

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра информатики и
программирования

**РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ АВТОМАТИЗАЦИИ
ТЕСТИРОВАНИЯ WEB-ПРИЛОЖЕНИЙ**
АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 441 группы
направления 02.03.03 — Математическое обеспечение и администрирование
информационных систем
факультета КНиИТ
Мирзоевой Екатерины Алексеевны

Научный руководитель
зав. кафедрой, к. ф.-м. н., доцент _____

М. В. Огнева

Консультант
ведущий разработчик,
директор ООО «Перспективные
технологии», к. с. н. _____

К. Е. Медведев

Заведующий кафедрой
к. ф.-м. н., доцент _____

М. В. Огнева

Саратов 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Краткое содержание работы	4
1.1 Автоматизированное тестирование	4
1.2 Практическая часть. Реализация инструмента для автоматизации тестирования web-приложений	5
1.2.1 Серверная часть приложения	6
1.2.2 Клиентская часть приложения	7
1.3 Пользовательский интерфейс	7
ЗАКЛЮЧЕНИЕ	12
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	13

ВВЕДЕНИЕ

Автоматизированное тестирование набирает особую популярность, так как объем данных и масштабы проектов набирают обороты и ручное тестирование часто не справляется с поставленными задачами в короткие сроки.

Целью бакалаврской работы является разработка инструмента для автоматизации тестирования web-приложений.

Задачи бакалаврской работы:

- рассмотреть основные понятия теории тестирования программного обеспечения;
- ознакомиться с инструментами для тестирования, такими как Selenium, Screenster, PhantomJS, и выяснить их плюсы и минусы;
- разработать программу для автоматизации тестирования программного обеспечения, включающую в себя API-методы работы с тестовыми сценариями, набор доступных операций и интерфейс пользователя;
- реализовать возможность просмотра отчета тестирования и истории выполнения тестов в интерфейсе приложения;
- реализовать разграничение доступа пользователя по проектам, для этого создать API-методы для создания пользователей и проектов;
- применить на практике разработанный инструмент — создать набор тестовых случаев.

В теоретической части работы описаны используемые технологии, термины и приведены все необходимые предварительные сведения для практической реализации. В практической части работы описан разработанный программный продукт, продемонстрированы примеры его тестовых запусков и сделаны соответствующие выводы.

Структура и объем работы. Бакалаврская работа состоит из введения, 6 разделов, заключения, списка использованных источников. Общий объем работы – 75 страниц, из них 69 страниц – основное содержание, включая 21 рисунок, цифровой носитель в качестве приложения, список использованных источников информации – 20 наименований.

1 Краткое содержание работы

1.1 Автоматизированное тестирование

Автоматизированное тестирование — отдельный вид тестирования, требующий от тестировщика особых навыков: владения каким-либо языком программирования, знания технологий [1].

При планировании тестирования и создания тест-плана на проекте необходимо обдумать, каким образом будет проводиться тестирование: нужно понять, какие плюсы и минусы у того или иного подхода.

Автоматизированное тестирование имеет ряд недостатков и преимуществ.

Преимущества автоматизированного тестирования:

- Однообразность выполнения. Получается, что исключается человеческий фактор. То есть тесты выполняются от версии к версии, а значит человек по невнимательности не сможет пропустить какой-либо дефект;
- Быстрое регрессионное тестирование. Вместо того, чтобы тестировать приложение заново руками, стоит просто запустить набор тестов;
- Отчетность. Отчеты о результате тестирования составляются автоматически;
- Выполнение без вмешательства. Во время выполнения тестов тестировщик может выполнять другие задачи.

Недостатки автоматизированного тестирования:

- Однообразность. Написанные тесты будут делать все время одно и то же. Это одновременно является и недостатком, так как тестировщик, выполняя тест вручную, может обратить внимание на некоторые детали и, проведя несколько дополнительных операций, найти дефект. Скрипт этого сделать не может;
- Трудность поддержки тестов. Даже незначительное изменение в API или структуре DOM-дерева может повлечь за собой серьезные сбои в тестах и команда автоматизации тестирования будет вынуждена вносить правки в тесты;
- Дороговизна. Для заказчика проекты с автоматизацией стоят дороже, чем проекты, на которых работает только команда ручных тестировщиков;
- Стоимость инструмента для автоматизации. В случае если используется лицензионное ПО, его стоимость может быть достаточно высока. Сво-

бодно распространяемые инструменты как правило отличаются более скромным функционалом и меньшим удобством работы;

- Пропуск мелких ошибок — автоматический скрипт может пропускать мелкие ошибки, на проверку которых он не запрограммирован.

Применение автоматизированного тестирования на проектах — не всегда правильное решение. Иногда автоматизация может повлечь за собой ряд проблем, например, проблемы со сроками или проектным бюджетом. Нужно грамотно подходить к планированию тестовых работ. Ручное тестирование стоит использовать в случаях, когда проект большой и кратковременный [2].

Автоматизированное тестирование же нужно внедрять на проекты в следующих случаях:

- Проект долгосрочный;
- Команда тестирования обладает необходимыми навыками;
- На проведение регрессионного тестирования уходит очень много времени.

1.2 Практическая часть. Реализация инструмента для автоматизации тестирования web-приложений

В ходе работы было реализовано клиент-серверное приложение, позволяющее автоматизировать тестирование web-приложений.

Инструмент имеет 2 роли: администратор и тестировщик. Администратор имеет возможность создавать пользователей и давать им доступ к проектам, а также создавать проекты. Для администратора нет интерфейса и выполнение http-запросов выполняется из http-клиента (например, из приложения Postman). Роль тестировщика имеет пользовательский интерфейс, в котором можно создавать подпроекты, тестовые сценарии и тестовые случаи (тест-кейсы). Сценарии, тест-кейсы и шаги тест-кейса могут быть запущены из интерфейса, затем будет построен отчет, в котором для каждого шага будет содержаться время выполнения, а для тест-кейса помимо времени выполнения будет содержаться актуальный и фактический результат. Шаги, тест-кейсы и сценарии, которые закончились с ошибкой, будут выделены красным цветом.

Отчеты сохраняются в истории запуска для удобства отслеживания ошибок и времени их возникновения. Отчеты позволяют пользователю выяснить, какое требование было нарушено в тестируемом приложении.

В инструмент также можно добавлять дополнительные проверки, тем

самым расширять его возможности и функционал. Например, был реализован алгоритм Луна, проверяющий корректность номера введенной карты.

1.2.1 Серверная часть приложения

Серверная часть приложения представляет собой компонент, с помощью которого можно выполнять следующие задачи:

- Создавать проекты и пользователей с помощью API администратора;
- Проходить процесс авторизации пользователя;
- Создавать и редактировать подпроекты, тестовые сценарии, тест-кейсы и шаги;
- Запускать автоматические тесты по имеющимся шагам, тест-кейсам или тестовым сценариям;
- Получать отчеты по прохождению тестовых сценариев.

Фактически эта программа является Spring Boot приложением, при запуске которого автоматически поднимается контейнер сервлетов Apache Tomcat [3]. Само приложение собирается инструментом сборки Maven, в котором декларативно описаны все необходимые зависимости [4].

Была разработана модель данных, соответствующая предметной области. Для сохранения и повторного переиспользования тестовых данных используется база данных H2 [5]. Отображение модели данных на таблицы БД производится через реализацию механизма Java Persistence API (JPA) — Hibernate. Для обеспечения доступа к информации, хранящейся в БД, используются репозитории данных из фреймворка Spring Data JPA, предоставляющие набор стандартных методов для работы с базой.

Бизнес-слой приложения содержится в сервисах, предоставляющих возможность выполнения различных операций в браузере с использованием технологии Selenide. Выполнение бизнес-логики производится с объектами модели данных предметной области, переданными в сами сервисы.

Взаимодействие с клиентом выполняется через REST API, описанный в контроллерах. Данные в методы REST-сервиса передаются с клиента в формате JSON и преобразовываются в объекты модели данных, которые передаются в бизнес-слой приложения. В проекте используется Swagger — технология, которая позволяет документировать REST-сервисы и предоставляет UI для просмотра документации.

1.2.2 Клиентская часть приложения

Клиентская часть приложения является web-интерфейсом, позволяющим создавать и запускать тесты, а также просматривать отчеты об их выполнении, при помощи вызова REST API серверного приложения. Клиентская часть разработана с использованием библиотеки ReactJS и подхода Redux. Для вызова REST API используется библиотека axios [6]. Также используется библиотека Material UI с набором готовых компонент для верстки web-страниц приложения [7].

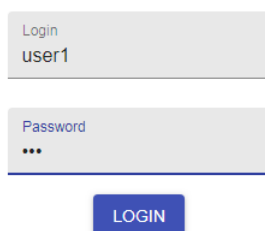
Код клиентской части разбит на несколько разделов:

- actions — Действия, используемые в компонентах;
- components — Компоненты без состояния;
- containers — Контейнеры, связывающие компоненты с состояниями;
- reducers — Набор механизмов, изменяющих состояние приложения.

1.3 Пользовательский интерфейс

В данном разделе будет продемонстрирован разработанный интерфейс пользователя с ролью тестировщика.

Страница login представлена на рисунке 1.



The image shows a login form with two input fields and a button. The first field is labeled 'Login' and contains the text 'user1'. The second field is labeled 'Password' and contains three dots '...'. Below these fields is a blue button with the text 'LOGIN' in white capital letters.

Рисунок 1 – Страница логина

Страницу с проектами можно осмотреть на рисунке 2. С этой страницы можно перейти в проект, где будут расположены подпроекты. Создавать проекты невозможно. У разных пользователей может быть разный список проектов.

Projects

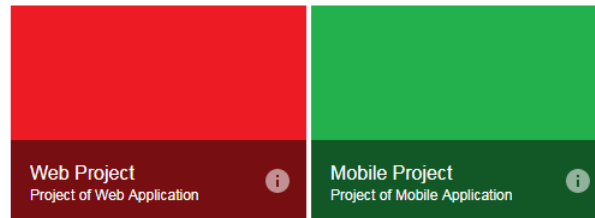


Рисунок 2 – Список проектов пользователя

На странице проекта можно увидеть список подпроектов, кнопку для создания нового подпроекта и кнопку домой для возврата на страницу со списком проектов. Страница проекта продемонстрирована на рисунке 3. Диалог создания подпроекта продемонстрирован на рисунке 4.

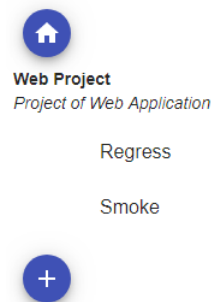


Рисунок 3 – Страница проекта

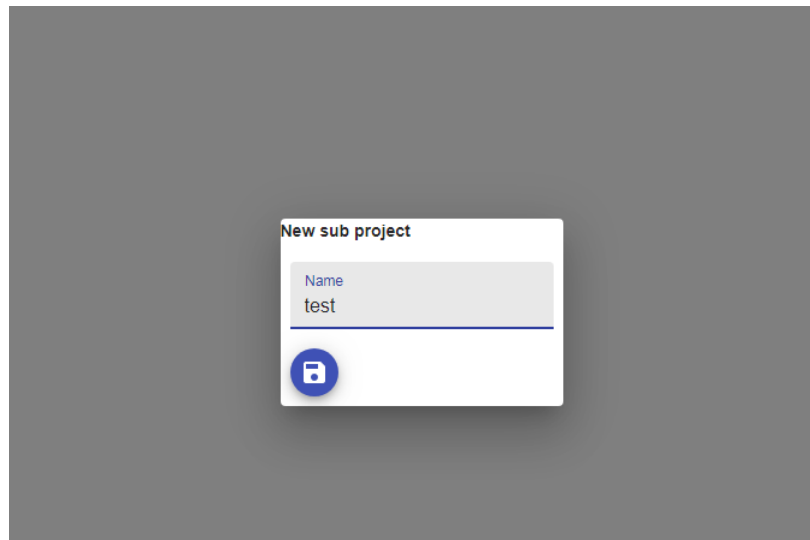


Рисунок 4 – Диалог создания подпроекта

На рисунке 5 продемонстрирована страница с подпроекта, на которой развернут сценарий. Подпроект состоит из множества сценариев, а сценарий состоит из тест-кейсов. На этой странице можно создать новый сценарий (кнопка + под списком сценариев), запускать выбранный сценарий, изменять и сохранять сценарий и добавлять новые тест-кейсы в сценарий. Если запустить сценарий, то на странице отобразится с отчет с результатами запуска. Результат запуска и история запусков отображается на рисунке 6.

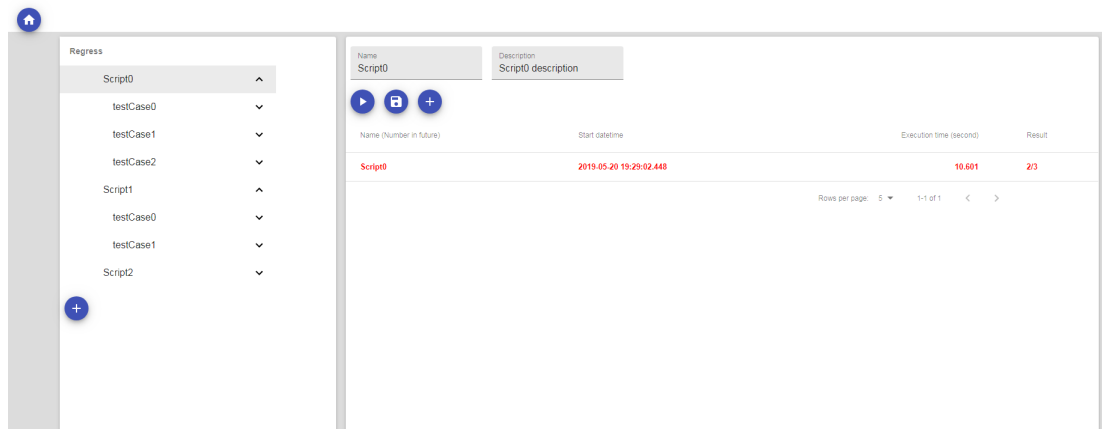


Рисунок 5 – Подпроект. Сценарий

Name (Number in future)	Start datetime	Execution time (second)	Result
General	2019-05-20 19:46:24.457	12.216	30
General	2019-05-20 19:42:31.394	16.449	20

Рисунок 6 – История запусков сценария

Страница, содержащая тест-кейс, показана на рисунке 7. Тест-кейс состоит из множества шагов. На этой странице можно создать новый сценарий (кнопка + под списком сценариев), запускать выбранный сценарий, изменять и сохранять тест-кейс и добавлять новые тест-кейсы в сценарий. Тест-кейс состоит из названия, описания, ожидаемого результата и типа проверки. Тип проверки может быть Equals, Not Contains, Not visible, Luhn algorithm и поля комментария и value. В value необходимо написать локатор элемента.

Рисунок 7 – Страница с тест-кейсом

Страница описания шага продемонстрирована на рисунке 8. На этой странице можно создать новый сценарий (кнопка + под списком сценариев), запускать выбранный шаг, изменять и сохранять шаг. Шаг состоит из названия, операции и набора полей, который разный для операций (например, для операции Input появляются поля Component, Text и Comment). Тип оператора может быть Input, Open и Click. В component необходимо написать локатор элемента или сайт в зависимости от типа оператора.

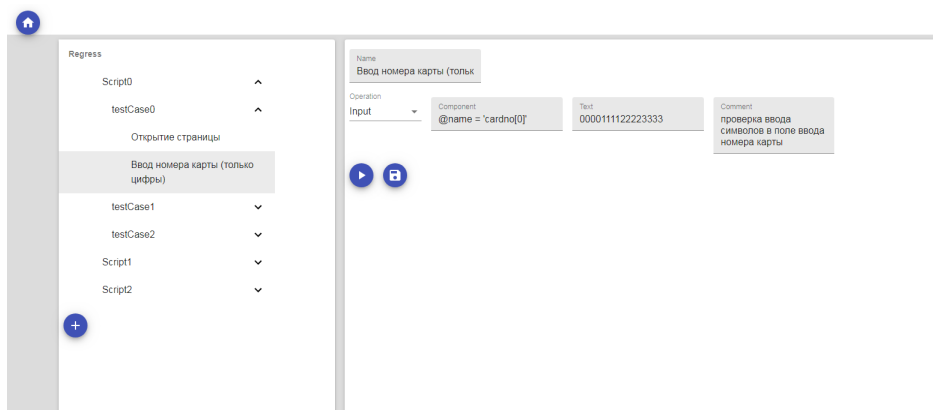


Рисунок 8 – Страница шага тест-кейса

Страница отчета отображается на странице 9. На странице видно, какие тест-кейсы были выполнены успешно, а какие с ошибкой. Время выполнения каждого шага также можно посмотреть в составленном отчете.

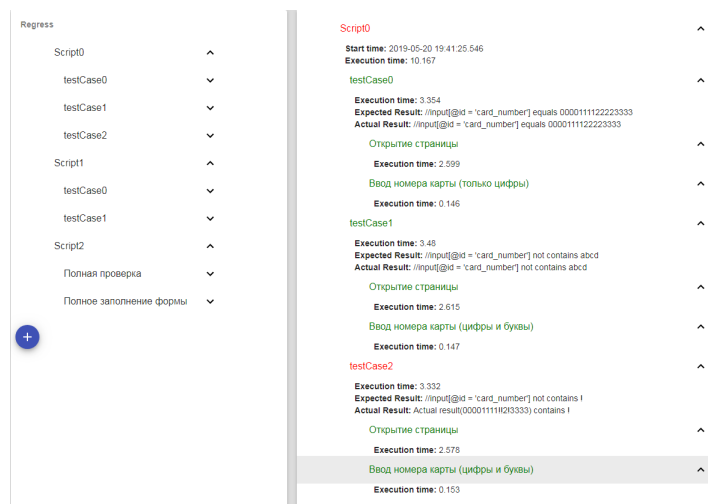


Рисунок 9 – Отчет о выполнении скрипта

ЗАКЛЮЧЕНИЕ

Итогом работы стало приложение, позволяющее писать и выполнять автотесты.

В ходе выполнения работы были выполнены следующие задачи:

- рассмотрены основные понятия теории тестирования программного обеспечения;
- изучены инструменты для тестирования, такие как Selenium, Screenster, PhantomJS, и выяснены их плюсы и минусы;
- разработана программа для автоматизации тестирования программного обеспечения, включающая в себя API-методы работы с тестовыми сценариями, набор доступных операций и интерфейс пользователя;
- реализована возможность просмотра отчета тестирования и истории выполнения тестов в интерфейсе приложения;
- реализовано разграничение доступа пользователя по проектам, для этого создать API-методы для создания пользователей и проектов;
- применен на практике разработанный инструмент — создан набор тестовых случаев.

Практическая значимость разработанного приложения в том, что оно позволяет автоматизировать рутинные задачи без написания кода и анализировать выполненные тесты. Автоматизировать можно любое web-приложение независимо от его функциональности. Код приложения можно пополнять дополнительными методами пользователя для тестирования того или иного функционала. Таким образом с помощью инструмента можно повышать качество тестируемого сайта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 *Савин, Р.* Тестирование Дот Ком / Р. Савин. — Москва: Дело, 2007.
- 2 *Канер, С.* Тестирование программного обеспечения / С. Канер. — Киев: Диа-софт, 2001.
- 3 *Guthrie, J.* Spring Microservices in Action / J. Guthrie. — New York: Manning, 2017.
- 4 Apache Maven Project [Электронный ресурс]. — URL: <https://maven.apache.org/> (Дата обращения 20.04.2019). Загл. с экр. Яз. рус.
- 5 Embedded H2 Database [Электронный ресурс]. — URL: <https://confluence.atlassian.com/conf59/embedded-h2-database-792499591.html> (Дата обращения 29.04.2018). Загл. с экр. Яз. англ.
- 6 Axios Module [Электронный ресурс]. — URL: <https://axios.nuxtjs.org/> (Дата обращения 01.05.2019). Загл. с экр. Яз. рус.
- 7 Material-UI [Электронный ресурс]. — URL: <https://material-ui.com/> (Дата обращения 01.05.2019). Загл. с экр. Яз. рус.