МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка системы автоматизации тестирования для программного обеспечения

Обучающийся А.Д. Мотырева
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н Рогова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Бакалаврская работа выполнена на тему «Разработка системы автоматизации тестирования для программного обеспечения».

Цель данной работы – разработка системы автоматизации тестирования для программного обеспечения.

Во введении раскрываются актуальность исследования, объект и предмет работы, а также ее цель и поставленные задачи.

Первая глава посвящена анализу компании «Metropolis», моделированию бизнес-процесса, описывающего работу системы автоматизации тестирования. Также определены требования к разрабатываемому программному продукту и проведена оценка существующих решений для выполнения поставленных задач.

Во второй главе представлено проектирование программного продукта: разработаны UML-диаграммы, а также построены концептуальная и логическая модели базы данных.

Третья глава включает реализацию нескольких сценариев функционирования системы, а также анализ экономической эффективности внедрения автоматизированного тестирования программного обеспечения.

Бакалаврская работа состоит из 52 страницы и включает 33 рисунка, 6 таблиц, 20 источников.

Оглавление

Введение

Тема выпускной квалификационной работы «Разработка системы автоматизации тестирования для программного обеспечения».

Любая разработка и внедрение новых программных продуктов требует значительных материальных и временных затрат. Привлекаемые финансовые и материальные средства должны обеспечить надлежащий эффект и отдачу. Именно поэтому возникает необходимость повышения качество выпускаемого программного обеспечения, сокращение времени на его разработку, тестирование и сопровождение.

Рыночные отношения требуют, чтобы программное обеспечение для их заказчика было качественным и подготовленное в короткий срок. Следовательно, продуктивно должны работать и разработчики программного обеспечения, то есть программисты, и тестировщики этого программного обеспечения. Добиться продуктивности в этом вопросе (качества, быстроты, минимальных затрат) можно за счет дальнейшей автоматизации и модернизации бизнес-процессов с помощью программных приложений.

«Автоматизация работ по тестированию имеет огромную ценность особенно там, где тестовые скрипты повторяются. Такое тестирование на стадиях разработки и интеграции, когда повторно используемые скрипты могут выполняться много раз, весьма эффективно»[8].

Объект исследования – процесс тестирования программного обеспечения.

Предмет исследования – методики и инструменты для автоматизации процесса тестирования программного обеспечения.

Цель работы – разработка системы автоматизации тестирования для программного обеспечения.

Задачи исследования:

– проанализировать проблему, затронутую в теме исследования, и обосновать ее актуальность;

– провести сравнительный анализ существующих аналогов;

– сформулировать концепцию предлагаемого решения;

– спроектировать базу данных для информационной системы;

– разработать прототип информационного обеспечения и программного модуля;

– протестировать работу программы в различных сценариях.

Методы исследования:

– анализ данных;

– алгоритмы функционального и автоматизированного тестирования;

– проектной разработки баз данных.

Работа состоит из введения, трех глав, в которых рассматриваются вопросы постановки задачи, проектирования и реализации системы автоматизированного тестирования, а также заключение, список литературы и источников.

Глава 1 Постановка задачи на разработку системы автоматизации тестирования для программного обеспечения

1.1 Значение системы автоматизации тестирования для программного обеспечения предприятия

Предприятие на базе, которого проходила разработка выпускной квалификационной работы – компания «Metropolis». «Metropolis» работает на рынке разработки корпоративного программного обеспечения с 2008 года.

«Metropolis» ведет проекты на всех стадиях проектирования – концепция, проектная документация, тендерная документация, рабочая документация, авторский надзор и технический аудит.

«Основным видом экономической деятельности компании является 62.02 «Деятельность консультативная и работы в области компьютерных технологий». Также «Metropolis» работает еще по нескольким направлениям. Дополнительные виды деятельности связаны напрямую с продажами программного и аппаратного обеспечения, а также с деятельностью по настройке программного обеспечения и обучению пользователей» [1]:

– 46.51 – торговля розничная компьютерами, периферийными устройствами к ним и программным обеспечением в специализированных магазинах;

– 58.29 – издание прочих программных продуктов;

– 62.01 – разработка компьютерного программного обеспечения;

– 62.02.1 – деятельность по планированию, проектированию компьютерных систем;

– 62.02.9 – деятельность консультативная в области компьютерных технологий прочая;

– 62.03.12 – деятельность по управлению компьютерными системами дистанционно;

– 62.09 – деятельность, связанная с использованием

вычислительной техники и информационных технологий, прочая;

– 63.99.1 – деятельность по оказанию консультационных и информационных услуг.

Организационная структура компании «Metropolis» представлена на рисунке 1. Численность персонала компании – 50 человек.

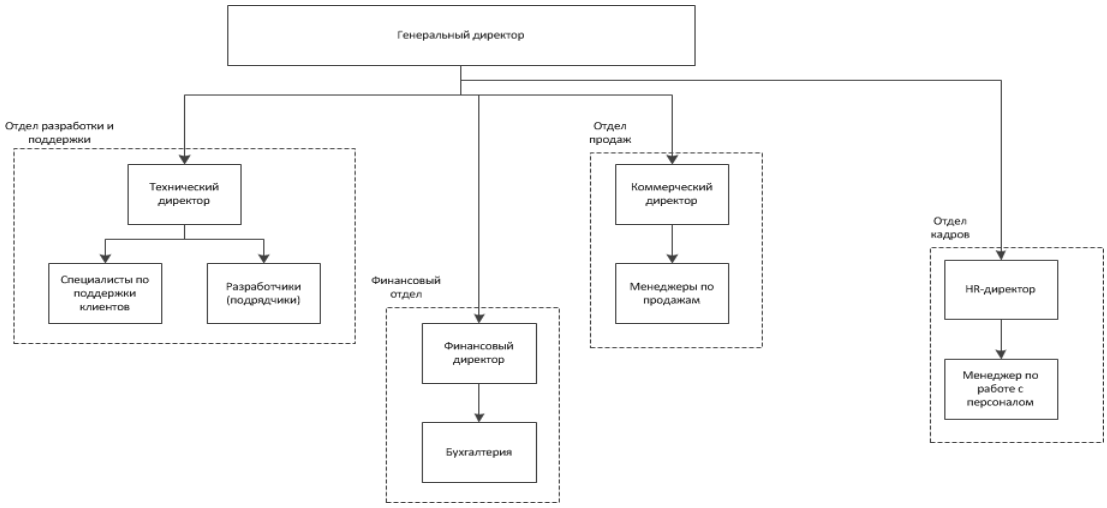


Рисунок 1 – Организационная структура предприятия

В ходе выполнения выпускной квалификационной работы рассматривалась работа отдела разработки программного обеспечения. «В отдел разработки входит: технический директор, команда разработчиков и тестировщиков программного обеспечения, они отвечают за следующие функции»[3]:

– «разработка и постановка технических задач команде разработки подрядчика;

– контроль выполнения технических задач со стороны подрядчика;

– техническая поддержка пользователей»[4].

В выпускной квалификационной работе необходимо разработать систему автоматизации тестирования для программного обеспечения, необходимость в такой разработке обусловлена следующими причинами:

– повышение качества программных продуктов, которые разрабатываются в компании. Ожидается, что разработка системы позволит снизить время между написанием кода и его проверкой.

– снижение затрат на ручное тестирование, благодаря хранению тестов в системе и созданию универсальных тестов;

– документирование тестов, а именно его автоматизация, хранения, поиск по описанию позволит уменьшить время работы тестировщика. А для руководителей позволит отслеживать количество проведенных тестов и собирать агрегированные отчеты.

Этапы разработки системы автоматизированного тестирования содержать несколько блоков, которые показаны на рисунке 2.

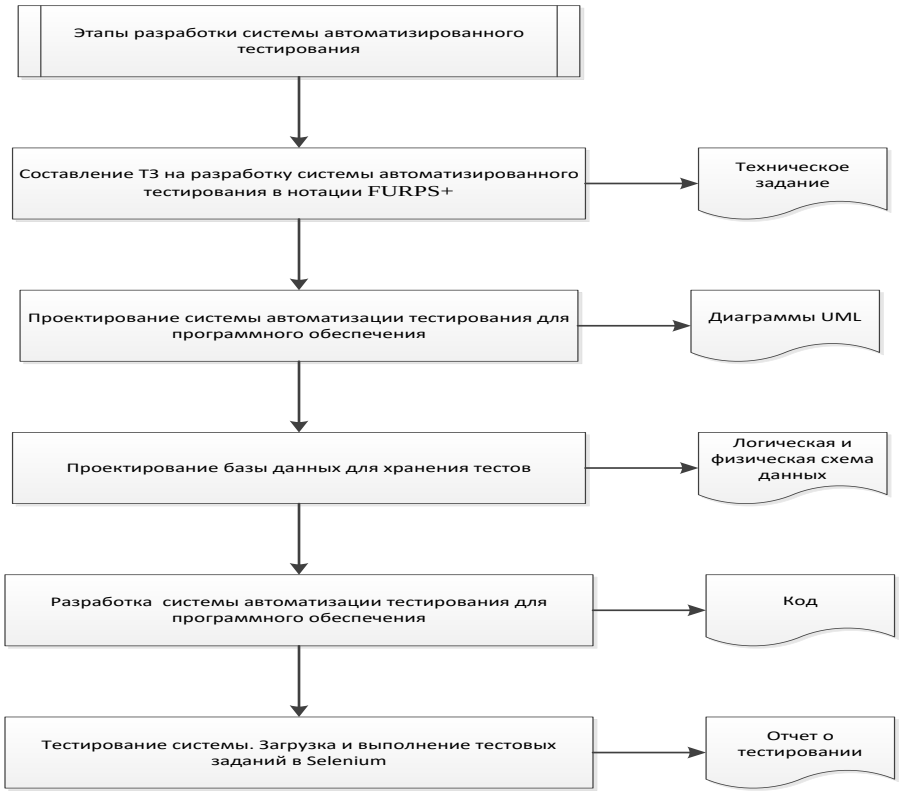


Рисунок 2 – Этапы разработки системы автоматизированного тестирования

На первом этапе «Составление ТЗ на разработку системы автоматизированного тестирования в нотации FURPS+», необходимо

провести анализ требований и проблем, которые будут решены в системы. Результатом данного этапа будет техническое задание на разработку.

На втором этапе, при написании выпускной квалификационной работы спроектировано описание, поведение системы, выделены сотрудники, которые будут работать с системой автоматизированного тестирования. Результатом этого этапа будет диаграммы UML.

На третьем этапе спроектирована база данных, задачей которой будет хранение тестов и результатов проведения тестирования.

На следующем этапе будет разработан код, для загрузки тестов, результатом этого этапа является код тестов.

И на заключительном этапе формируются отчеты о проведении тестов.

При разработке схемы этапов, из которой состоит разработка системы автоматизированного тестирования, учитывалось, что в компании могут быть разработаны и автотесты и тесты для ручного тестирования. Стоит отметить какие бы тесты не использовали при разработки программного кода на фазе «Тестирования». Тестирование является этапом гарантии качества, всего разработанного кода. И автоматизация данного этапа позволит добиться следующих экономических показателей, которые были получены в ходе дальнейшей опытной эксплуатации системы, показаны на рисунках 3-5.

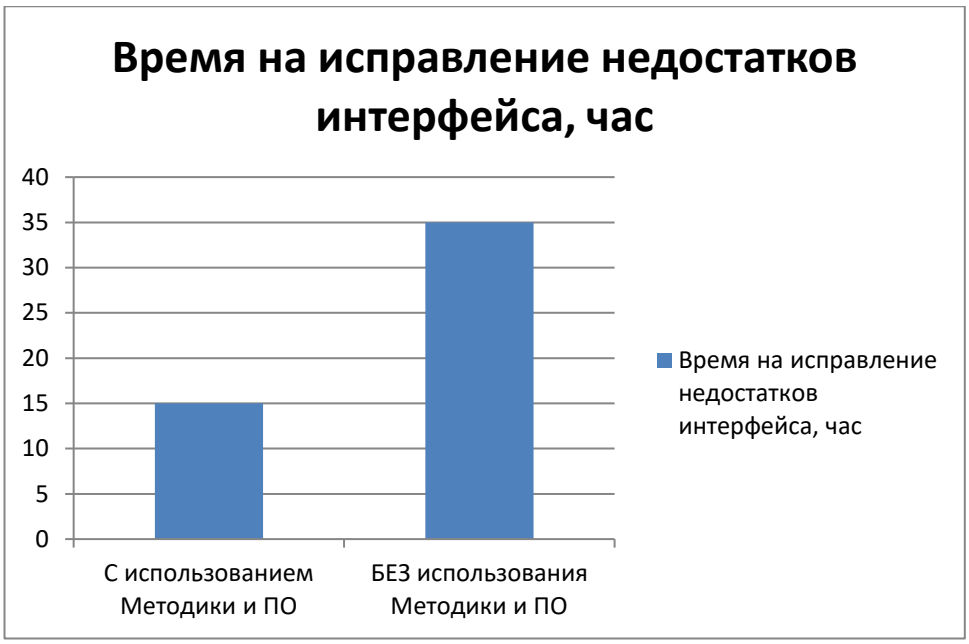


Рисунок 3 – Время на поиск ошибок кода, после использования системы

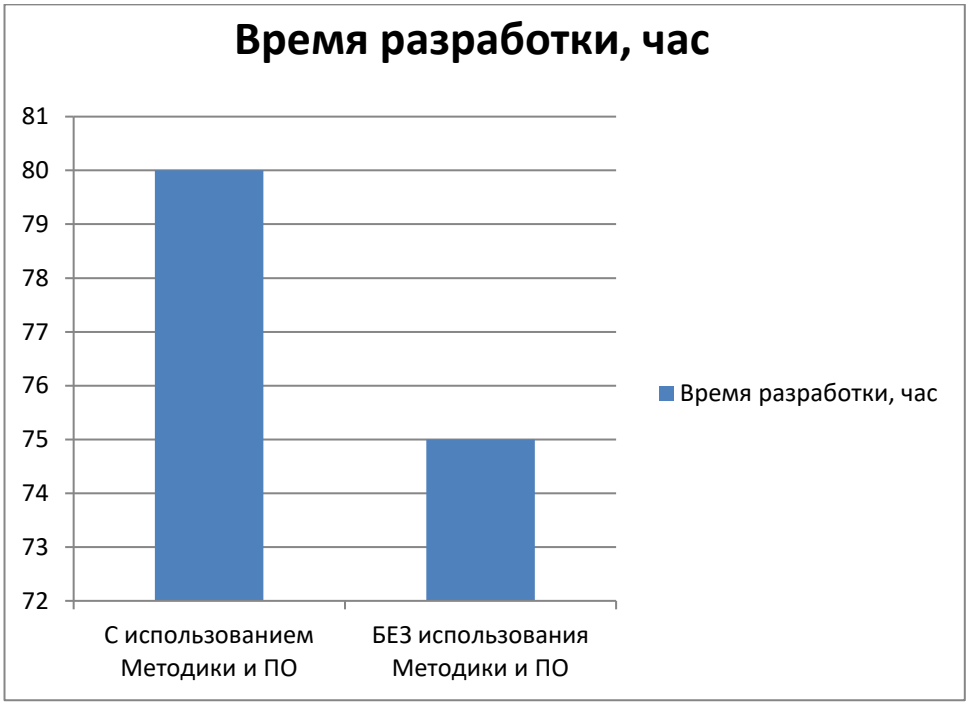


Рисунок 4 – Время разработки теста

Рисунок 5 – Удовлетворенность качеством документации

Как видно из приведенных графиков, хоть время разработки тестов с использование предложенной системы увеличивается (так как каждый тест и его результат необходимо загрузить в систему), но за счет уменьшения времени на поиск ошибок и высокими показателями качества получаемой документации, разработку предложенной системы можно считать эффективной и актуальной задачей.

1.2 Основные функции системы автоматизации тестирования для программного обеспечения предприятия

Применение методологии FURPS+ позволило сформулировать требования к разработке информационной системы, которая обеспечит:

- полноту функционала для автоматизации задач тестировщика компании.
- удобство и безопасность работы пользователей.
- высокую производительность и надежность системы.
- простоту поддержки и возможность дальнейшего развития.

Эти требования станут основой для создания высококачественной информационной системы, которая повысит эффективность работы компании и упростит выполнение административных процессов.

Требования к системе автоматизации тестирования для программного обеспечения в методологии FURPS+ показаны в таблице 1.

Таблица 1 – Требования к системе автоматизации тестирования для программного обеспечения в методологии FURPS+

Вид требований	Содержание требований
F. Функциональные требования к системе автоматизации тестирования для программного обеспечения	− разработка авто тестов − поддержка регрессионных тестов − просмотры входных и выходных результатов тестов
U. Удобство использования системе автоматизации тестирования для программного обеспечения	− работа с разными операционными системами, − выгрузка отчетов в различные системы командной работы
R. Надежность системе автоматизации тестирования для программного обеспечения	– резервное копирование; – хранение результатов тестирования с ведением детализированного лога, что позволяет анализировать выполнение тестов и выявлять ошибки
P. Производительность системе автоматизации тестирования для программного обеспечения	− Потребление ресурсов.
S. Поддерживаемость системе автоматизации тестирования для программного обеспечения	− Масштабируемость (до 10 подключений)
+. Ограничения системе автоматизации тестирования для программного обеспечения	− Гибкость в работе с данными (текст, документы Word, рисунке в формате *.bmp)

После определение требования требований к разрабатываемой системе переходим к ее разработке.

1.3 Обзор существующих систем автоматизации тестирования для программного обеспечения предприятия и их анализ

В ходе исследования предметной области были проанализировано множество информационных источников (сайтов), несколько информационных систем связанных с автоматизацией тестирования и были выбраны следующие аналоги:

«IBM Rational Functional Tester — инструмент для автоматизированного функционального и регрессионного тестирования, а также тестирования UI и на основе данных. Использует технологию ScriptAssure и шаблонный поиск для повышения устойчивости тестов при изменениях интерфейса. Поддерживает сценарии на Java (Eclipse) и Visual Basic .Net (Visual Studio .Net)»[2].

«Robot Framework – это фреймворк, предназначенный в первую очередь для автоматизации приемочного тестирования, но его возможности выходят далеко за эти рамки. Robot Framework использует концепцию keyword-driven тестирования. Подход, при котором разрабатываются ключевые слова, которые потом могут использоваться для построения автоматических тестов. На основании таких ключевых слов можно легко построить другие тесты с разными тестовыми данными на входе, или же превратить тесты в читаемый и исполняемый текст. RobotFrameworkимеет в своем наличии библиотеки для автоматизации веб приложений, баз данных, действий с файловой системой, SSH, Swing, SWT, Windows GUIs и многое другое**.** Архитектура Robot Framework построена таким образом, чтобы расширять существующую функциональность и писать собственные библиотеки на Python или Java»[7].

«SmartBear Software TestComplete – это инструмент для автоматизации тестирования, позволяющий создавать и выполнять тесты для любых Windows и Web-приложений, как простых, так и обладающих богатой функциональностью. TestComplete может выполнять в запланированное время. Это позволяет более эффективно распределять нагрузку на аппаратные

средства. Автоматизированные тесты могут выполняться днем и ночью, в наиболее подходящее время без участия оператора. Минусы TestComplete:

– не поддерживаются в API работы с хранилищем логов при остановленном скрипте;

– сложная работа с таблицами в веб-приложениях;

– нет поддержки third-party» [5].

В ходе аналитической работы были тщательно исследованы готовые рыночные решения, способные удовлетворить текущие и перспективные потребности предприятия. Для этого использовались критерии, такие как стоимость владения, гибкость и масштабируемость системы, возможность интеграции с существующими компонентами ИТ-инфраструктуры компании «Metropolis», а также уровень технической поддержки и обновлений. Однако анализ показал, что большинство коммерчески доступных ИС рассчитаны на стандартные процессы тестирования и имеют ограниченную способность к адаптации под специфические нужды предприятия, что является критичным недостатком.

Следующим этапом стала оценка варианта приобретения ИС с последующей доработкой под нужды компании «Metropolis». Этот подход потенциально позволяет использовать преимущества стандартного решения, сопрягая его с гибкостью кастомизации. Тем не менее, детальный разбор данного пути выявил значительные риски, связанные с неопределенностью сроков и бюджета на доработку, а также с вероятностью возникновения сложностей при обновлении и поддержке модифицированной системы.

В ходе анализа был также рассмотрен вариант самостоятельной разработки ИС. Исходя из специфики деятельности компании «Metropolis», данная стратегия представляется особенно привлекательной, так как позволяет полностью адаптировать систему под уникальные процессы и методы тестирования, используемые в компании. Собственная разработка предоставляет возможность полного контроля над функционалом, безопасностью и интеграцией системы, но требует значительных начальных

инвестиций и квалифицированных ресурсов для реализации и последующего сопровождения.

Исходя из специфики деятельности предприятия и необходимости обеспечения максимальной адаптивности системы к уникальным бизнес-процессам организации, принято решение «склониться к разработке собственной информационной системы. Этот подход предполагает создание ПО, которое идеально соответствует всем требованиям и особенностям работы предприятия, что обеспечивает максимальную эффективность и гибкость внедряемого решения»[20].

Выводы по главе 1

В первой главе выпускной квалификационной работы рассмотрены общие вопросы бизнес-процессов работы системы тестирования.

Определена потребность в разработки системы автоматизации тестирования для программного обеспечения.

Глава 2 Процесс разработки системы автоматизации тестирования для программного обеспечения предприятия

2.1 Проектирование системы автоматизации тестирования для программного обеспечения

«На этапе концептуального моделирования информационных систем можно использовать несколько подходов, но самой распространенной является технология UML»[9].

UML – это «объектно-ориентированный язык со следующими характеристиками:

- модель – объект, отображающий наиболее значимые для конкретной задачи характеристики системы. Модели бывают разные – нематериальные и материальные, естественные и искусственные, математические и декоративные,
- подсистема — показывает поведения других элементов,
- система - совокупность управляемых взаимосвязанных подсистем, которых объединили с общей целью» [6]

Все варианты использования, связаны с требованиями к функциональности разрабатываемой системы (рисунок 6).

Основные функции системы включают в себя следующие модули:

- модуль автоматизации подготовки тестовых сценариев. Данный модуль обеспечивает создание, редактирование и управление тестовыми сценариями, позволяя тестировщикам эффективно распределять ресурсы и временные затраты на подготовительный этап.
- модуль запуска тестов. Интегрирован с системой непрерывной интеграции (CI), позволяет инициировать тестовые прогоны как в автоматическом режиме при создании pull-request-ов и слиянии изменений в основную ветку, так и вручную.
- модуль сбора и анализа результатов. Систематизирует полученные

данные по завершении тестовых прогонов, формируя отчеты, которые позволяют быстро выявлять проблемные участки в коде и улучшать качество продукта.

− модуль управления версиями тестов. Поддерживает трассировку изменений тестовых скриптов, обеспечивая возможность отката к предыдущим версиям и анализа истории модификаций.

− модуль взаимодействия с базой данных. Обеспечивает хранение, поиск и извлечение данных, необходимых для инициализации и проведения тестов, а также хранение результатов тестирования.

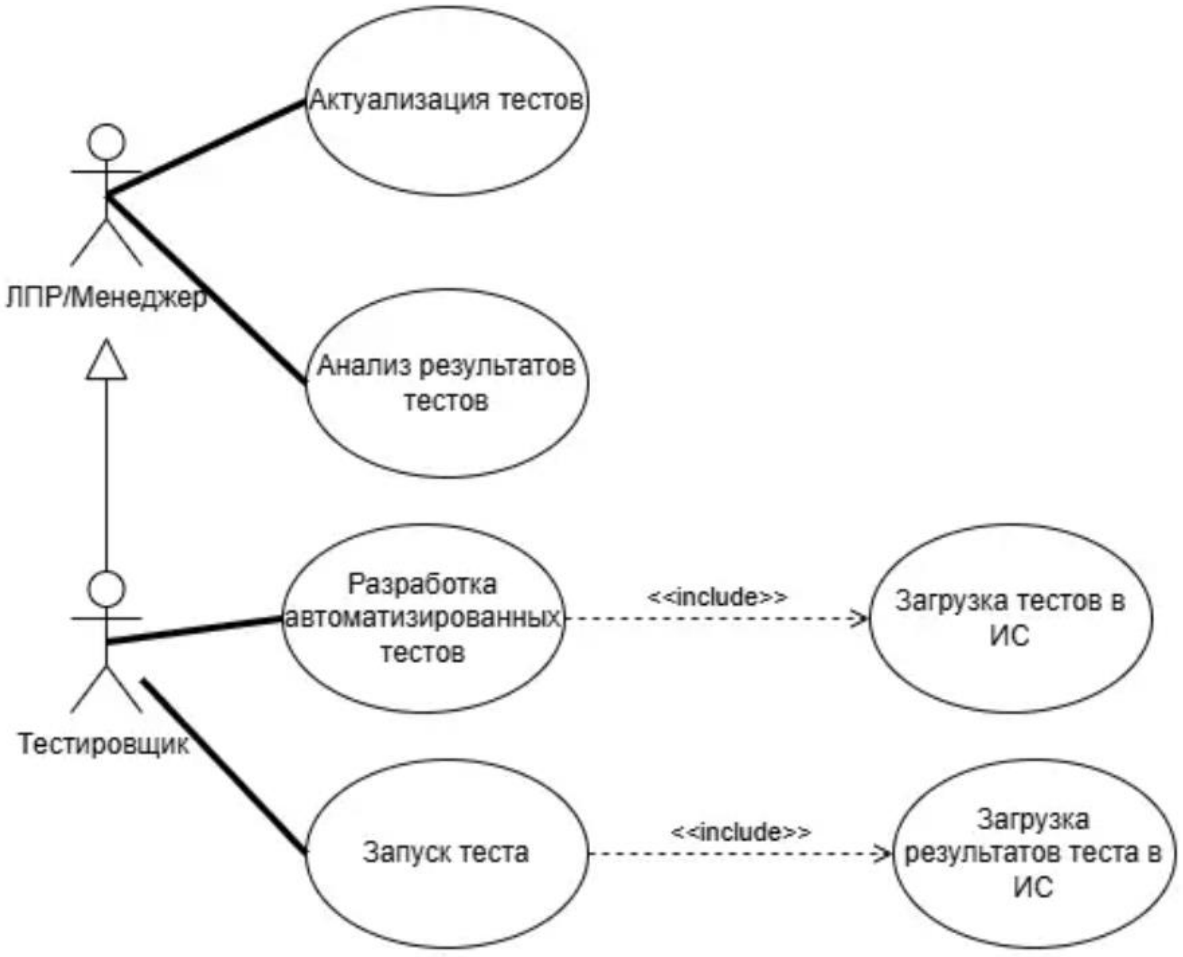


Рисунок 6 – UML диаграмма взаимодействия с пользователем

Тестировщики выполняют ключевые функции, включая разработку автоматизированных тестов, запуск тестов, а также актуализацию тестов в соответствии с последними изменениями в программном продукте. Помимо этого, они анализируют результаты тестирования для выявления потенциальных ошибок и оценки качества продукта. Процесс тестирования включает в себя динамичное взаимодействие с модулями разработки, что требует глубокой интеграции между отделами разработки и тестирования.

Менеджеры проектов контролируют и направляют процесс тестирования, обеспечивая его соответствие установленным требованиям и стандартам качества. Они также отвечают за организацию рабочего процесса, распределение ресурсов и координацию между участниками проекта.

Эта структура позволяет рационально организовать работу отдела тестирования, оптимизировать расход ресурсов и минимизировать время на выявление и устранение дефектов ПО, ускоряя процесс разработки и повышая его эффективность.

Для описания функциональных возможностей системы построены диаграммы состояний и диаграммы деятельности, которые отражают процесс тестирования (рисунок 7-8)

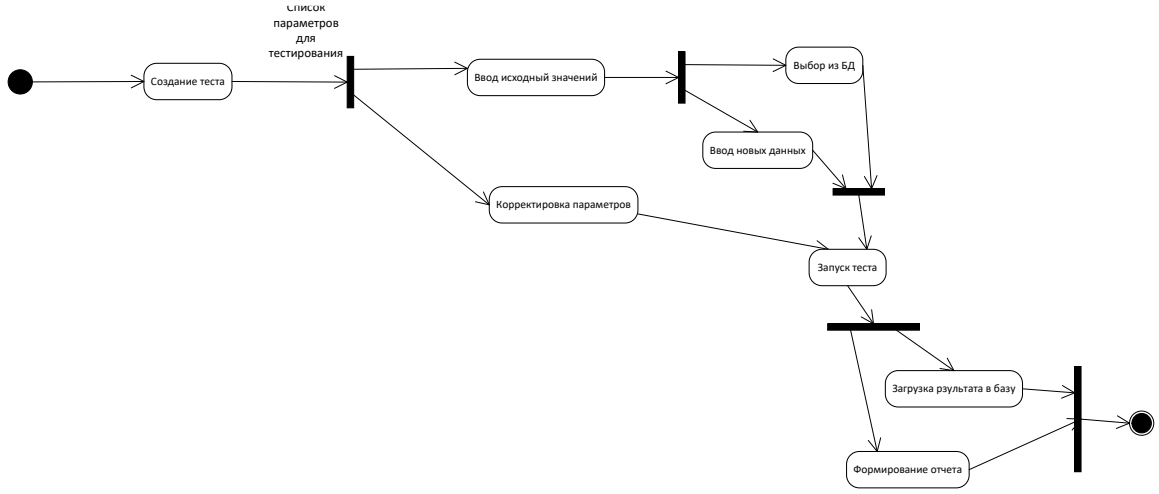


Рисунок 7 – Диаграмма состояний «Разработка теста»

При разработке тестов, процесс начинается с создания теста, далее выбираются параметры тестирования, вводятся исходные значения или при корректировки или процессе регрессионного тестирования производится корректировка параметров. Из состояния после того как введены все исходные значения тестировщику надо принять решение выбрать тесты из базы информационной системы или сохранить как новый тест. После того как значение параметров теста определены происходит запуск теста. И далее происходит два действия параллельно, данные результата теста будут сохранены и результат станет доступен в виде отчета или так называемого тест-кейса. Результаты могут быть просмотрены как тестировщиком, так и лицом принимающим решение.

Далее на рисунке показана диаграмма деятельности (рисунок 8)

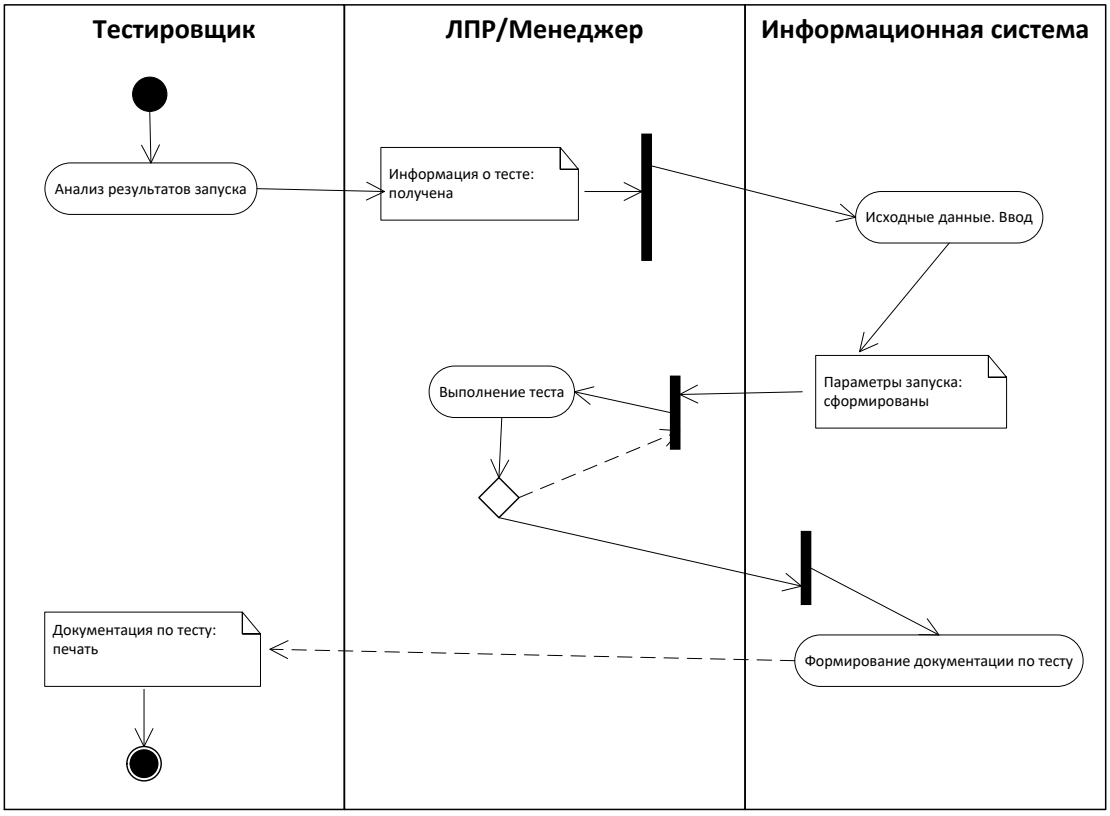


Рисунок 8- Диаграмма деятельности

Диаграмма деятельности отражает процесс анализа полученных

результатов тестирования и участие в нем Тестировщика, Менеджера и разработанной информационной системы. Процесс запускается Тестировщиком после того как результаты получены и уже могут быть проанализированы и распечатаны. Менеджер имеет доступ ко всей информации о тесте, о его целях и задачах, о параметрах теста, и результатах его выполнения. Так же благодаря информации из системы Менеджер имеет возможность анализировать такие параметры, как количество разработанных тестов тестировщиком, количество модифицированных тестов, количество и качество полученных результатов тестирования.

На рисунке 9 показана диаграмма компонентов.

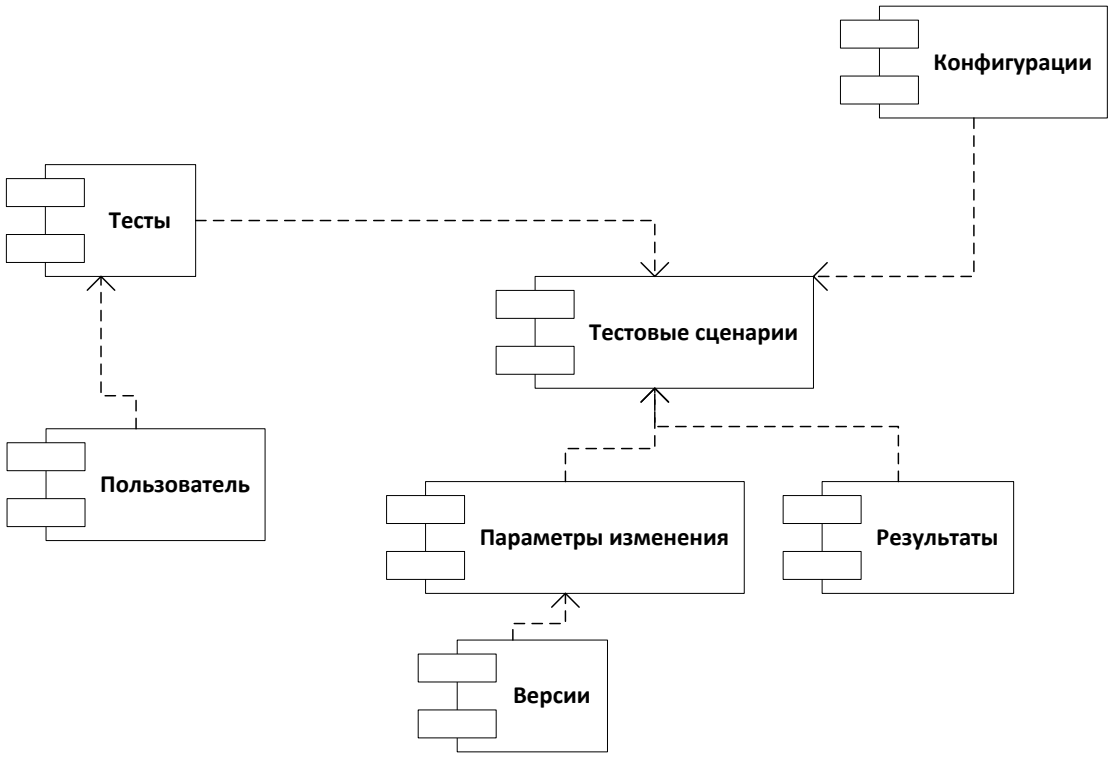


Рисунок 9- Диаграмма компонентов

На диаграмме показаны следующие компоненты, которые необходимы:

- тесты – для проверки работоспособности ПО,
- пользователь – тестировщики, разрабатывающие тесты,
- тестовые сценарии – различные сценарии работы тестов,

– параметры, изменяемые в тестовых сценариях,

– версии – класс необходимый для регрессионного тестирования,

– результаты, которые получаются при выполнении тестов,

– конфигурации программного обеспечения и технических устройств, на которых запускаются тесты.

Основными преимуществами автоматизации тестирования будет:

– сокращение количества ошибок;

– сокращение объема теста.

Основные таблицы базы данных (рисунок 10):

Таблица TestCases (Тест-кейсы):

– поле id: первичный ключ, уникальный идентификатор тест-кейса;

– поле name: название тест-кейса;

– поле description: описание тест-кейса;

– поле created_at: дата и время создания тест-кейса;

– поле updated_at: дата и время последнего обновления тест-кейса.

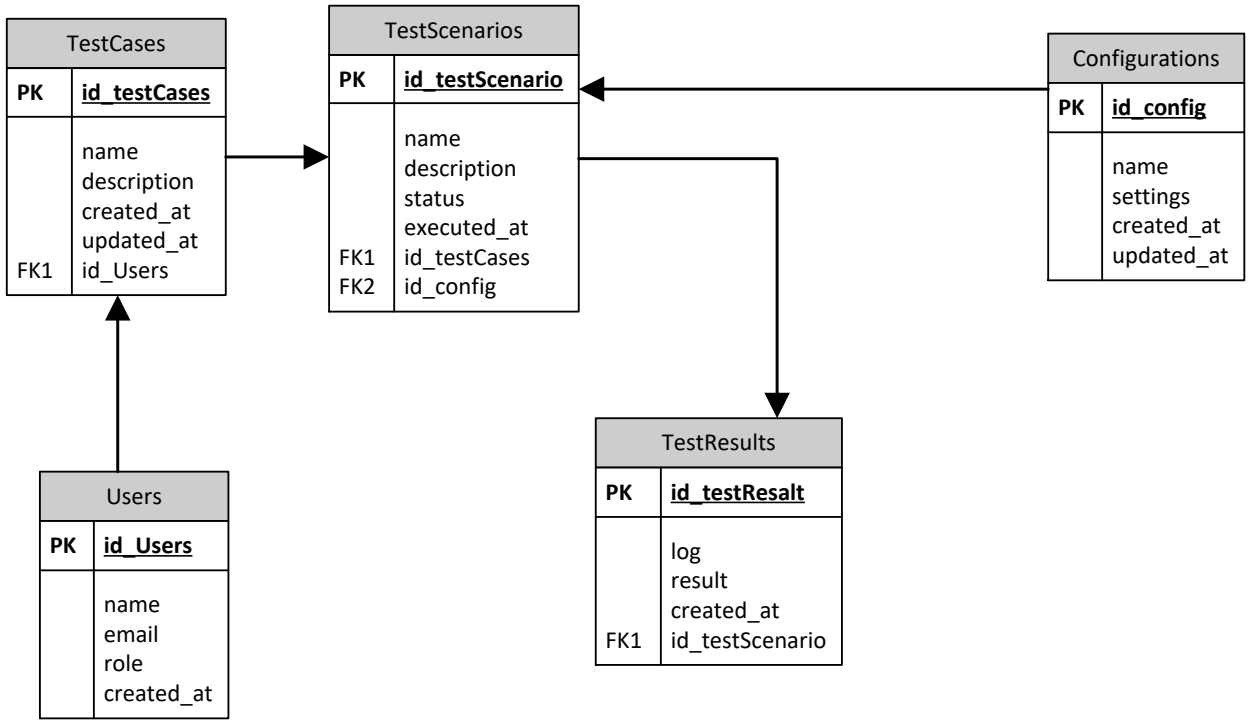


Рисунок 10 – ER-диаграмма

Таблица TestScenarios (Тестовые сценарии):

- поле id: первичный ключ, уникальный идентификатор тестового сценария;
- поле test_case_id: внешний ключ, идентификатор связанного тест-кейса.
- поле name: название тестового сценария;
- поле description: описание тестового сценария;
- поле status: статус выполнения сценария (например, "Passed", "Failed");
- поле executed_at: дата и время выполнения сценария;

Таблица TestResults (Результаты тестирования):

- поле id: первичный ключ, уникальный идентификатор результата тестирования;
- поле test_scenario_id: внешний ключ, идентификатор связанного тестового сценария;
- поле log: лог выполнения теста;
- поле result: результат теста (например, "Success", "Error");
- поле created_at: дата и время создания результата тестирования.

Таблица Users (Пользователи):

- поле id: первичный ключ, уникальный идентификатор пользователя;
- поле name: имя пользователя;
- поле email: адрес электронной почты пользователя;
- поле role: роль пользователя (например, "Admin", "Tester");
- поле created_at: дата и время создания записи о пользователе.

Таблица Configurations (Конфигурации):

- поле id: первичный ключ, уникальный идентификатор конфигурации;
- поле name: название конфигурации;
- поле settings: настройки конфигурации;

– поле created_at: дата и время создания конфигурации;

– поле updated_at: дата и время последнего обновления конфигурации.

Взаимосвязи между таблицами:

– связь между TestCases и TestScenarios: один тест-кейс может иметь множество тестовых сценариев. Это связь один ко многим.

– связь между TestScenarios и TestResults: один тестовый сценарий может иметь множество результатов тестирования. Это связь один ко многим.

– связь между Users и TestCases: один пользователь может создать множество тест-кейсов. Это связь один ко многим.

– связь между Configurations и TestScenarios:

В результате получилась следующая физическая схема базы данных (рисунок 11).

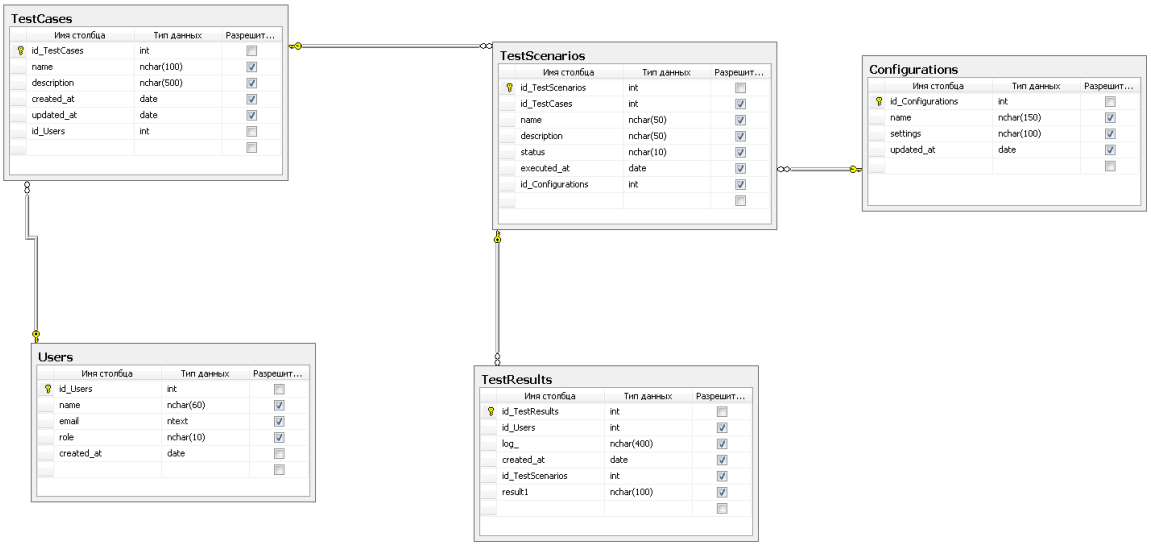


Рисунок 11 – Физическая модель баз данных

После проектирования схем и базы данных переходим к разработке системы.

2.2 Выбор технологий и инструментов для разработки системы автоматизации тестирования для программного обеспечения

«На сегодняшний день выделяют следующие виды архитектур:

– клиент-серверная архитектура в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

– Web – сервис – это идентифицируемая веб-адресом программная система со стандартизированными интерфейсами.

– программы для мобильных устройств»[10].

«В рамках рассматриваемой задачи, более подходящей является клиент-серверная архитектура, так как программа будет использоваться в локальной сети, работниками одной компании»[11].

Разрабатываемые компоненты системы тестирования представляют собой упорядоченный и структурированный набор команд, при передаче которых в систему автоматизированного тестирования Selenium, интерпретируются и передаются в web-приложение.

Web-приложение, в свою очередь получив команду от ядра Selenium начинает процесс её выполнения. Selenium формирует отчет о прохождении загруженного компонента. Компонент считается успешно выполненным, если Selenium осуществил передачу всех команд из компонента и получил во всех случаях ожидаемый результат.

Требования к программному обеспечению:

- «Операционная система— Linux;
- web-сервер— Apache, Tomcat;
- система управления базами данных— SQL Server;
- язык реализации— Java»[19].

На рисунке 12 показана диаграмма развертывания системы

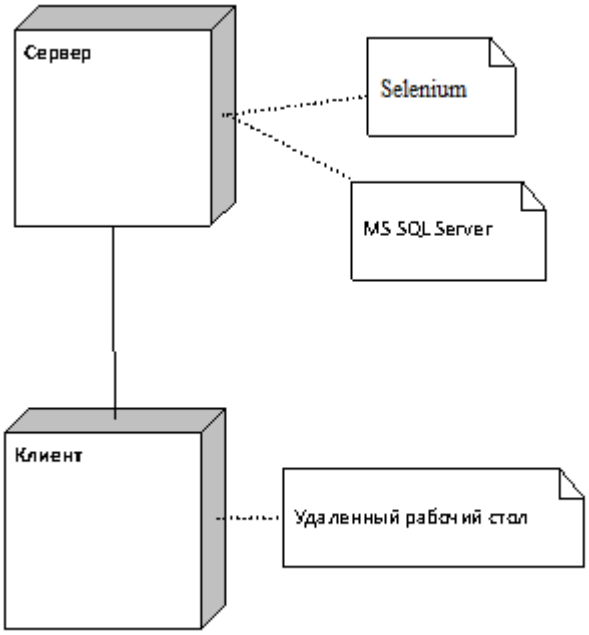


Рисунок 12 –Диаграмма развертывания

В ходе работы над выпускной квалификационной работы было разработаны компоненты для функционального тестирования интернет-магазина, которые были загружены в базу данных системы автоматизации тестирования.

2.3 Реализация функциональных требований системы автоматизации тестирования для программного обеспечения

Для тестирования административной стороны интернет-магазина были разработаны следующие компоненты:

Компонент тестирования с каталогом магазина, осуществляющий проверку следующих функциональных возможностей:

- добавление новых разделов каталога;
- изменение параметра существующих разделов:
- редактирование наименований каталогов;
- загрузка картинки в каталог;

– «возможность очищения раздела (удалять содержимое разделов);

– возможность удаления раздела (только для пустых или очищенных разделов);

– добавление в каталог новых товаров;

– изменение параметра существующих товаров;

– изменение цены;

– активность товара;

– изменение конфигураций продукта;

– удаление товара»[18].

Данный компонент осуществляет проверку оговоренного функционала за 12 секунд и при завершении тестирования без обнаруженных ошибок, гарантирует исправность данной функциональной части системы.

Компонент является уникальным так как тестируемые функциональный возможности не повторяются. При разработке данного компонента использовались следующие команды (таблица 2).

Таблица 2 - Используемые команды Selenium при разработке компонентов

Функция	**Описание функции**
verifyTitle/assertTitle	«проверяет соответствие заголовка страницы ожидаемому»[18].
verifyTextPresent	«проверяет наличие ожидаемого текста где-либо на странице»[12].
verifyElementPresent	«проверяет страницу на наличие ожидаемого элемента по его HTML-тегу»[13].
verifyText	«проверяет наличие на странице ожидаемого текста и соответствующего ему HTML-тега»[14].
verifyTable	«проверяет таблицу на наличие ожидаемого содержимого»[15].
waitForPageToLoad	«временно прекращает выполнение теста до загрузки ожидаемой страницы. Автоматически вызывается при использовании команды “clickAndWait”»[16]
waitForElementPresent	«приостанавливает выполнение до появления ожидаемого элемента интерфейса пользователя с определенным HTML-тегом»[17]

Компонент, осуществляющий автоматизированное тестирование функциональной возможности работы с заказами, а именно при работе разработанного компонента проверяется:

- возможность просмотра содержимого заказа;
- изменение состояния заказа;
- отмена заказа;
- удаление заказа;
- просмотр даты и время заказа;

Для повышения надежности функциональной части, были разработаны и внедрены SQL запросы для проверки корректной работы web-приложения с БД Mag.

Данный компонент был разработан на языке Selenium IDE и внедренными в него SQL запросами позволяющие сравнить выводимую и хранимую информацию в БД Mag и web-странице корзина покупателя. Главным элементом компонента является модуль получения отображаемой информации со страницы интернет-магазина, адаптированный под каждый пункт поставленной задачи:

```
{
Assert.AreEqual(css=span.minicart-title.hand
driver.FindElement(By.CssSelector("a.product-title")).Text);
        Assert.AreExpor(cssq=zakaz);
}
```

Принцип работы компонента заключается в распараллеливании запросов и сравнении полученных результатов в модуле “Анализ результатов”. На каждом этапе проверки Selenium загружает очередную команду из разработанного компонента и передает её в браузер на выполнение, браузер отправляет запрос в БД на предоставление запрашиваемой информации. Получив ответ, браузер согласно скриптам выводит информацию на web-странице. В этот момент Selenium загружает следующую команду из компонента, а именно Assert.AreEqual на передачу

отображенной (полученной из БД) информации в модуль “Анализ результата”, параллельно этому процессу Selenium исполняет команду на осуществление “прямого” запроса в БД разработанного согласно предоставленной документации о структуре БД. Получит результат наступает фаза сравнения полученных результатов, результата отображенной информации от веб-страницы и результат от “прямого” SQL запроса в БД интернет-магазина.

Структурная схема данного компонента представлена на рисунке 13

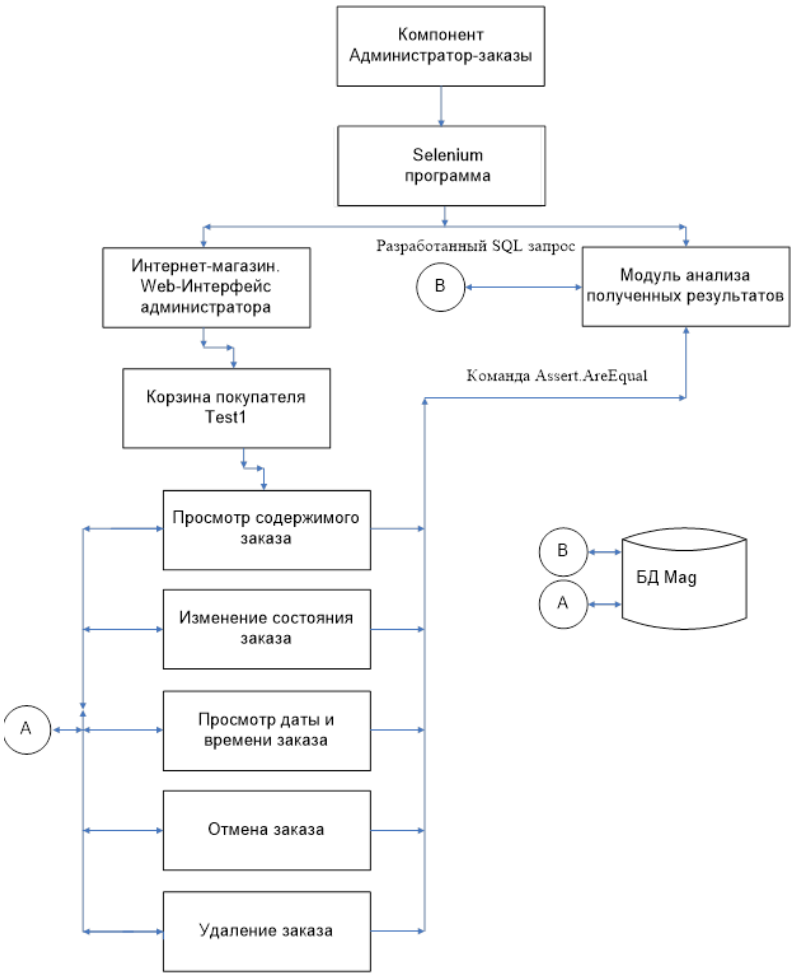


Рисунок 13 – Структурная схема работы компонента «Администратор-заказы»

Данный интернет-магазин осуществляет создание «двойника» (абсолютная копия сайта) для внедрения и тестирования новых модулей, позволяющих ускорить процесс работы самого ресурса и преобразить

интерфейс магазина для привлекательности для покупателей. В связи с этим был разработан отдельный компонент позволяющий произвести проверку перенесенной информации.

Компонент проверки текстовой информации:

– информация страницы «Гарантия и сервис»;
– информация страницы «Бесплатная доставка»;
– информация страницы «Адрес и телефон»;

И проверка остальных страниц. Разработка данного компонента могла занять больше времени в связи с занесением информации, расположенной на странице, был разработан компонент позволяющий сравнивать код «старой» и «новой». Для этого было произведено инспектирование элементов обоих сайтов и по тегам, расположенным в коде страниц, осуществлялось сравнение информации (рисунок 14).

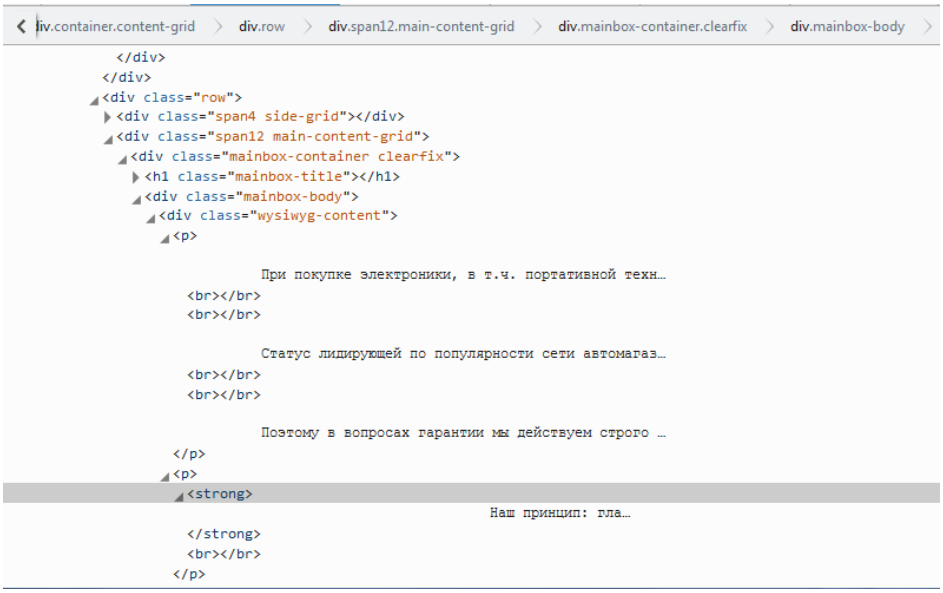


Рисунок 14 –Инспектирование элементов сайта

Принцип работы данного компонента прост для понимания. Selenium выполняет последовательно все команды заложенные в компоненте. В компонент было внедрена команда благодаря которой, при возникновении

расхождения в тексте между страницами и сайтами, компонент не прекращал свое выполнение, а обращался к модулю формирования отчета занеся информацию о странице в которой было найдено расхождение и информации о несоответствии

Данное условие выглядит следующим образом:

{driver.Navigate().GoToUrl(baseURL + "/service.html");

notstopForPageToLoadnext(By.LinkTextError).

driver.FindElement(By.LinkText(baseURL + "/service.html)).Click(); }

Модуль формирования отчета по проверке страниц является так же разработанным. За основу взят код спецификации формирования отчетов от QC контроля качеством ПО и доработан для большей информативности программиста. Добавлена необходимая информация о страницах и характере ошибок (рисунок 15).

Рисунок 15 –Образец отчета компонента проверки перенесенной информации

На рисунке 16 отображена схема работы компонента проверки текста.

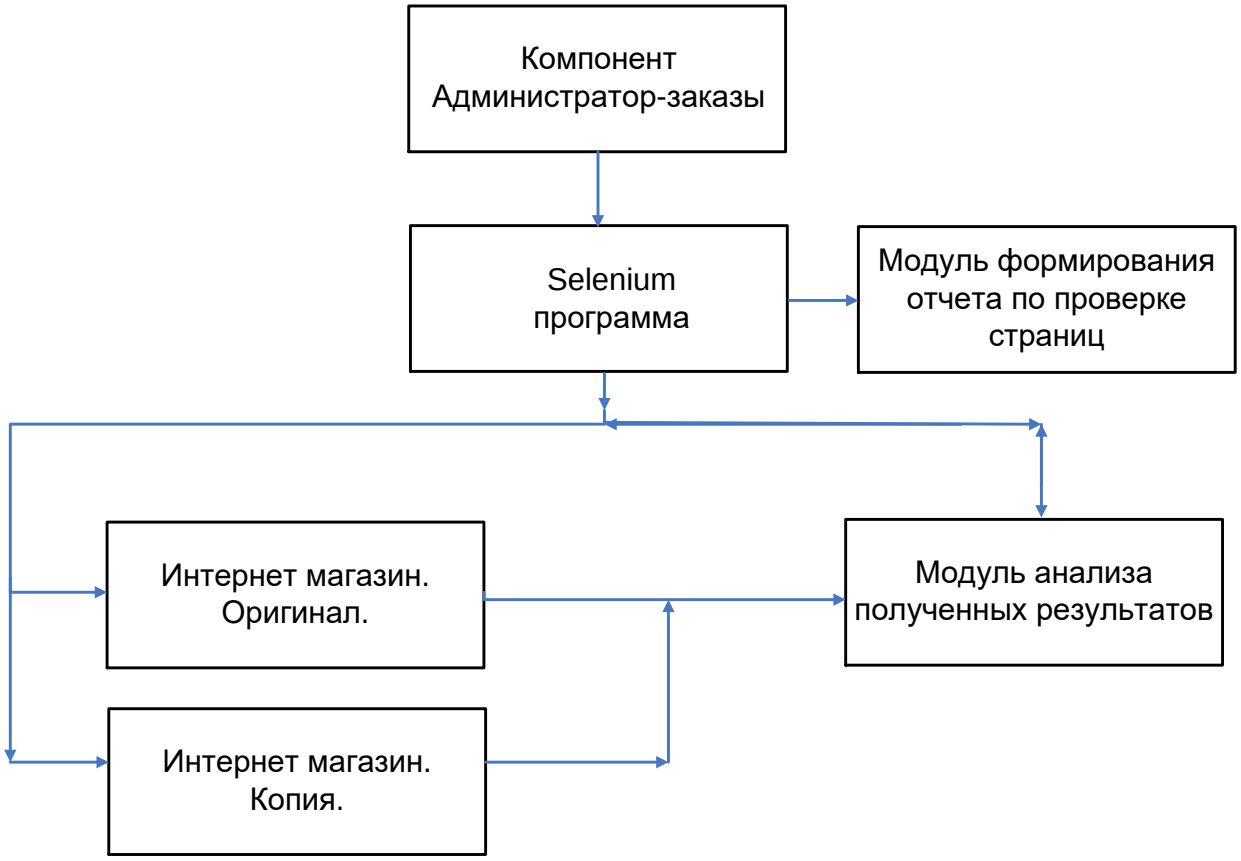


Рисунок 16 –Структурная схема работы компонента проверки текста

Следующей функциональной возможностью интернет-магазина и пользователя администратора, является возможность подгружать товар из прайса листа, а также менять цену по загрузке Exsel Для этого были разработаны соответствующие компоненты. Принцип которых заключался в простом сравнении загруженного Exsel файла и находящихся БД информации о товаре. В разработанном компоненте не используется модуль формирования отчета, в связи с внедренным SQL запросом, производящим замену ошибочной информации в БД на актуальную из предоставленного файла. Актуальность данного компонента заключается в том, что приведенный сайт при повторном импортировании файла не проверяет существует такой товар или нет. В любом случае информация о товаре попадает в БД под новым ключевым значением. Данный компонент исключает возможность выполнения такой ситуации (рисунок 17).

Компонент обеспечивает возможность не только проверки, но и замены не корректной информации о товаре на информацию из файла, а именно изменять параметры существующих товаров:

– наименование, цену;

– краткое и подробное описания;

– "активность товара";

– динамически отображать на главной странице такие категории;

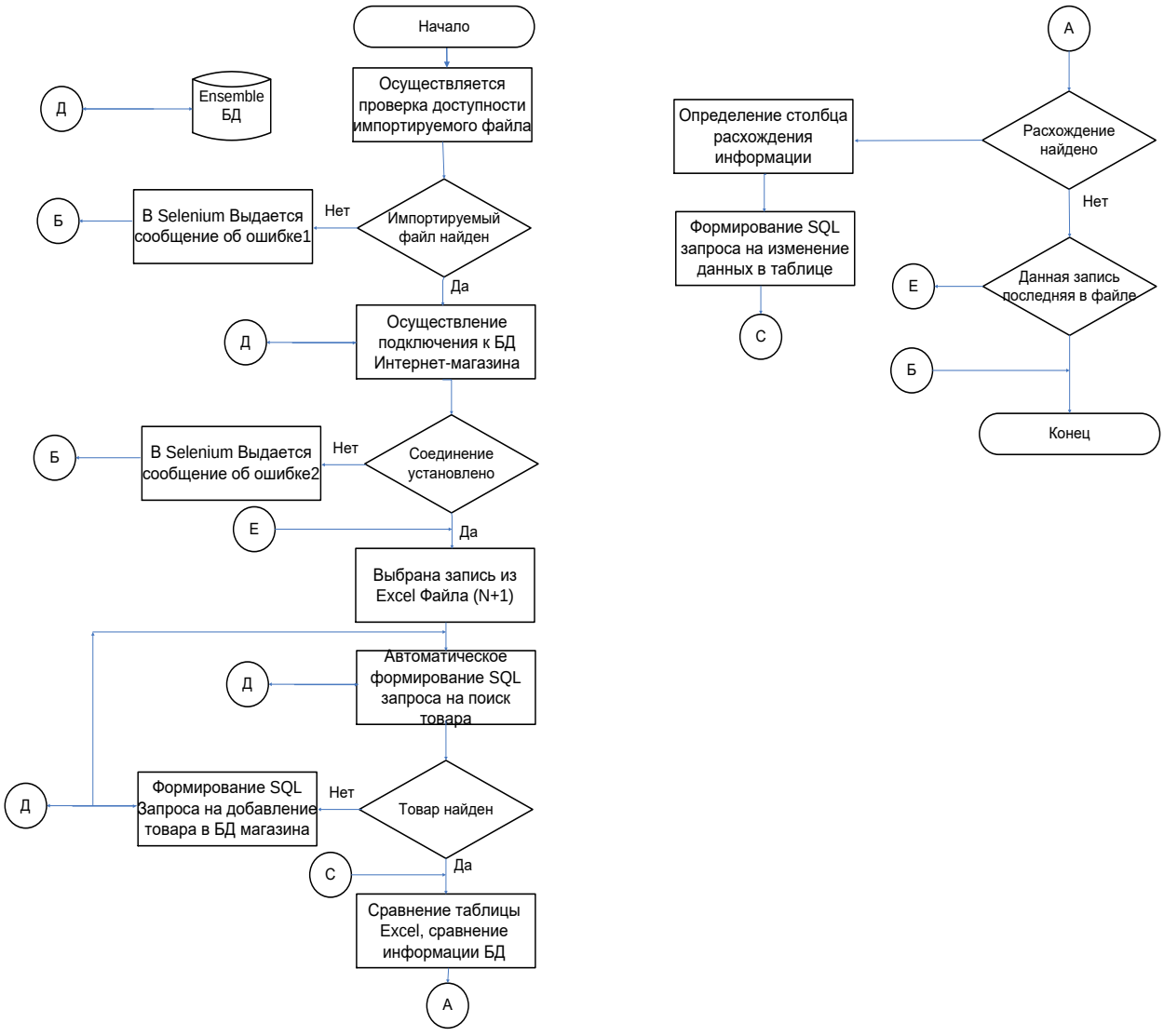


Рисунок 17 – Алгоритм работы компонента «Проверка импортируемого Excel файла и записей в БД»

Ниже приведен фрагмент кода компонента устанавливающего связь с БД интернет магазина.

```
DataTable.Import «\\bell-bac\MEGAFON E-
COMMERCE_conf\config_1.xls»
Dim connect, connectionString, MP, MDP,Login,Password
connectionString = DataTable ("connectionString", dtGlobalSheet)
Set connect = CreateObject("ADODB.Connection")
For Each ADOErr In connect.Errors
Debug.Print ADOErr.Number
Debug.Print ADOErr.Description
connect.Open connectionString
Set rs = CreateObject("ADODB.Recordset")
```

Данный фрагмент кода подтверждает надежность подключения к БД и самое главное безопасность.

Выводы по главе 2

Во второй главе были рассмотрены вопросы логического проектирования системы по автоматизации тестирования для программного обеспечения.

На функциональной схеме выделены пользователи системы и основные функции. На диаграмме последовательности определен порядок ввода и вывода данных.

Сформулированы требования к функциональности, надежности и удобстве использования разрабатываемой системы по автоматизации тестирования для программного обеспечения.

Глава 3 Оценка эффективности системы автоматизации тестирования для программного обеспечения

3.1 Тестирование и отладка приложения

Процесс тестирования разделен на две части. Вначале были написаны тесты для тестирования интернет-магазина, и как раз для их хранения, удобного поиска ошибок и отслеживания всех проведенных тестов, была спроектирована и создана система для хранения и навигации по разработанным тестам, т.е. система для автоматизации тестирования.

О разработанных тестах описано ниже, так как они представляют более глубокое понимания процесса тестирования и тестирования функционального – тестирования черного ящика.

В разработанной системе хранится тесты, с помощью которых тестировали интернет-магазин.

Процесс тестирования собственной системы представляет собой тестирование белого ящика и это тестирование описано в таблицах 3-6. В базу данных функциональные тесты, с помощью которых тестировали интернет-магазин будут загружены в соответствующие таблицы базы данных.

Разработаем тест кейсы для тестирования разработанной системы. Начнем проверку с базы данных и загрузки информацию в соответствующие таблицы. Результаты представлены в таблицах 3-6

Таблица 3 - Тест-кейс 1

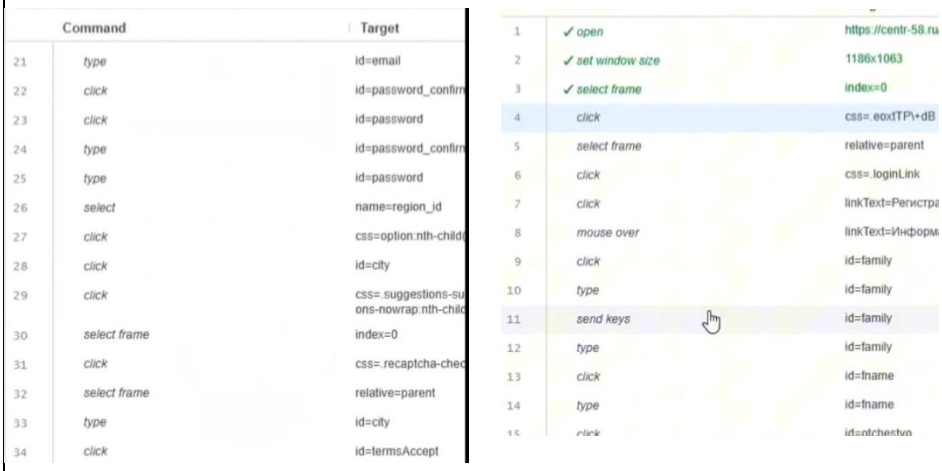
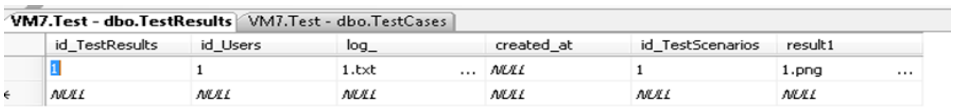
Пункт теста	Содержание пункта тест-кейса
№ теста	1
Входная информация	Автотест формы регистрации, содержащийся в файле 1.txt и результат теста в 1.png (рисунок 18)  Рисунок 18 – Входные данные Тест-кейса1
Процедура выполнения	1. В поле log прикрепляем файл 1.txt 2. В поле result – прикрепляем файл `result`.
Ожидаемый результат	Заполненная строка в таблице `TestResults`
Полученный результат	Полученный результат показан на рисунке 19.  Рисунок 19 – Результат Тест-кейса1

Таблица 4 - Тест-кейс 2

Пункт теста	Содержание пункта тест-кейса
№ теста	2
Идея теста	Проверить ввод данных пользователей, которые проводят тестирование
Входная информация	Данные содержат ФИО и электронную почту, а также роль, с которой работает в системе пользователь.
Процедура выполнения	1. В поле `name` - вводим Иван 2. В поле `email` – вводим ivan@mail.ru 3. В поле `role` - вводим Tester 4. В поле `created_at` - вводим `2025-05-01`

Продолжение таблицы 4

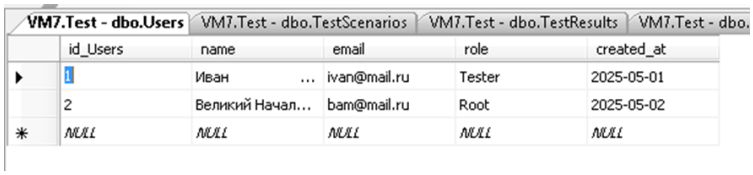
Пункт теста	Содержание пункта тест-кейса
Ожидаемый результат	Заполненная строка в таблице `Users`
Полученный результат	Полученный результат показан на рисунке 20.  Рисунок 20 –Результат Тест-кейса2

Таблица 5 - Тест-кейс 3

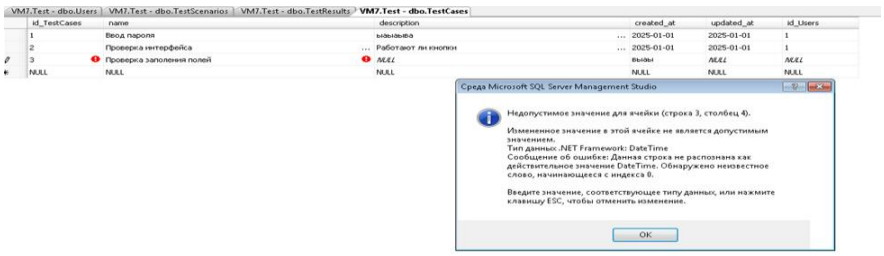
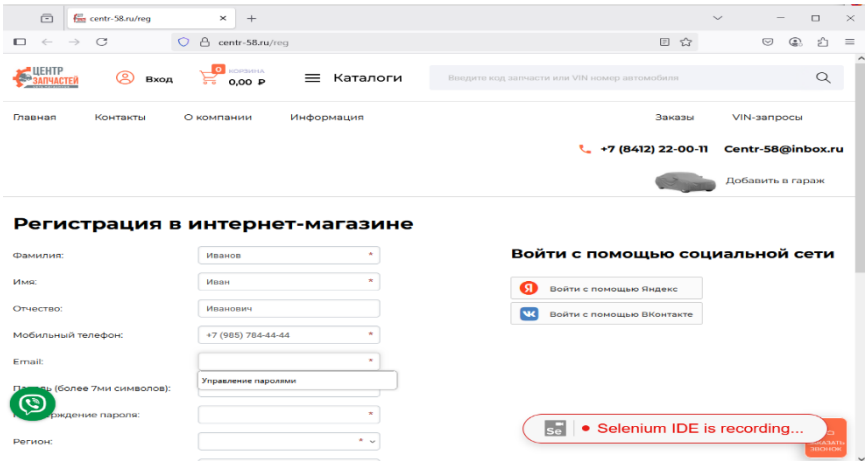
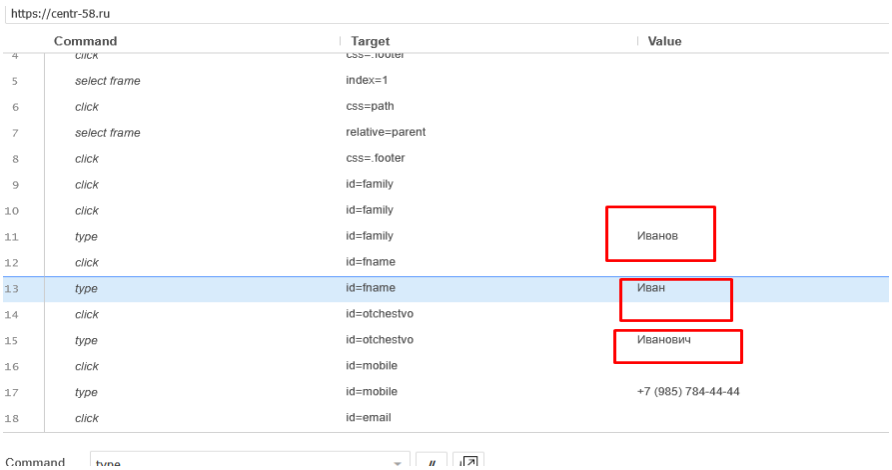
Пункт теста	Содержание пункта тест-кейса
№ теста	3
Идея теста	Негативный тест. Проверка заполнения неверными данными
Входная информация	Данные даты содержат некорректные значения.
Процедура выполнения	В поле `created_at -` вводим `символы`
Ожидаемый результат	Сообщение СУБД об ошибке
Полученный результат	Полученный результат показан на рисунке 21.  Рисунок 21 – Результат Тест-кейса3

Таблица 6- Тест-кейс 4

Пункт теста	Содержание пункта тест-кейса
№ теста	4
Идея теста	Создание автотеста для теста 1, Проверка регистрации. Внесение данных
Входная информация	Заполняем поля регистрационной формы. Все поля имеют возможность заполнения данными. Данные занесенные в форму отображаются в автотесте
Процедура выполнения	1. В поле ФИО – вносим Иванов 2. В поле имя – Иван 3. В поле Отчество Иванович
Ожидаемый результат	Ожидаемый результат показан на рисунке 22.  Рисунок 22 –Результат Тест-кейса2
Полученный результат	Полученный результат показан на рисунке 23.  Рисунок 23 – Результат Тест-кейса4

Далее будут приведены примеры разработанных компонентов для тестирования Интернет-магазина с ролью пользователь.

Основополагающей функциональной возможностью клиента, зашедшего на сайт, является его регистрация на данном интернет-ресурсе, в случае если пользователь уже зарегистрирован, то выполнение авторизации в приложении. Основное внимание при тестировании регистрации и авторизации обращалось на поле «Логин» и «Пароль» в данные поля заносилась различная информация как «положительная», так и «негативная»:

Компонент проверки регистрации и авторизации. Преимуществом данного компонента является скорость его выполнения, оба действия удается протестировать за 12 секунд. Перебрав все возможные комбинации.

Данный компонент разработан таким образом, чтобы не только обеспечить проверку регистрации нового пользователя в БД и его авторизацию на сайте, но и проверит корректность выводимых ошибок при до пускания ошибок в регистрации и авторизации. Ниже приведен текст программы, разработанный на языке Selenium. Данный фрагмент кода проверяет отображение диалоговых окон при введении не верных данных при авторизации:

```
driver.FindElement(By.LinkText("Войти")).Click();
driver.FindElement(By.Id("login_popup3")).SendKeys("тест");
driver.FindElement(By.Id("psw_popup3")).SendKeys("тест");
driver.FindElement(By.CssSelector("div.buttons-container.buttons-container-picker > div.body-bc.clearfix > div.float-right > span.button-submit.button-wrap-left > span.button-submit.button-wrap-right > input[name=\"dispatch[auth.login]\"]")).Click(); try
{Assert.AreEqual("E-Mail адрес в поле E-mail неправильный.", driver.FindElement(By.CssSelector("#login_popup3_error_message > p")).Text);
}
{
```

```
Assert.AreEqual("Поле Пароль обязательное.",
driver.FindElement(By.CssSelector("#psw_popup3_error_message > p")).Text);
 }
```

При выполнении данного компонента был выявлен дефект в авторизации. При выводе сообщения об ошибке была ссылка на таблицу в БД, что не допустимо для любого интернет-ресурса.

Следующим компонентом позволяющим проверить качество интернет – магазина, является компонент оповещения пользователя.

При разработке данного компонента заказчик поставил следующие цели:

– проверить возможность пользователю получать информацию о новых товарах и возможностях магазина;

– проверить работоспособность поиска необходимого товара;

– проверить возможность добавления товара в корзину;

– проверить возможность удаления товара из корзины;

– проверить возможность изменить количество того или иного товара в покупательской корзине;

– проверить возможность пересчитать стоимость товара в корзины;

– проверить возможность оформления заказа;

– проверить получения подтверждения заказа по электронной почте.

И множество других требований, для которых необходимо разработать и реализовать компоненты.

Наиболее важным разрабатываемым компонентом для тестирования приложения со стороны пользователя является компонент по работе с корзиной (рисунок 24).

Рисунок 24 – Структурная схема компонента проверка оформления заказа

Основа данного компонента заложена на проверку работы пользователя с корзиной. Добавление, удаление товара, оформление товара и главной функциональной частью является отправка письма на электронный адрес покупателя. Для тестирования данного функционала был разработан соответствующий компонент. Действия которого описаны ниже:

```
driver.FindElement(By.LinkText("Все для автозвука")).Click();
driver.FindElement(By.LinkText("FM модуляторы")).Click();
driver.FindElement(By.LinkText("FM - модулятор - UBN-22")).Click();
try
driver.FindElement(By.LinkText("Пересчитать")).Click();
driver.FindElement(By.Id("button_cart")).Click();
try
{
```

Assert.AreEqual("Ваша корзина пуста", driver.FindElement(By.CssSelector("p.no-items")).Text);

}

Фрагмент приведенного кода описывает действия компонента по добавлению товара в корзину, проверки что именно данный товар добавлен в список покупок пользователя. В данном компоненте не внедрен SQL запрос в БД так как на момент проверки данным компонентом подтверждено корректная работа с БД магазина, но присутствует проверка по названию товара. Для покрытия области тестирования и возможности применения компонента в любой стадии разработки.

Компонент выполняет следующие функции:

– добавление товара в корзину;

– проверка добавленного товара;

– добавление еще одного товара;

– проверка функциональной возможности сайта пересчитать сумму покупки;

– удаление товаров из корзины;

– проверка корректности сообщения об отсутствие товара в корзине покупателя;

– добавление товара в корзину;

– осуществление оформления покупки;

– проверка занесения записи о покупке в БД в соответствующую таблицу;

– проверка почтового ящика тестируемого пользователя для определения получения сообщения о покупке пользователем.

Данный компонент на своё выполнение затратил 21 секунду. При тестировании был выявлен дефект в ПО, в БД заносилась запись о покупке не с тем пользователем, что приводила к неверной работе функциональной части системы Интернет-магазина. Проверка данный компонентом осуществляется

каждую неделю для проверки правильности работы Интернет магазина, так как данная разработка охватывает большую часть функционального взаимодействия между пользователем и системой.

Следующим разработанным компонентом является тестирование возможностей пользователя по работе с учетной записью, данный компонент производит проверку следующих функций:

– работа с обязательными полями к заполнению;

– корректность выводимого сообщения при не заполнении обязательного поля;

– изменение данных веденных в необязательные поля;

– корректность выводимой информации о пользователи на web-странице;

– уведомление пользователя о внесение изменений в учетную запись, уведомление представляет собой электронное письмо высылаемое на email указанный в учетной записи;

– осуществление загрузки фотографий;

– проверка отображения изменённой информации в БД системы.

Для реализации данного компонента использовался язык Selenium с внедренными в него SQL запросами на подобие используемых в предыдущих компонентах.

При тестировании интернет-магазина своевременно выявлено 21 дефект и 12 улучшений, что позволило в кратчайшие сроки наладить и запустить в работу интернет-магазин. Регрессионное тестирование осуществляется по разработанным компонентам.

Схему взаимодействия моделей системы Selenium можно представить следующим видом (рисунок 25).

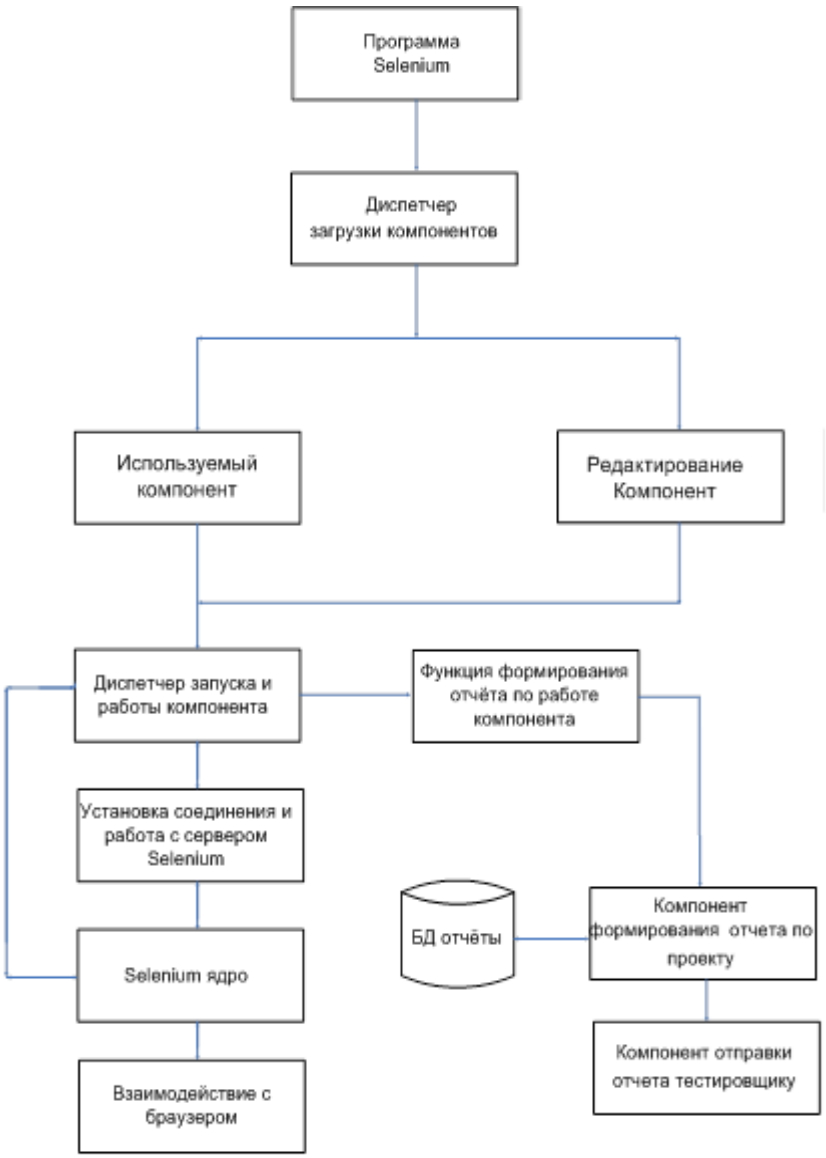


Рисунок 25 – Схема взаимодействия модулей системы Selenium

Для осуществления загрузки компонента в программу Selenium необходимо выполнить ряд действий: открыть вкладку «File», нажать клавишу «Open» (рисунок 26).

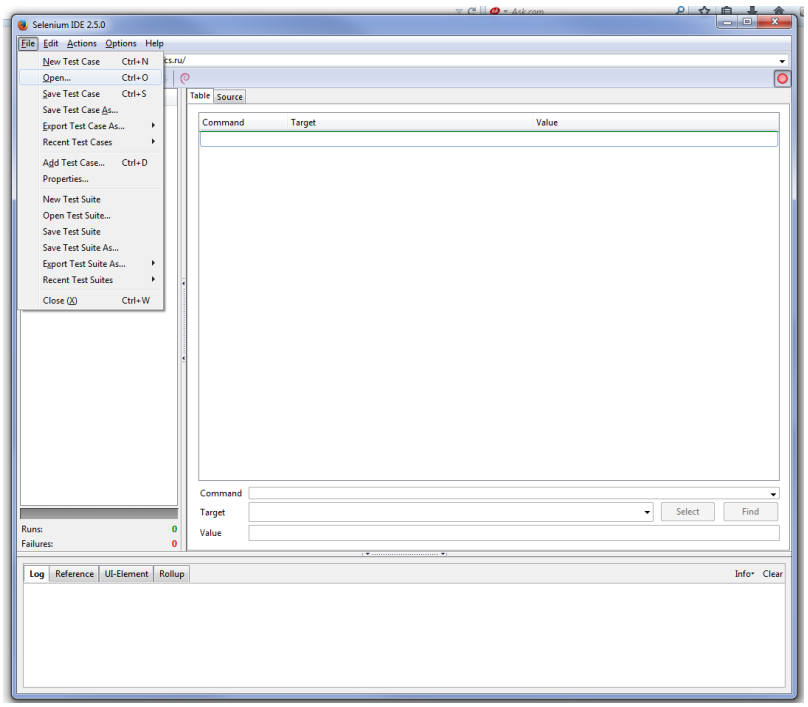


Рисунок 26 – Открытие окна Select a File

Выбрать необходимый для тестирования компонент или набор компонентов, нажать клавишу «Открыть» (рисунок 27).

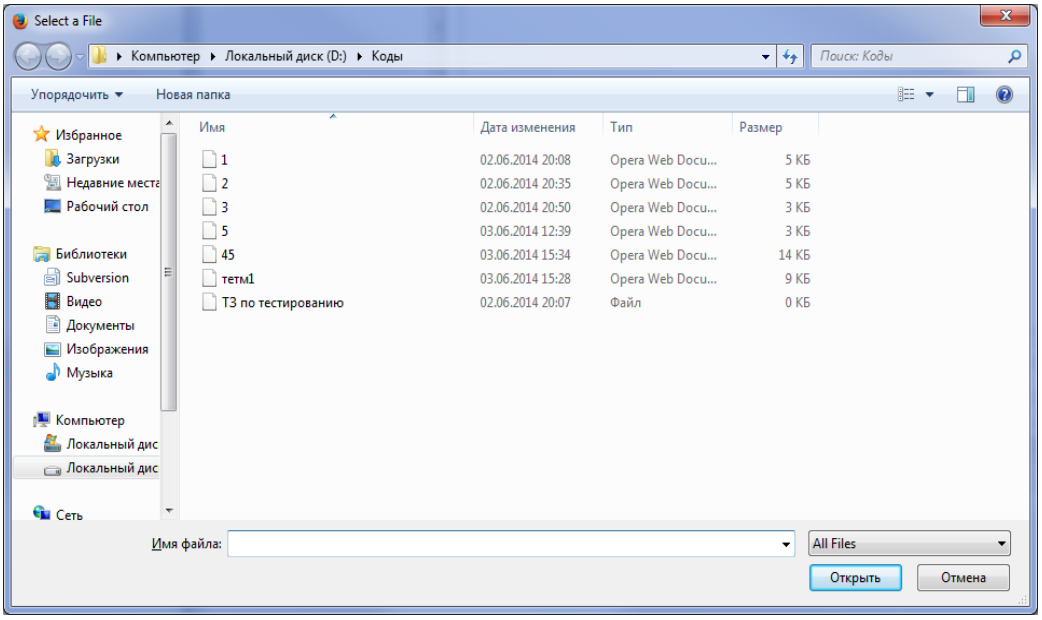


Рисунок 27 –Окно Select a File

3.2 Оценка удобства и функциональности приложения

В ходе выполнения работы проведены следующие виды тестирования:

По объекту тестирования:

– функциональное тестирование (functional testing);

– тестирование интерфейса пользователя (ui testing).

По степени автоматизации:

– автоматизированное тестирование (automated testing).

По признаку позитивности сценариев:

– позитивное тестирование (positive testing);

– негативное тестирование (negative testing).

По степени подготовленности к тестированию:

– тестирование по документации (formal testing).

Процесс работы компонента при прохождении одного из разработанных тест-кейсов.

При записи теста Selenium записывает описания объектов интерфейса тестируемого приложения, с которыми тест взаимодействует, в Object Repository. Свойства Selenium записываемые для каждого типа объектов определяется настройками Object Idenfication (рисунок 28).

Command	Target
open	/
clickAndWait	xpath=(//a[contains(text(),'Личный кабинет')])[2]
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
type	id=LOGIN
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
type	id=passwdId
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
type	id=codeId
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=submitBtnId
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=codeId
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=menu_item_SUBS_CHARGES
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=menu_item_REPORT_REORDER_FORM
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=SRRD_ID_7571059
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=radio-email
selectWindow	name=;P_USER_LANG_ID=1;loginURL=https://volgasg.megafon.ru/?_ga=1.234987587.33076121.1399883292&SECONDARY_LOGIN=1&P_USER_
click	id=idInputSrdt0

Рисунок 28 – Окно идентификации графических объектов

У созданных скриптов две задачи одна — это проведение всех необходимых операций для проверки работоспособности приложения, т.е. выполнение исполняющих команд по сайту, а второе это вывод сообщения о том в каком месте приложения произошла ошибка при тестировании.

Для запуска компонента в SELENIUM необходимо провести действия, описанные ниже:

– выбрать необходимый компонент из таблицы TestCase;

– перейти на вкладку Table и если это необходимо внести поправки в программный код компонента;

– указать скорость работы компонента, с помощью вкладки «Fast-Slow»;

– нажать клавишу «Play current test case» (рисунок 29).

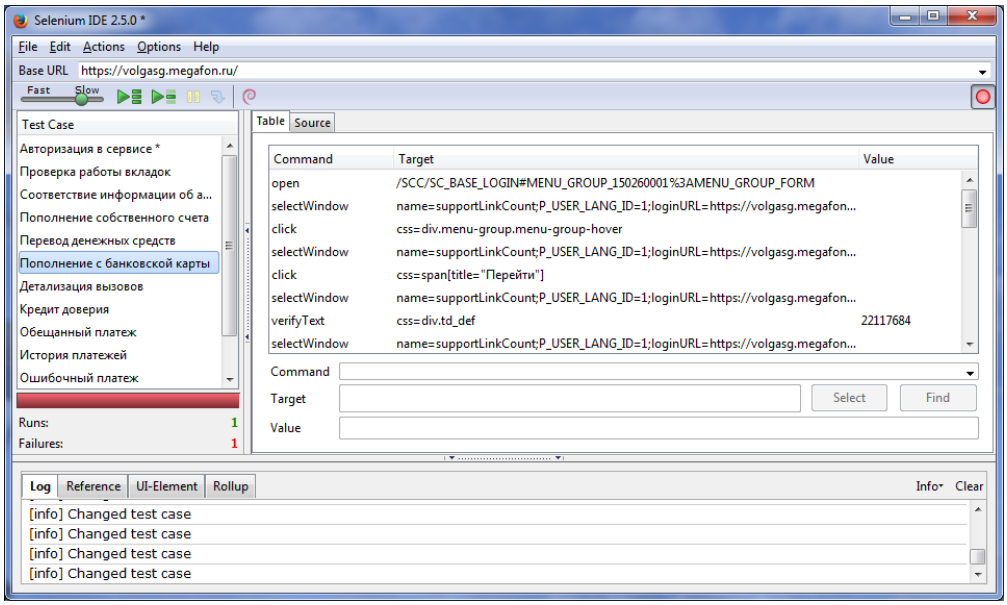


Рисунок 29 – Внешний вид Selenium IDE

Selenium открывает браузер и начинает передавать заложенные в компонент команды. В браузере можно видеть, что Selenium выполняет каждый шаг, который был сделан во время записи теста; выполняемая команда выделяется цветом (рисунок 30).

	Command	Target	Value
1	✓ open	/reg	
2	✓ set window size	2576x1416	
3	click	linkText=закрыть	
4	click	css=.footer	
5	select frame	index=1	
6	click	css=path	
7	select frame	relative=parent	
8	click	css=.footer	
9	click	id=family	
10	click	id=family	
11	type	id=family	Иванов
12	click	id=fname	
13	click	css=.fr-flex-col-lg-5	
14	type	id=fname	Иван
15	click	id=otchestvo	

Рисунок 30 –Выполнение автоматизированного теста

3.3 Анализ результатов использования программного обеспечения

Когда Selenium завершает выполнение теста, он отображает всю информацию о завершении тестирования (рисунок 31).

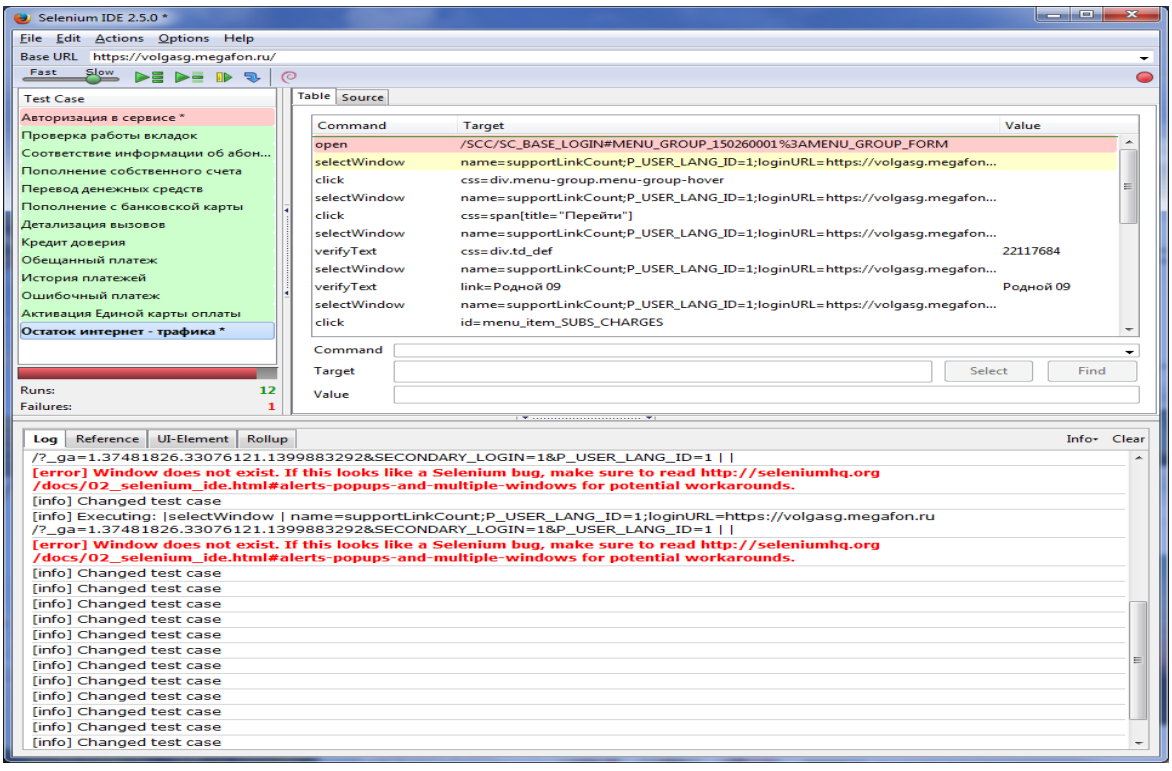


Рисунок 31 –Результат выполнения компонентов

Selenium зеленым цветом выделяются TestCase выполненные успешно, красным цветом отображаются TestCase выполнение которых было не завершено в связи с дефектами в проверяемой программе. Так же Selenium отображает числовое значение успешных и не успешно выполненных проверок и приводит Log файл с выполнением каждого компонента, с помощью которых программист с легкостью сможет определить причину ошибочного завершения выполнения компонента.

А также выделит красным цветом во вкладке Table строку, на которой произошел данный инцидент.

Выполнение теста считается удачным, если Selenium смог воспроизвести на сайте все записанные действия. В данном случае при тестировании была обнаружена одна ошибка на первом шаге теста (рисунок 32).

Рисунок 32 – Вывод сообщения об ошибке при тестировании

В Selenium так же предусмотрена возможность прохождения каждого шага TestCase тестировщиком, то есть тестировщик может кликнуть необходимую команду из компонента и Selenium запустит её на выполнение, но только ту, которую указал тестировщик.

Ниже приведена часть кода выполняющаяся при работе компонента (рисунок 33).

Command	Target	Value
open	/SCC/SC_BASE_LOGIN#MENU_GROUP_150260001%3AMENU_GROUP_FORM	
click	css=div.menu-group.menu-group-hover	
click	id=menu_item_SUBS_CHARGES	
verifyText	css=div.td_def	22117684
click	css=div.but.next	
click	css=div.but.next	
click	css=div.but.next	
click	id=menu_item_REPORT_REORDER_FORM	
click	id=SRRD_ID_11862463	
click	id=radio-email	
click	id=idInputSrdt0	
click	id=idInputSrdt0	
click	id=idInputSrdt0	
verifyText	css=div.td_def	22117684
click	css=div.menu-group.menu-group-hover	
click	id=menu_item_MONEY_TRANSFER_FORM	
type	id=idMsisdnTo	9374013017
type	id=idP_AMOUNT	12
open	/SCC/SC_BASE_LOGIN#MONEY_TRANSFER_FORM%3AMONEY_TRANSFER_FORM	
click	css=input.button_forward.button_forward-hover	

Рисунок 33 – Часть кода компонента «Перевод средств»

При первоначальном тестировании с использованием компонента, было выявлено 4 дефекта. Время, затраченное на выполнение компонента составляет 12 секунд. И может быть протестировано не ограниченное количество раз, что позволяет тестировщику данного проекта, не тратить время на регрессионное тестирование старого функционала, а сосредоточить свое внимание на интеграционном тестировании и тестировании новых модулей.

Выводы по главе 3

В третьей главе были рассмотрены вопросы практической реализации системы автоматизации тестирования для программного обеспечения.

Заключение

В ходе выполнения исследования была подтверждена актуальность задачи автоматизации тестирования программного обеспечения. Проведен комплексный анализ существующих аналогов, выявлены их преимущества и ограничения. На основании полученных результатов разработано собственное решение поставленной проблемы.

В результате сформирован комплекс информационного обеспечения, ориентированный на автоматизацию процессов тестирования ПО, а также разработан и проведен тестирование программного модуля. Определены виды тестирования, подлежащие автоматизации, включая функциональное тестирование для проверки соответствия системы заданным требованиям и регрессионное тестирование, эффективность которого была повышена за счет использования автоматизированных средств.

Разработана модель системы тестирования и сформирован набор ручных тестовых сценариев, обеспечивающих проверку соответствия программного модуля техническому заданию и корректности его функционирования. Внедрение автоматизации способствовало увеличению эффективности тестирования в среднем в два раза, ускорению проверки соответствия требованиям, сокращению сроков тестирования и снижению количества ошибок, характерных для ручной проверки.

Перспективы развития системы автоматизации тестирования связаны с автоматической генерацией тестов на основе анализа исходного кода и пользовательского поведения, расширением автоматизации на всех этапах жизненного цикла ПО и внедрением автоматического анализа покрытия тестами различных функциональных областей. Реализация указанных направлений позволит существенно повысить эффективность тестирования, сократить время вывода продукта на рынок и улучшить качество программного обеспечения.

Список используемой литературы и используемых источников

1. Аниче, М. Эффективное тестирование программного обеспечения : практическое руководство / М. Аниче ; пер. с англ. А. Н. Киселева. - Москва : ДМК Пресс, 2023. - 370 с. - ISBN 978-5-97060-997-2.
2. Дэвид А.Марка, Клемент МакГоуэн Методология структурного анализа и проектирования SADT - McGraw-Hill Companies – 2019.
3. Карпович, Е. Е. Жизненный цикл программного обеспечения : лабораторный практикум / Е. Е. Карпович. - Москва : Изд. Дом МИСиС, 2016. - 130 с. - Текст : электронный. - URL: https://znanium.com/catalog/product/1232186 (дата обращения: 24.01.2025).
4. Конструирование программного обеспечения : учебное пособие / под ред. Л.Г. Гагариной. — Москва : ИНФРА-М, 2024. — 319 с. — (Высшее образование). — DOI 10.12737/1893880. - ISBN 978-5-16-017861-5
5. Котляров, В. П. Основы тестирования программного обеспечения : краткий курс / В. П. Котляров. - Москва : ИНТУИТ, 2022. - 252 с. - ISBN 5-9556-0027-2.
6. Мкртычев С.В., Гущина О.М., Очеповский А.В. Прикладная информатика. Бакалаврская работа [Электронный ресурс] : электрон. учеб-метод. пособие. Тольятти. ТГУ: Изд-во ТГУ, 2019. URL: https://dspace.tltsu.ru/handle/123456789/8868 (дата обращения: 09.01.2025).
7. Морозова, Ю. В. Тестирование программного обеспечения : учебное пособие / Ю. В. Морозова. - Томск : Эль-Контент, 2019. - 120 с. - ISBN 978-5-4332-0279-5
8. Похилько, А.Ф. CASE-технология моделирования процессов с использованием средств BPWin и ERWin: учебное пособие / А.Ф. Похилько, И.В. Горбачев. - Ульяновск: УлГТУ, 2016. - 120 с.
9. Проектирование современных баз данных: Учебно-методическое пособие / Дадян Э.Г. - М.:НИЦ ИНФРА-М, 2017. - 120 с.
10. Свод знаний по управлению бизнес-процессами. BPM CBOK 3.0:

Учебное пособие / Под ред. Белайчук А.А. - М.:Альпина Пабл., 2016. - 480 с

11. Синицын, С. В. Верификация программного обеспечения : краткий учебный курс / С. В. Синицын, Н. Ю. Налютин. - Москва : ИНТУИТ, 2023. - 320 с. - ISBN 978-5-94774-825-3.

12. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие / Г.Н. Федорова. — Москва : КУРС : ИНФРА-М, 2024. — 336 с. — (Среднее профессиональное образование). - ISBN 978-5-906818-41-6.

13. Черников, Б. В. Оценка качества программного обеспечения. Практикум : учебное пособие / Б. В. Черников, Б. Е. Поклонов ; под ред. Б. В. Черникова. — Москва : ФОРУМ : ИНФРА-М, 2022. — 400 с. : ил. — (Высшее образование). - ISBN 978-5-8199-0516-6

14. BpWin [Электронный ресурс] URL: http://habrahabr.ru/.(дата обращения: 09.01.2025)

15. Fewster, M., & Graham, D. (1999). Software Test Automation. Addison-Wesley.

16. Kaner, C., Falk, J., & Nguyen, H. Q. (1999). Testing Computer Software (2nd ed.). Wiley.

17. Meszaros, G. (2007). xUnit Test Patterns: Refactoring Test Code. Addison-Wesley.

18. Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing (3rd ed.). John Wiley & Sons.

19. Visio 2010: руководство для начинающих [Электронный ресурс]. URL: support.office.com (дата обращения: 09.01.2025)

20. Whittaker, J. A. (2009). Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design. Addison-Wesley.