|  |
| --- |
| Отчет по тестированию |
| Применение техники анализа граничных значений для тестирования API сайта зоомагазина |
| Петухова Ирина Сергеевна |

|  |
| --- |
| Ижевск  2025 г. |

Оглавление

[Введение 2](#_Toc207213181)

[Таблица с граничными значениями 3](#_Toc207213182)

[Тест-кейсы 4](#_Toc207213183)

[Коллекция в Postman 5](#_Toc207213184)

[Результаты тестирования 8](#_Toc207213185)

[Заключение 11](#_Toc207213186)

# Введение

В данной работе применены техники анализа граничных значений для тестирования API сайта зоомагазина. Я тестировала API для создания и обновления данных питомца в зоомагазине Petstore с применением техники анализа граничных значений к полям запросов. Для тестирования составлена таблица с граничными значениями, написаны тест-кейсы на основе данной таблицы и создана коллекция в Postman с методами создания и обновления данных питомца с примерами на основе написанных тест-кейсов.

# Таблица с граничными значениями

В таблице граничных значений содержатся 6 столбцов: «Тип данных», «Поле», «Граничные значения», «Данные», «Ожидаемый код ответа» и «Маркировка тестов».

В солбце «Тип данных» указано, к какому типу данных относится поле.

Столбец «Поле» указывает, для какого поля написаны проверки.

В столбце «Граничные значения» указано, для какого типа значений проводятся проверки: для минимально допустимого, максимально допустимого, значения на границе – 1 или + 1.

Столбец «Данные» — это те значения, которые нужно использовать при прохождении тест-кейсов, подставляя их в тело запроса.

Столбец «Ожидаемый код ответа» показывает, какой код ответа должен приходить в ответе на запрос.

В столбце «Маркировка тестов» указано, тест позитивный или негативный.

Примечания к заполнению таблицы граничных значений:

В Swagger указано, что у id питомца тип данных integer($int64), значит максимальное допустимое значение будет равно 9223372036854775807.

Также в Swagger не указана максимальная длина для строковых типов данных, поэтому предположим, что она будет 25 символов.

# Тест-кейсы

На основе таблицы граничных значений я составила 15 тест-кейсов, проверяющих методы создания и обновления данных питомца в Petstore.

В сущности Pet (питомец) тестируются методы POST и PUT, то есть добавление нового питомца в магазин и обновление данных о существующем питомце.

Для тестирования граничных значений применяются минимальная и максимальная граница и их приграничная область.

В тест-кейсах присутствуют такие поля, как ID и название тест-кейса, предусловия для метода PUT, URL и метод запроса, тестируемые граничные значения и тело запроса в формате JSON, ожидаемый код ответа и содержимое, также присутствует комментарии о цели тестирования.

# Коллекция в Postman

В программе Postman создала коллекцию Swagger Petstore BVA Project (BVA - анализ граничных значений).

В коллекции в разделе Variables добавила переменную baseUrl со значением <https://petstore.swagger.io/v2>, в раздел Scripts добавила тесты на проверку статус кода и времени получения ответа от сервера.

В коллекции создала папку pet для сущности Pet (животное) с методами POST (добавление нового питомца в магазин) и PUT (обновление данных о существующем питомце).

В методе POST ввела переменную {{baseUrl}}, содержащую в себе URL зоомагазина и путь к ресурсу /pet.

На вкладке Headers (Заголовки) добавила Content-Type: application/json, чтобы сообщить серверу, что данные в теле запроса отформатированы в формате JSON.

На вкладке Body (Тело) выбрала raw и тип JSON, написала тело запроса:

{

  "id": 0,

  "category": {

    "id": 0,

    "name": "string"

  },

  "name": "doggie",

  "photoUrls": [

    "string"

  ],

  "tags": [

    {

      "id": 0,

      "name": "string"

    }

  ],

  "status": "available"

}

Отправила запрос, нажав кнопку Send и получила ответ от сервера с телом ответа:

{

    "id": 9223372036854741000,

    "category": {

        "id": 0,

        "name": "string"

    },

    "name": "doggie",

    "photoUrls": [

        "string"

    ],

    "tags": [

        {

            "id": 0,

            "name": "string"

        }

    ],

    "status": "available"

}

и кодом ответа 200.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 Создание коллекции в Postman

После того, как я убедилась, что запрос работает, в методе Post я создала готовые к отправке запросы, которые служат примерами тестирования различных сценариев, содержащие данные из ранее созданной таблицы граничных значений.

Для этого в методе Post изменила тело запроса, исходя из тест-кейса TC\_001, нажала кнопку Send, затем нажала кнопку Save Response. В методе Post создался пример, в котором присутствует только что отправленный запрос и полученный ответ от сервера со статус-кодом, заголовками и телом ответа. Данный пример я переименовала исходя из названия тест-кейса.

То же самое проделала с остальными тест-кейсами и с методом PUT.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 Создание примера внутри коллекции

# Результаты тестирования

По результатам тестирования API для создания и обновления данных питомца с применением техники анализа граничных значений из 15 тест-кейсов было успешных 10 и 5 с дефектами. Блокирующих или критических дефектов не зафиксировано. Все дефекты, включая значительные, выявлены снаружи границ диапазона.

|  |  |
| --- | --- |
| **Показатель** | **Значение** |
| Проверено тест-кейсов | 15 |
| Успешных тестов | 10 (67%) |
| Найдено дефектов | 5 |
| Значительные дефекты | 2 |
| Незначительные дефекты | 2 |
| Тривиальные дефекты | 1 |

Рисунок 3 Диаграмма состояния дефектов

Чек-лист по найденным дефектам:

* Значительные дефекты.

1. Название теста: TC\_002 "Добавление нового питомца в магазин со значением id на границе - 1 от допустимого."

Результат: ❌ FAIL

Комментарий: при запросе с id = 0 в результате код ответа 200, значение id = 9223372036854775807**,** ожидался код ответа 400.

1. Название теста: TC\_012 "Обновление данных о существующем питомце c большим количеством элементов в category."

Результат: ❌ FAIL

Комментарий: при запросе c большим количеством элементов в category в результате код ответа 200, не отображаются элементы, выходящие за диапазон значений category**,** ожидался код ответа: 400.

* Незначительные дефекты.

1. Название теста: TC\_007 "Добавление нового питомца в магазин со значением category с минимальной длиной - 1 от допустимых."

Результат: ❌ FAIL

Комментарий: при запросе при пустом значении объекта category в объекте выводится "id": 0**,** ожидаемый результат - код ответа: 400.

1. Название теста: TC\_010 "Обновление данных о существующем питомце c пустым значением status."

Результат: ❌ FAIL

Комментарий: при запросе при пустом значении поля status в ответе выводится status = ""**,** ожидаемый результат - код ответа: 400.

* Тривиальные дефекты.

1. Название теста: TC\_003 "Добавление нового питомца в магазин со значением tags: id на границе + 1 от допустимого."

Результат: ❌ FAIL

Комментарий: при запросе с tags: id = 9223372036854775808 в результате код ответа 500**,** ожидался код ответа 400.

# Заключение

В данной работе я протестировала API для создания и обновления данных питомца в зоомагазине Petstore с применением техники анализа граничных значений к полям запросов. В процессе тестирования была составлена таблица с тестовыми данными по технике анализа граничных значений, созданы 15 тест-кейсов, которые содержат информацию о том, как необходимо тестировать методы API и создана коллекция в Postman с методами POST и PUT с примерами на основе написанных тест-кейсов.

По итогам тестирования из 15 проверок было выявлено 5 дефектов, 2 из которых являются значительными, еще 2 незначительными и 1 тривиальный. Блокирующих и критических дефектов не обнаружено. Можно сделать вывод, что API в целом работоспособен, но требует доработок для значений, выходящих за границы диапазона.