

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №2 по дисциплине «Автоматизация тестирования»

Обучающийся группы 6232-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Александров А.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146881081)

[2 Результаты работы 4](#_Toc146881082)

[2.1 Результаты задания 1 4](#_Toc146881083)

[2.2 Результаты задания 2 6](#_Toc146881084)

[2.3 Результаты задания 3 11](#_Toc146881085)

[3 Вывод 14](#_Toc146881086)

[ПРИЛОЖЕНИЕ А Листинг программы 15](#_Toc146881087)

1. Постановка задачи

**Цель и постановка задания**

В процессе выполнения заданий ознакомиться с фреймворком для управления миграциями баз данных Flyway и фреймворком для интеграционного тестирования Testcontainers.

**Задание 1**

На основе первой лабораторной работы создать базу данных для хранения вычислений. Таблица должна содержать – первое число и его систему счисления, второе число и его систему счисления (числа необходимо хранить в строковом виде), дату и время вычисления, тип операции (сложение/вычитание/умножение/деление).

**Задание 2**

На основе первой лабораторной работы создать Spring Boot приложение, в котором реализовать следующие рест енд-поинты:

1. Получение результата сложения/вычитания/умножения/деления двух чисел. Числа могут быть в разных системах счисления. Все входные данные сохраняются в БД, в том числе время и дата выполнения запроса.
2. Получение данных обо всех вычислениях за определенный момент времени, с учетом операции и используемой системы счисления для каждого из чисел.
3. Все DDL операции должны быть оформлены в виде миграций (Flyway).

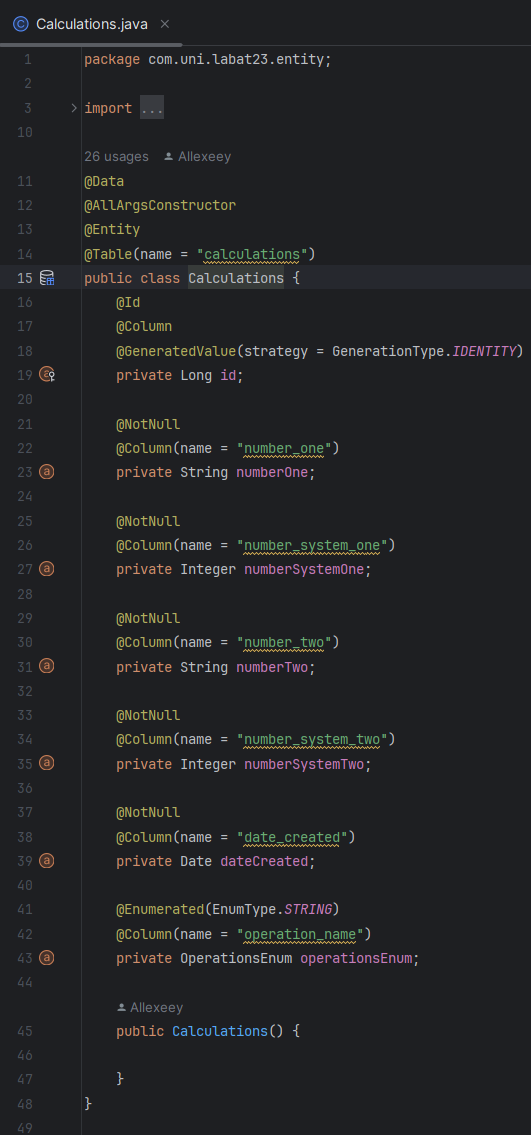
**Задание 3**

Провести интеграционное тестирование приложения с использованием фреймворка Testcontainers. Миграции с тестовыми данными (20-30 записей) должны накатываться при запуске тестов (для каждого теста отдельно или перед запуском всех тестов)

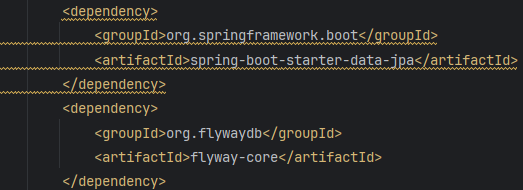
1. Результаты работы
   1. Результаты задания 1

Для работы с базой данных использовался Framework Flyway. Были описаны все сущности и силами инструментария созданы миграция, которые в дальнейшем накатываются на базу данных.

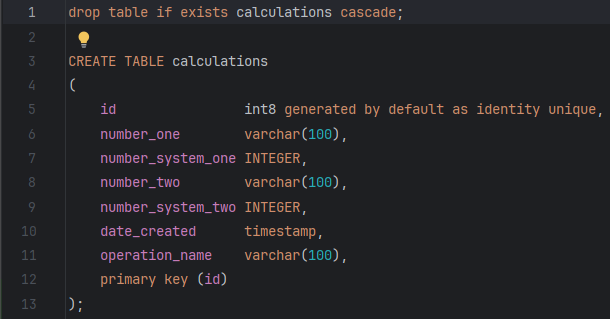
Реализованная сущность для хранения истории вычислений Entity:



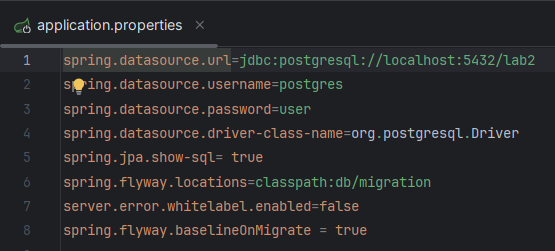
Добавление необходимых зависимостей в pom:



Код миграции создания таблиц:



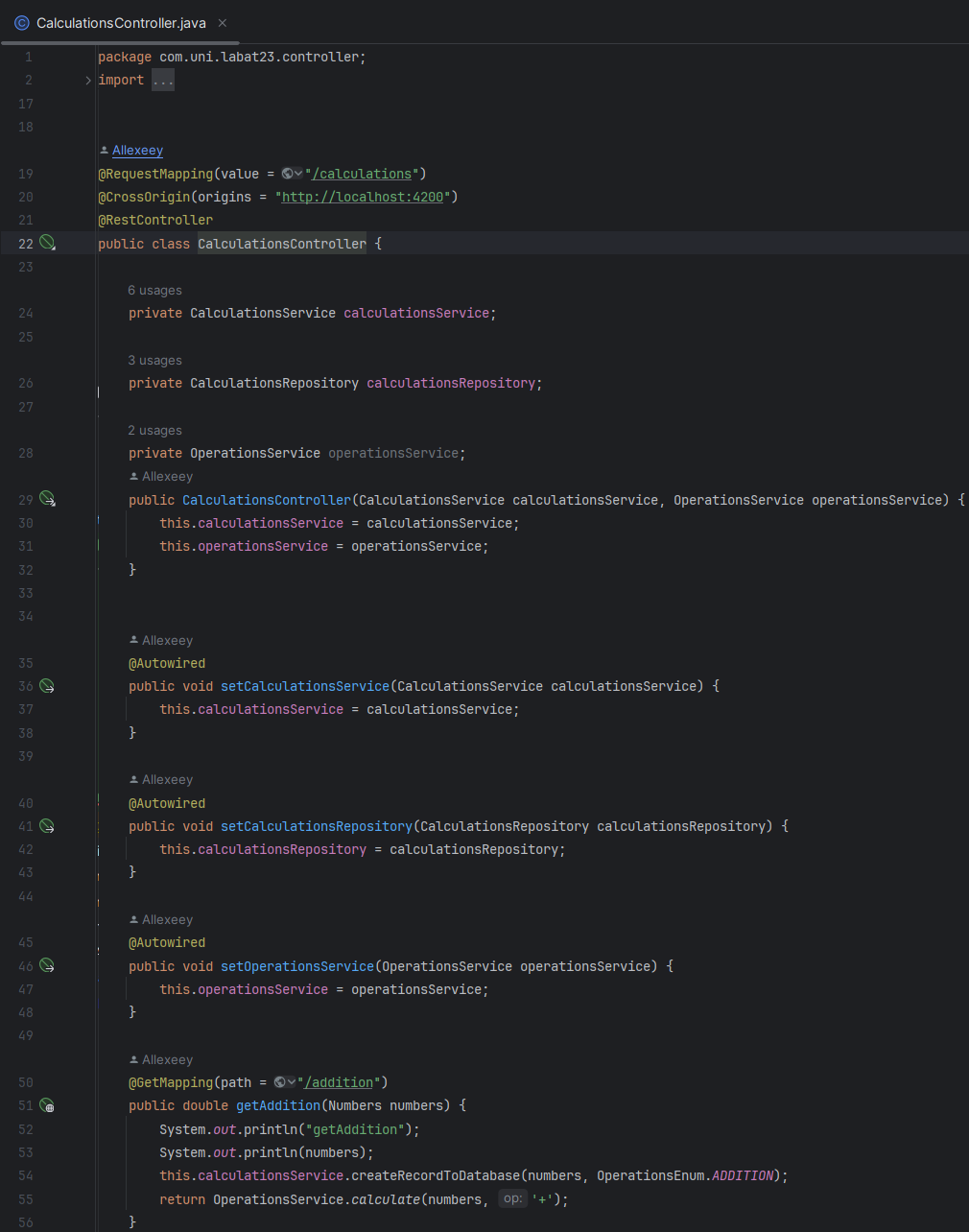
Настройки подключения к созданной базе данных:

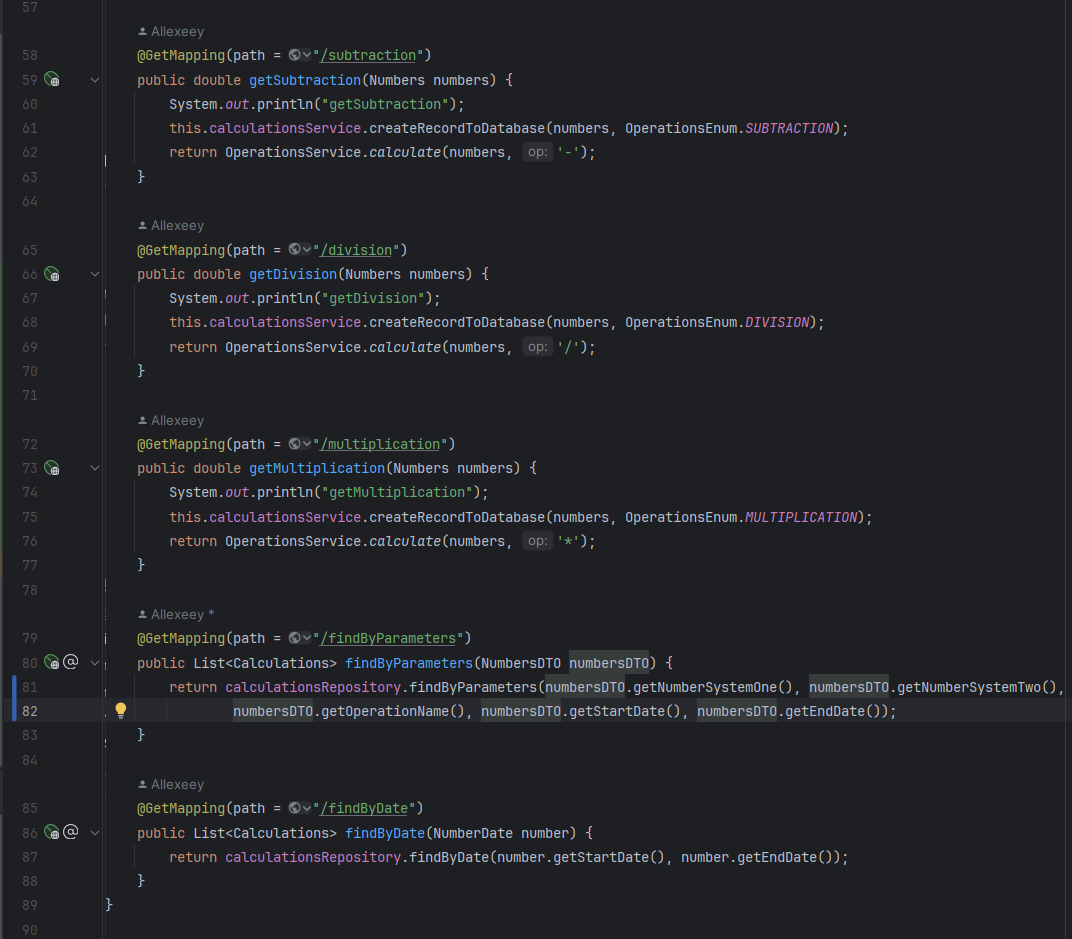


* 1. Результаты задания 2

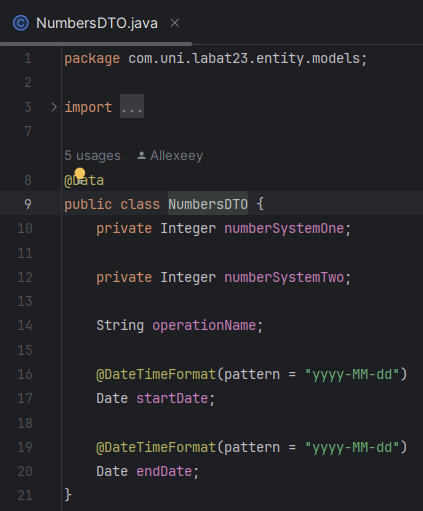
Для приложения были созданы соответствующие заданию Entity, Dto, Repository, Service и реализованы. Всё взаимодействие с пользователями описано в контроллерах.

Контроллер:

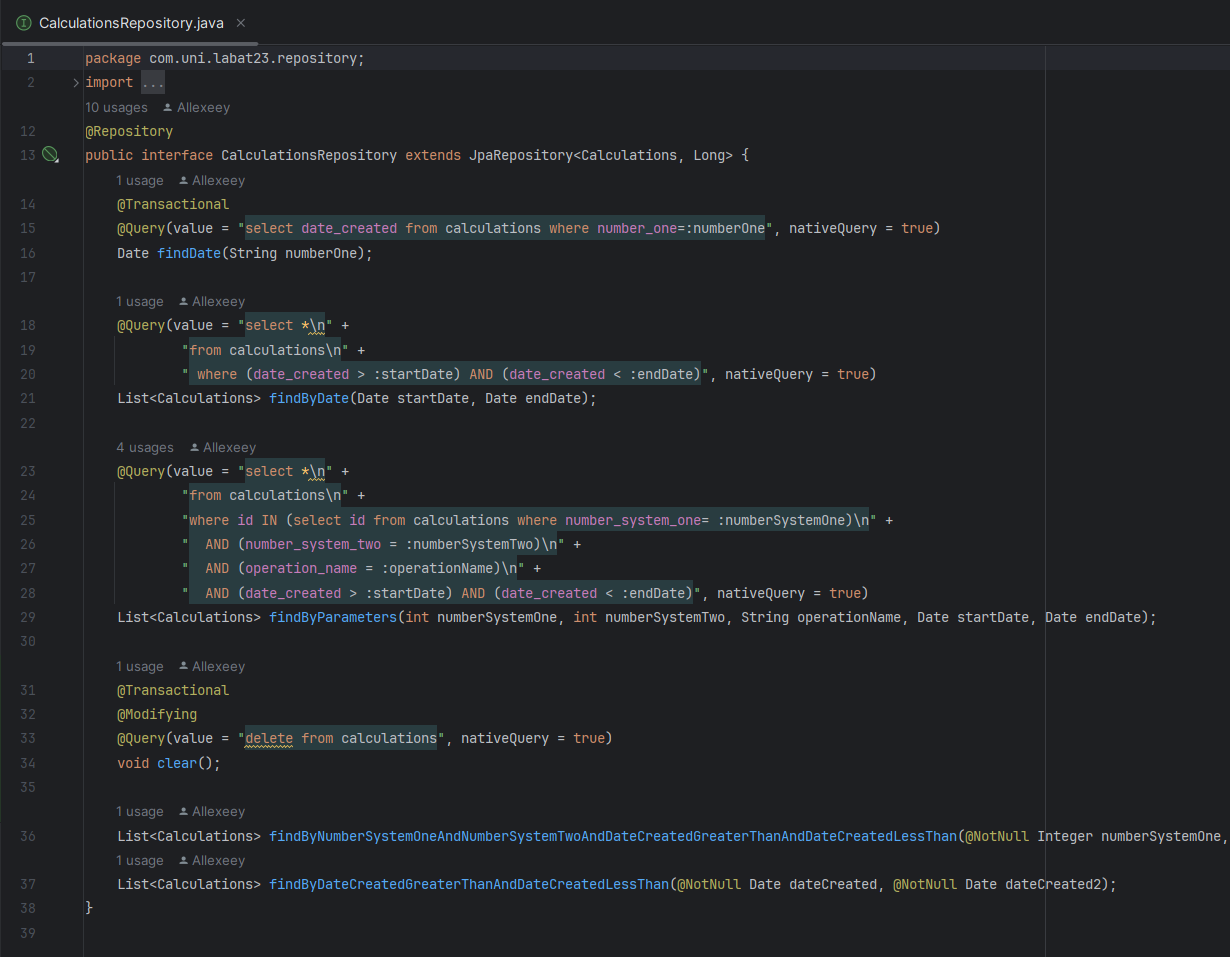




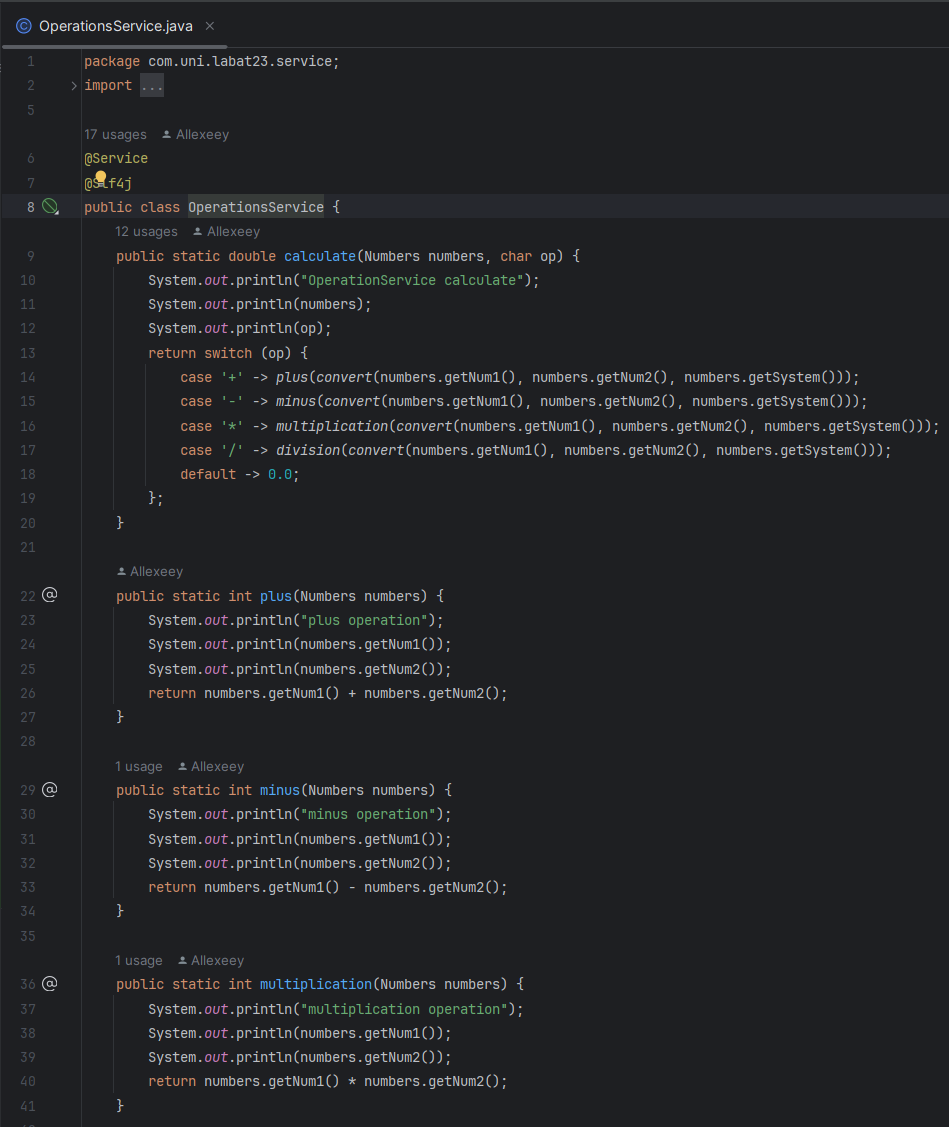
Реализация Dto:

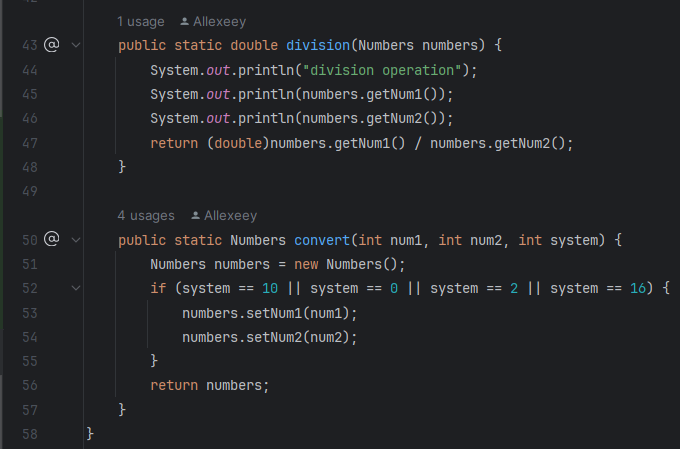


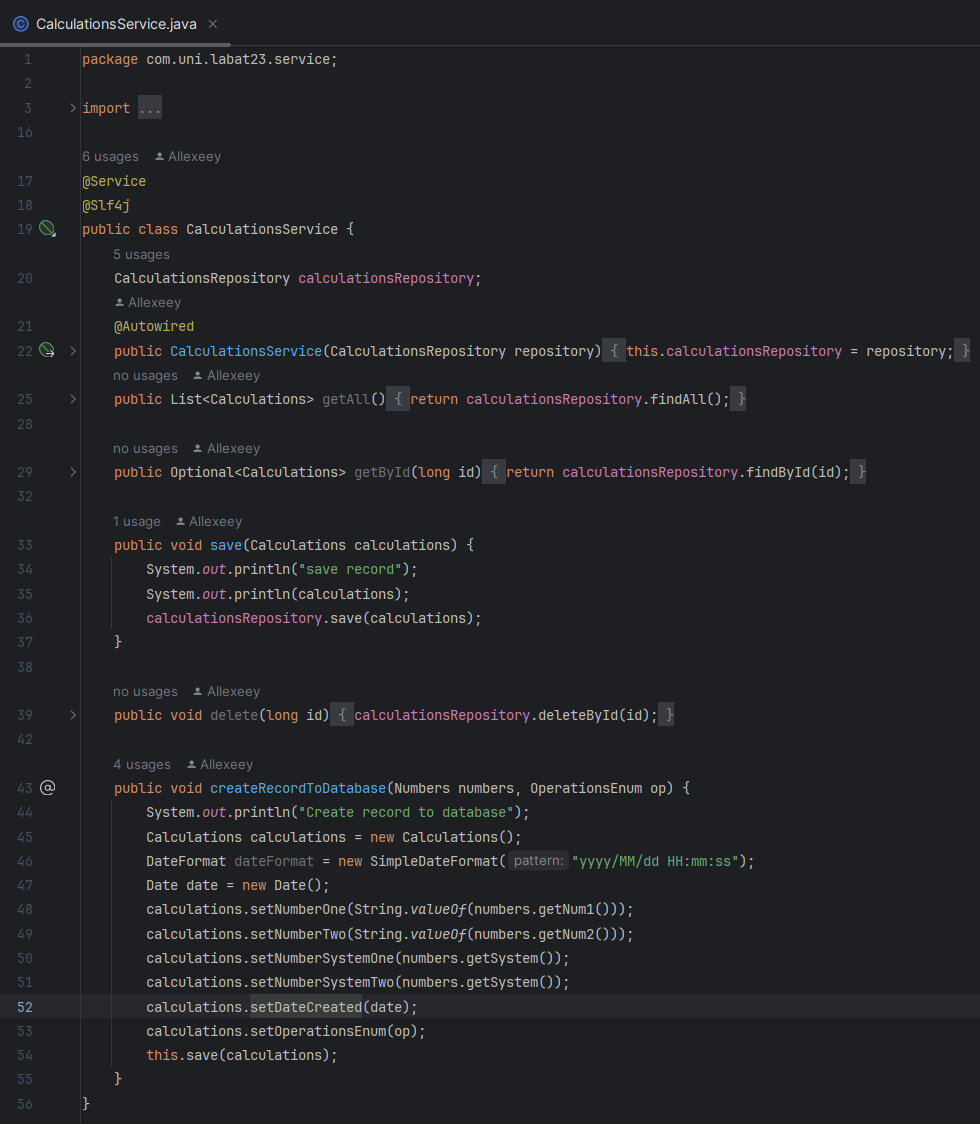
Реализация Repository:



Реализация Service:



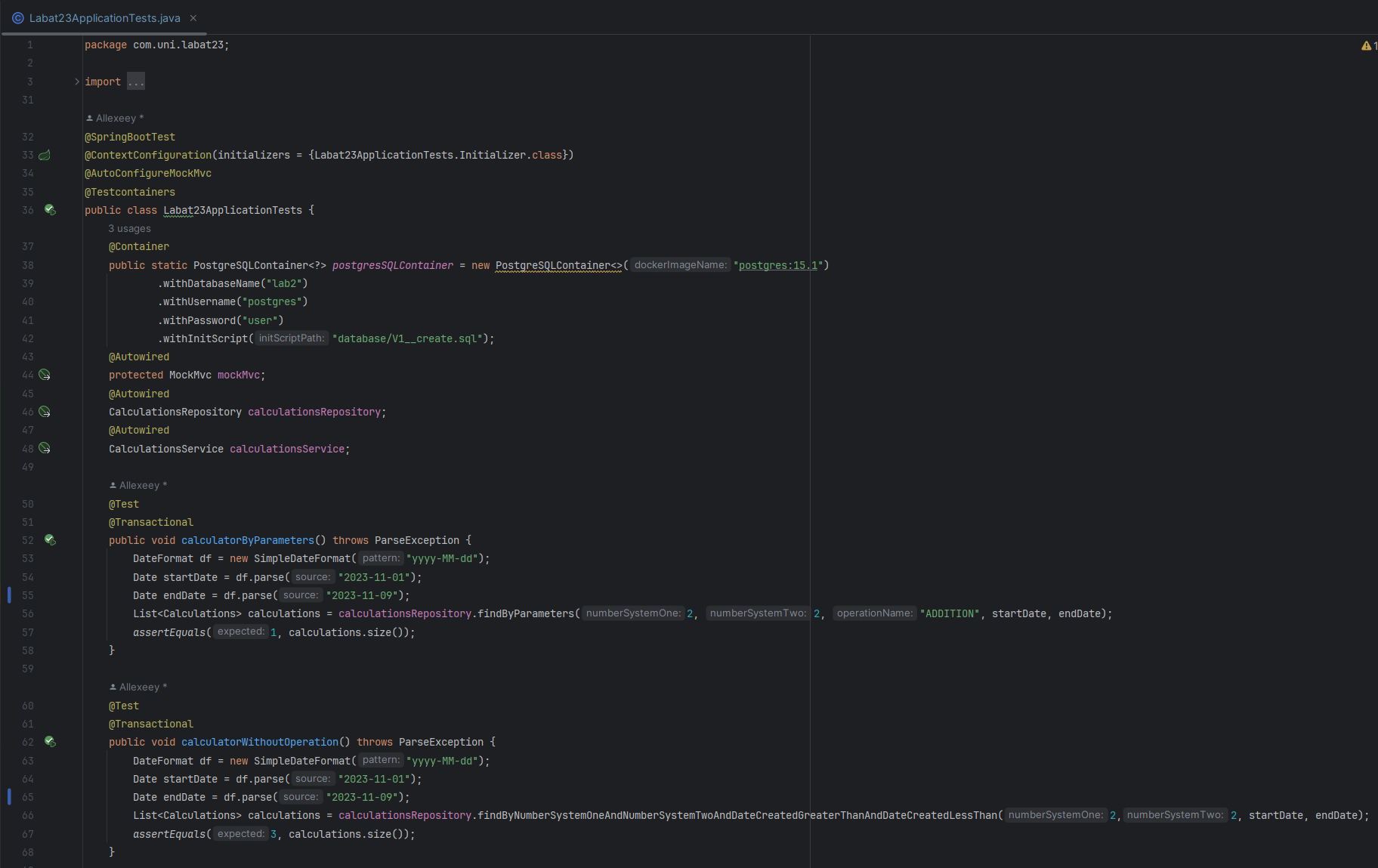




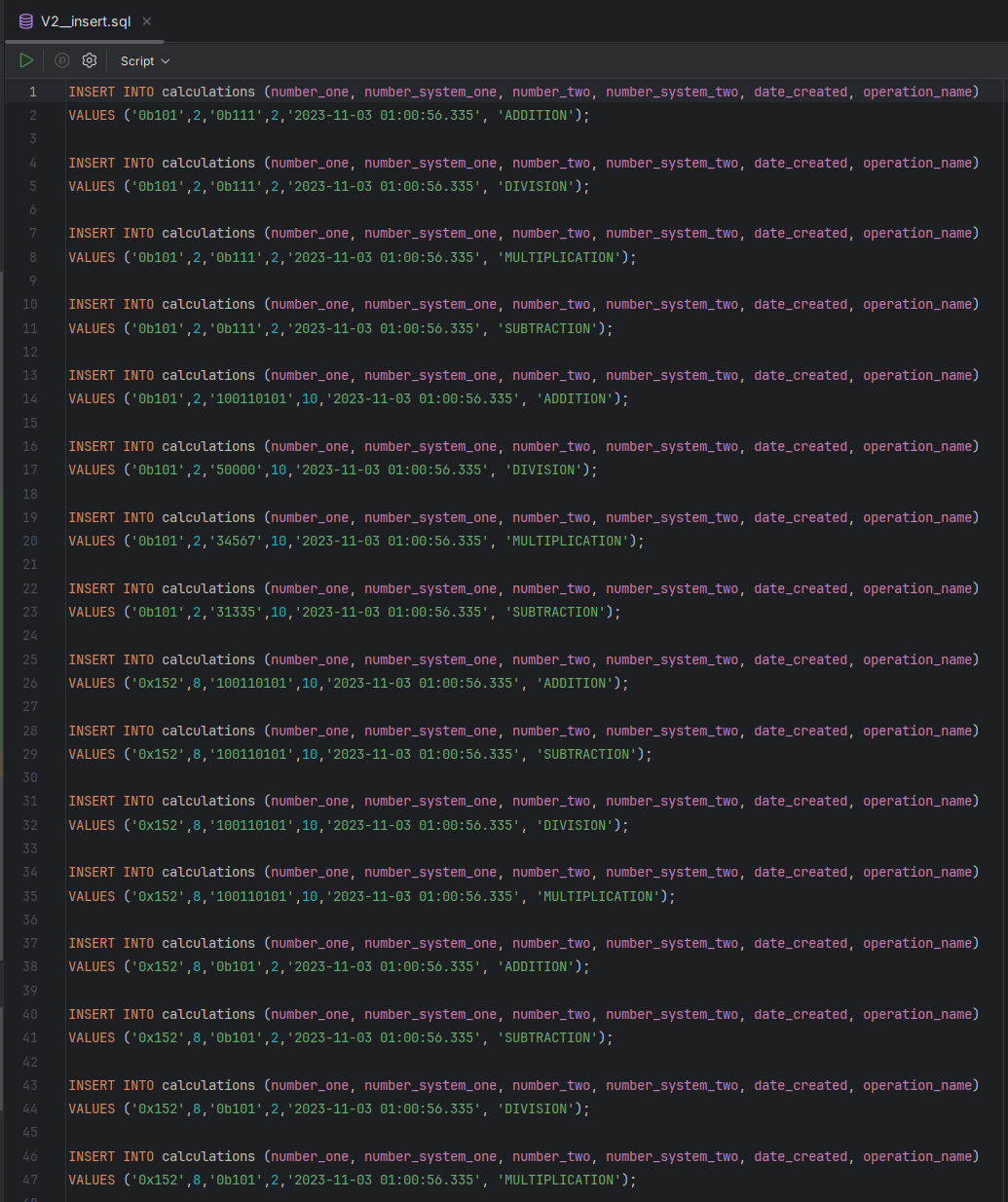
* 1. Результаты задания 3

Проведем интеграционное тестирование приложения с использованием фреймворка Testcontainers. Миграции с тестовыми данными 20 записей накатываются перед запуском всех тестов.

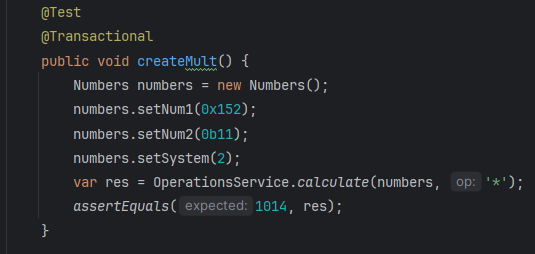
Создадим контейнер в котором регистрируется подключение к базе данных созданной посредством Testcontainers:

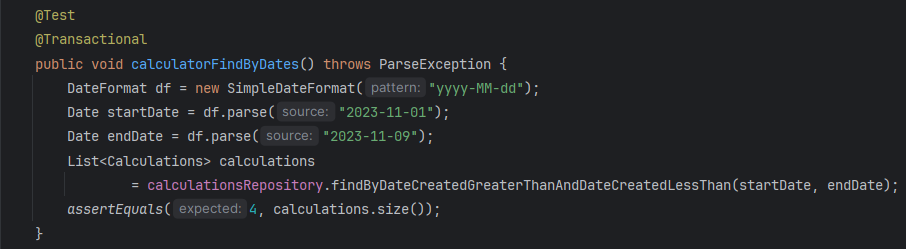


Добавляем записи в нашу таблицу:

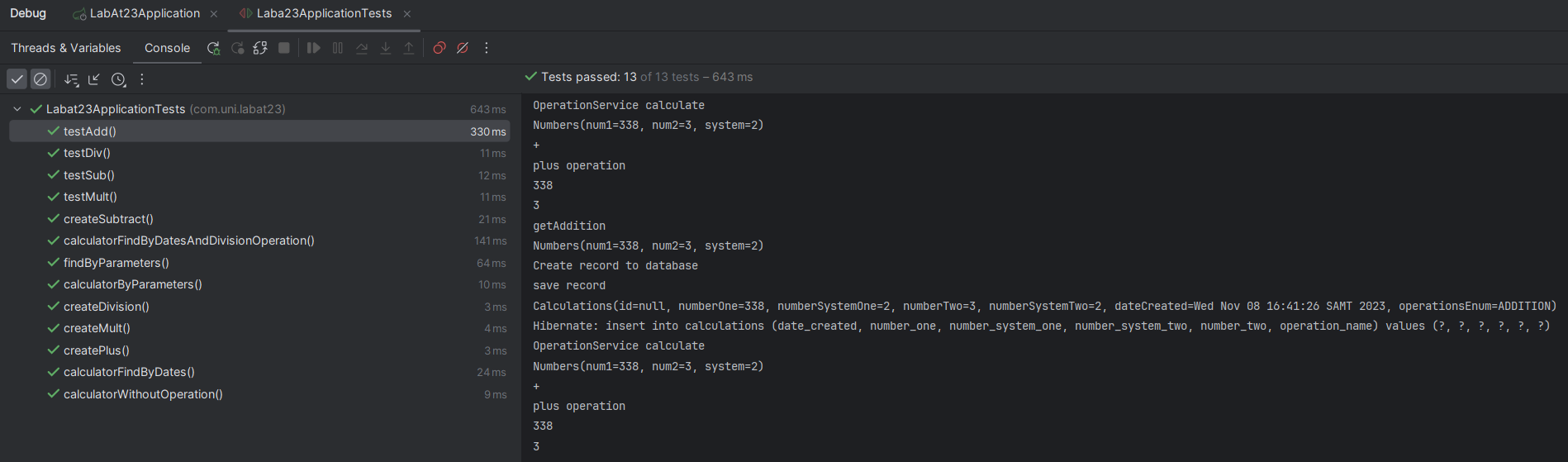


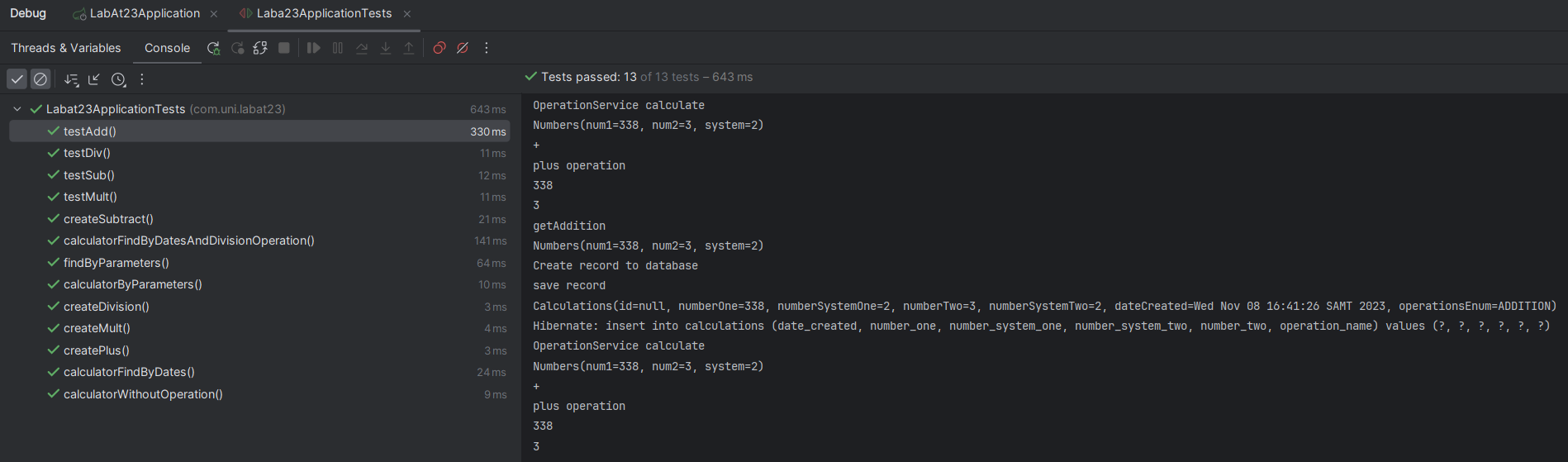
Сами тесты представляют собой обычные unit-тесты, в которых помимо корректности вычислений проверяется корректность выдаваемой истории, с разными параметрами:





Результаты тестов:





1. Вывод

В результате выполнения лабораторной работы было реализовано простое web приложение калькулятор, позволяющее по запросу производить следующие арифметические операции в 4-х системах счисления: сложение, вычитание, умножение, деление, а также выдавать историю вычислений по заданным параметрам поиска (дата, система счисления, тип операции)

Написаны и выполнены unit тесты, с динамическим окружением, включающим создаваемую базу данных в контейнере силами фреймворка Testcontainers.

Написан отчёт.

[Исходный код доступен по ссылке](https://github.com/Allexeey/autotests-lab1/tree/lab-23)

ПРИЛОЖЕНИЕ А  
Листинг программы

package com.uni.labat23;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import org.springframework.web.client.RestTemplate;

@SpringBootApplication

public class LabAt23Application {

public static void main(String[] args) {

SpringApplication.run(LabAt23Application.class, args);

}

@Bean

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

}

package com.uni.labat23.service;

import com.uni.labat23.entity.Calculations;

import com.uni.labat23.entity.models.Numbers;

import com.uni.labat23.entity.models.enums.OperationsEnum;

import com.uni.labat23.repository.CalculationsRepository;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import java.util.Optional;

@Service

@Slf4j

public class CalculationsService {

CalculationsRepository calculationsRepository;

@Autowired

public CalculationsService(CalculationsRepository repository) {

this.calculationsRepository = repository;

}

public List<Calculations> getAll() {

return calculationsRepository.findAll();

}

public Optional<Calculations> getById(long id) {

return calculationsRepository.findById(id);

}

public void save(Calculations calculations) {

System.out.println("save record");

System.out.println(calculations);

calculationsRepository.save(calculations);

}

public void delete(long id) {

calculationsRepository.deleteById(id);

}

public void createRecordToDatabase(Numbers numbers, OperationsEnum op) {

System.out.println("Create record to database");

Calculations calculations = new Calculations();

DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

Date date = new Date();

calculations.setNumberOne(String.valueOf(numbers.getNum1()));

calculations.setNumberTwo(String.valueOf(numbers.getNum2()));

calculations.setNumberSystemOne(numbers.getSystem());

calculations.setNumberSystemTwo(numbers.getSystem());

calculations.setDateCreated(date);

calculations.setOperationsEnum(op);

this.save(calculations);

}

}

package com.uni.labat23.service;

import com.uni.labat23.entity.models.Numbers;

import lombok.extern.slf4j.Slf4j;

import org.springframework.stereotype.Service;

@Service

@Slf4j

public class OperationsService {

public static double calculate(Numbers numbers, char op) {

System.out.println("OperationService calculate");

System.out.println(numbers);

System.out.println(op);

return switch (op) {

case '+' -> plus(convert(numbers.getNum1(), numbers.getNum2(), numbers.getSystem()));

case '-' -> minus(convert(numbers.getNum1(), numbers.getNum2(), numbers.getSystem()));

case '\*' -> multiplication(convert(numbers.getNum1(), numbers.getNum2(), numbers.getSystem()));

case '/' -> division(convert(numbers.getNum1(), numbers.getNum2(), numbers.getSystem()));

default -> 0.0;

};

}

public static int plus(Numbers numbers) {

System.out.println("plus operation");

System.out.println(numbers.getNum1());

System.out.println(numbers.getNum2());

return numbers.getNum1() + numbers.getNum2();

}

public static int minus(Numbers numbers) {

System.out.println("minus operation");

System.out.println(numbers.getNum1());

System.out.println(numbers.getNum2());

return numbers.getNum1() - numbers.getNum2();

}

public static int multiplication(Numbers numbers) {

System.out.println("multiplication operation");

System.out.println(numbers.getNum1());

System.out.println(numbers.getNum2());

return numbers.getNum1() \* numbers.getNum2();

}

public static double division(Numbers numbers) {

System.out.println("division operation");

System.out.println(numbers.getNum1());

System.out.println(numbers.getNum2());

return (double)numbers.getNum1() / numbers.getNum2();

}

public static Numbers convert(int num1, int num2, int system) {

Numbers numbers = new Numbers();

if (system == 10 || system == 0 || system == 2 || system == 16) {

numbers.setNum1(num1);

numbers.setNum2(num2);

}

return numbers;

}

}

package com.uni.labat23.entity;

import com.uni.labat23.entity.models.enums.OperationsEnum;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotNull;

import lombok.AllArgsConstructor;

import lombok.Data;

import java.util.Date;

@Data

@AllArgsConstructor

@Entity

@Table(name = "calculations")

public class Calculations {

@Id

@Column

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotNull

@Column(name = "number\_one")

private String numberOne;

@NotNull

@Column(name = "number\_system\_one")

private Integer numberSystemOne;

@NotNull

@Column(name = "number\_two")

private String numberTwo;

@NotNull

@Column(name = "number\_system\_two")

private Integer numberSystemTwo;

@NotNull

@Column(name = "date\_created")

private Date dateCreated;

@Enumerated(EnumType.STRING)

@Column(name = "operation\_name")

private OperationsEnum operationsEnum;

public Calculations() {

}

}

package com.uni.labat23.entity.models;

import lombok.Data;

import org.springframework.format.annotation.DateTimeFormat;

import java.util.Date;

@Data

public class NumbersDTO {

private Integer numberSystemOne;

private Integer numberSystemTwo;

String operationName;

@DateTimeFormat(pattern = "yyyy-MM-dd")

Date startDate;

@DateTimeFormat(pattern = "yyyy-MM-dd")

Date endDate;

}

package com.uni.labat23.entity.models;

import lombok.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Numbers {

private int num1;

private int num2;

private int system;

public static int toBinary(int num) {

return Integer.parseInt(String.valueOf(num), 2);

}

public static int toOctalString(int num) {

return Integer.parseInt(String.valueOf(num), 8);

}

public static int toHexString(int num) {

return Integer.parseInt(String.valueOf(num), 16);

}

}

package com.uni.labat23.entity.models;

import lombok.Data;

import org.springframework.format.annotation.DateTimeFormat;

import java.util.Date;

@Data

public class NumberDate {

@DateTimeFormat(pattern = "yyyy-MM-dd")

Date startDate;

@DateTimeFormat(pattern = "yyyy-MM-dd")

Date endDate;

}

package com.uni.labat23.entity.models.enums;

public enum OperationsEnum {

ADDITION, DIVISION, SUBTRACTION, MULTIPLICATION

}

package com.uni.labat23.controller;

import com.uni.labat23.entity.Calculations;

import com.uni.labat23.entity.models.NumberDate;

import com.uni.labat23.entity.models.Numbers;

import com.uni.labat23.entity.models.NumbersDTO;

import com.uni.labat23.entity.models.enums.OperationsEnum;

import com.uni.labat23.repository.CalculationsRepository;

import com.uni.labat23.service.CalculationsService;

import com.uni.labat23.service.OperationsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RequestMapping(value = "/calculations")

@CrossOrigin(origins = "http://localhost:4200")

@RestController

public class CalculationsController {

private CalculationsService calculationsService;

private CalculationsRepository calculationsRepository;

private OperationsService operationsService;

public CalculationsController(CalculationsService calculationsService, OperationsService operationsService) {

this.calculationsService = calculationsService;

this.operationsService = operationsService;

}

@Autowired

public void setCalculationsService(CalculationsService calculationsService) {

this.calculationsService = calculationsService;

}

@Autowired

public void setCalculationsRepository(CalculationsRepository calculationsRepository) {

this.calculationsRepository = calculationsRepository;

}

@Autowired

public void setOperationsService(OperationsService operationsService) {

this.operationsService = operationsService;

}

@GetMapping(path = "/addition")

public double getAddition(Numbers numbers) {

System.out.println("getAddition");

System.out.println(numbers);

this.calculationsService.createRecordToDatabase(numbers, OperationsEnum.ADDITION);

return OperationsService.calculate(numbers, '+');

}

@GetMapping(path = "/subtraction")

public double getSubtraction(Numbers numbers) {

System.out.println("getSubtraction");

this.calculationsService.createRecordToDatabase(numbers, OperationsEnum.SUBTRACTION);

return OperationsService.calculate(numbers, '-');

}

@GetMapping(path = "/division")

public double getDivision(Numbers numbers) {

System.out.println("getDivision");

this.calculationsService.createRecordToDatabase(numbers, OperationsEnum.DIVISION);

return OperationsService.calculate(numbers, '/');

}

@GetMapping(path = "/multiplication")

public double getMultiplication(Numbers numbers) {

System.out.println("getMultiplication");

this.calculationsService.createRecordToDatabase(numbers, OperationsEnum.MULTIPLICATION);

return OperationsService.calculate(numbers, '\*');

}

@GetMapping(path = "/findByParameters")

public List<Calculations> findByParameters(NumbersDTO numbersDTO) {

return calculationsRepository.findByParameters(numbersDTO.getNumberSystemOne(), numbersDTO.getNumberSystemTwo(), numbersDTO.getOperationName(), numbersDTO.getStartDate(), numbersDTO.getEndDate());

}

@GetMapping(path = "/findByDate")

public List<Calculations> findByDate(NumberDate number) {

return calculationsRepository.findByDate(number.getStartDate(), number.getEndDate());

}

}