Отчет о результатах тестирования API для сущности Pet

Объект тестирования:
API сущности **Pet** на сайте Swagger Petstore. Тестируются основные операции CRUD.

Цель тестирования:
Проверить корректную работу эндпоинтов API для сущности **Pet**, а также проверить обработку как позитивных, так и негативных сценариев. Убедиться, что API правильно обрабатывает запросы, возвращает корректные ответы и коды ошибок.

Окружение тестирования:

- **URL:** https://petstore.swagger.io/v2
- **Методы:** POST, GET, PUT, DELETE
- **Инструменты тестирования:**
  - Библиотека Python: requests
  - Фреймворк для тестирования: pytest
- **Операционная система:** Mac OS Sonoma 14.6.1
- **Python:** 3.12.3
- **Дата проведения:** 30 сентября 2024 года

Описание тестов:

1. **Тест на создание питомца (POST /pet)**
   - **Описание:** Тест проверяет успешное создание нового питомца с уникальными параметрами.
   - **Ожидаемый результат:** Код ответа 200 OK, данные о питомце совпадают с переданными в запросе.
   - **Фактический результат:** Код ответа 200 OK, питомец успешно создан, данные совпадают с ожидаемыми.
2. **Тест на получение питомца по ID (GET /pet/{petId})**
   - **Описание:** Тест проверяет успешное получение данных о питомце по его уникальному ID.
   - **Ожидаемый результат:** Код ответа 200 OK, возвращены корректные данные о питомце.
   - **Фактический результат:** Код ответа 200 OK, данные питомца соответствуют ID и содержимому.
3. **Тест на обновление статуса питомца (PUT /pet)**
   - **Описание:** Тест проверяет возможность изменения статуса питомца (с "available" на "sold").
   - **Ожидаемый результат:** Код ответа 200 OK, статус питомца обновлен на "sold".
   - **Фактический результат:** Код ответа 200 OK, статус питомца успешно обновлен на "sold".
4. **Тест на удаление питомца (DELETE /pet/{petId})**
   - **Описание:** Тест проверяет удаление питомца по его уникальному ID.
   - **Ожидаемый результат:** Код ответа 200 OK, питомец удален, повторный запрос на получение возвращает код 404 Not Found.
   - **Фактический результат:** Код ответа 200 OK, питомец удален, повторный запрос возвращает код 404 Not Found.
5. **Негативный тест на получение несуществующего питомца (GET /pet/{non-existent-id})**

- **Описание:** Тест проверяет, что запрос на получение данных о несуществующем питомце возвращает корректную ошибку.
- **Ожидаемый результат:** Код ответа 404 Not Found.
- **Фактический результат:** Код ответа 404 Not Found.

Тестовые данные:

- **Pet ID:** 12345
- **Имя питомца:** Rex
- **Категория:** Dogs
- **Фото URL:** ["https://example.com/photo"]
- **Теги:** [{"id": 1, "name": "tag1"}]
- **Статус:** «available» на «sold» (при обновлении)

Результаты тестирования:

№	Тест	Ожидаемый результат	Фактический результат	Статус
1	Создание питомца (POST /pet)	200 OK, питомец создан	200 OK, питомец создан	Успешно
2	Получение питомца (GET /pet/{id})	200 OK, данные верны	200 OK, данные верны	Успешно
3	Обновление питомца (PUT /pet)	200 OK, статус обновлен	200 OK, статус обновлен	Успешно
4	Удаление питомца (DELETE /pet/{id})	200 OK, питомец удален	200 OK, питомец удален	Успешно
5	Негативный тест (GET /pet/{invalid-id})	404 Not Found	404 Not Found	Успешно

Выводы:

1. **Все тесты прошли успешно**. Система корректно обрабатывает операции создания, получения, обновления и удаления сущностей.
2. **Негативные тесты** также работают корректно: попытка получения несуществующего питомца возвращает ожидаемый код ошибки 404.
3. **API стабильно**: ни один запрос не вызвал ошибки сервера, время ответа соответствует ожиданиям (в пределах 1 секунды).

Рекомендации:

1. Провести дополнительные тесты для работы с большим количеством питомцев, чтобы проверить производительность системы.
2. Проверить работу API с различными типами данных (например, нагрузочное тестирование с большими объемами информации или с некорректными форматами данных).

Заключение:
Функциональность API для сущности **Pet** работает в полном соответствии с ожиданиями. Все тестовые сценарии выполнены успешно, результаты соответствуют требованиям.

Отчет о результатах тестирования API для сущности User

Объект тестирования:
API сущности **User** на сайте Swagger Petstore. Тестируются основные операции CRUD.

Цель тестирования:
Проверить корректную работу эндпоинтов API для сущности **User**, а также проверить обработку как позитивных, так и негативных сценариев. Убедиться, что API правильно обрабатывает запросы, возвращает корректные ответы и коды ошибок.

Окружение тестирования:

- **URL:** https://petstore.swagger.io/v2
- **Методы:** POST, GET, PUT, DELETE
- **Инструменты тестирования:**
  - Библиотека Python: requests
  - Фреймворк для тестирования: pytest
- **Операционная система:** Mac OS Sonoma 14.6.1
- **Python:** 3.12.3
- **Дата проведения:** 30 сентября 2024 года

Описание тестов:

1. **Тест на создание пользователя (POST /user)**
   - **Описание:** Тест проверяет успешное создание нового пользователя с уникальными параметрами.
   - **Ожидаемый результат:** Код ответа 200 OK, данные о пользователе совпадают с переданными в запросе.
   - **Фактический результат:** Код ответа 200 OK, пользователь успешно создан, данные совпадают с ожидаемыми.
2. **Тест на получение пользователя по имени (GET /user/{username})**
   - **Описание:** Тест проверяет успешное получение данных о пользователе по его уникальному имени.
   - **Ожидаемый результат:** Код ответа 200 OK, возвращены корректные данные о пользователе.
   - **Фактический результат:** Код ответа 200 OK, данные пользователя соответствуют переданному имени.
3. **Тест на обновление пользователя (PUT /user/{username})**
   - **Описание:** Тест проверяет возможность обновления данных пользователя.
   - **Ожидаемый результат:** Код ответа 200 OK, данные пользователя обновлены.
   - **Фактический результат:** Код ответа 200 OK, данные пользователя успешно обновлены.
4. **Тест на удаление пользователя (DELETE /user/{username})**
   - **Описание:** Тест проверяет удаление пользователя по его уникальному имени.
   - **Ожидаемый результат:** Код ответа 200 OK, пользователь удален, повторный запрос на получение возвращает код 404 Not Found.
   - **Фактический результат:** Код ответа 200 OK, пользователь удален, повторный запрос возвращает код 404 Not Found.
5. **Негативный тест на получение несуществующего пользователя (GET /user/{non-existent-username})**

- **Описание:** Тест проверяет, что запрос на получение данных о несуществующем пользователе возвращает корректную ошибку.
- **Ожидаемый результат:** Код ответа 404 Not Found.
- **Фактический результат:** Код ответа 404 Not Found.

Тестовые данные:

- **Username:** johndoe
- **Первое имя:** John
- **Фамилия:** Doe
- **Email:** johndoe@example.com
- **Новый Email:** newemail@example.com
- **Пароль:** password123
- **Телефон:** +1234567890
- **Статус пользователя:** active

Результаты тестирования:

№	Тест	Ожидаемый результат	Фактический результат	Статус
1	Создание пользователя (POST /user)	200 OK, пользователь создан	200 OK, пользователь создан	Успешно
2	Получение пользователя (GET /user/{username})	200 OK, данные верны	200 OK, данные верны	Успешно
3	Обновление пользователя (PUT /user/{username})	200 OK, данные обновлены	200 OK, данные обновлены	Успешно
4	Удаление пользователя (DELETE /user/{username})	200 OK, пользователь удален	200 OK, пользователь удален	Успешно
5	Негативный тест (GET /user/{non-existent-username})	404 Not Found	404 Not Found	Успешно

Выводы:

1. **Все тесты прошли успешно.** API для управления пользователями корректно обрабатывает операции создания, получения, обновления и удаления.
2. **Негативные тесты работают корректно.** Попытка получения несуществующего пользователя возвращает код ошибки 404.
3. **API стабильно:** ни один запрос не вызвал ошибки сервера, и время отклика соответствует ожиданиям.

Рекомендации:

1. Провести тесты на производительность, включая массовое создание пользователей.
2. Проверить работу API с различными типами данных, включая неверные или частично корректные запросы.

Отчет о результатах тестирования API для сущности Store

Объект тестирования:
API сущности **Store** на сайте Swagger Petstore. Тестируются операции по управлению заказами в магазине.

Цель тестирования:
Проверить корректную работу эндпоинтов API для сущности **Store**, а также обработку позитивных и негативных сценариев. Убедиться, что API корректно обрабатывает запросы, возвращает правильные ответы и коды ошибок.

Окружение тестирования:

- **URL:** https://petstore.swagger.io/v2
- **Методы:** POST, GET, DELETE
- **Инструменты тестирования:**
  - Библиотека Python: requests
  - Фреймворк для тестирования: pytest
- **Операционная система:** Mac OS Sonoma 14.6.1
- **Python:** 3.12.3
- **Дата проведения:** 30 сентября 2024 года

Описание тестов:

1. **Тест на создание заказа (POST /store/order)**
   - **Описание:** Тест проверяет успешное создание заказа с уникальными параметрами.
   - **Ожидаемый результат:** Код ответа 200 OK, данные о заказе совпадают с переданными в запросе.
   - **Фактический результат:** Код ответа 200 OK, заказ успешно создан, данные совпадают с ожидаемыми.
2. **Тест на получение заказа по ID (GET /store/order/{orderId})**
   - **Описание:** Тест проверяет успешное получение данных о заказе по его уникальному ID.
   - **Ожидаемый результат:** Код ответа 200 OK, возвращены корректные данные о заказе.
   - **Фактический результат:** Код ответа 200 OK, данные заказа соответствуют переданному ID.
3. **Тест на удаление заказа (DELETE /store/order/{orderId})**
   - **Описание:** Тест проверяет удаление заказа по его уникальному ID.
   - **Ожидаемый результат:** Код ответа 200 OK, заказ удален, повторный запрос на получение возвращает код 404 Not Found.
   - **Фактический результат:** Код ответа 200 OK, заказ удален, повторный запрос возвращает код 404 Not Found.
4. **Тест на получение инвентаря (GET /store/inventory)**
   - **Описание:** Тест проверяет успешное получение инвентаря магазина.
   - **Ожидаемый результат:** Код ответа 200 OK, возвращены корректные данные инвентаря.
   - **Фактический результат:** Код ответа 200 OK, данные инвентаря успешно получены.
5. **Негативный тест на получение несуществующего заказа (GET /store/order/{non-existent-id})**
   - **Описание:** Тест проверяет, что запрос на получение данных о несуществующем заказе возвращает корректную ошибку.

- **Ожидаемый результат:** Код ответа 400 Bad request.
- **Фактический результат:** Код ответа 400 Bad request.

Тестовые данные:

- **Order ID:** 98765
- **Pet ID:** 12345
- **Количество:** 2
- **Статус:** placed
- **Поле complete:** true

Результаты тестирования:

№	Тест	Ожидаемый результат	Фактический результат	Статус
1	Создание заказа (POST /store/order)	200 OK, заказ создан	200 OK, заказ создан	Успешно
2	Получение заказа (GET /store/order/{orderId})	200 OK, данные верны	200 OK, данные верны	Успешно
3	Удаление заказа (DELETE /store/order/{orderId})	200 OK, заказ удален	200 OK, заказ удален	Успешно
4	Получение инвентаря (GET /store/inventory)	200 OK, инвентарь получен	200 OK, инвентарь получен	Успешно
5	Негативный тест (GET /store/order/{non-existent-id})	400 Bad request	400 Bad request	Успешно

Выводы:

1. **Все тесты прошли успешно.** Система корректно обрабатывает операции создания, получения и удаления заказов, а также предоставляет информацию об инвентаре.
2. **Негативные тесты работают корректно:** запрос на получение несуществующего заказа возвращает код ошибки 400.
3. **API стабильно:** ни один запрос не вызвал ошибки сервера, и время отклика соответствует ожиданиям.

Рекомендации:

1. Провести тесты на производительность с большим количеством заказов для проверки масштабируемости.
2. Тестировать API с разными статусами заказа для проверки всех возможных сценариев (например, отмененные заказы).