость хранения и просмотр результатов вычислений.