МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ API И UI

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Фруминой Софии Марковны

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Зав.кафедрой,
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

ВВЕДЕНИЕ

В современной индустрии разработки программного обеспечения автоматизация тестирования играет ключевую роль в обеспечении качества продукта. Эта работа посвящена разработке системы автоматизированного тестирования для API сервисов CleanUI и RandomUser, а также для пользовательского интерфейса сервиса DemoQA.

Цель работы заключается в создании комплексной системы тестирования, способной обеспечивать высокий уровень надежности и стабильности веб-приложений путем автоматизации процесса проверки их функциональности и интерфейса.

Для достижения поставленной цели были определены следующие **задачи**:

1. Изучить и проанализировать документацию API сервисов CleanUI и RandomUser.
2. Разработать тестовые сценарии для положительных и отрицательных случаев использования API.
3. Реализовать автоматизированные тесты с использованием Java, Maven, JUnit, Rest Assured и Selenide.
4. Протестировать пользовательский интерфейс сервиса DemoQA, включая валидацию форм и проверку результатов отправки данных.
5. Интегрировать разработанные тесты в систему непрерывной интеграции для регулярного мониторинга качества веб-приложений.

Данная работа представляет собой вклад в область автоматизации тестирования, направленный на повышение эффективности и надежности веб-приложений, что является **актуальной** задачей в современном мире информационных технологий.

Структура и объём работы.

Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 2 основные главы, заключение, список использованных источников из 20 наименований и 2 приложения. Работа изложена на 40 страницах, содержит 11 рисунков.

Первая глава имеет название «Анализ предметной области. Автоматизированное тестирование» и содержит основную теоретическую информацию о технологиях и подходах, которые использовались при разработке системы автоматизированного тестирования API и UI.

Вторая глава имеет название «Реализация проекта по автоматизированному тестированию», данная глава содержит подробное описание процесса выполнения работы.

Выпускная квалификационная работа заканчивается заключением, списком использованных источников, а также приложениями с кодом А-Б.

1 Основное содержание работы

1.1 Анализ предметной области. Автоматизированное тестирование

Автоматизированное тестирование API

Автоматизированное тестирование ПО — это процесс тестирования программного обеспечения, при котором основные функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, производятся автоматически с помощью инструментов для автоматизированного тестирования.

API (Application Programming Interface) — это механизм, позволяющий различным программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. Это позволяет системам и приложениям обмениваться данными и функциями, обеспечивая интеграцию и совместимость между ними.

Автоматизированное тестирование UI Тестирование пользовательского интерфейса (UI) является важной частью процесса разработки программного обеспечения, направленной на обеспечение качества и надежности графического интерфейса пользователя (GUI). Оно охватывает различные аспекты взаимодействия пользователя с приложением, включая функциональность элементов интерфейса, визуальное восприятие и удобство использования.

1.2 Постановка задачи автоматизированного тестирования API и UI

Используемым стеком технологий для данного проекта являлись:

— Java 17 является одним из последних релизов языка программирования Java, который предоставляет множество новых функций и улучшений по сравнению с предыдущими версиями. Это включает в себя улучшения производительности, новые возможности языка, такие как switch выражения, текстовые блоки, паттерн матчинг для instanceof и другие. Java 17 также фокусируется на безопасности и стабильности, делая его отличным выбором для современных приложений.

— Maven — это инструмент для автоматизации сборки проектов на основе описания их структуры в файле pom.xml. Он позволяет управлять зависимостями проекта, автоматически загружая необходимые библиотеки из центральных репозиториев. Maven также поддерживает плагины, которые могут быть использованы для выполнения различных задач, таких как

компиляция кода, запуск тестов, создание отчетов и многое другое.

— JUnit 5 — это последняя версия популярной библиотеки для модульного тестирования в Java. Она предлагает ряд улучшений по сравнению с JUnit 4, включая упрощенный API, поддержку параметризованных тестов, возможность использования лямбда-выражений и улучшенную интеграцию с другими инструментами тестирования. JUnit 5 делится на три основных модуля: junit-platform (для запуска тестов), junit-jupiter (для написания тестов) и junit-vintage (для обеспечения совместимости с JUnit 4).
— IntelliJ IDEA — это среда разработки (IDE) для Java и других языков программирования. Она предлагает широкий спектр функций, таких как автодополнение кода, рефакторинг, интеграция с системами контроля версий, поддержка тестирования и многие другие инструменты, которые облегчают процесс разработки. IntelliJ IDEA доступна в двух вариантах: Community Edition (бесплатный) и Ultimate Edition (платный с дополнительными возможностями).
— Rest Assured — это библиотека для тестирования RESTful веб-сервисов в Java. Она предоставляет удобный DSL (Domain Specific Language) для отправки HTTP-запросов и проверки ответов, что делает процесс тестирования API более простым и понятным. Rest Assured поддерживает различные типы запросов, аутентификацию, работу с JSON и XML, а также другие возможности, необходимые для полноценного тестирования веб-API.
— Selenide — это библиотека для автоматического тестирования веб-интерфейсов. Она построена поверх Selenium WebDriver и предлагает более простой и выразительный API для работы с браузерами. Selenide позволяет легко находить элементы на странице, взаимодействовать с ними, выполнять действия, такие как клики или ввод текста, и проверять состояние интерфейса. Библиотека также поддерживает асинхронное ожидание, что уменьшает количество проблем с таймаутами во время тестирования.
— AssertJ — это библиотека для написания утверждений в тестах на Java. Она предлагает богатый набор методов для проверки состояния объектов, коллекций, строк и других данных. AssertJ делает тесты более читаемыми и позволяет писать более точные и информативные утверждения, чем стандартные средства JUnit.

— GitHub — это сервис для хостинга IT-проектов и совместной работы над кодом. Он использует систему контроля версий Git, которая позволяет разработчикам эффективно работать над проектами вместе, отслеживать изменения, создавать ветки для новой функциональности и исправлений ошибок. GitHub также предлагает инструменты для код-ревью, управления задачами, документации и другие возможности, которые помогают организовать процесс разработки.

Для реализации проекта были конкретизированы задачи:

API тестирование

CleanURI

— Написать автотесты для тестирования API сервиса cleanui.com.
— Составить тестовый набор, включая положительные и отрицательные сценарии.
— Тестовые данные должны загружаться из внешнего файла.

RandomUser

— Написать автотесты для тестирования API сервиса randomuser.me.
— Использовать query-параметры при выполнении запросов.
— Распарсить ответы сервиса в Java-объекты и выполнить проверки с их помощью.

UI тестирование

DemoQA

— Заполнить форму валидными данными и нажать на кнопку Submit.
— Проверить корректность заполнения формы.
— Реализовать негативные проверки формы.

Sovcombank

— Перейти на страницу вакансий и выбрать фильтры.
— Собрать результаты работы фильтра в коллекцию.
— Проверить соответствие найденных результатов критериям фильтра.
— Применить паттерн PageObject и использовать XPath.

1.3 Сценарии тестирования

Postman Postman представляет собой инструмент для тестирования API, обеспечивающий комплексные возможности для создания, выполнения и анализа тестовых сценариев. Он позволяет автоматизировать тестирование, интегрировать тесты в процессы CI/CD, а также использоваться для мониторинга и

отладки API.

Основные преимущества использования Postman включают:

— Автоматизация тестирования: Postman позволяет создавать и запускать тесты с использованием предварительно настроенных фрагментов кода, что упрощает процесс тестирования различных архитектур API, включая REST, GraphQL, SOAP и gRPC.
— Интеграция с CI/CD: Инструменты Newman и Postman CLI позволяют интегрировать тесты в пайплайны непрерывной интеграции и доставки, обеспечивая автоматический запуск тестов при каждом изменении кода.
— Мониторинг и отладка: Postman Console предоставляет детальную информацию о сетевых вызовах, включая заголовки, сертификаты, запросы и ответы, что облегчает отладку и анализ результатов тестирования.
— Создание специализированных тестовых сред: Postman позволяет хранить значения переменных на уровне окружения, что позволяет проводить автоматизированное тестирование API в специализированной тестовой среде перед развертыванием кода в продакшн.

Сценарии тестирования для API

CleanURI API используется для сокращения длинных URL-адресов до коротких, удобных для использования ссылок.

Основная функциональность CleanURI API заключается в том, что разработчики могут отправлять запросы на сокращение URL-адресов через HTTP POST запросы к указанному конечной точке API.

Позитивные сценарии. Создание короткой ссылки:

— Отправить POST-запрос на /shorten с валидным URL.
— Проверить, что статус ответа 200 OK.
— Проверить, что в ответе присутствует поле shortUrl с новой короткой ссылкой.

Негативные сценарии. Создание короткой ссылки с невалидным URL:

— Отправить POST-запрос на /shorten с невалидным URL.
— Проверить, что статус ответа 400 Bad Request.

Сценарии для RandomUser

Random User API — это бесплатный, открытый API, предназначенный для генерации случайных данных пользователей, таких как имя, электронная почта, адрес и имя пользователя. Этот API идеально подходит для использования

в качестве заполнителей или для тестирования приложений. Он предоставляет возможность генерации до 5000 пользователей за один запрос, используя параметр «results». Кроме того, API поддерживает различные параметры запроса, такие как gender для фильтрации пользователей по полу, «nat» для выбора национальности и «seed» для генерации одних и тех же пользователей каждый раз при использовании этого ключа.

Позитивные сценарии. Получение пользователя и фильтрация:

— Получение информации о пользователе по гендеру: Запросить данные пользователя с указанным гендером (например, gender=female) и проверить, что полученные данные соответствуют заданному критерию.
— Получение определенного количества пользователей: Использовать параметр results для получения конкретного числа пользователей (например, results=5) и убедиться, что количество возвращенных записей соответствует запросу.
— Фильтрация по национальности: Использовать параметр nat для фильтрации пользователей по стране происхождения (например, nat=us,gb) и проверить, что все возвращенные пользователи соответствуют указанной национальности.

Негативные сценарии. Отсутствие параметров или их некорректность:

— Неверный параметр запроса: Попытаться выполнить запрос с параметром, которого не существует (например, unknownParam=value), и проверить, что сервис корректно обрабатывает такой запрос, возможно, игнорируя неизвестный параметр или возвращая ошибку.
— Запрос с недопустимым значением параметра: Использовать допустимый параметр запроса, но с недопустимым значением (например, gender=alien). Сервис должен либо вернуть ошибку, либо обработать запрос, игнорируя недопустимое значение.
— Запрос без обязательных параметров: попытаться выполнить запрос без обязательных параметров и убедиться, что сервис возвращает соответствующее сообщение об ошибке.

Сценарии тестирования для UI

DemoQA —— это платформа, предназначенная для обучения и практики в области автоматизации тестирования. Она предоставляет различные интерактивные элементы и страницы, которые могут быть использованы для демон-

страции и отладки тестовых сценариев. В данной работе будет рассмотренно тестирование форм.

Позитивные сценарии:

— Заполнение всех полей формы валидными данными.
— Нажатие на кнопку Submit.
— Проверка корректности формы во всплывающем окне.

Негативные сценарии:

— Заполнение всех полей формы невалидными данными.
— Отсутствие заполнения полей формы.

Sovkombank — сайт известного банка, в котором будет покрыт тестами раздел «Vacancies», где содержиться актуальная информация о доступных позициях и требованиях к кандидатам, навигация по сайту удобна за счет фильтрации — корректность работы, которой будет проверяться. Позитивные сценарии:

— Переход на страницу сайта.
— Нажать на кнопку «Вакансии».
— Выбрать в фильтре вакансий Город = «Москва», Компания = «Совкомбанк Технологии».
— Собрать в коллекцию результаты работы фильтра (все найденные после фильтрации вакансии).
— Проверить, что у всех найденных результатов указан город «Москва» и компания «Совкомбанк Технологии».

1.4 Анализ и оценка качества тестирования

Оценка качества тестирования включает в себя анализ эффективности и надежности проведенного тестирования. Ключевые аспекты, которые следует учитывать при оценке качества тестирования, включают:

— Контроль покрытия кода тестами: Оценка того, какой процент кода был протестирован. Используются различные метрики, например, покрытие кода тестами с помощью инструментов, таких как JaCoCo для Java.
— Степень детализации тестов: Оценка того, насколько тщательно были протестированы различные аспекты системы. Например, тесты на краевые условия, тесты на отказоустойчивость.
— Время и стоимость тестирования: Сравнение затрат времени и ресурсов, необходимых для проведения тестирования, с ожидаемыми результатами.

— Чувствительность тестов: Способность тестов выявлять реальные проблемы в системе, а не ложные срабатывания.

1.5 Структура проекта

В данной практической части будет рассмотрен процесс создания и реализации фреймворка для автоматизированного тестирования API и UI. Мы обсудим структуру проекта, основные компоненты и классы, используемые для тестирования, а также подходы и инструменты, применяемые для обеспечения качественного и надежного тестирования.

Проект включает тесты для API и UI, реализованные с использованием подхода Page Object Model (POM) для UI тестов и структурированного подхода для API тестов.

Проект организован следующим образом:

— src/main/java - исходные файлы приложения.
— src/test/java - файлы с тестами.
— src/test/resources - ресурсы для тестов (JSON файлы с тестовыми данными).

1.6 Тестирование API

В рамках практической части работы было проведено тестирование API сервиса для сокращения URL. Это важно для обеспечения надежности и безопасности сервиса, а также для проверки корректности его функционирования. Для автоматизации процесса тестирования использовался RestAssured — инструмент, который позволяет удобно работать с REST API на языке Java.

В рамках практической части работы было проведено тестирование API сервиса randomuser, предназначенного для генерации случайных профилей пользователей. Это позволяет исследовать различные аспекты работы с данными пользователя и проверять корректность фильтрации по различным параметрам. Тестирование осуществлялось с использованием библиотеки RestAssured для Java, которая упрощает написание тестов для RESTful веб-сервисов. Были разработаны несколько тестовых сценариев для проверки основных функциональных возможностей API.

1.7 Тестирование UI

В рамках практической части работы была разработана и протестирована форма регистрации на примере веб-сайта сервиса demoqa. Для автоматизации

тестирования использовались инструменты Selenium и Selenide, что позволяет упростить процесс написания тестов и увеличить их читаемость. Вручную была проведена предпроверка тестируемого материала. Сайт представляет собой заполнение формы регистрации для студентов.

В рамках практической части работы был разработан и протестирован сценарий поиска вакансий на сайте Совкомбанка.Для автоматизации тестирования использовались инструменты: Selenium и Selenide.

1.8 Интеграция проекта в GitHub

Интеграция проекта в GitHub критическо важна для управления версиями и удобства при дальнейшем расширении проекта. Для интеграции данного проекта понадобилось сделать следующие шаги:

- Шаг 1: Создание нового репозитория на GitHub.
- Шаг 2: Инициализация Git в проекте.
- Шаг 3: Добавление файлов в репозиторий.
- Шаг 4: Коммит изменений.
- Шаг 5: Связывание локального репозитория с удаленным на GitHub.
- Шаг 6: Отправка изменений на GitHub.

ЗАКЛЮЧЕНИЕ

Работа над проектом позволила глубже понять принципы работы API сервисов и разработать тестовые сценарии для различных случаев использования, включая как положительные, так и отрицательные. Была продемонстрирована возможность применения современных технологий и инструментов, таких как Java, Maven, JUnit, Rest Assured и Selenide, для создания эффективных автоматизированных тестов.

В результате выполнения работы была достигнута поставленная цель — создание системы автоматизированного тестирования, способной обеспечивать высокий уровень надежности и стабильности веб-приложений.

Подводя итоги, можно сказать, что при выполнении данной работы были достаточно подробно изложены и реализованы такие задачи как:

- задача изучения основных принципов и инстурментов разработки автоматизированных тестов;
- Создание авто-тестов для API и UI;
- Интеграция проекта в среду GitHub.