**Проект по автоматизации тестирования приложения “webcalculator”**

**Автор: Карболин Иван**

**email: karboliniv@gmail.com**

**Репозиторий проекта:** <https://github.com/karbolinivan/infotecs>

**Отчет Allure:** <https://karbolinivan.github.io/infotecs/>

# **Тест-кейсы**

Общее количество -110 (группы пересекаются)

* smoke - 12
* functional -30
* negative - 83
* api -77
* app - 7

# **Структура проекта**

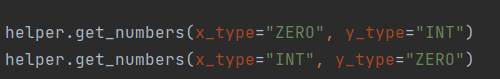
* src - ресурсы необходимые для написание тестов
  + base - описывает базовые функции фреймворка
    - assertion - описание проверок
    - client - описание запросов CRUD
    - helps - вспомогательные функции (генерация данных, получение данных)
    - logs - логирования и прикрепления артефактов в отчет Allure
  + enums - описывает кейсы и константы
    - cases - тест кейсы используемые в тестах и параметризация
    - endpoints - описывает пути запроса к приложению
    - messages - сообщения ожидаемые от сервера
    - paths - пути к файлам
  + service - описывает взаимодействие с приложением
    - api - описание запросов к приложению
    - build - управление приложением (start, stop, restart и т.д.)
    - models - модели ответа от сервера
    - settings - настройки запуска приложений
* tests - автотесты
  + тестирования функций приложения (запуск, останов, рестарт и т.д)
  + тестирование api приложения (state, options, addition, ответы от сервера и т.д.)
  + тестирование функций вычисления (проверки комбинаций валидных значений x и y)
* conftest - фикстуры управляющие запуском приложения перед началом тестирования
* pytest - настройки pytest
* requirements - библиотеки используемые в проекте

# **Автотесты**

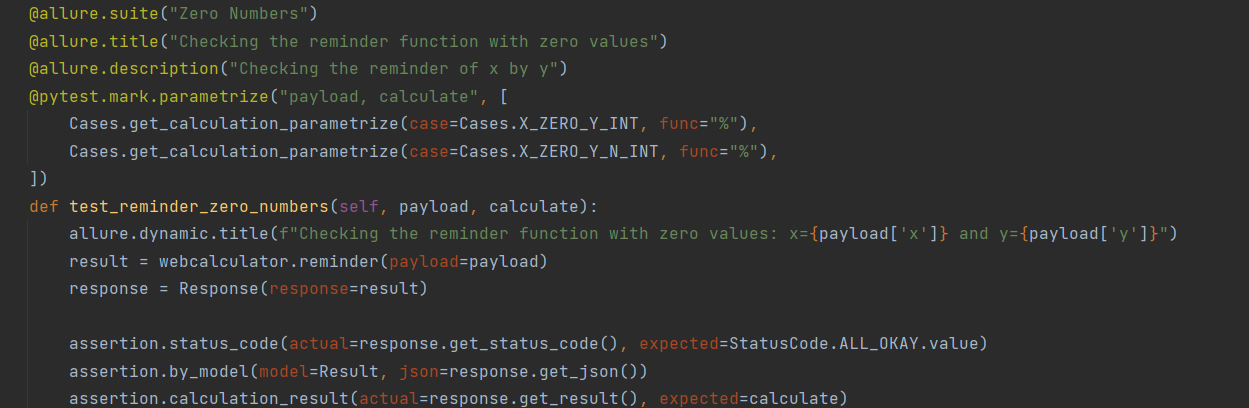
Для всех автотестов данные генерируются автоматически с использованием библиотеки random:



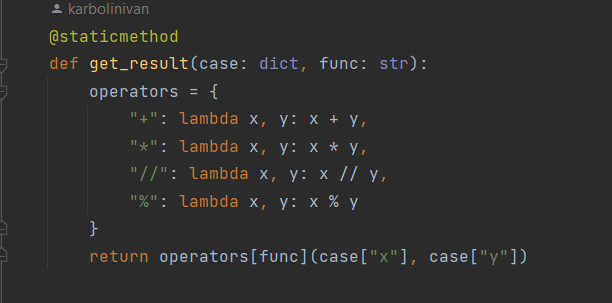
Тест-кейсы составлялись на основе комбинаций валидных и невалидных значений:



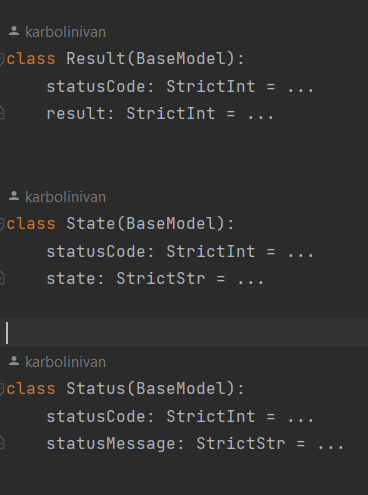
Для наборов тестов с общим результатом используется параметризация, в которую передаются наборы сгенерированных данных:



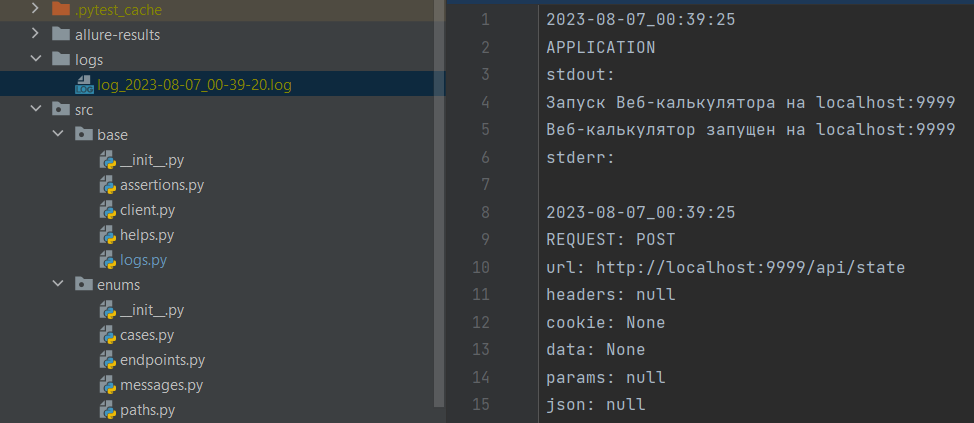
Результат вычисления приложения проверяется автоматически основе сгенерированных данных:



Ответы от сервера валидируются по моделям Pydantic:

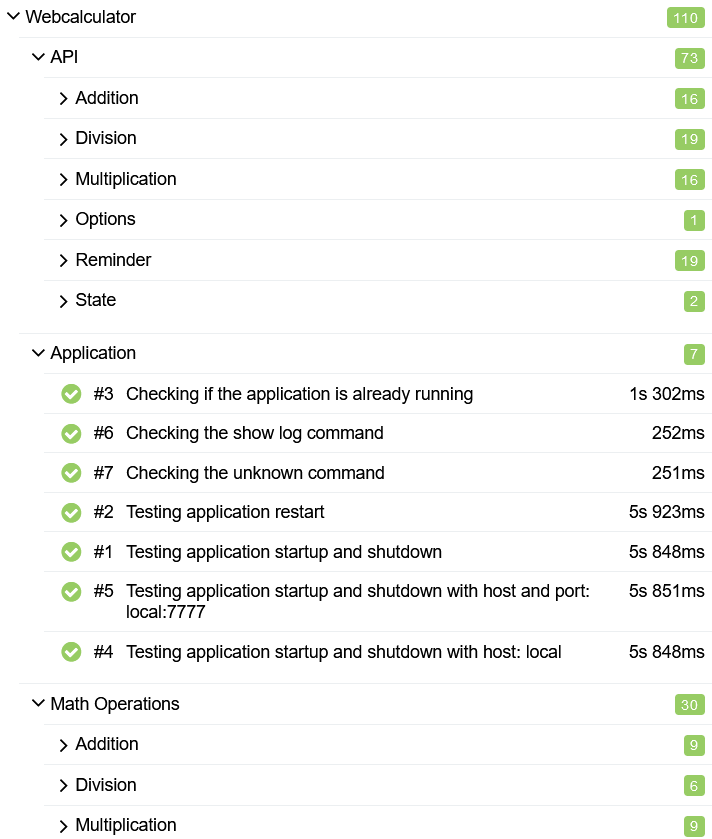


При запуске автотестов формируется общий файл с логом:

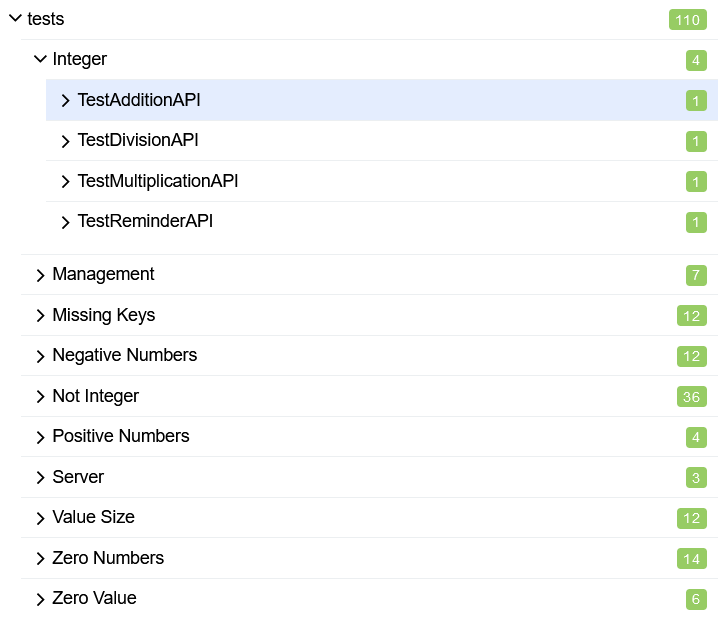


# **Отчет allure**

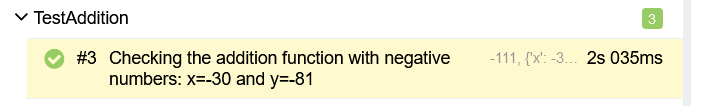
Все тесты разделены по группам:



Деление по тестовым наборам (suite):



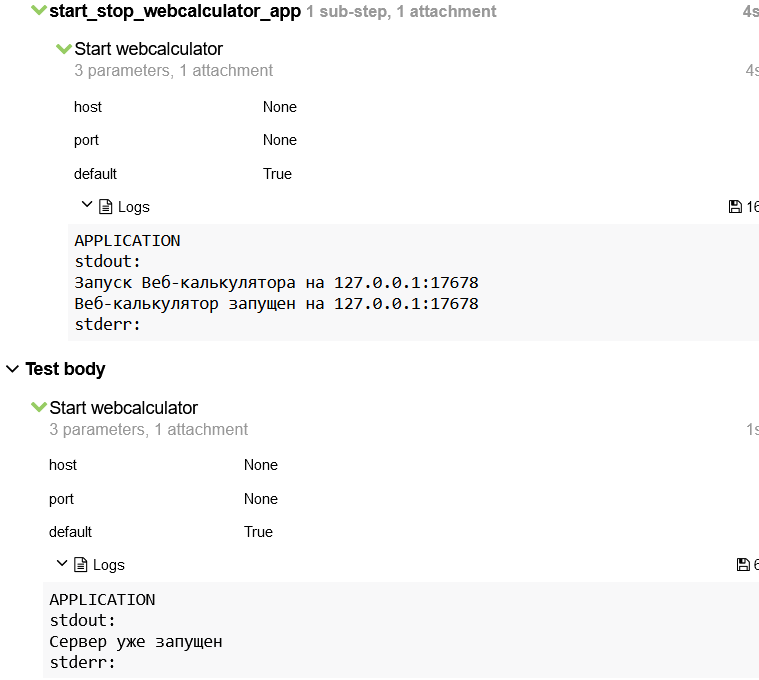
Использование динамических титулов, позволяет указывать используемые в автотесте наборы данных:



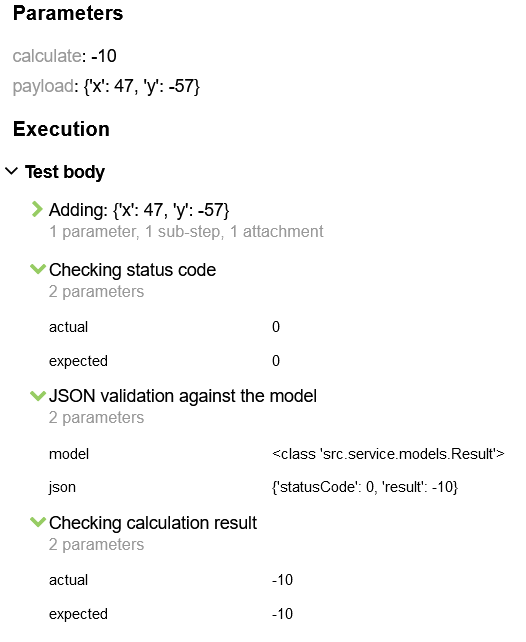
Для каждого запроса формируется лог с запросом/ответом и прикрепляется к отчету:



При действиях с приложением также пишется лог:



Осуществляются проверки результата вычисления и ответа сервера для каждого автотеста:



# **Несоответствия с документацией**

Сообщения от сервера:

|  |  |  |
| --- | --- | --- |
| **Код ошибки** | **Описание ошибки** | **Актуальные сообщения** |
| 1 | Ошибка вычисления | Ошибка вычисления |
| 2 | Не хватает ключей в теле запроса | Не указаны необходимые параметры |
| 3 | Одно из значений не является целым числом | Значения параметров должны быть целыми |
| 4 | Превышен размер одного из значений | Превышены максимальные значения параметров |
| 5 | Неправильный формат тела запроса | Не верное имя задачи или тип HTTP запроса |
| 0 | Все хорошо | Все хорошо |

При выполнении запроса по несуществующему эндпоинту неверный статус код:

Данные несоответствия не брались в учет при тестировании.

# **Запуск проекта на windows**

Предварительно должен быть уставлен scoop и allure - <https://docs.qameta.io/allure/#_windows>

1. Установка окружения:

* python3 -m venv venv

1. Активация окружения:

* venv/Scripts/activate

1. Установка зависимостей:

* pip3 install -r requirements.txt

1. Запуск отчета с сохранением отчета allure:

* pytest -s -v --alluredir=allure-results

1. Запуск отчета Allure:

* allure serve allure-results