ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Разработка системы управления тестированием с автоматической генерацией тест-кейсов для API на основе спецификаций

«СОГЛАСОВАНО»

студентом 4 курса ИВТ 1 группы

Воложанин Владислав Олегович

_____

Санкт-Петербург
2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В условиях стремительного развития программного обеспечения и возрастающей сложности веб-сервисов, эффективное тестирование API становится критически важным элементом разработки качественных программных продуктов. Современные компании сталкиваются с необходимостью оптимизации процессов тестирования, сокращения времени на создание тест-кейсов и повышения охвата тестирования. Автоматизация генерации тестовых сценариев на основе спецификаций API представляет собой перспективное решение этих задач.

Данное техническое задание описывает требования к разработке системы управления тестированием, которая позволит автоматически генерировать тест-кейсы на основе спецификаций API. Система предназначена для команд разработки и тестирования программного обеспечения, работающих с REST API, и будет использоваться для автоматизации процесса создания, управления и выполнения тестовых сценариев. Это решение призвано значительно сократить время на разработку тестов, повысить качество тестирования и обеспечить более полный охват функциональности API.

НАЗНАЧЕНИЕ РАЗРАБОТКИ

Система предназначена для команд разработки и тестирования, работающих с REST API. Она будет предоставлять следующие возможности:

- Автоматическая генерация тест-кейсов на основе загруженных спецификаций API в форматах OpenAPI/Swagger, включая позитивные и негативные сценарии тестирования всех эндпоинтов;
- Управление тестовыми наборами с возможностью организации тест-кейсов по проектам, категориям и приоритетам, а также редактирования и дополнения автоматически сгенерированных тестов;
- Выполнение сгенерированных тестов с формированием подробных отчетов о результатах тестирования, включая статистику успешных и неуспешных тестов, время выполнения и подробные логи;
- Отслеживание изменений в спецификациях API с автоматическим обновлением соответствующих тест-кейсов и уведомлением команды о необходимости проверки измененной функциональности;
- Интеграция с популярными системами непрерывной интеграции (CI/CD) для автоматического запуска тестов при обновлении кодовой базы или спецификаций API.

2.1 Цель разработки

Целью разработки является создание автоматизированной системы управления тестированием, которая будет эффективно генерировать и управлять тест-кейсами для API на основе их спецификаций. Система должна стать надежным инструментом для команд разработки и тестирования, значительно сокращая время на создание тестовых сценариев, повышая качество тестирования и обеспечивая максимальный охват функциональности API. Разрабатываемое решение призвано оптимизировать процесс тестирования, минимизировать человеческие ошибки при создании тестов и ускорить выявление потенциальных проблем в работе API.

2.2 Обоснование необходимости разработки

Разработка системы управления тестированием с автоматической генерацией тест-кейсов является необходимым шагом для оптимизации процесса тестирования API в современных условиях разработки программного обеспечения. В частности, создание этой системы имеет несколько оснований:

- Дипломная работа - разработка данной системы является проектом дипломной работы, что способствует углубленному изучению современных подходов к автоматизированному тестированию, работе со спецификациями API и развитию навыков в области разработки систем автоматизации;
- Актуальная потребность в автоматизации - в современных условиях быстрой разработки и частых изменений API существует острая необходимость в автоматизации создания и поддержки тестовых сценариев. Ручное создание и обновление тест-кейсов требует значительных временных затрат и подвержено человеческим ошибкам;
- Оптимизация ресурсов - автоматическая генерация тест-кейсов позволит существенно сократить время на разработку тестов, позволяя командам сосредоточиться на более сложных аспектах тестирования и улучшении качества продукта.

ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

3.1 Структуры данных

3.1.1 Формат спецификаций API

Система должна поддерживать работу со следующими форматами спецификаций API:

OpenAPI/Swagger спецификация (версии 2.0 и 3.0), содержащая:

- Описание доступных эндпоинтов
- HTTP методы
- Параметры запросов
- Форматы запросов и ответов
- Схемы данных
- Примеры данных
- Описание ошибок и кодов ответов

3.1.2 Структура тест-кейсов

Тест-кейсы должны содержать следующую информацию:

1. Уникальный идентификатор теста
2. Название теста
3. Описание теста
4. Тип теста (позитивный/негативный)
5. Приоритет выполнения
6. Предусловия
7. Шаги выполнения:
    - URL эндпоинта
    - HTTP метод
    - Заголовки запроса
    - Параметры запроса
    - Тело запроса
8. Ожидаемые результаты:
    - Код ответа
    - Формат ответа
    - Схема валидации ответа
9. Статус выполнения

10. Время последнего запуска
11. История выполнения

3.1.3 Данные о результатах тестирования

Система должна хранить и обрабатывать следующие данные о результатах выполнения тестов API:

1. Общая информация о прогоне тестов:
    - Уникальный идентификатор тестового прогона
    - Дата и время начала выполнения
    - Дата и время завершения выполнения
    - Версия тестируемого API
    - Общее количество выполненных тестов
    - Количество успешных тестов
    - Количество неуспешных тестов
    - Количество пропущенных тестов
    - Общее время выполнения всех тестов
2. Информация по каждому выполненному тест-кейсу:
    - Идентификатор тест-кейса
    - Название и описание теста
    - Статус выполнения (успешно/неуспешно/пропущено)
    - Время начала и окончания выполнения
    - Отправленный запрос (URL, метод, заголовки, параметры, тело)
    - Полученный ответ (код статуса, заголовки, тело)
    - Описание ошибки (если тест не пройден)
    - Стек вызовов (в случае технической ошибки)
3. Метрики и статистика:
    - Процент успешных тестов
    - Процент покрытия API тестами
    - Среднее время выполнения теста
    - Распределение ошибок по типам
    - Тренды успешности выполнения
    - Статистика по времени отклика API
    - История изменений результатов по каждому тест-кейсу

Данные о результатах тестирования должны быть доступны для просмотра в веб-интерфейсе системы и предоставляться в форматах, удобных для анализа и формирования отчетов (JSON, CSV, PDF).

3.2 Генерация тест-кейсов

3.2.1 Алгоритм генерации тестов

Система должна автоматически генерировать тест-кейсы на основе спецификации API, следуя следующему алгоритму:

1. Анализ спецификации API:
    - Извлечение всех доступных эндпоинтов
    - Определение HTTP методов для каждого эндпоинта
    - Анализ параметров запросов
    - Анализ форматов данных запросов и ответов
2. Формирование базовых тест-кейсов:
    - Создание позитивных тестов для каждого эндпоинта
    - Генерация тестовых данных на основе схем
    - Определение ожидаемых ответов
3. Генерация граничных случаев и негативных сценариев

3.2.2 Типы генерируемых тестов

Система должна генерировать следующие типы тест-кейсов:

1. Позитивные тесты:
    - Проверка успешного выполнения запросов
    - Валидация корректных данных
    - Проверка всех допустимых комбинаций параметров
2. Негативные тесты:
    - Проверка обработки некорректных данных
    - Тестирование отсутствующих параметров
    - Проверка некорректных форматов данных
3. Тесты безопасности:
    - Проверка аутентификации
    - Тестирование авторизации
    - Валидация токенов доступа

3.2.3 Валидация данных

Система должна обеспечивать следующие проверки при генерации тестов:

1. Валидация входных данных:
    - Проверка соответствия типов данных
    - Валидация обязательных полей
    - Проверка форматов данных
2. Валидация выходных данных:
    - Проверка структуры ответа
    - Валидация статус-кодов
    - Проверка заголовков ответа
3. Дополнительные проверки:
    - Валидация бизнес-логики
    - Проверка зависимостей между полями
    - Валидация сложных условий

Система должна позволять настраивать правила генерации тестов и добавлять пользовательские шаблоны для специфических случаев тестирования.

3.3 Требования к пользовательскому интерфейсу

3.3.1 Навигация

Переход между страницами приложения будет осуществляться через верхнюю панель навигации, которая будет включать следующие разделы:

- Проекты - отображение всех тестовых проектов и их статусов;
- Тест-кейсы - просмотр и управление тестовыми сценариями;
- Отчеты - доступ к результатам тестирования и аналитике;
- Пользователь может быстро переключаться между этими разделами с помощью меню в верхней части экрана.

3.3.2 Панель управления

На панели управления будет отображаться основная информация и элементы управления:

- Статистика выполнения тестов;
- Количество активных проектов;
- Количество тест-кейсов;

- Результаты последних запусков;
- Критические ошибки; при нажатии на элементы статистики будет открываться подробная информация по выбранному показателю.

3.3.3 Страница проектов

На странице проектов будет отображаться список проектов с карточками, каждая из которых включает:

- Название проекта;
- Статус проекта;
- Количество тест-кейсов;
- Дата последнего обновления;
- Процент успешных тестов;
- Краткое описание. При нажатии на карточку проекта откроется подробная информация о выбранном проекте.

3.3.4 Страница тест-кейсов

Страница тест-кейсов будет отображаться со следующими элементами:

1. Список тест-кейсов с информацией:
   - Идентификатор теста;
   - Название теста;
   - Тип теста;
   - Статус выполнения;
   - Приоритет;
   - Дата последнего запуска;
2. Фильтры по:
   - Типу теста;
   - Статусу;
   - Приоритету;
3. Возможность создания, редактирования и удаления тест-кейсов.

3.3.5 Страница отчетов

Страница отчетов будет содержать подробную информацию о результатах тестирования:

1. Общая статистика:
    - Количество выполненных тестов;
    - Процент успешных тестов;
    - Среднее время выполнения;
    - Тренды выполнения;
2. Детальные отчеты по:
    - Проектам;
    - Типам тестов;
    - Периодам времени;
3. Возможность экспорта отчетов в различные форматы (PDF, CSV, JSON).

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

4.1 Платформа и технологии

Система должна быть реализована как веб-приложение, доступное через современные браузеры (Chrome, Firefox, Safari, Edge) последних версий. Серверная часть должна быть развернута на операционных системах, совместимых с Linux.

4.2 Язык программирования и фреймворк

Серверная часть системы должна быть разработана с использованием:

- Python 3.10 или выше
- Fast API для реализации REST API
- SQLAlchemy для работы с базой данных Frontend часть должна использовать:
- React.js для построения пользовательского интерфейса
- TypeScript для обеспечения типизации
- Material-UI или Tailwind CSS для компонентов интерфейса

4.3 Интеграция со спецификациями API

Система должна поддерживать работу со следующими форматами спецификаций:

- OpenAPI/Swagger версий 2.0 и 3.0
- JSON Schema для валидации данных
- Поддержка импорта спецификаций в форматах YAML и JSON
- Автоматическое обновление при изменении спецификаций

4.4 Архитектура системы

Система должна быть построена на основе:

- Микросервисной архитектуры
- REST API для взаимодействия между компонентами
- JWT для аутентификации и авторизации
- PostgreSQL для хранения данных
- Redis для кэширования и очередей задач

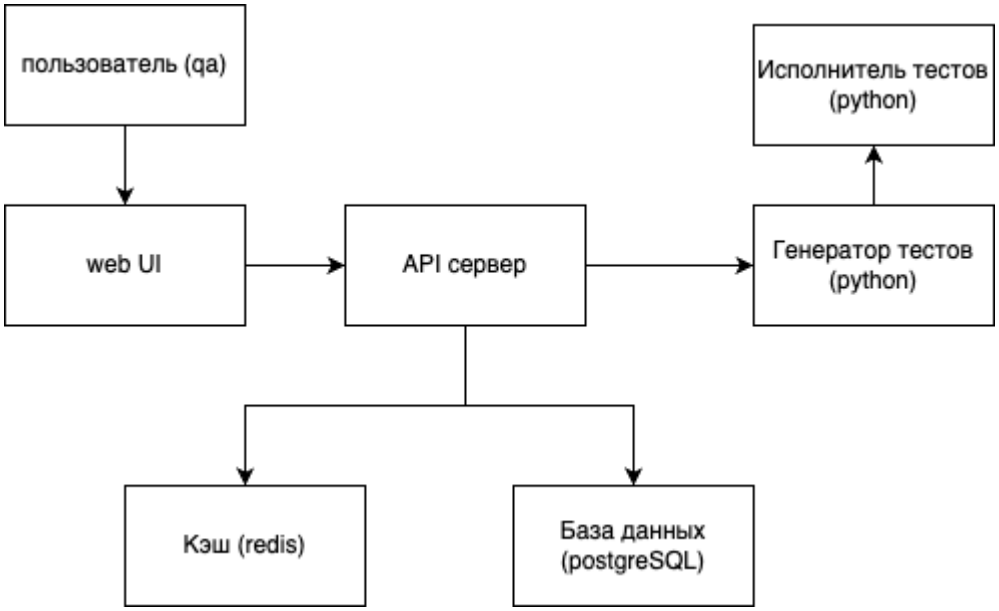
4.5 Требования к производительности

Система должна обеспечивать следующие показатели производительности:

- Время отклика API не более 500 мс для 95% запросов
- Поддержка одновременной работы не менее 100 пользователей
- Возможность параллельного выполнения до 50 тестовых сценариев
- Время генерации тестовых сценариев не более 30 секунд на один API-интерфейс

АРХИТЕКТУРНАЯ ДИАГРАММА

Архитектурная диаграмма представляет основные компоненты системы и их взаимодействие:



ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ

В этом разделе описаны ожидаемые результаты разработки системы управления тестированием с автоматической генерацией тестовых сценариев для API.

Требования к конечному продукту:

Полностью функциональная веб-система для управления тестированием API со следующими возможностями:

- Автоматическая генерация тестовых сценариев на основе спецификаций OpenAPI/Swagger
- Выполнение сгенерированных тестов с формированием подробных отчетов
- Удобный пользовательский интерфейс для управления проектами и тестами
- Интеграция с различными форматами API-спецификаций

Ключевые показатели успешности проекта:

- Система способна автоматически генерировать не менее 90% необходимых тестовых сценариев на основе спецификации API
- Время генерации тестовых сценариев не превышает 30 секунд на один конечную точку API
- Возможность параллельного выполнения до 50 тестовых сценариев
- Поддержка одновременной работы не менее 100 пользователей
- Время отклика системы не более 500 мс для 95% запросов

Готовность к внедрению:

- Система прошла полное функциональное тестирование
- Документация по установке и использованию системы подготовлена
- Разработаны инструкции для конечных пользователей
- Система готова к развертыванию в промышленной среде
- Реализованы все требования к безопасности и производительности

ТЕСТИРОВАНИЕ

7.1 План тестирования

1. Тестирование генерации тест-кейсов:
    - Корректность парсинга OpenAPI/Swagger спецификаций
    - Правильность генерации тестовых данных для разных типов параметров
    - Полнота охвата всех endpoint'ов API
    - Генерация позитивных и негативных тестовых сценариев
2. Функциональное тестирование веб-интерфейса:
    - Работа с проектами (создание, редактирование, удаление)
    - Управление тест-кейсами (просмотр, запуск, анализ результатов)
    - Корректность отображения отчетов и статистики
    - Работа системы авторизации и прав доступа
3. Интеграционное тестирование:
    - Взаимодействие между компонентами системы
    - Корректность сохранения и получения данных из БД
    - Работа с внешними API
    - Производительность системы кэширования

7.2 Критерии приемки

1. Функциональные критерии:
    - Успешная генерация тест-кейсов для всех поддерживаемых форматов спецификаций
    - Корректное выполнение всех типов тестов
    - Точность валидации ответов API
    - Полнота формируемых отчетов
2. Технические критерии:
    - Время отклика веб-интерфейса не более 500 мс
    - Успешное выполнение параллельных тестовых запусков
    - Корректная работа в разных браузерах
    - Отсутствие критических ошибок в логах
3. Пользовательские критерии:
    - Интуитивность навигации по системе

- Удобство работы с тест-кейсами
- Понятность представления результатов
- Эффективность работы с большими наборами тестов

7.3 Нагрузочное тестирование

1. Сценарии нагрузочного тестирования:
   - Одновременная генерация тест-кейсов для нескольких API
   - Параллельное выполнение множества тестовых сценариев
   - Работа множества пользователей с системой
   - Обработка больших объемов тестовых данных
2. Метрики производительности:
   - Время отклика системы при различных уровнях нагрузки
   - Потребление ресурсов (CPU, память, диск)
   - Пропускная способность при параллельном выполнении тестов
   - Стабильность работы компонентов системы
3. Критерии успешности:
   - Поддержка одновременной работы 100+ пользователей
   - Выполнение до 50 параллельных тестовых сценариев
   - Время генерации тест-кейсов не более 30 секунд на endpoint
   - Сохранение производительности при длительной работе

ЗАКЛЮЧЕНИЕ

Разработанная система управления тестированием с автоматической генерацией тест-кейсов для API представляет собой комплексное решение, отвечающее современным требованиям к автоматизации процессов тестирования программного обеспечения. В ходе работы были успешно реализованы все поставленные цели по созданию эффективного инструмента для команд разработки и тестирования.

Система значительно оптимизирует процесс тестирования API благодаря автоматической генерации тест-кейсов на основе спецификаций, что существенно сокращает временные затраты и минимизирует возможность человеческих ошибок. Реализованные функции управления тестовыми наборами, автоматического обновления тестов при изменении спецификаций и интеграции с CI/CD системами обеспечивают комплексный подход к контролю качества API.

Особую ценность представляет возможность автоматического отслеживания изменений в спецификациях API и своевременного обновления тестовых сценариев, что позволяет поддерживать актуальность тестового покрытия без дополнительных усилий со стороны команды. Внедрение данной системы позволит организациям существенно повысить эффективность процессов тестирования, улучшить качество разрабатываемых API и ускорить процесс выпуска программного обеспечения.