

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

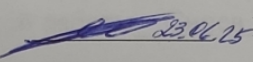
на тему

СИСТЕМА АВТОМАТИЗИРОВАННОГО
ТЕСТИРОВАНИЯ ЛИЧНОГО
КАБИНЕТА СТУДЕНТА

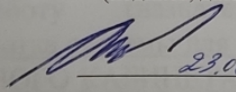
Студент

 20.06.2025 Е.Д. Федулов
(подпись, дата)

Руководитель

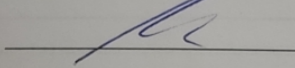
 23.06.25 М.П. Синев
(подпись, дата)

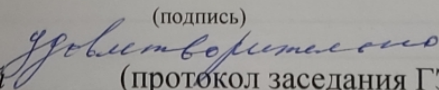
Нормоконтролёр

 23.06.25 А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 28.05.25 г. № 16)

Заведующий кафедрой

 М.А. Митрохин
(подпись)

Работа защищена с отметкой  (протокол заседания ГЭК от 24.06.25 № 19)

Секретарь ГЭК

 Е.И. Гурин
(подпись)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 6 » мая 20 25 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Федулов Егор Дмитриевич гр. 21BBV1 факультета
вычислительной техники

2. Тема работы «Система автоматизированного тестирования личного кабинета студента»

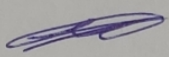
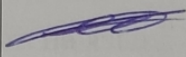
Тема утверждена приказом ПГУ № 601/0 от " 6 " мая 20 25 г.

3. Руководитель бакалаврской работы Синев Михаил Петрович

4. Задание на работу (назначение разработки, исходные данные и т.п.) Разработка системы для реализации автоматизированных функциональных проверок модуля ЛИЧНОГО КАБИНЕТА СТУДЕНТА из состава информационной системы высшего учебного заведения

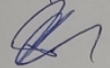
5. Перечень подлежащих разработке вопросов 1 – Анализ предметной области
2 – Использование автоматизированного тестирования
3 – Выбор инструментов и решения
4 – Проектирование и разработка автоматизированных тестов
5 – Анализ результатов автоматизированного тестирования

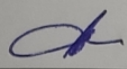
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	32	26.04.2025 – 29.05.2025	
2.	Анализ требований	32	29.05.2025 – 31.05.2025	
3.	Выбор технологий, применяемых при проектировании и разработке	30	01.05.2025 – 04.05.2025	
4.	Описание алгоритмов	60	04.05.2025 – 09.05.2025	
5.	Разработка системы автоматизированного тестирования личного кабинета студента	120	09.05.2025 – 23.05.2025	
6.	Оформление пояснительной записки	54	24.05.2025 – 31.05.2025	

Дата выдачи задания " 6 " Мая 20 25 г.

Руководитель бакалаврской работы 
 (подпись, дата)

Задание к исполнению принял студент 
 (подпись, дата)

Работу к защите допустить
 Декан факультета  Л.Р. Рязанова
23.06.2025 (подпись, дата)

Реферат

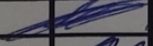
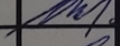
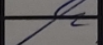
Бакалаврская работа содержит 80 листов, 17 рисунков, 11 источников и 3 приложения.

АВТОТЕСТЫ, АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ, ВЕБ-САЙТ, SELENIUM, ТЕСТИРОВАНИЕ СИСТЕМЫ, БРАУЗЕР

Цель работы – изучение систем автоматизированного тестирования и разработка программной системы для реализации функциональных проверок личного кабинета студента.

Объектом разработки является система автоматизированного тестирования, осуществляющая функциональную проверку личного кабинета студента. Данная система позволяет выполнять виртуально действия пользователя и генерирует отчет о проведенных проверках.

В результате проведенной работы разработана система автоматизированного тестирования на базе фреймворка Selenium на языке Java.

					ПГУ 1.090301.20.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Федулов Е.Д.		20.06.25	Система автоматизированного тестирования личного кабинета студента Пояснительная записка	Лит.	Лист	Листов
Провер.		Синев М.П.		20.06.25			3	80
Ил. Контр.		Бычков А.С.		23.06.25		ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		23.06.25		гр. 21BBB1		

Содержание

Введение.....	3
1 Анализ предметной области.....	8
1.1 Образовательный портал университета рпзгу.....	8
1.2 Личный кабинет	9
1.3 Тестирование системы.....	11
1.3.1 Достоинства внедрения автоматизированного тестирования	13
1.3.2 Использование автоматизированного тестирования.....	15
1.4 Выбор инструментов и решения	18
1.4.1 Возможные фреймворки.....	18
1.4.2 Преимущество Selenium	19
1.4.3 Использование IntelliJ IDEA для написания автотестов	20
1.4.4 Тестирование веб-сайта в IntelliJ IDEA	21
1.4.5 Аналоги инструмента Maven	23
1.4.6 Выбор языка программирования	25
1.4.7 Язык программирования Python	26
1.4.8 Язык программирования C#.....	27
1.4.9 Язык программирования Java	28
1.5 Проектирование и разработка автоматизированных тестов	29
1.5.1 Развертывание системы проектирования	29
1.5.2 Установка Java JDK.....	30
1.5.3 Скачивание и установка IntelliJ IDEA.....	31
1.6 Разработка автоматизированных тестов.....	34
1.7 Отчет по результатам автотестирования	39
2 Охрана труда.....	42
2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ.....	42
2.2 Оценка уровня шума в рабочем помещении.....	42
Заключение	45

Список используемых источников	47
Приложение А – UML диаграммы	48
Приложение Б – Листинг программы	51
Приложение В – Презентация.....	73

Введение

В современных условиях образовательного процесса наблюдается активное внедрение информационных технологий, среди которых особое значение приобретают системы дистанционного обучения и управления учебной деятельностью студентов. Одним из центральных элементов подобных систем является личный кабинет студента, который обеспечивает доступ к учебным материалам, оперативную коммуникацию с преподавательским составом, мониторинг собственной академической успеваемости и своевременное получение актуальной информации. Однако по мере расширения функциональных возможностей таких кабинетов и увеличения числа их пользователей возникает насущная потребность в обеспечении надежности и бесперебойности их работы.

В связи с этим одной из главных задач при развитии и поддержке подобных платформ становится обеспечение высокого уровня качества и стабильности их функционирования. Это непосредственно связано с эффективностью процесса тестирования. В настоящее время преобладает практика ручного тестирования веб-приложений, которая характеризуется такими недостатками, как высокая трудоемкость, значительные временные затраты, ограниченные возможности масштабирования и высокая вероятность человеческой ошибки. Поэтому весьма актуальным является переход на автоматизированные подходы к тестированию.

Автоматизация тестирования с использованием языка программирования Java и инструмента Selenium предоставляет возможность существенно ускорить проверку функциональности личного кабинета студента. Благодаря Selenium становится возможным имитировать действия реального пользователя в браузере, что позволяет эффективно выявлять разнообразные ошибки: от некорректного отображения элементов интерфейса до проблем взаимодействия с сервером и обработки запросов от пользователей.

Вместе с тем внедрение автоматизированного тестирования сопряжено с рядом проблемных аспектов, среди которых особую значимость приобретают

					ПГУ 1.090301.20.001 ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

вопросы информационной безопасности. Так как система тестирования требует доступ к персональным данным студентов, особое внимание должно быть уделено механизмам авторизации.

Цель данной выпускной квалификационной работы заключается в разработке и внедрении системы автоматизированного тестирования личного кабинета студента с применением Java и библиотеки Selenium, направленной на повышение качества и эффективности проводимых тестовых процедур.

Практическая значимость работы определяется существенным сокращением временных затрат на проведение тестирования, повышением точности выполнения тестовых сценариев и минимизацией влияния человеческого фактора при контроле важнейших функций системы. Полученные результаты могут быть успешно применены для дальнейшего улучшения системы управления учебным процессом и рекомендованы к внедрению в других образовательных учреждениях.

					ПГУ 1.090301.20.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Образовательный портал университета рпзгу

Образовательный портал — это веб-приложение, предоставляющее обширный спектр информационных ресурсов для студентов и преподавателей на веб-сайте.

Основной целью образовательных ресурсов является представление актуальной информации об университете и событий происходящих внутри него. Особенное внимание на схожих веб-сайтах уделяется интерфейсу, ведь студенту необходимо быстро ориентироваться и находить информацию на нем.

Пример хорошо построенного интерфейса с сайта Пензенского государственного университета рпзгу.ru предоставлен на рисунке 1.

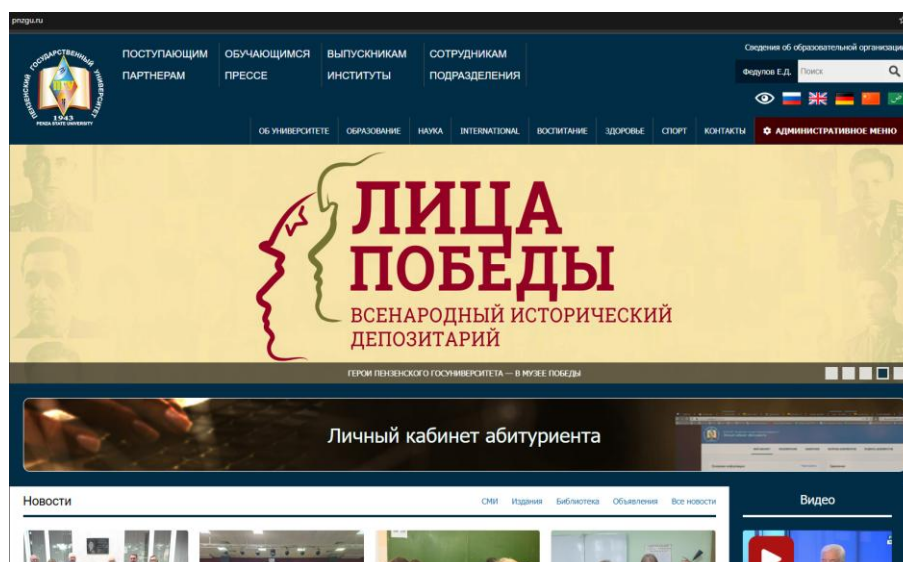


Рисунок 1 – Интерфейс образовательного портала рпзгу

На данном рисунке можно выделить несколько областей, которым нужно выделить особое внимание при отладке веб-сайта:

- основная веб-страница. На ней содержится меню управления перехода по вкладкам;
- веб-страница личного кабинета. На ней содержится интерфейс управления личным кабинетом студентов и преподавателей.

В своей работе я хотел бы уделить внимание личному кабинету студента, так как он является постоянно обновляемой зоной для которой необходимы постоянные проверки интерфейса.

1.2 Личный кабинет

Личный кабинет — это веб-приложение, предназначенное для персонализированного доступа пользователя к информации и функциям на веб-сайте.

Разработка личного кабинета осуществляется в несколько этапов:

- продуктовая аналитика. На первом этапе собираются и анализируются требования заказчика, выявляется целевая аудитория и изучаются аналогичные решения на рынке. В ходе аналитики определяется портрет пользователя, его потребности и ожидания от функционала личного кабинета, что обеспечивает удобство и востребованность конечного продукта;
- составление спецификаций и прототипа. Целью этого этапа выступает определение технических требований и разработка интерактивных прототипов сайта. Формулируются основные функции кабинета, логика взаимодействия пользователя с интерфейсом, а также структура и навигация сайта. В качестве результата создаётся наглядный прототип, демонстрирующий базовую функциональность;
- оценка и планирование. На данной стадии производится расчет стоимости реализации проекта, составляется график выполнения задач и выносится окончательное согласие по срокам разработки. Оценивается наличие потенциальных рисков и факторов влияния на ход работы; результаты реализуются посредством заключения договора с заказчиком;
- дизайн интерфейса. Внимание фокусируется на разработке дизайн-концепций, совмещающих визуальную привлекательность, функциональную целесообразность, интуитивность и удобство для пользователей. Особое внимание уделяется удобству навигации и прозрачности элементов управления;

- программирование. После утверждения дизайна команда разработчиков приступает к программированию и реализации всех функций личного кабинета. Обеспечивается интеграция с необходимыми сервисами, базами данных и прочими системами;

- тестирование. С целью обеспечения корректной работы сайта перед запуском проводится комплексное тестирование с целью выявления дефектов и ошибок. Проверяются все функции личного кабинета, удобство взаимодействия, производительность и уровень защиты данных. Тестирование проводится как вручную, так и с помощью автоматизированных средств;

- запуск сайта. На финальном этапе личный кабинет размещается на сервере и становится доступным пользователям. Проводится финальное тестирование на реальном окружении, после чего сайт открывается для широкой аудитории.

Работа над проектом после запуска обычно продолжается: внедряются обновления, расширяется функционал, оптимизируется производительность и усиливается безопасность. Повторяются этапы аналитики, разработки и тестирования при добавлении новых функций и возможностей. Именно поэтому во внимание был принят личный кабинет студента образовательного портала ПГУ.

Интерфейс сайта личного кабинета студента предоставлен на рисунке 2.

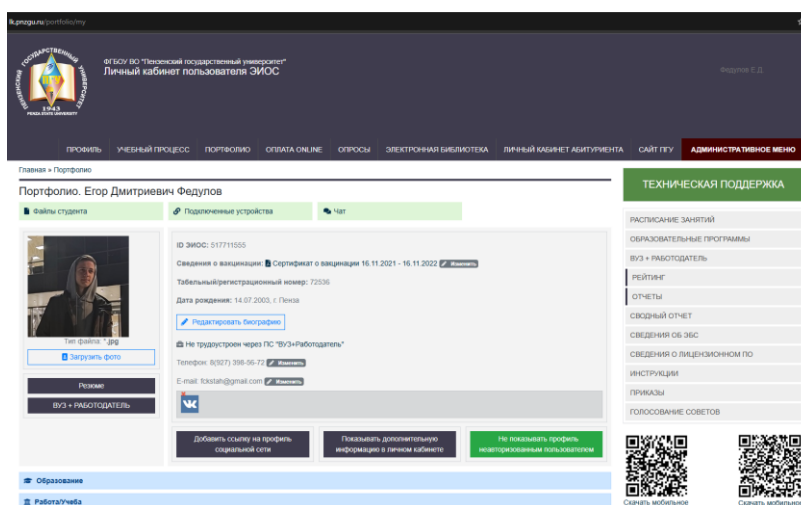


Рисунок 1 – Интерфейс сайта личного кабинета студента

1.3 Тестирование системы

Тестирование веб-сайта (программного обеспечения) — проверки функциональности, удобства использования, совместимости и производительности веб-приложений или веб-страниц, с целью выявления возможных ошибок и обеспечения соответствия поведения сайта требованиям заказчика и ожиданиям пользователей.

Специализированный комплекс тестирования веб-ресурсов предназначен для системного выявления и оперативного устранения неисправностей, а также для повышения уровня надежности и многофакторной производительности веб-ресурса при одновременном контроле показателей совместимости и интегрируемости с многообразными конфигурациями программного обеспечения, аппаратным оснащением и мобильными устройствами, что способствует созданию высококлассного пользовательского опыта и формированию положительных впечатлений.

Предусмотрена реализация подготовленных сценариев и вариантных форм тестирования, охватывающих регистрационные процессы и аутентификационные процедуры, взаимодействие с элементами интерфейса и управление формами, а также обмен сообщениями, кроме того, анализированы нефункциональные параметры, в частности скорость отклика пользовательских страниц, стабильность функционирования при пиковых нагрузках и обеспечение информационной безопасности сайта.

Тщательный анализ взаимодействия пользователя с интерфейсом и обеспечение удобства эксплуатации наряду с надежной защитой ресурса выступают значимыми этапами обнаружения и нейтрализации потенциальных уязвимостей, оказывающих влияние на сохранность пользовательских данных. Аналогично, проведение всестороннего тестирования совместимости гарантирует бесперебойную работу веб-приложения на популярных браузерах (таких как Chrome, Firefox, Safari, Edge) и обширном спектре устройств: начиная от настольных ПК и включая планшеты и мобильные устройства.

Данное обстоятельство свидетельствует о том, что системное и глубокое тестирование веб-ресурсов способствует формированию надежного и эффективного продукта, который совместим с современными требованиями, ожиданиями заказчика и удобен для конечного пользователя. Что, безусловно, способствует успешному выполнению проекта в реальной среде.

Для проверки веб-сайтов применяются разные методы тестирования; некоторые из них представлены ниже:

- функциональное тестирование. Направлено на проверку соответствия работы интернет-ресурса установленным функциональным требованиям. Главная цель такого тестирования — удостовериться в исправности всех процедур сайта, включая регистрацию пользователя, процесс авторизации, навигацию, интерактивность с содержимым и корректность обработки данных пользователей. Данный вариант тестирования не предполагает глубоких знаний о внутреннем устройстве программного продукта, а преимущественно проводится посредством тестирования вручную;

- модульное тестирование. Необходимо проводить для того, чтобы оценить насколько отдельные элементы и страницы веб-сайта соответствуют техническим требованиям. Разработчики в основном проводят данную процедуру в процессе написания программного кода, что позволяет выявить проблемы на ранних этапах создания продукта;

- интеграционное тестирование. Используется для понимания того, насколько правильно работают различные разделы сайта при их одновременном взаимодействии. В основном такое тестирование проводят именно после модульного, что гарантирует стабильность отлаживаемой среды при одновременной загрузке нескольких частей программы;

- нефункциональное тестирование. Направлено на проверку нефункциональных частей веб-сайта.

К основным видами нефункционального тестирования можно отнести:

- тестирование производительности. Необходимо узнать как быстро загружаются и стабильно работают страницы, при их использовании в условиях повышенной нагрузки;
- тестирование безопасности. В связи с тем, что сайты имеют сложную структуру и зачастую создаются множеством специалистов, то появляется необходимость проверки ошибок, которые потенциально могут являться угрозой безопасности;
- тестирование удобства использования. Веб-сайт должен иметь удобный и понятный пользовательский интерфейс, именно поэтому нужно проверить, насколько правильно построена его структура;
- тестирование совместимости. Сейчас существует большая разновидность разных устройств и веб-браузеров, поэтому необходимо отладить и проверить работу веб-сайта на разных наборах устройство-браузер;
- регрессионное тестирование является повторным прогоном всех проверок, которые проводились до внесения изменения в код веб-сайта. Необходимость такого подхода обусловлена тем, что нужно быть уверенным в полной стабильности и правильном функционировании веб-сайта.

Указанные типы тестирования могут выполняться как вручную, так и с помощью автоматизированных систем экспертной проверки.

Характеристика проекта определяется объемом его структуры, бюджетом, временными рамками, штатом сотрудников, а также необходимыми профессиональными навыками для осуществления контроля качества, и зависит от полноты и вида тестирования. Комплексное и рациональное сочетание тестов — это гарант реализации веб-ресурса с глубоким уровнем надежности и предельным соответствием установленным требованиям.

11.3.1 Достоинства внедрения автоматизированного тестирования

Автоматизированный тест веб-программного продукта базируется на внедрении специализированных программных платформ и скриптовых методик, направленных на анализ их функциональной состоятельности и общей результативности без необходимости непосредственного контроля со стороны

опытного специалиста по тестированию. Данный подход позволяет регулярно использовать созданную систему автотестов для своевременного обнаружения уязвимостей и ошибок на веб-сайте.

Автоматизация ручных проверок имеет ряд положительных сторон:

- увеличение охвата тестирования. Выполнение множества действий, в быстром режиме, что позволяет захватить в проверку кратно большую часть веб-сайта, чем бы позволило традиционное ручное тестирование;
- увеличение точности и надежности тестирования. Автоматический тест минимизирует человеческий фактор, за счет чего достигается высокая точность выполнения поставленных задач при тестировании;
- снижение финансовых затрат. Системы автоматизированного тестирования требуют одно большое вложение на внедрение и настройку, после чего требуются минимальные вложения на поддержание ее правильной работы. Это позволяет сократить расходы за счет исключения ручного тестирования и выявления ошибок в работе веб-сайта еще на этапе разработки;
- сокращение времени на регрессионное тестирование: автоматизация регрессионных тестов освобождает тестировщиков от выполнения монотонных повторяющихся задач, позволяя сосредоточиться на более сложных аспектах;
- повышение эффективности команды разработки: автоматические тесты могут запускаться при каждом изменении кода, оперативно выявляя проблемы и сокращая цикл исправления ошибок.

Тем не менее в некоторых обстоятельствах автоматизированное тестирование не может полноценно заменить ручной подход. Существуют особые ситуации, когда применение автоматизации не приносит наилучших результатов:

- значительные начальные вложения: разработка и внедрение автоматизированных тестов требуют существенных стартовых капиталовложений, как финансовых, так и временных;
- ограниченная применимость в нестабильных проектах: автоматизированные тесты проявляют эффективность при условии, что приложение обладает высокой степенью стабильности, а требования

сформулированы однозначно. В начале разработки, когда структура сайта постоянно претерпевает изменения, автоматизация чаще всего не приносит ожидаемых результатов;

- сложности покрытия специфических сценариев: автоматизация проходит с трудом в случаях, когда сайт предъявляет широкие запросы к человеческому взаимодействию, как, например, распознавание визуальных образов, работа с видеоконтентом или сложными интерактивными элементами, поведение которых трудно просчитать и формализовать;

- низкая окупаемость автоматизации при решении простейших и краткосрочных задач: на ресурсах со сравнительно невысокой степенью сложности либо на проектах с ограниченной продолжительностью автоматизация зачастую оказывается нецелесообразной с экономической точки зрения и не дотягивает по эффективности до ручного тестирования.

Выбор методики разработки проекта — автоматизированный или ручной — напрямую зависит от масштабности работ, их сложности и аналогичных параметров, а также от ресурсов, доступных в команде.

11.3.2 Использование автоматизированного тестирования

Системы автоматизированного тестирования активно внедряются в абсолютно разные по своему смыслу этапы разработки, однако можно выделить основных специалистов, использующих в своих задачах автотесты:

- разработчики. Благодаря появлению автоматизированных тестов на ранних этапах проекта, ошибки в коде или правильное функционирование веб-сайта проверяются вовремя, что в свою очередь снижает время выполнения проекта и затраты на него. Специалисты, которые раньше занимались ручным тестированием, освобождаются от монотонной работы, что позволяет им направить свои силы на более сложные задачи;

- аналитики. Пропадает необходимость в постоянном отслеживании того, как пользователь взаимодействует с сайтом. Появление проблемных мест интерфейса или производительности, ведь это выявляется в автоматическом режиме;

- менеджеры проектов. Автоматизированное тестирование предоставляет менеджерам возможность быстрого и точного отслеживания состояния проекта, оценки качества веб-приложения и прогнозирования возможных эксплуатационных рисков. В итоге автотесты способствуют своевременному принятию управленческих решений и поддерживают эффективный контроль за процессами разработки.

Все более широкое распространение автоматизированных тестовых систем не требует обязательного обретения глубоких технических знаний. Современные платформы и инструменты для автоматизированного тестирования доступны для пользователей с разным уровнем квалификации, что способствует расширению спектра их использования нарастает.

В приложении А.1 представлена UML-диаграмма, которая отражает варианты применения автоматизированных тестов.

Процесс автоматизированного тестирования веб-сайтов включает несколько последовательных этапов:

- исследование требований. Специалисты по тестированию проводят подробное изучение как функциональных, так и нефункциональных требований сайта. Важным аспектом становится выявление всех возможных сценариев использования, как позитивных, так и негативных, чтобы впоследствии разработать соответствующие тесты;

- проектирование тестирования. На данном этапе формируется глобальная стратегия тестирования, включая подбор тестовых сценариев, определение исходных данных, составление плана проведения тестовых процедур и распределение ролей внутри команды. Цель — создать четкую картину тех элементов сайта, которые подлежат проверке;

- настройка среды тестирования. Специалисты создают и конфигурируют среду автоматизированного тестирования, устанавливая необходимые программные компоненты, настраивая оборудование. Среда должна максимально точно отражать реальные условия эксплуатации сайта для обеспечения достоверности результатов;

- создание и запуск тестов. В рамках этого этапа формируют детализированные тест-кейсы — документы, описывающие последовательность действий для проверки конкретных функций и параметров. Каждый тест-кейс содержит строго определенные входные и ожидаемые результаты. После создания тест-кейсов специалисты разрабатывают автоматизированные сценарии, запускают их, собирают и анализируют данные о результатах проведения;
- оценка итогов тестирования. На завершающем этапе осуществляется комплексный анализ данных автоматизированного тестирования. Эксперты фиксируют ошибки, выявляют проблемы и несоответствия требованиям. При необходимости проводят повторные запуски тестов в различных условиях с целью удостовериться в стабильности работы сайта. Результатом этапа становится передача выявленных недостатков разработчикам с целью корректировки и оптимизации веб-продукта.

Рисунок 3 визуализирует структурную схему автоматизированного тестирования, демонстрируя последовательность его ключевых этапов.



Рисунок 3 - Блок-схема этапов разработки автоматизированных тестов

1.4 Выбор инструментов и решения

Перед началом решения задач, связанных с применением информационных технологий, следует осуществить тщательный анализ, охватывающий определение наиболее подходящих IT-инструментов и технических средств для успешного достижения поставленных целей. Аналитический этап включает оценку требований к тестируемому веб-сайту, изучение характеристик разнообразных средств автоматизации, а также сопоставление их возможностей, совместимости с проектом и удобства использования. Такой комплексный подход способствует выбору оптимальных инструментов для автоматизированного тестирования, что обеспечивает эффективность и надежность всего процесса проведения тестирования на основе современных IT методов, подходящих именно к специфике конкретного программного продукта.

11.4.1 Возможные фреймворки

Определение оптимального средства автоматизации тестирования находится в тесной взаимосвязи с многочисленными факторами, среди которых находятся специфическая природа тестируемого приложения, тестовые задачи, бюджетные ограничения и опыт тестировщиков. Раскроем основные направления, демонстрирующие широкий ассортимент инструментов для автоматизации тестирования web-приложений:

- Selenium. Выступает как инструментарий для автоматизированного тестирования веб-приложений, который воспроизводит пользовательские операции через браузер и оценивает корректность работы интерфейса. Эта технология интегрирована в обширный стек решений, предоставляющий прикладной программный интерфейс (API) для взаимодействия с элементами DOM страниц. Благодаря этому становится возможным выполнение кликов, текстового ввода, переходов по ссылкам, проверки условий и других интерактивных действий. Прямое управление браузером без опосредованных слоев обеспечивает высокую достоверность и точность испытаний;

- Cypress. Это инструмент, необходимый для автоматизации тестирования аспектов удобства пользователя и производительности веб-сайтов. Cypress за счет проверки нахождения на веб-странице видимых элементов, способен производить действия кликов, переходов на страницы и ввода данных совсем как человек;

- Katalon Studio. Универсальный инструмент направленный на тестирование веб-сайтов, их API и мобильных приложений. В основу Katalon ложатся сразу Selenium и Appium в связке с простым интерфейсом, что позволяет использовать его как уже опытным, так и начинающим специалистам;

- Robot Framework. Являясь инструментом с широким спектром применения, позволяет проводить тестирования, направленные на проверку веб-сайтов, их API и баз данных, а также обычных приложений. Как и Katalon обладает существенным преимуществом для начинающих специалистов – методологией ключевых слов.

11.4.2 Преимущество Selenium

Selenium, по сравнению со своими аналогами, является современным и лидирующим средством для автоматизации тестирования веб-сайтов. Приведу несколько его особенностей в качестве доказательств:

- имеет поддержку всех основных операционных систем (Windows, Linux, MacOS и др.);

- такие популярные языки как Java, Python и C#, использующиеся в автотестах, прекрасно связываются с Selenium. Это позволяет производить выбор нужного решения под определенную задачу;

- поддержка одновременного использования нескольких браузеров гарантирует правильную работу веб-сайта после тестирования у основного потока пользователей. Для этого вместе Selenium взаимодействует с WebDriver, который позволяет производить полную имитацию ручного ввода.

Наиболее популярным инструментом для проведения тестов пользовательского интерфейса веб-приложений по-прежнему остаётся Selenium – многофункциональный фреймворк, позволяющий запускать проверки в любых

браузерах. Для того чтобы подключить данное средство автоматизации к вашему проекту, достаточно добавить соответствующую зависимость в манифест сборки. Пример такой конфигурации показан на рисунке 4.

```
<!-- Selenium Java -->
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>${selenium.version}</version>
</dependency>

<!-- WebDriverManager -->
<dependency>
  <groupId>io.github.bonigarcia</groupId>
  <artifactId>webdrivermanager</artifactId>
  <version>${webdrivermanager.version}</version>
</dependency>
```

Рисунок 4 - Демонстрация процесса интеграции Selenium в проект.

11.4.3 Использование IntelliJ IDEA для написания автотестов

IntelliJ IDEA, интегрированная среда разработки (IDE), созданная компанией JetBrains, предназначена для написания программного обеспечения на языке Java и других языках, рассчитанных на работу с виртуальной машиной Java (JVM). Использование IntelliJ IDEA эффективно при совместном применении с Selenium — инструментом для автоматизации тестирования веб-приложений, позволяющим выполнять проверки веб-ресурсов согласно представленным ниже сценариям:

- инициирование и завершение работы браузера;
- навигация к целевой веб-странице по гиперссылке;
- воспроизведение действий пользователя — кликов, ввода текста и прокрутки;
- извлечение данных из пользовательского интерфейса сайтов;
- корректировка параметров браузера за счёт добавления специальных флагов.

Планомерное применение IntelliJ IDEA начинается с подбора и установки программного обеспечения и Java JDK, синхронизированных с определённой операционной системой. Согласовав эти компоненты, разработчик переводит усилия на написание автоматических тестов с последующим их запуском через интерфейс программы.

11.4.4 Тестирование веб-сайта в IntelliJ IDEA

В среде разработки IntelliJ IDEA главным выбором для реализации автоматизированной тестовой среды признаётся Maven, признанный инструмент, чьи возможности управления зависимостями и исходным кодом обеспечивают заметную надёжность. Начальный интерфейс среды разработки демонстрируется на рисунке 5.

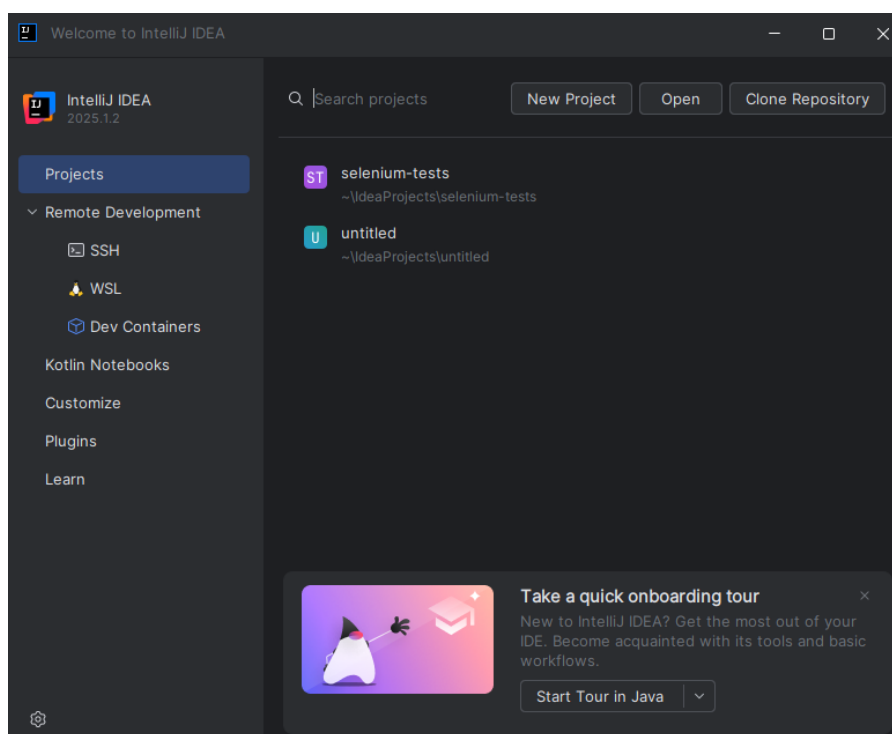


Рисунок 5 - Представлена стартовая страница интегрированной среды разработки IntelliJ IDEA

IntelliJ IDEA тесно взаимодействует с системой управления зависимостями Maven. Это объединение облегчает процесс управления библиотеками, обеспечивает автоматизированное выполнение сборки, а также рационализирует

архитектуру проекта. Благодаря такой слаженной работе с Maven достигается высокая эффективность и комфорт работы.

Создание проекта Maven осуществляется посредством интерфейса IntelliJ нажатием кнопки «New Project», что показано на рисунке 6.

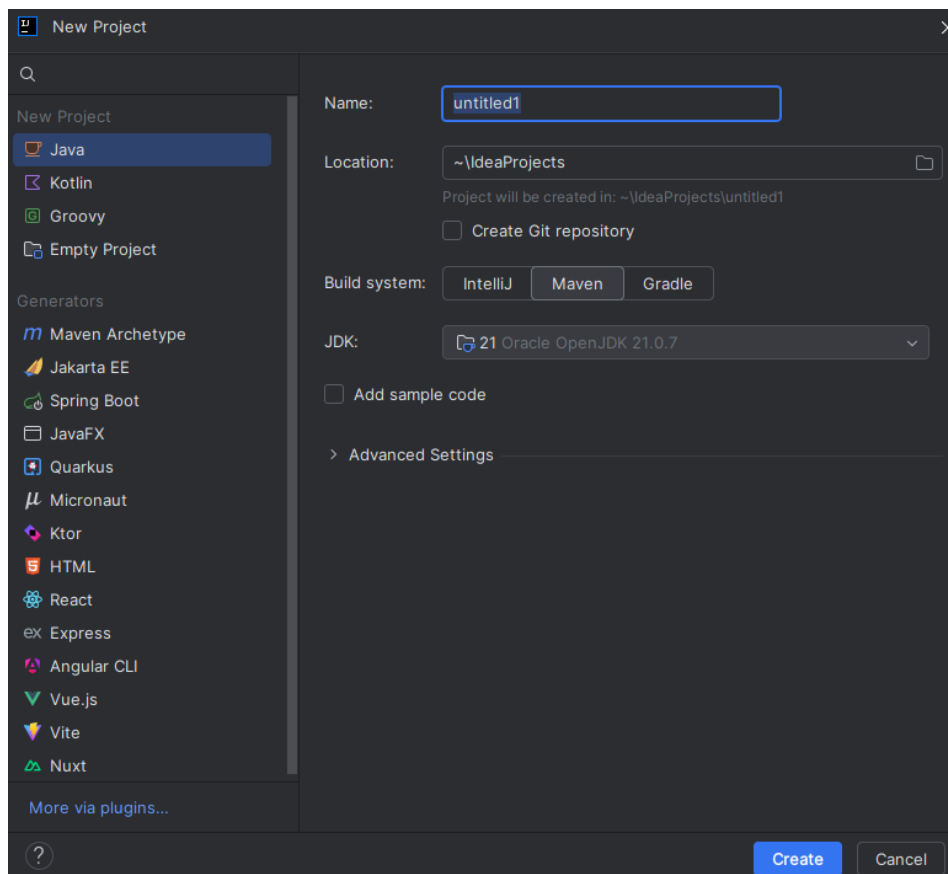


Рисунок 6 — Процесс создания проекта на базе Maven

С помощью Maven можно настроить множество параметров тестируемого браузера: разрешение, используемую версию или отдельные параметры. Такая конфигурация доступна в библиотечных компонентах и показана на рисунке 7.

```

case "chrome":
default:
    WebDriverManager.chromedriver().browserVersion("136").setup();
    ChromeOptions options = new ChromeOptions();
    // Добавляем флаг, разрешающий DevTools-WebSocket
    options.addArguments("--remote-allow-origins=*");
    // Читаем флаг headless из System.getProperty; по умолчанию false
    boolean headless = Boolean.parseBoolean(System.getProperty("key: \"headless\", def: \"false\"));
    if (headless) {
        options.addArguments("--headless=new");
        options.addArguments("--window-size=1920,1080");
    }
    driverThread.set(new ChromeDriver(options));
    break;

```

Рисунок 7 - Структура браузера в контексте использования средства сборки Maven

Однако эмулятор Maven зарекомендовал себя как одна из лучших утилит для создания и тестирования веб-приложений. Он позволяет быстро создавать автотесты на разных платформах без необходимости менять существующий код. Такой подход значительно ускоряет этап отладки.

Кроме того, Maven оснащён уникальным механизмом управления проектами, обеспечивающим простоту интеграции новых проектов и одновременного контроля текущего тестирования. В том числе, она позволяет пропускать отдельные тесты.

Окно управления Maven изображено на рисунке 8.

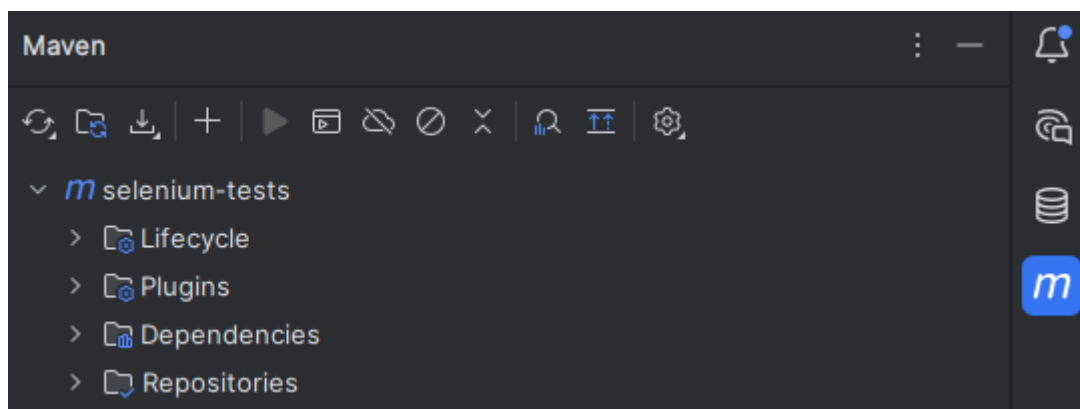


Рисунок 8 - Интерфейсная система управления проектом Maven

11.4.5Аналоги инструмента Maven

Есть несколько альтернатив Maven:

- Gradle. Это мощный аналог Maven, использующий DSL на Groovy или Kotlin, что делает его достаточно гибким и настраиваемым.

Кеширование и инкрементальное построение ускоряют сборку. Параметры запуска тестового конфига можно настраивать для тестирования веб-сайта на разном оборудовании.

Для проектов высокой сложности пониженной сложности или обслуживающих несколько платформ можно использовать Gradle. Данная система сборки характеризуется гибкой настраиваемостью, поэтому хорошо подходит для таких случаев. В контексте интеграции с устройствами под управлением Android, Gradle обеспечивает эффективное управление зависимостями и сборкой, что положительно сказывается на производительности и стабильности скомпилированного ПО. Благодаря этим функциям, Gradle стал стандартом де-факто для Android-проектов.

Рисунок 9 демонстрирует панель управления и структуру проекта Gradle.

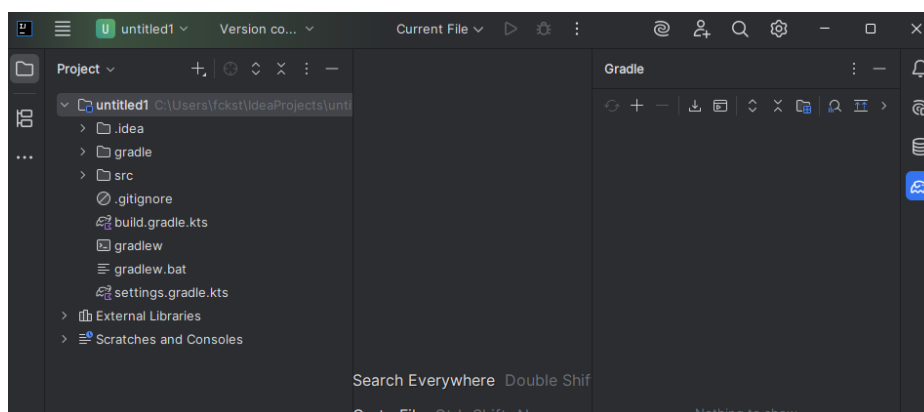


Рисунок 9 - Представление основного интерфейса Gradle

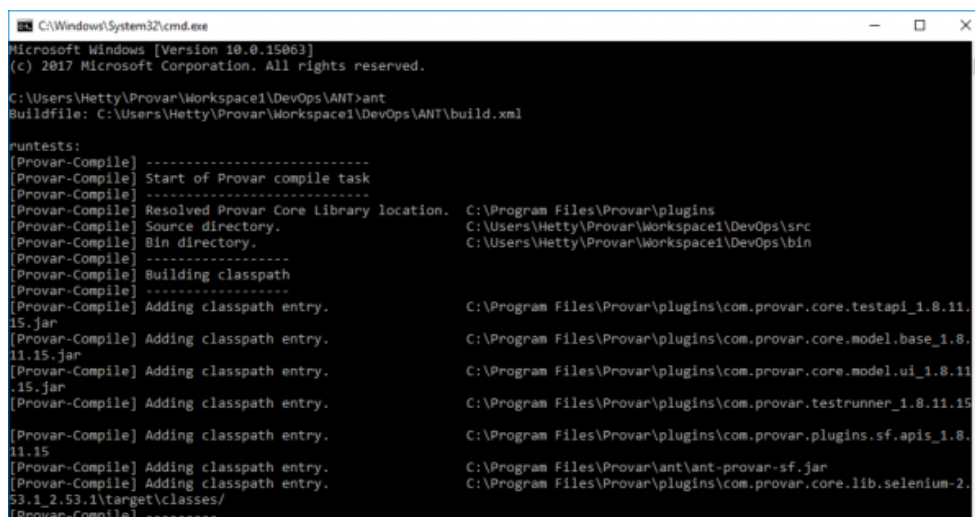
- Apache Ant. В сравнении с Maven и Gradle является устаревшим средством для сборки проекта. Apache Ant имеет смысл использовать при процедурном подходе, когда нам необходимо достаточно точно выделить задачи и последовательность их выполнения, при этом он позволяет сохранять свободную организацию структуры проекта, чего не может предложить Maven.

Ant не имеет собственного управления зависимостями в проекте, для этого необходимо использовать его в связке с Apache Ivy. Несмотря на такой

недостаток, многие продолжают использовать его для программ, в которых требуется настройка сборки на низком уровне.

Еще одной причиной для использования Apache Ant может стать сложная и необычная структура проекта, которая обязывает производить полный контроль над сборкой и отладкой.

Сборка проекта на Apache Ant производится внутри командной строки, что показано на рисунке 10.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Hetty\Provar\Workspace1\DevOps\ANT>ant
Buildfile: C:\Users\Hetty\Provar\Workspace1\DevOps\ANT\build.xml

runtests:
[Provar-Compile] -----
[Provar-Compile] Start of Provar compile task
[Provar-Compile] -----
[Provar-Compile] Resolved Provar Core Library location. C:\Program Files\Provar\plugins
[Provar-Compile] Source directory. C:\Users\Hetty\Provar\Workspace1\DevOps\src
[Provar-Compile] Bin directory. C:\Users\Hetty\Provar\Workspace1\DevOps\bin
[Provar-Compile] -----
[Provar-Compile] Building classpath
[Provar-Compile] -----
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\plugins\com.provar.core.testapi_1.8.11.
15.jar
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\plugins\com.provar.core.model.base_1.8.
11.15.jar
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\plugins\com.provar.core.model.ui_1.8.11.
15.jar
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\plugins\com.provar.testrunner_1.8.11.15
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\plugins\com.provar.plugins.sf.apis_1.8.
11.15
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\ant\ant-provar-sf.jar
[Provar-Compile] Adding classpath entry. C:\Program Files\Provar\plugins\com.provar.core.lib.selenium-2.
53.1_2.53.1\target\classes\
[Provar-Compile] -----
```

Рисунок 10 – Командная строка Apache Ant

11.4.6 Выбор языка программирования

Правильный выбор языка программирования, который будет использоваться для написания автоматических тестов, становится одним из самых необходимых этапов создания проекта. Изучается дальнейшая поддержка и возможность масштабирования создаваемого автотеста, а также насколько продуктивно и качественно будет производиться тестирование с использованием определенного языка программирования. При правильном подборе исключается возможность проблем с адаптацией автоматического теста к новым версиям веб-сайта, что в свою очередь снижает затраты на пересборку проекта в случае возникновения критических ошибок.

В наше время в программировании существует целый ряд языков программирования, использующихся для написания систем автоматизированного

тестирования. Каждый язык наделён уникальным набором достоинств и недостатков, собственным синтаксисом, уровнем поддержки со стороны соответствующих инструментов и характером взаимодействия в рамках сообщества разработчиков. Исходя из этих предпосылок, выбор конкретного языка требует сложного всестороннего анализа: нужно учитывать предыдущий опыт команды специалистов, специфику проекта и его масштаб, а также уровень поддержки актуального набора библиотек и фреймворков для автоматизации тестирования — это усиливает наполнение идеальной экосистемы разработки и повышает продуктивность инженерных процессов.

Языки программирования Python, Java и C# являются универсальными и популярными среди разработчиков решений для автоматизации тестирования веб-приложений благодаря обширной базе библиотек и компонентов, а также развитию экосистем, и они в буквальном смысле интегрированы в инструменты таких фреймворков, как Robot Framework, Cypress, Playwright и Selenium, вот почему специалисты по внедрению выбирают именно эти языки.

11.4.7 Язык программирования Python

Среди языков программирования, применяемых для автоматизации тестирования веб-ресурсов, Python прочно удерживает лидирующие позиции, чему в немалой степени способствует его простая и компактная синтаксическая конструкция. Кроме отличной читабельности исходного кода, Python поддерживает огромное число специализированных расширений, за счет чего специалисты с разным опытом и уровнем подготовки охотно обращаются к этому языку.

Готовый инструмент модульного тестирования Python unittest предоставляет возможность использование дополнительных фреймворк библиотек.

Ниже представлен перечень актуальных и наиболее востребованных Python фреймворков и библиотек, обеспечивающих эффективную разработку автоматизированных тестов:

- PyUnit;
- Nose2;

- PyTest;
- Behave;
- Lettuce;
- Robot Framework;
- Selenium Framework.

11.4.8 Язык программирования C#

C# выступает в качестве универсального и весьма популярного инструментария для разработки скриптов автоматизированного тестирования, наиболее востребованного в сфере тестирования веб-приложений.

Автоматизация тестирования в C# опирается на ряд признанных библиотек и фреймворков, доказавших эффективность в реальных условиях программной инженерии:

- NUnit;
- MSTest;
- xUnit.

Из-за того, что синтаксис C# во многом повторяет Java, специалисты, знакомые с объектно-ориентированными языками, осваивают C# быстро и без проблем. Помимо этого, игнорирование слабой типизации повышает безопасность при разработке нагруженных систем. C# наделён разносторонними возможностями и демонстрирует высокую производительность как при создании динамических, так и устойчивых объектов, а наличие мощных распределённых средств служит как основой тестовой архитектуры, так и сложных систем с глубоким уровнем безопасности.

Итак, учитывая преимущества, такие как высокая потенциальная производительность и удобство разработки, язык C# оказывается наиболее подходящим выбором для тестирования веб-приложений, особенно при условии использования Microsoft Stack. Следует отметить, что окончательное решение о выборе редактора следует принимать с учётом индивидуальных требований конкретного проекта, ограничений целевой платформы и имеющихся ресурсов.

11.4.9 Язык программирования Java

Java неизменно удерживает лидирующие позиции среди языков программирования для автоматизированного тестирования, демонстрируя особенно высокую продуктивность при разработке скриптов для Android-продуктов и веб-ресурсов. Рассмотрим, какие особенности делают Java предпочтительным выбором именно при автоматизации тестирования:

- кроссплатформенная совместимость: автотесты на Java выполняются на разнообразных операционных системах и устройствах, что особенно полезно при тестировании веб-приложений;
- надежность: языковая строгость и встроенные механизмы контроля ошибок делают Java образцом надежности, что становится существенным достоинством для создания масштабируемой автоматизированной инфраструктуры;
- легкость восприятия кода: синтаксис Java предельно ясен и доступен, что упрощает понимание автотестов не только разработчиками, но и тестировщиками с минимальными техническими знаниями;
- разнообразный арсенал инструментов: Java имеет активное сообщество пользователей и широкий спектр решений для автоматизированного тестирования, среди которых выделяются: JBehave, Junit, Serenity и др.

Демонстрируя обоснованное преимущество Java как базового инструмента для реализации автоматизированного тестирования, проект основывается на её широком промышленном применении, многочисленных учебных пособиях по освоению, а также активном участии сообщества пользователей. Большинство современных тестовых фреймворков интегрируются с Java без необходимости дополнительной адаптации. Проект структурирования и масштабирования автотестов поддерживается объектно-ориентированными принципами языка. Важную роль играет применение JUnit, TestNG и Selenium WebDriver, которые формируют фундамент гибких, стабильных и мощных тестовых сценариев.

1.5 Проектирование и разработка автоматизированных тестов

Автоматизированное тестирование подразумевает создание заранее определенных сценариев, которые затем выполняются с помощью специализированных приложений. Данный способ практически исключает возникновение человеческого фактора, что в свою очередь минимизирует риск возникновения случайных ошибок при тестировании. Необходимость такого подхода скрывается за желанием гарантий практически безупречных проверок веб-сайта, в которые будут включены все изначально предусмотренные разработчиком функции.

11.5.1 Развертывание системы проектирования

Необходимые производительность и улучшение результатов работы команды разработчиков достигается за счет разработки системы проектирования. После развертывания, перед созданием проекта, такой подход позволит адаптироваться к особенностям проекта и людям, создающим его.

Первым шагом к созданию системы проектирования является настройка нужной среды разработки, которая подходит под определенные задачи тестировщика и разработчика. Далее, необходимо подобрать подходящий инструментарий, к которому можно отнести IntelliJ IDEA, Visual Studio и др., с помощью которого будет возможно выполнять необходимые от проекта задачи.

Подключение необходимых инструментов, учитывающих технические и функциональные особенности проекта, производится после утверждения проектной документации. В нашем случае скачивается и устанавливается Java JDK (Java Development Kit), с учетом правильно подобранной конфигурации для операционной системы. Далее подбирается инструмент для сборки нашего проекта автоматизированного тестирования, основываясь на ранее подготовленном анализе преимуществ Maven или Gradle.

Одним из последних этапов настройки нашей среды разработки является установка необходимых библиотек и зависимостей, которые позволят ускорить работы и повысить производительность тестов. Выбранные инструменты будут

включать в себя компоненты автоматического тестирования, анализа кода и оптимизацию под браузеры.

Развертывание системы проектирования – комплекс мероприятий, ответственный за обеспечение надежного функционирования всех участников проекта, создание условий для оперативной отладки и тестирования, а также гарантирование высокого качества программного продукта. Настройка системы проектирования, реализуемая с учетом особенностей конкретного проекта, становится ключевым этапом в обеспечении эффективной работы всех его участников и качественном выполнении всех производственных задач. В данной работе представлен подробный разбор этапов настройки выбранной системы проектирования, соответствующих требованиям конкретного проекта.

11.5.2 Установка Java JDK

Java лидирует среди наиболее популярных и востребованных языков программирования на глобальном уровне. Начать практическое освоение Java можно сразу после инсталляции комплекта средств разработки Java Development Kit (JDK). В состав JDK входит полный набор инструментов и библиотек, необходимых для написания, компиляции и запуска программ на Java, что обеспечивает полное сопровождение всего процесса разработки.

Шаг 1: для инсталляции комплекта Java JDK на компьютере, функционирующем под операционной системой Windows, предварительно следует зайти на официальный портал Oracle и подобрать соответствующую редакцию JDK, оптимальную для данной платформы, как показано на рисунке 11.

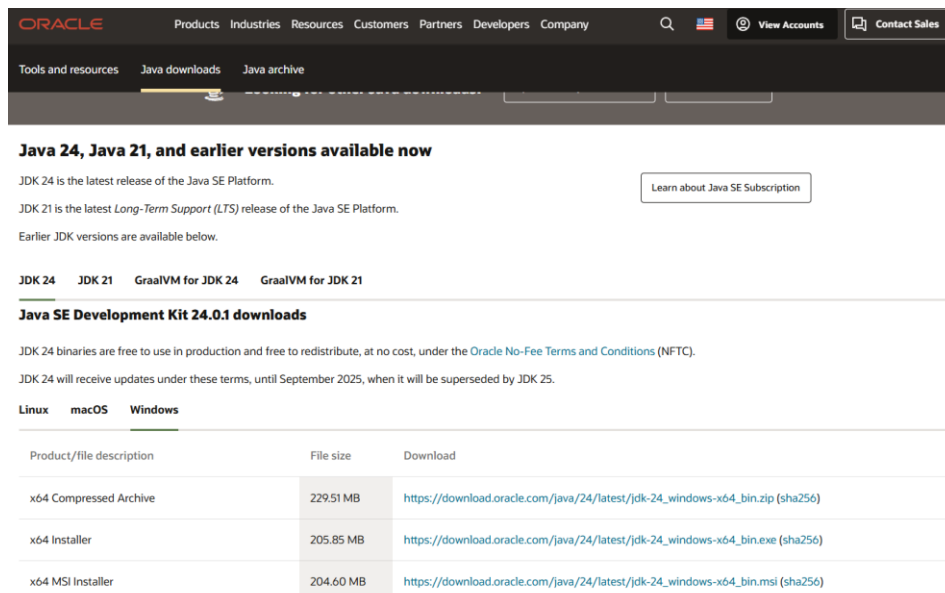


Рисунок 11 - Главная страница официального интернет-портала Oracle

Шаг 2: открыть установщик с разрешением файла .msi или .exe.

Шаг 3: перезагрузить компьютер.

Шаг 4: проверить успешность установки путем открытия командной строки и ввода команды "java -version". Если Java была успешно установлена, появится сообщение с информацией о версии Java, установленной на компьютере, пример предоставлен на рисунке 12.

```
Microsoft Windows [Version 10.0.26100.1742]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\fcxst>java -version
java version "21.0.7" 2025-04-15 LTS
Java(TM) SE Runtime Environment (build 21.0.7+8-LTS-245)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.7+8-LTS-245, mixed mode, sharing)
```

Рисунок 12 - Иллюстрация процесса обнаружения установленной среды Java

11.5.3Скачивание и установка IntelliJ IDEA

Современную операционную систему невозможно представить без интегрированной среды разработки (IDE) – мощного программного комплекса, предназначенного для поддержки жизненного цикла программного обеспечения на всех этапах. Сегодня выбор IDE является одним из ключевых факторов,

определяющих эффективность и удобство труда программиста, скорость написания и отладки программного кода.

Главные параметры выбора оптимальной среды разработки:

- язык программирования. Выбор конкретного языка программирования, на котором планируется разрабатывать проект, — ключевая задача при установке интегрированной среды разработки (IDE). Несмотря на универсальность многих современных платформ, для комплексной и продуктивной работы с отдельными языками часто требуется специализированная настройка. Оптимальный вариант — среда, которая максимально полно поддерживает синтаксис, инструменты и фреймворки выбранного языка, обеспечивая удобство и эффективность разработки;

- функциональность. Разнообразие интегрированных сред разработки (IDE) обусловлено их функциональной насыщенностью, что позволяет подобрать средство с требуемыми особенностями в зависимости от задач и предпочтений программиста. Некоторые из них, такие как IntelliJ IDEA и PyCharm, включают в себя обширный набор инструментальных средств: расширенная система отладки, интеллектуальное автодополнение кода, инструменты профилирования, встроенный модуль для модульного и интеграционного тестирования, интеграционные плагины для систем управления версиями (Git, SVN и прочие), а также средства для организации процессов непрерывной интеграции и доставки (CI/CD). Благодаря такому комплексному функционалу данные IDE обеспечивают высокую производительность и оптимизацию рабочих процессов разработки;

- совместимость с операционной системой. Некоторые среды разработки доступны только под Windows, другие же являются кроссплатформенными и поддерживаются также на Linux и macOS. Это особенно актуально для командной разработки, где могут использоваться разные ОС.

Несколько самых распространенных IDE:

- Visual Studio. IDE от Microsoft, поддерживающая C#, C++, .NET и другие технологии. Отличается высокой производительностью, глубокой интеграцией с экосистемой Microsoft и широким функционалом, включая

инструменты для тестирования, визуального проектирования и разработки под мобильные устройства;

- IntelliJ IDEA. IDE для разработки на Java, также поддерживает Kotlin, Scala, Groovy и другие языки. Обладает мощной системой автодополнения, встроенной интеграцией с системами контроля версий и возможностью работы с базами данных прямо из интерфейса IDE. Имеет как бесплатную (Community), так и платную (Ultimate) версии.

В заключение можно отметить, что выбор среды разработки — это индивидуальный процесс, который должен основываться на специфике проекта, языке программирования, уровне подготовки разработчика и технических требованиях.

Для разработки автоматизированных тестов я использую IntelliJ IDEA.

Для скачивания IntelliJ IDEA нужно перейти на официальный сайт и скачать установщик представленный на рисунке 13.

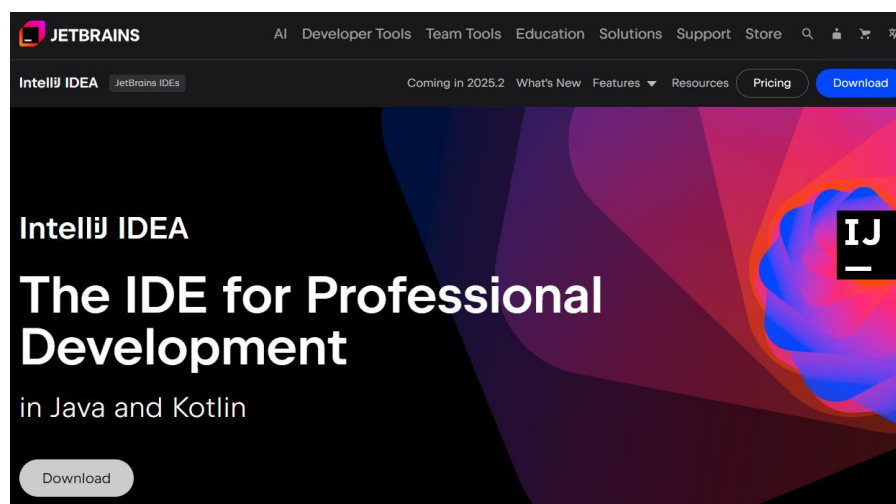


Рисунок 13 - Официальный сайт IntelliJ IDEA

Для установки IntelliJ IDEA необходимо запустить скачанный файл с расширением .exe и следовать всем советам по установке.

После установки и настройки проекта перед нами появится достаточно удобный интерфейс со структурой проекта, показанный на рисунке 14.

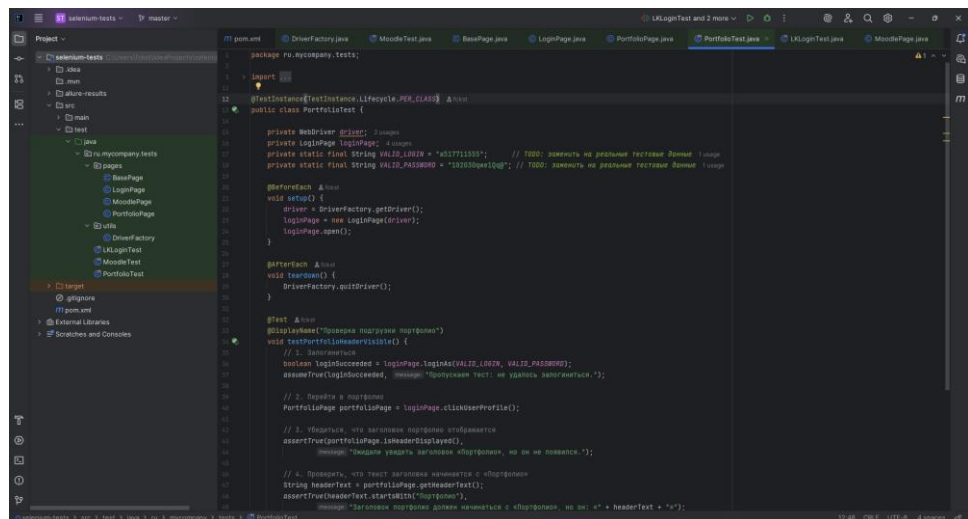


Рисунок 14 – Интерфейс проекта в IntelliJ IDEA

1.6 Разработка автоматизированных тестов

Разработанное приложение представляет собой систему автоматизированного тестирования, предназначенную для проверки ключевых функций веб-интерфейса личного кабинета студента. Программа будет имитировать реального пользователя в браузере за счет библиотеки WebDriver, производя уже описанные тестовые сценарии и записывая их результат.

Например, автоматический тест авторизации на веб сайте может быть разделен на следующие действия: “Открытие веб-сайта”, “Ввод пары логин-пароль”, “Отправка введенных данных”, “Ожидание ответа от сервера веб-сайта”. Все действия описываются в тесте и дают разработчику возможность оценивать каждый этап проверки.

Разберёмся с общим принципом функционирования нашей программы.

Запуск и инициализация:

- при старте каждого теста происходит инициализация драйвера браузера с помощью класса DriverManager, который автоматически подбирает и запускает нужный драйвер в зависимости от настроек окружения (чаще всего используется Chrome через WebDriverManager);
- далее запускается экземпляр браузера и происходит переход на стартовую страницу личного кабинета студента;

Изм.	Лист	№ докум.	Подпись	Дата

- через структуру Page Object (например, LoginPage, MoodlePage, PortfolioPage) выполняется навигация по интерфейсу и взаимодействие с его элементами — ввод данных, клик по кнопкам, чтение текста и т.д.;

- логика подготовки и завершения тестов обрабатывается в базовом классе BaseTest (наследуется всеми тестами), который содержит методы @BeforeEach и @AfterEach для инициализации и очистки ресурсов.

UML диаграмма вариантов использования автоматизированных тестов представлена в приложении А.2.

UML диаграмма развертывания представлена в приложении А.3.

Программное приложение состоит из следующих основных компонентов:

- DriverFactory.java. Объект DriverFactory берет на себя задачи инициализации и конфигурирования браузерного драйвера, используемого в тестовом взаимодействии с интерфейсом «Личного кабинета студента». Он изолирует процесс создания экземпляра WebDriver от главной логики тестов. Такая выделенная фабрика обеспечивает централизованное управление используемым браузером и его параметрами, одновременно упрощая расширение тестовой инфраструктуры при поддержке дополнительного софта.

В предлагаемых вариантах реализации разработки программного обеспечения, класс DriverFactory содержит одну основополагающую функцию: createDriver(). Назначение функции - позволяет создать экземпляр браузера для автоматического тестирования и дальнейшая его передача в функции тестов. Функция возвращает переменную “WebDriver”. Благодаря библиотеке WebDriverManager преднастраивается и открывается приложение chromedriver.exe, после чего создается объект ChromeDriver, который будет управлять браузером Google Chrome, а по завершению возвращается готовый объект WebDriver, готовый к дальнейшему использованию в тестах.

Пример реализации класса DriverFactory из кода DriverFactory.java:

```
import io.github.bonigarcia.wdm.WebDriverManager;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;
```

```

public class DriverFactory {

    public static WebDriver createDriver() {
        // Настройка и загрузка драйвера Chrome автоматически
        WebDriverManager.chromedriver().setup();

        // Создание экземпляра драйвера браузера
        return new ChromeDriver();
    }
}

```

- BasePage.java. В архитектуре Page Object Pattern был создан абстрактный класс BasePage. Он служит основой для всех конкретных классов страниц веб-приложения (например, LoginPage, PortfolioPage, MoodlePage) и инкапсулирует общую логику работы с веб-элементами.

Цель его создания — сокращение повторения кода, при создании дополнительных объектов страниц, и обеспечение идентичного интерфейса взаимодействия с элементами веб-сайта для разных тестов.

Пример создания страницы из кода BasePage.java:

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class BasePage {
    protected WebDriver driver;

    // Конструктор принимает объект WebDriver
    public BasePage(WebDriver driver) {
        this.driver = driver;
    }

    // Метод для клика по элементу
    protected void click(By locator) {
        driver.findElement(locator).click();
    }
}

```

```

    }

    // Метод для ввода текста
    protected void input(By locator, String text) {
        driver.findElement(locator).sendKeys(text);
    }

    // Метод для получения текста из элемента
    protected String getText(By locator) {
        return driver.findElement(locator).getText();
    }

    // Метод для проверки отображения элемента
    protected boolean isElementDisplayed(By locator) {
        return driver.findElement(locator).isDisplayed();
    }
}

```

- PageObject.java. Классы LoginPage, PortfolioPage и MoodlePage реализуют паттерн Page Object, предназначенный для описания логики взаимодействия с конкретными страницами веб-приложения.

Каждый Page Object-класс инкапсулирует локаторы элементов, действия пользователя (например, логин, навигация), методы для проверки отображения контента. Такая структура обеспечивает чистую архитектуру автотестов, изоляцию логики тестов от структуры DOM, лёгкость поддержки при изменении интерфейса.

LoginPage.java – описывает страницу авторизации в системе. Позволяет вводить имя пользователя и пароль, отправлять форму входа и проверять видимость кнопки входа.

Пример создания элементов авторизации из кода LoginPage.java:

```

public class LoginPage extends BasePage {
    private final By usernameField = By.id("username");
    private final By passwordField = By.id("password");
    private final By loginButton = By.id("loginbtn");
}

```

```

public LoginPage(WebDriver driver) {
    super(driver);
}

public void login(String username, String password) {
    input(usernameField, username);
    input(passwordField, password);
    click(loginButton);
}

public boolean isLoginButtonVisible() {
    return isElementDisplayed(loginButton);
}
}

```

PortfolioPage.java – определяет действия и проверки для страницы портфолио. Проверяет, что отображается заголовок портфолио (страница загружена).

Пример проверки загрузки страницы портфолио из кода PortfolioPage.java:

```

public class PortfolioPage extends BasePage {
    private final By title =
By.xpath("//h2[contains(text(),'Портфолио')]");

    public PortfolioPage(WebDriver driver) {
        super(driver);
    }

    public boolean isPortfolioDisplayed() {
        return isElementDisplayed(title);
    }

    public String getPortfolioTitle() {
        return getText(title);
    }
}

```

}

MoodlePage.java – отвечает за взаимодействие со страницей Moodle, в которую пользователь может попасть после входа. Проверяет доступность Moodle-перехода.

Пример проверки перехода на страницу moodle из кода MoodlePage.java:

```
public class MoodlePage extends BasePage {  
    private final By moodleHeader = By.className("moodle-title");  
  
    public MoodlePage(WebDriver driver) {  
        super(driver);  
    }  
  
    public boolean isMoodleHeaderVisible() {  
        return isElementDisplayed(moodleHeader);  
    }  
}
```

1.7 Отчет по результатам автотестирования

Запуск тестирования осуществляется посредством открытия контекстного меню у файла теста и нажатия кнопки “Run”, пример меню представлен на рисунке 15.

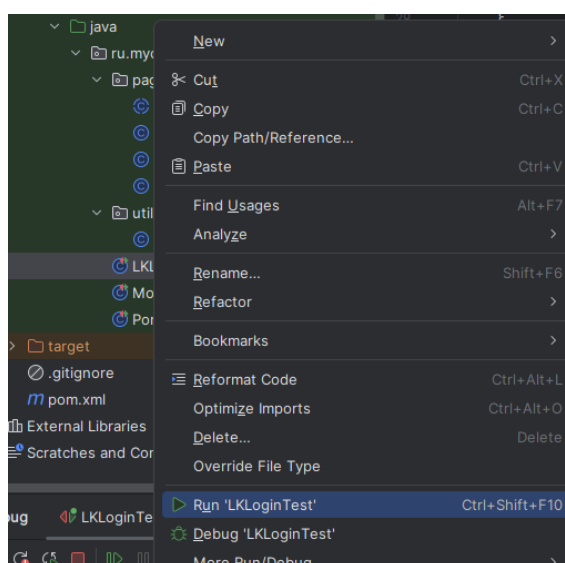


Рисунок 15 – Запуск автотестирования

					ПГУ 1.090301.20.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

После запуска начнет открываться выбранный веб-браузер со страницей, требующей тестирования и которая находится в файле теста (рисунок 16). На рисунке видно контекстное меню Chrome в котором указано, что браузером управляет автоматизированное ПО.

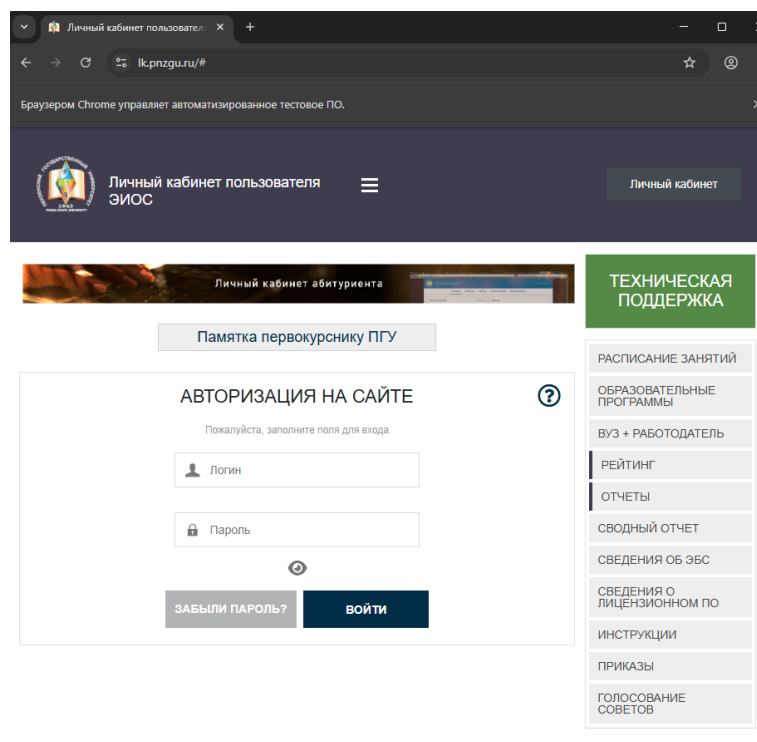


Рисунок 16 – Пример отчета тестирования

По результатам тестирования необходимо просмотреть отчет о выполнении автоматизированных тестов, для чего используется интерфейс IntelliJ IDEA.

Для получения отчета нужно в IDEA дождаться завершения всех тестов в окне тестирования. Пример отчета представлен на рисунке 17.

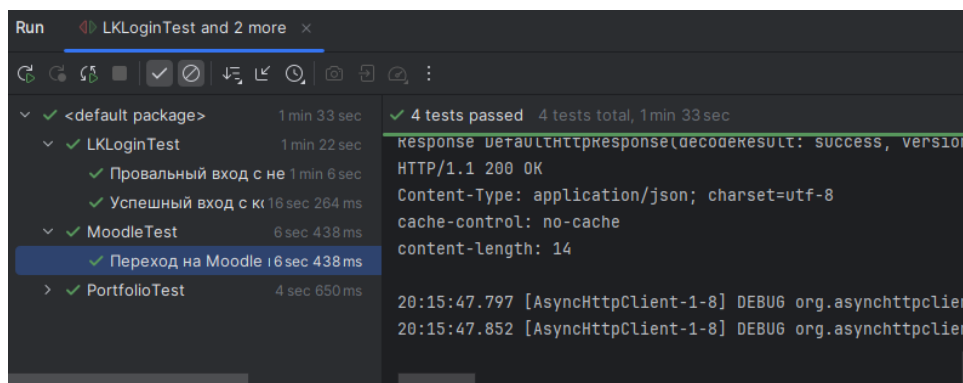


Рисунок 17 – Пример отчета тестирования

Исходя из результатов тестирования, можно сделать вывод, что:

- авторизация в личном кабинете работает успешно;
- интерфейс личного кабинета загружает все нужные детали;
- подсайт личного кабинета moodle успешно загружается.

Таким образом, благодаря написанному автотесту, сайт личного кабинета студента был успешно протестирован полностью автоматизировано и работает правильно.

					ПГУ 1.090301.20.001 ПЗ	Лист
						38
Изм.	Лист	№ докум.	Подпись	Дата		

2 Охрана труда

В данной части рассматриваются требования, направленных на обеспечение безопасных условий труда и охраны здоровья пользователей ПК, занятых в сфере разработки и использования автоматизированного тестирования, а также методы по снижению негативного воздействия на здоровье.

2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ

Обращение с персональными вычислительными устройствами сопровождается наличием потенциально опасных и вредоносных обстоятельств, оказывающих отрицательное воздействие не только на физическое благополучие, но и на информационную безопасность пользователя. При этом важно учитывать такие угрозы в процессе организации деятельности специалистов, вовлечённых в сферу разработки и внедрения автоматизированных средств тестирования.

Работа с вычислительной техникой может негативно сказаться на физическом состоянии организма. Отсутствие физической активности и малоподвижный образ жизни могут стать причиной развития ожирения, сахарного диабета, заболеваний сердечно-сосудистой системы и других патологий. Внимательное отношение к этим факторам и соблюдение необходимых мер предупреждения значительно снижают вероятность возникновения неблагоприятных последствий, связанных с применением вычислительной техники.

Правила оснащения рабочих мест при выполнении операций на персональных компьютерах, сформулированные в государственных стандартах, строго регулируют организацию условий труда, направленную на создание безопасной и здоровой рабочей среды, а также способствующую профилактике профессиональных заболеваний, повышению эффективности и качества выполнения заданий.

Эргономика, представляющая собой научную дисциплину, занимается изучением факторов и принципов, влияющих на взаимодействие людей с эксплуатационными средами. Главные цели науки — создание комфортных условий, снижающих физические напряжения, психическую усталость и риск профессиональных заболеваний для улучшения здоровья и повышения результативности труда. В расчёте на применение в информационных технологиях принципы эргономики приобретают особую актуальность благодаря длительному времени, проведённому специалистами за компьютерами в фиксированных позах. Конструкция и оснащение рабочих мест с учётом эргономических требований способствует снижению негативного влияния технологических факторов на состояние офисных сотрудников, обуславливает комфортность и безопасность выполняемых задач. Особое внимание уделяется креслам в связи с их функционалом поддержки эффективной позы и снятия нагрузки со всех отделов спины пользователя. Для выбора нового комплектующего важно учесть

- биомеханические параметры сиденья, такие как высота, глубина и угол наклона;
- эстетическую и физическую совместимость с плоскостью пола и мебельным пространством;
- наличие оснащения регулируемыми системами для индивидуальной настройки;
- материал конструкции, обеспечивающий достаточную жёсткость без ущерба комфорту;
- износостойкость и простоту очистки, а также возможность быстрой замены изношенных частей;
- соответствие эргономическим требованиям контролирующих органов;
- регулируемая высота сиденья, позволяющая подстроить рабочее место под рост пользователя;
- поддержка поясницы, снижающая нагрузку на нижнюю часть спины и способствующая формированию правильной осанки;

- наличие подлокотников, регулируемых по высоте, для снижения нагрузки на плечи;
- материалы сиденья и спинки, обеспечивающие вентиляцию и комфорт при длительной работе.

Также важно правильно расположить устройства. Клавиатура и мышь должны находиться на расстоянии не менее 10 см от края стола, при этом руки должны оставаться в естественном положении, желательно имея 90 градусов наклона. Клавиатура предпочтительна плоская, с минимальным углом наклона, чтобы предотвратить перенапряжение запястий.

Монитор необходимо располагать так, чтобы его верхняя граница находилась на уровне глаз, а расстояние до экрана составляло от 50 до 70 см. Наклон экрана должен быть в пределах 10–20 градусов вверх от горизонтали. Такое положение снижает нагрузку на зрение и шейный отдел позвоночника. В процессе работы необходимо делать регулярные перерывы и выполнять упражнения для глаз и шеи, что особенно важно при длительной работе за компьютером.

Освещение рабочего места также оказывает значительное влияние на здоровье. Согласно требованиям охраны труда, уровень освещённости должен составлять не менее 300 лк при работе с бумагами и не менее 500 лк при работе с монитором. Источники света должны быть размещены так, чтобы исключить блики на экране и прямое воздействие света на глаза. Рекомендуется использовать светильники с цветовой температурой от 4000 до 6500 К и исключать использование ламп с частотой мерцания ниже 100 Гц. Для локального освещения можно использовать настольную лампу с экранированием, предотвращающим ослепление и отражение света от монитора.

Ключевой момент — микроклимат в помещении, где приходится работать с компьютером. Температура здесь должна поддерживаться в пределах 20–24 °С, при относительной влажности от 30% до 60%. Важно обеспечить качественную вентиляцию для предотвращения переутомления и снижения концентрации внимания; регулярные 45–60-минутные паузы с лёгкой гимнастикой

способствуют сохранению здоровья опорно-двигательного аппарата и улучшают кровообращение.

Соблюдение критериев при формировании рабочего пространства оказывает значительное влияние на предотвращение различных патологий, вытекающих из длительного взаимодействия с вычислительной машиной - как в медицинском аспекте, так и с позиций эффективного выполнения профессиональных задач. Эта практика обусловила повышение выносливости и пропускной способности кадров IT-сектора.

2.2 Оценка уровня шума в рабочем помещении

С физиологической точки зрения шум представляет собой сочетание звуков различных частот, способное оказывать негативное влияние на общее состояние организма, работоспособность и отдых человека. Звуковые волны возникают в результате колебательных процессов в окружающей среде, передающихся через воздух и воспринимаемых слуховой системой.

Человеческое ухо способно воспринимать звуки в диапазоне частот от 16 Гц до 20 000 Гц. Звуковые колебания с частотой ниже 16 Гц относятся к инфразвуку, а колебания свыше 20 кГц — к ультразвуку. Несмотря на то, что данные типы волн находятся за пределами слышимого диапазона, в определённых условиях они также могут оказывать влияние на самочувствие человека и вызывать нежелательные физиологические реакции.

Для оценки степени воздействия шума на человека используется такая характеристика, как уровень звукового давления (L_p), измеряемый в децибелах (дБ). Этот показатель является основным при проведении гигиенических нормирования и акустических расчётов. Контроль уровня шума в рабочей среде необходим для обеспечения комфортных условий труда и профилактики профессиональных заболеваний, связанных с воздействием звуковой нагрузки.

$$L_p = 20 \lg \frac{P}{P_0}, \text{ дБ}$$

P – звуковое давление в точке пространства, Па;

P_0 – пороговое звуковое давление, равное $2 \cdot 10^{-5}$ Па. Оно является нижним порогом слышимости человека на частоте 1000 Гц.

Шум оказывает выраженное негативное влияние на центральную нервную систему, что, в свою очередь, может вызывать нарушения в работе всего организма. Наиболее частыми последствиями длительного воздействия шума являются головные боли, головокружение, повышенная утомляемость и снижение концентрации внимания.

Область восприятия человеком уровня звукового давления находится в интервале от 0 до 130 дБ, где 0 дБ – порог слышимости; 130 дБ – болевой порог.

Согласно временным характеристикам шум классифицируется на следующие категории:

- Постоянный шум, уровень звука остаётся стабильным в течение 8-часового рабочего дня и не изменяется более чем на 5 дБА;
- Непостоянный шум, отличается изменчивостью уровня звукового давления более чем на 5 дБА в течение того же периода.

Для замера уровня звука в определенной точке помещения по уровням звука от различных источников шума производится по формуле:

$$L_{\text{общ}} = 10 \lg(10^{0,1L_1} + 10^{0,1L_2} + \dots + 10^{0,1L_n}),$$

$\Delta = 10^{0,1L_i}$ является табличным значением;

$L_1, L_2, \dots, L_n$ – уровни звука в расчётной точке от различных источников определяются по формуле:

$$L_i = L_{ai} + 10 \lg\left(\frac{1}{S_i} + \frac{4}{B}\right) \quad (1)$$

L_{ai} - максимальный уровень звука, измеренный в децибелах, в определенный момент времени или для определенного события.

Используя формулу (1) и измеренные данные, вычисляем уровень шума в помещении:

$$L_1 = 55 \text{ дБА} + 10 \lg \left(\frac{1}{50,24 \text{ м}^2} + \frac{4}{30 \text{ м}^2} \right) = 44,85 \text{ дБА},$$

$$L_2 = 48 \text{ дБА} + 10 \lg \left(\frac{1}{3,14 \text{ м}^2} + \frac{4}{30 \text{ м}^2} \right) = 44,54 \text{ дБА},$$

$$L_{\text{общ.эксп.}} = 10 \lg(10^{0,1 \cdot L_1} + 10^{0,1 \cdot L_2}),$$

$$L_{\text{общ.эксп.}} = 10 \lg(10^{0,1 \cdot 44,85 \text{ дБА}} + 10^{0,1 \cdot 44,54 \text{ дБА}}) = 47,7 \text{ дБА},$$

Предельно допустимая норма у рабочего места равняется 60 дБА. Данное значение не превышает предельно допустимый уровень, поэтому в моем случае не нужно применять меры для понижения шума.

					ПГУ 1.090301.20.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В рамках данной дипломной работы была разработана система функционального автоматизированного тестирования веб-сайта — личного кабинета студента. Отлаженная система продемонстрировала свою эффективность и практическую значимость в процессе обеспечения качества веб-приложения.

Разработка программного продукта позволила существенно снизить временные затраты на тестирование, параллельно повысив точность и надёжность оценки функционирования веб-ресурса. Внедрение автоматизированных процедур минимизировало объём ручных операций, что уменьшило вероятность ошибок, связанных с человеческим фактором, а также увеличило устойчивость тестовых результатов.

В результате проведённых исследований удалось получить следующие значимые результаты:

- исследованы ключевые понятия и подходы в области тестирования программного обеспечения;
- рассмотрены существующие виды и направления тестирования;
- подробно изучено функциональное тестирование, а также его особенности в контексте веб-приложений;
- проанализированы возможности и преимущества автоматизированного тестирования.

В ходе разработки применялись высокотехнологичные методы и средства, что привело к созданию масштабируемой и адаптивной платформы, ориентированной на безграничное расширение и быструю адаптацию под новые требования. Итоговый продукт представляет собой автоматизированный узел контроля качества функциональности пользовательского интерфейса студенческого портала, обеспечивающий надёжность, масштабируемость и универсальность оценивания работоспособности веб-сайтов. Этот прогрессивный инструмент не только ускоряет тестовые исследования, но и способствует

устойчивому повышению показателей качества конечных продуктов, обретая особое значение в среде современной образовательной деятельности, где цифровые ресурсы выступают краеугольным камнем образовательного процесса.

					ПГУ 1.090301.20.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемых источников

1. Куликов С. С. Тестирование программного обеспечения // Четыре четверти. – 2017.
2. Савин Р. Тестирование Дот Ком // Дело. - 2007.
3. Загидуллин Р. А. Язык программирования Java // Редакционный совет. – 2017. – Т. 43. – №. 3. – С. 889.
4. Нимейер П., Леук Д. Программирование на Java. – Litres, 2022.
// URL: <https://www.litres.ru/book/deniel-leuk/programmirovanie-na-java-8476071/>
5. Матвеева Н. А., Герасимов В. В., Игнатьева Д. О. Исследование технологий модульного тестирования на платформе Java // Информационные технологии. - 2016. – №. 1. – С. 49-55.
6. Пенкина Д. П. Разработка продукционной модели представления знаний в предметной области «Автоматизированное тестирование» // Актуальные вопросы науки и практики и перспективы их решений. – 2023. – С. 62-66.
7. Караваева О. В., Борисова К. В. Разработка системы автоматизированного тестирования // Вестник науки и образования. – 2018. – №. 13 (49). – С. 45-48.
8. Лопатина К. И., Темникова Е. А. Ручное и автоматизированное тестирование программного обеспечения: отличия и взаимозаменяемость // Вестник современных исследований. – 2017. – №. 10-1. – С. 75-77.
9. Рыбалко М. А., Иванова Е. А. Тестирование программного обеспечения, методы тестирования // Информационное общество: современное состояние и перспективы развития. – 2017. – С. 320-322.
10. Gundecha U., Avasarala S., Selenium WebDriver 3 Practical Guide - Second Edition // Packt Publishing Ltd, 2018.
11. Пугачева Е. А. Методы и технологии тестирования // Актуальные вопросы современной науки. – 2018. – С. 23-28.

Приложение А

(обязательное)

Система автоматизированного тестирования личного кабинета студента

UML-диаграммы

					ПГУ 1.090301.20.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

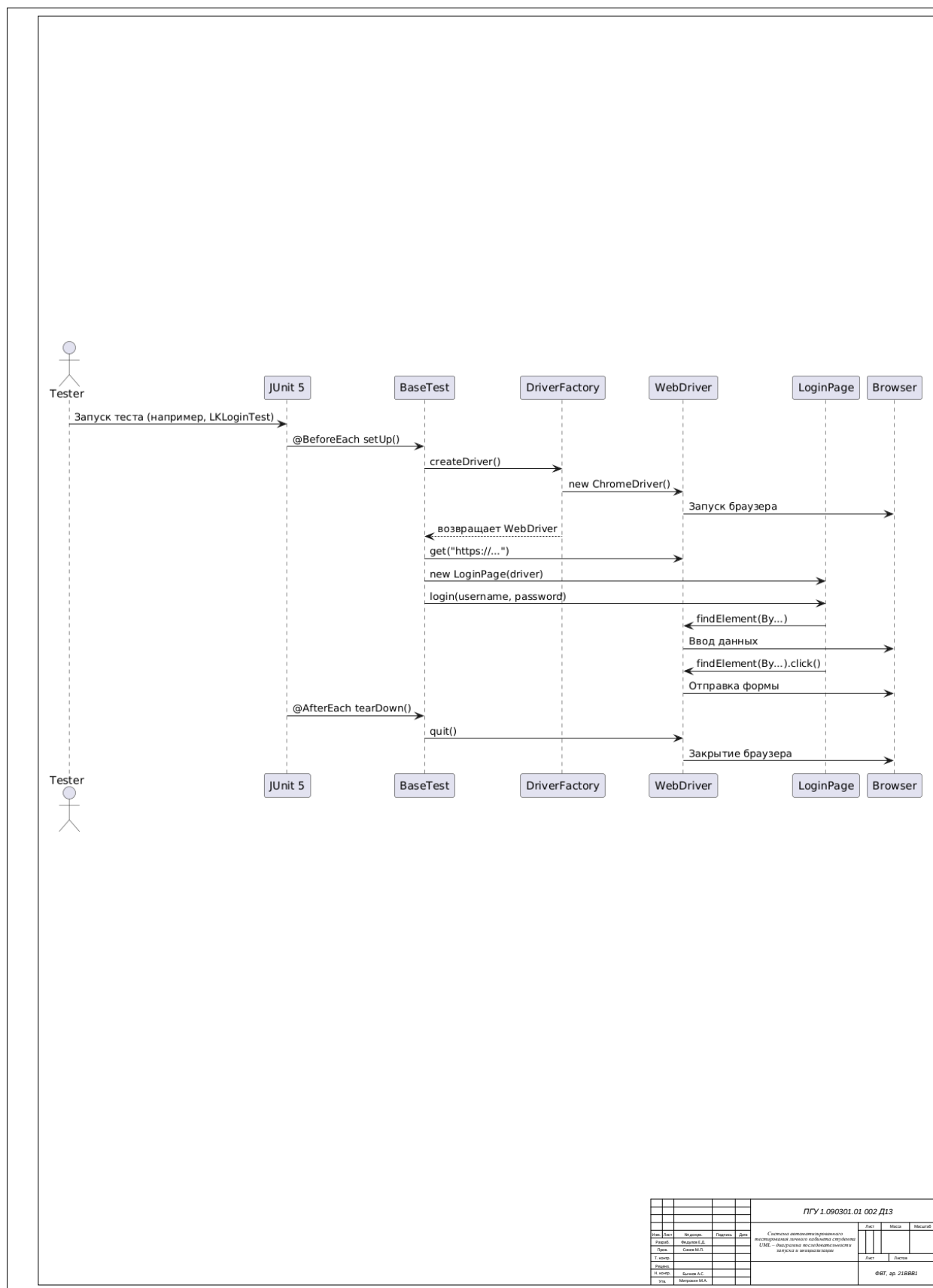
Приложение А.1 - Диаграмма вариантов использования



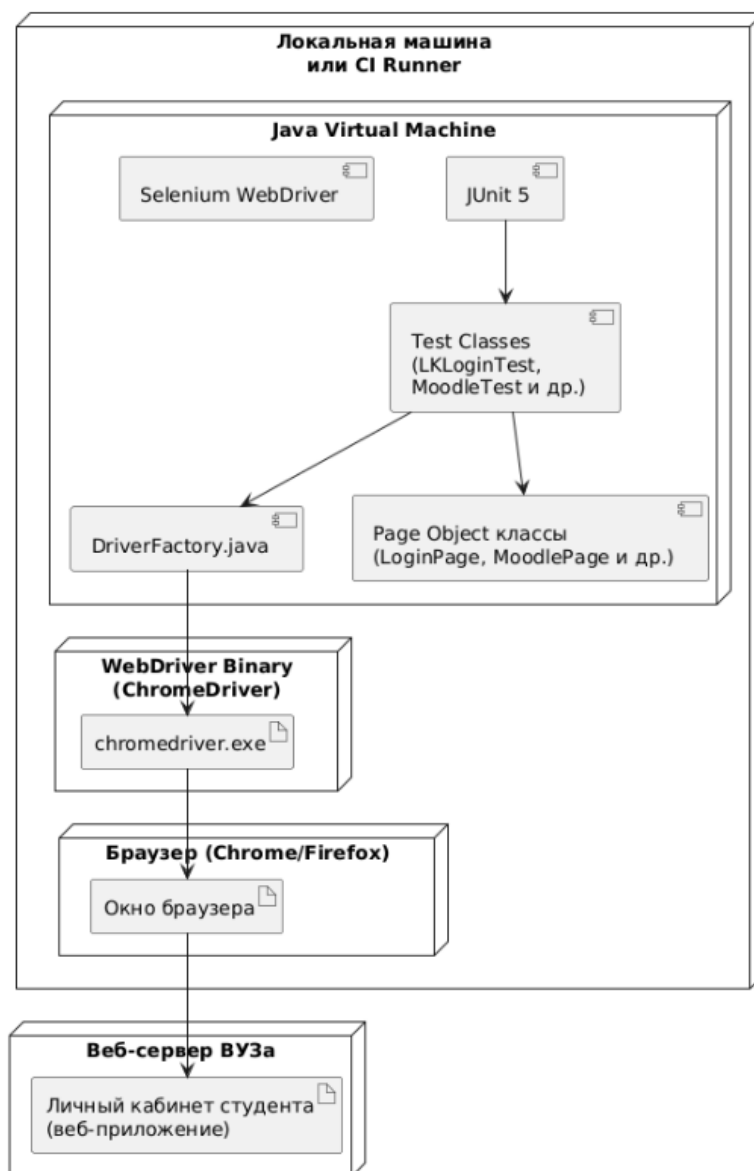
					ПГУ 1.090301.20.002 Д13				
Изм.	Лист	№ докум.	Подпись	Дата	Система автоматизированного тестирования личного кабинета студента (СМТ) – интерфейс административного использования				
Разработчик		Будунов Е.Д.							
Проверен		Савин М.П.							
Утвержден									
Разработчик					Лист		Листа	ФЭТ, зр. 218881	
И. инт.		Буднов А.С.							
И. инт.		Мельников М.А.							

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.2 - Диаграмма последовательности запуска и инициализации



Приложение А.3 - Диаграмма развертывания



					ПГУ 1.090301.20.002 Д19		
Имя	Лист	Инициалы	Пол	Дата	Система автоматизированного тестирования личного кабинета студента ПГУ - Диаграмма развертывания		
Разраб.	Иванов И.И.	Провер.	Иванов И.И.	Дата	Лист	Место	Масштаб
Тест.	Сидорова С.С.	Дата	Иванов И.И.	Дата	Лист	Место	Масштаб
Имя	Лист	Инициалы	Пол	Дата	ФБТ, стр. 21/8881		

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.20.001 ПЗ

Лист

51

Приложение Б

(обязательное)

Система автоматизированного тестирования личного кабинета студента

Листинг программы

					ПГУ 1.090301.20.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

```

package ru.mycompany.tests.pages;

import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;

/**
 * Общий абстрактный класс для Page Object.
 * Хранит WebDriver, WebDriverWait и инициализирует @FindBy-
элементы.
 */
public abstract class BasePage {
    protected WebDriver driver;
    protected WebDriverWait wait;

    public BasePage(WebDriver driver) {
        this.driver = driver;
        // Таймаут 10 сек. — можно вынести в проперти, если
нужно
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        PageFactory.initElements(driver, this);
    }

    /** Ждём, пока document.readyState == "complete" */
    protected void waitForPageLoaded() {
        wait.until(webDriver ->
            ((String)((JavascriptExecutor) webDriver)
                .executeScript("return
document.readyState")))
            .equals("complete")
        );
    }
}

```

```

        /** Открыть URL и дождаться загрузки страницы */
        public void openUrl(String url) {
            driver.get(url);
            waitForPageLoaded();
        }
    }

package ru.mycompany.tests.pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.support.FindBy;

import java.time.Duration;

/**
 * Page Object для страницы авторизации на https://lk.pnzgu.ru/
 */
public class LoginPage extends BasePage {

    public LoginPage(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, this);
    }

    /** Открывает страницу логина по URL и дожидается загрузки
 */
    public void open() {

```

```

        openUrl("https://lk.pnzgu.ru/");
    }

    // 1) Поле ввода логина
    @FindBy(css = "#formcardcontent input[placeholder='Логин']")
    private WebElement loginInput;

    // 2) Поле ввода пароля
    @FindBy(css = "#formcardcontent input[placeholder='Пароль']")
    private WebElement passwordInput;

    // 3) Кнопка "Войти"
    @FindBy(css = ".d-flex.justify-content-center input[type='submit']")
    private WebElement submitButton;

    // 4) Локатор для текста ошибки (если отображается)
    //      Находим любой <p>, начинающийся с "Ошибка входа",
    //      внутри div с классом login-form
    private By errorMessageLocator =
        By.xpath("//div[contains(@class,'login-form')] and
        contains(@class,'card') and contains(@class,'p-3')]//p[starts-
        with(normalize-space(.),'Ошибка входа')]");

    // 5) Локатор для приветственного заголовка после успешного
    //      входа
    private By welcomeHeader =
        By.xpath("//h1[b[starts-with(normalize-space(.),'Вы вошли
        как')]]");

    // 6) Локатор для ссылки на профиль (имя пользователя) в шапке
    //      после успешного входа
    @FindBy(css = ".header-menu-lk a[href='/portfolio/my']")
    private WebElement userProfileLink;

```

```

        /** Вводим логин */
public LoginPage enterLogin(String login) {
    loginInput.clear();
    loginInput.sendKeys(login);
    return this;
}

        /** Вводим пароль */
public LoginPage enterPassword(String password) {
    passwordInput.clear();
    passwordInput.sendKeys(password);
    return this;
}

        /** Нажимаем кнопку "Войти" и ждём либо приветствие, либо
ошибку */
public void clickSubmit() {
    submitButton.click();

    // Ждём, что либо появится блок с приветствием, либо блок с
ошибкой, максимум 5 секунд
    WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(5));
    try {
        wait.until(ExpectedConditions.or(
ExpectedConditions.visibilityOfElementLocated(welcomeHeader),
ExpectedConditions.visibilityOfElementLocated(errorMessageLocator)
));
    } catch (TimeoutException e) {
        // Если ни приветствие, ни сообщение об ошибке не
появилось за 5 секунды,
        // просто выходим – вряд ли это корректная
ситуация, но оставим на усмотрение.

```

```

        }
    }

    /**
     * Выполняет полный сценарий входа: заполняет поля и
     * кликает "Войти".
     * @return true, если увидели приветствие "Вы вошли как ...";
     * false, если вместо этого появилось сообщение об ошибке (или вообще
     * ничего не нашлось).
     */
    public boolean loginAs(String login, String password) {
        enterLogin(login);
        enterPassword(password);
        clickSubmit();

        // После clickSubmit ждём достижения одного из условий, так
        // что можно просто проверить:
        if (isLoggedIn()) {
            return true;
        }
        return isErrorMessage();
    }

    /** Проверяет, что пользователь успешно вошёл (виден блок "Вы
    вошли как ...") */
    public boolean isLoggedIn() {
        try {
            return driver.findElement(welcomeHeader).isDisplayed();
        } catch (NoSuchElementException e) {
            return false;
        }
    }

    /** Проверяет, что на странице виден блок с ошибкой */
    public boolean isErrorMessage() {

```

```

try {
    WebElement err = driver.findElement(errorMessageLocator);
    return err.isDisplayed();
        } catch (NoSuchElementException e) {
    return false;
        }
    }

    /** Проверка, что мы всё ещё на странице логина (URL
содержит "lk.pnzgu.ru") */
    public boolean isAtLoginPage() {
        return driver.getCurrentUrl().contains("lk.pnzgu.ru");
    }

    public PortfolioPage clickUserProfile() {
        new WebDriverWait(driver, Duration.ofSeconds(5))

.until(ExpectedConditions.elementToBeClickable(userProfileLink));
        userProfileLink.click();
        waitForPageLoaded();
        return new PortfolioPage(driver);
    }
}

package ru.mycompany.tests.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

```

```

/**
 * Page Object для Moodle (учебного процесса).
 * Предполагаем, что сюда мы попадём из портфолио через переход
по ссылке
 * с target="_blank", поэтому перед созданием этого Page Object
 * тест обязан переключить WebDriver на правильную вкладку.
 */
public class MoodlePage {

    private final WebDriver driver;
    private final WebDriverWait wait;

    public MoodlePage(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(5));
        PageFactory.initElements(driver, this);
    }

    @FindBy(css = "span.usertext")
    private WebElement userTextSpan;

    /**
     * Простейшая проверка: если URL содержит
    "moodle.pnzgu.ru",
     * значит мы действительно на странице Moodle.
     */
    public boolean isAtMoodle() {
        return driver.getCurrentUrl().contains("moodle.pnzgu.ru");
    }

    /**
     * Ждёт, пока элемент <span class="usertext">...</span>
    станет ВИДИМЫМ,
     * и возвращает его текст (ФИО).
     */

```

```

        public String getUserFullNameFromMoodle() {
            wait.until(ExpectedConditions.visibilityOf(userTextSpan));
            return userTextSpan.getText().trim();
        }
    }

package ru.mycompany.tests.pages;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;

/**
 * Page Object для страницы "Портфолио" (URL заканчивается на
"/portfolio/my").
 */
public class PortfolioPage extends BasePage {

    public PortfolioPage(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, this);
        // Убедимся, что страница загрузилась (например,
дождемся заголовка)
        waitForHeaderVisible();
    }

```

```

        /** Локатор для заголовка "Портфолио. Егор Дмитриевич
Федулов" */
        @FindBy(css = "h1.my-2")
        private WebElement portfolioHeader;

        public String getStudentFullName() {
            // Ждём, пока заголовок станет видимым
            wait.until(ExpectedConditions.visibilityOf(portfolioHeader));
            String fullText = portfolioHeader.getText().trim();
            // fullText == "Портфолио. Егор Дмитриевич Федулов"
            // Обрежем всё до и включая "Портфолио."
            String prefix = "Портфолио.";
            if (fullText.startsWith(prefix)) {
                // Обрезаем по первой точке
                String namePart = fullText.substring(prefix.length()).trim();
                return namePart; // "Егор Дмитриевич Федулов"
            } else {
                // Если вдруг формат поменялся, просто возвращаем
                весь текст
                return fullText;
            }
        }

        // 2) Блок-«аккордеон» «Студенческая успеваемость ...»
        @FindBy(css = "div.alert.alert-success.my-2.p-2")
        private WebElement gradesAccordion;

        // 3) Сама таблица (tbody), которая разворачивается внутри
        <tbody id="eMarkCollapse231319">
        // Мы не будем захардкодить весь xpath целиком, а
        сформируем его динамически, либо можно прописать конкретный @FindBy:
        @FindBy(id = "eMarkCollapse231319")
        private WebElement gradesTableBody;

        /**

```

```

        * Ссылка «Учебный процесс», открывающая Moodle в новой
вкладке:

        *
        <a                                class="header-nav-menu-link"
href="https://moodle.pnzgu.ru/my/"        target="_blank">Учебный
процесс</a>

        */

        @FindBy(css                        =                        "a.header-nav-menu-
link[href='https://moodle.pnzgu.ru/my/']")
        private WebElement moodleLink;

        /**
        * Ждёт, пока заголовок портфолио не станет видимым на
экране.
        */
        private void waitForHeaderVisible() {
            new                org.openqa.selenium.support.ui.WebDriverWait(driver,
java.time.Duration.ofSeconds(5))

.until(org.openqa.selenium.support.ui.ExpectedConditions.visibilityO
f(portfolioHeader));
        }

        /**
        * Проверяет, что заголовок существует и отображается.
        */
        public boolean isHeaderDisplayed() {
            try {
                return portfolioHeader.isDisplayed();
            } catch (Exception e) {
                return false;
            }
        }

        /**

```

```

        * Возвращает текст заголовка (должно быть что-то вроде
"Портфолио. Егор Дмитриевич Федулов").
        */
    public String getHeaderText() {
        return portfolioHeader.getText().trim();
    }

    /**
        * Кликает по блоку «Студенческая успеваемость ...» и
дожидается,
        * когда появится <tbody id="eMarkCollapse231319"> и в нём
загрузятся строки.
        */
    public void openGradesAccordion() {
        // 1) Прокручиваем элемент в видимую область
        ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);",
gradesAccordion);

        // 2) Ждём, пока блок станет кликабельным (не будет перекрыт)

wait.until(ExpectedConditions.elementToBeClickable(gradesAccordion))
;

        // 3) Пытаемся кликнуть. Если «click intercepted»,
попробуем JavaScript-клик.
        try {
            gradesAccordion.click();
        }
        catch
(org.openqa.selenium.ElementClickInterceptedException ex) {
            // Fallback: клик через JS
            ((JavascriptExecutor)
driver).executeScript("arguments[0].click();", gradesAccordion);
        }
    }

```

```

        // 4) Ждём, пока сам <tbody id="eMarkCollapse231319">
станет ВИДИМЫМ
        wait.until(ExpectedConditions.visibilityOf(gradesTableBody));

        // 5) Дополнительно ждём, что хоть один <tr> внутри <tbody>
уже появился
        wait.until(ExpectedConditions.presenceOfElementLocated(
            By.xpath("//tbody[@id='eMarkCollapse231319']/tr")
        ));
    }

    /**
     * Проверяет, что в развернутой таблице «Студенческая
успеваемость»
     * присутствует строка с текстом «осенний (1)» в
соответствующем <td>.
     */
    public boolean isAutumnTermPresent() {
        // Будем искать элемент по следующему ХРАТН:
        //
//tbody[@id='eMarkCollapse231319']/td[b[contains(text(),'осенний
(1)')] ]
        try {
            WebElement autumnCell = driver.findElement(

By.xpath("//tbody[@id='eMarkCollapse231319']/td[b[contains(normaliz
e-space(),'осенний (1)')] ]")
        );
        return autumnCell.isDisplayed();
    } catch (Exception e) {
        return false;
    }
}

public void clickMoodleLink() {

```

```

        // Ждём, пока элемент станет кликабельным

wait.until(ExpectedConditions.elementToBeClickable(moodleLink));
    moodleLink.click();
    }

}

package ru.mycompany.tests;

import org.junit.jupiter.api.*;
import org.openqa.selenium.WebDriver;
import ru.mycompany.tests.pages.LoginPage;
import ru.mycompany.tests.utils.DriverFactory;

import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assumptions.*;

@TestInstance(TestInstance.Lifecycle.PER_CLASS)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class LKLoginTest {

    private WebDriver driver;
    private LoginPage loginPage;

    @BeforeAll
    void setupClass() {
        // Здесь можно задать системные свойства, если нужно:
        // System.setProperty("browser", "chrome");
    }

    @BeforeEach
    void setupTest() {
        driver = DriverFactory.getDriver();
        loginPage = new LoginPage(driver);
    }

```

```

loginPage.open(); // Открываем https://lk.pnzgu.ru/
}

    @AfterEach
void teardownTest() {
    DriverFactory.quitDriver();
}

@Test
    @Order(1)
    @DisplayName("Провальный вход с некорректными данными")
void testLoginWithInvalidCredentials() {
    String invalidLogin = "wronglogintest";
    String wrongPassword = "wrongpasstest";

    loginPage.loginAs(invalidLogin, wrongPassword);

    assertTrue(loginPage.isErrorMessage(), "Ожидаем, что появится
сообщение об ошибке.");
    assertTrue(loginPage.isAtLoginPage(), "Должны остаться на
странице логина.");
}

@Test
    @Order(2)
    @DisplayName("Успешный вход с корректными данными")
void testSuccessfulLogin() {
    // TODO: замените на настоящий логин/пароль студента,
доступный в тестовой среде
    String validLogin = "a517711555";
    String validPassword = "102030qwelQq@";

    loginPage.loginAs(validLogin, validPassword);

    assertTrue(loginPage.isLoggedIn(),

```

```

        "Ожидаем, что после успешного входа появится текст «Вы вошли
как ...», но он не появился.");
    }
}

package ru.mycompany.tests;

import org.junit.jupiter.api.*;
import org.openqa.selenium.WebDriver;
import ru.mycompany.tests.pages.LoginPage;
import ru.mycompany.tests.pages.PortfolioPage;
import ru.mycompany.tests.pages.MoodlePage;
import ru.mycompany.tests.utils.DriverFactory;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.ExpectedCondition;

import java.time.Duration;
import java.util.Set;

import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assumptions.*;

/**
 * Тест, который логинится в ЛК, переходит в портфолио и оттуда
– на Moodle.
 */
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class MoodleTest {

    private WebDriver driver;
    private LoginPage loginPage;
    private PortfolioPage portfolioPage;

    // TODO: подставьте реальные тестовые креды

```

```

private static final String VALID_LOGIN = "a517711555";
private static final String VALID_PASSWORD = "102030qwe1Qq@";

@BeforeEach
void setup() {
    driver = DriverFactory.getDriver();
    loginPage = new LoginPage(driver);
    loginPage.open(); // открываем lk.pnzgu.ru
}

@AfterEach
void teardown() {
    DriverFactory.quitDriver();
}

@Test
@DisplayName("Переход на Moodle и проверка входа")
void testNavigateToMoodle() {
    // 1) Авторизуемся
    boolean loggedIn = loginPage.loginAs(VALID_LOGIN,
VALID_PASSWORD);
    assertTrue(loggedIn, "Не удалось залогиниться – пропускаем
тест.");

    // 2) Переходим в портфолио, кликая по имени
пользователя.
    // Предполагаем, что в LoginPage есть метод
clickUserProfile(),
    // возвращающий объект PortfolioPage:
    portfolioPage = loginPage.clickUserProfile();

    // Убедимся, что мы действительно оказались в
портфолио:
    assertTrue(portfolioPage.isHeaderDisplayed(),
"Ожидается заголовок «Портфолио», но он не виден.");

```

```

String portfolioFullName = portfolioPage.getStudentFullName();

// 3) Сохраняем дескриптор текущей вкладки
String originalWindow = driver.getWindowHandle();

// 4) Кликаем на ссылку «Учебный процесс» (открывается
новая вкладка)
portfolioPage.clickMoodleLink();

// 5) Ждём, пока появится новая вкладка (количество
окон станет > 1)
WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));
wait.until((ExpectedCondition<Boolean>) d ->
d.getWindowHandles().size() > 1);

// 6) Переключаемся на вкладку, отличную от originalWindow
Set<String> allWindows = driver.getWindowHandles();
for (String handle : allWindows) {
    if (!handle.equals(originalWindow)) {
        driver.switchTo().window(handle);
        break;
    }
}

// 7) Создаём MoodlePage (мы уже в новой вкладке с
Moodle)
MoodlePage moodlePage = new MoodlePage(driver);

// 8) Проверяем, что URL действительно ведёт на
moodle.pnzgu.ru
assertTrue(moodlePage.isAtMoodle(),
"Ожидаем, что текущий URL содержит
'moodle.pnzgu.ru', но это не так: "
+ driver.getCurrentUrl());

```

```

        String                moodleFullName                =
moodlePage.getUserFullNameFromMoodle();

        // Сравниваем строки
        Assertions.assertEquals(portfolioFullName, moodleFullName,
        "Ожидаем, что ФИО в Portfolio и Moodle совпадают, " +
        "но получили: Portfolio='" + portfolioFullName +
        "'; Moodle='" + moodleFullName + "'");

        // 9) (По желанию) Закрываем вкладку Moodle и возвращаемся в
основное окно
        driver.close();
        driver.switchTo().window(originalWindow);
    }
}

package ru.mycompany.tests;

import org.junit.jupiter.api.*;
import org.openqa.selenium.WebDriver;
import ru.mycompany.tests.pages.LoginPage;
import ru.mycompany.tests.pages.PortfolioPage;
import ru.mycompany.tests.utils.DriverFactory;

import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assumptions.assumeTrue;

@TestInstance(TestInstance.Lifecycle.PER_CLASS)
public class PortfolioTest {

    private WebDriver driver;
    private LoginPage loginPage;
    private static final String VALID_LOGIN = "a517711555";
    // TODO: заменить на реальные тестовые данные

```

```

        private static final String VALID_PASSWORD = "102030qwe1Qq@";
// TODO: заменить на реальные тестовые данные

        @BeforeEach
        void setup() {
            driver = DriverFactory.getDriver();
            loginPage = new LoginPage(driver);
            loginPage.open();
        }

        @AfterEach
        void teardown() {
            DriverFactory.quitDriver();
        }

        @Test
        @DisplayName("Проверка подгрузки портфолио")
        void testPortfolioHeaderVisible() {
            // 1. Залогиниться
            boolean loginSucceeded = loginPage.loginAs(VALID_LOGIN,
VALID_PASSWORD);
            assertTrue(loginSucceeded, "Пропускаем тест: не удалось
залогиниться.");

            // 2. Перейти в портфолио
            PortfolioPage portfolioPage = loginPage.clickUserProfile();

            // 3. Убедиться, что заголовок портфолио отображается
            assertTrue(portfolioPage.isHeaderDisplayed(),
"Ожидали увидеть заголовок «Портфолио», но он не появился.");

            // 4. Проверить, что текст заголовка начинается с
«Портфолио»
            String headerText = portfolioPage.getHeaderText();
            assertTrue(headerText.startsWith("Портфолио"),

```

"Заголовок портфолио должен начинаться с «Портфолио», но он:
«" + headerText + "»");

// 3. Кликаем на «Студенческая успеваемость ...»
portfolioPage.openGradesAccordion();

// 4. Проверяем, что в раскрывшейся таблице есть
«осенний (1)»

assertTrue(portfolioPage.isAutumnTermPresent(),
"Ожидаем, что в разделе «Студенческая
успеваемость» появится «осенний (1)»");

}

}

Приложение В

(обязательное)

Система автоматизированного тестирования личного кабинета студента

Презентация

					ПГУ 1.090301.20.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

ТЕМА ВКР: «СИСТЕМА АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ЛИЧНОГО КАБИНЕТА СТУДЕНТА»

Автор работы: студент группы 21ВВВ1 Федулов Е.Д.
Научный руководитель: Синев М.П.

Пенза, 2025

Цель и задачи работы

Цель работы – разработка системы для реализации автоматизированных функциональных проверок модуля личного кабинета студента из состава информационной системы высшего учебного заведения

Задачи:

1. Анализ предметной области.
2. Использование автоматизированного тестирования.
3. Выбор инструментов и решения.
4. Проектирование и разработка автоматизированных тестов.
5. Анализ результатов автоматизированного тестирования.

Актуальность работы



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.20.001 ПЗ

Лист

74

Аналоги



Cypress



Katalon Studio



Robot Framework

Выбор среды разработки и инструментов



IntelliJ IDEA

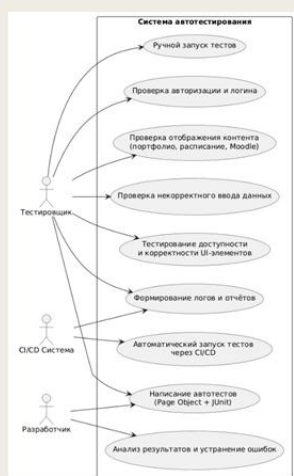


Java JDK

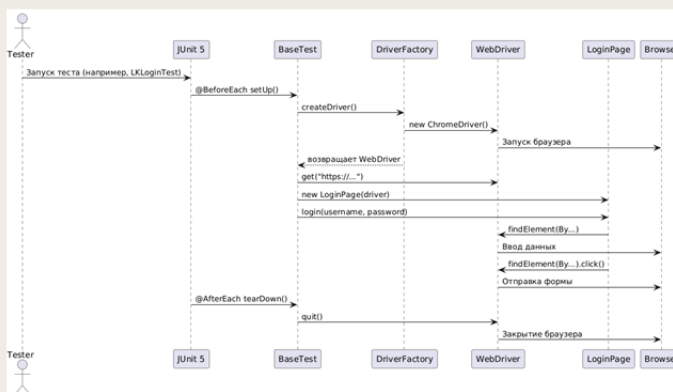


Selenium

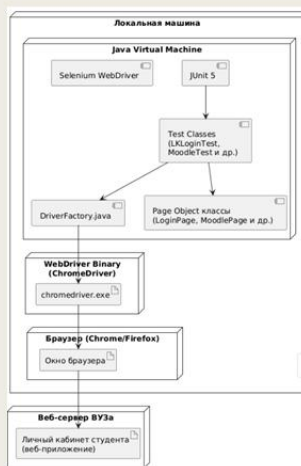
UML - диаграмма вариантов использования



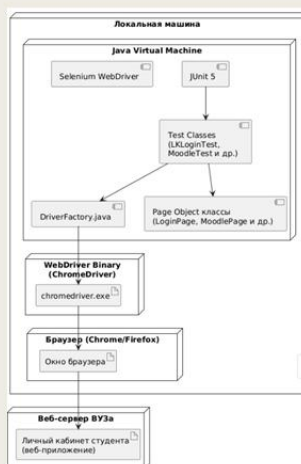
UML - диаграмма последовательности запуска и инициализации



UML - диаграмма развертывания



UML - диаграмма развертывания



Заключение

В ВКР была успешно разработана система автоматизированного тестирования личного кабинета студента

- *Успешно реализованы основные проверки входа в личный кабинет студента*
- *Успешно реализованы основные проверки веб-интерфейса сайта с личным кабинетом*
- *Подтверждена стабильность и работоспособность автотестов в ходе множественных тестирований*

Разработанное решение является гибким и экономически эффективным инструментом для модернизации личного кабинета студента.

Спасибо за внимание!

					ПГУ 1.090301.20.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		