Министерство образования и науки Республики Казахстан

Евразийский национальный университет им. Л. Н. Гумилева

Кафедра Информационных систем

**ОТЧЕТ ПО РЕЗУЛЬТАТАМ МОДУЛЬНОГО, ИНТЕГРАЦИОННОГО И СИСТЕМНОГО ТЕСТИРОВАНИЯ ВЕБ ПРИЛОЖЕНИЯ**

Выполнила: Наги Алина ИС-31

Елюбай Дастан ИС-31

Проверила: Жукабаева Т.К.

Нур-Султан

2021

Содержание

1.Краткое руководство о проекте………………………………………………………………....3

2.Модульное тестирование……………………………………………………………………………..4

3.Интеграцционное тестирование…………………………………………………………………11

4.Системное тестирование…………………………………………………………………………......13

4.1.Тестирование функциональности…………………………………………………………….13

4.2.Тестирование удобства использования…………………………………………………..15

4.3.Тестирование совместимости…………………………………………………………………..16

4.4.Тестирование производительности…………………………………………………………18

4.5.Тестирование безопасности……………………………………………………………………..18

5.Вывод…………………………………………………………………………………………….....…………20

6.Список использованной литературы……………………………………………………………21

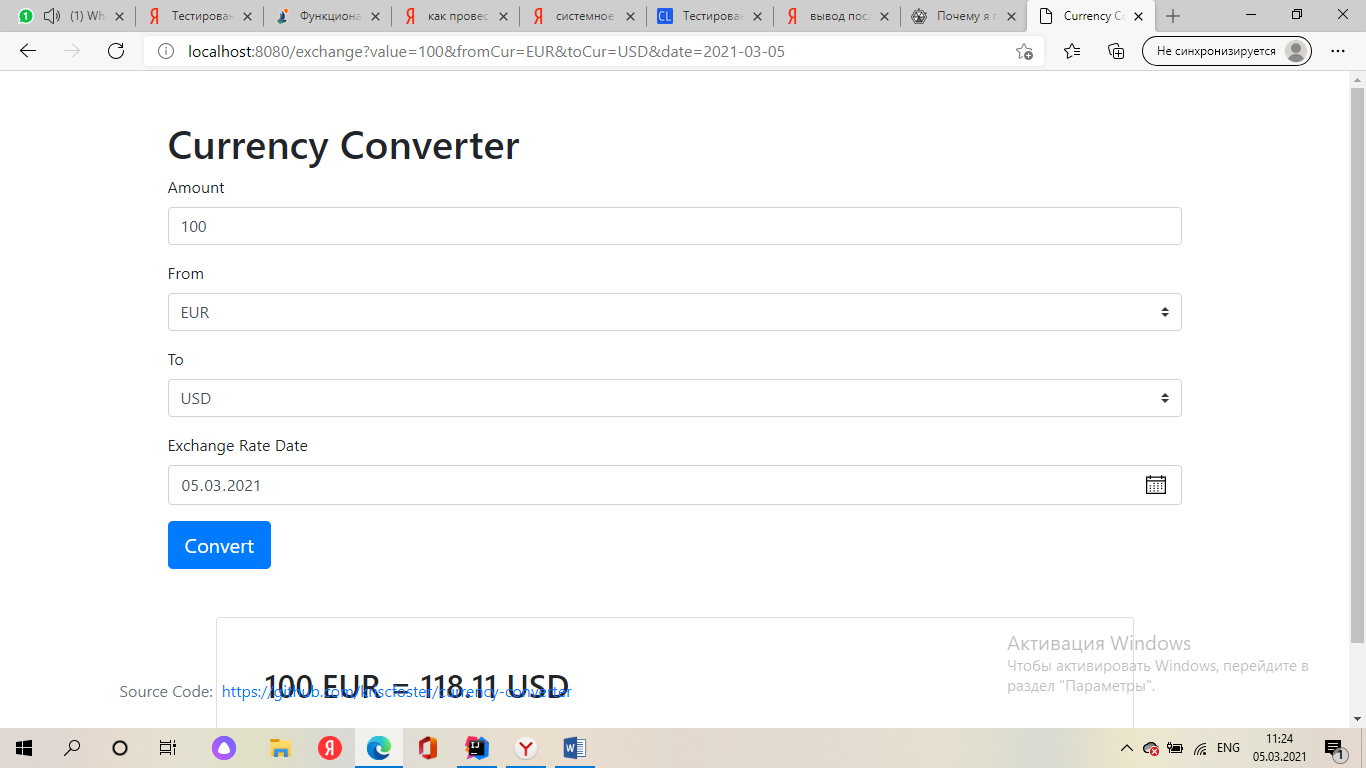
**Краткое руководство о проекте**

для начала продемонстрируем работу нашего проекта

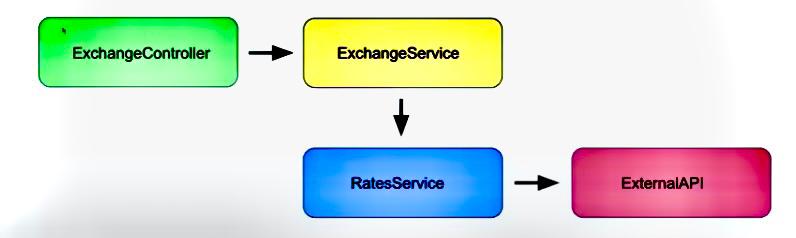
приложение, которое мы тестируем - это обмен валюты

у нас есть API, который берет одну валюту и обменивает в другую, а также дату, обменные курсы, стоимость, которую мы хотим обменять.

Так например



структура нашего приложения



ExchangeController – контроллер обмена, то, что раскрывает API

ExchangeService – обменный сервис, который имеет бизнес-логику для выполнения обмена. Этот обменный сервис имеет зависимость RatesService.

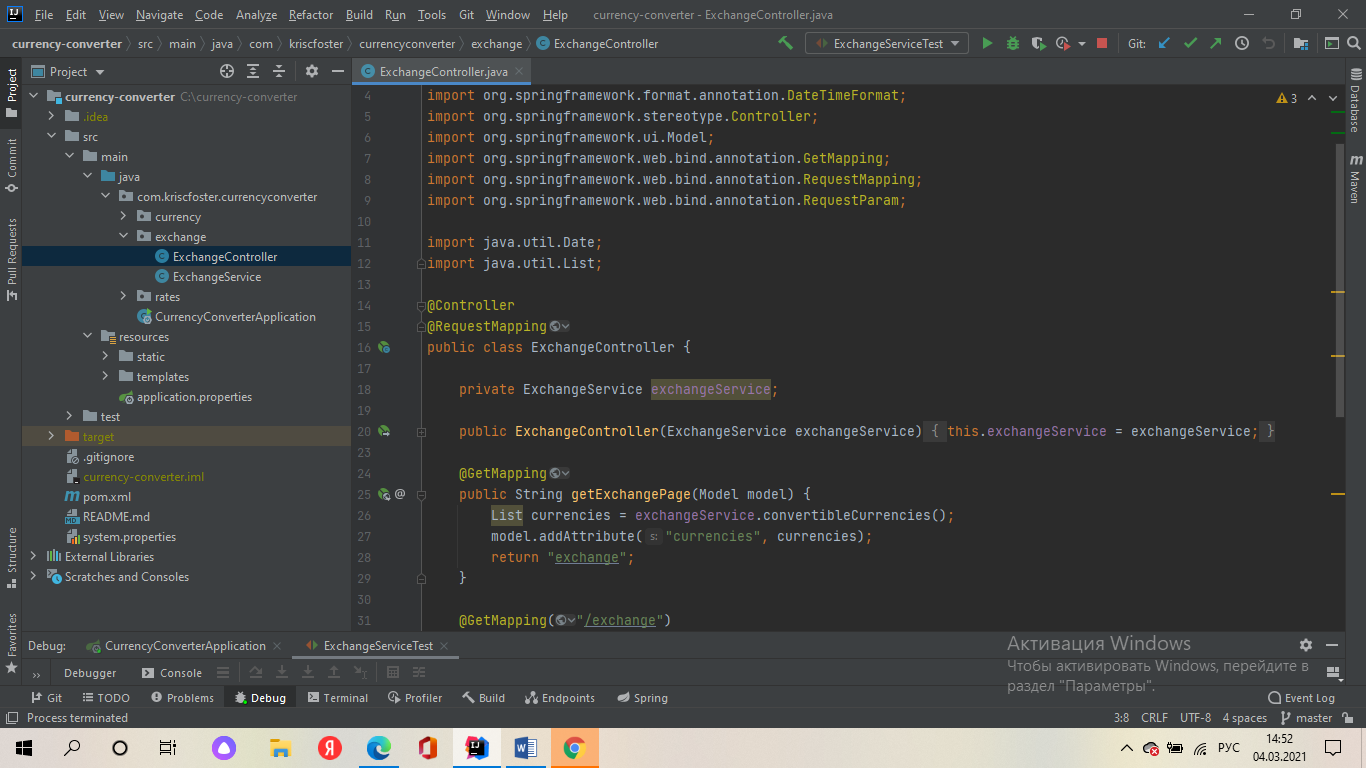
**Модульное тестирование**

**Цель теста:** проверка корректности функционирования отдельных модулей кода программы, чтобы убедиться, что отдельные функции модуля или компонента работают нормально.

При написании модульных тестов применялся инструментарий фреймворка Spring Boot. В качестве ориентира взяла официальный гайд спринга по тестированию.

Провела модульные тесты для сервиса обмена(ExchangeService), испульзуя его зависимости, а именно, RatesService, поэтому в основном тестировала данный класс.

Перейдем к коду

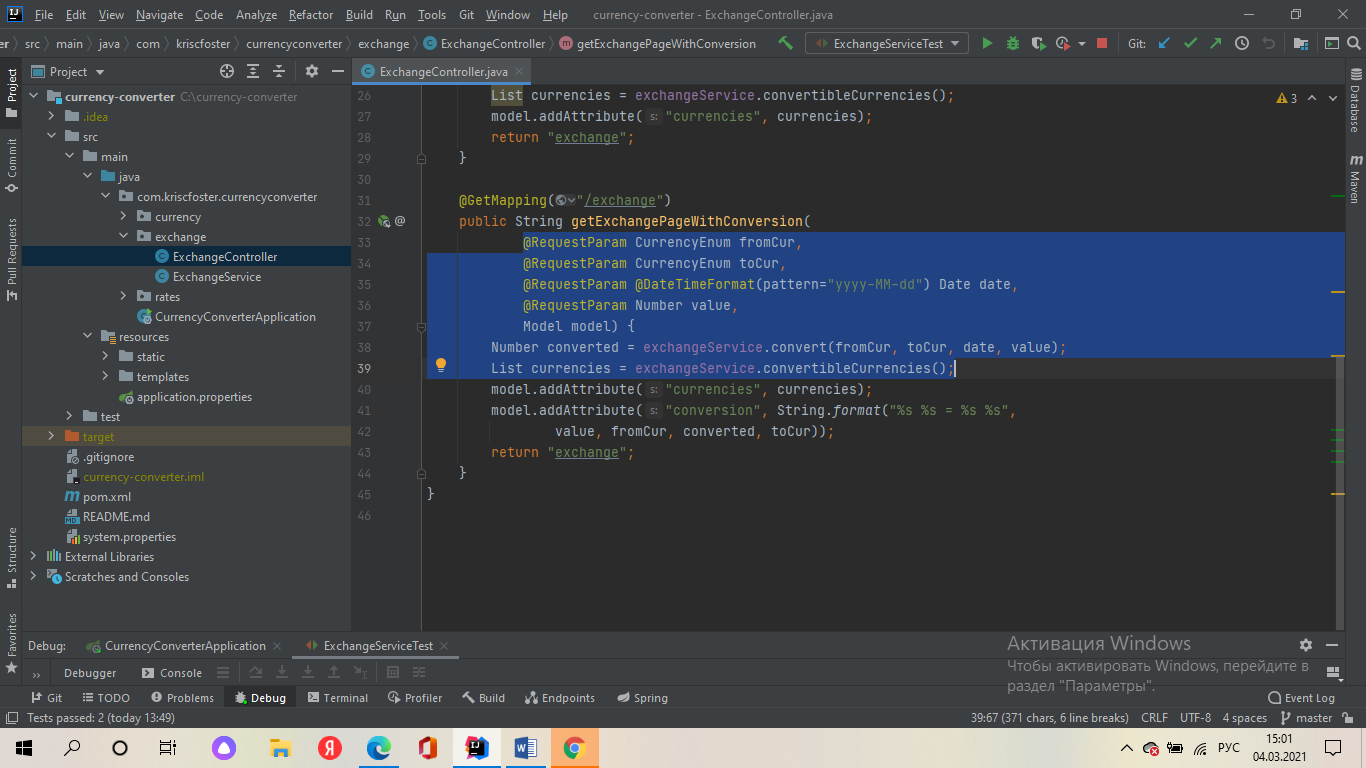
Сначала видим контроллер обмена, который раскрывает нашу конечную точку API

Здесь мы можем отслеживать все, что происходит в браузере

То есть у нас есть определенные значения, которые передаются (дата, валюта и тд).

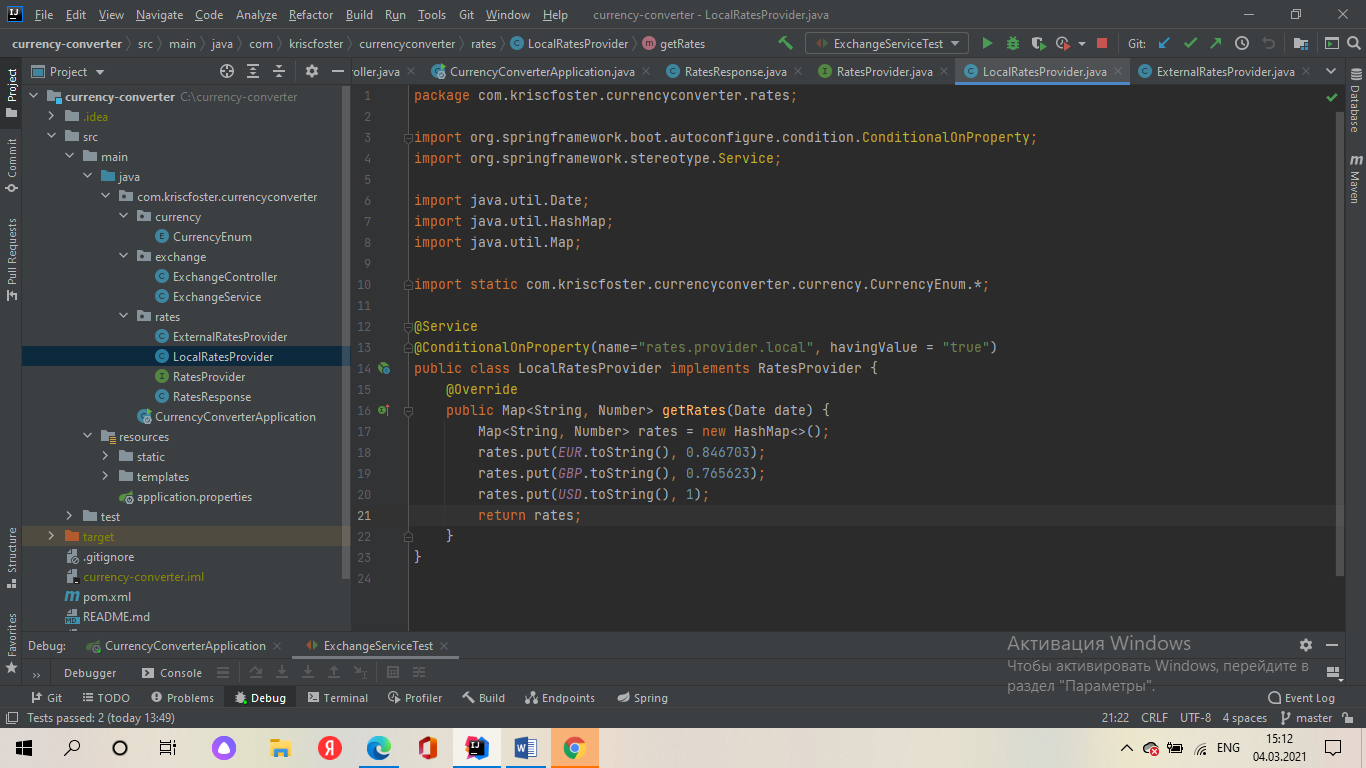
После этого используется обменный сервис для преобразования этих данных.

*Как было показано, ранее, в диаграмме ExchangeService* использует RatesService для получения текущих обменных курсов или же сведения о курсе на предыдущие даты.

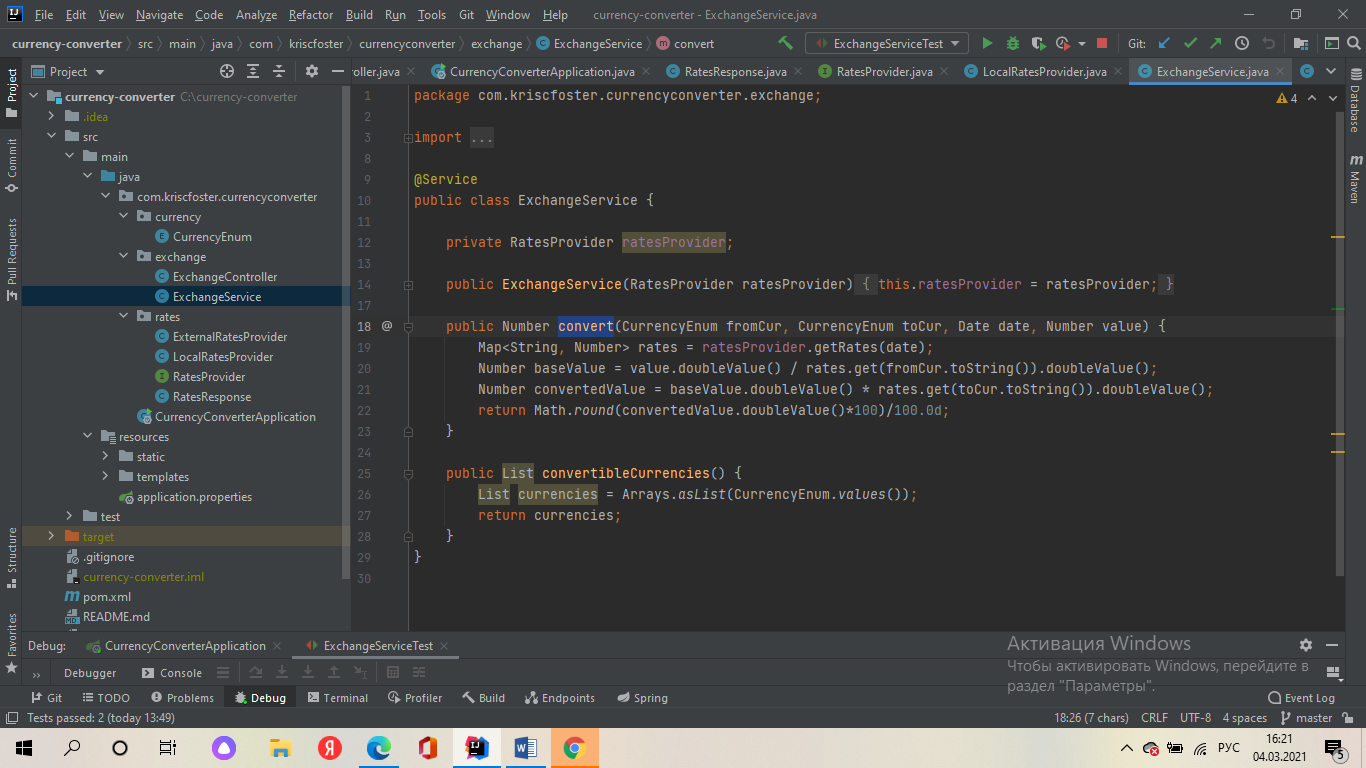


|  |  |
| --- | --- |
| success | Возвращает trueзначение or falseв зависимости от того, успешно ли выполняется запрос. |
| terms | Возвращает ссылку на условия и положения currencylayer. |
| privacy | Возвращает ссылку на политику конфиденциальности currencylayer. |
| timestamp | Возвращает точную дату и время (UNIX), когда были собраны обменные курсы. |
| source | Возвращает валюту, к которой все обменные курсы являются относительными. (по умолчанию: USD) |
| quotes | Содержит все значения обменного курса, состоящие из валютных пар и соответствующих им курсов конвертации. |

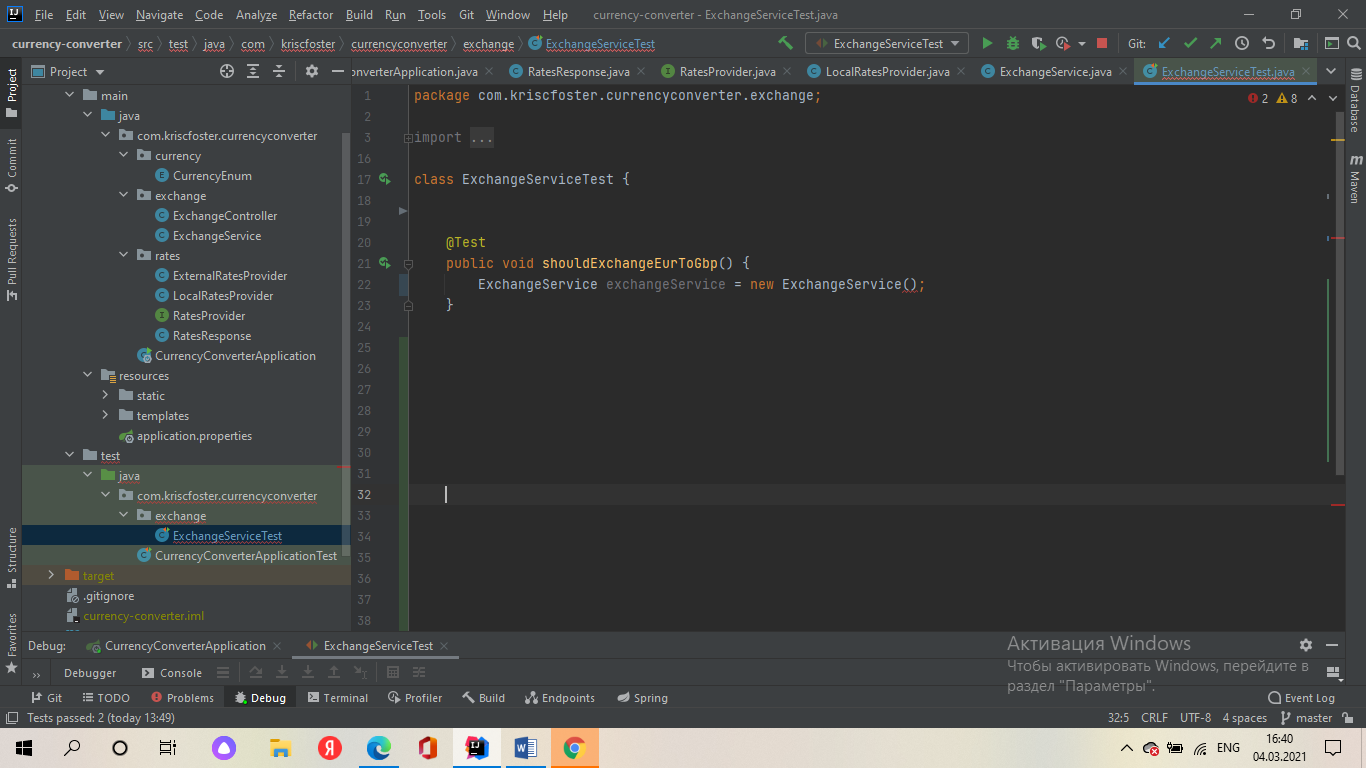
RatesService выглядит следующим образом



Напишем модульные тесты для метода преобразования



Далее сгенерируем тест для этого метода

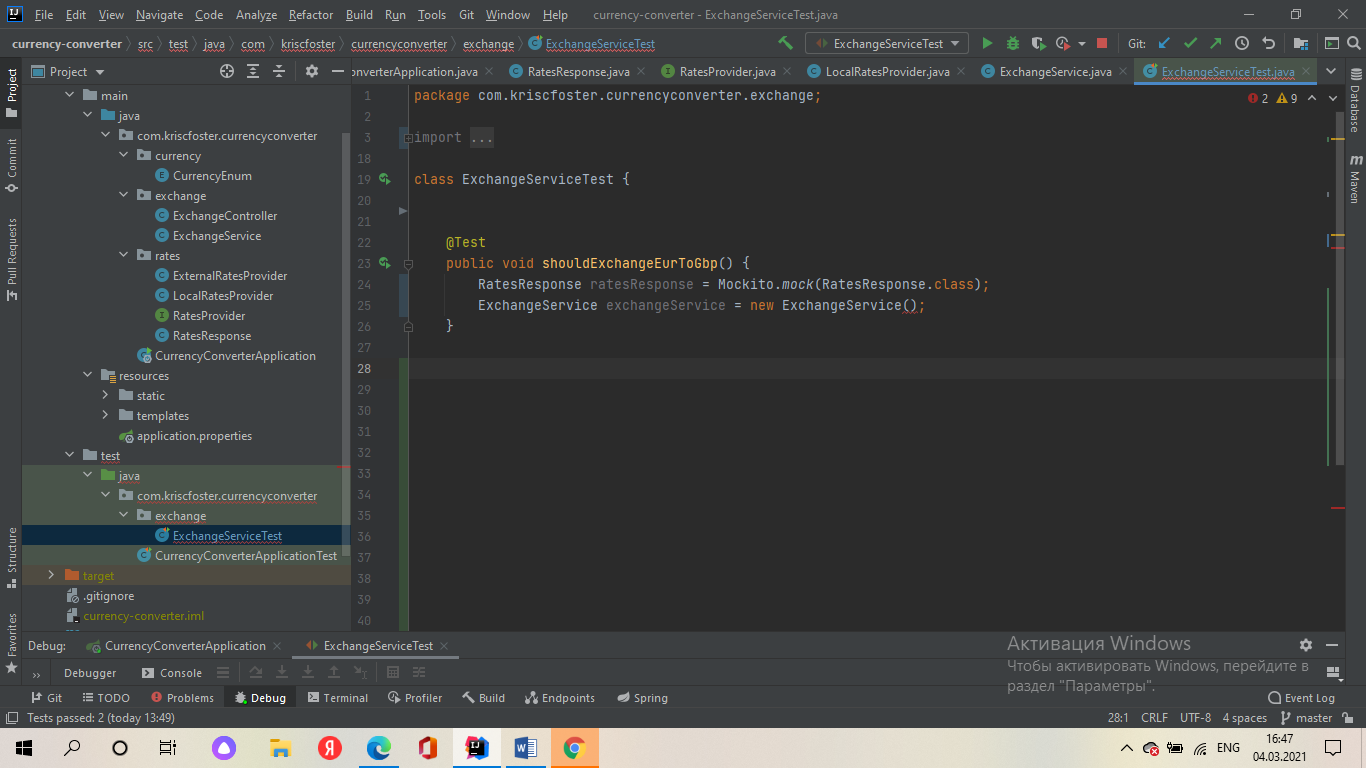


Тест называется конвертируем евро в фунт

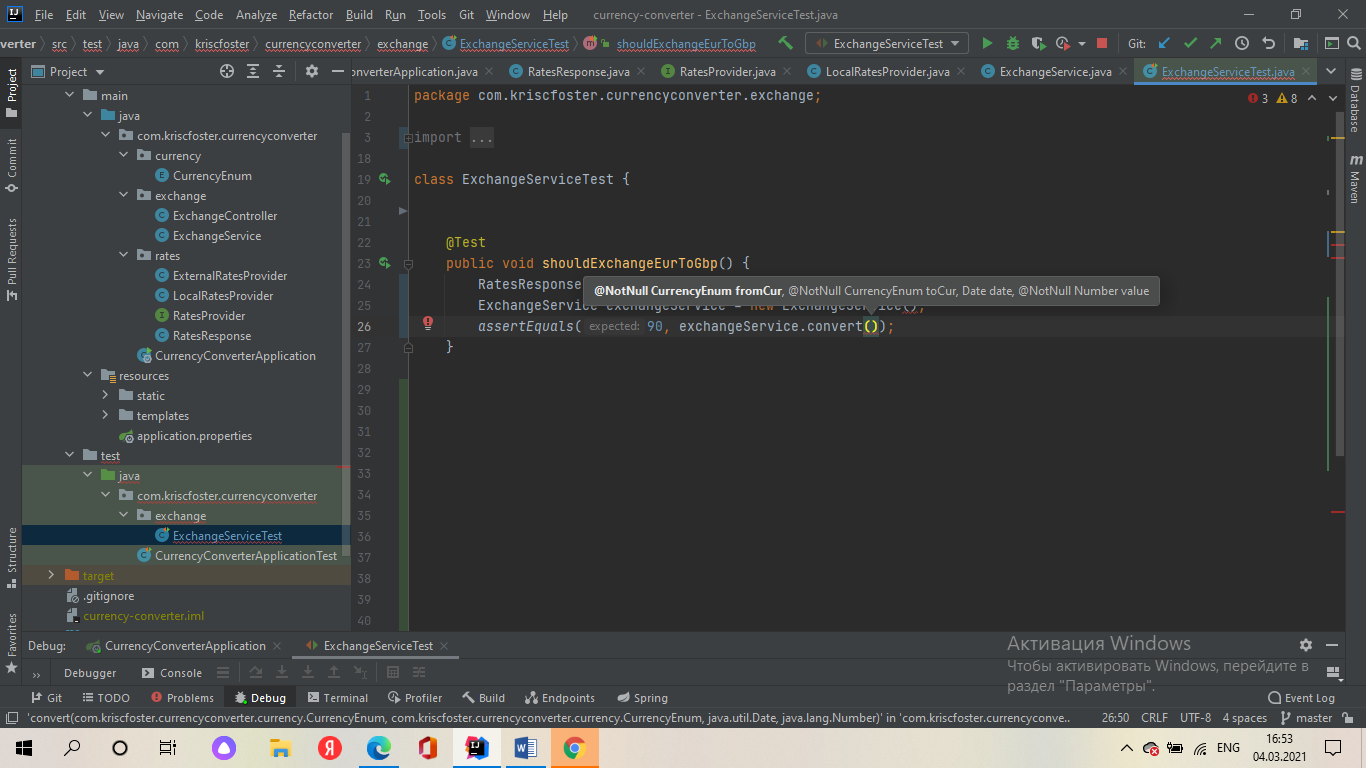
Для тестирования нам нужен экземпляр сервиса обмена чтобы иметь возможность вызвать на нем метод convert

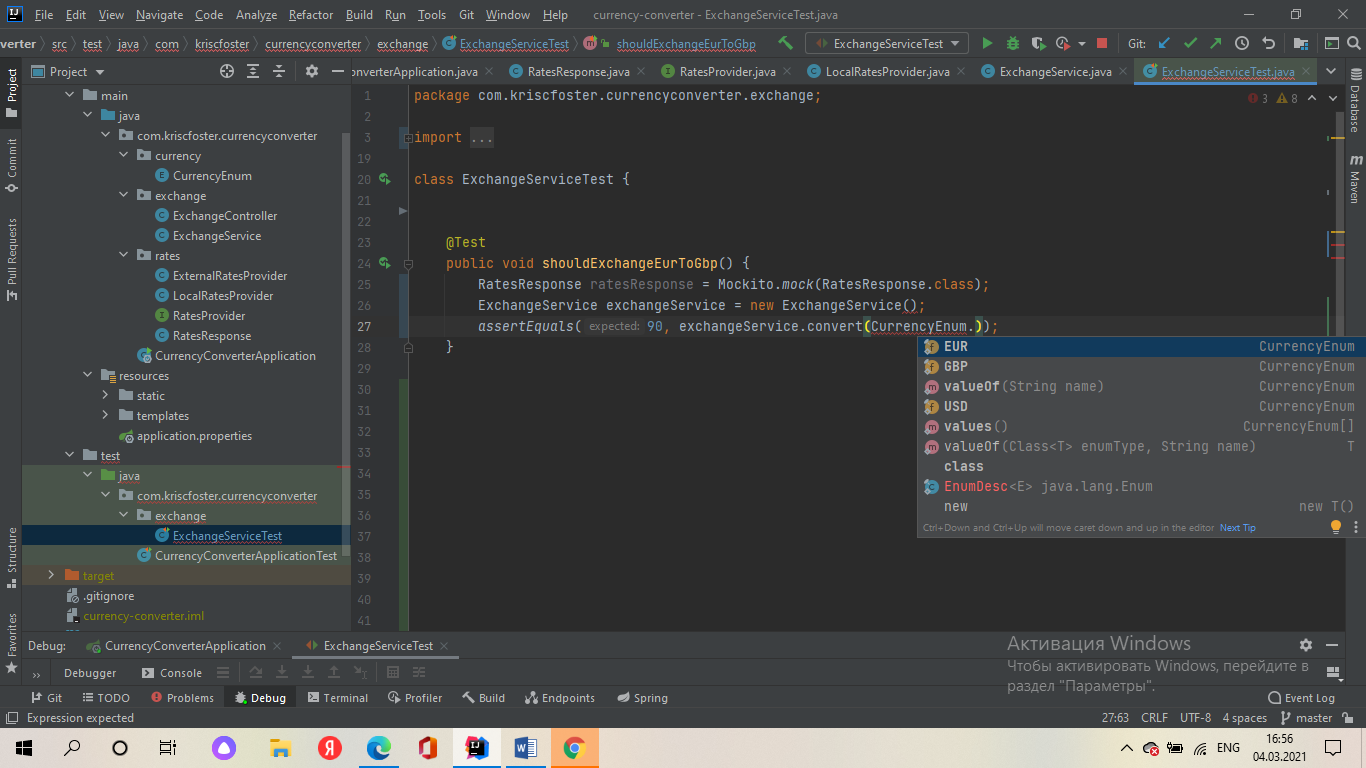
Далее создаем новый экземпляр RatesService

Используем Mockito, далее импортируем (для внедрения Mockito Mocks в Spring Beans нам нужно добавить зависимость Mockito-core в наш файл конфигурации сборки).



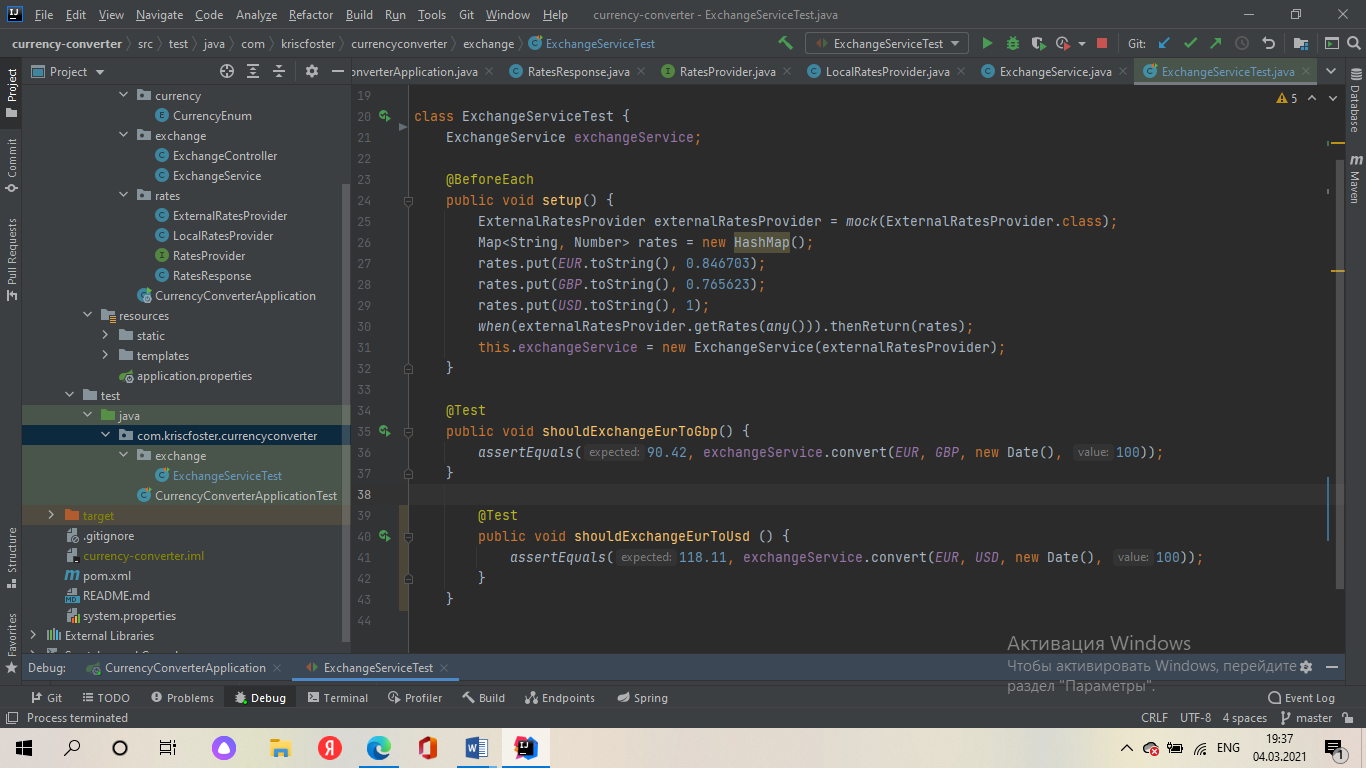
После чего нам необходимо использовать assertEquals, нас уведомляют, что этот метод требует некоторых параметров



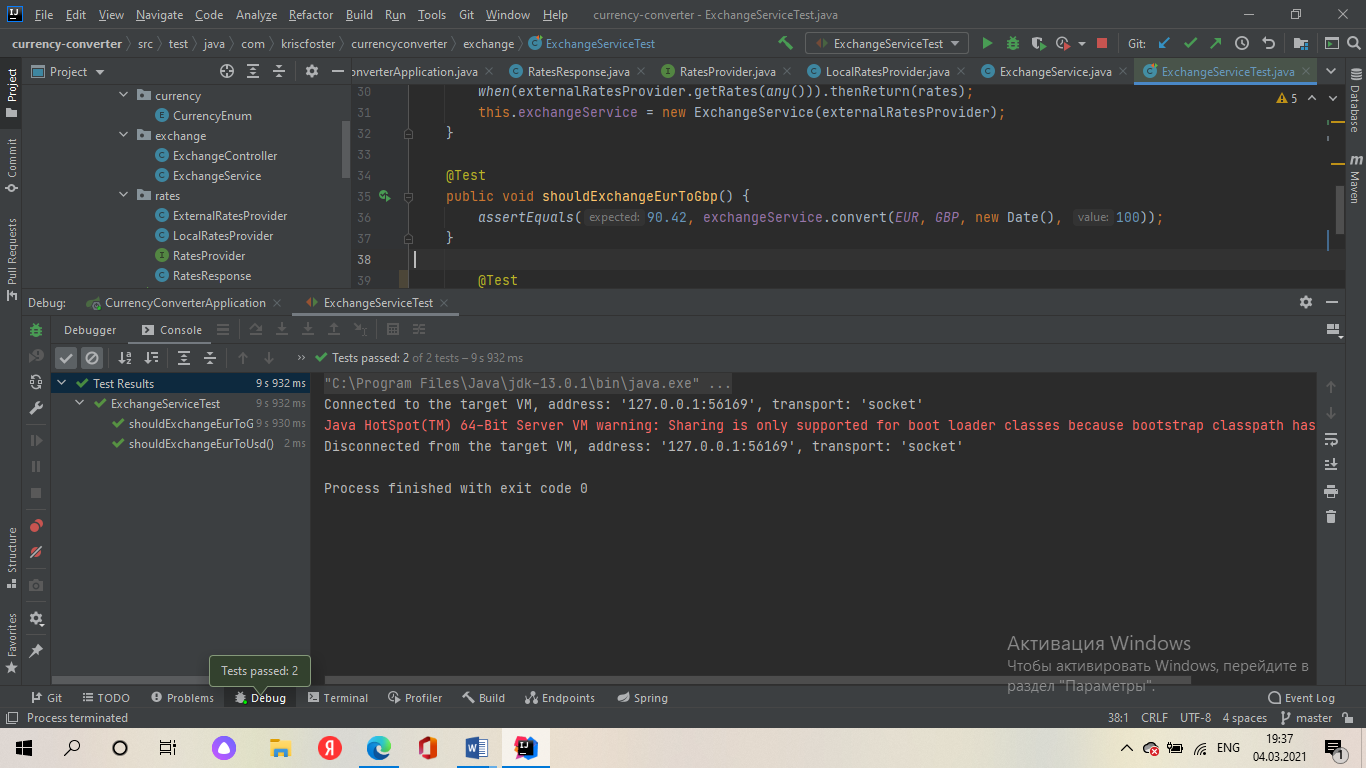


Конвертируем евро в фунты

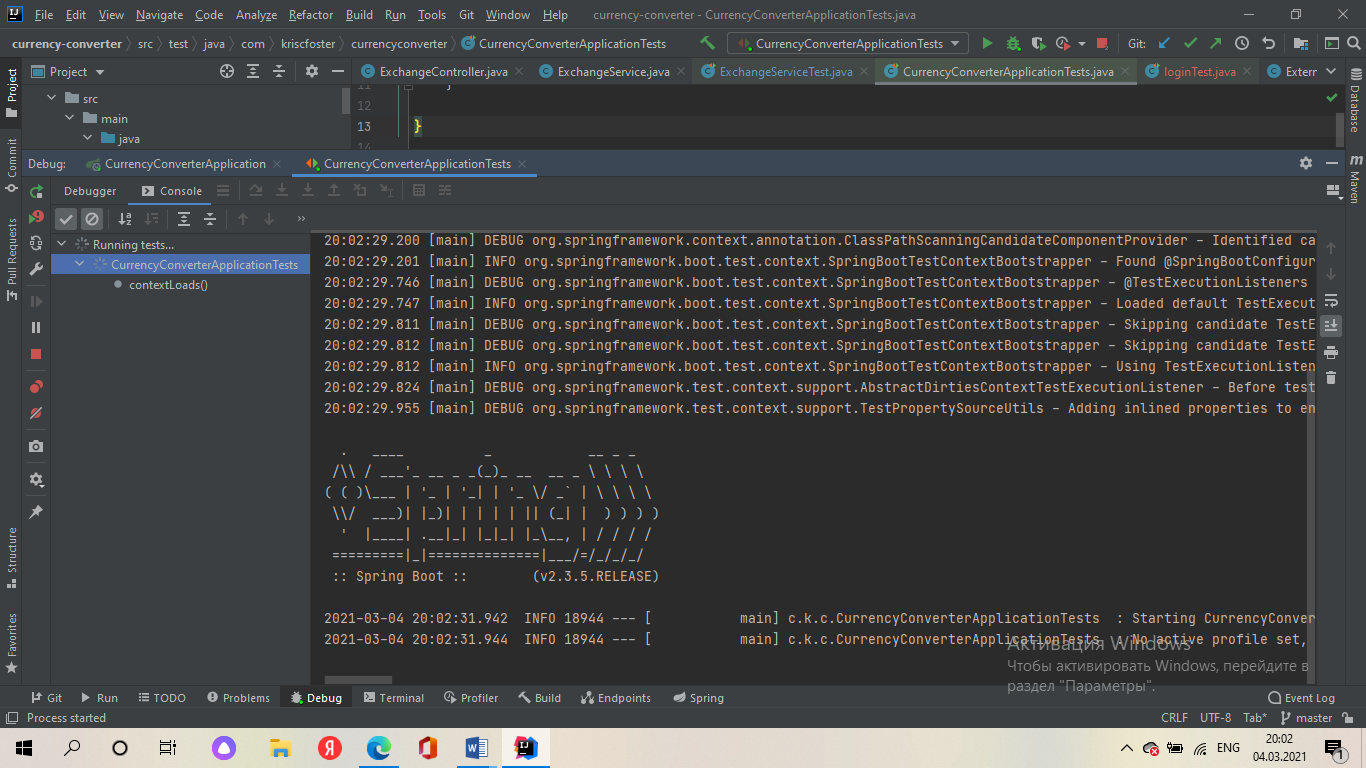
Добавляем дату, чтобы получить конвертацию на текущий период.



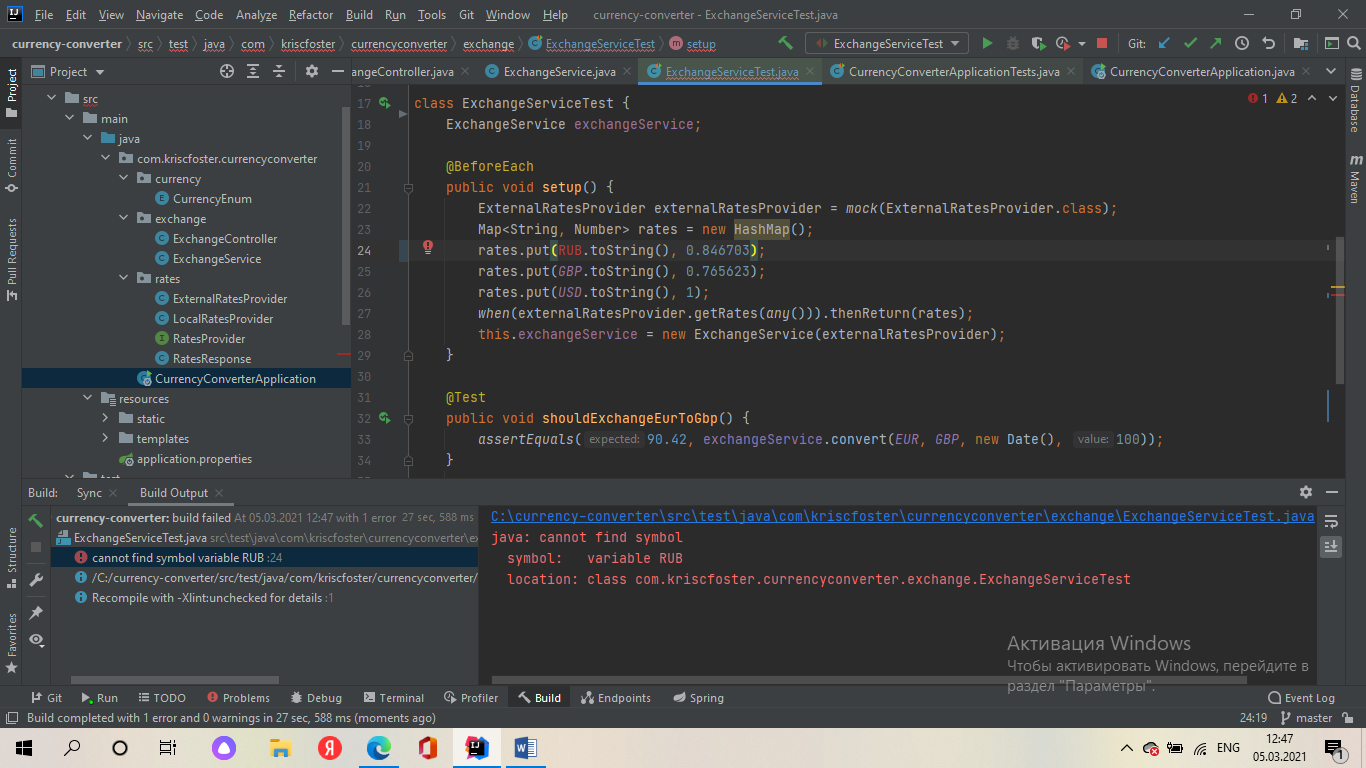
Все нужные настройки помещаем в аннотацию Before



Тест успешно пройден



Сделаем специально ошибку:



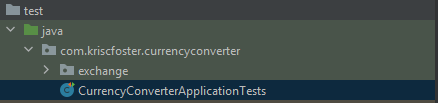
**Результаты:**

В системе с хорошо проработанными API вы можете протестировать каждый класс, прежде чем двигаться дальше, чтобы выяснить, какая часть логики нарушена. Например, в моей программе метод «get» вызывает getRates для взаимодействия с параметрами. Сначала я бы протестировала getRates и удостоверилась, что тесты успешно пройдены и отмечены зеленым. Таким образом, я буду знать, что если упадет какой-либо из тестов над контроллерами, то это возможно, будет связано с тем, как я вызываю getData.

**Интеграционное тестирование**

**Цель теста**: удостовериться в том, что взаимосвязи между программными компонентами отвечают всем функциональным требованиям.

Создаем тест CurrencyConverterAplicationTests в папке test нашего проекта



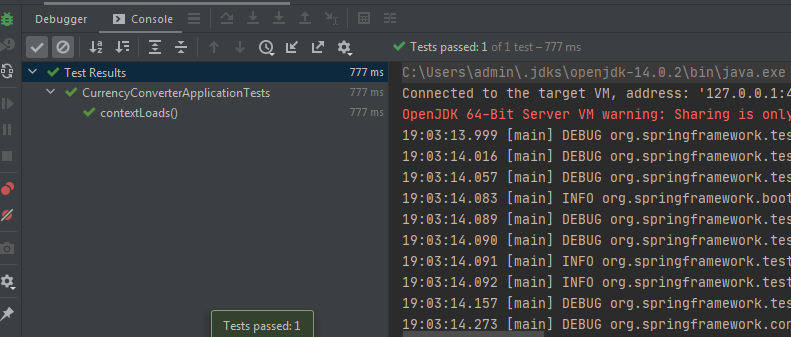
Код:

package com.kriscfoster.currencyconverter;  
  
import com.kriscfoster.currencyconverter.exchange.ExchangeController;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;  
  
import static org.assertj.core.api.Java6Assertions.*assertThat*;  
  
@SpringBootTest  
@AutoConfigureMockMVC  
class CurrencyConverterApplicationTests {  
   
 @Autowired  
 private ExchangeController controller;  
  
 @Test  
 public void contextLoads() throws Exception {  
 *assertThat*(controller).isNotNull();  
 }  
  
}

SpringBootTest- является комбо аннотацией она указывает что мы запускаем тесты в окружений SpringBoot приложений и будет выполнять тесты в этом контексте

Test- аннотация помечает все тестовые методы которые находятся в классе тест-кейс

Проверяем тест:

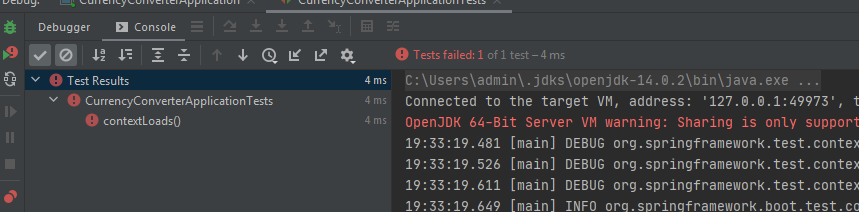


Тест пройден

Делаем ошибку в коде:

@SpringBootTest  
@AutoConfigureMockMVC  
class CurrencyConverterApplicationTests {  
 @Autowired  
 private MockMvc mockMvc;  
  
 @Autowired  
 private ExchangeController controller;  
  
 @Test  
 public void contextLoads() throws Exception {  
 this.mockMvc.perform(get("/"))  
 .andDo(print())  
 .andExpect((ResultMatcher) status().isOk())  
 .andExpect((ResultMatcher) content().string(ContainsString("Hello guest")));  
 }

Проверяем тест:



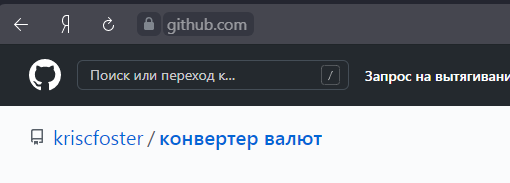
Тест не пройден

**Системное тестирование**

**Цель теста**: оценка сквозных технических характеристик системы.

**1.Тестирование функциональности:**

Проверка всех ссылок



Ссылка перенаправляет на репозитори

Пройдено

Проверка файлов Cookies

Файл cookie — это небольшой фрагмент информации, отправляемый веб-сервером для хранения в веб-браузере, чтобы впоследствии он мог быть прочитан браузером. Таким образом, браузер запоминает определенную личную информацию. Если хакер получает информацию о cookie-файлах, это может привести к проблемам с безопасностью.

В данном проекте отсутствует Cookies

Не пройдено

**Проверка pom файла на наличие ошибок в коде:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.3.5.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.kriscfoster</groupId>  
 <artifactId>currency-converter</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>currency-converter</name>  
 <description>Demo project for Spring Boot</description>  
  
 <properties>  
 <java.version>11</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 <exclusions>  
 <exclusion>  
 <groupId>org.junit.vintage</groupId>  
 <artifactId>junit-vintage-engine</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Ошибок не обнаружено**

**Пройдено**

**Тестирование БД:**

В данном проекте отсутствует БД

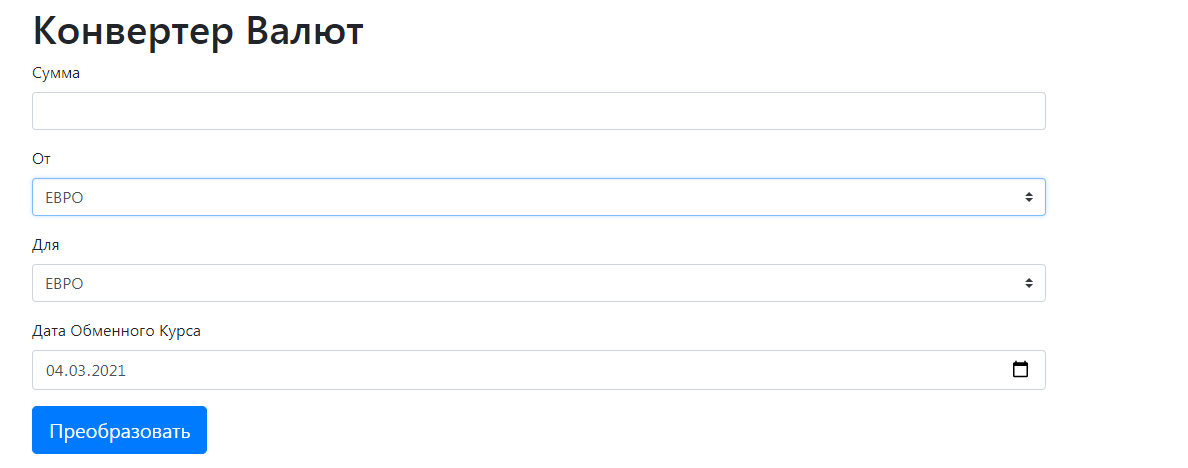
Не пройдено

**2.Тестирование удобства использования:**

Любое приложение создается для того, чтобы им воспользовались. Удобство использования - важный качественный показатель программы. Тестирование юзабилити включает в себя детальный анализ поведения пользователей.

**Проверка юзабилити теста**

Веб приложение очень простое имеет одну кнопку, 1 textbox, 2 listbox, 1datelist



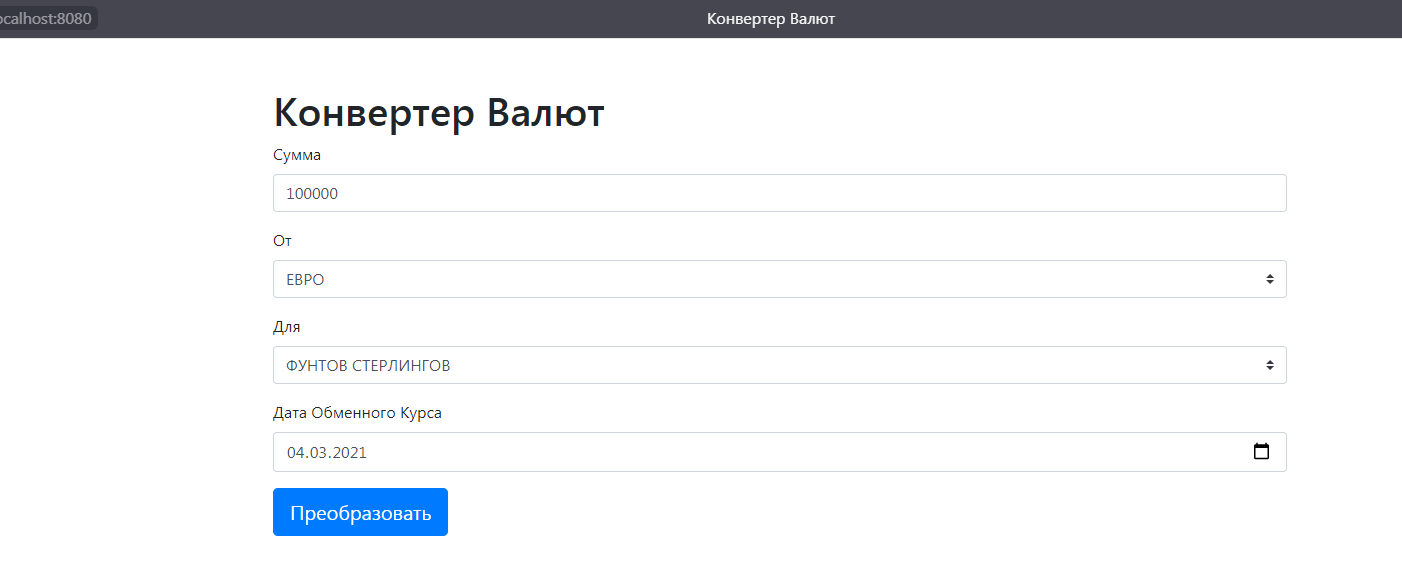
**Пройдено**

**3.Тестирование совместимости:**

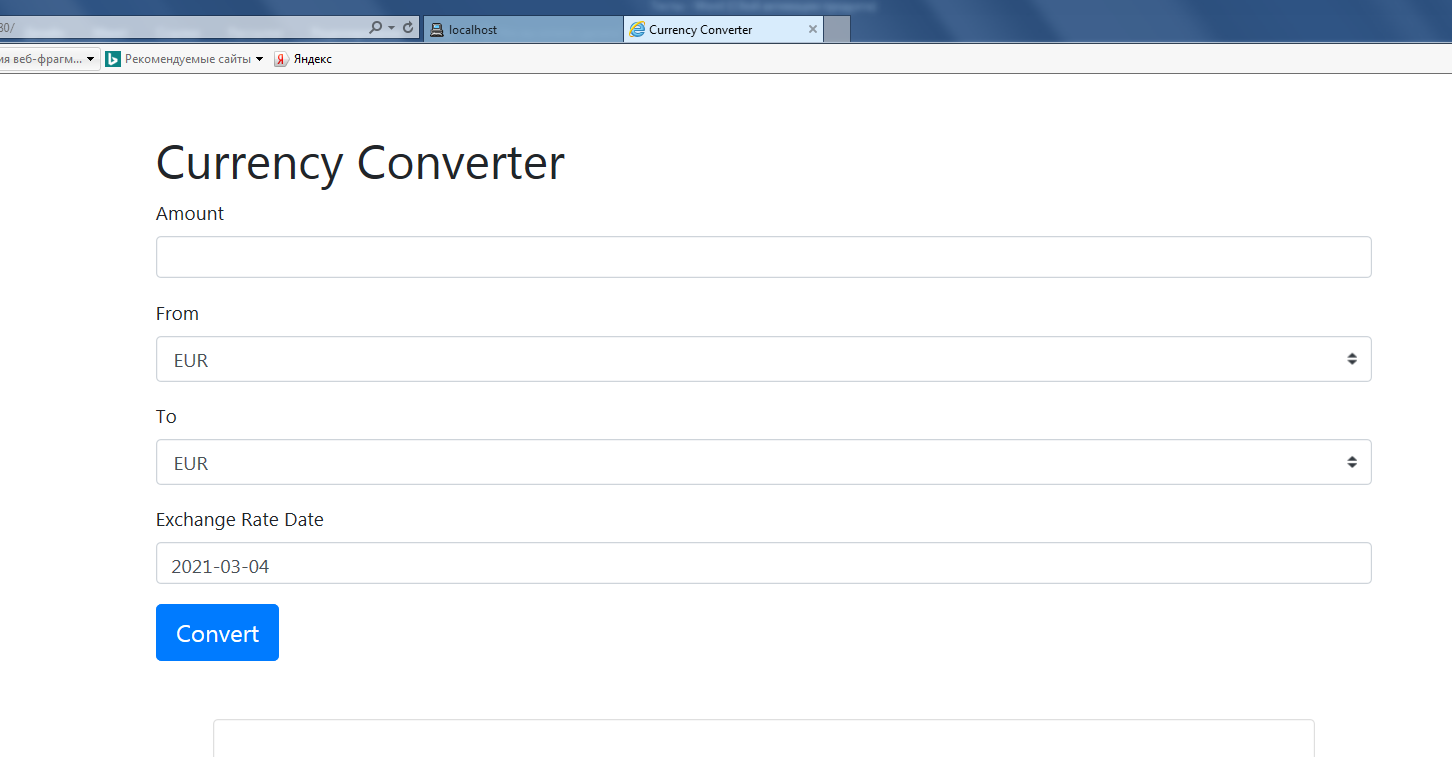
**Проверка совместимости**

Совместимость с браузерами:

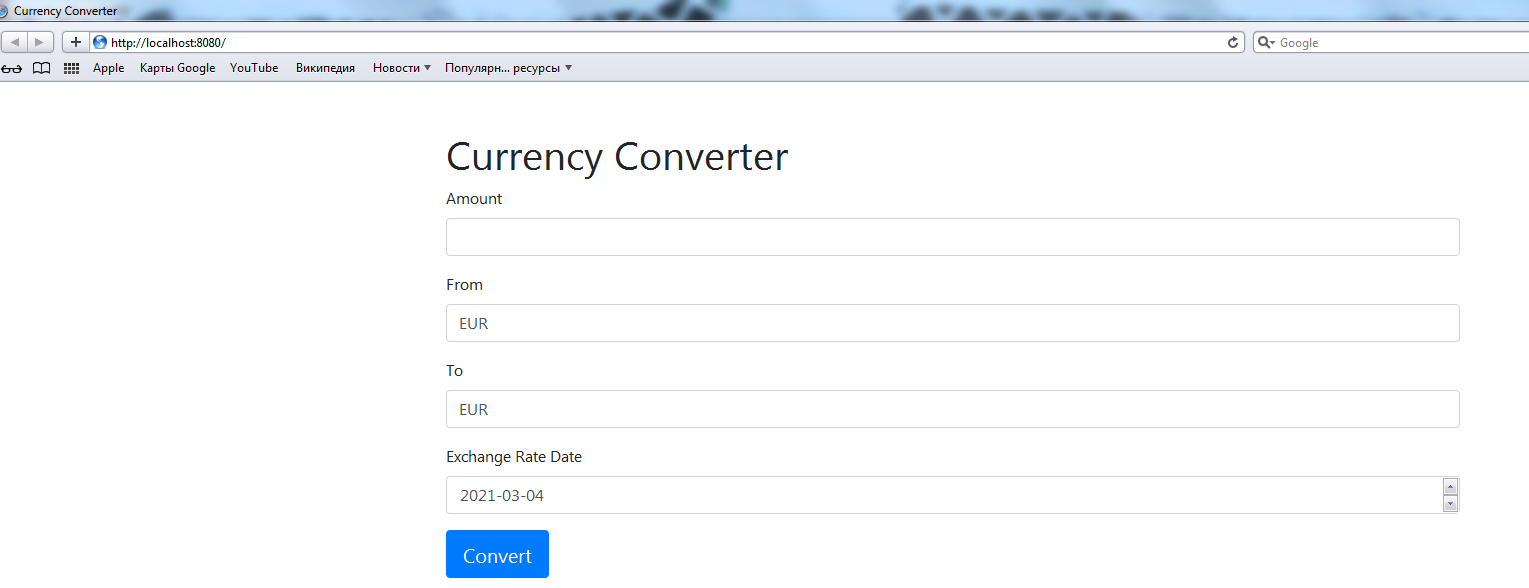
Yandex:



Internet Explorer:

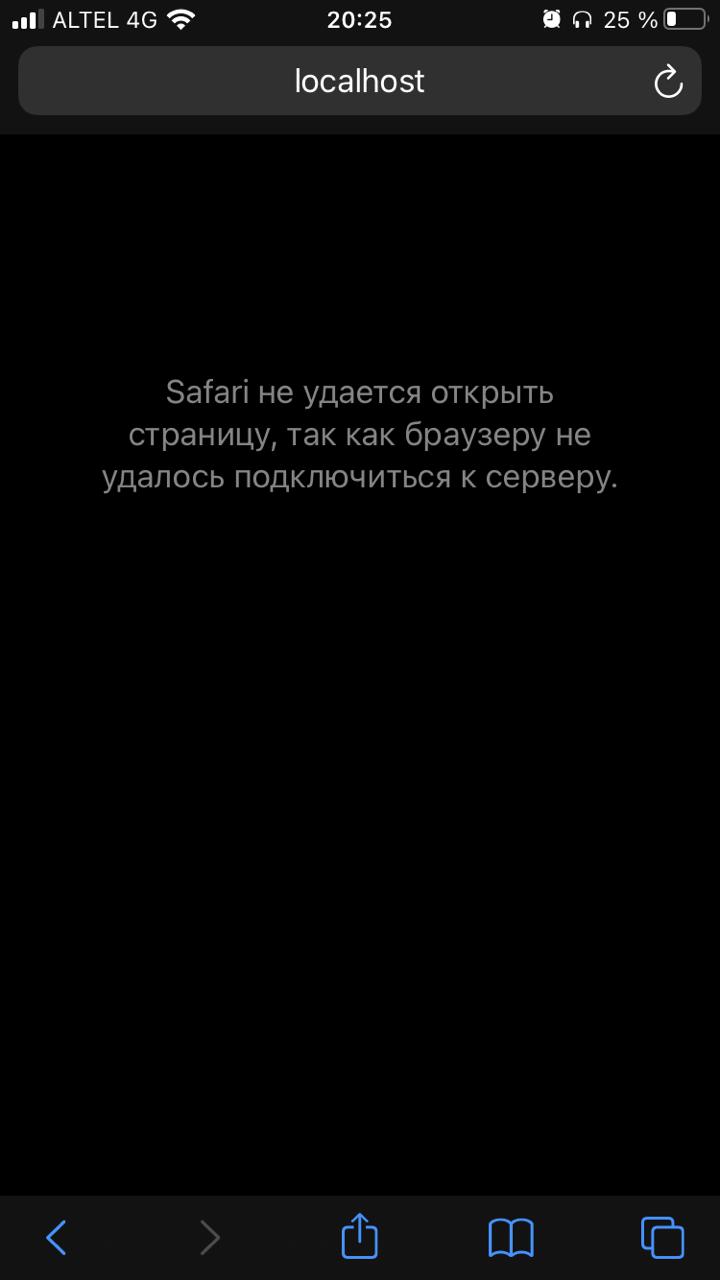


Safari:



Пройдено

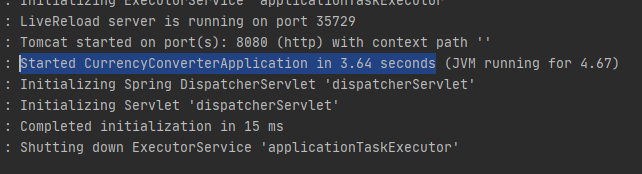
**Просмотр на мобильных устройствах:**

****

Не пройдено

**4.Тестирование производительности веб приложения**

**Скорость запуска приложения и виртуальной машины(JVM):**



**Пройдено**

**Скорость соединения и ответа сервера:**

**Для ПК:**

**4.155 seconds**

**Пройдено**

**Нагрузочное тестирование**

Нагрузочное тестирование - процесс анализа производительности тестируемой системы под воздействием нагрузок. Цель нагрузочного тестирования- определить способность приложения к внешним нагрузкам

Запускали приложение с других классов



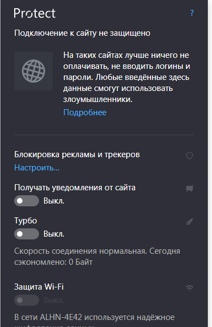
Не пройдено

**5.Тестирование безопасности**

Данное тестирование направлено на выявление всех возможных лазеек и слабых мест в программной системе, которые могут привести к потере информации.

Тестирование безопасности особенно актуально для наших web-приложений, приложений с важной коммерческой или персональной информацией, платежных систем, приложений, требующих целостности информации, социальных приложений, приложений с коммерческой лицензией.

На самом приложение данная функция отсутствует, но в браузере Yandex данная функция реализована



Не пройдено

Диаграмма на основании вышеперечисленных тестов:

**Вывод**

В ходе данной лабораторной работы мы выполнили следующее:

Составили тест-план, где был описан весь объём работ по тестированию и определено, когда их закончить.

Провели дымовое тестирование, чтобы удостовериться что все запускается корректно.

После этого программу проверили на разных уровнях:

Модульное тестирование.

Интеграционное тестирование

Системное тестирование с применением юзабилити, нагрузочного, безопаности и т.д.

Конечный продукт – это отчет о проделанных испытаниях в разрезе общего количества пройденных тестовых сценариев с их результатами.

**Список использованной литературы:**

[Getting Started | Serving Web Content with Spring MVC](https://spring.io/guides/gs/serving-web-content/)

[Основы тестирования ПО 2020: самый подробный гайд - Первый Онлайн ИНститут Тестировщиков (pointschool.ru)](https://pointschool.ru/osnovi-tstirovaniya-po-2020-vs-o-qa/)

[Spring | Guides](https://spring.io/guides)

[Getting Started | Testing the Web Layer (spring.io)](https://spring.io/guides/gs/testing-web/)

[Testing (spring.io)](https://docs.spring.io/spring-framework/docs/current/reference/html/testing.html)

[Unit and integration testing · dotnet-architecture/eShopOnContainers Wiki (github.com)](https://github.com/dotnet-architecture/eShopOnContainers/wiki/Unit-and-integration-testing)

[50 Best Integration Testing Tools | Threat Stack](https://www.threatstack.com/blog/50-best-integration-testing-tools)

[Тестирование программного обеспечения - Краткое руководство - CoderLessons.com](https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/izuchite-testirovanie-programmnogo-obespecheniia/testirovanie-programmnogo-obespecheniia-kratkoe-rukovodstvo)