

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 №3 по дисциплине «Автоматизация тестирования»

Обучающийся группы 6231-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гижевская В.Д.

Обучающийся группы 6231-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алкеев М.Г.

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

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc153152670)

[2 Результаты работы 4](#_Toc153152671)

[2.1 Результаты задания 1 4](#_Toc153152672)

[2.2 Результаты задания 2 4](#_Toc153152673)

[ПРИЛОЖЕНИЕ А Листинг программы 8](#_Toc153152674)

1. Постановка задачи

В процессе выполнения заданий познакомиться с языком Gherkin и с фреймворком Cucumber.

Задание 1

Разработать несколько user story для программы, реализованной в процессе 2 лабораторной работы

Задание 2

1. Описать созданные user story в нотации Gherkin.
2. Исключить миграции по накатыванию данных в БД. Перед запуском тестов БД должна быть пустая.
3. Реализовать интеграционное тестирование по разработанным user story с помощью фреймворка Cucumber. Перед запуском сценариев в feature файле устанавливается стартовое состояние БД (background). После прохода сценария, БД должна возвращаться в исходное состояние (для этого можно использовать хуки).
4. В user story должен присутствовать минимум по одному сценарию для передачи коллекций с помощью таблиц, с помощью класса, с помощью кастомного разделителя. Сценарий с использованием даты и просто обычный сценарии с несколькими аргументами
5. Результаты работы
   1. Результаты задания 1

Разработка User Story

1. Выполнение арифметических операций

Роль: Пользователь

Цель: Возможность выполнять арифметические операции в разных системах счисления.

Сценарий: Ввод операндов и выбор операции (сложение, вычитание, умножение, деление). Просмотр результата операции в выбранной системе счисления.

1. Поиск операций по дате

Роль: Пользователь

Цель: Получение списка операций за определенный период времени.

Сценарий: Ввод диапазона дат. Получение списка операций, выполненных в указанный период.

1. Сохранение истории операций

Роль: Пользователь

Цель: Сохранение истории выполненных операций для последующего просмотра.

Сценарий: Выполнение арифметических операций. Просмотр истории операций с указанием времени выполнения.

* 1. Результаты задания 2

Описание User Story в нотации Gherkin

User Story 1: Выполнение арифметических операций

Feature: Arithmetic Operations

Background:

Given Empty database

@calculate

Scenario: Perform arithmetic operations in different number systems

Given Operation is "add"

And Number system is "binary"

When First number is "1101"

And Second number is "1010"

Then Result is "10111"

@calculate

Scenario: Execute arithmetic operation with decimal numbers

Given Operation is "add"

And Number system is "decimal"

When Numbers are "23;12"

Then Result is "35"

@calculate

Scenario Outline: Execute division with decimal numbers

Given Operation is "divide"

And Number system is "decimal"

When First number is "<num1>"

And Second number is "<num2>"

Then Result is "<result>"

Examples:

| num1 | num2 | result |

| 30 | 3 | 10 |

| 4 | 4 | 1 |

| 180 | 60 | 3 |

| 36 | 6 | 6 |

User Story 2: поиск операций по дате

Feature: Search Operations by Date

Background:

Given Empty database

@count

Scenario: Find operations between dates with decimal numbers

Given Execute operations

| num1 | num2 | base | operation |

| 20 | 10 | decimal | divide |

| 30 | 20 | decimal | multiply |

| 40 | 30 | decimal | subtract |

| 40 | 30 | decimal | add |

And Number system is "decimal"

And Operation is "add"

When Start date is "2023-11-08"

And End date is "2023-11-08"

Then The number of operations should be "1"

User Story 3: Сохранение истории операций

Feature: Save and View Operation History

Background:

Given Empty database

@find

Scenario: View history of multiplication operations

Given Execute operations

| num1 | num2 | base | operation |

| 20 | 10 | decimal | divide |

| 30 | 20 | decimal | multiply |

| 40 | 30 | decimal | subtract |

| 40 | 30 | decimal | add |

Then Get all "multiply" operations

Исключаем миграции данных в БД путём установки базы данных в пустое состояние в блоке «Background».

Интеграционное тестирование с использованием Cucumber: для тестирования User Story 2 используются сценарии, проверяющие корректность поиска операций в указанном временном диапазоне.

Фреймворк Cucumber обеспечивает выполнение сценариев на уровне внешних запросов к программе, что позволяет проверить полную целостность функционала.

ПРИЛОЖЕНИЕ А  
Листинг программы

**SpringTest:**

package com.example.definitions;  
  
import com.example.App;  
import io.cucumber.spring.CucumberContextConfiguration;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.http.\*;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.web.util.UriComponentsBuilder;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@CucumberContextConfiguration  
@SpringBootTest(classes = App.class, webEnvironment = SpringBootTest.WebEnvironment.DEFINED\_PORT)  
public class SpringTest {  
 static HttpEntity<?> latestResponse = new HttpEntity<>(new HttpHeaders());  
  
 @Autowired  
 protected RestTemplate restTemplate = new RestTemplate();  
 static int status;  
  
 void executeGet(String url, String num1, String num2, String base1, String base2, String operation) {  
 HttpHeaders headers = new HttpHeaders();  
 headers.set(HttpHeaders.ACCEPT, MediaType.APPLICATION\_JSON\_VALUE);  
 HttpEntity<?> entity = new HttpEntity<>(headers);  
  
 String urlTemplate = UriComponentsBuilder.fromHttpUrl(url)  
 .path("/calculate/{num1}/{base1}/{num2}/{base2}/{operation}")  
 .buildAndExpand(num1, base1, num2, base2, operation)  
 .encode()  
 .toUriString();  
  
 Map<String, String> params = new HashMap<>();  
 params.put("num1", num1);  
 params.put("base1", base1);  
 params.put("num2", num2);  
 params.put("base2", base2);  
 params.put("operation", operation);  
  
 ResponseEntity<String> response = restTemplate.exchange(  
 urlTemplate,  
 HttpMethod.GET,  
 entity,  
 String.class,  
 params  
 );  
 status = response.getStatusCodeValue();  
 latestResponse = response;  
 }  
  
 void executeGetOperationsByType(String url, String operation) {  
 HttpHeaders headers = new HttpHeaders();  
 headers.set(HttpHeaders.ACCEPT, MediaType.APPLICATION\_JSON\_VALUE);  
 HttpEntity<?> entity = new HttpEntity<>(headers);  
  
 String urlTemplate = UriComponentsBuilder.fromHttpUrl(url)  
 .path("/getoperations/{operation}")  
 .buildAndExpand(operation)  
 .encode()  
 .toUriString();  
  
 Map<String, String> params = new HashMap<>();  
 params.put("operation", operation);  
  
 ResponseEntity<String> response = restTemplate.exchange(  
 urlTemplate,  
 HttpMethod.GET,  
 entity,  
 String.class,  
 params  
 );  
 status = response.getStatusCodeValue();  
 latestResponse = response;  
 }  
}

**StepsDefs:**

package com.example.definitions;  
  
import com.example.Calculator.CalculatorRepository;  
import io.cucumber.java.After;  
import io.cucumber.java.AfterAll;  
import io.cucumber.java.Before;  
import io.cucumber.java.ParameterType;  
import io.cucumber.java.en.And;  
import io.cucumber.java.en.Given;  
import io.cucumber.java.en.Then;  
import io.cucumber.java.en.When;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.convert.Delimiter;  
import static org.hamcrest.MatcherAssert.assertThat;  
import static org.hamcrest.Matchers.\*;  
import static org.assertj.core.api.Assertions.assertThat;  
  
import java.sql.Date;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Map;  
  
import static org.hamcrest.MatcherAssert.assertThat;  
import static org.hamcrest.Matchers.is;  
  
public class StepsDefs extends SpringTest {  
  
 @Autowired  
 CalculatorRepository calculatorRepository;  
  
 private String operation;  
 private String base;  
 private String num1;  
 private String num2;  
 private String startDate;  
 private String endDate;  
// private int expectedOperationCount;  
  
  
 @ParameterType("(?:.+;)+.+")  
 public List<String> stringList(final String raw) {  
 final String[] values = raw.split(";");  
 return Arrays.asList(values);  
 }  
  
 @Given("Operation is {string}")  
 public void operationIs(String operation) {  
 this.operation = operation;  
 }  
  
 @Given("Number system is {string}")  
 public void baseIs(String base) {  
 this.base = base;  
 }  
  
 @When("First number is {string}")  
 public void num1Is(String num1) {  
 this.num1 = num1;  
 }  
  
 @And("Second number is {string}")  
 public void num2Is(String num2) {  
 this.num2 = num2;  
 }  
  
 @When("Numbers are \"{stringList}\"")  
 public void numbersAre(@Delimiter(";") final List<String> numbers) {  
 this.num1 = numbers.get(0);  
 this.num2 = numbers.get(1);  
 }  
  
 @When("Start date is {string}")  
 public void startDateIs(String startDate) {  
 this.startDate = startDate;  
 }  
  
 @And("End date is {string}")  
 public void endDateIs(String endDate) {  
 this.endDate = endDate;  
 }  
  
 @Given("Execute operations")  
 public void executeOperations(List<Map<String, String>> operations) {  
 for (Map<String, String> operationData : operations) {  
 String num1 = operationData.get("num1");  
 String num2 = operationData.get("num2");  
 String base = operationData.get("base");  
 String operation = operationData.get("operation");  
  
 executeGet("http://localhost:8080",num1,num2,base,base,operation);  
 }  
 }  
  
 @Then("Result is {string}")  
 public void resultIs(String expectedResult) {  
 executeGet("http://localhost:8080",num1,num2,base,base,operation);  
 assertThat(SpringTest.latestResponse.getBody()).isEqualTo("Result: " + expectedResult);  
 }  
  
 @Then("Get all {string} operations")  
 public void getOperationsByType(String operation) {  
 executeGetOperationsByType("http://localhost:8080",operation);  
 }  
  
 @Then("The number of operations should be {string}")  
 public void countOperationsByTypeBetweenDates(String expectedResult) {  
 Date fromDate = Date.valueOf(startDate);  
 Date toDate = Date.valueOf(endDate);  
  
 String actualOperationCount = calculatorRepository.countByDateBetweenAndOperationTypeAndFirstNumSystemAndSecondNumSystem(  
 fromDate, toDate, operation, base, base  
 );  
 assertThat(actualOperationCount).isEqualTo(expectedResult);  
 }  
  
 @Given("Empty database")  
 public void prepareDatabase() {  
 calculatorRepository.clear();  
 }  
  
}

**test.feature:**

Feature: Calculations Testing  
  
 Background:  
 Given Empty database  
  
 @calculate  
 Scenario: Calculate addition with binary numbers  
 Given Operation is "add"  
 And Number system is "binary"  
 When First number is "1101"  
 And Second number is "1010"  
 Then Result is "10111"  
  
 @calculate  
 Scenario: Calculate addition with decimal numbers  
 Given Operation is "add"  
 And Number system is "decimal"  
 When Numbers are "23;12"  
 Then Result is "35"  
  
 @calculate  
 Scenario Outline: Calculate division with decimal numbers  
 Given Operation is "divide"  
 And Number system is "decimal"  
 When First number is "<num1>"  
 And Second number is "<num2>"  
 Then Result is "<result>"  
 Examples:  
 | num1 | num2 | result |  
 | 30 | 3 | 10 |  
 | 4 | 4 | 1 |  
 | 180 | 60 | 3 |  
 | 36 | 6 | 6 |  
  
 @count  
 Scenario: Find operations between dates with decimal numbers  
 Given Execute operations  
 | num1 | num2 | base | operation |  
 | 20 | 10 | decimal | divide |  
 | 30 | 20 | decimal | multiply |  
 | 40 | 30 | decimal | subtract |  
 | 40 | 30 | decimal | add |  
 And Number system is "decimal"  
 And Operation is "add"  
 When Start date is "2023-11-08"  
 And End date is "2023-11-08"  
 Then The number of operations should be "1"  
  
 @find  
 Scenario: Find multiply operations  
 Given Execute operations  
 | num1 | num2 | base | operation |  
 | 20 | 10 | decimal | divide |  
 | 30 | 20 | decimal | multiply |  
 | 40 | 30 | decimal | subtract |  
 | 40 | 30 | decimal | add |  
 Then Get all "multiply" operations