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



Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине

# «Тестирование и верификация программного обеспечения»

|  |  |
| --- | --- |
| **Выполнили:**  Студенты группы ИКБО-36-22 | Ярош В. Э.  Утенков Ю. Ю. |
| **Проверил:** | ассистент Петрова А. А. |

Москва 2024 г.

СОДЕРЖАНИЕ

[«Тестирование и верификация программного обеспечения» 1](#_Toc177491661)

[ФОРМУЛИРОВКА ЗАДАНИЯ 4](#_Toc177491662)

[ВЫПОЛНЕНИЕ ЗАДАНИЯ 6](#_Toc177491663)

[Техническое задание на разработку программного веб-калькулятора. 6](#_Toc177491664)

[1. Введение 6](#_Toc177491665)

[2. Общие сведения 6](#_Toc177491666)

[2.1 Назначение 6](#_Toc177491667)

[2.2 Краткий обзор 6](#_Toc177491668)

[3. Назначение и цели создания системы 8](#_Toc177491669)

[3.1 Назначение системы 8](#_Toc177491670)

[3.2 Цели создания системы 8](#_Toc177491671)

[4. ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ 9](#_Toc177491672)

[4.1.1 Объект автоматизации 9](#_Toc177491673)

[4.1.2 Условия эксплуатации и характеристики среды 10](#_Toc177491674)

[5. Требования к автоматизированной системе 11](#_Toc177491675)

[5.1 Функциональные требования 11](#_Toc177491676)

[5.1.1 Ввод данных: 11](#_Toc177491677)

[5.1.2 Выбор операции: 11](#_Toc177491678)

[5.1.3 Выполнение вычислений: 11](#_Toc177491679)

[5.1.4 Отображение результата: 11](#_Toc177491680)

[5.1.5 Перенаправление на форму: 12](#_Toc177491681)

[5.1.6 Обработка исключений: 12](#_Toc177491682)

[5.1.7 Многоразовое использование формы: 12](#_Toc177491683)

[5.1.8 Интерфейс: 12](#_Toc177491684)

[5.1.9 Кроссбраузерная совместимость: 12](#_Toc177491685)

[5.2 Нефункциональные требования 12](#_Toc177491686)

[5.2.1 Надежность 12](#_Toc177491687)

[5.2.2 Производительность 13](#_Toc177491688)

[6. Состав и содержание работ по созданию (развитию) системы 13](#_Toc177491689)

[7. Порядок разработки автоматизированной системы 14](#_Toc177491690)

[8. Порядок контроля и приемки автоматизированной системы 14](#_Toc177491691)

[9. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие 14](#_Toc177491692)

[10. Требования к документации 15](#_Toc177491693)

[11. Источники разработки 15](#_Toc177491694)

[12. Программный код разрабатываемого продукта 15](#_Toc177491695)

[13. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА ДРУГОГО АВТОРА 19](#_Toc177491696)

[13.1 Unit тесты другого автора. 19](#_Toc177491697)

[13.2 Ошибки в тестировании 21](#_Toc177491698)

[13.3 Исправление ошибок 21](#_Toc177491699)

[13.4 Итоговое тестирование. 23](#_Toc177491700)

[ЗАКЛЮЧЕНИЕ 24](#_Toc177491701)

# ФОРМУЛИРОВКА ЗАДАНИЯ

**Этап №1. Создание простого модуля программы.**

Каждый из участников команды создаёт свой простой модуль программы (с минимальным количеством функций равной 5 и при желании более) на любом языке программирования, с которым знакомы всё члены команды. В одну из функций закладывается ошибка.

Например, это может быть:

* Калькулятор (с функциями сложения, вычитания, умножения и деления).
* Программа для работы с массивами (поиск максимального значения, сортировка массива).
* Программа для работы со строками (проверка палиндромов, подсчет символов)

**Этап №2. Разработка документации модуля программы.**

Каждый участник пишет документацию по своему выбранному программному продукту и передает ее следующему участнику.

В документации должно быть полное описание функционала приведенного кода, описание работы, выбранного для тестирования программного продукта или отдельно взятого кода.

**Этап №3. Тестирование ПО.**

Участники внутри своей группы производят обмен программными продуктами по кругу. Для полученного ПО пишутся Unit-тесты для каждой из функций в программе. Цель тестирования найти заложенную ошибку.

Убедитесь, что у вас установлены необходимые библиотеки (например, pytest для Python, JUnit для Java).

**Этап №4. Исправление ошибки.**

После проведения тестирования и нахождения ошибки в коде ПО участник должен оформить её по следующему шаблону:

***Краткое описание ошибки:*** «Неверное преобразование заглавных букв в строчные».

***Статус ошибки:*** открыта («Open»).

***Категория ошибки:*** серьезная («Major»).

***Тестовый случай:*** «Проверка алгоритма функционирования программы».

***Описание ошибки:***

1. Загрузить программу.

2. В поле ввода ввести строку «ABCD».

3. Нажать кнопку "Пуск".

4. Полученный результат: «ABCD».

***Ожидаемый результат:*** «abcd».

(Категория ошибки может принимать следующие значения: **блокирующие («Blocker»), критические («Critical»), серьезные («Major»), незначительные («Minor»), тривиальные («Trivial»)**)

Созданная документация на ошибку передаётся обратно разработчику ПО для её исправления.

**Этап №5. Итоговое тестирование.**

После исправления ошибки, ПО возвращается на повторную проверку. Участник команды, проводящий тестирование, должен удостовериться, что все ранее написанные тесты проходят успешно и что раннее найденная ошибка исправлена.

**Этап №6: Отчет**

По итогам проделанной работы формируется отчёт, включающий в себя подробное описание всех этапов с 1 по 5 со скриншотами проведения тестирования. В приложении к отчету участник прикрепляет код своего ПО и Unit-тестов для тестируемого ПО.

# ВЫПОЛНЕНИЕ ЗАДАНИЯ

Техническое задание на разработку программного веб-калькулятора.

1. Введение

Тестирование программного обеспечения охватывает множество аспектов, включая функциональное, нагрузочное и регрессионное тестирование. Каждый из этих типов тестирования имеет свои цели и методы, что позволяет выявить различные виды дефектов на разных этапах разработки. Например, функциональное тестирование направлено на проверку корректности работы всех функций программы и её соответствие заданным требованиям, в то время как нагрузочное тестирование помогает оценить производительность системы при высоких нагрузках. Современные инструменты автоматизации тестирования значительно повышают эффективность процесса проверки. Использование автоматизированных тестов позволяет не только ускорить выявление ошибок, но и снизить вероятность человеческого фактора, который может привести к пропускам критически важных дефектов. Однако, несмотря на все преимущества автоматизации, ручное тестирование по-прежнему остается важной частью QA-процесса, особенно в случаях, когда необходима интуитивная оценка пользовательского опыта.

1. Общие сведения
2. Назначение

Программное приложение веб-калькулятор.

* 1. Краткий обзор

Веб-калькулятор — это программное приложение, реализованное на Kotlin, которое предоставляет функциональность базовых математических операций. Основные возможности:

1. Визуальный интерфейс, позволяющий пользователю легко взаимодействовать с калькулятором через веб-браузер.
2. Поддержка следующих основных математических операций:
   * Сложение
   * Вычитание
   * Умножение
   * Деление
3. Простой и интуитивно понятный интерфейс, где пользователь может вводить числа и выбирать операции для выполнения расчетов.
4. Результаты расчетов отображаются сразу после нажатия кнопки "Вычислить".
5. Приложение построено на современных технологиях Spring Boot и архитектуре MVC и использует лучшие практики разработки, обеспечивая высокую производительность и надежность работы.
6. Код написан на Kotlin, что позволяет эффективно использовать возможности этого языка программирования.
7. Архитектура приложения ориентирована на масштабируемость и легкость поддержки.
   1. **Разработчик**

Состав команды: Утенков Ю. Ю.

* 1. **Заказчик**

Ярош В. Э.

* 1. **Основание для разработки**

Договор № 123.45 от 16.09.2024 на разработку автоматизированной системы.

1. Назначение и цели создания системы
   1. Назначение системы

Веб-калькулятор предназначен для выполнения базовых арифметических операций (сложение, вычитание, умножение, деление) через веб-интерфейс. Система предоставляет пользователю возможность ввести два числа и выбрать нужную операцию для расчета. Результаты вычислений отображаются в браузере с использованием HTML-шаблонов. Основное назначение системы — предоставить простой, удобный и интуитивно понятный интерфейс для выполнения математических вычислений, который доступен из любого браузера без необходимости установки дополнительного программного обеспечения.

3.2 Цели создания системы

1. Система позволяет пользователям быстро производить базовые математические вычисления через удобный веб-интерфейс, что может быть полезно в повседневной жизни и работе.
2. Проект разработан как пример использования языка программирования Kotlin и фреймворка Spring Boot для создания простого веб-приложения с поддержкой MVC (Model-View-Controller). Он демонстрирует архитектурные принципы построения веб-приложений с разделением логики и представления.
3. Веб-калькулятор спроектирован так, чтобы легко добавлять новые функции и операции в будущем. Код организован таким образом, чтобы каждая арифметическая операция была отдельной функцией, что упрощает поддержку и добавление нового функционала.
4. Система предусматривает модульное тестирование основных функций для обеспечения надежной работы. Это позволяет убедиться, что каждая арифметическая операция (сложение, вычитание, умножение, деление) работает корректно при любых входных данных.
5. Веб-приложение должно быть доступно на любых устройствах, поддерживающих современные браузеры (ПК, ноутбуки, планшеты, смартфоны), обеспечивая одинаковый пользовательский опыт на всех платформах.
6. Система должна корректно обрабатывать исключительные ситуации, такие как деление на ноль, и предоставлять пользователю понятные сообщения об ошибках или неординарном результате выполнение ошибочных операций, обеспечивая дружелюбный пользовательский интерфейс.
7. ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ
   * 1. Объект автоматизации

Объектом автоматизации для данного веб-калькулятора является процесс выполнения базовых арифметических операций (сложение, вычитание, умножение, деление) через веб-интерфейс. Система автоматизирует ввод и обработку числовых данных, выполнение математических операций на сервере и отображение результатов пользователям в браузере.

Автоматизация также охватывает следующие аспекты:

* Формирование и отправка запросов с числовыми параметрами через веб-форму.
* Выполнение выбранной арифметической операции на серверной стороне.
* Обработка ошибок и исключительных ситуаций, таких как деление на ноль.
* Отображение результата вычислений в удобном и понятном виде на веб-странице.
* Поддержка многоразового использования без перезагрузки страницы для выполнения новых расчетов.

Приложение веб-калькулятор позволяет конечным пользователям использовать базовые арифметические операции в удобном графическом интерфейсе браузера.

* + 1. Условия эксплуатации и характеристики среды

Программное обеспечение веб-калькулятора будет эксплуатироваться в условиях стандартной пользовательской среды с доступом к интернету. Пользователи взаимодействуют с системой через браузер на различных устройствах, таких как компьютеры, ноутбуки, планшеты или смартфоны.

Система разработана на основе технологий Kotlin и Spring Boot и не требует постоянного хранения данных, так как выполняет расчеты в реальном времени и не сохраняет результаты. Для корректной работы веб-приложения необходим сервер, поддерживающий запуск Java-приложений.

Условия эксплуатации:

* Для доступа к веб-интерфейсу и выполнения расчетов необходимо стабильное интернет-соединение.
* Совместимость с современными браузерами (Google Chrome, Mozilla Firefox, Microsoft Edge и др.), поддерживающими стандартные веб-технологии (HTML5, CSS, JavaScript).
* Поддержка Java-платформы, достаточная для работы Spring Boot приложений. Минимальные требования включают сервер с установленной Java Runtime Environment (JRE) и контейнером для веб-приложений (например, Tomcat или встроенный сервер Spring Boot).
* Серверное программное обеспечение совместимо с операционными системами, поддерживающими запуск Java (Windows, Linux, macOS).

Среда эксплуатации не имеет особых требований к аппаратным ресурсам, поскольку калькулятор выполняет простые арифметические вычисления.

1. Требования к автоматизированной системе
   1. Функциональные требования
      1. Ввод данных:

* Пользователь должен иметь возможность ввести два числовых значения (аргументы) через веб-форму.
* Система должна проверять корректность введённых данных (только числовые значения).
  + 1. Выбор операции:
* Пользователь должен иметь возможность выбрать одну из четырёх арифметических операций: сложение, вычитание, умножение, деление.
* Каждая операция должна выполняться через отдельный URL-адрес: /addition, /subtraction, /multiplication, /division.
  + 1. Выполнение вычислений:
* При отправке формы система должна выполнить выбранную арифметическую операцию на сервере.
* В случае деления на ноль система должна обрабатывать исключение и выводить сообщение об ошибке или нечисловой результат выполнения операции.
  + 1. Отображение результата:
* Результат вычисления должен быть отображён на веб-странице.
* В случае успешного выполнения арифметической операции результат должен быть показан в виде: "Результат сложения: 10.0".
* В случае ошибки, например, деления на ноль, система должна выводить сообщение: "infinity". По сути оповещая, что результатом операции является нечисловой формат.
  + 1. Перенаправление на форму:
* После выполнения операции пользователь должен быть перенаправлен обратно на страницу с формой для ввода новых данных без необходимости обновления всей страницы.
  + 1. Обработка исключений:
* Если один из параметров не был указан, система должна выводить сообщение об ошибке: "Пожалуйста, введите оба числа."
* При возникновении любой другой ошибки система должна корректно отображать соответствующее сообщение.
  + 1. Многоразовое использование формы:
* Пользователь должен иметь возможность выполнять несколько операций подряд, не обновляя страницу и не перезагружая приложение.
  + 1. Интерфейс:
* Веб-интерфейс должен быть простым и интуитивно понятным. Форма для ввода данных и кнопки для выбора операций должны быть видны и легко доступны.
* Результат вычислений должен отображаться в том же окне, где происходит ввод данных.
  + 1. Кроссбраузерная совместимость:
* Приложение должно корректно работать во всех современных браузерах (Google Chrome, Firefox, Safari, Microsoft Edge и др.).
  1. Нефункциональные требования
     1. Надежность

Система должна обрабатывать все ошибки, возникающие при вычислениях или вводе данных, корректно отображая сообщения пользователю.

В случае сбоя серверной части или недоступности сервера пользователь должен видеть соответствующее сообщение.

* + 1. Производительность

Время ответа сервера на запросы не должно превышать 1 секунды для вычислений любой сложности.

Приложение должно обрабатывать не менее 100 одновременных запросов без снижения производительности.

1. Состав и содержание работ по созданию (развитию) системы

* Разработка технического задания:
* Содержание: Определение требований, создание и согласование технического задания.
* Результат: Утвержденное техническое задание.
* Проектирование системы:
* Содержание: Разработка архитектуры системы, проектирование базы данных и интерфейсов.
* Результат: Документация по архитектуре, проект базы данных, макеты интерфейсов.
* Разработка программного обеспечения:
* Содержание: Написание и тестирование кода.
* Результат: Рабочее программное обеспечение.
* Интеграция и тестирование:
* Содержание: Интеграция модулей, функциональное и нагрузочное тестирование.
* Результат: Протестированная система.
* Внедрение системы:
  + Содержание: Установка, настройка, обучение пользователей.
  + Результат: Рабочая система и обученные пользователи.
* Поддержка и сопровождение:
* Содержание: Техническая поддержка, обновления.
* Результат: Стабильная работающая система.

1. Порядок разработки автоматизированной системы

Процесс создания системы проходит через несколько ключевых фаз: анализ требований, проектирование, программирование, тестирование и внедрение. Перед началом каждого нового этапа требуется получение одобрения от заказчика. Важным аспектом являются промежуточные проверки для обеспечения соответствия системы установленным требованиям.

План разработки согласуется с ассистентом кафедры МОСИТ Петровой А.А.

Необходимые этапы при разработки автоматизированной системы: аналитический, проектный, программный, тестирование и внедрение.

1. Порядок контроля и приемки автоматизированной системы

Контроль и приемка системы включают функциональные, интеграционные и нагрузочные испытания, проводимые в соответствии с действующими стандартами. Испытания охватывают проверку всех компонентов системы на соответствие требованиям технического задания. Приемка выполняется поэтапно: документация проверяется и утверждается в установленные сроки, а участие в приемке принимает заказчик, разработчик и, при необходимости, сторонние эксперты. Приемочная комиссия, состоящая из представителей заказчика и разработчика, утверждает результаты и принимает решение о готовности системы к эксплуатации.

1. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие

Для подготовки объекта автоматизации к внедрению системы необходимо выполнить ряд ключевых мероприятий. Это включает в себя: подготовку информации для обработки в системе, внесение изменений в объект автоматизации, обеспечение соответствующих условий для функционирования системы, создание необходимых служб и подразделений, а также организацию обучения персонала. Эти мероприятия должны быть согласованы с заказчиком и выполнены до начала эксплуатации системы, а их детали уточняются на стадии разработки и опытной эксплуатации.

1. Требования к документации

Для автоматизированной системы требуется разработка и поддержка следующих документов: техническое задание, проектная документация, руководство пользователя и эксплуатационная документация. Все документы должны соответствовать стандартам ГОСТ и внутренним нормативам. Документация будет предоставляться как в бумажном, так и в электронном виде в программном продукте непосредственно.

1. Источники разработки

* Договор № 123.45 от 16.09.2024;
* ГОСТ 34.602 – 2020 "Техническое задание на создание автоматизированной системы".

### Программный код разрабатываемого продукта

Ниже приведён листинг разрабатываемой программы с соответствующей документацией по каждому методу (модулю функционирования программы). Листинг программного кода представлен на языке программирования Kotlin. (листинг 1).

|  |
| --- |
| package ru.example.webcalculator  import org.springframework.stereotype.Controller import org.springframework.ui.Model import org.springframework.web.bind.annotation.GetMapping import org.springframework.web.bind.annotation.RequestMapping import org.springframework.web.bind.annotation.RequestParam import org.springframework.web.servlet.mvc.support.RedirectAttributes  @Controller @RequestMapping("/calculator") class CalculatorController {  */\*\*  \* Метод для возвращения html страницы с формой для заполнения  \* @return HTML-шаблон, который отображает дефолтную форму.  \*  \*/* @GetMapping  fun form(): String  {  return "calculate"  }   */\*\*  \* Метод для обработки конкретной арифметической операции.  \*  \* @param a Первое число для сложения.  \* @param b Второе число для сложения.  \* @param operation Модель для передачи данных в представление.  \* @param redirectAttributes Объект Spring Framework для выполнения редиректа  \* @return Строка с редиректом на конкретный URL для выполнения определённой операции.  \*/* @GetMapping("/calculate")  fun calculate(  @RequestParam(value = "a", required = false) a: Double,  @RequestParam(value = "b", required = false) b: Double,  @RequestParam(value = "operation") operation: String,  redirectAttributes: RedirectAttributes  ): String  {  redirectAttributes.addAttribute("a", a)  redirectAttributes.addAttribute("b", b)  return when (operation)  {  "addition" -> "redirect:/calculator/addition"  "subtraction" -> "redirect:/calculator/subtraction"  "multiplication" -> "redirect:/calculator/multiplication"  "division" -> "redirect:/calculator/division"  else -> "redirect:/calculator"  }  }   */\*\*  \* Метод для обработки операции сложения.  \*  \* @param a Первое число для сложения.  \* @param b Второе число для сложения.  \* @param model Модель для передачи данных в представление.  \* @return Имя HTML-шаблона, который отображает результат.  \*  \* Пример запроса:  \* GET /calculator/addition?a=5&b=3  \*  \* Пример ответа: Страница с текстом "Результат операции: 8.0"  \*/* @GetMapping("/addition")  fun addition(@RequestParam(value = "a", required = false) a: Double,  @RequestParam(value = "b", required = false) b: Double, model: Model): String  {  model.addAttribute("result", "Результат операции: ${a + b}")  return "result"  }   */\*\*  \* Метод для обработки операции вычитания.  \*  \* @param a Уменьшаемое.  \* @param b Вычитаемое.  \* @param model Модель для передачи данных в представление.  \* @return Имя HTML-шаблона, который отображает результат.  \*  \* Пример запроса:  \* GET /calculator/subtraction?a=10&b=4  \*  \* Пример ответа: Страница с текстом "Результат операции: 6.0"  \*/* @GetMapping("/subtraction")  fun subtraction(@RequestParam(value = "a", required = false) a: Double,  @RequestParam(value = "b", required = false) b: Double, model: Model): String  {  model.addAttribute("result", "Результат операции: ${a - b}")  return "result"  }   */\*\*  \* Метод для обработки операции умножения.  \*  \* @param a Первый множитель.  \* @param b Второй множитель.  \* @param model Модель для передачи данных в представление.  \* @return Имя HTML-шаблона, который отображает результат.  \*  \* Пример запроса:  \* GET /calculator/multiplication?a=7&b=3  \*  \* Пример ответа: Страница с текстом "Результат операции: 21.0"  \*/* @GetMapping("/multiplication")  fun multiplication(@RequestParam(value = "a", required = false) a: Double,  @RequestParam(value = "b", required = false) b: Double, model: Model): String  {  model.addAttribute("result", "Результат операции: ${a \* b}")  return "result"  }   */\*\*  \* Метод для обработки операции деления.  \*  \* @param a Делимое.  \* @param b Делитель.  \* @param model Модель для передачи данных в представление.  \* @return Имя HTML-шаблона, который отображает результат.  \*  \* Пример запроса:  \* GET /calculator/division?a=12&b=4  \*  \* Пример ответа: Страница с текстом "Результат операции: 3.0"  \*  \* Особая обработка деления на ноль. В случае попытки деления на ноль,  \* возвращается сообщение об ошибке.  \*/* @GetMapping("/division")  fun division(@RequestParam(value = "a", required = false) a: Double,  @RequestParam(value = "b", required = false) b: Double, model: Model): String  {  model.addAttribute("result", "Результат операции: ${a / b}")  return "result"  } } |

Листинг 1 – Код программы, ответственный за выполнение арифметических операций.

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА ДРУГОГО АВТОРА

13.1 Unit тесты другого автора.

Все юнит-тесты написаны, их описание представлено на рисунках 1-4.

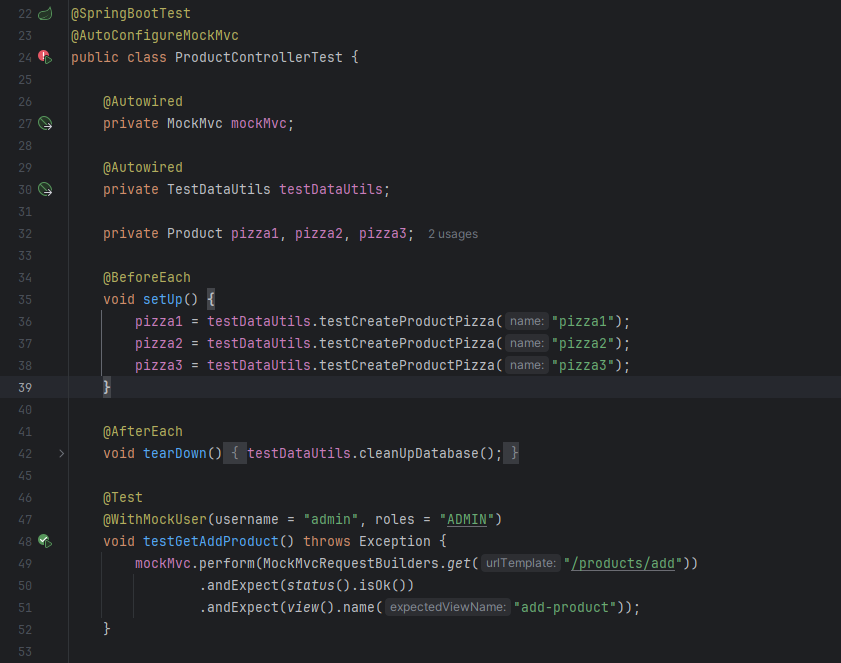


Рисунок 1 – методы unit теста, часть 1

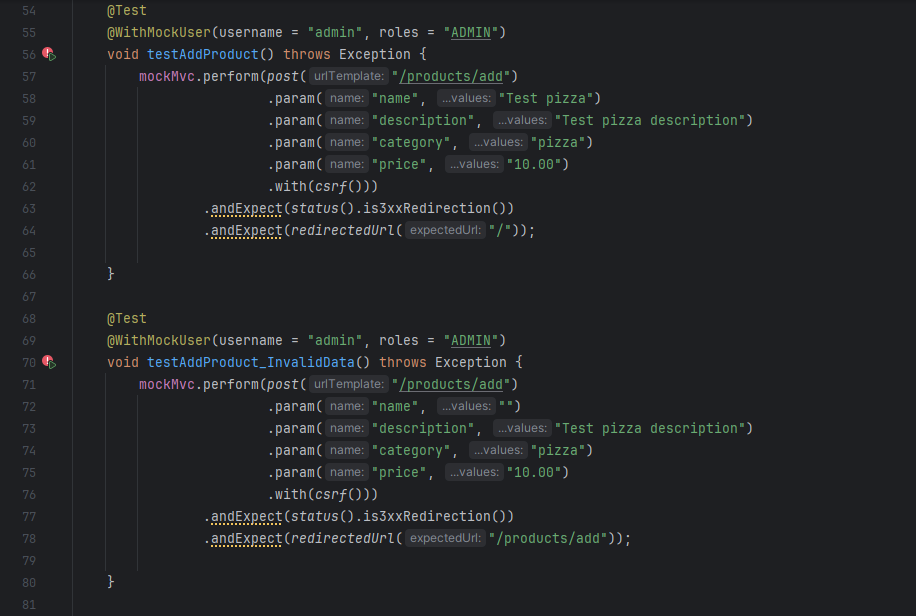


Рисунок 2 – методы unit теста, часть 2



Рисунок 3 – методы unit теста, часть 3



Рисунок 4 – методы unit теста, часть 4

13.2 Ошибки в тестировании

В ходе проведения unit тестов были выявлены 2 ошибки в работе программы (рисунок 5).

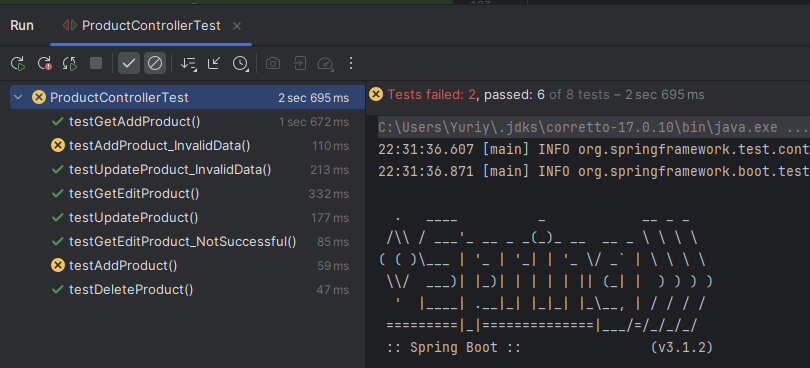


Рисунок 5 – Выполнение unit тестов

13.3 Исправление ошибок

**Ошибка №1:**

1. Краткое описание ошибки: Неверный URL-адрес в методе testAddProduct\_invalidData() для добавления продукта.
2. Статус ошибки: открыта («Open»). Категория ошибки: серьезная («Major»).
3. Тестовый случай: Проверка корректности URL-адреса в методе для добавления нового продукта.
4. Описание ошибки:

* Загрузить программу. Перейти на страницу добавления продукта по адресу "/products/add".
* Заполнить форму добавления продукта корректными данными.
* Нажать кнопку для добавления продукта. Полученный результат: Клиент получает ошибку статуса 405 (Method Not Allowed).

1. Ожидаемый результат: Перенаправление на главную страницу или обратно на форму добавления продукта при ошибке.

**Ошибка №2:**

1. Краткое описание ошибки: Неверный URL-адрес в методе testAddProduct() для добавления продукта.
2. Статус ошибки: открыта («Open»). Категория ошибки: серьезная («Major»).
3. Тестовый случай: Проверка корректности URL-адреса в методе для добавления нового продукта.
4. Описание ошибки:

* Загрузить программу. Перейти на страницу добавления продукта по адресу "/products/add".
* Заполнить форму добавления продукта корректными данными.
* Нажать кнопку для добавления продукта. Полученный результат: Клиент получает ошибку статуса 405 (Method Not Allowed).

1. Ожидаемый результат: Перенаправление на главную страницу или обратно на форму добавления продукта при ошибке.

13.4 Итоговое тестирование.

После возвращения ПО на повторную проверку, все unit тесты успешно были пройдены, никаких ошибок нет.

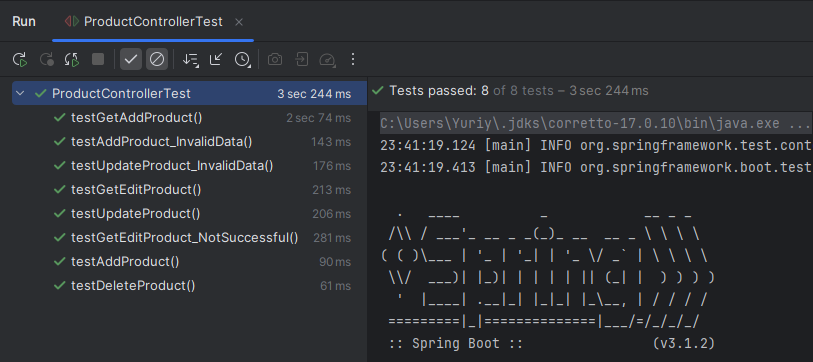


Рисунок 6 – Успешное прохождение тестирования программного продукта

# ЗАКЛЮЧЕНИЕ

Наша команда познакомилась с концепцией модульного тестирования, научилась проектировать и реализовывать модульные тесты для отдельных компонентов программного обеспечения.