МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра математической теории упругости и биомеханики

Разработка функциональных автотестов для веб-приложений

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 442 группы

направление 09.03.03 — Прикладная информатика

механико-математического факультета

Попова Даниила Сергеевича

Научный руководитель
к.ю.н., доцент _____ Р.В. Амелин

Зав. кафедрой
д.ф.-м.н., профессор _____ Л.Ю. Коссович

Саратов 2024

Введение. Автоматизированное тестирование — это важный и эффективный инструмент для проверки качества программного продукта[1]. В данный момент автоматизированное тестирование является непосредственно частью разработки продукта перед выходом в продакшн. Включая в систему автотесты разработчики могут быстро находить ошибки при внесении каких-либо изменений в тестовой среде.

Автоматизированное тестирование играет ключевую роль в обеспечении качества программного обеспечения. Оно позволяет разработчикам быстро и эффективно находить ошибки в коде, что способствует повышению надежности и безопасности продукта.

В современном мире, где технологии развиваются с невероятной скоростью, автоматизация тестирования становится неотъемлемой частью процесса разработки. Она помогает сократить время на тестирование, снизить риски выпуска некачественного продукта и обеспечить его соответствие требованиям пользователей.

Автоматизированное функциональное тестирование включает в себя создание и выполнение автоматических тестов, которые имитируют действия пользователя и проверяют работоспособность различных функций приложения.

Применение автоматизированного тестирования особенно актуально для крупных проектов с большим количеством кода и функций. Оно позволяет обеспечить высокое качество продукта и его соответствие ожиданиям пользователей.

Объектом исследования в данной работе будет выступать Фонд президентских грантов.

Предметом исследования работы является проект автоматизированного тестирования веб-приложения.

Целью работы является разработка функциональных автоматизированных тестов.

Для достижения цели следует выполнить следующие задачи:

- выполнить анализ предметной области;
- спроектировать систему тестирования;
- создать потоковую диаграмму работы автотеста;
- создать таблицу покрытия тестирования;
- создать тест-кейсы;
- разработать автотесты;
- провести испытания написанных автотестов.

В первом разделе рассматривается и анализируется предметная область.

Функциональные тесты пишутся для организации «Фонд президентских грантов» .

Один из последних проектов фонда - платформа НКО, которая является цифровым агрегатором открытых данных о деятельности некоммерческих организаций. Платформа предназначена для упрощения процесса взаимодействия между НКО, государством и обществом, а также для повышения прозрачности и открытости работы НКО, содействия развитию гражданского общества.

Платформа НКО предоставляет пользователям доступ к обширной базе данных о деятельности НКО, включая информацию об их проектах, программах, мероприятиях и финансовых отчетах. Это позволяет гражданам лучше понимать, чем занимаются НКО, и поддерживать те из них, которые наиболее близки их интересам и ценностям.

Структура платформы НКО включает в себя главную страницу с общей информацией, страницу "О платформе" с информацией о источниках данных, страницу "Организации" с карточками организаций и расширенным поиском, страницу "Новости" с новостями о деятельности Фонда и анонсами предстоящих мероприятий, а также карточки НКО с подробной информацией о компании.

Платформа НКО в настоящее время тестируется разработчиком и коллегами вручную, что занимает много времени и не покрывает весь функционал. Поэтому необходимо автоматизировать процесс тестирования для

более эффективной и полной проверки платформы.

Во втором разделе описывается проектирование системы тестирования, которая состоит из

- архитектуры;
- потоковой диаграммы;
- таблицы покрытия тестами;
- тест-кейсы.

Архитектура системы тестирования должна учитывать паттерны программирования.

Архитектура системы тестирования должна быть разделена на три основных слоя: слой основных компонентов, слой тестовых компонентов и слой компонентов запуска.

В результате, все слои взаимодействуют в этой архитектуре, обеспечивая модульность, читаемость и легкость поддержки. Данная структура фокусируется на подготовке и запуске самих тестов и обеспечивает эффективное и правильное их выполнение. Результат готовой архитектуры показан на рисунке 1.

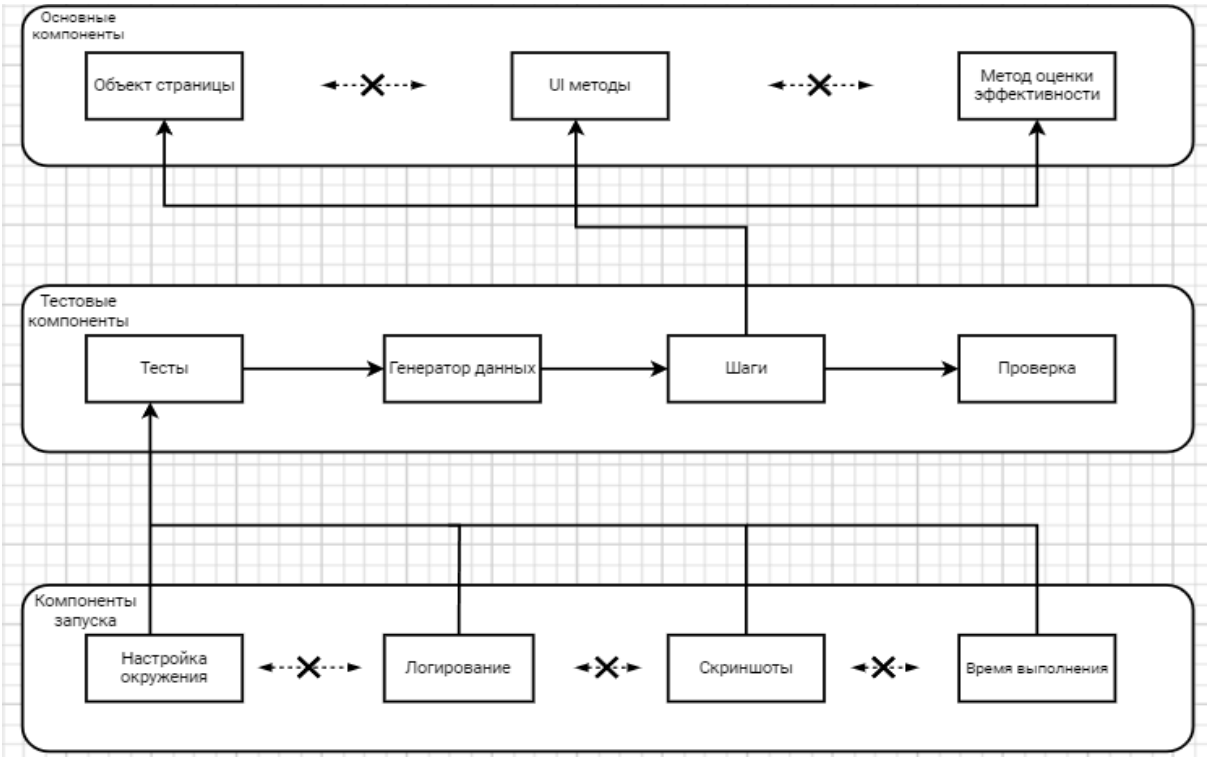

Рисунок 1 - Архитектура системы тестирования

Проектирование потоковых диаграмм в контексте автоматизированного тестирования играет ключевую роль в обеспечении эффективного и надежного процесса тестирования программного обеспечения. Потоковые диаграммы графически отображают последовательность выполнения тестовых сценариев и взаимодействие между различными компонентами системы, что упрощает понимание и анализ процесса тестирования.

Потоковая диаграмма будет состоять из основных блоков страниц, модулей конкретной страницы и самих автотестов. Последовательность действий будет следующей: генерация данных, шаг, проверка, запись результата.

Результат шага необходимо записать и внести в итоговой отчет о работе автотеста. В зависимости от результата шага, необходимо перейти к следующему шагу или прервать автотест и перейти к следующему, если есть автотесты в очереди.

В результате получится такая последовательность выполнения автотеста в зависимости от результата (рисунок 2)

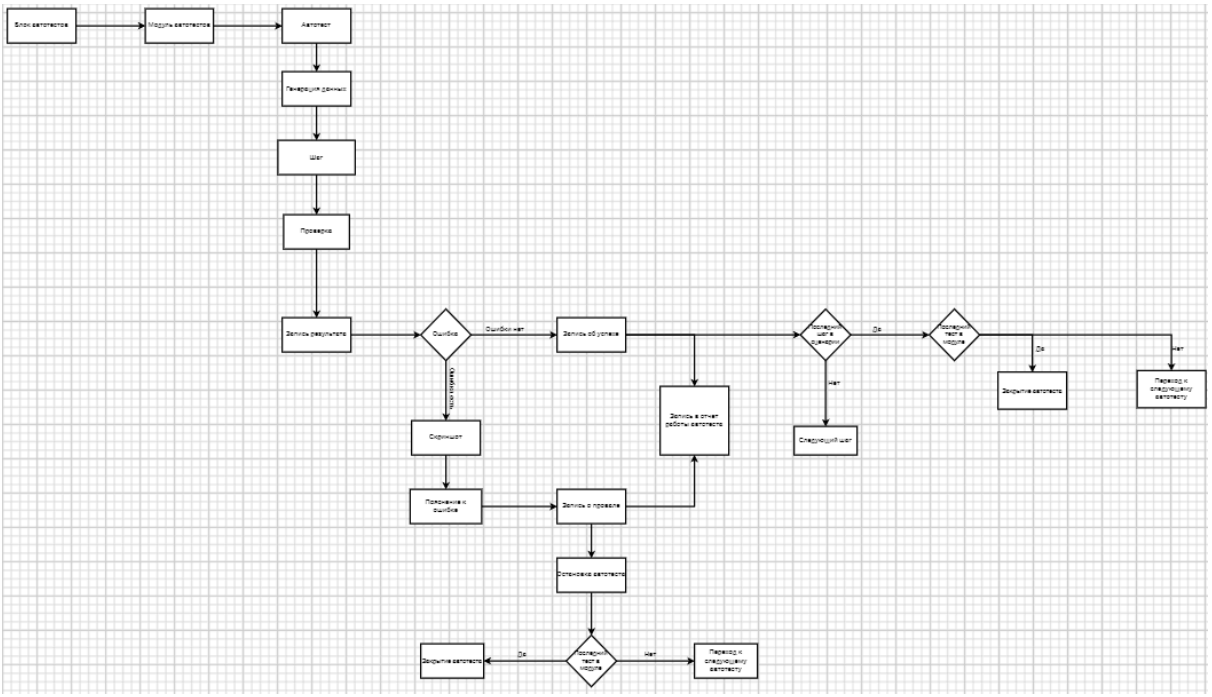

Рисунок 2 - Потоковая диаграмма автотеста

Для эффективного автоматизированного тестирования веб-приложения, необходимо разделить его на блоки и модули. Блоками будут выступать

веб-страницы, а модули - функциональные элементы на этих страницах.

Каждый модуль или функционал имеет свой приоритет, который отражает важность тестирования.

Высокий приоритет присваивается критически важным функциям, средний - важным, но не критическим, а низкий - необязательным или имеющим ограниченное влияние на работу системы.

Структура тестирования будет отображена в таблице, чтобы удобно описывать покрытие тестирования. Результатом будет полученная таблица (рисунок 3).

Блоки	Модули	Функционал	Приоритет
Шапка страниц	Ссылки на другие проекты Фонда		Низкий
	Навигационные ссылки		Высокий
Подвал страниц	Ссылка на соц-сети		Низкий
	Ссылка на другие проекты фонда		Низкий
	Контакты	Ссылка на страницу с контактной информацией	Высокий
		Ссылка на почту	Низкий
Главная страница	Тело страницы	Поиск	Высокий
		Карточки компаний	Высокий
		Новости	Средний
		Организации с проектами топ-100	Средний

Рисунок 3 - Таблица покрытия тестирования

Тест-кейсы важны для проектирования автотестов, так как без них создание автотестов невозможно. Автоматизированные тесты, основанные на тест-кейсах, ускоряют процесс тестирования, позволяя проводить его регулярно и быстро выявлять дефекты.

Тест-кейс - это документ, описывающий шаги для проверки функции или

ее части. Написание тест-кейсов позволяет структурировать подход к тестированию, вычислять метрики тестового покрытия, отслеживать соответствие плану и уточнять взаимопонимание между заказчиком, разработчиками и тестировщиками.

Пример готового тест-кейса изображен на рисунке 4.

Шаги

Шаг	**Ожидаемый результат**
Кликнуть на ссылку в шапке страницы “О платформе”	Переход на страницу “О платформе”
Кликнуть на ссылку в шапке страницы “Организации”	Переход на страницу “Организации”
Кликнуть на ссылку в шапке страницы “Новости”	Переход на страницу “Новости”

Рисунок 4 - Пример тест-кейса

В третьем разделе рассматривается реализация и сама разработка автотестов.

Автотесты написаны на языке JavaScript с использованием библиотеки Taiko и фреймворка Gauge для тестовых сценариев.

В соответствии с спроектированной системой получилась система функциональных автотестов.

Пример кода автотестов представлен на рисунках 5-6.

```
Run Scenario | Debug Scenario
## Search page org INN
* Activation search
* Search INN

Run Scenario | Debug Scenario
## Seach page org  OGRN
* Activation search
* Search OGRN

Run Scenario | Debug Scenario
## Goto url org
* Goto organization
```

Рисунок 5 - Код Gauge для страницы «Организации»

```
step("Activation search", async () => {
  await click($(`//input[@name='search']`));
});

1 reference(s)
step("Search INN", async () => {
  await write("3528098649");
});

1 reference(s)
step("Search OGRN", async () => {
  await write("1053500221092");
});

1 reference(s)
step("Goto organization", async () => {
  await click($(`//div[@class='col-12 col-lg-8 position-static']`));
});
```

Рисунок 6 - JS код для страницы «Организации»

Отчеты о результате выполненных автотестов находится в специальном html-файле (рисунок 7).

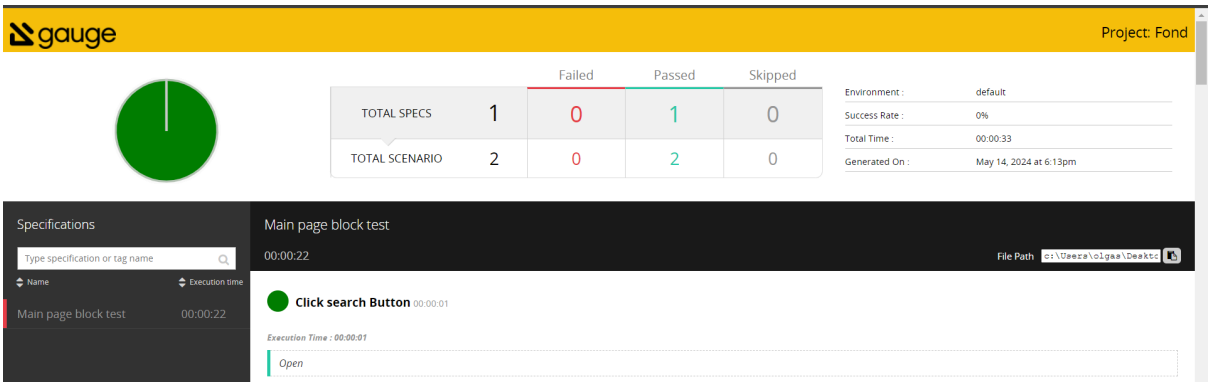

Рисунок 7 - Отчеты о выполненных автотестах

Тут можно полностью посмотреть все выполненные сценарии и шаги, а также время потраченное на это.

В случае возникновения каких-либо ошибок делается скриншот (рисунок 8).

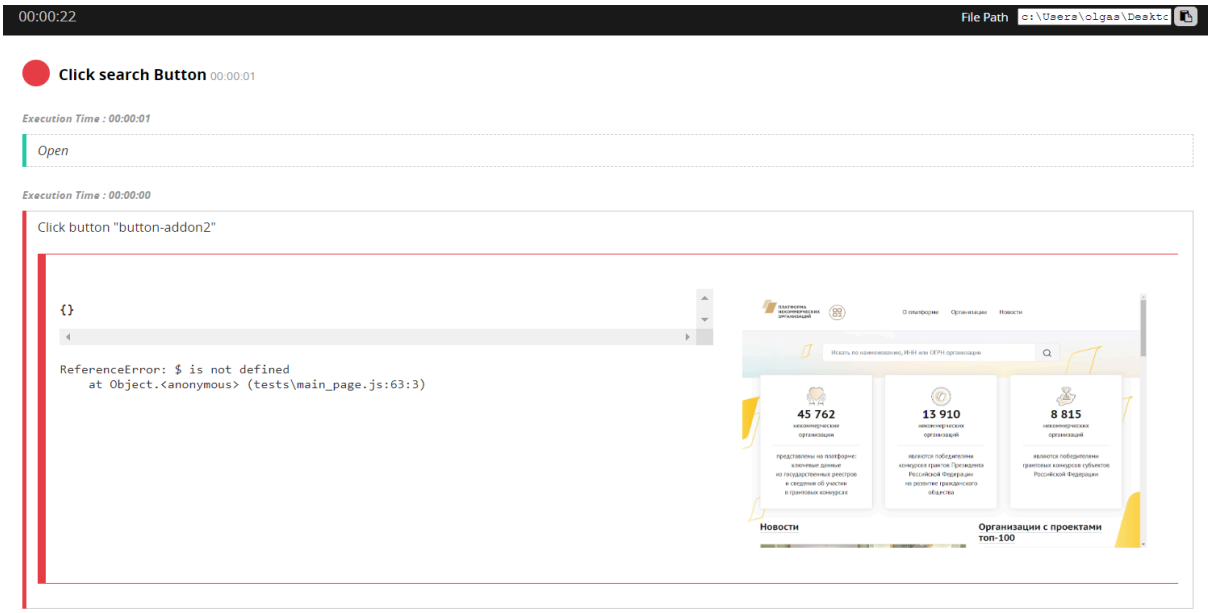

Рисунок 8 - Ошибка при выполнении автотеста

Заключение. Автоматизированное тестирование позволяет быстро находить ошибки, повышает надежность и безопасность продукта, а также сокращает время на тестирование и снижает риски выпуска некачественного продукта.

В рамках данной работы были выполнены анализ предметной области, спроектирована система тестирования, создана потоковая диаграмма работы

автотеста, сделана таблица покрытия тестирования, созданы тест-кейсы, разработаны и проверены автотесты.