

Аннотация

Тема выпускной квалификационной работы «Разработка модуля автоматизированного тестирования для системы CRM (Customer Relationship Management)».

Целью выпускной квалификационной работы является разработка модуля для автоматизированного регрессионного тестирования в CRM-системе в ООО «Холмонт БР».

Пояснительная записка состоит из введения, четырех глав и заключения. Во введении раскрывается общее состояние проблемы, описание предмета и объекта проекта, обосновывается актуальность темы, представлена новизна проекта, определяются цель и задачи проекта, практическое применение результатов проекта. В первой главе приведены технико-экономическая характеристика деятельности предприятия и характеристика проекта, произведен анализ текущего состояния тестирования на проекте и выявлена и обоснована необходимость разработки. Далее произведено концептуальное моделирование информационной системы, сформулированы требования к проектируемому модулю автоматизированного тестирования. Вторая глава содержит описание обоснования проектных решений по выбору технологий логического моделирования и описание логического проектирования информационной системы. Третья глава содержит описание обоснования проектных решений по выбору технического, программного и технологического обеспечения, а также непосредственную реализацию проектных решений, описанных в первой главе. Четвертая глава содержит результаты тестирования разработанного модуля для автоматизированного тестирования. В заключении содержатся результаты, полученные в ходе выполнения исследования основных положений бакалаврской работы.

В работе представлено 3 таблицы, содержит 41 рисунок, список использованной литературы содержит 36 источников. Общий объем выпускной квалификационной работы составляет 65 страницы.

Abstract

The theme of the graduation work is "Development of an automated testing module for the CRM-system (Customer Relationship Management)".

The aim is the development of a module for automated regression testing in a CRM system in Haulmont LLC.

Structurally the work includes an introduction, four chapters and a conclusion.

The introduction reveals the general state of the problem, a description of the subject and object of the project, substantiates the relevance of the topic, presents the novelty of the project, defines the goal and objectives of the project, and the practical application of the results of the project.

In the first chapter, the technical and economic characteristics of the enterprise's activities and the characteristics of the project are given, the current state of testing on the project is analyzed and the need for development is identified. The conceptual modeling of the information system was made, the requirements for the designed automated testing module were formulated.

The second chapter contains a description of the choice of logical modeling technologies and a description of the logical design of an information system.

The third chapter contains a description of design decisions on the choice of hardware, software and technological support, and an implementation of the design decisions described in the first chapter.

The fourth chapter contains the results of testing the developed module for automated testing.

The conclusion presents the results and conclusions on the work performed.

The work presents 3 tables, 41 figures, the list of references contains 36 sources. The total amount of the graduation work is 63 pages

Оглавление

Введение.....	6
Глава 1 Функциональное моделирование модуля автоматизированного тестирования для системы CRM.....	9
1.1 Технико-экономическая характеристика предприятия.....	9
1.2 Концептуальное моделирование информационной системы.....	14
1.3 Постановка задачи на разработку проекта создания модуля для автоматизированного тестирования.....	17
1.4 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»	18
Глава 2 Логическое проектирование модуля автоматизированного тестирования для системы CRM.....	21
2.1 Выбор технологий логического проектирования модуля автоматизированного тестирования для системы CRM	21
2.2 Разработка логической модели проектируемого модуля	23
2.3 Разработка архитектуры проектируемого модуля автоматизированного тестирования.....	27
2.4 Требования к аппаратно-программному обеспечению информационной системы	28
Глава 3 Физическое проектирование модуля автоматизированного тестирования для системы CRM.....	30
3.1 Выбор технологии разработки модуля автоматизированного тестирования.....	30
3.2 Выбор среды разработки модуля автоматизированного тестирования.....	35
3.3 Разработка модуля автоматизированного тестирования в системе CRM.....	37
Глава 4 Тестирование разработанного модуля автоматизированного тестирования для системы CRM.....	53
4.1 Подготовка к тестированию.....	53

4.2 Прохождение выделенных кейсов	54
Заключение	61
Список используемой литературы	63

Введение

Программное обеспечение (ПО) в современном мире используется повсеместно, практически во всех сферах жизни, в его разработку вкладывается огромное количество ресурсов, как финансовых, временных, так и человеческих. Применение автоматизированных информационных систем облегчает процессы ведения бизнеса, образования, здравоохранения, управления и т.д. Ввиду столь высокой востребованности ПО задачи снижения стоимости его разработки и улучшения качества являются одними из наиболее приоритетных в сфере информационных технологий.

Для повышения качества выпускаемой продукции компании-разработчики осуществляют тестирование разработанной продукции [3]. Автоматизация тестирования позволяет снижать расходы, экономить время и людские ресурсы, понизить риск выпуска некачественного продукта на рынок. В связи с этим постоянно появляются новые технологии автоматизации тестирования, набирают все большую популярность, обрастают обширным комьюнити среди разработчиков [18]. Этим определяется актуальность темы, выбранной для бакалаврской работы.

На сегодня известно, что оптимизации производства и повышения качества продукции уже недостаточно, особенно в сфере услуг, так как отношения с клиентами играет такую же большую роль, как и качество продукции. Из этого следует, что для развития компании крайне важна продуктивная работа с клиентами, чтобы удержать и собрать новых клиентов.

Одна из главных ролей в компаниях отведена менеджерам по продажам. Именно они осуществляют взаимодействие с клиентами. Чтобы упростить работу менеджеров по продажам, существует CRM система.

В процессах ведения бизнеса используются CRM-системы (сокращение от англ. Customer Relationship Management) – прикладное ПО для автоматизации взаимодействия с заказчиками (клиентами), позволяющее:

- сохранять информацию о клиентах и истории взаимоотношений с ним;
- устанавливать и улучшать бизнес-процессы;
- анализировать результаты;
- улучшить обслуживание клиентов и т.д. [26].

Также CRM-система может обладать любым необходимым функционалом, таким как взаимодействие с внешними системами (интеграция с системой 1С, мессенджерами и т.д.), или проведение тестирования базового функционала веб-приложения (сайта) организации для проверки его доступности и работоспособности. На данный момент для того, чтобы найти возможную ошибку в работе сайта, необходимо проводить периодический мониторинг страниц вручную, а также проверять наличие необходимых бэкапов в удаленных хранилищах и на сервере. Это занимает значительное время – около двух часов в день, которые могут быть использованы более рационально при применении автоматизации.

Разработка ориентирована для внутреннего использования автоматизации работы QA Engineer в ООО «Холмонт БР».

Объектом исследования является автоматизированное тестирование функционала веб-приложения в ООО «Холмонт БР».

Предметом исследования является клиент-серверное приложение.

Цель бакалаврской работы — разработка модуля для автоматизированного регрессионного тестирования в CRM-системе в ООО «Холмонт БР».

Задачи исследования — конкретно сформулированные отдельные этапы работы. Для достижения поставленной цели выпускной квалификационной работы нужно решить следующие задачи:

- произвести анализ предметной области;
- разработать функциональную и логическую модели проектируемого модуля автоматизированного тестирования;
- сформулировать требования к клиент-серверному приложению;

– реализовать и протестировать модуль.

Результатом бакалаврской работы будет готовый к использованию (разработанный и протестированный) модуль для автоматизированного тестирования в CRM-системе.

Пояснительная записка состоит из введения, четырех глав и заключения.

В первой главе приведены технико-экономическая характеристика деятельности предприятия и характеристика проекта, произведен анализ текущего состояния тестирования на проекте и выявлена и обоснована необходимость разработки. Далее произведено концептуальное моделирование информационной системы, сформулированы требования к проектируемому модулю автоматизированного тестирования.

Вторая глава содержит описание обоснования проектных решений по выбору технологий логического моделирования и описание логического проектирования информационной системы.

Третья глава содержит описание обоснование проектных решений по выбору технического, программного и технологического обеспечения, а также непосредственную реализацию проектных решений, описанных в первой главе.

Четвертая глава содержит результаты тестирования разработанного модуля для автоматизированного тестирования.

В заключении содержатся результаты, полученные в ходе выполнения исследования основных положений бакалаврской работы.

Работа изложена на 63 страницах, содержит 41 рисунок и 3 таблицы.

Глава 1 Функциональное моделирование модуля автоматизированного тестирования для системы CRM

1.1 Техничко-экономическая характеристика предприятия

Компания «HAULMONT» работает на рынке разработки корпоративного программного обеспечения с 2008 года. Компания развивается во множестве направлений:

- разработка популярных отраслевых продуктов;
- разработка open-source фреймворков;
- разработка комплексных решений - проектов для отдельных заказчиков и т.д.

На текущий момент в компании реализовано множество проектов, среди которых можно выделить следующие:

- система электронной документации (СЭД) «Тезис» - готовое решение для автоматизации документооборота, канцелярии и управления поручениями, в то же время являющееся расширяемой BPM платформой для автоматизации произвольных бизнес-процессов предприятия;
- open-source фреймворк CUBA Platform, ставший основой для следующего проекта Jmix Platform - высокоуровневая Java-платформа с открытым кодом для создания корпоративных информационных систем;
- комплексное решение для служб такси «Sherlock Taxi» - система, которая автоматизирует все аспекты работы служб такси, от приема заказа и построения маршрута до выставления счета и оплаты труда водителя;
- облачная CRM-система «Параплан» - CRM для учебных центров, которая создана с учетом специфики работы учебных центров, детских и спортивных клубов и содержит все необходимое в одном приложении;
- плагин для среды разработки IntelliJ Idea JPA Buddy – инструмент для разработки проектов с JPA и т.д.

Компания разделена на несколько бизнес-юнитов (отделов), во главе которых стоит отдельный руководитель. По мере необходимости и работы над очередным проектом отделы вовлекаются в процесс и сотрудничают.

Организационную структуру компании в упрощенном виде можно представить на рисунке 1.

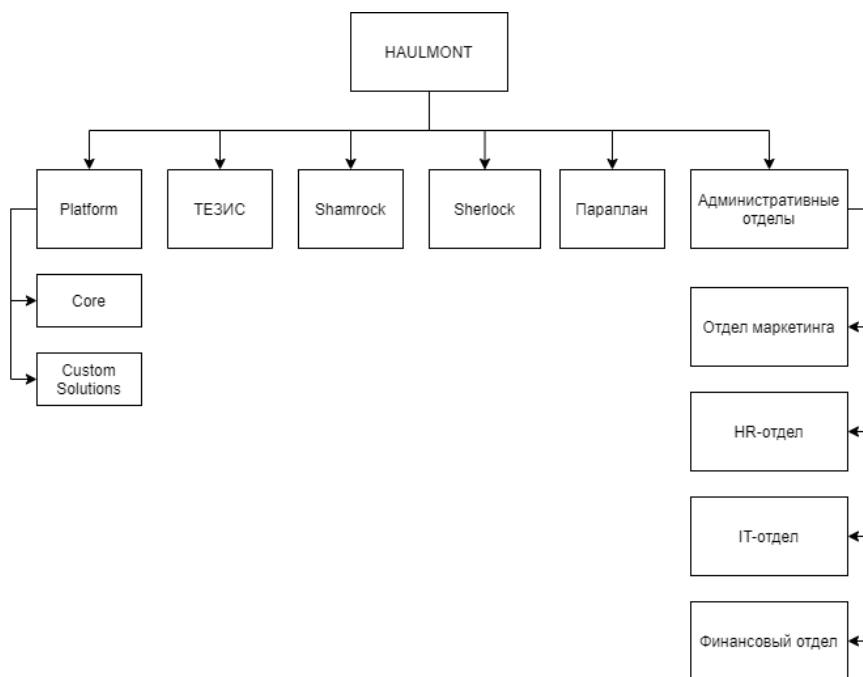


Рисунок 1 – Организационная структура «HAULMONT»

Для облегчения управления в каждом из отделов присутствует система иерархии менеджмента для разделения обязанностей во избежание перенагруженности сотрудников.

Разработка модуля ведется в рамках проекта из отдела Custom Solutions для интернета-магазина.

1.1.1 Характеристики модуля CRM в проекте

CRM-система, используемая на проекте, обладает обширным функционалом, таким как:

- обработка и ведение заказов клиентов;

- обработка заявок клиентов;
- обработка бронирований клиентов
- работа с каталогом товаров;
- работа с характеристиками товаров, их брендами и моделями;
- формирование отчетности и выгрузка по запросу оператора в форматы Excel и CSV;
- формирование графиков рассылок писем клиентам;
- формирование и рассылка сообщений на подразделения предприятия и т.д.

Скриншот меню CRM-системы представлен на рисунке 2.

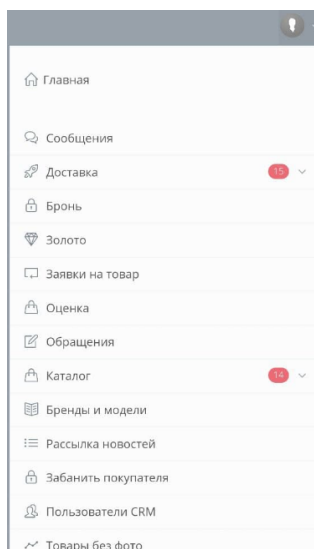


Рисунок 2 – Скриншот меню CRM-системы

В данной работе будет рассматриваться внедрение модуля автоматизированного тестирования в CRM-систему отдельного проекта. В дальнейшем наработки можно будет интегрировать и в другие проекты.

1.1.2 Характеристики тестирования на проекте

Тестирование является важной частью процесса разработки программных продуктов и входит в число наиболее эффективных способов

обеспечения их качества. Причем под качеством в сфере разработки программных средств подразумевается не только надежность программы или удобство пользования. Тестирование – это одна из наиболее трудоемких фаз разработки программного обеспечения, занимающая большое количество времени. При этом цена исправления ошибки, как правило, напрямую зависит от времени, которое проходит с момента ее возникновения до момента обнаружения.

На практике тестирование осуществляется путем выполнения определенного набора действий в тестируемом приложении, получении результатов выполнения этих действий и дальнейшей сверки их с данными, которые определены как эталонные.

При добавлении нового функционала в существующую систему важным пунктом является проверка на отсутствие конфликтов и блокирования существующих функциональностей внесенными изменениями. Данный вид тестирования называется регрессионным (англ. regression testing, от лат. regressio — движение назад) [13].

Регрессионное тестирование включает в себя несколько подвидов:

- new bug-fix — проверка исправления вновь найденного дефекта;
- old bug-fix — проверка, что исправленный ранее и верифицированный дефект не воспроизводится в системе снова;
- side-effect — проверка того, что не нарушилась работоспособность работающей ранее функциональности, если её код мог быть затронут при исправлении некоторых дефектов в другой функциональности.

Обычно используемые методы регрессионного тестирования включают повторные прогоны предыдущих тестов, а также проверки, не попали ли регрессионные ошибки в очередную версию в результате слияния кода.

Выполняться этот процесс может как вручную, специалистами по тестированию, так и автоматически, с использованием различных программных средств. Именно такой процесс верификации программного обеспечения, в ходе которого основные функции и шаги теста, такие как

запуск, инициализация, выполнение, анализ и выдача результата, выполняются автоматически при помощи инструментов для автоматизированного тестирования, и называется автоматизированным тестированием программного обеспечения [18].

У автоматизированного тестирования есть как свои преимущества, так и недостатки. К сильным сторонам автоматизированного тестирования относят:

- быстрая скорость выполнения, намного превосходящая возможности человека;
- отсутствие влияние «человеческого фактора» (невнимательность, усталость);
- возможность многократного выполнения тестов и снижение затрат через это;
- выполнение тест-кейсов особенной сложности, недоступных человеку;
- способность хранить, анализировать в удобной форме колоссальные объемы данных;
- способность выполнять низкоуровневые действия с приложением, с операционной системой и т.д.

К недостаткам автоматизированного тестирования относятся:

- необходимость привлечения высококвалифицированного персонала для автоматизации взамен возможности использовать низкоквалифицированный труд тестировщиков;
- затраты на средства автоматизации, на разработку и сопровождение тестов;
- финансовые затраты и риски, связанные с наличием большого количества средств автоматизации и сложностью выбора;
- устаревание тестов в случаях изменений требований, переработки интерфейсов тестируемых продуктов и т.п.

Автоматизация тестирования не позволяет полностью исключить ручное тестирование из процесса разработки, но значительно снижает его

долю, помогает убрать рутину из процесса тестирования, снизить стоимость и значительно улучшить качество разрабатываемых программных продуктов [18].

Для интернет-магазина очень важным является непрерывное функционирование сайта для минимизации потерь прибыли. При внедрении нового функционала/рефакторинге существующего производится ручное тестирование, но порой оно может быть неполным, что приводит к ошибкам отображения/функциональным ошибкам, которые затрудняют работу с веб-приложением. Также причиной ошибок могут быть невыполненная команда для сборки фронтенд-модуля при сборке продакшена, неверно указанный путь к файлу стилей, просроченный SSL-сертификат, неверно указанный селектор у элемента на странице и т.д.

Основополагающая цель QA Engineer – минимизация ошибок в разрабатываемом продукте и обеспечение наилучшего качества при наименьших затратах времени и ручного труда. В связи с этим можно выделить ряд задач:

- выделение базового функционала продукта и его компонентов, которые могут проверяться автоматически;
- создание инструмента для автоматизированного тестирования функционала;
- интеграция инструмента с внешними мессенджерами для оповещения в случае нахождения ошибок.

Можно сделать вывод, что необходимость разработки обусловлена целью специалиста.

1.2 Концептуальное моделирование информационной системы

Разработка и анализ модели бизнес-процесса «КАК ЕСТЬ»

Модель AS-IS - «КАК ЕСТЬ», модель существующего процесса или функции. При построении функциональной модели фиксируются процессы и

объекты, которые осуществляются на предприятии. Обследование бизнес-процессов является неотъемлемой составляющей при разработке проекта или развития системы [23].

Для моделирования была выбран процесс регрессионного тестирования выбранного компонента сайта. В модели «КАК ЕСТЬ» представлено то, как на данный момент происходит процесс тестирования заданного заранее компонента. Тестировщик получает данные для теста, проходит по пунктам специального чек-листа для тестирования, отмечая, пройден или не пройден тест и в итоге получает результат проверки, который в дальнейшем анализируется.

Для моделирования бизнес-процесса «КАК ЕСТЬ» была выбрана диаграмма вариантов использования методологии UML, описывающая взаимоотношения и зависимости между группами вариантов использования и действующими лицами, участвующими в процессе. Действующие лица могут быть как реальными людьми (например, пользователями системы), так и другими компьютерными системами или внешними событиями [22].

Модель «КАК ЕСТЬ» представлена на рисунке 3.

В данной системе выделено 2 актера: «Тестировщик» – сотрудник, производящий тестирование компонента, и «Веб-приложение» - само веб-приложение, отображающее страницы по запросу пользователя.

Сотрудник «Тестировщик» выполняет тестирование необходимого компонента посредством варианта использования «Тестирование компонента сайта». Данный вариант использования включает в себя несколько вариантов использования:

- при необходимости тестировщик обновляет чек-лист, используемый при тестировании, посредством варианта использования «Обновление чек-листа тестирования»;
- тестировщик проходит действия, указанные в чек-листе, посредством варианта использования «Прохождение чек-листа». Данный

вариант использования включает в себя вариант использования «Выполнение требуемого действия на веб-странице»;

- тестировщик заполняет чек-лист по мере получения позитивного/негативного результата выполнения пунктов чек-листа, посредством варианта использования «Заполнение чек-листа»;

- тестировщик анализирует результаты тестирования посредством варианта использования «Анализ результатов».

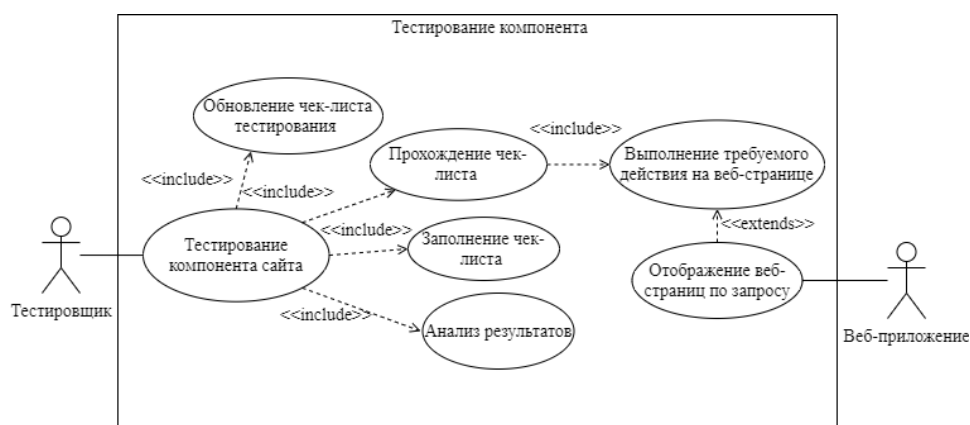


Рисунок 3 – Модель «КАК ЕСТЬ»

«Веб-приложение» отображает запрашиваемые веб-страницы посредством варианта использования «Отображение веб-страниц по запросу». Данный вариант использования наследуется от варианта использования «Выполнение требуемого действия на веб-странице», таким образом происходит взаимодействие тестировщика и веб-приложения.

Таким образом, в данном пункте было произведено моделирование бизнес-процесса «Тестирование компонента» на текущий момент, и после анализа были выделены следующие недостатки:

- необходимость вручную проходить заданные сценарии, что занимает довольно много времени, является рутинным и однообразным;

– увеличение числа тестируемых компонентов прямо пропорционально увеличивает количество затрачиваемого времени на тестирование.

Для оптимизации рабочих процессов для подобных рутинных задач является целесообразным применение автоматизации. Для управления автоматизированным тестированием может быть создан модуль в CRM-системе, требования к которой будут поставлены в следующем пункте.

1.3 Постановка задачи на разработку проекта создания модуля для автоматизированного тестирования

Проектируемый модуль в CRM-системе, разрабатываемый для ООО «Холмонт БР», будет использоваться для автоматизированного тестирования функционала сайта. Целью создания данной системы является необходимость удобного инструмента для обеспечения качественной проверки работы компонентов сайта. Необходимость автоматизации данного процесса обусловлена тем, что на ручное тестирование уходит много времени, сил и данный процесс является однообразным.

Основные требования к модулю для тестирования:

- разработка модуля в соответствии с заданной CRM-системой;
- понятный интерфейс для работы с тестами, возможность записи действий автоматизированного ПО путем записи видеофайла или создания скриншотов по ходу выполнения теста;
- интеграция модуля с внешними системами (мессенджерами) для уведомления пользователей о результатах проводимых тестов, возможность выбора канала отправления результатов;
- централизованное управление всеми имеющимися тестами, возможность редактирования и добавления новых тестов.

Таким образом, в данном пункте была описана цель создания системы, обусловлена необходимость разработки и выявлены основные требования к модулю.

1.4 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»

При построении модели «ТО-ВЕ» происходит внедрение модуля для автоматизированного тестирования в исходную CRM-систему. Благодаря использованию модуля тестировщик может не проводить все необходимые проверки вручную, а перепоручить их автоматизированному ПО, которое будет проходить по заданным сценариям и уведомлять о результатах проверки.

При построении модели «ТО-ВЕ» также была выбрана диаграмма вариантов использования методологии UML, описывающая взаимоотношения и зависимости между группами вариантов использования и действующими лицами, участвующими в процессе.

Диаграмма процесса «ТО-ВЕ» отображена на рисунке 4. В данной модели выделено 5 актеров: «Тестировщик» – сотрудник, производящий тестирование компонента, «Веб-приложение» - само веб-приложение, отображающее страницы по запросу пользователя, «CRM-система» - используемая на проекте CRM-система, «Драйвер» - драйвер для браузера, имитирующий действия пользователя и «Мессенджер» - внешняя система, с помощью которой будет производиться уведомление о результатах тестирования.

Сотрудник «Тестировщик» выполняет тестирование необходимого компонента посредством варианта использования «Тестирование компонента». Данный вариант использования включает в себя несколько вариантов использования:

- при необходимости тестировщик обновляет или добавляет новый тест на странице модуля в CRM-системе, посредством варианта использования «Обновление/добавление теста»;
- тестировщик запускает необходимый тест странице модуля в CRM-системе, посредством варианта использования «Запуск теста»;
- тестировщик анализирует результаты тестирования посредством варианта использования «Анализ результатов».

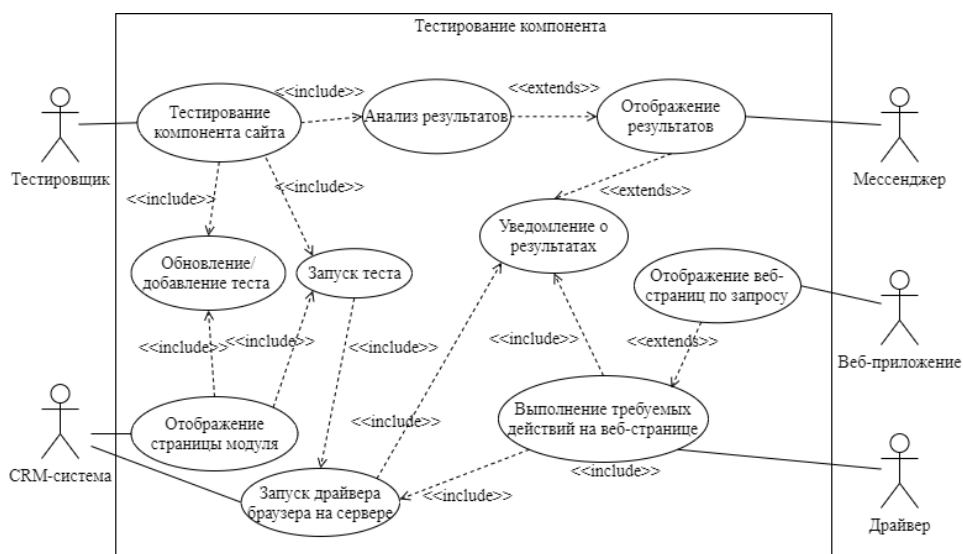


Рисунок 4 – Модель TO-VE

«CRM-система» предоставляет возможность управления тестами посредством варианта использования «Отображение страницы модуля». Данный вариант использования также включает в себя вышеописанные варианты использования «Обновление/добавление теста» и «Запуск теста». Вариант использования «Запуск теста» включает в себя запуск драйвера браузером на сервере посредством варианта использования «Запуск драйвера браузера на сервере». При ошибке запуска драйвера браузера отправляется уведомление в внешнюю систему (мессенджер) посредством варианта использования «Уведомление о результатах». «Драйвер» выполняет указанные в тесте действия посредством варианта использования

«Выполнение требуемого действия на веб-странице», включающего в себя вариант использования «Запуск драйвера браузера на сервере». После выполнения тестов отправляется уведомление с результатами выполнения в внешнюю систему (мессенджер) посредством варианта использования «Уведомление о результатах». «Веб-приложение» отображает запрашиваемые веб-страницы посредством варианта использования «Отображение веб-страниц по запросу». Данный вариант использования наследуется от варианта использования «Выполнение требуемого действия на веб-странице», таким образом происходит взаимодействие браузера, запускаемого на сервере, и веб-приложения. Внешняя система «Мессенджер» отображает результаты тестирования посредством варианта использования «Отображение результатов», который наследуется от варианта использования «Уведомление о результатах». Вариант использования «Анализ результатов» наследуется от варианта использования «Отображение результатов», т.к. тестирующий анализирует полученные результаты во внешней системе, таким образом происходит взаимодействие данных актеров.

Таким образом, автоматизированное тестирование с помощью модуля в CRM-системе повысит эффективность работы тестировщика в отслеживании возможных нарушений работы сайта, позволяя использовать время не на рутинные проверки, а на более сложные задачи.

Выводы по первой главе

Была выполнена технико-экономическая характеристика деятельности фирмы, выполнена разработка и анализ модели «КАК ЕСТЬ», на основе чего была выполнена разработка и анализ модели «КАК ДОЛЖНО БЫТЬ». Были поставлены и определены задачи и цели разработки, основываясь на которых, выявлена необходимость разработки модуля для автоматизированного тестирования в CRM-системе для ООО «Холмонт БР».

Глава 2 Логическое проектирование модуля автоматизированного тестирования для системы CRM

2.1 Выбор технологий логического проектирования модуля автоматизированного тестирования для системы CRM

При выборе технологии логического проектирования следует провести сравнительный анализ наиболее известных нотаций: IDEF0, UML, ARIS и учитывать следующие критерии сравнения:

- легкость в изучении;
- подход к проектированию;
- удобство по созданию моделей;
- возможность декомпозиции;
- актуальность;
- система хранения данных модели.

IDEF0 (Integration Definition for Function Modeling) – нотация, которая описывает и формализует бизнес-процессы, основанная на методологии и стандартах функционального моделирования. Ее отличительными особенностями являются акцент на соподчиненности объектов и логических отношениях между работами [2]. Данная нотация является довольно устаревшей и очень редко применяется в настоящее время.

UML (Unified Modeling Language) – унифицированный язык моделирования, который представляет собой графическую нотацию, предназначенную для моделирования и описания всех процессов, протекающих при разработке проекта. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью [4].

ARIS (Architecture of Integrated Information Systems) – методология, а также комплекс средств, который формализует информацию, анализирует и оптимизирует деятельность предприятия и представляет ее в виде

графических моделей. Отличительная особенность - любая организация в методологии ARIS рассматривается с пяти точек зрения: организационной, функциональной, обрабатываемых данных, структуры бизнес-процессов, продуктов и услуг [21].

Результаты сравнительного анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ методологий

Критерий сравнения	Нотация IDEF0	Нотация ARIS	Нотация UML
Легкость изучения	Легко	Очень сложно	Легко
Система хранения данных модели	Модели хранятся в файлах	Объектная база данных	Модели хранятся в файлах
Возможность декомпозиции	Неограниченная	Неограниченная	Неограниченная
Удобство по созданию моделей	Простая панель управления	Сложная панель управления	Простая панель управления
Актуальность	Устаревшая нотация	Редко применяется	Используется повсеместно
Подход к проектированию	Функциональный	Процессный	Объектно-ориентированный

После анализа можно сделать вывод, что предпочтительнее использовать методологию UML, так как она легка в изучении, имеет объектно-ориентированный подход к проектированию и удобна в создании моделей. Несмотря на простоту создания моделей в нотации IDEF0, данная нотация является устаревшей и почти не применяется на данный момент. Нотация ARIS имеет сложную панель управления и трудна для восприятия и изучения.

Таким образом, для проектирования была выбрана методология UML, что позволяет перейти к логическому проектированию модуля.

2.2 Разработка логической модели проектируемого модуля

Для разработки логической модели была использована диаграмма вариантов использования - модель бизнес-процесса «КАК ДОЛЖНО БЫТЬ», представленная на рисунке 4. Данная модель диаграммы описывает процесс работы системы с указанием всех ее действующих лиц и компонентов.

Основным действующим лицом, инициирующим процесс тестирования, является тестировщик, поэтому для понимания того, как должна быть построена архитектура системы, была произведена декомпозиция диаграммы вариантов использования с точки зрения тестировщика, представленная на рисунке 5.

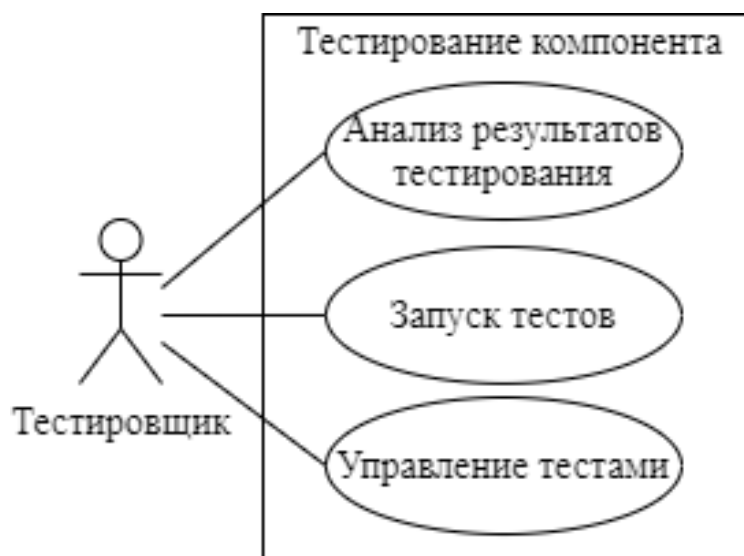


Рисунок 5 – Диаграммы вариантов использования Тестировщик

Наиболее точное описание данной диаграммы можно получить с помощью описания прецедентов диаграммы. Они представлены ниже в Таблице 2.

Таблица 2 - Описание прецедентов

Описание прецедента «Управление тестами»	
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Открыть страницу модуля в CRM. 2. Добавить тест в форме добавления. 3. Сохранить тест. 	<p>Альтернативный поток:</p> <p><i>A1 Если нужно отредактировать существующий тест</i></p> <ol style="list-style-type: none"> 1. Открыть страницу модуля в CRM. 2. Отредактировать нужный тест в форме редактирования. 3. Сохранить тест.
Описание прецедента «Запуск тестов»	
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Открыть страницу модуля в CRM. 2. Указать параметры запуска. 3. Выбрать тесты для запуска. 4. Запустить тесты. 5. Получить уведомление об успешном запуске. 	<p>Альтернативный поток:</p> <p><i>A1 Если неверно указаны параметры запуска</i></p> <ol style="list-style-type: none"> 1. Открыть страницу модуля в CRM. 2. Неверно указать параметры запуска/ не выбрать тесты для запуска. 3. Запустить тесты. 4. Получить уведомление об ошибке.
Описание прецедента «Анализ результатов тестирования»	
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Открыть внешнюю систему (мессенджер). 2. Просмотреть результаты выполненных тестов. 	<p>Альтернативный поток:</p> <p><i>A1 Если произошла ошибка запуска драйвера на сервере</i></p> <ol style="list-style-type: none"> 1. Открыть внешнюю систему (мессенджер). 2. Просмотреть сообщение с ошибкой.

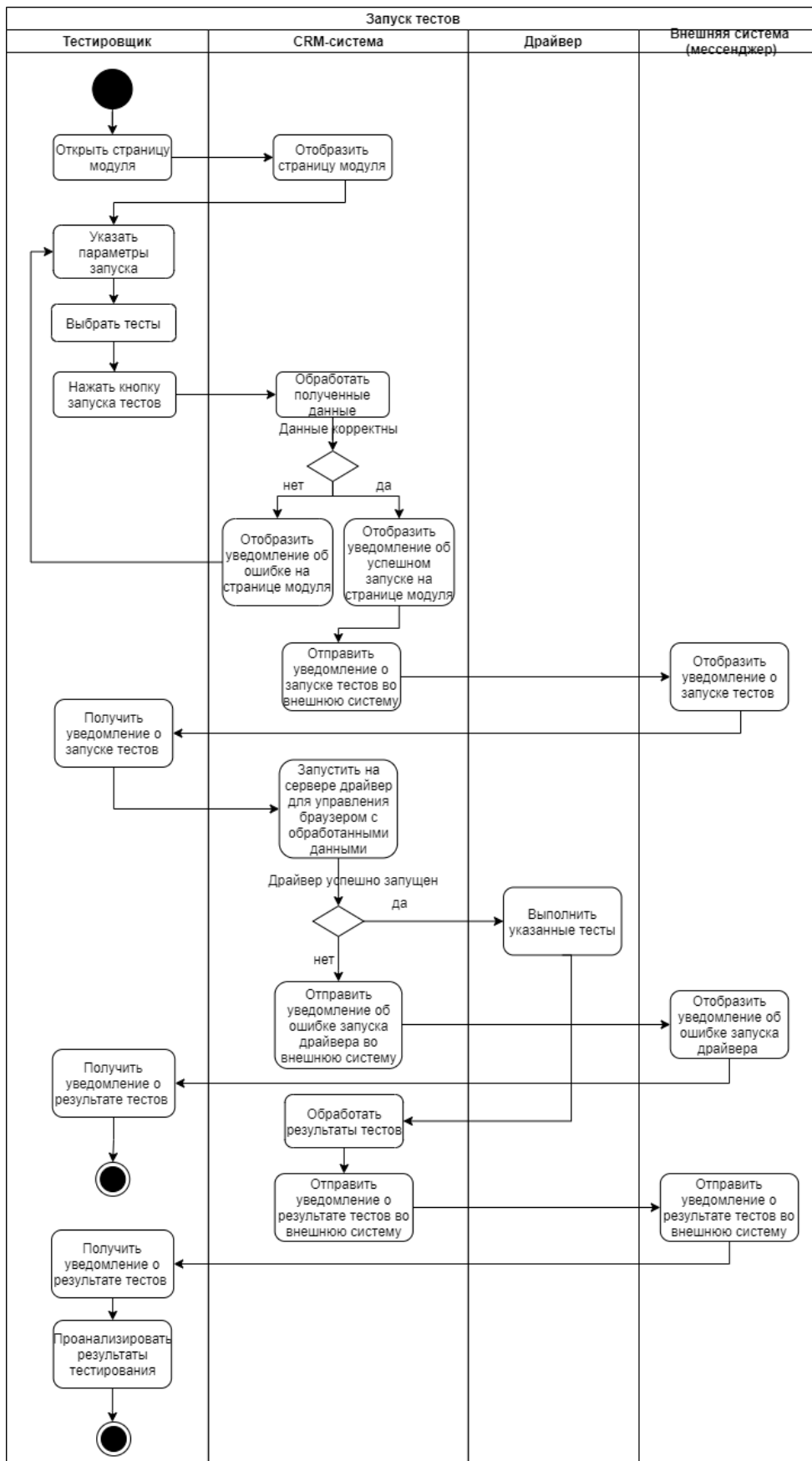


Рисунок 6 – Диаграмма деятельности процесса «Запуск тестов»

Таким образом, можно сделать вывод о функциональном назначении проектируемого модуля. Он будет использоваться для автоматизации процессов тестирования.

Для прецедента «Запуск тестов» была построена диаграмма деятельности, представленная на рисунке 6. Диаграммы деятельности создаются для отражения последовательности выполнения операций и представляют собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой [20].

С помощью диаграммы деятельности была проверена логика функционирования системы и описана последовательность работы в ходе процесса «Запуск тестов». Инициатором процесса является тестировщик, который напрямую взаимодействует с CRM-системой в ходе запуска тестов, а также взаимодействует с внешней системой при получении уведомлений в процессе тестирования.

CRM-система взаимодействует со всеми участниками процесса, обрабатывая данные, заданные пользователем, показывая уведомления и запуская на сервере драйвер для управления браузером, а также обрабатывая итоговые результаты тестирования.

Внешняя система (мессенджер) отображает уведомления, присылаемые ей в процессе тестирования, уведомления могут содержать описание результатов тестирования либо описание ошибки в случае ошибки запуска драйвера на сервере. Процесс заканчивается анализом пользователя полученных уведомлений.

Таким образом был смоделирован основной бизнес-процесс проектируемого модуля, что позволяет перейти к непосредственной разработке архитектуры модуля.

2.3 Разработка архитектуры проектируемого модуля автоматизированного тестирования

Для моделирования архитектуры модуля была выбрана и реализована диаграмма развертывания UML, позволяющая представить физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения.

Для реализации был выбран тот же процесс «Запуск тестов», так как при его исполнении задействованы все компоненты проектируемой системы. Данная диаграмма представлена на рисунке 7.

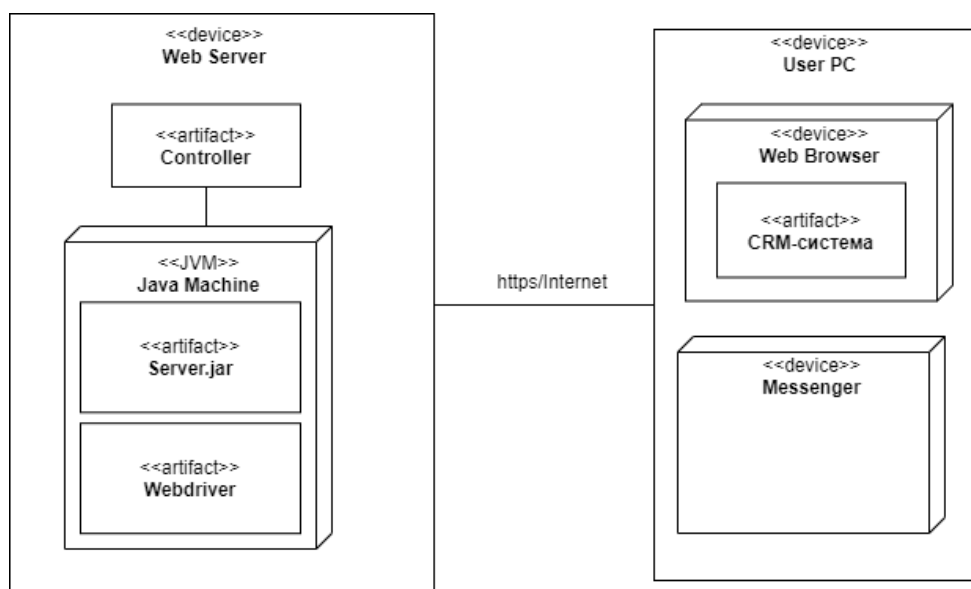


Рисунок 7 – Диаграмма развертывания

На данной диаграмме выделены следующие компоненты:

- физическое устройство «User PC» – рабочая станция сотрудника. В нее входит программная среда «Web Server», с помощью которой сотрудник работает с веб-приложением «CRM-система», запуская процесс тестирования. Также на рабочей станции размещена внешняя система (мессенджер) «Messenger», при помощи которой сотрудник уведомляется о результатах;

– физическое устройство «Web Server» - сервер, на котором размещено веб-приложение. Запросы на сервер отправляются через сеть Интернет по протоколу HTTPS. При получении данных программный файл «Controller» обрабатывает их и в среде исполнения «Java Virtual Machine» запускает исполняемый файл «Server.jar», где запускается браузер и исполняемый файл «Webdriver», при помощи которого выполняются заданные в тесте действия.

Таким образом, для функционирования проектируемого модуля требуются следующие компоненты:

- сервер с настроенным окружением;
- рабочая станция (пользовательский компьютер) с предустановленным необходимым программным обеспечением;
- интернет-соединение.

В проектируемом модуле не используется база данных, поэтому проектирование логической и физической модели базы данных не требуется.

Таким образом, было произведено моделирование архитектуры модуля автоматизированного тестирования и выяснены требования к необходимым для функционирования модуля компонентам.

2.4 Требования к аппаратно-программному обеспечению информационной системы

Техническое обеспечение системы, а также требования к безопасности и производительности должны эффективно выполняться, чтобы обеспечить качество работы и использовать существующие на предприятии технические средства.

Аппаратно-программное обеспечение должно соответствовать следующим требованиям:

- операционная система – Windows 7 и выше;
- центральный процессор – частота не менее 2 ГГц;

- оперативная память – 8 GB и выше;
- свободное место на жестком диске – 500Мб и больше;
- подключение к сети Интернет.

Серверная часть:

- работа 24 часа;
- операционная система – Unix (Linux);
- процессор – не ниже Intel Xeon E5530 2.4 ГГц;
- оперативная память – от 8 Гб; жесткий диск – от 1 Тб.

Требования к безопасности: требуется разграничить доступ к исходному код общей части информационной системы. Требуется разграничение доступа к запуску тестирования и к итоговым данным – результатам тестирования.

После анализа требований к аппаратно-программному обеспечению для проектируемого модуля был сделан вывод, что имеющееся оборудование можно успешно использовать для реализации спроектированной системы.

Таким образом, были проанализированы и сформулированы требования к аппаратно-программному обеспечению и требования к безопасности проектируемого модуля.

Выводы по главе

В данной главе описано логическое проектирование модуля автоматизированного тестирования. Для этого была выбрана технология логического проектирования модуля, на основе модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ» были разработаны необходимые диаграммы (диаграмма вариантов использования и диаграмма деятельности) с описанием процессов. После анализа основного процесса была выполнена разработка архитектуры проектируемого модуля при помощи построения диаграммы развертывания, на которой были описаны основные компоненты. Затем были сформулированы и описаны требования к аппаратно-программному обеспечению и требования к безопасности для проектируемого модуля.

Глава 3 Физическое проектирование модуля автоматизированного тестирования для системы CRM

3.1 Выбор технологии разработки модуля автоматизированного тестирования

При выборе технологий реализации было необходимо определиться с несколькими компонентами модуля:

- выбрать технологию реализации клиентской и серверной частей модуля;
- выбрать технологию автоматизации;
- выбрать внешнюю систему (мессенджер) для возможности демонстрации результатов тестирования и уведомления пользователей.

3.1.1 Выбор технологии реализации клиентской и серверной частей

При выборе технологии разработки модуля необходимым условием была совместимость с уже существующей системой CRM и технологиями, которые были использованы при ее реализации.

Серверная часть системы реализована с использованием стека LEMP – варианта стека LAMP – комплекса серверного программного обеспечения, широко применяемого в Всемирной паутине. Его популярность обусловлена тем, что все компоненты являются открытыми и бесплатными, а также данная связка является гибкой и высокопроизводительной.

Название стека является акронимом и составлено из первых букв названий компонентов – Linux(L), Engine-X (E), MariaDB (M), PHP (P). Более подробная информация о составе данного стека представлена ниже:

- Linux – семейство Unix-подобных операционных систем на базе ядра Linux. Дистрибутивы Linux уже давно используются в качестве серверных операционных систем и заняли значительную долю этого рынка; по данным компании Netcraft на февраль 2014 года, семь из десяти самых

надёжных интернет-компаний, предоставляющих хостинг, используют Linux на своих веб-серверах [5];

- Engine-X (nginx) – веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах (тестировалась сборка и работа на FreeBSD, OpenBSD, Linux, Solaris, macOS, AIX и HP-UX). Nginx является простым в использовании, быстрым, надёжным и не перегруженным функциями сервером;

- MariaDB – реляционная система управления базами данных (СУБД), ответвление от СУБД MySQL, разрабатываемое сообществом под лицензией GNU GPL. В состав MariaDB включена подсистема хранения данных XtraDB для возможности замены InnoDB, как основной подсистемы хранения;

- PHP (англ. PHP: Hypertext Preprocessor) – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов [7]. PHP отличается наличием большого набора встроенных средств и дополнительных модулей для разработки веб-приложений.

Клиентская часть CRM-системы выполнена с помощью классической тройки HTML, CSS и JavaScript, с использованием jQuery, AJAX и tpl-шаблонизатора для упрощения работы с отображением данных.

Исходя из сформулированного выше требования к соответствию модуля с заданной CRM-системой, для реализации был выбран тот же язык программирования PHP, и те же средства для отображения клиентской части.

3.1.2 Выбор технологии автоматизации

Инструменты, позволяющие автоматизировать действия веб-браузера, пользуются большой популярностью, так как могут быть использованы для широкого ряда задач:

- для тестирования Web-приложений;
- для решения рутинных задач администрирования сайта;
- для регулярного получения данных из различных источников (сайтов) и т.д.

В процессе выбора технологии автоматизации для модуля были рассмотрены несколько инструментов, их положительные и отрицательные стороны.

Selenium – один из наиболее давно известных и используемых инструментов (разрабатывается с 2004 года) [34]. Его достоинствами являются:

- кроссплатформенность;
- бесплатное распространение;
- поддерживаются все основные языки программирования, с помощью которых могут быть реализованы скрипты.

Среди недостатков Selenium можно выделить необходимость уверенных навыков программирования и написания скриптов для работы с инструментом. Это может затруднить его освоение начинающими тестировщиками. Также с помощью Selenium можно осуществлять тестирование только Web-приложений, т.е. его функциональность является ограниченной.

Katalon Studio – фреймворк для автоматизации тестирования веб-приложений, API, мобильных и настольных приложений. Его достоинствами являются:

- кроссплатформенность;
- наличие бесплатной (наряду с коммерческой) лицензии для скачивания и использования;
- упрощенная запись скриптов.

Среди недостатков инструмента можно выделить поддержку малого количества языков программирования и отсутствие детализированных отчетов.

UFT One – инструмент для функционального тестирования веб-приложений, мобильных приложений, API и корпоративных приложений, разработанный компанией Micro Focus. Его достоинствами являются:

- широкий спектр тестируемых приложений;
- упрощенная запись скриптов;
- функционал обновления интерфейса с минимальными изменениями скриптов.

Среди недостатков инструмента можно выделить его моноплатформенность (предназначен для работы в рамках ОС Windows) и ограниченность в используемых языках программирования (VBScript).

По итогам изучения инструментов была составлена сравнительная таблица характеристик (таблица 3):

Таблица 3 – Сравнительная таблица характеристик технологий автоматизации

Сравниваемые инструменты	Selenium	Katalon Studio	UFT One
Характеристики			
Кроссплатформенность	Есть	есть	нет
Условия использования	Бесплатное распространение	Наличие как свободной лицензии, так и коммерческих для командного использования	Платная лицензия
Поддерживаемые языки программирования	Все основные языки программирования	Groovy, Java	VBScript
Легкость в освоении	Требует знаний в программировании и написании скриптов	Легок в освоении, упрощенная запись скриптов	Легок в освоении, упрощенная запись скриптов

После проведенного анализа была выбрана технология Selenium, т.к. для поставленной цели – модуля, интегрированного в существующую систему –

она подходит наиболее хорошо. Для используемого языка программирования (PHP) существует библиотека PHP Web-Driver, разработанная Facebook, что является немаловажным фактором для дальнейшей поддержки модуля.

3.1.3 Выбор внешней системы (мессенджера) для отображения результатов тестирования

Первоначальный выбор производился из наиболее популярных и используемых для коммуникации на проекте мессенджеров. Таким методом было выделено 3 мессенджера: Skype, Discord и Telegram. Все мессенджеры являются кроссплатформенными.

Skype – бесплатное проприетарное программное обеспечение с закрытым кодом, обеспечивающее текстовую, голосовую и видеосвязь через Интернет между компьютерами (IP-телефония), опционально используя технологии пиринговых сетей, а также платные услуги для звонков на мобильные и стационарные телефоны.

Из недостатков данного ПО можно выделить отсутствие открытого API для разработчиков, что не позволяет использовать его для вышеозначенных целей.

Telegram – мессенджер с функциями VoIP, позволяющий обмениваться текстовыми, голосовыми и видеосообщениями, стикерами и фотографиями, файлами многих форматов. Также можно совершать видео- и аудиозвонки, организовывать конференции, многопользовательские группы и каналы.

Telegram предоставляет два разных вида API:

- Bot API служит только для отправки и приёма сообщений из Telegram. Подойдёт для создания ботов или отправки уведомлений через Telegram;

- клиентское API (или, как его иногда некорректно называют, MTProto), позволяет использовать все функции Telegram — точно так же, как это делают официальные клиенты.

Из недостатков можно выделить отсутствие возможности настройки собственного сервера с группировкой каналов, а также периодические блокировки Telegram на территории России и других стран.

Discord – проприетарный бесплатный мессенджер с поддержкой айпи-телефонии (IP-телефония, VoIP) и видеоконференций, предназначенный для использования различными сообществами по интересам, наиболее популярен у геймеров и учащихся. Discord предоставляет разработчикам свой API для создания ботов и прочих целей [28].

После проведенного исследования выбор был остановлен на мессенджере Discord, как на наиболее подходящем для обозначенных целей. В данном ПО есть возможность создания собственного сервера, что является готовым механизмом для разграничения доступа к информации. На сервере можно создавать несколько каналов для вывода различной информации (результатов тестирования, логирования ошибок). API Discord открыто и доступно для разработчиков, а так же к данному мессенджеру не предъявлялось претензий от Роскомнадзора и его можно спокойно использовать.

3.2 Выбор среды разработки модуля автоматизированного тестирования

При выборе среды разработки учитывалась сложность модуля в плане разработки. Написанные тесты хранятся в формате JSON, для вывода данных на экран не используются сложных фреймворков типа React/Angular/Vue JS, поэтому использование интегрированных сред обработки (IDE) было возможным, но не обязательным.

Для разработки модуля использовался Sublime Text 3 - проприетарный текстовый редактор с поддержкой плагинов на языке программирования Python [27]. Его достоинствами являются:

- кроссплатформенность;

– поддержка большого количества языков программирования с подсветкой синтаксиса; также пользователи могут загружать плагины для поддержки других языков.

– менеджер пакетов, позволяющий пользователю находить, устанавливать, обновлять и удалять пакеты без перезагрузки программы. Менеджер поддерживает установленные пакеты в актуальном состоянии, загружая новые версии из репозитория;

– удобный интерфейс с отображением кода в виде мини-карты, при помощи которой можно осуществлять навигацию по файлу (рис. 8);



```
selenium_app_controller 2.php [ ]
}
self::$params = [];
}

public function argsParse()
{
    if (!empty($_SERVER['argv'])) {
        foreach ($_SERVER['argv'] as $key => $value) {
            switch (true) {
                case preg_match('~^slow~', $value):
                    $t = explode('=', $value);
                    $this->slow = (int) $t[1] ?: 1;
                    break;
                case preg_match('~^t~', $value):
                    $t = explode('=', $value);
                    $this->tests = explode(',', $t[1]);
                    break;
                case preg_match('~^vnc~', $value):
                    $this->vnc = true;
                    break;
                case preg_match('~^screen~', $value):
                    $this->screen = true;
                    break;
                case preg_match('~^silent~', $value):
                    $this->silent = true;
                    break;
                case preg_match('~^wait~', $value):
                    $this->waitChange = true;
                    $t = explode('=', $value);
                    $this->wait = (int) $t[1] ?: $this->wait;
                    break;
                case preg_match('~^file~', $value):
```

Рисунок 8 – Скриншот интерфейса редактора Sublime Text

- встроенный навигационный инструмент, который позволяет пользователям перемещаться между файлами, а также внутри них, с помощью нечёткого поиска;

- выделение столбцов, множественная правка, автодополнение кода и т.д.

Таким образом, в данном пункте была проанализирована сложность разработки модуля и на основе проведенного анализа был выбран в качестве среды разработки текстовый редактор Sublime Text 3.

3.3 Разработка модуля автоматизированного тестирования в системе CRM

3.3.1 Разработка страницы управления тестами

Разработанная страница модуля тестирования в системе представлена на рисунке 9.

Основными элементами страницы являются:

- форма запуска автотестов;
- форма добавления автотеста;
- форма редактирования автотестов.

Для создания страницы модуля в проекте был создан каталог `tests`, содержащий в себе контроллер страницы `test_app_controller.php` и каталоги `src` и `views` (рис.10).

В каталоге `views` хранятся шаблоны формата `tpl`, используемые для вывода данных на страницу (рис.11).

УПРАВЛЕНИЕ ЗАПУСКОМ ТЕСТОВ ИЗМЕНИТЬ ТЕСТ [ДОБАВИТЬ ТЕСТ](#)

Файл тестов:

Опции запуска

- Запись видео
- VNC:5900
- Скриншоты
- Замедление
- Канал дискорда
- Канал дискорда

Тесты

- проверка свободного места
- проверка бэкапов
- проверка выгрузки
- проверка почты в блеклисте
- проверка авторизации и закрытия стартовой модальки
- проверка страницы каталога
- проверка страницы формы обратной связи
- проверка сертификатов
- проверка работы Elasticsearch
- проверка личного кабинета
- проверка наличия расширенных фильтров

[Запустить](#)

Рисунок 9 – Скриншот страницы модуля тестирования в системе

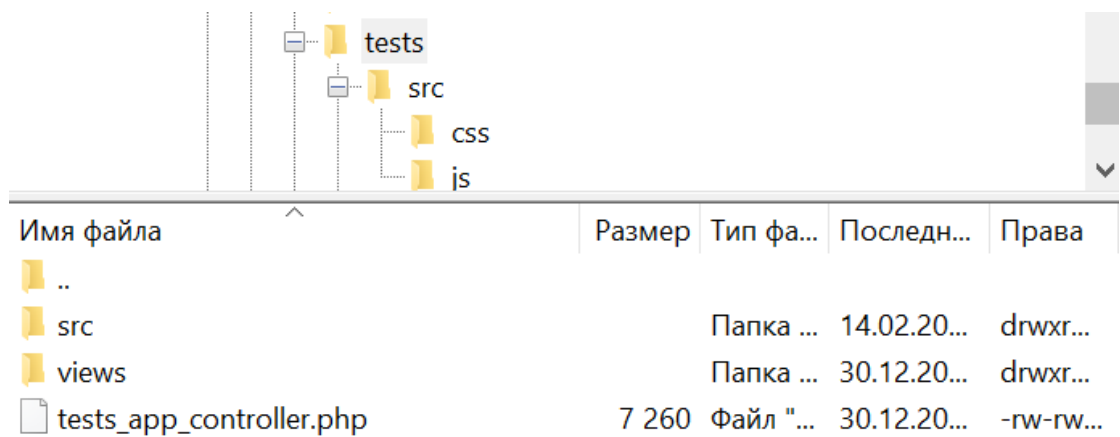


Рисунок 10 - Каталог tests с подкаталогами

channels.tpl	214	Файл "...	18.12.20...	-rw-rw...
index.tpl	6 992	Файл "...	30.12.20...	-rw-rw...
testadd.tpl	1 811	Файл "...	30.12.20...	-rw-rw...
testbody.tpl	857	Файл "...	30.12.20...	-rw-rw...
tests.tpl	1 138	Файл "...	30.12.20...	-rw-rw...
testselect.tpl	1 537	Файл "...	30.12.20...	-rw-rw...

Рисунок 11 - Шаблоны для вывода данных на странице

Трл-шаблоны выводят на экран данные из передаваемых им при рендере компонента массивов данных. В качестве примера показан шаблон channels.tpl, выводящий на экран список возможных для выбора каналов в Discord для отправки результатов тестирования. Шаблон начинается конструкцией “if \$channels”, то есть в случае, если не будет передан массив, компонент не будет заполнен. В цикле foreach перебираются значения в массиве channels и в элемент select добавляются новые option (рис.12).

```

{if $channels}
  <select class="form-control" name="channel" id="form_control_2">
    {foreach $channels as $c}
      <option value="{ $c.key }">{ $c.value }</option>
    {/foreach}
  </select>
{/if}

```

Рисунок 12 - Листинг кода шаблона channels.tpl

Каталог src содержит в себе подкаталоги js и css, в которых хранятся файл index.css с дополнительными настройками стилей и index.js с обработчиками событий действий пользователя на странице модуля.

Основные настройки стилей (цвета и обозначения кнопок, форма и размер чекбоксов, кегль и семейство шрифта и т.д.) используются из уже прописанных в общих файлах стилей самой CRM. В локальном файле

надстроек могут прописываться при необходимости пользовательские настройки стилей.

В файле `index.js` содержатся вышеописанные обработчики событий нажатий на кнопки и другие элементы страницы. Обработчики написаны с использованием JavaScript-библиотеки `jQuery` и концепции `AJAX` – технологии динамического обращения к серверу «на лету», без необходимости перезагрузки всей страницы полностью [30].

В качестве примера для реализации обработчика события нажатия на кнопку будет рассмотрен слушатель нажатия на кнопку запуска тестов (рис.13).

```
// запуск тестов
$('#startTest .js-ok').on('click', function(event) {
    event.preventDefault();
    // spinner button
    var l = Ladda.create(this);
    l.start();
    $.ajax({
        url: '/tests/startTest/',
        type: 'POST',
        dataType: 'json',
        data: $('#startTest').serialize(),
    })
    .done(function(r) {
        if (r.status == 'ok') {
            toastr['success'](r.message);
        } else {
            toastr['error'](r.message);
        }
    })
    .fail(function(r) {
        toastr['error']('ошибка аjax, смотри консоль');
        console.log(r);
    })
    .always(function() {
        setTimeout(function() {
            l.stop();
        }, 500);
    });
});
```

Рисунок 13 – Листинг кода обработчика события нажатия на кнопку запуска тестов

При срабатывании события 'click' у элемента страницы с указанным селектором отправляется POST-запрос на указанный url '/tests/startTest/', который является методом startTest() в контроллере. Все данные из полей ввода, выпадающих списков и выбранных тестов сериализуются и отправляются в формате JSON на сервер в теле запроса. В случае, если запрос отправлен успешно, сервер возвращает ответ, который обрабатывается в блоке done. В случае ошибки пользователь уведомляется всплывающим сообщением об ошибке и логи выводятся в консоль браузера [30].

В качестве примера для реализации обработчика события смены выбранного элемента в выпадающем списке будет рассмотрен слушатель события смены выбранного файла теста (рис.14).

```
// смена файла тестов
$('#[name="file"]').on('change', function(event) {
    event.preventDefault();
    var self = $(this);
    $.ajax({
        url: '/tests/ajaxTests/',
        type: 'POST',
        data: {file: self.val()},
    })
    .done(function(r) {
        $('#tests').html(r);
    })
    .fail(function(r) {
        toastr['error']('ошибка');
        console.log(r);
    })
    .always(function() {
    });
});
```

Рисунок 14 – Обработчик события выбора файла тестов в выпадающем списке

При срабатывании события 'change' у элемента страницы с указанным селектором отправляется POST-запрос указанный url '/tests/ajaxTests/',

который является методом `star ajaxTests()` в контроллере. Значение выбранного в выпадающем списке `option` передается на сервер в теле запроса, и в случае успешного запроса на странице появляются тесты из выбранного файла, в случае ошибки пользователь также уведомляется всплывающим сообщением и ошибка логируется в консоль браузера.

Файл `test_app_controller.php` содержит в себе массивы с данными о каналах в Discord, данными о файлах тестов, которые потом выводятся на экран с помощью шаблонов, и методы для обработки запросов, поступающих с клиента (рис. 15).



```

    $data['tests'] = $this->getData($file);
    return $this->_tpl->render('tests', $data);
}

public function getSelectTests($file){
    $data['testselect'] = $this->getData($file);
    return $this->_tpl->render('testselect', $data);
}

public function renderAddModal(){
    $data['fileid'] = $_POST['file'];
    return $this->_tpl->render('testadd', $data);
}

public function getChosenTest(){
    $json = $this->getJSON($_POST['file']);

    foreach ($json as $key => $value) {
        if ($key == $_POST['name']){
            $data['testbody'][] = [
                'name' => $key,
                'content' => json_encode($value, JSON_PRETTY_PRINT | JSON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE),
            ];
            break;
        }
    }

    return $this->_tpl->render('testbody', $data);
}

```

Рисунок 15 - Скриншот фрагмента файла `test_app_controller.php`

В качестве примера метода для отображения шаблона с данными на странице будет рассмотрен метод `getChannels()`, который заполняет уже рассмотренный выше шаблон `channels.tpl` данными и рендерит элемент на странице. В массив `$data['channels'][]` добавляются элементы из ассоциативного массива `$this->channels`, после чего метод возвращает

результат работы функции `$this->_tpl->render('channels', $data)`, куда передаются в качестве аргументов название шаблона и созданный массив (рис.16).

```
public function getChannels(){  
    foreach ($this->channels as $key => $value) {  
        $data['channels'][] = [  
            'key' => $key,  
            'value' => $value,  
        ];  
    }  
    return $this->_tpl->render('channels', $data);  
}
```

Рисунок 16 - Листинг функции `getChannels()`

Формы добавления нового теста и редактирования существующих становятся видимыми при нажатии кнопок «Добавить тест»/ «Изменить тест» соответственно (рис.17).

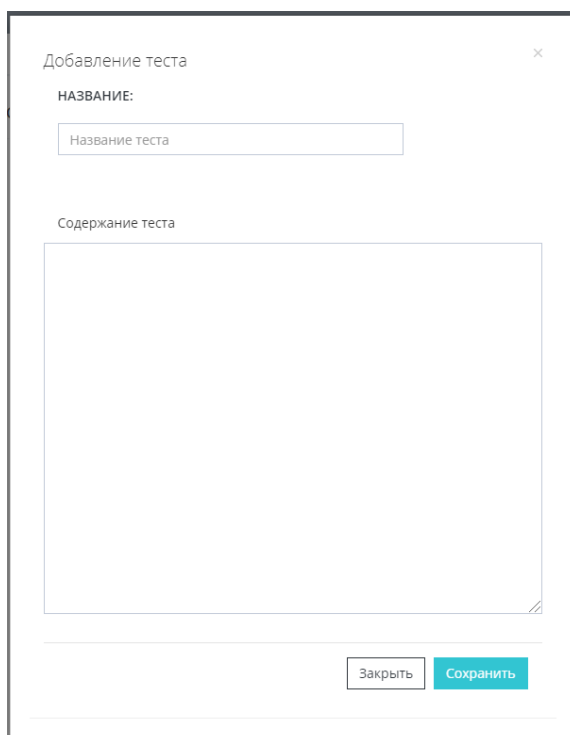


Рисунок 17 – Скриншот формы добавления теста

Для отображения модального окна используется функция `renderAddModal()`, в которой в массив `$data['fileid']` добавляется код выбранного файла тестов из поля формы `$_POST['file']`. Данный массив передается в функцию `$this->_tpl->render('testadd', $data)`, которая рендерит фрагмент `testadd.tpl` (рис.18).

```
public function renderAddModal(){
    $data['fileid'] = $_POST['file'];
    return $this->_tpl->render('testadd', $data);
}
```

Рисунок 18 - Листинг функции `renderAddModal()`

Для сохранения теста используется функция `saveAddedTest()`. В ней в переменную `$json` сохраняется файл с тестами в формате ассоциативного массива «Название теста»:«Содержимое теста» содержимое поля формы `$_POST['file']`. В переменную `$path` сохраняется абсолютный путь к файлу тестов, который является строкой, полученной в результате конкатенации глобальной переменной `INDEX` и локального пути к файлу из массива `$this->files[]`. Далее осуществляется проверка введенного теста, полученного из поля формы `$_POST['testcontent']`, на соответствие синтаксису JSON-файлов, в случае ошибки пользователь уведомляется и процесс сохранения прерывается. В массив `$json` добавляется новое значение с именем теста, полученным из поля формы `$_POST['testname']`, и его содержимым, преобразованным из JSON в PHP-переменную функцией `json_decode()` [31]. Затем файл тестов перезаписывается функцией `file_put_contents()`, куда передаются в качестве аргументов переменные `$path` и преобразованный в JSON массив `$json`. Пользователь уведомляется об успешном добавлении теста. Листинг функции представлен на рис.19.

```

public function saveAddedTest(){
    $json = $this->getJSON($_POST['file']);

    $path = INDEX . $this->files[$_POST['file']];

    if (!json_decode($_POST['testcontent'])){
        die(json_encode([
            'status' => 'error',
            'message' => 'некорректный синтаксис в тексте теста',
        ]));
    }

    $json[$_POST['testname']] = json_decode($_POST['testcontent'], JSON_PRETTY_PRINT / JSON_UNESCAPED_SLASHES / JSON_UNESCAPED_UNICODE);

    file_put_contents($path, json_encode($json, JSON_PRETTY_PRINT / JSON_UNESCAPED_SLASHES / JSON_UNESCAPED_UNICODE));

    echo json_encode([
        'status' => 'ok',
        'message' => 'тест создан успешно',
    ]);

    die();
}

```

Рисунок 19 – Листинг функции saveAddedTest()

В этом пункте была рассмотрена и описана разработка модуля управления тестами в CRM. В следующем пункте будет рассмотрена и описана разработка контроллера для управления selenium.

3.3.2 Разработка контроллера для управления selenium

Для создания страницы модуля в проекте был создан каталог selenium, содержащий в себе контроллер страницы selenium_app_controller.php (рис.20).

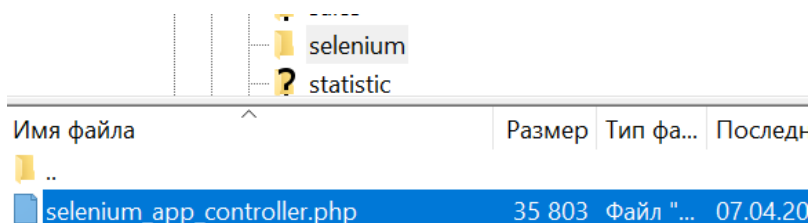


Рисунок 20 - Каталог selenium

В контроллере импортируются необходимые файлы для написания методов, имитирующих различные действия на экранах, из каталога Facebook\WebDriver\ (рис.21).

```

use Facebook\WebDriver\Exception\NoSuchElementException;
use Facebook\WebDriver\Exception\TimeoutException;
use Facebook\WebDriver\Exception\WebDriverCurlException;
use Facebook\WebDriver\Exception\InvalidSessionIdException;
use Facebook\WebDriver\Exception\InvalidSelectorException;
use Facebook\WebDriver\Remote\DesiredCapabilities;
use Facebook\WebDriver\Remote\RemoteWebDriver;
use Facebook\WebDriver\WebDriverBy;
use Facebook\WebDriver\WebDriverDimension;
use Facebook\WebDriver\WebDriverExpectedCondition;
use Facebook\WebDriver\Chrome\ChromeOptions;

```

Рисунок 21 – Импорт необходимых файлов в контроллер

В контроллере объявляются необходимые переменные:

- \$channels - массив с значениями вебхуков каналов в Discord;
- \$selenium - адрес, на котором производится запуск selenium;
- \$wait – максимально допустимое количество секунд, которое необходимо для нахождения элемента на странице;
- \$channel – значение вебхука канала в Discord, используемое по умолчанию и т.д.

На рисунке 22 представлен скриншот фрагмента контроллера с объявлением переменных.

```

class selenium_app_controller extends app_controller
{
    public $selenium      = 'http://localhost:4444/wd/hub';
    public $wait          = 7;
    public $slow          = false; // замедление между действиями
    public $status_child = [];
    public $channels = [
        '672432082248794142/FSKZeLP1fTHZ87t4CrFFsD1Ysvt8tFqx-eJhTr
        '672431973980962826/3C36L_iD08mbz0RQ27nIlajSzxTR0uFrSfp26:
        разработка
        '702762966419898439/d0q8wpcNiKiYz9LIIdjLfbtKLCfQDWgLzC38Hi:
    ];
    public $testfiles = [
        'selenium/autotests.json - автотест',
        'selenium/tests_desktop.json - разработка: декстоп',
        'selenium/tests_mobile.json - разработка: мобильная версия
    ];

    public $channel = '672432082248794142/FSKZeLP1fTHZ87t4CrFFsD1'
    public $driver;
}

```

Рисунок 22 – Листинг объявления переменных в контроллере

Перед запуском функцией `parseArgs()` обрабатываются все переданные из CRM данные: выбранные тесты, выбранный канал в Discord, время задержки между выполнением команд браузером, запись видео/скриншотов во время выполнения и т.д. На рисунке 23 изображен листинг фрагмента функции.

```
public function argsParse()  
{  
    if (!empty($_SERVER['argv'])) {  
        foreach ($_SERVER['argv'] as $key => $value) {  
            switch (true) {  
                case preg_match('~^slow~', $value):  
                    $t = explode('=', $value);  
                    $this->slow = (int) $t[1] ?: 1;  
                    break;  
                case preg_match('~^t~', $value):  
                    $t = explode('=', $value);  
                    $this->tests = explode(',', $t[1]);  
                    break;  
                case preg_match('~^vnc~', $value):  
                    $this->vnc = true;  
                    break;  
            }  
        }  
    }  
}
```

Рисунок 23 – Листинг фрагмента функции `argsParse()`

Запуск автотестов производится с помощью функции `index()`, где выполняются все необходимые проверки перед запуском `selenium` (рис.24).

```
public function index()  
{  
    $this->argsParse();  
    $this->getTests();  
    $this->checkBrowser('chrome');  
    $this->checkDriver();  
    $this->checkXvfb();  
    $this->checkX11vnc();  
    $this->checkJava();  
    $this->checkVideo();  
    $this->startServer();  
}
```

Рисунок 24 – Листинг функции `index()`

В функции `startServer()` производятся все необходимые действия для запуска сервера `selenium` и выполнения автотестов с заданными аргументами. Листинг фрагмента функции `startServer()` представлен на рисунке 25.

```
$this->checkConnect();
$this->startBrowser();
if (!empty($this->tests)) {
    foreach ($this->tests as $test) {
        $this->startTest($test);
    }
}
$this->exitBrowser();

// проверяем не завершился ли какой-либо тест с ошибкой
if (!empty($this->status_child)) {
    foreach ($this->status_child as $v) {
        if ($v > 0) {
            $fail = true;
        }
    }
}
```

Рисунок 25 - Листинг фрагмента функции `startServer()`

Также в контроллере разработаны функции для имитаций действия пользователя в браузере: нажатие по заданному элементу, скролл страницы, поиск элемента с указанным селектором и т.д. [29]. Функции могут быть любой сложности, как атомарными, так и выполнять заданные сценарии.

В качестве примера атомарной функции будет рассмотрена функция `checkCertificate()`, предназначенная для проверки наличия у текущей страницы SSL-сертификата (рис.26). В переменную `$exc` сохраняется результат выполнения функции `executeScript()`, которая выполняет переданный ей в качестве аргумента JS-скрипт.

```
/**
 * Проверка наличия в window ошибки SSL-сертификата на странице
 */
public function checkCertificate()
{
    $exc = $this->driver->executeScript("var d = window.certificateErrorPageController === undefined; return d;");
    return $exc;
}
```

Рисунок 26 – Листинг функции `checkCertificate()`

В качестве примера более сложной функции будет рассмотрена функция `checkAllCertificates()`, которой передаются параметры `$selector` – селектор, по которому будет формироваться массив элементов и `$attr` – атрибут, значение которого извлекается из найденных элементов. Функция предназначена для проверки действительности SSL-сертификатов на всех страницах (в данном случае поддоменов сайта), ссылки на которые будут найдены в указанном атрибуте найденных элементов. Функция в цикле выполняет переход по всем найденным ссылкам, проверяет действительность сертификата, и если результат отрицательный, то записывает ссылку в массив `$answ`. В результате работы функции, если массив `$answ` непустой, то с помощью функции `implode()` из него формируется единая строка и возвращается как результат работы функции, иначе возвращаемый результат – пустая строка. Листинг функции представлен на рисунке 27.

```
/**
 * Проверка наличия ошибок SSL-сертификата у всех поддоменов
 */
public function checkAllCertificates($selector, $attr)
{
    if (!$this->checkCertificate()){
        $answ[] = $this->driver->getCurrentURL();
    }

    $elements = $this->driver->findElements(
        WebDriverBy::cssSelector($selector)
    );

    foreach ($elements as $item) {
        $urls[] = $item->getAttribute($attr);
    }

    foreach ($urls as $item) {
        $this->driver->get($item);

        if (!$this->checkCertificate()){
            $answ[] = $item;
        }
    }
    if (!empty($answ)){
        return implode("\n", $answ);
    }
    else{
        return "";
    }
}
```

Рисунок 27 – Листинг функции `checkAllCertificates()`

Для упрощения создания файлов тестов при написании в них используются не полные названия функций, а сокращенные директивы, которые затем парсятся в функции `startTest()` и выполняются необходимые функции (рис.28).

```
case 'click':
    $this->click($v);
    break;
case 'check_cert':
    ${$v} = $this->checkCertificate();
    break;
case 'check_all_cert':
    ${$v[2]} = $this->checkAllCertificates($v[0],$v[1]);
    break;
case 'check_domains':
    ${$v[3]} = $this->checkDomains($v[0],$v[1],$v[2]);
    break;
```

Рисунок 28 – Фрагмент листинга функции `startTest()`

В этом пункте была рассмотрена и описана разработка контроллера для управления `selenium`. В следующем пункте будут рассмотрены и описаны формирование каталога с данными и настройка окружения.

3.3.3 Формирование каталога с необходимыми файлами для запуска сервера и настройка окружения

Для хранения необходимых для запуска автотестов файлов на сервере был создан отдельный каталог `selenium`. В нем содержится подкаталог `php-webdriver` с файлами библиотеки `Facebook\WebDriver`, драйвер для браузера Chrome `chromedriver`, сервер для запуска `selenium server.jar` и файлы тестов: `autotests.json`, `tests_desktop.json` и т.д., а также файл `readme`, содержащий в себе информацию о командах для запуска (рис.29).

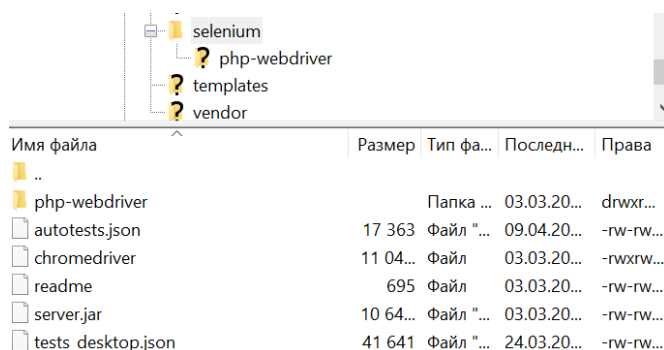


Рисунок 29 – Скриншот содержимого каталога selenium

Также на сервере была установлена JDK 1.8 для запуска сервера.

В этом пункте были рассмотрены и описаны формирование каталога с данными и настройка окружения. В следующем пункте будет рассмотрено написание файлов с автотестами.

3.3.4 Написание файлов с автотестами

Автотесты должны быть написаны с соблюдением синтаксиса формата JSON. В начале указывается название теста, затем непосредственно записывается набор команд, которые необходимо выполнить. Для того чтобы указать выполняемое действие, используются директивы, и через двоеточие указываются параметры, которые затем используются в соответствующей функции [24]. В начале каждого теста указывается разрешение экрана, которое будет использовано при тестировании, оно может быть либо «maximize» - 1920*1080, используется для тестирования десктопной версии, либо «414*736», используется для тестирования мобильной версии.

В качестве примера будет разобран тест проверки на работоспособность галереи фотографий на карточке товара, возможность переключения изображений на мобильной версии сайта. В браузере открывается заданная ссылка – страница каталога, закрывается приветственное модальное окно. Далее открывается карточка товара, на ней происходит переключение изображений, в переменные var, var2 var3 записываются значения true или

false, если необходимый элемент отображался/нет соответственно. В конце теста все переменные проверяются на истинность, и если хоть одна из них false, то тест считается проваленным. На рисунке 30 показан листинг описанного автотеста.

```
"Мобилка: карусель фотогалереи":[
  {"size":"414*736"},
  {"go":"https://xn--63-5cdesg4ei.xn--plai/catalog/telefony/sotovye-telefony/"},
  {"wait":"div[am-modal='city_availability']"},
  {"click":"div[am-modal='city_availability'] div[am-modal-close]"},
  {"attr":["div[am-cards='normal'] div[am-card] a", "href", "link"]},
  {"go": ["link"]},
  {"click":"div[aria-label='Next slide']"},
  {"sleep":"2"},
  {"is_displayed":["div[am-image='photo_2']", "var"]},
  {"click":"div[aria-label='Previous slide']"},
  {"sleep":"2"},
  {"is_displayed":["div[am-image='photo_1']", "var2"]},
  {"click":"div[am-image='photo_1']"},
  {"sleep":"2"},
  {"is_displayed":["div[class='pswp__container'] div[class='pswp__zoom-wrap']", "var3"]},
  {"delete_cookies_all":""},
  {"catch_js_errors":""},
  {"if":["true == $var && true == $var2 && true == $var3", "ok", "false"]}
],
```

Рисунок 30 – Листинг автотеста

В этом пункте было рассмотрено написание файлов с автотестами.

Выводы по главе

В данной главе был произведен выбор технологии разработки модуля для автоматизированного тестирования.

Также была выбрана среда разработки для модуля – Sublime Text 3.

Затем по поставленным задачам был разработан модуль для CRM-системы:

- добавлена новая страница для управления автотестами;
- разработан контроллер для взаимодействия с движком selenium;
- сформирован каталог с необходимыми файлами и настроено окружение на сервере; написаны файлы с тестами.

Глава 4 Тестирование разработанного модуля автоматизированного тестирования для системы CRM

4.1 Подготовка к тестированию

Для тестирования системы был использован уже созданный ранее сервер Discord, и были добавлены несколько текстовых каналов (рис.31). Их вебхуки в дальнейшем были добавлены в файл `selenium_app_controller.php`, а текстовые описания – в `tests_app_controller.php` для дальнейшего их использования.

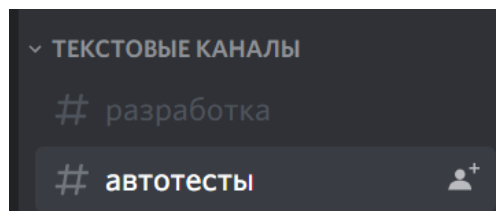


Рисунок 31 – Добавленные каналы на сервере

Также были созданы несколько файлов с тестами и добавлены на сервер в каталог `selenium`.

Для тестирования разработанного модуля были проверены следующие кейсы:

- добавление нового теста через CRM с валидным и невалидным содержимым теста;
- редактирование уже существующего теста с попытками сохранения валидного и невалидного содержимого теста;
- запуск теста с ожиданием позитивного результата выполнения;
- запуск теста с ожиданием негативного результата выполнения;
- запуск теста с записью видео во время исполнения теста;
- запуск теста с формированием скриншотов во время выполнения.

В связи с невозможностью раскрытия коммерческой тайны было принято решение для демонстрации функционала писать примерные автотесты для сайта Тольяттинского государственного университета.

4.2 Прохождение выделенных кейсов

Для проверки кейса «Добавление нового теста с невалидным содержанием» в разработанном модуле CRM был выбран нужный файл тестов, было вызвано модальное окно добавления теста, добавлен текст с несоблюдением формата JSON и нажата кнопка «Сохранить». Ожидаемым результатом было прерывание процесса сохранения теста и появление уведомления об ошибке (рис.32).

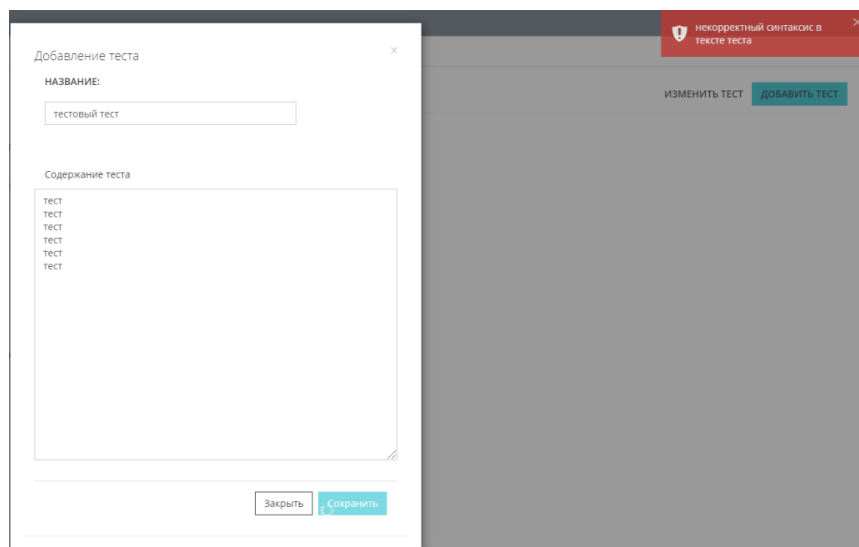


Рисунок 32 – Скриншот страницы модуля при добавлении невалидного теста

Ожидаемый результат совпадает с действительным, данный кейс успешно пройден.

Для проверки кейса «Добавление нового теста с валидным содержанием» в разработанном модуле CRM был выбран нужный файл

тестов, было вызвано модальное окно добавления теста, добавлен текст с соблюдением формата JSON и нажата кнопка «Сохранить». В тесте было проверено наличие и отображение текста кнопки «Сведения об образовательной организации» на сайте ТГУ (рис.33).

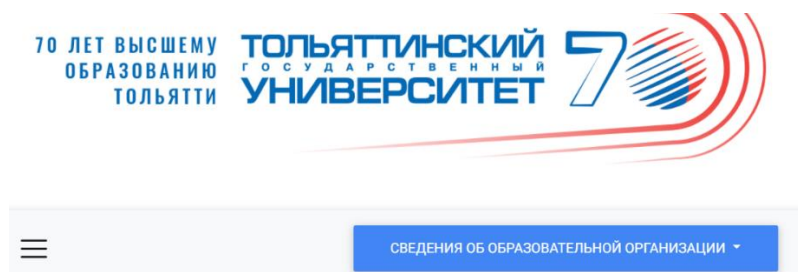


Рисунок 33 – Скриншот фрагмента сайта

Ожидаемым результатом было успешное сохранение теста, обновление списка тестов и появление уведомления об добавлении (рис.34).

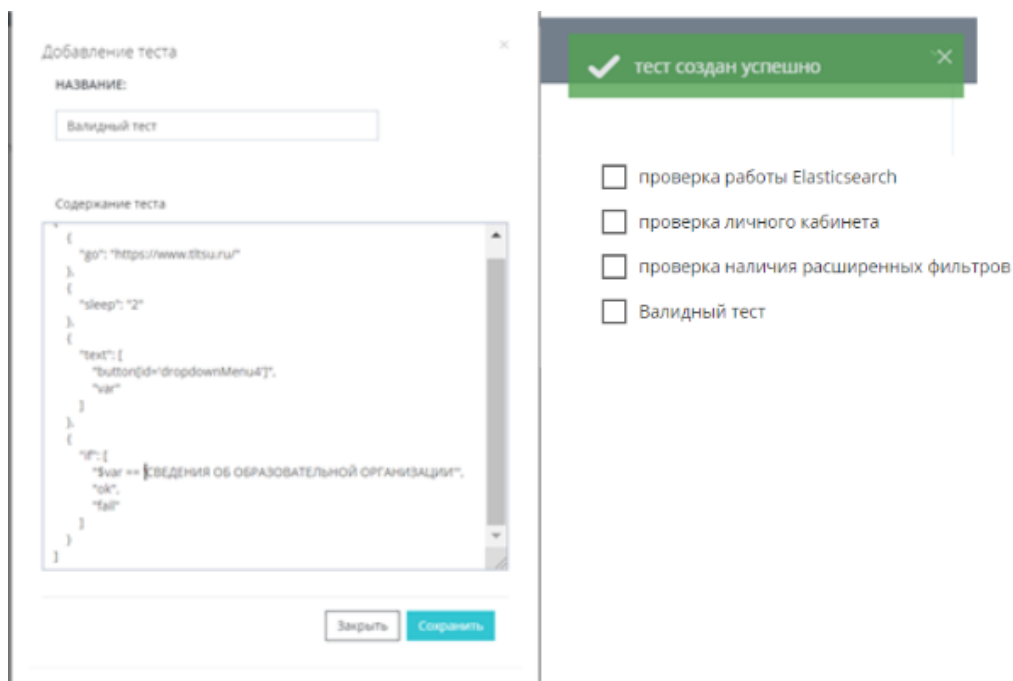


Рисунок 34 – Скриншот страницы модуля при добавлении валидного теста

Ожидаемый результат совпадает с действительным, данный кейс успешно пройден.

Для проверки кейса «Запуск теста с ожиданием позитивного результата выполнения» в разработанном модуле CRM был выбран нужный файл тестов, в списке тестов отмечен к выполнению тест, добавленный в предыдущем кейсе, и нажата кнопка «Запустить» (рис.35).

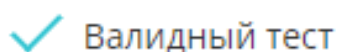
A green checkmark icon followed by the text 'Валидный тест' (Valid test).A teal rectangular button with the text 'Запустить' (Run) in white.

Рисунок 35 – Скриншот страницы модуля при запуске теста

Ожидаемым результатом было возникновение уведомления о запуске теста в CRM и сообщение в выбранном канале в Discord о запуске и успешном прохождении теста (рис.36).

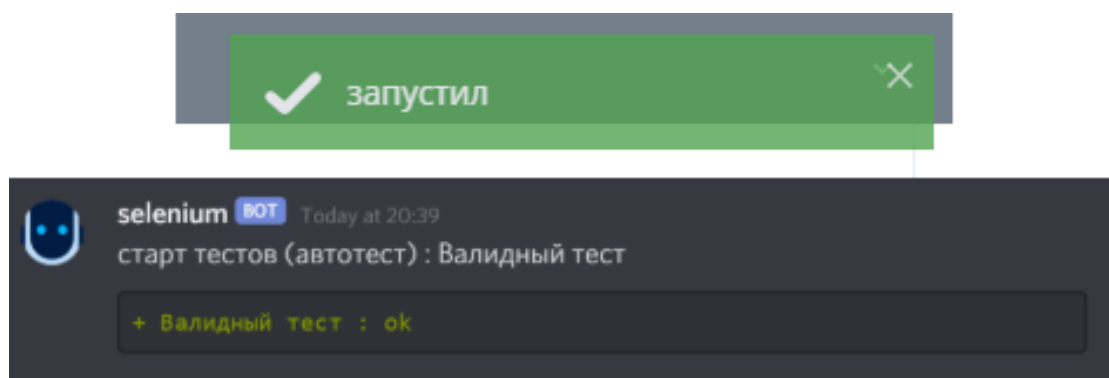


Рисунок 36 – Уведомления об успешно пройденном тесте в CRM и Discord

Ожидаемый результат совпадает с действительным, данный кейс успешно пройден.

Для проверки кейса «Запуск теста с ожиданием негативного результата выполнения» в разработанном модуле CRM был выбран нужный файл тестов, в тест «Валидный тест» были внесены изменения – у указанной кнопки был намеренно указан неверный селектор, сам тест был переименован в «Валидный тест с ошибкой», и тест был сохранен и запущен.

Ожидаемым результатом было возникновение уведомления о запуске теста в CRM и сообщении в выбранном канале в Discord о запуске и неуспешном прохождении теста с указанием ошибки (рис.37).

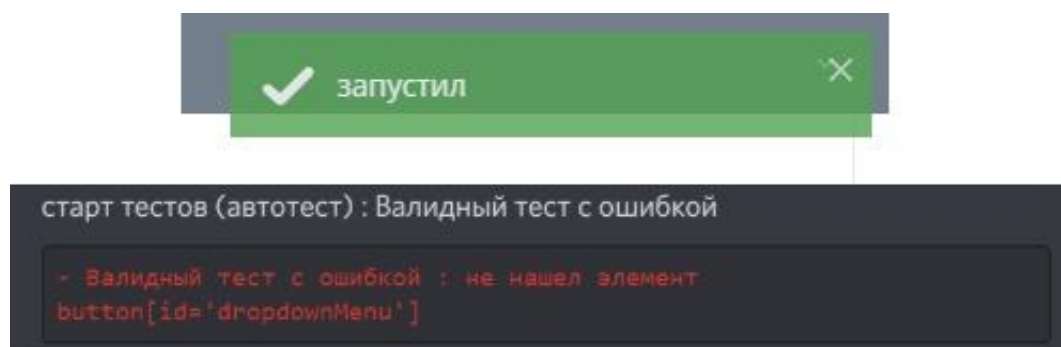


Рисунок 37– Уведомления об неуспешно пройденном тесте в CRM и Discord

Ожидаемый результат совпадает с действительным, данный кейс успешно пройден.

Для проверки кейса «Запуск теста с записью видео во время исполнения теста» в разработанном модуле CRM в опциях запуска был отмечен чекбокс необходимости записи видео (рис.38).

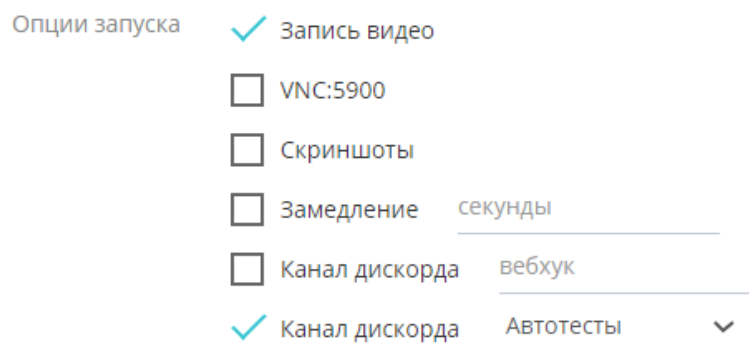


Рисунок 38 – Опции запуска теста

Был выбран тот же файл тестов и запущен тест «Валидный тест с ошибкой», использованный в предыдущем кейсе.

Ожидаемым результатом было возникновение уведомления о запуске теста в CRM и сообщение в выбранном канале в Discord о запуске и неуспешном прохождении теста с указанием ошибки, а также видео с записью работы браузера (рис.39).

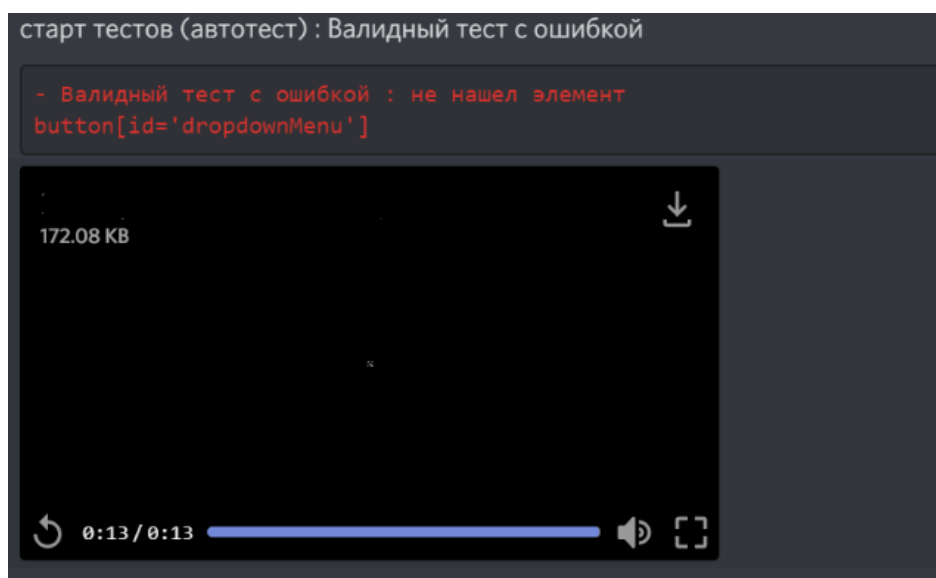


Рисунок 39 – Уведомления об неуспешно пройденном тесте в Discord с приложенной видеозаписью

Ожидаемый результат совпадает с действительным, данный кейс успешно пройден.

Для проверки кейса «Запуск теста с формированием скриншотов во время выполнения» в разработанном модуле CRM в опциях запуска был отмечен чекбокс необходимости записи скриншотов (рис. 40).

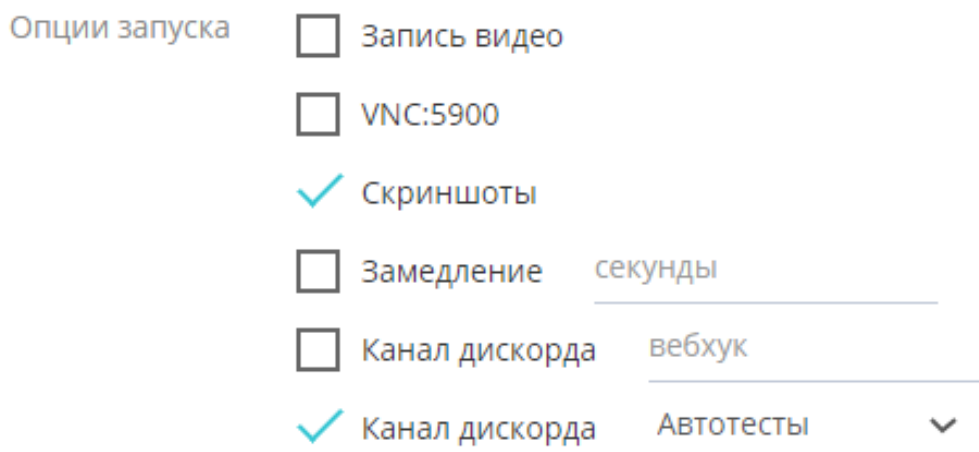


Рисунок 40 – Опции запуска теста

Был выбран тот же файл тестов и запущен тест «Валидный тест с ошибкой», использованный в предыдущем кейсе.

Ожидаемым результатом было возникновение уведомления о запуске теста в CRM и сообщение в выбранном канале в Discord о запуске и неуспешном прохождении теста с указанием ошибки, а также скриншот из браузера страницы сайта, на которой прекратилась работа теста (рис.41).

Ожидаемый результат совпадает с действительным, данный кейс успешно пройден.

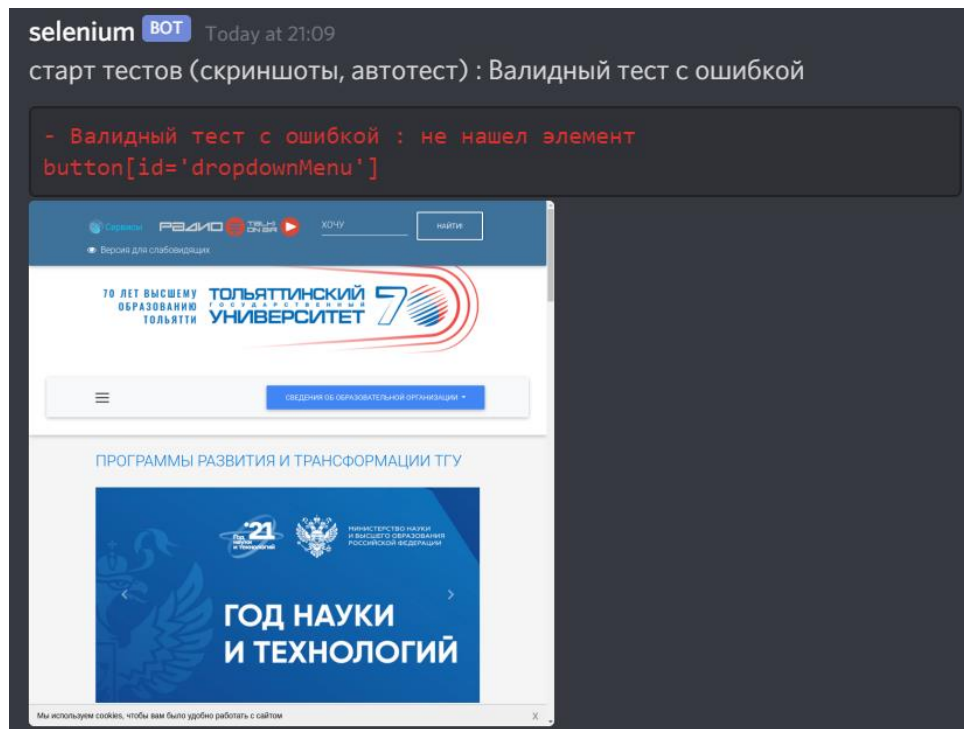


Рисунок 41 – Уведомления об неуспешно пройденном тесте в Discord с приложенной видеозаписью

Выводы по главе

В четвертой главе было произведено тестирование разработанного модуля для автоматизированного тестирования. Для этого были добавлены несколько текстовых каналов в Discord, сформированы несколько файлов тестов, выделен список проверяемых кейсов и описано их прохождение с документированием результатов в виде скриншотов. По итогам все кейсы были успешно пройдены.

Заключение

Итогом выполнения выпускной квалификационной работы является готовый к использованию (разработанный и протестированный) модуль автоматизированного тестирования в CRM-системе.

В ходе функционального моделирования была выполнена технико-экономическая характеристика деятельности предприятия, выполнена разработка и анализ модели «КАК ЕСТЬ», на основе чего была выполнена разработка и анализ модели «КАК ДОЛЖНО БЫТЬ». Были поставлены и определены задачи и цели разработки, основываясь на которых, выявлена необходимость разработки модуля для автоматизированного тестирования в CRM-системе для ООО «Холмонт БР».

В ходе логического проектирования системы определена и выбрана технология логического проектирования, на основе модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ» были разработаны необходимые диаграммы (диаграмма вариантов использования и диаграмма деятельности) с описанием процессов.

В ходе физического проектирования системы был произведен выбор технологии разработки модуля для автоматизированного тестирования, который был обоснован текущим стеком проекта. Таким образом был сформирован список технологий, который включает в себя:

- технологии реализации клиентского модуля – стек LAMP (Linux(L), Engine-X (E), MariaDB (M), PHP (P));
- технология автоматизации – selenium;
- внешнюю систему (мессенджер) - Discord.

Также была выбрана среда разработки для модуля – текстовый редактор Sublime Text 3.

Затем по поставленным задачам был разработан модуль для CRM-системы:

- добавлена новая страница для управления автотестами;

- разработан контроллер для взаимодействия с движком selenium;
- сформирован каталог с необходимыми файлами и настроено окружение на сервере;
- написаны файлы с тестами.

После завершения физического проектирования системы было произведено тестирование разработанного модуля для автоматизированного тестирования. Для этого были добавлены несколько текстовых каналов в Discord, сформированы несколько файлов тестов, выделен список проверяемых кейсов и описано их прохождение с документированием результатов в виде скриншотов. По итогам все кейсы были успешно пройдены.

Дальнейшее развитие модуля автоматизированного тестирования в CRM-системе предусматривает увеличение функциональных возможностей, таких, как:

- автоматический запуск тестов наиболее важного функционала и выполнение их по расписанию;
- добавление в модуль необходимой справочной информация по составлению тестов и описанию функций и т.д.

Подводя итоги, можно сказать, что цели и задачи бакалаврской работы были выполнены. Модуль был успешно разработан, добавлен в CRM-систему, и протестирован, так что является готовым к использованию.

Список используемой литературы

1. Бибо Беэр. jQuery в действии / Бибо Беэр, Кац Иегуда, де Роза Аурелио. – Питер, 2017. – 528 с.
2. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем / А.М. Вендров. – М. : Финансы и статистика, 2013. – 176 с.
3. Гленфорд Майерс. Искусство тестирования программ, 3-е издание / Гленфорд Майерс, Том Баджетт, Кори Сандлер. — Диалектика, 2012. — 272 с.
4. Гради Буч. Введение в UML от создателей языка / Гради Буч, Джеймс Рамбо, Ивар Якобсон. – ДМК Пресс, 2015. – 496 с.
5. Колисниченко Д. Linux. От новичка к профессионалу. – БХВ-Петербург, 2020. – 672 с.
6. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования. – Вильямс, 2019. – 736 с.
7. Кузнецов М. Самоучитель PHP 7 / Кузнецов М., Симдянов И. — 2-е изд.. — СПб., 2018. — 448 с.
8. Ларман, К. Применение UML и шаблонов проектирования: Уч. Пос/ К. Ларман. - М.: Издательский дом «Вильямс», 2013. - 496 с.
9. Леоненков, А.В. Самоучитель UML 2 / А.В. Леоненков. – СПб.:БХВ - Петербург, 2015. – 576с.
10. Локхарт Джош. Современный PHP. Новые возможности и передовой опыт. – ДМК Пресс, 2016. – 304 с.
11. МакГрат Майк. PHP7 для начинающих. – Эксмо, 2017. – 256 с.
12. Перерва А., В. Иванова. Путь аналитика. Практическое руководство IT-специалиста – СПб.; Питер, 2014 – 304 с.
13. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах. – Дело, 2007. – 312 с.

14. Фаулер, М. UML. Основы. Третье издание. / М. Фаулер. – М.:Символ-Плюс, 2016. – 192 с.
15. Фаулер, М. Архитектура корпоративных программных приложений/ М. Фаулер. – М.: Издательский дом «Вильямс», 2014. – 544 с.
16. Этан Браун. Изучаем JavaScript. Руководство по созданию современных веб-сайтов. – Альфа-книга, 2017. – 368 с.
17. Якобсон, А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо. - СПб.: Питер, 2013. - 496 с.
18. Автоматизированное тестирование: что это? Краткое учебное пособие. / [Электронный ресурс] URL: https://logrocon.ru/news/automation_testing, (дата обращения: 11.04.2021).
19. Кликать по Web-элементам при помощи Selenium WebDriver / [Электронный ресурс] URL: <https://www.software-testing.ru/library/testing/testing-automation/3082--web-selenium-webdriver>, (дата обращения: 10.05.2021).
20. Логическая модель предметной области / [Электронный ресурс] URL: <http://analyst.by/diagrams/logicheskaya-model-predmetnoy-oblasti>, (дата обращения: 11.04.2021).
21. Моделирование бизнес процессов с помощью ARIS (express and cloud) области / [Электронный ресурс] URL: <https://businessarchitecture.ru/bussiness-modeling-aris/>, (дата обращения: 11.04.2021).
22. Основы UML – диаграммы использования (use-case) / [Электронный ресурс] URL: <https://pro-prof.com/archives/2594>, (дата обращения: 11.04.2021).
23. Описание бизнес процессов / [Электронный ресурс] URL: <http://www.interface.ru/home.asp?artId=22559>, (дата обращения: 11.04.2021).
24. Разбираемся с данными JSON / [Электронный ресурс] URL: <https://software-testing.ru/library/testing/testing-automation/3037-understanding-json-data/>, (дата обращения: 29.04.2021).

25. Ручное и автоматизированное тестирование: как выбрать эффективный подход / [Электронный ресурс] URL: <https://www.a1qa.ru/blog/ruchnoe-i-avtomatizirovannoe-testirovanie-kak-vybrat-effektivnyj-podhod/>, (дата обращения: 15.04.2021).
26. CRM системы: что это? Простыми словами / [Электронный ресурс] URL: <https://habr.com/ru/post/342446/>, (дата обращения: 11.04.2021).
27. Sublime Text — Краткое руководство / [Электронный ресурс] URL: <https://coderlessons.com/tutorials/raznoe/vyuchit-vozvyshennyi-tekst/sublime-text-kratkoe-rukovodstvo>, (дата обращения: 11.04.2021).
28. Discord Developer Portal — Documentation — Intro / [Электронный ресурс] URL: <https://discord.com/developers/docs/intro>, (дата обращения: 11.04.2021).
29. Example command reference - php-webdriver/php-webdriver – Wiki GitHub·/ [Электронный ресурс] URL: <https://github.com/php-webdriver/php-webdriver/wiki/Example-command-reference>, (дата обращения: 15.04.2021).
30. jQuery API Documentation / [Электронный ресурс] URL: <https://api.jquery.com/>, (дата обращения: 29.04.2021).
31. PHP: Documentation / [Электронный ресурс] URL: <https://www.php.net/docs.php/>, (дата обращения: 29.04.2021).
32. Arthur M. Analysis and Design of Information Systems. Springer, 2015.
33. Johanan Joshua, Khan Talha, Zea Ricardo. Developer's Reference Guide. – Packt Publishing, 2017. – 838 p.
34. Mark Chatham. Selenium By Example - Volume III. Selenium WebDriver. – Lulu Press, 2014. – 124 p.
35. Robbins J.R. Learning Design: A Beginner's Guide to HTML, CSS, JavaScript, and Graphics. -4th Edition, 2013.
36. Uhlmann T. Instant Lift Applications. – Packt Publishing, 2013. – 336 p.