С

МИНОБРНАУКИ РОССИИ

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)

# Отчет по практической работе №2

по дисциплине «Тестирование и верификация программного обеспечения»

**Выполнили:**  Студенты группИКБО-20-21 Квашнин Ю. В.

Емельянов И. В.

Мухаметшин А. Р.

Хитров Н.С.

**Проверил:**  ассистент Овчинникова М.А.

2023 г.

## **Цель практической работы**

Целью данной практической работы является получение навыков совместной разработки программного продукта с использованием модульного тестирования и документирования кода.

## **Документация на программный продукт**

Класс DataProcessor содержит один метод calculateSum, который позволяет вычислять сумму элементов в массиве целых чисел. Метод calculateSum принимает в качестве параметра массив целых чисел и возвращает сумму всех элементов в этом массиве (рис. 1).

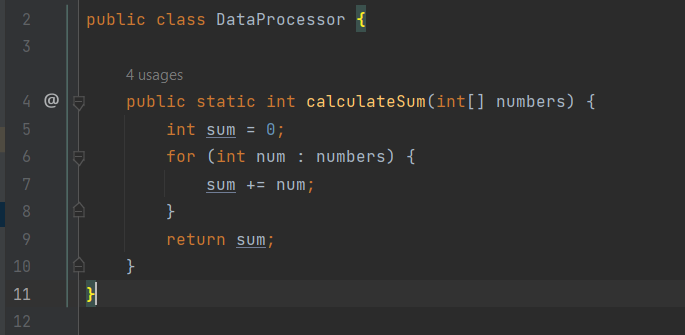


Рисунок 1 – метод calculateSum

1. **Модульное тестирование**

Модульные тесты проверяют корректность работы функции calculateSum для разных входных данных (рис. 2-3).



Рисунок 2 – модульные тесты для метода calculateSum

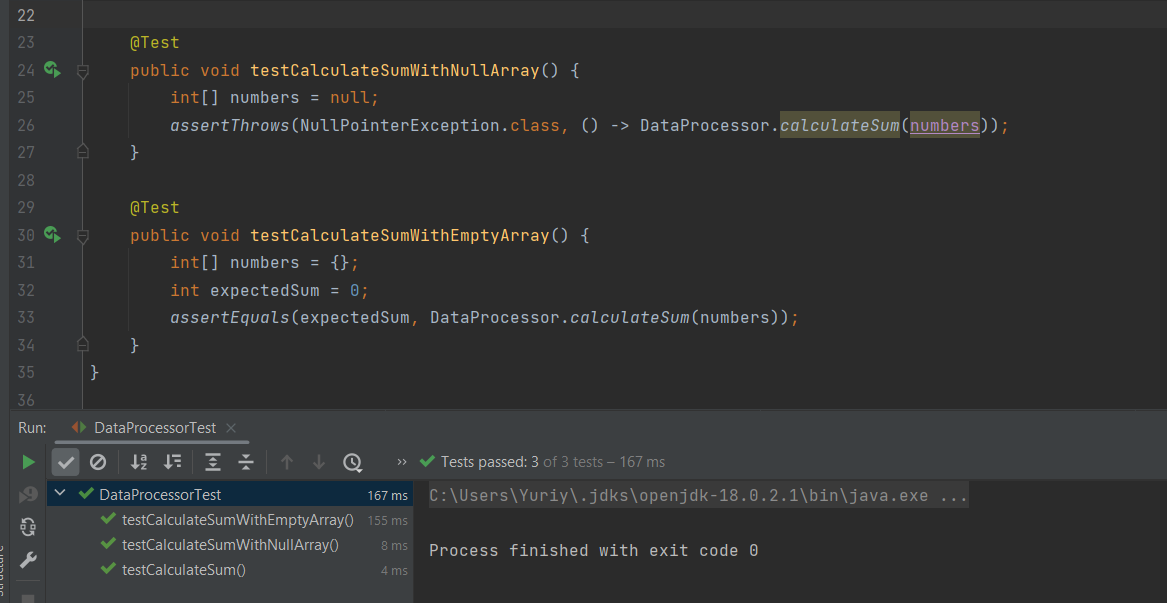


Рисунок 3 – результат тестирования

Тесты проверяют следующие сценарии:

1. testCalculateSum: Проверяет, что метод обработки данных возвращает нужную сумму, когда передаётся массив целых чисел.

2. testCalculateSumWithNullArray: Проверяет, что метод обработки данных бросает исключение NullPointerException, когда на вход передается null.

3. testCalculateSumWithEmptyArray: Проверяет, что метод возвращает 0, когда передается пустой массив.

**4. Документация на программный продукт**

Класс PalindromeChecker предоставляет метод isPalindrome, который используется для определения, является ли заданная строка палиндромом (строкой, которая читается одинаково как с начала, так и с конца). Метод принимает входную строку и возвращает true, если строка является палиндромом, и false, если нет. Если входная строка пуста или равна null, метод бросает исключение IllegalArgumentException.

Этот класс полезен для проверки строк на палиндромы и может быть использован в приложениях, связанных с обработкой текста или строковых данных (рис. 4).

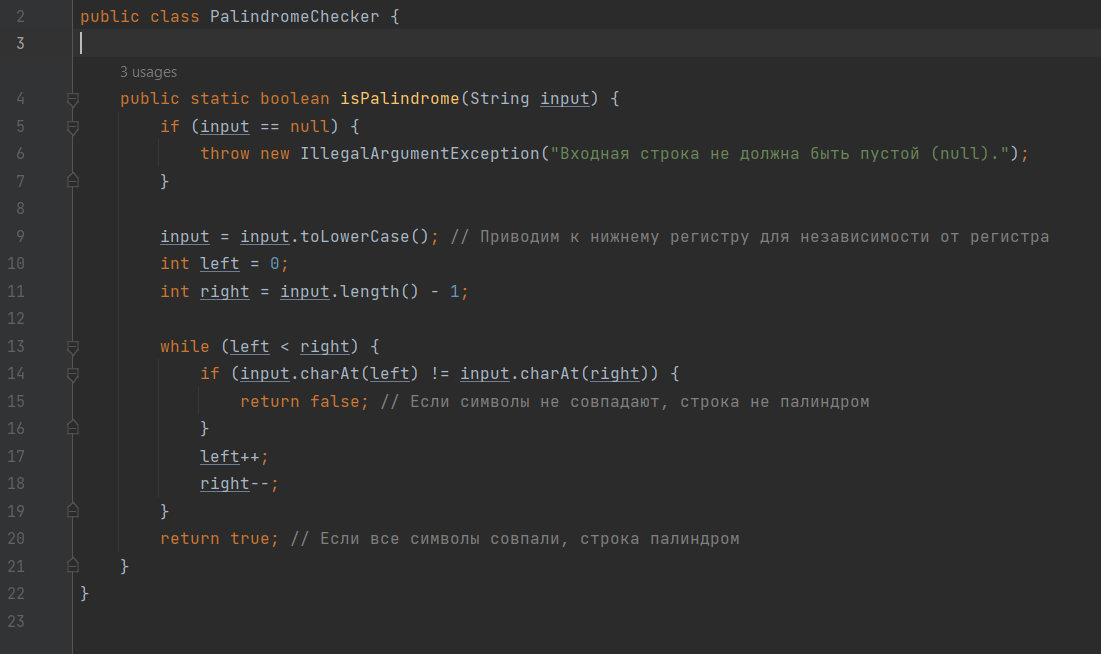


Рисунок 4 – функция isPalindrome

1. **Модульное тестирование**

Модульные тесты проверяют корректность работы функции isPalindrome для разных входных данных (рис. 5).

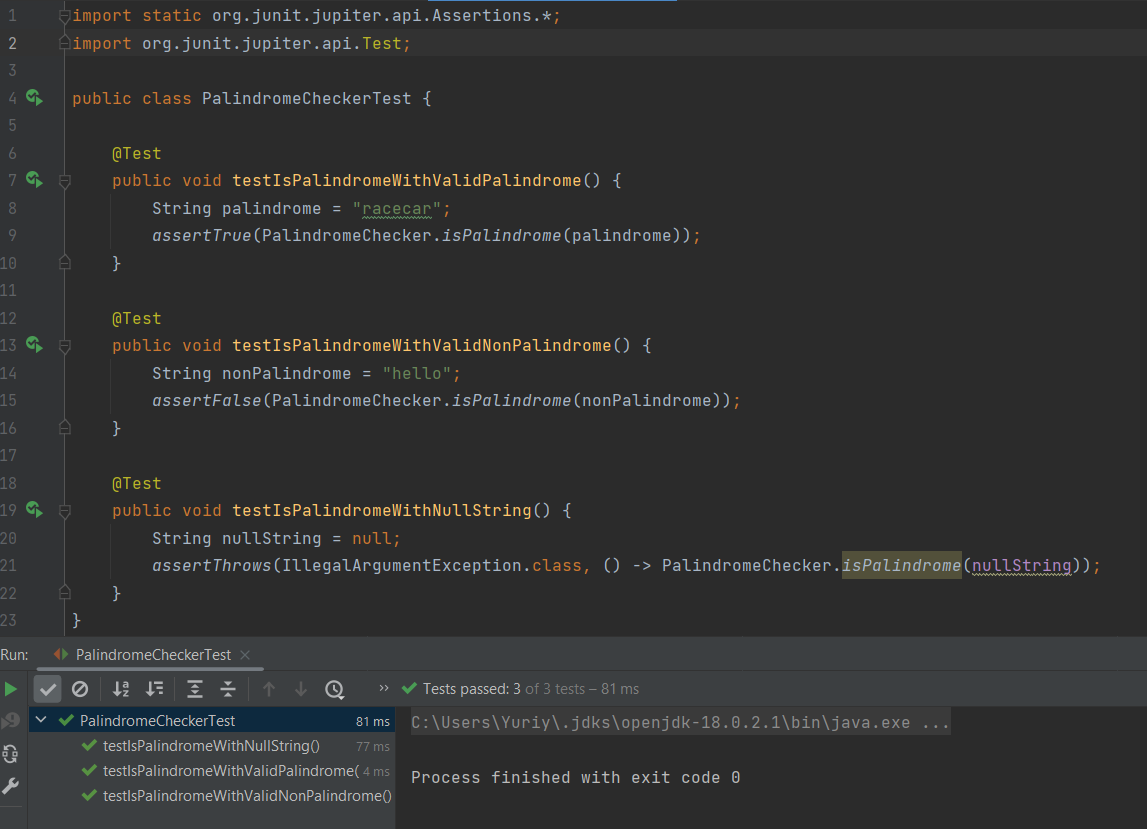


Рисунок 5 – тесты для функции isPalindrome

Эти тесты проверяют следующие сценарии:

1. testIsPalindromeWithValidPalindrome: Проверяет, что функция правильно определяет палиндром (строку, которая является палиндромом).
2. testIsPalindromeWithValidNonPalindrome: Проверяет, что функция правильно определяет строку, которая не является палиндромом.
3. testIsPalindromeWithNullString: Проверяет, что функция бросает исключение IllegalArgumentException, если входная строка равна null.

**6. Документация на программный продукт №2**

Программный продукт представляет собой бекенд-приложение для ресторана, предоставляющее RESTful API для управления меню. API поддерживает следующие функции:

* Создание меню:
  + Возможность добавления новых элементов меню (блюда, подменю) с указанием названия, описания, цены.
  + Получение информации о меню:
  + Возможность получения списка всех элементов меню.
  + Возможность получения информации о конкретном элементе меню по его идентификатору.
* Обновление меню:
  + Возможность изменения информации о существующих элементах меню, включая название, описание, цену и другие характеристики.
* Удаление меню

Технические требования

* Использование RESTful API: Реализация API соответствует принципам REST.
* Использование HTTP методов: для выполнения операций CRUD следует используются соответствующие HTTP методы:
  + POST для создания
  + GET для чтения
  + UPDATE для обновления
  + DELETE для удаления

База данных: используется PostgreSQL и Redis.

**Функции, выбранные для тестирования:**

Функция create\_new\_menu получает на вход словарь значений menu, вызывает функцию create\_menu и возвращает её результат. Функция create\_menu получает на вход объект класса Menu и на выходе выводит словарь из значений title и description.

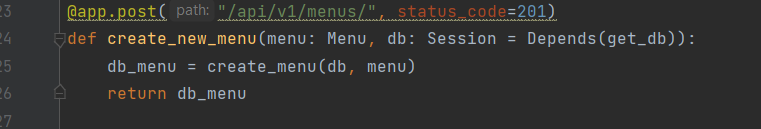


Рисунок 6 – функция добавления в меню

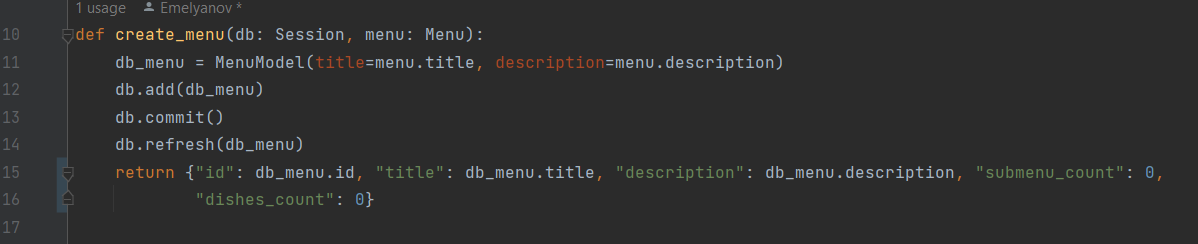


Рисунок 7 – функция добавления меню

**Тестирование функции добавления меню**

Для тестирования функции добавления меню были рассмотрены следующие случаи:

1. Функция тестирования test\_menu\_post(). Успешное добавление меню. Код состояния ответа HTTP = 201. Ответ должен содержать те же значения, которые получил в виде json, также ответ должен содержать сгенерированный id меню.

2. Функция тестирования test\_menu\_incorrect\_post(). Добавление меню с измененными атрибутами. Функция получает на вход json с измененными наименованиями атрибутов, в результате чего в результате выполнения должна получить код состояния ответа HTTP = 422

3. Функция тестирования test\_menu\_nonfull\_post().Добавление меню с не всеми необходимыми атрибутами. Функция не получила все необходимые атрибуты для добавления меню, в результате чего должен был получен код состояния ответа HTTP = 422.

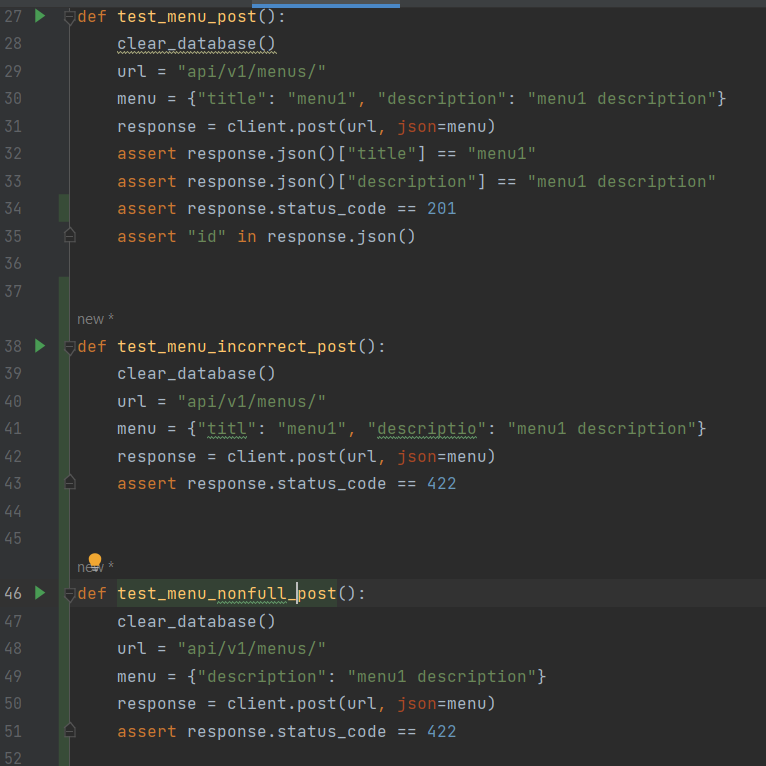


Рисунок 8 – тесты для функции добавления в меню

Тестирование успешно, функция работает корректно.

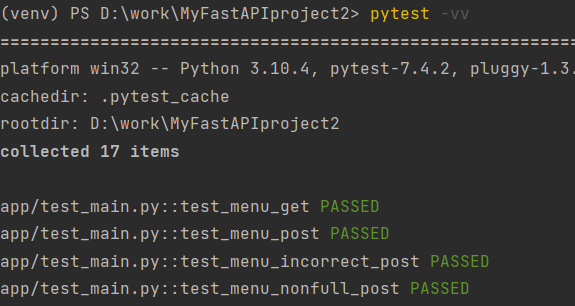


Рисунок 9 – результат тестирования

Функция update\_concreate\_menu получает на вход id обновляемого меню и словарь из атрибутов для обновления меню, вызывает функцию update\_menu и возвращает её результат. Функция update\_menu получает на вход Id обновляемого меню и словарь, на выходе выводит обновленное меню, а в случае, если такое меню отсутствует - код состояния ответа HTTP 404 и сообщение об ошибке.

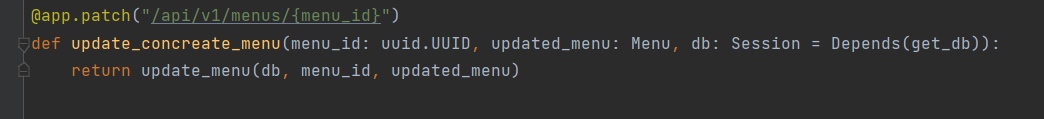


Рисунок 10 – функция обновления меню

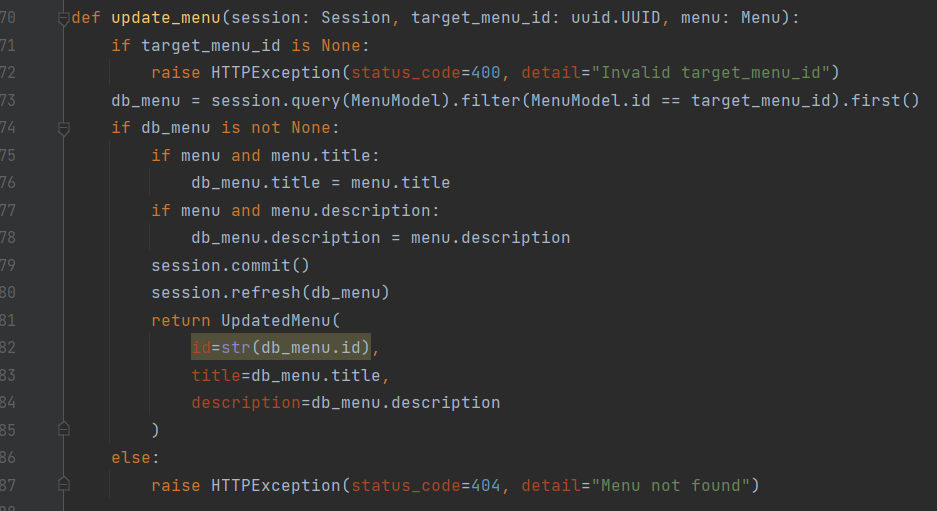


Рисунок 11 – функция обновления меню

**Тестирование функции обновления меню**

Для тестирования функции обновления меню были рассмотрены следующие случаи:

1) test\_menu\_update(). Данное тестирование проверяет работу функции при работе с корректными данными. Функция в результате работы должна получить код состояния HTTP 200 и корректные данные.

2) test\_menu\_update\_error(). Данное тестирование проверяет работу функции при работе с некорректными данными. Функция обновляет меню по несуществующему id, в результате чего должна получить код состояния ответа HTTP 404 и сообщение об ошибке.

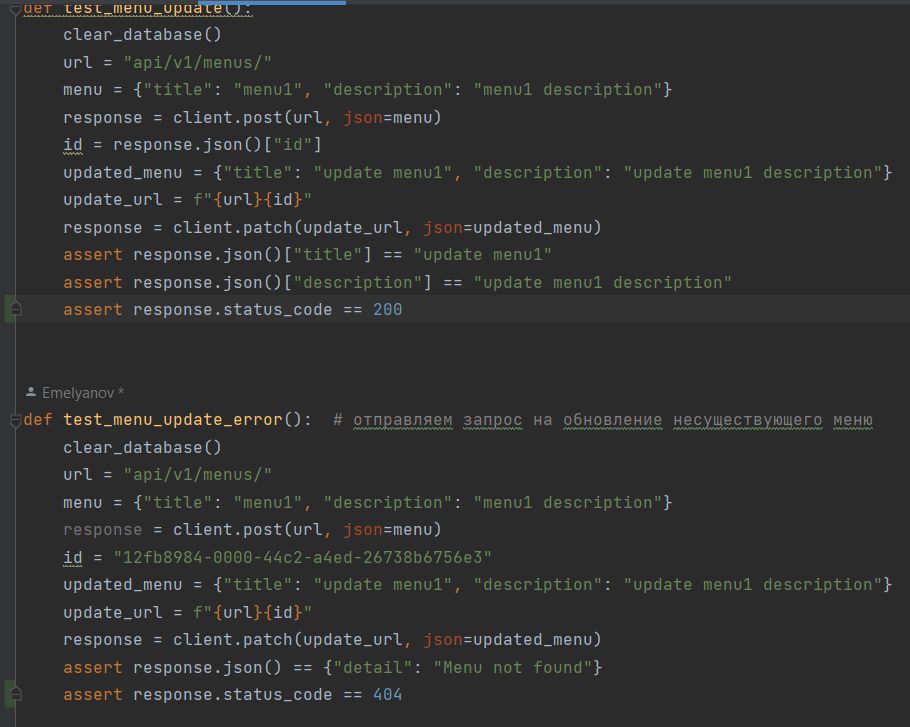


Рисунок 12 – тестирование функции обновления меню

Тестирование успешно, функция работает корректно.



Рисунок 13 – результат тестирования

1. **Документация на программный продукт №3**

Программный продукт представляет собой бекенд-приложение для новостного сайта, предоставляющее RESTful API для управления статьями. API поддерживает следующие функции:

* Создание статьи:
  + Возможность добавления новых статей.
  + Возможность получения списка всех статей.
  + Возможность получения информации о конкретной статье по его идентификатору.
* Обновление статьи:
  + Возможность изменения информации о существующих статьях, включая название, описание и содержание.
* Удаление статьи

Технические требования

* Использование RESTful API: Реализация API соответствует принципам REST.
* Использование HTTP методов: для выполнения операций CRUD следует используются соответствующие HTTP методы:
  + POST для создания
  + GET для чтения
  + UPDATE для обновления
  + DELETE для удаления

База данных: используется PostgreSQL и Redis.

**Функции, выбранные для тестирования:**

Для тестирования был выбран класс ArticleService осуществляющий процесс добавления, поиска и удаления статей. В нем представленны 4 функции.

Функция поиска всех статей возвращает массив содержащий сущности всех статей.

Функция поиска статей по id принимает номер искомой статьи и возвращает сущность искомой статьи.

Функция добавления статьи принимает сущности статьи и добавляет ее в массив существующих статей.

Функция удаления принимает id удаляемой статьи и исключает его из списка статей.

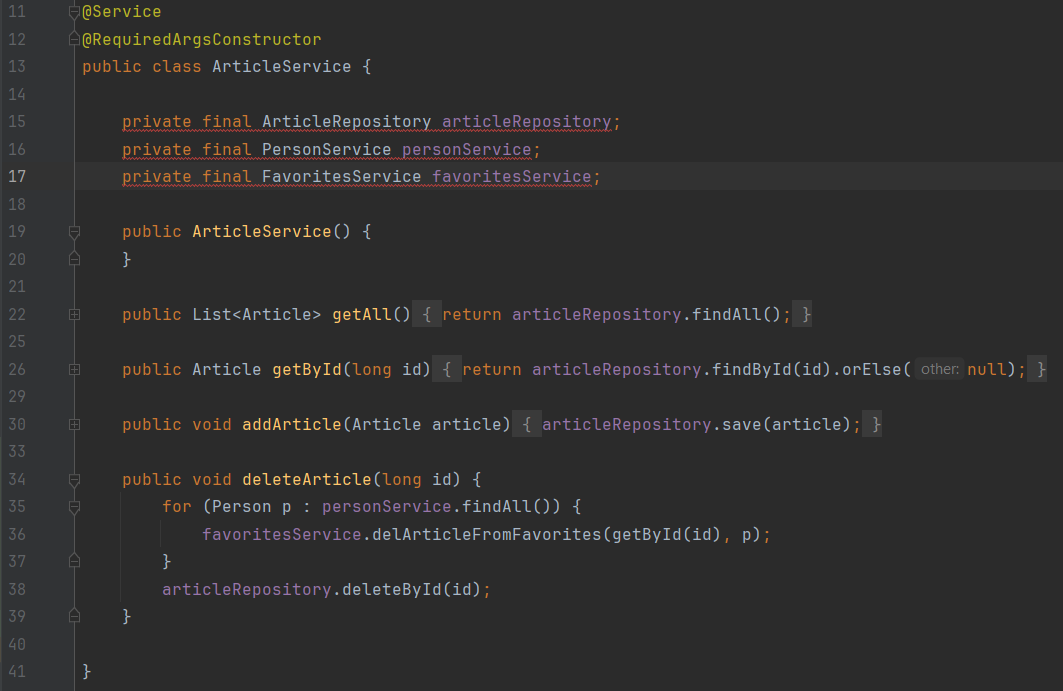


Рисунок 14 – функции класса ArticleService

**Тестирование функций**

Модульные тесты проверяют корректность работы всех функций. Проверяется корректность вывода всех статей, корректность добавления и удаления статей, а так же два случая поиска статьи по id.

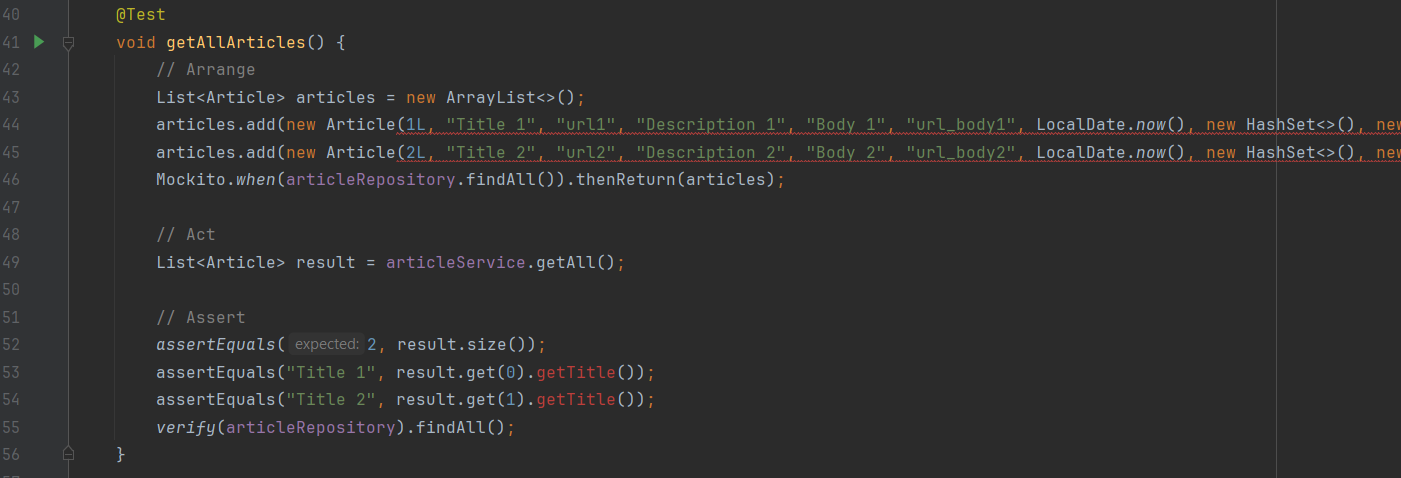
****

Рисунок 14 – тестирование функции получения статей

****

Рисунок 15 – тестирование функций получения статьи по id

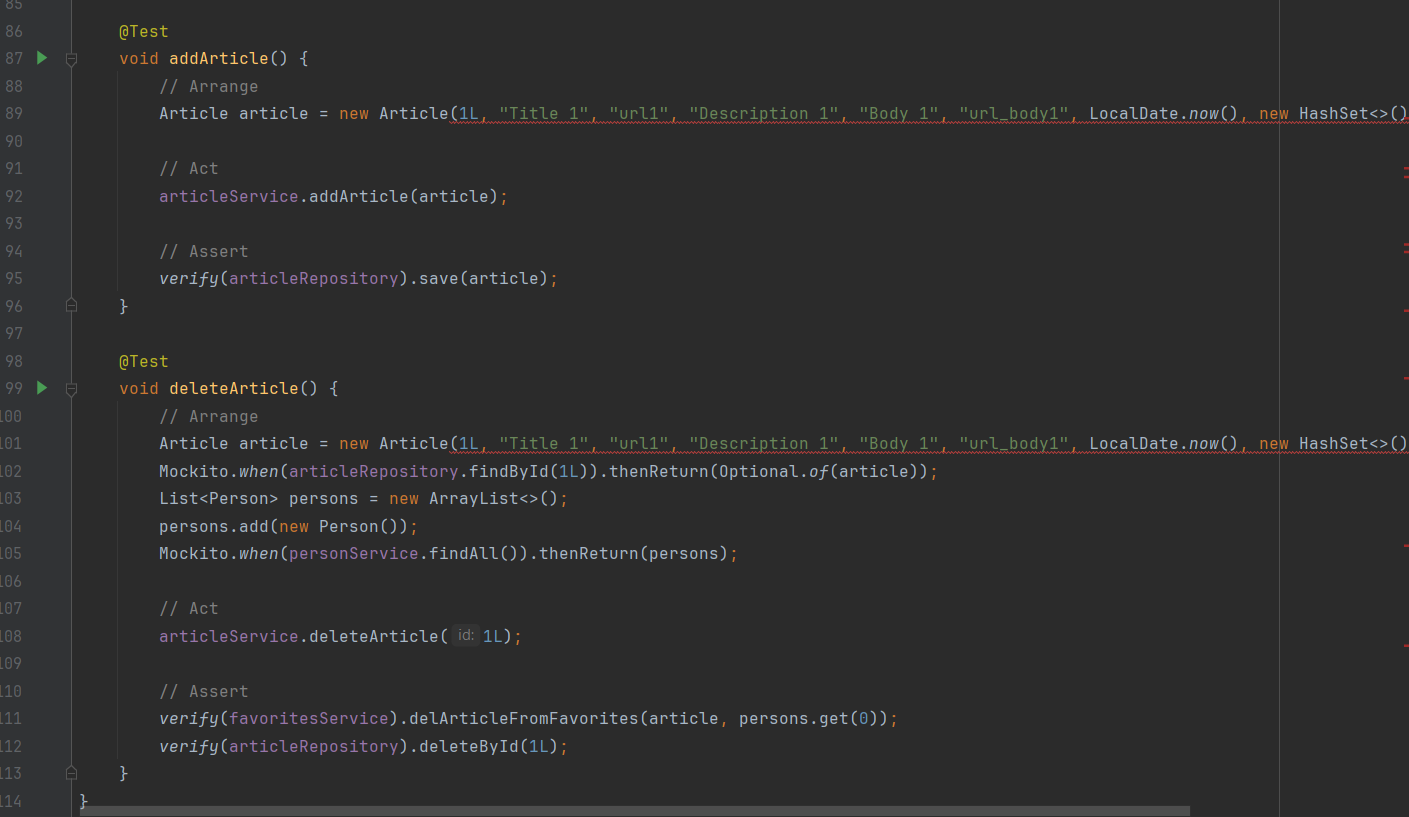


Рисунок 16 – тестирование добавления и удаления статей

**Результаты тестирования**

Все тесты успешно пройдены. Ошибок и багов найдено не было

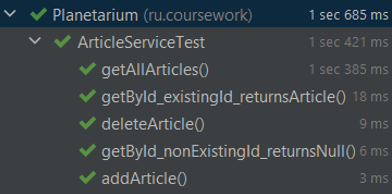


Рисунок 17 – результат тестирования

1. **Документация на программный продукт №4**

Функция createDelayFunction(func, delay) создает для функции, передаваемой в качестве аргумента, функцию обертку, которая вызовет целевую функцию с задержкой в мс, размер которой передается в качестве аргумента delay.

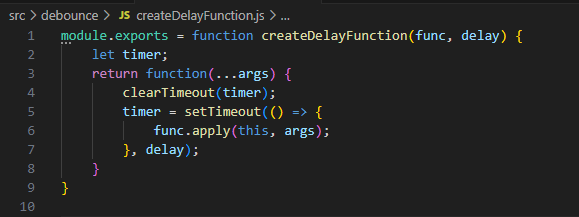


Рисунок 18

Модульные тесты проверяют корректность работы функции createDelayFunction для разных входных данных (рис. 2-3).



Рисунок 19

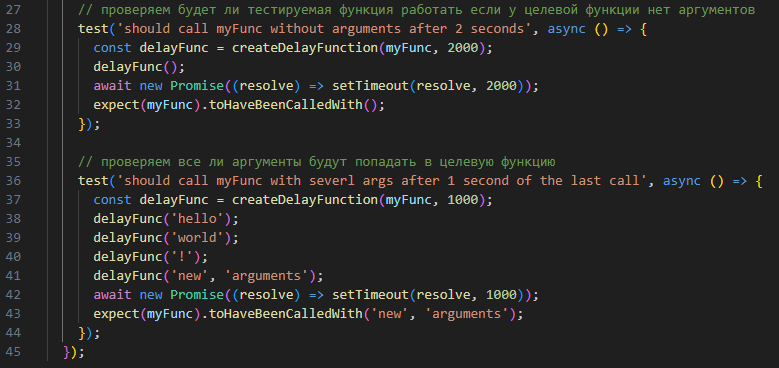


Рисунок 20

Тесты проверяют следующие сценарии:

1. should call with «hello» after 1 seconds: Проверяет, что целевая функция будет вызвана с задержкой в одну секунду.

2. should call with «!» after 0.5 seconds: Проверяет, что при нескольких вызовах функции-обертки во временных пределах задержки таймауты будут очищаться и обновляться, таким образом, что целевая функция будет вызвана только с аргументом(ами) переданными в последний вызов функции обертки.

3. should call myFunc without arguments after 2 seconds: Проверяет, что функция работает если отсутствуют аргументы передаваемые в обертку.

4. should call myFunc with several args after 1 second of the last call: Проверяет, что все аргументы будут попадать в целевую функцию

Функция getData() используется для получения и преобразования данных полученных с сервера при запросе. Получает массив пользователей в формате JSON с сервера и возвращает массив отсортированных id пользователей. Если сервер не работает обрабатывает ошибку и выбрасывает новую – «API is down». Если массив пользователей пуст, возвращает пустой массив.

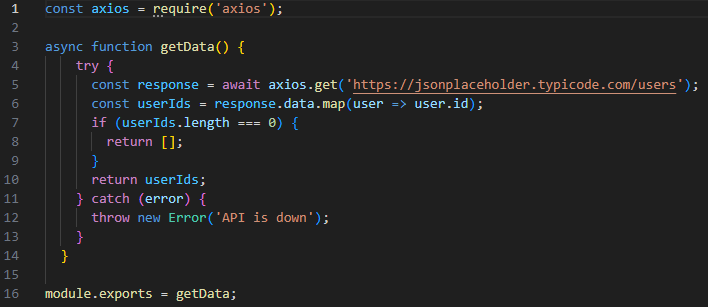


Рисунок 21

Модульные тесты проверяют корректность работы функции getData (рис. 5 - 7).



Рисунок 22



Рисунок 23

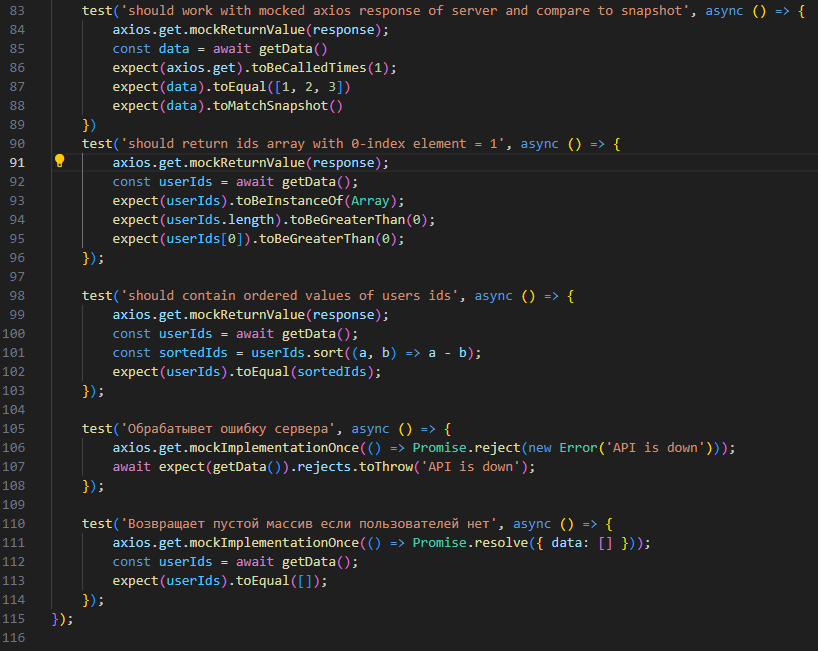


Рисунок 24

Эти тесты проверяют следующие сценарии:

1. Should work with mocked axios response of server and compare to snapshot: имитирует работу замоканного модуля axios и проверяет, что функция была вызвана только единожды, вернула правильный массив и совпадает со снапшотом (историей результатов тестирования).
2. Should return ids array with 0-index element = 1: Проверяет, что функция возвращает правильный массив id пользователей, и нулевой элемент этого массива положителен.
3. Should contain ordered values of users ids: Проверяет, что функция возвращает масив id пользователей в порядке возрастания.
4. Should treat the server error: Проверяет что при ошибке сервера, функция обрабатывает ее и выбрасывает ошибку «API is down».
5. Should return empty array if no users from server: Проверяет, что в случае, когда сервер возвращает пустой массив без пользователей, функция также возвращает пустой массив.