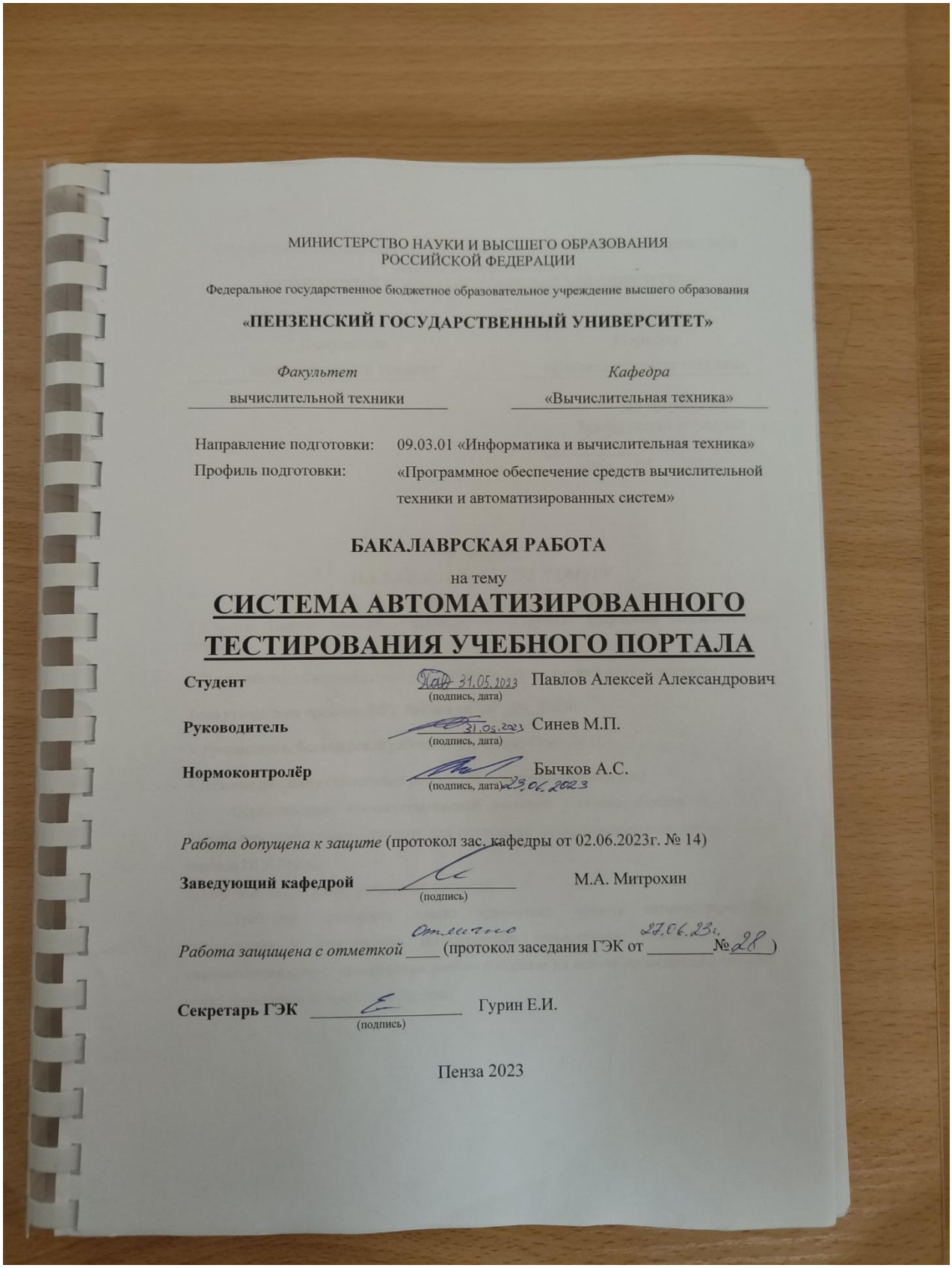


МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СИСТЕМА АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ УЧЕБНОГО ПОРТАЛА

Студент 31.05.2023 (подпись, дата) Павлов Алексей Александрович

Руководитель 31.05.2023 (подпись, дата) Синев М.П.

Нормоконтролёр (подпись, дата) 23.06.2023 Бычков А.С.

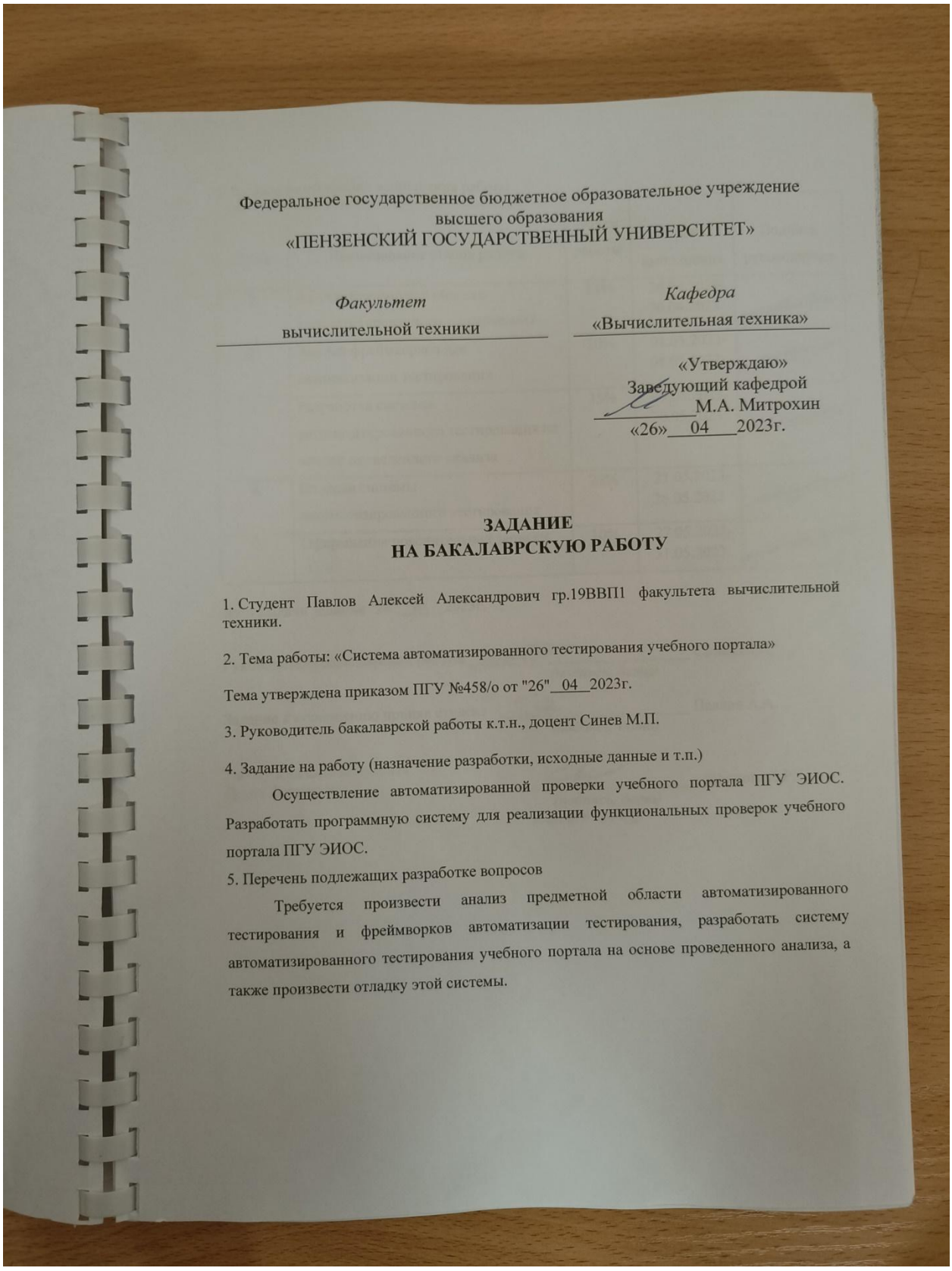
Работа допущена к защите (протокол зас. кафедры от 02.06.2023г. № 14)

Заведующий кафедрой _____ (подпись) М.А. Митрохин

Работа защищена с отметкой отлично (протокол заседания ГЭК от 27.06.23г. №28)

Секретарь ГЭК _____ (подпись) Гурин Е.И.

Пенза 2023



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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«26»__04___2023г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Павлов Алексей Александрович гр.19ВВП1 факультета вычислительной техники.

2. Тема работы: «Система автоматизированного тестирования учебного портала»

Тема утверждена приказом ПГУ №458/о от "26"__04__2023г.

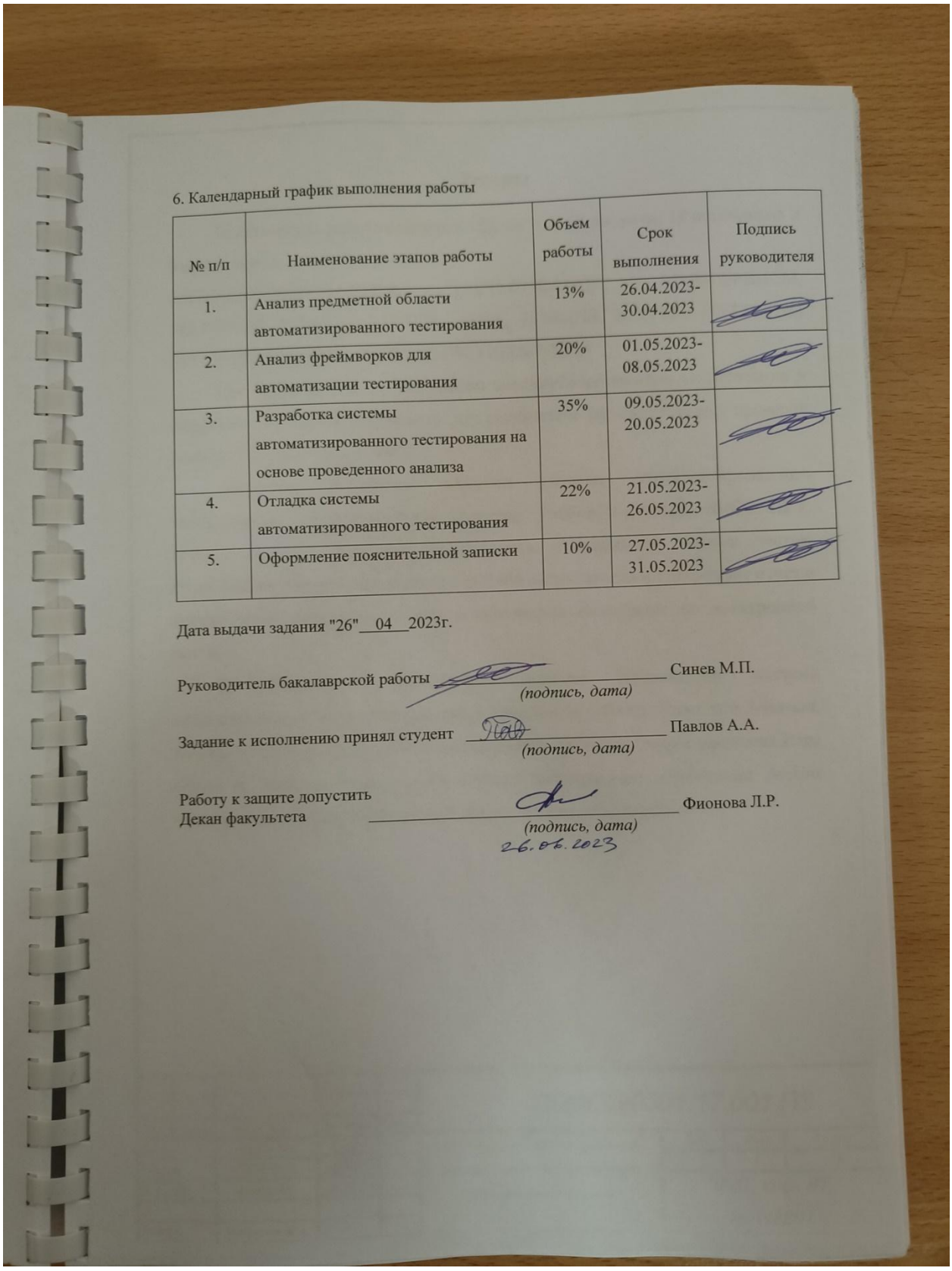
3. Руководитель бакалаврской работы к.т.н., доцент Синев М.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Осуществление автоматизированной проверки учебного портала ПГУ ЭИОС. Разработать программную систему для реализации функциональных проверок учебного портала ПГУ ЭИОС.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области автоматизированного тестирования и фреймворков автоматизации тестирования, разработать систему автоматизированного тестирования учебного портала на основе проведенного анализа, а также произвести отладку этой системы.



6. Календарный график выполнения работы

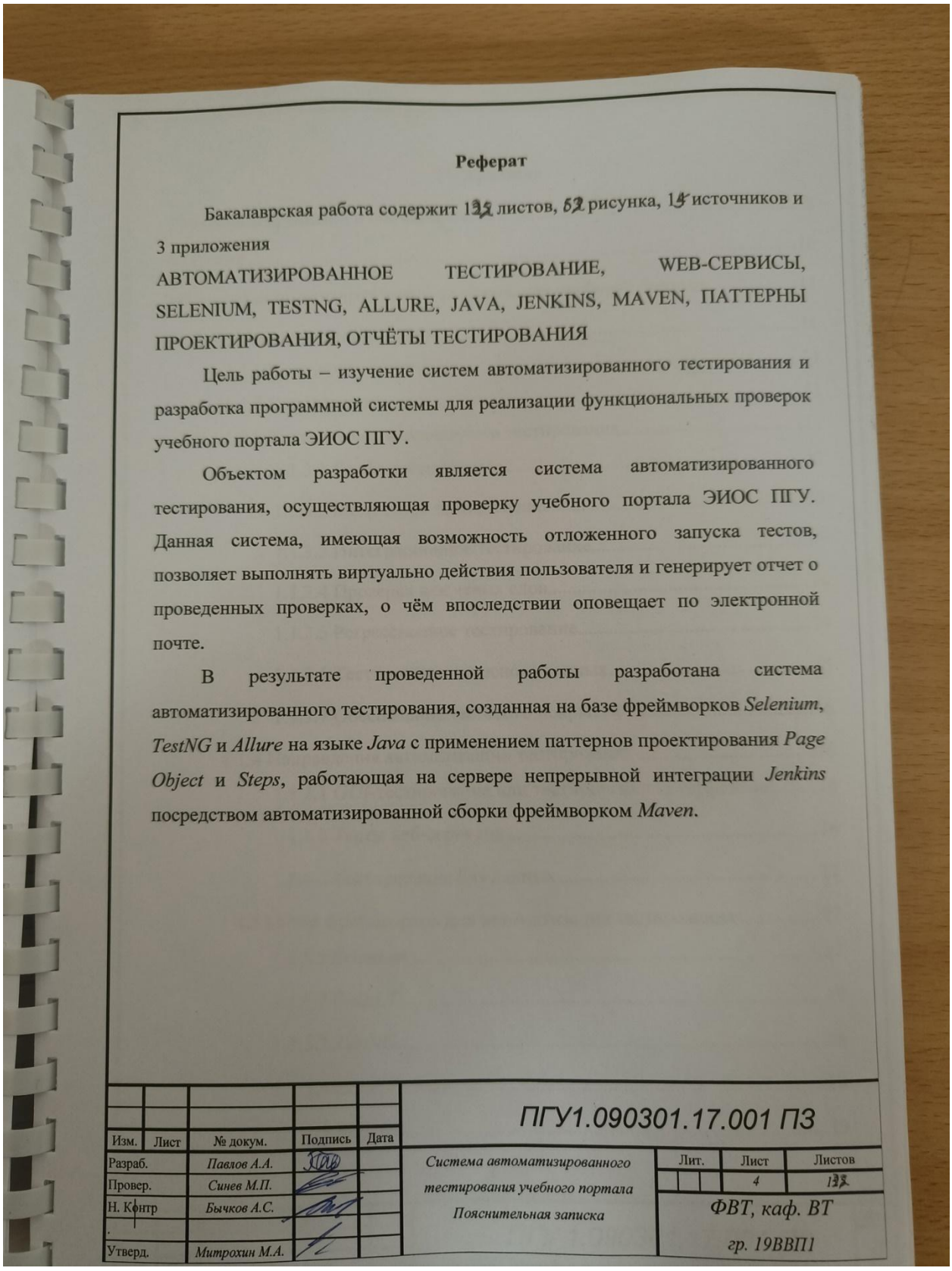
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области автоматизированного тестирования	13%	26.04.2023-30.04.2023	
2.	Анализ фреймворков для автоматизации тестирования	20%	01.05.2023-08.05.2023	
3.	Разработка системы автоматизированного тестирования на основе проведенного анализа	35%	09.05.2023-20.05.2023	
4.	Отладка системы автоматизированного тестирования	22%	21.05.2023-26.05.2023	
5.	Оформление пояснительной записки	10%	27.05.2023-31.05.2023	

Дата выдачи задания "26" 04 2023г.

Руководитель бакалаврской работы _____ Синев М.П.
(подпись, дата)

Задание к исполнению принял студент _____ Павлов А.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)
26.06.2023



Реферат

Бакалаврская работа содержит 132 листов, 52 рисунка, 14 источников и 3 приложения

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ, WEB-СЕРВИСЫ, SELENIUM, TESTNG, ALLURE, JAVA, JENKINS, MAVEN, ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ, ОТЧЁТЫ ТЕСТИРОВАНИЯ

Цель работы – изучение систем автоматизированного тестирования и разработка программной системы для реализации функциональных проверок учебного портала ЭИОС ПГУ.

Объектом разработки является система автоматизированного тестирования, осуществляющая проверку учебного портала ЭИОС ПГУ. Данная система, имеющая возможность отложенного запуска тестов, позволяет выполнять виртуально действия пользователя и генерирует отчет о проведенных проверках, о чём впоследствии оповещает по электронной почте.

В результате проведенной работы разработана система автоматизированного тестирования, созданная на базе фреймворков *Selenium*, *TestNG* и *Allure* на языке *Java* с применением паттернов проектирования *Page Object* и *Steps*, работающая на сервере непрерывной интеграции *Jenkins* посредством автоматизированной сборки фреймворком *Maven*.

Изм.	Лист	№ докум.	Подпись	Дата	ПГУ1.090301.17.001 ПЗ		
Разраб.		Павлов А.А.			Система автоматизированного тестирования учебного портала Пояснительная записка	Лит.	Лист 4 / Листов 132
Провер.		Синев М.П.					
Н. Контр		Бычков А.С.				ФВТ, каф. ВТ гр. 19ВВП1	
Утверд.		Митрохин М.А.					

Содержание

Перечень сокращений и обозначений 8

Введение 10

1 Система автоматизированного тестирования учебного портала 11

1.1 Анализ предметной области 11

1.1.1 Постановка задачи 11

1.1.2 Определение 11

1.1.3 Типы автоматизированного тестирования 13

1.1.3.1 *Smoke*-тестирование 13

1.1.3.2 Модульное тестирование 13

1.1.3.3 Интеграционное тестирование 13

1.1.3.4 Проверка ключевых слов 14

1.1.3.5 Регрессионное тестирование 14

1.1.3.6 Тестирование на основе данных 15

1.1.3.7 Обеспечение тестового покрытия 16

1.1.4 Направления автоматизации тестирования 16

1.1.4.1 *GUI*-тестирование или тестирование интерфейсов 16

1.1.4.2 Тесты веб-сервисов 16

1.1.4.3 Тестирование баз данных 17

1.1.5 Обзор фреймворков для автоматизации тестирования 17

1.1.5.1 *Selenium* 17

1.1.5.2 *SoapUI* 17

1.1.5.3 *TestNG* 18

1.1.5.4 *Katalon Studio* 18

1.1.5.5 *Appium* 19

1.1.6 Объект тестирования 19
1.1.7 Описание работы тестируемого объекта 19
1.1.7.1 Авторизация и информация в личном кабинете 20
1.1.7.2 Просмотр промежуточной аттестации 22
1.1.8 Выводы 24
1.2 Проектирование тестового окружения 25
1.2.1 Инструменты разработки 25
1.2.1.1 *IntelliJ IDEA* 25
1.2.1.2 *JDK* 28
1.2.1.3 *Maven* 29
1.2.1.4 *ChromeDriver* 33
1.2.1.5 *Jenkins* 37
1.2.2 Реализация бизнес-логики и функциональности автотеста посредством реализации структуры проекта с использованием паттернов проектирования 40
1.2.2.1 Основные разделы проекта 40
1.2.2.2 Использование паттерна *PageObject* 43
1.2.2.3 Использование паттерна *Steps* 44
1.2.3 Сценарии функционального тестирования 47
1.2.3.1 Требования к тест-кейсам для автоматизации тестирования 47
1.2.3.2 Пример реализации сценария 48
1.2.4 Выводы 52
1.3 Результаты работы системы автоматизированного тестирования 53
1.3.1 Результаты выполнения и структура отчета 53
1.3.2 Выводы 56
2 Охрана труда 57

2.1 Загрязнение воздуха вредными веществами 57
2.2 Повышенный уровень шума на рабочем месте 58
2.3 Недостаточная освещенность рабочей зоны 58
2.4 Повышенная яркость света 60
Заключение 62
Список используемых источников 63
Приложение А 65
Приложение Б 70
Приложение В 116

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств

ПГУ – Пензенский государственный университет

ПО – программное обеспечение

СанПиН – санитарно-эпидемиологические правила и нормы

ФГОС ВО – федеральные государственные образовательные стандарты высшего образования

ХОБЛ – хроническая обструктивная болезнь легких

ЭБС - электронные библиотечные системы

ЭИОС – электронно-информационная образовательная среда

CI/CD (Continuous Integration, Continuous Delivery) – *DevOps* практика непрерывной интеграции и доставки

DevOps (Development & Operations) – это методология и набор практик, объединяющих разработку (*Development*) и операции (*Operations*) с целью улучшения сотрудничества, качества и скорости доставки программного обеспечения

EPA (U.S. Environmental Protection Agency) - агентство по охране окружающей среды США

.EXE (Executable) – исполнимый файл

HTML (HyperText Markup Language) — язык гипертекстовой разметки

iFrame (*inline frame*) – встроенный фрейм

JSON (JavaScript Object Notation) — это формат передачи данных, который используется при взаимодействии веб-сервера и браузера

*PM*2.5 *(Particulate Matter* 2.5*)* - взвешенные твердые микрочастицы и мельчайшие капельки жидкости (10 нм - 2,5 мкм в диаметре), содержащиеся в воздухе

POM (*Project Object Model*) – объектная модель проекта

SOAP (*Simple Object Access Protocol*) - простой протокол доступа к объектам

REST (*Representational state transfer*) – архитектурный стиль взаимодействия компонентов распределённого приложения в сети

UI (*User Interface*) – пользовательский интерфейс

UML (Unified Modeling Language) – унифицированный язык моделирования

UTF-8 (*Unicode Transformation Format*-8) – формат преобразования Юникода 8-бит

.*WAR (Web application ARchive)* – архив веб-приложений – файл архива, который используется для упаковки и развертывания веб-приложений в среде Java

XML (*eXtensible Markup Language*) – расширяемый язык разметки

XPath (XML Path Language) – язык запросов к элементам *XML*-документа

.*ZIP (Zone Improvement Plan)* – зональный план улучшения – формат архивации файлов и сжатия данных без потерь.

Введение

Актуальность исследования обусловлена стремительными изменениями во всех сферах общества, глобализацией системы образования и возрастающей потребности в информатизации образования, в частности в создании электронных образовательных ресурсов и массовом внедрении платформы «Электронной информационной образовательной среды (ЭИОС) ПГУ» в образовательный процесс вуза.

Для проверки работы информационных систем каждый разработанный модуль или компонент тестируется уже на этапе разработки программного обеспечения, также тестируется вся информационная система перед выпуском. Для этого, помимо программистов, существуют тестировщики, которые выполняют комплекс проверок как всей системы в целом, так и ее отдельных модулей с целью выявить и устранить ошибки в работе программной системы.

Одна из ключевых проверок включает в себя функциональное тестирование, то есть проверку соответствия выполнения задач, описанных в функциональных требованиях. При разработке сложных информационных систем существует множество функциональных требований с различным объёмом и видом задач. В таких случаях важно автоматизировать процесс тестирования. Глобальные тенденции в отрасли указывают на то, что автоматизация тестирования является в текущее время одним из наиболее заметных направлений в развитии информационных систем.

Интернет-сервисы образовательных организаций представляют собой обширные информационные системы, к которым предъявляется разнообразный набор требований. Тестирование этих сервисов вручную может быть трудоемко и дорогостояще. Поэтому автоматизация процесса тестирования является необходимой задачей.

1 Система автоматизированного тестирования учебного портала

1.1 Анализ предметной области

1.1.1 Постановка задачи

Необходимо разработать систему для автоматизированной проверки функционала электронно-информационно образовательной среды ФГБОУ ВО ПГУ.

Система автоматизированного тестирования должна быть реализована на языке программирования Java с использованием фреймворков *Selenium WebDriver*, *TestNG*.

Система должна запускаться с помощью инструмента для автоматизации сборки и развертывания ПО на сервере *Jenkins* с имеющимся функционалом отложенного запуска тестов и оповещений о результатах сборок по электронной почте.

Автотест должен работать на браузере *Chrome* операционной системы *Windows*.

В качестве результата система автоматизированного тестирования должна генерировать *Allure* отчёт с логом выполненных действий.

1.1.2 Определение

Тестирование программного обеспечения представляет собой процесс анализа программного средства и соответствующей документации с целью обнаружения дефектов и повышения качества программного продукта.

Функциональное тестирование является одним из видов тестирования, основная цель которого заключается в проверке правильной функциональности приложения, т.е. соответствия реализации функциональным требованиям. Часто функциональное тестирование ассоциируется с методом "черного ящика", однако с применением метода "белого ящика" также можно проверить корректность реализации

функционала.

Существуют три различных подхода к тестированию ПО: тестирование белого, серого и черного ящиков. Каждый подход имеет свои сильные и слабые стороны, поэтому необходимо комбинировать их использование для максимального качества тестирования.

Главной целью тестирования белого ящика является проверка кода, внутренней структуры и дизайна. Тестировщики имеют доступ к коду на данной стадии, что также называется открытым тестированием или проверкой кода. Эти термины указывают на прозрачный процесс и возможность проверки процессов ввода-вывода, основываясь на внутренних алгоритмах работы.

При поведенческом тестировании, также известном как тестирование "черного ящика", тестировщик осуществляет проверку программного продукта без полного знания его внутренней структуры, применения и дизайна. В этом подходе тесты обычно фокусируются на проверке функциональности. Тестировщик просматривает веб-страницы с использованием браузера, вводит определенные данные и анализирует полученные результаты.

Тестирование серого ящика сочетает в себе частичную осведомленность о внутренних процессах. Этот метод является комбинацией тестирования белого и черного ящиков, и направлен на обнаружение всех проблем функционирования и ошибок в коде.

Автоматизированное тестирование – это использование инструментальных средств для разработки и выполнения тестовых скриптов с целью удовлетворения требований к тестированию в минимальные сроки. Оно является важным механизмом проверки точности и стабильности программного продукта при выходе каждой новой версии. Автоматизированное тестирование позволяет повторно использовать ранее созданные тестовые скрипты, тем самым обеспечивая значительную отдачу и

экономию ресурсов.

1.1.3 Типы автоматизированного тестирования

1.1.3.1 *Smoke*-тестирование

Инженерное сообщество использовало понятие дымовое тестирование в контексте ввода в эксплуатацию нового оборудования: тестирование считалось успешным, если из установки не пошёл дым. В программной среде, дымовое тестирование – это краткий цикл тестирования, который выполняется для проверки того, что приложение, установленное после сборки кода, запускается и выполняет свои базовые функции [1].

Оценка работоспособности приложения производится на основании быстрого тестирования ключевых модулей приложения на предмет возможности выполнения задач, а также поиска критических и блокирующих дефектов. Если дефекты не выявлены, то smoke-тестирование считается успешно пройденным, и полный цикл тестирования может быть запущен, если же дефекты были выявлены, то приложение отправляется на доработку.

1.1.3.2 Модульное тестирование

Модульное (компонентное) тестирование (*unit testing, module testing, component testing*) предназначено для проверки отдельных небольших компонентов приложения, которые в обычных условиях могут быть протестированы изолированно от других подобных компонентов. Этот вид тестирования может включать в себя проверку отдельных функций или методов класса, взаимодействие между классами, небольшие библиотеки или отдельные части приложения. Часто модульное тестирование выполняется с использованием специализированных технологий и инструментов для автоматизации тестирования, которые значительно упрощают и ускоряют процесс разработки соответствующих тестовых сценариев.

1.1.3.3 Интеграционное тестирование

Интеграционное тестирование представляет собой специальный вид тестирования, который направлен на совместное групповое тестирование

программных модулей. Обычно разрабатываемый продукт состоит из нескольких программных модулей, созданных различными программистами. Эти модули подвергаются отдельному тестированию, а затем объединяются в один целостный программный продукт.

1.1.3.4 Проверка ключевых слов

Тестирование на основе ключевых слов (*Keyword-driven testing*) – это метод тестирования, в котором тест-дизайнеры создают набор ключевых слов, каждое из которых описывает определенное действие или событие. Эти ключевые слова создают "скрипты", которые затем используются для экспериментов в различных комбинациях, которые позволяют проводить функциональное тестирование на всех уровнях приложения.

Например, вы можете создать ключевые слова, такие как "открыть браузер", "ввести логин", "ввести пароль", "нажать кнопку", "проверить наличие элемента" и т.д. Затем эти ключевые слова можно использовать для создания конкретных тестовых сценариев, где входные данные для каждого шага являются значениями аргументов ключевого слова.

Этот метод позволяет быстро создавать тестирование именно на основе функциональных требований, а не на основе кода. Это упрощает тестирование и сокращает время, необходимое для разработки скриптов тестирования.

1.1.3.5 Регрессионное тестирование

Регрессионное тестирование – это процесс проверки уже протестированных функций продукта после внесения изменений или добавления нового функционала с целью обнаружения ошибок, которые могут возникнуть из-за этих изменений. Его цель – убедиться в том, что уже проверенные функции продукта продолжают работать корректно, несмотря на любые изменения и доработки. Регрессионное тестирование важно для обеспечения качества продукта и для минимизации риска возникновения ошибок после внесения изменений [2].

Источники возможных изменений, которые могут потребовать регрессионного тестирования, могут включать в себя:

– изменения в требованиях к продукту;

– исправления программных ошибок;

– добавление нового функционала;

– изменения в среде выполнения продукта (операционная система, браузер и т.д.);

– изменения в конфигурационных файлах.

Чтобы корректно провести регрессионное тестирование, необходимо провести тесты на всех уровнях приложения, включая модули, интеграцию, системное и пользовательское тестирование.

1.1.3.6 Тестирование на основе данных

Тестирование на основе данных (*Data-driven testing*) – это метод тестирования, в котором входные и ожидаемые значения тестов берутся из внешних источников данных, таких как таблицы баз данных, текстовые файлы, электронные таблицы и т.д. Значения, получаемые из этих источников, используются для создания скриптов тестирования.

С помощью этого метода можно быстро создавать большое количество тестов, используя различные комбинации входных данных. Это помогает проверить функционирование приложения с разными входными данными и обнаружить ошибки, которые могут возникнуть только при определенных значениях входных данных. Например, можно создать набор тестов для проверки валидности пароля, используя различные комбинации символов, длины и формата.

Тестирование на основе данных может проводиться как вручную, так и с помощью автоматизированных средств тестирования. В автоматизированной версии, скрипты тестирования многократно запускаются с разными входными данными и сравниваются ожидаемые и полученные результаты.

1.1.3.7 Обеспечение тестового покрытия

Обеспечение тестового покрытия – это процесс, направленный на проверку того, что все части программного продукта прошли достаточное тестирование. Он включает в себя анализ требований, разработку тест-кейсов, и определение количества тестов, необходимых для проверки каждой части продукта. Цель обеспечения тестового покрытия - убедиться, что все возможные пути и возможности, которые могут привести к ошибкам, были покрыты тестами. Тестирование должно быть полным и комплексным, с использованием различных методов, чтобы убедиться, что все части продукта были достаточно проверены и не содержат ошибок.

При этом важно учесть, что полное покрытие тестами не гарантирует отсутствие ошибок в продукте, однако уменьшает риски и повышает вероятность выявления проблем.

1.1.4 Направления автоматизации тестирования

1.1.4.1 *GUI*-тестирование или тестирование интерфейсов

Автоматизированное *GUI*-тестирование использует современные инструменты для проверки работы системы на соответствие техническому заданию через эмуляцию работы реальных пользователей и проверки интерфейса. Это необходимо для тестирования приложений, которые работают через графический интерфейс пользователя, таких как *web*-приложения, клиентские и банковские приложения, платежные системы и другие. Его цель - проверить работу всех элементов интерфейса и убедиться, что система соответствует заданным требованиям.

1.1.4.2 Тесты веб-сервисов

В основном все веб-сервисы применяются как промежуточное звено между клиентской частью приложения и данными заказчика. Веб-сервисы могут использовать сразу множество пользователей, поэтому надежность его работы зависит от доступности решений клиента, что позволяет выявить возможные дефекты работы веб-сервиса и проверить его реализацию на

соответствие отраслевым стандартам.

1.1.4.3 Тестирование баз данных

Тестирование баз данных – это одно из направлений тестирования программного обеспечения, нацеленное на проверку корректности и эффективности взаимодействия с БД. Оно позволяет убедиться в правильности работы БД и снизить риски при запуске системы в реальную эксплуатацию.

1.1.5 Обзор фреймворков для автоматизации тестирования

1.1.5.1 *Selenium*

Selenium – это инструмент для автоматизации тестирования веб-приложений. Данный фреймворк поддерживает различные языки программирования: *Java, Python, C#, Ruby* и тд, что позволяет разработчикам писать тесты на любом языке, с которым они знакомы. *Selenium* имеет функционал, который позволяет эмулировать действия пользователя на веб-странице, такие как клики по кнопкам, ввод текста в формы и другие. *Selenium* также может выполнять тесты в различных браузерах, будь то *Chrome, Firefox, Safari, IE*. Это мощный инструмент для автоматизации тестирования веб-приложений, который позволяет сократить время проверки качества и повысить надежность конечного продукта.

1.1.5.2 *SoapUI*

SoapUI – это инструмент для тестирования веб-сервисов, который позволяет разработчикам и тестировщикам проводить функциональное и нагрузочное тестирование веб-сервисов на основе протокола *SOAP* (*Simple Object Access Protocol*) и *REST* (*Representational State Transfer*)[2]. *SoapUI* предлагает возможность создавать, запускать и управлять тестами, а также генерировать отчеты. Он предоставляет широкий набор инструментов для автоматизации тестирования, включая возможность записи тестовых сценариев, создания и использования переменных и условий, расширяемые функции для создания собственных модулей и многое другое. *SoapUI*

поддерживает различные форматы данных, такие как *XML*, *JSON*, *HTML*, что позволяет проводить тестирование веб-сервисов на различных уровнях сложности.

1.1.5.3 *TestNG*

TestNG – это фреймворк для модульного и интеграционного тестирования. Он предоставляет широкий набор возможностей для тестирования, в том числе:

– аннотации для настройки и запуска тестовых методов;

– группировка тестов;

– параметризованные тесты;

– определение тестовых зависимостей;

– обработка исключительных ситуаций;

– отчетность;

– параллельное выполнение тестов.

TestNG также обеспечивает улучшенную интеграцию с средой сборки проектов, такими как *Maven* или *Gradle*, что обеспечивает легкую автоматизацию тестирования в рамках проектов различных размеров.

1.1.5.4 *Katalon Studio*

Katalon Studio представляет собой комплексный набор инструментов, разработанный для автоматизации тестирования *web* и мобильных приложений. Этот инструментарий обладает мощными возможностями, способными решить распространенные проблемы, связанные с автоматизацией *UI* тестирования, такие как *pop-up*, *iFrame*, и *wait-time*. Благодаря своей гибкости и простоте использования, *Katalon Studio* помогает тестировщикам улучшить процесс тестирования, повысить скорость работы и достигать высокого качества программного обеспечения благодаря интеллектуальным возможностям, которые предоставляются на протяжении всего процесса автоматизации тестирования. Каждый компонент инструментария предлагает разнообразные способы упрощения технического

обслуживания и расширения возможностей проекта.

1.1.5.5 *Appium*

Appium – это бесплатный кроссплатформенный инструмент с открытым исходным кодом для автоматизации тестирования мобильных приложений на *iOS* и *Android* платформах. *Appium* разработан с использованием базовых веб-технологий, таких как *JavaScript, HTML, CSS*, и работает преимущественно в контексте *Selenium WebDriver*. Он поддерживает многие языки программирования: *Java, Python, Ruby, JavaScript*. Фреймворк дает возможность эмулировать действия пользователя, такие как касание, ввод текста и т.д. Это мощный инструмент для автоматизации, который помогает проверять функциональность, поведение и производительность мобильных приложений.

1.1.6 Объект тестирования

В ПГУ широко используется Электронная информационная образовательная среда (ЭИОС). В ней можно найти нужные для учебы материалы, с её помощью возможно сдавать задания по многим предметам, общаться на учебных форумах, пользоваться электронной библиотекой, узнавать свои оценки и многое другое.

Федеральные государственные образовательные стандарты высшего образования (ФГОС ВО) предусматривают внедрение ЭИОС в учебный процесс по всем направлениям подготовки, включая бакалавриат, специалитет, магистратуру и аспирантуру. ЭИОС представляет собой комплексные средства, включающие информационно-телекоммуникационные технологии и инструменты, способствующие эффективному освоению образовательных программ студентами [3].

1.1.7 Описание работы тестируемого объекта

Для составления плана тестов необходимо определить цели и область тестирования. Четко обозначить, что именно нужно достичь с помощью автотеста. Определить функциональность, которая подлежит проверке, и

установить приоритеты для различных тестовых случаев.

Составление плана теста включает изучение требований и документации. Требования к программному обеспечению и другие документы, связанные с ЭИОС ПГУ, описаны в системе управления курсами *Moodle* в разделе «Курс по ЭИОС». В разделе приведены материалы о проекте разработки ЭИОС ПГУ в целом, а также инструкции по её эксплуатации. На рисунке 1 представлена структура электронной информационной образовательной среды университета.

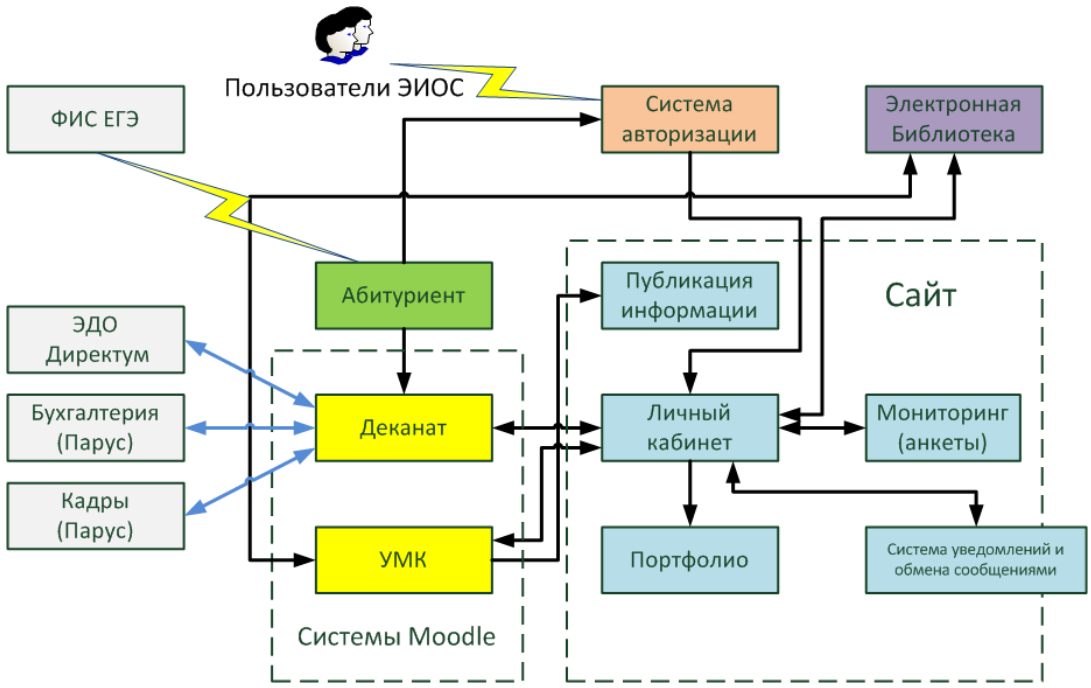


Рисунок 1 – Структура ЭИОС ПГУ

1.1.7.1 Авторизация и информация в личном кабинете

Для работы в ЭИОС следует авторизоваться в Личном кабинете ЭИОС. На форму авторизации можно попасть с сайта каждой из подсистем ЭИОС. На рисунке 2 представлена главная страница сайта ПГУ,

Рисунок 2 – Главная страница

После клика на ссылке «Личный кабинет» откроется окно авторизации, что показано на рисунке 3.

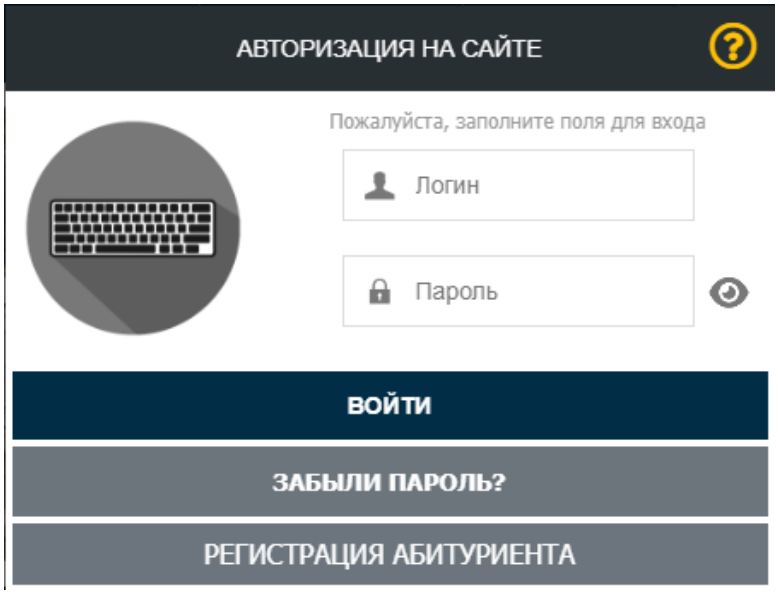


Рисунок 3 – Окно авторизации

После авторизации вы уже можете работать в ЭИОС под своей учетной записью. Если вы авторизовались в ЭИОС не через сайт «Личного кабинета», то для перехода в личное портфолио следует кликнуть по своей фамилии, которая появилась вместо надписи «Личный кабинет», что показано на рисунке 4.

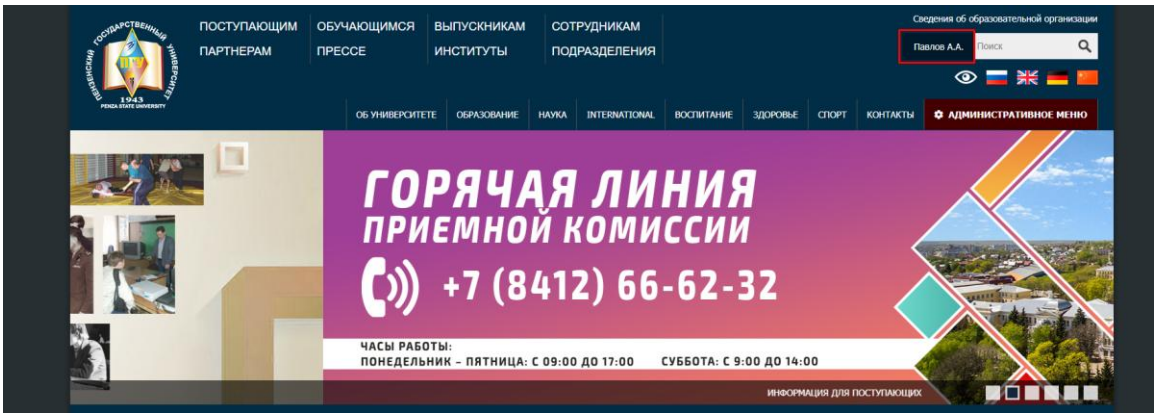


Рисунок 4 – Кнопка перехода в личное портфолио

В Личном кабинете написано имя пользователя. Вверху находится главное меню ЭИОС, позволяющее переходить в основные подсистемы ЭИОС. Дополнительно осуществлять авторизацию в других подсистемах ЭИОС не требует – сессия сохраняется на протяжении всего сеанса работы с ЭИОС до момента закрытия браузера.

Вид страницы личного кабинета показан на рисунке 5.

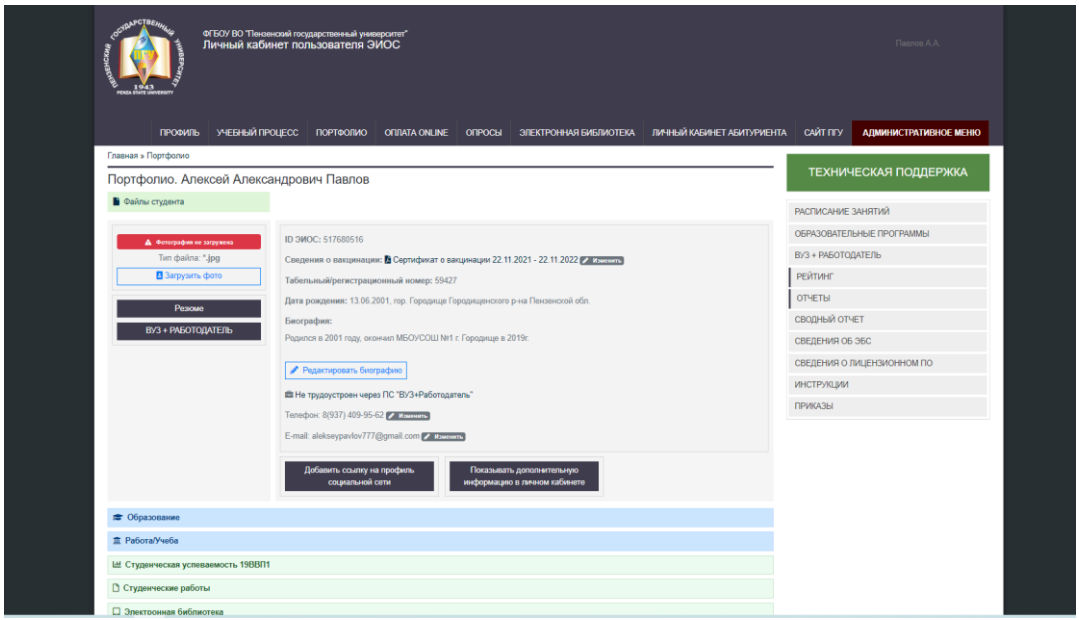


Рисунок 5 – Страница личного кабинета

1.1.7.2 Просмотр промежуточной аттестации

Выбрав пункт главного меню «Учебный процесс», происходит переход

в свой личный кабинет в подсистеме учебного процесса ПГУ – рисунок 6

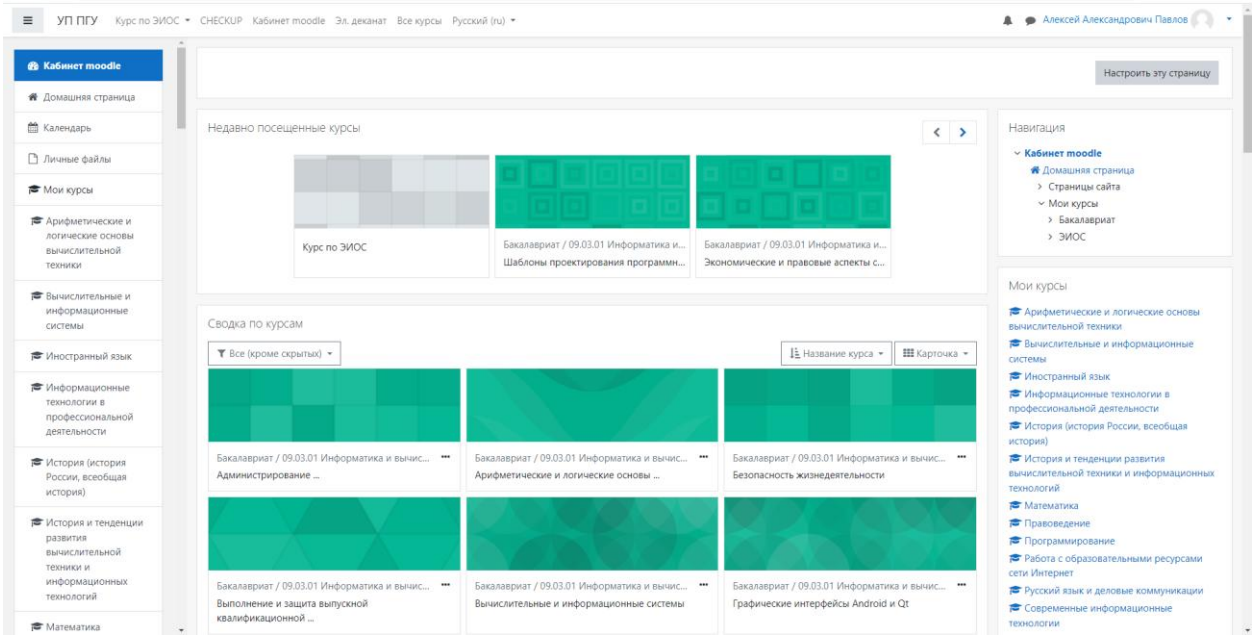


Рисунок 6 – Личный кабинет в подсистеме учебного процесса ПГУ

По центру приведен список доступных курсов в полном формате (с указанием курса и образовательной программы). Слева в «Мои курсы», также отображается список доступных курсов в кратком формате. Кликнув по названию ссылки, происходит переход в раздел курса, где представлены материалы по курсу. Здесь представлены учебные и методические материалы по курсу, ссылки на полезные книги, презентации и т.д. Многие преподаватели принимают через курсы в ЭИОС лабораторные работы, контрольные, курсовые, другие виды заданий. В каждом курсе имеется один или несколько форумов для обсуждения вопросов курса.

В верхнем меню есть кнопка «Эл. деканат». В электронном деканате следует выбрать Учащимся – Зачетная книжка.

Дальше необходимо перейти в «Просмотр промежуточной аттестации», где можно увидеть оценки за экзамены, зачеты, курсовые и пр. Данная страница представлена на рисунке 7.

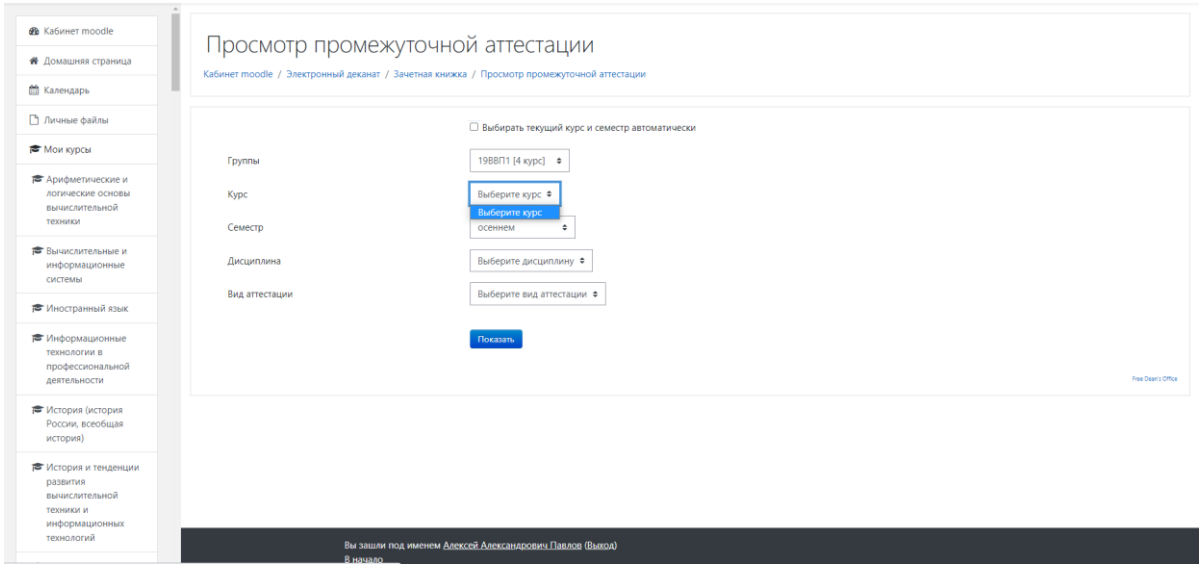


Рисунок 7 – Страница просмотра промежуточной аттестации

Для просмотра необходимо заполнить селект поля и нажать кнопку «Показать», в результате чего отобразится таблица с информацией по промежуточной аттестации. Пример таблицы представлен на рисунке 8.

Рисунок 8 – Таблица промежуточной аттестации

1.1.8 Выводы

В данной главе помимо постановки задачи были рассмотрены основные типы, направления и фреймворки для функционального автоматизированного тестирования, а также объект тестирование, и пример описания его функционала.

Для реализации поставленной задачи с учётом особенностей тестируемого учебного портала в качестве основного инструмента выбраны фреймворки *Selenium и TestNG*.

В следующей главе подробно излагаются теоретические и практические аспекты тестирования применительно к поставленной задаче.

1.2 Проектирование тестового окружения

1.2.1 Инструменты разработки

Для выполнения всех действий, связанных с тестированием сервиса в среде Selenium, необходимо установить некоторые программные компоненты.

1.2.1.1 *IntelliJ IDEA*

Для установки среды разработки *IntelliJ IDEA* надо на официальной странице *IntelliJ IDEA* [4] скачать *Community Edition* в расширении «*.exe*». Страница загрузки представлена на рисунке 9.

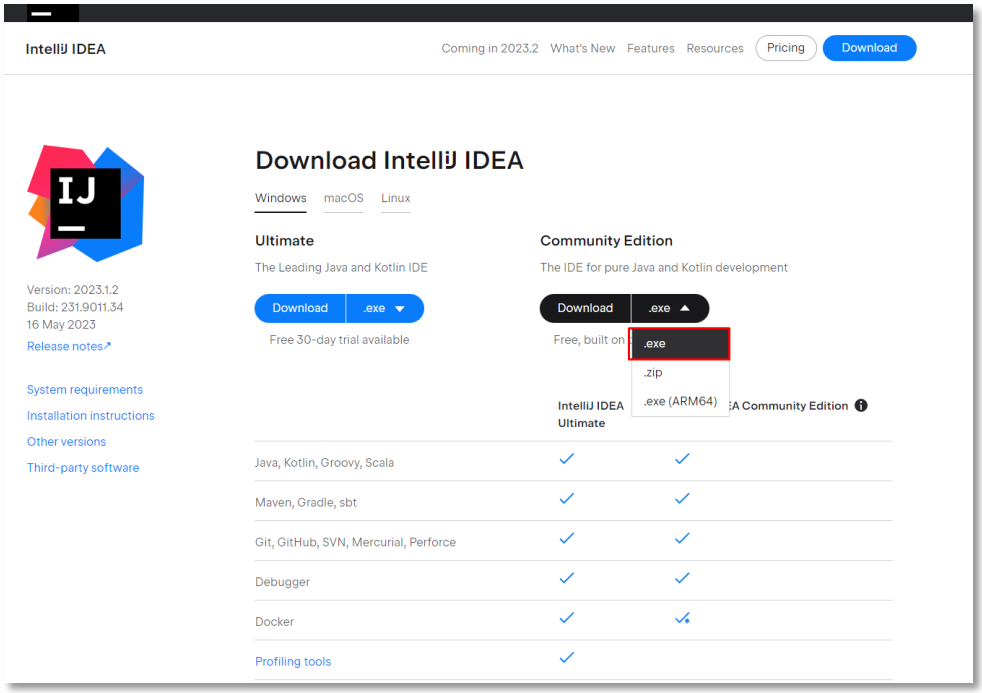


Рисунок 9 – Страница загрузки *IntelliJ IDEA*

Далее необходимо запустить скачанный «*.exe*» файл и установить среду разработки.

В параметрах установки выбрать:

– *Create Desktop Shortcut: IntelliJ IDEA Community Edition* — создание ярлыка на рабочем столе;

– *Add «Open Folder as Project»* — позволяет открывать папки с исходниками как проекты прямо через контекстное меню;

– *.java* — файлы такого расширения будет открывать *IDEA* (по умолчанию и через контекстное меню).

Указанные параметры выделены на рисунке 10.

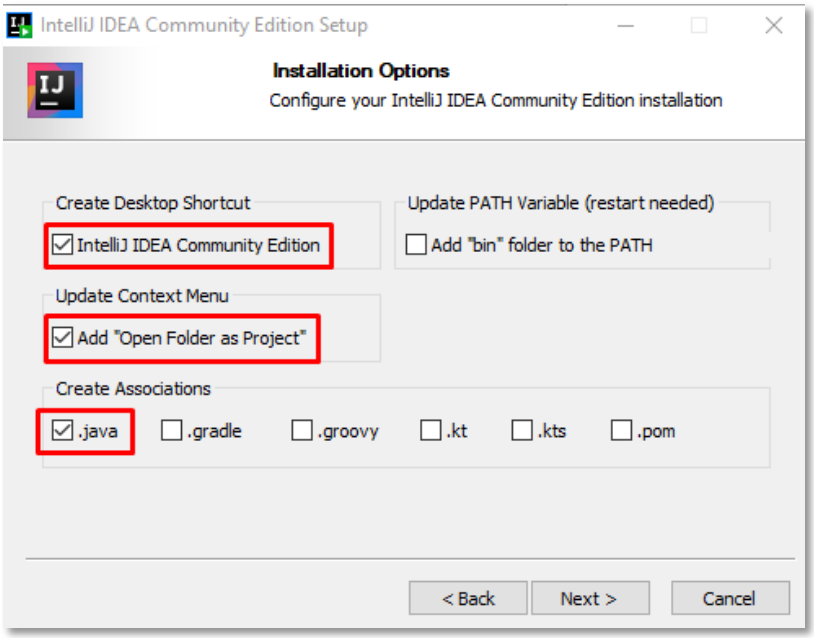


Рисунок 10 – Параметры установки

При запуске *IntelliJ IDEA* в настройке параметров оставить всё по умолчанию.

На рисунке 3 показано открывшееся окно приветствия, где необходимо перейти в *Projects – New Project*.

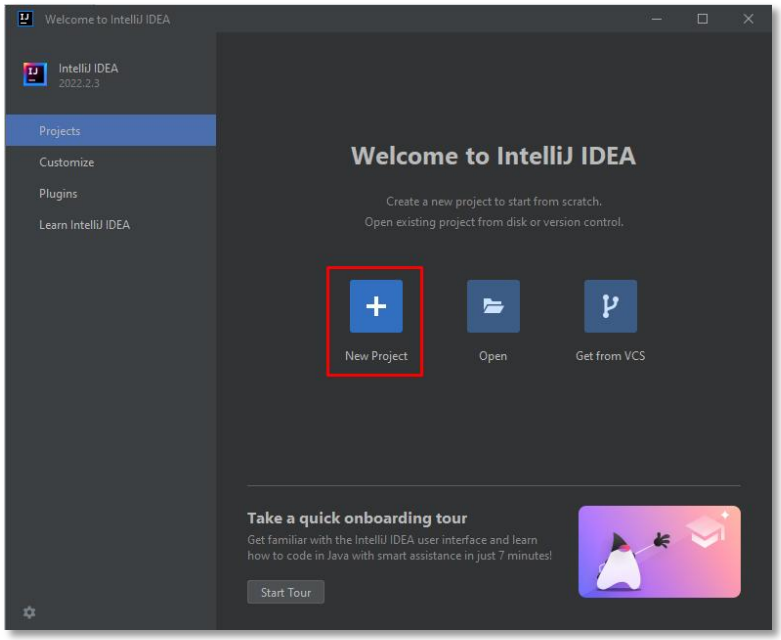


Рисунок 11 – Стартовое окно приветствия

На рисунке 4 выделены параметры создаваемого проекта.

Язык – Java.

Система сборки – *Maven.*

JDK – correto-1.8. Изначального в списке его не будет, подробнее о его установке рассказывается в следующем пункте.

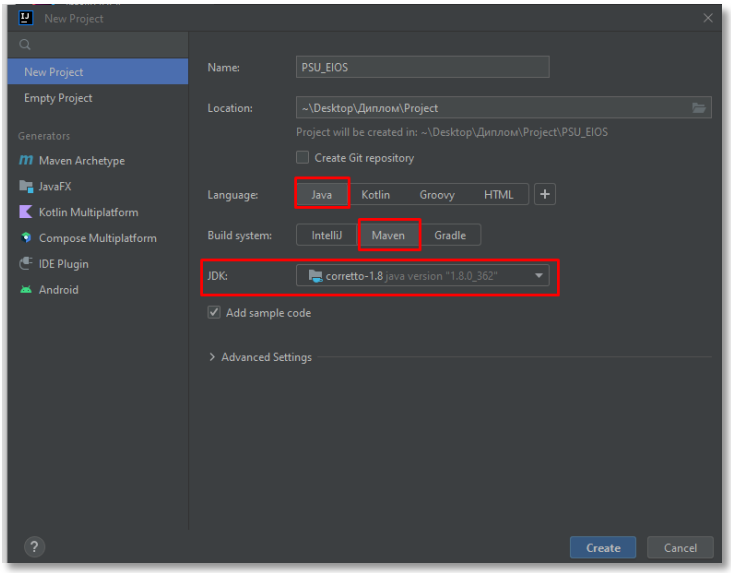


Рисунок 11 – Настройка параметров проекта

1.2.1.2 *JDK*

Для загрузки *JDK* в пункте меню нужно выбрать опцию «*Download JDK…*». Данное действие отображено на рисунке 12.

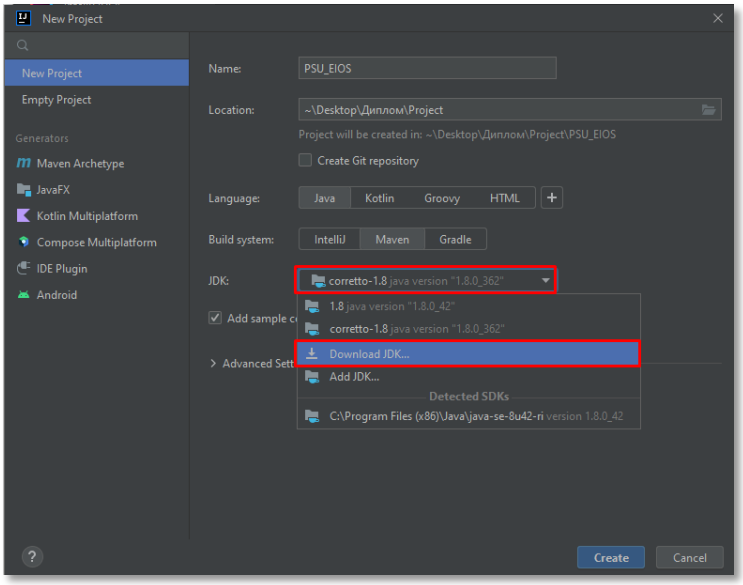


Рисунок 12 – Открытие меню загрузки *JDK*

В качестве поставщика (*Vendor*) выбираем *Amazon Corretto* и нажимаем *Download*. Это действие показано на рисунке 13.

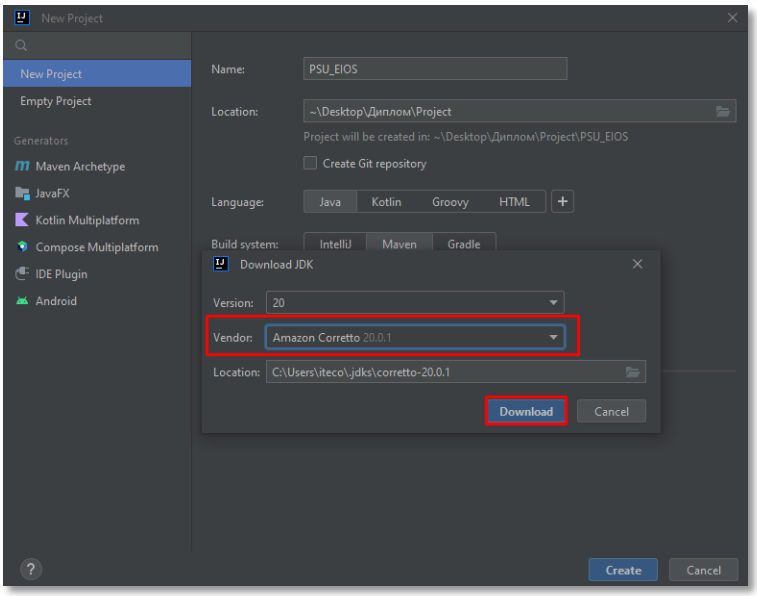


Рисунок 13 – Загрузка *JDK*

1.2.1.3 *Maven*

Вся структура проекта описывается в файле *pom.xml* (*POM – Project Object Model*), который должен находиться в корневой папке проекта. Это «*.xml*» файл, который содержит информацию о сборке текущего проекта детали проекта, конфигурации и зависимостей. Местонахождение упомянутого файла в окне навигации проекта выделено на рисунке 14.

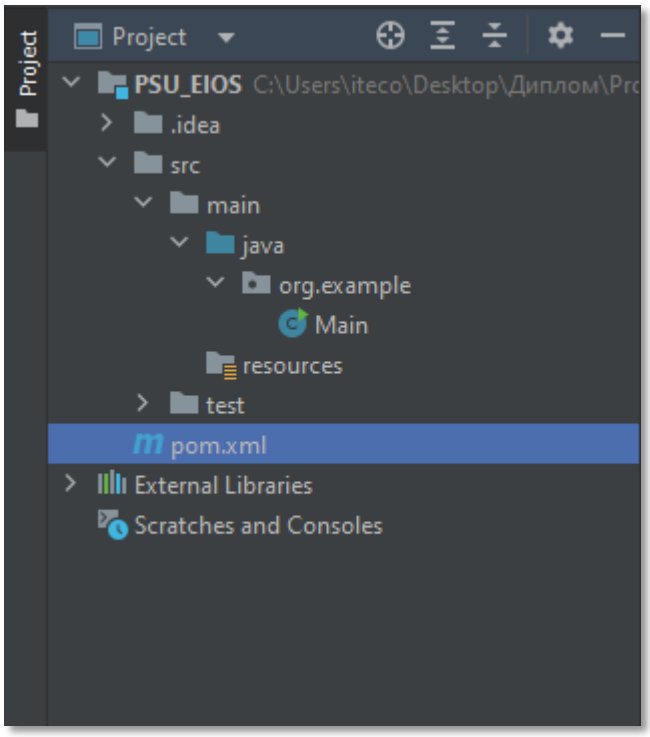


Рисунок 14 – Окно навигации проекта

В нём создаётся тег зависимостей <*dependencies*> </*dependencies*>, в котором описываются все используемые в проекте библиотеки. Каждую зависимость необходимо выделить тегом <*dependency*> </*dependency*> и указать уникальные идентификационные данные: *groupId*, *artifactId* и *version*. *Maven* сам подгрузит транзитивные зависимости.

Подключаем библиотеки *Selenium*, *Allure* и *TestNG*.

```
<dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
```

```
            <artifactId>selenium-java</artifactId>
            <version>4.3.0</version>
        </dependency>
        <dependency>
            <groupId>io.qameta.allure</groupId>
            <artifactId>allure-testng</artifactId>
            <version>2.17.3</version>
        </dependency>
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.5</version>
        </dependency>
    </dependencies>
```

В тег характеристик (*properties*) добавляем *AspectJ* (механизм аспектно-ориентированного программирования, который используется в *Allure Framework* для перехвата выполнения тестовых шагов, отмеченных аннотациями), *Maven Surefire Plugin* (предназначен для запуска тестов и генерации отчетов по результатам их выполнения), кодировку *UTF-8* и название *XML* файла, по которому будет собираться проект.

```
    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <aspectj.version>1.8.10</aspectj.version>
        <maven-surefire-plugin.version>2.20</maven-surefire-
plugin.version>
        <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
        <suiteXml>testng.xml</suiteXml>
    </properties>
```

Тег *<build> </build>* содержит информацию по самой сборке: где

находятся исходные файлы, где ресурсы, какие плагины используются. Добавляем *Maven Surefire Plugin* с конфигурацией *AspectJ* и системными параметрами с указанием папки формирования отчётов *Allure* и вторым плагином *Allure Maven*.

```
    <build>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-surefire-
plugin</artifactId>
                    <version>2.20</version>
                    <configuration>
                        <argLine>
                            -
javaagent:"${settings.localRepository}/org/aspectj/aspectjweaver
/${aspectj.version}/aspectjweaver-${aspectj.version}.jar"
                        </argLine>
                        <suiteXmlFiles>

<suiteXmlFile>src/test/resources/${suiteXml}</suiteXmlFile>
                        </suiteXmlFiles>
                    </configuration>
                    <dependencies>
                        <dependency>
                            <groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>${aspectj.version}</version>
                        </dependency>
                    </dependencies>
                </plugin>
                <plugin>
                    <groupId>io.qameta.allure</groupId>
```

```
                <artifactId>allure-maven</artifactId>
                <version>2.10.0</version>
            </plugin>
        </plugins>
    </build>
```

Далее надо нажать на кнопку обновления сборки проекта. Эта кнопка выделена на рисунке 15.

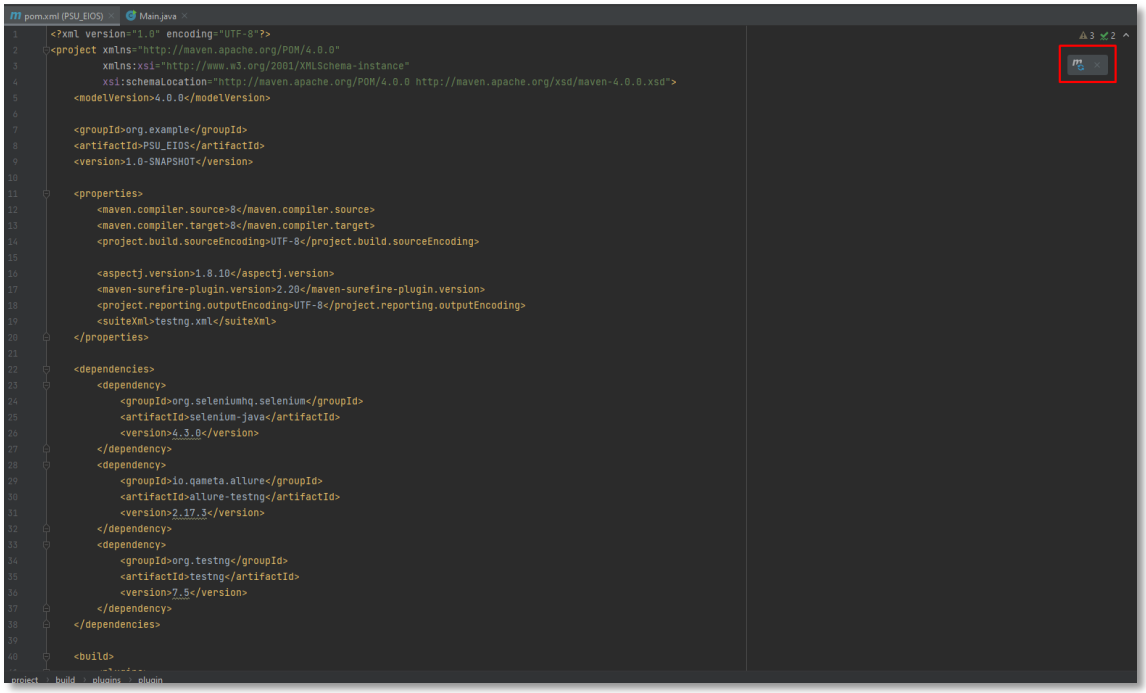


Рисунок 15 – Кнопка обновления *Maven* сборки проекта

И по завершении сборки проверить наличие *Allure Menu*. Данное меню показано на рисунке 16.

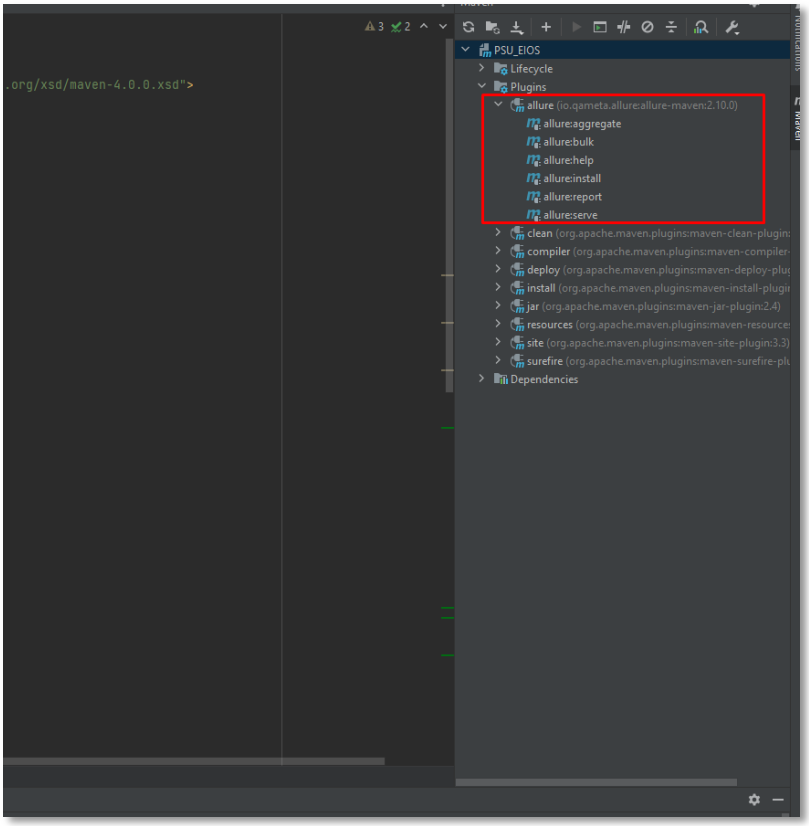


Рисунок 16 – *Allure* меню

1.2.1.4 *ChromeDriver*

ChromeDriver скачивается с официального сайта[5]. Страница загрузки отображена на рисунке 17. Здесь нужно выбрать номер версии, соответствующую установленной версии *Chrome*.

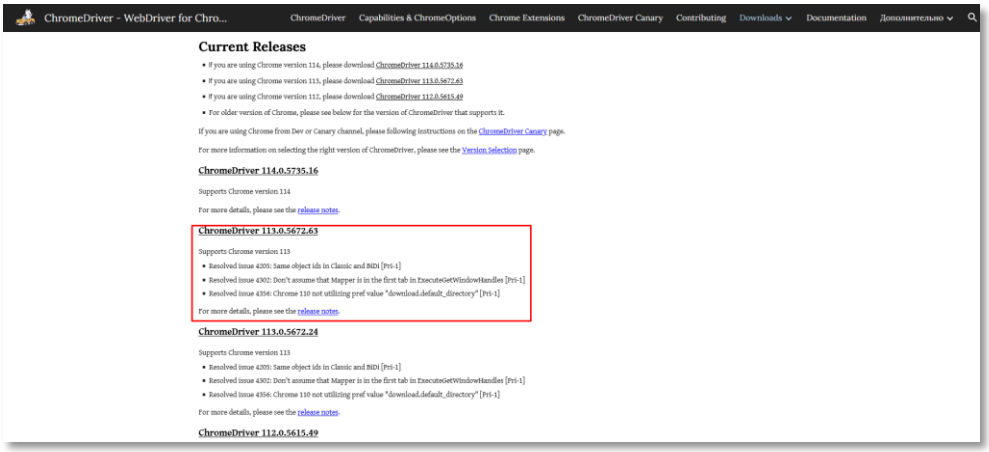


Рисунок 17 – Страница загрузки *ChromeDriver*

Из предложенных вариантов выбрать *win32.zip*. Указанный вариант показан на рисунке 18.

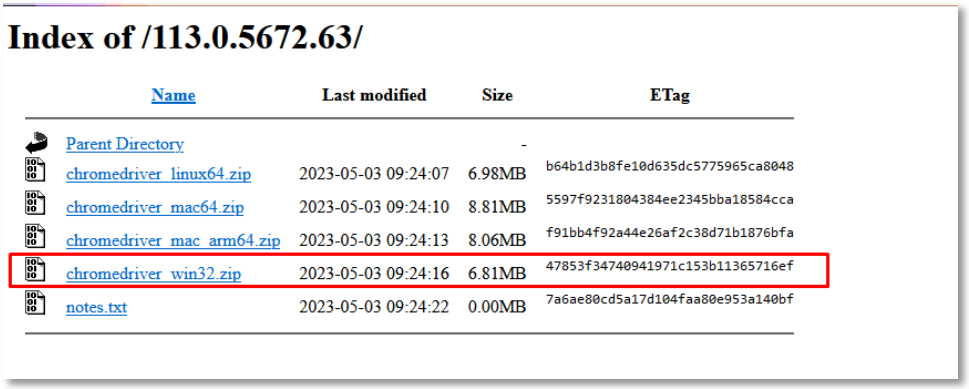


Рисунок 18 – Выбор *ChromeDriver* под платформу *Win32*

Разархивируем исполняемый файл «*chromedriver.exe*» в любое удобное место, после чего в проекте в каталоге *main.java* создаём новый каталог *common*. Создание каталога проиллюстрировано на рисунке 19. Для этого нажимаем ПКМ по каталогу *java – New – Package*.

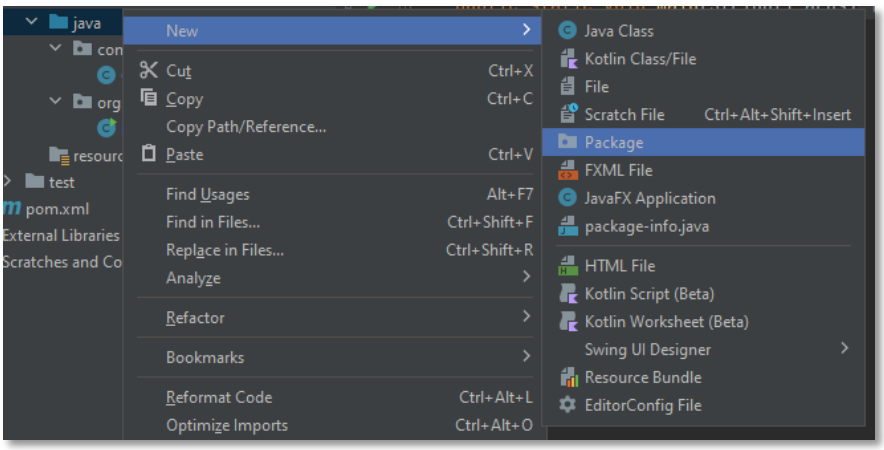


Рисунок 19 – Создание каталога в проекте

В каталоге *common* тем же образом создаётся *Java Class* «*Config*» и «*Driver*». В классе *Config*, который показан на рисунке 20, указывается путь к ранее скачанному *ChromeDriver*.

Рисунок 20 – Класс *Config*

В классе *Driver*, содержание которого показано на рисунке 21, определяются параметры запуска *ChromeDriver*.

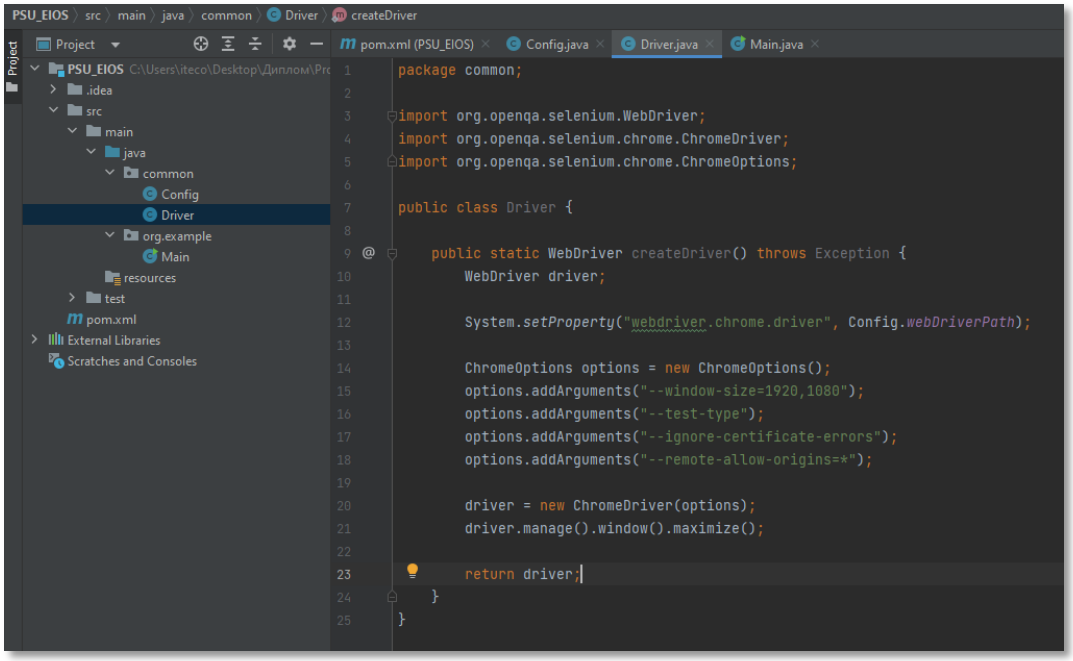


Рисунок 21 – Класс *Driver*

В каталоге *test.java* создаётся каталог *base* с классом *BaseTest*, в котором будет описываться метод инициализации драйвера.

Каталог base также создаётся в каталоге main.java, внутри которого ещё нужно создать файлы классов *BaseActions* и *BasePage*.

BaseActions будет содержать объект драйвера, а также в нём в дальнейшем будут описываться базовые действия тестирования (ожидания, клики, ввод и тд.). Путь к классу *BaseTest* и его содержание продемонстрированы на рисунке 22. Класс *BaseActions* показан на рисунке

23.

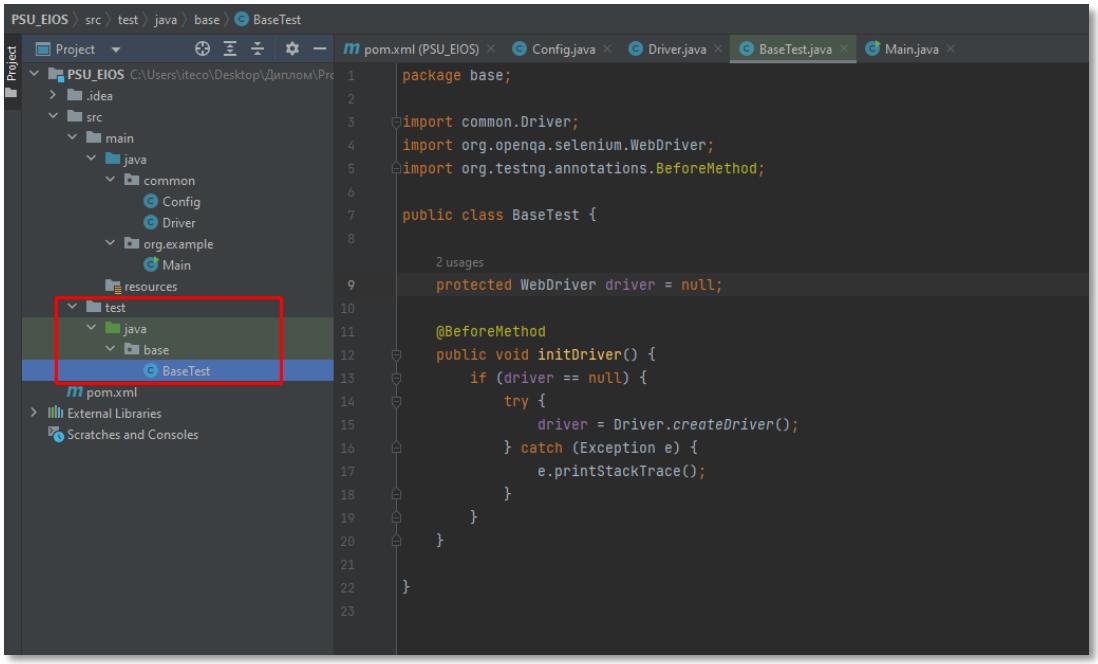


Рисунок 22 – Класс *BaseTest*

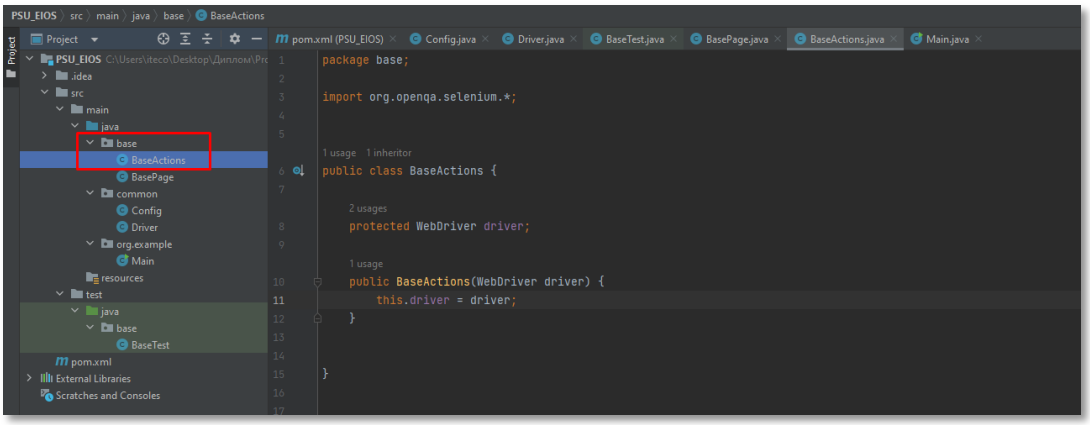


Рисунок 23 – Класс *BaseActions*

В *BasePage* – метод открытия исходной страницы при запуске теста. Этот класс изображен на рисунке 24.

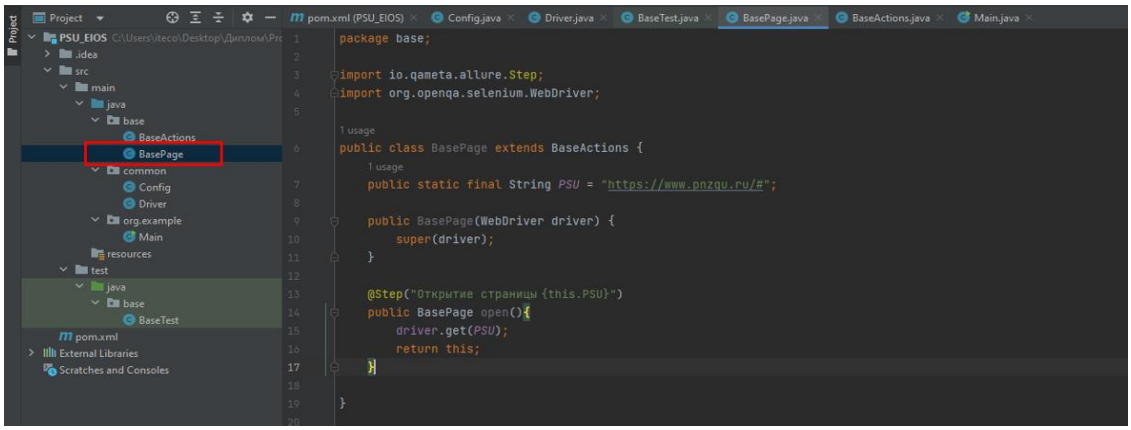


Рисунок 24 – Класс *BasePage*

1.2.1.5 *Jenkins*

Jenkins представляет собой популярный *DevOps* инструмент с открытым исходным кодом, который широко применяется в организации процессов непрерывной интеграции и непрерывной доставки программного обеспечения. Это кроссплатформенное приложение, разработанное на языке программирования *Java* и представляющее собой систему для сборки проектов, включая такие операции, как создание проектов из исходного кода, запуск модульных тестов и генерация отчетов о сборке, которые в дальнейшем могут быть переданы соответствующим участникам команды разработчиков [7]. Пример процесса работы показан на *UML*-диаграмме на рисунке 25.

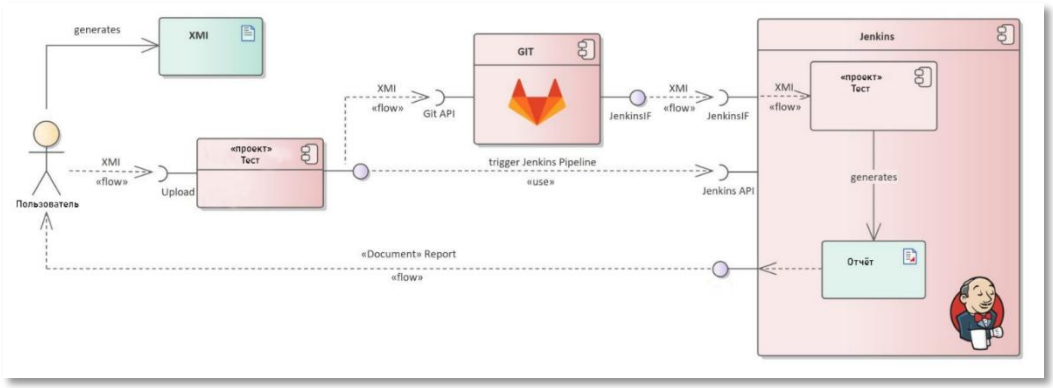


Рисунок 25 – *UML*-диаграмма процесса модульного тестирования с набором инструментов *CI/CT*

Jenkins требует наличия *JDK* версии 8 или выше для работы.

Для установки *Jenkins* выполняются следующие действия.

На официальном сайте *Jenkins* [6] скачивается *.war*-файл *Jenkins*.

Далее необходимо открыть командную строку и перейти к директории, в которую был загружен файл *.war*. Затем выполняется команда "*java -jar jenkins.war*", которая инициирует запуск сервера *Jenkins*.

Обычно *Jenkins* запускается на порту 8080 по умолчанию. В случае, если данный порт уже занят другим сервисом, можно указать альтернативный порт при запуске с помощью команды "*java -jar jenkins.war --httpPort*=*порт*".

При первом запуске *Jenkins* будет сгенерирован пароль, который понадобится в процессе регистрации.

По завершении процесса установки, пользователь получит сообщение о готовности *Jenkins* к работе *"Jenkins is fully up and running"*. Далее необходимо запустить веб-браузер и открыть страницу *localhost*, указав соответствующий порт. Ранее было отмечено, что по умолчанию *Jenkins* функционирует на порту 8080.

По завершении регистрации и предустановке стандартного пакета плагинов необходимо в настройках в разделе *Tools* установить *Maven* и *Allure Commandline*. Установка показана на рисунках 26 и 27.

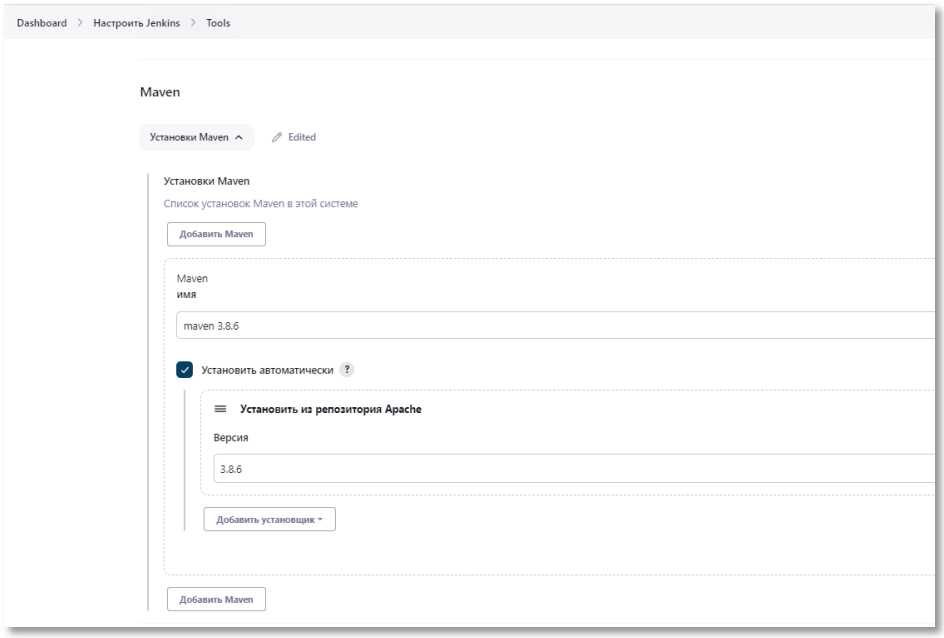


Рисунок 26 – Установка *Maven* в среде *Jenkins*

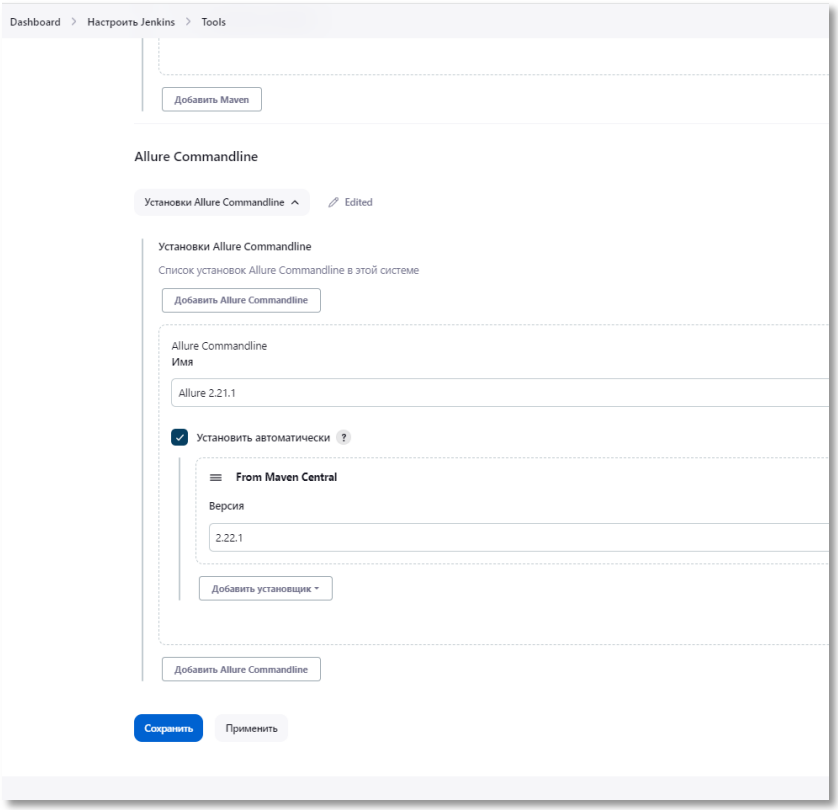


Рисунок 27 – Установка *Allure Commandline* в среде *Jenkins*

С помощью данных инструментов настраивается порядок шагов сборки проекта. Пример настройки порядка шагов сборки проекта показан на

рисунке 28.

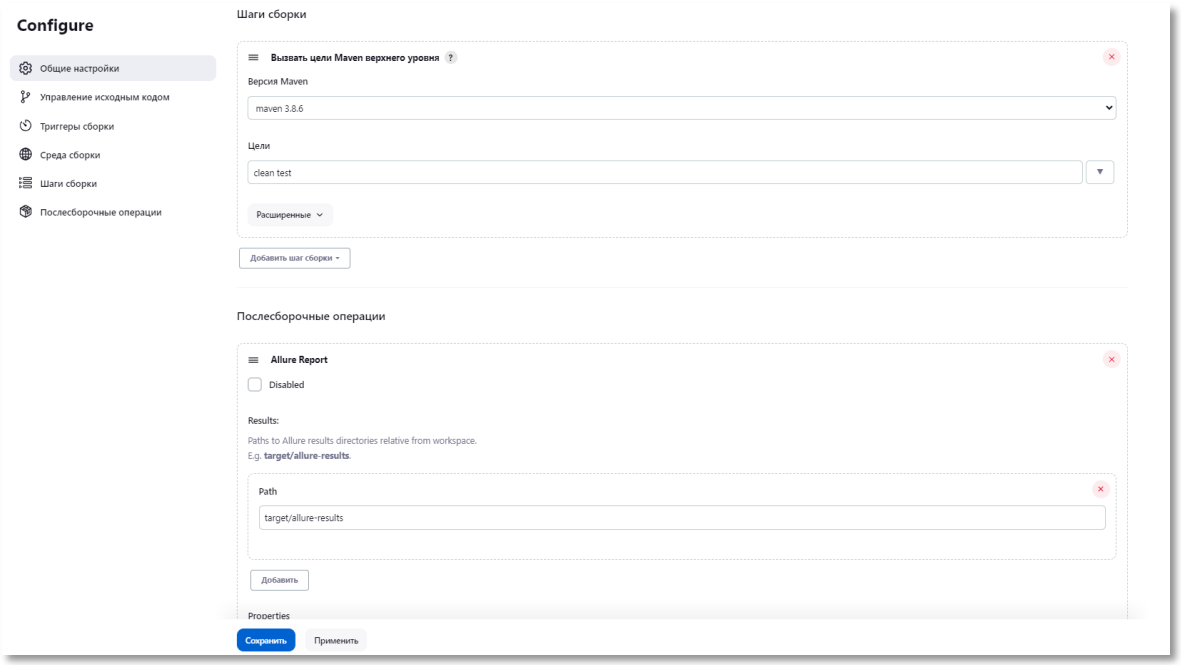


Рисунок 28 – Конфигурация сборки проекта

1.2.2 Реализация бизнес-логики и функциональности автотеста посредством реализации структуры проекта с использованием паттернов проектирования

1.2.2.1 Основные разделы проекта

Проект имеет два корневых пакета: *Main* и *Test*. В пакете *Main* собрана логика проекта, в пакете *Test* сами тесты, то есть их сценарии и основные аспекты.

Базовым классом для тестов (пакет *Test*), содержащим их общие аспекты, такие как запуск, завершение драйвера и браузера, сохранение скриншота на момент нахождения ошибки, является класс *BaseTest*. От этого класса наследуется класс *EiosTest*, в котором хранятся методы исполняемого кода, то есть вызова основных шагов тестирования.

Внутренняя реализация шагов тестирования (пакет *Main*) объединена в *PageObject* файлах. Это сделано с целью удобного подключения этих файлов в необходимые тесты. Подробнее об использовании паттерна *PageObject*

будет написано далее.

Разделение на уровне реализации логических шагов тестирования выполнено с помощью паттерна *Steps*. Подробнее этот паттерн будет описан в дальнейшем.

Пакет *Main* также содержит пакет *Common*, в котором хранятся общие классы работы фреймворка, как, например, настроек конфигурации драйвера и параметров его запуска.

В пакете *Models* реализован класс *UserModel*, посредством которого происходит передача необходимых в процессе тестирования данных о пользователе.

Сами же данные пользователей хранятся в файле *Users*.

В пакете *Elements* хранятся объекты для описания взаимодействия с элементами веб-страницы, как, например, селект элементов выпадающего списка, чекбоксов и тд. Подробнее об использовании паттерна *Elements Object* будет написано далее.

На рисунке 29 показана структура проекта с учетом вышеописанного. На рисунке 30 изображена *UML* диаграмма классов проекта.

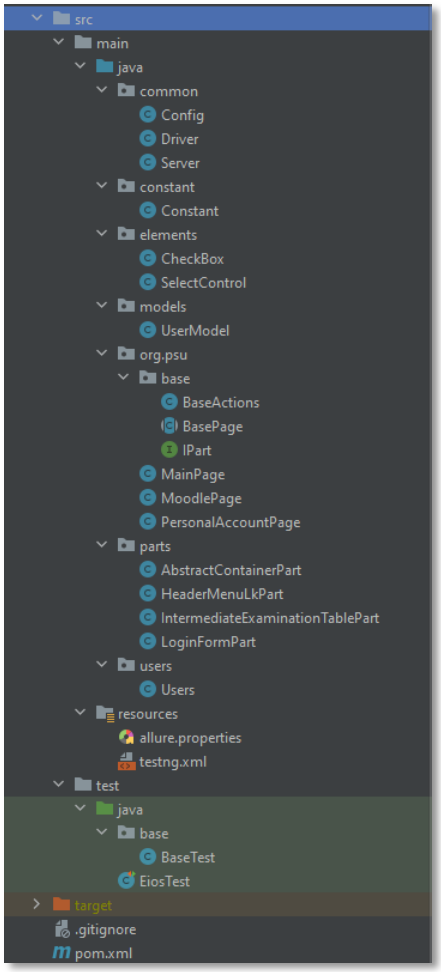


Рисунок 29 – Структура проекта, включающая содержимое основных каталогов

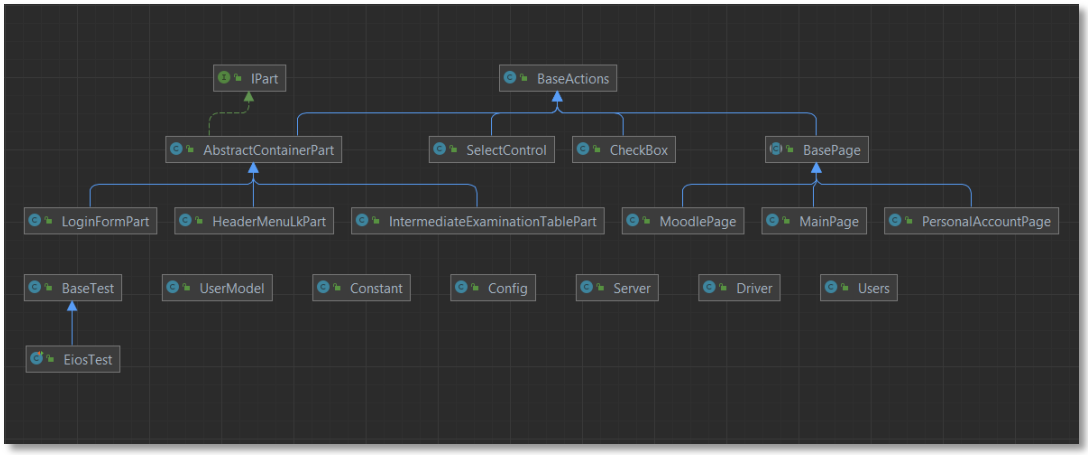


Рисунок 30 – *UML* диаграмма классов проекта

1.2.2.2 Использование паттерна *PageObject*

Данный паттерн был специально придуман для использования в автоматизированном тестировании и позволяет сделать код более чистым и понятным, разделить функциональность и вынести методы и поля отдельных страниц в отдельный класс.

*Page Objec*t определяет в себе части:

– *BasePage,* расширяемый *BaseActions* — реализует в себе необходимые методы для работы с *WebDriver;*

– *Page Objects* — реализуют методы для работы с элементами на веб-страницах. В данную категорию входят классы:

1) *MainPage.* Содержит функциональность главной страницы учебного портала ПГУ, а, конкретнее, выполнения авторизации в систему;

2) *PersonalAccountPage.* Хранит методы для взаимодействия со страницей личного кабинета;

3) *MoodlePage.* Содержит функционал проверки системы управления виртуальной обучающей среды *Moodle;*

– *Tests* — реализуют тесты, описанные бизнес-логикой тест-кейсов.

Классы *Page Objects* проверяют функциональность разделов бизнес-логики учебного портала. В них хранятся переменные, содержащие локаторы (XPath) элементов веб-страниц, а также логика взаимодействия с этими элементами.

Тесты реализованы в классе *EiosTest*.

Схема паттерна *Page Object* в данном проекте изображена на рисунке 31.

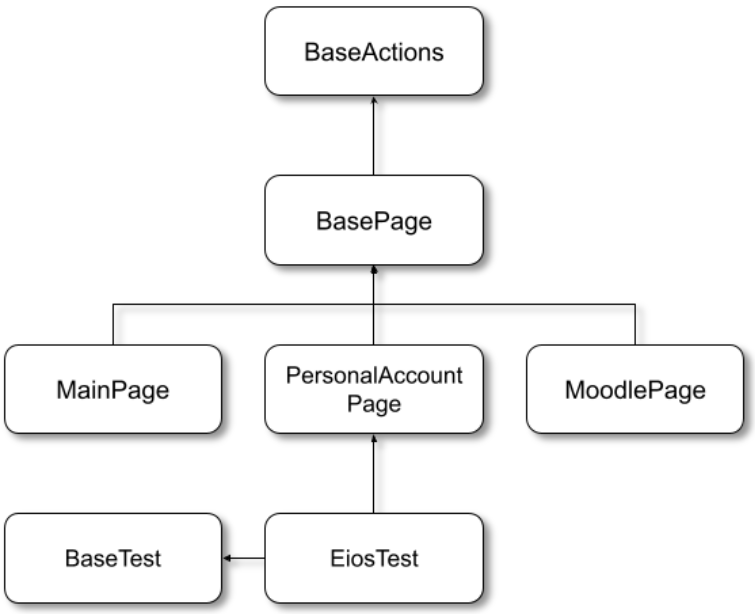


Рисунок 32 – Схема проекта по паттерну *Page Object*

1.2.2.3 Использование паттерна *Steps*

Паттерн автоматизации тестирования *Steps* направлен на систематизацию тест-кейсов путем разделения их на последовательные шаги. Каждый шаг представляет из себя определенную последовательность действий, направленных на проверку определенного функционала или процесса в приложении.

Преимущества использования паттерна *Steps* в автоматизации тестирования:

– удобное управление и масштабирование тестовых случаев;

– упрощение процесса отладки автотестов;

– улучшение качества тестирования и повышение точности результатов тестирования.

Подобное разделение методов позволяет реализовать необходимую бизнес-логику проекта, сохраняя такое же разделение шагов в автоматизированных тестах. Это облегчает ориентацию в проекте, способствует написанию чистого кода и позволяет оценить уровень

покрытия автотестами разделения бизнес-логики в интернет-системе.

Во фреймворке *TestNG* «*Step*» – это пользовательская аннотация для описания шага выполнения тест-кейса. Она позволяет разбить тест-кейс на множество отдельных шагов и задать для каждого из них определенное описание.

Аннотация *Step* применяется перед методом-шагом, который содержит код для выполнения определенного действия. После запуска тестов результаты работы каждого шага будут отображены в отчете *Allure*, что позволяет легко просматривать прохождение тест-кейса и определить, на каком шаге возникла ошибка.

В качестве примера подобного шага ниже представлен метод перехода на главную страницу учебного портала ПГУ.

```
    // Аннотация объемлющего шага, который будет отображаться в
отчёте
    @Step("Переход на сайт " + Server.PSU)
        public MainPage openPagePNZGU(){
    // Вложенный шаг открытия страницы
            open();
    // Вложенный шаг проверки перехода на страницу
            waitURL(Server.PSU, Constant.TimeOut.SHORT);
    // Вложенные шаги проверки отображения элементов страницы
            waitElementVisible("//div[@class='header-
logo']/a[@title='На главную']",
                    "активное лого с функциональной возможностью
перехода «На главную».");
            headerMenuLkPart.checkElements();
            checkLeftUpMenu();
            checkHeaderMenu();
            waitElementVisible("//div[@class='slider']",
"слайдер новостей");

            return this;
```

```
}
```

Пример результата работы данного метода в отчёте представлен ниже на рисунке 33.

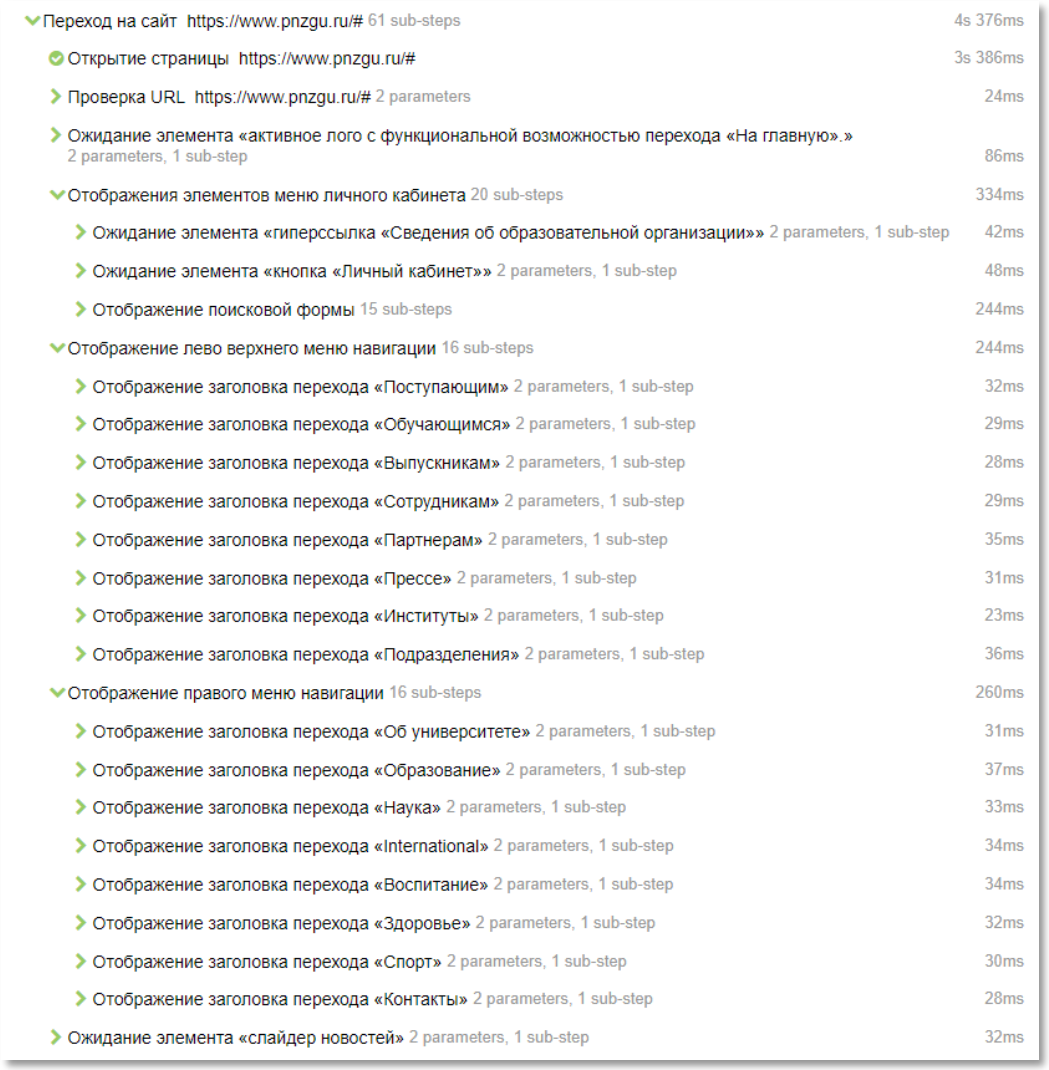


Рисунок 33 – Отчёт о результате работы метода перехода на главную страницу учебного портала ПГУ

Для наглядного представления реализации в проекте паттерна *Steps* на рисунке 34 представлена диаграмма *Step* методов в классах проекта.

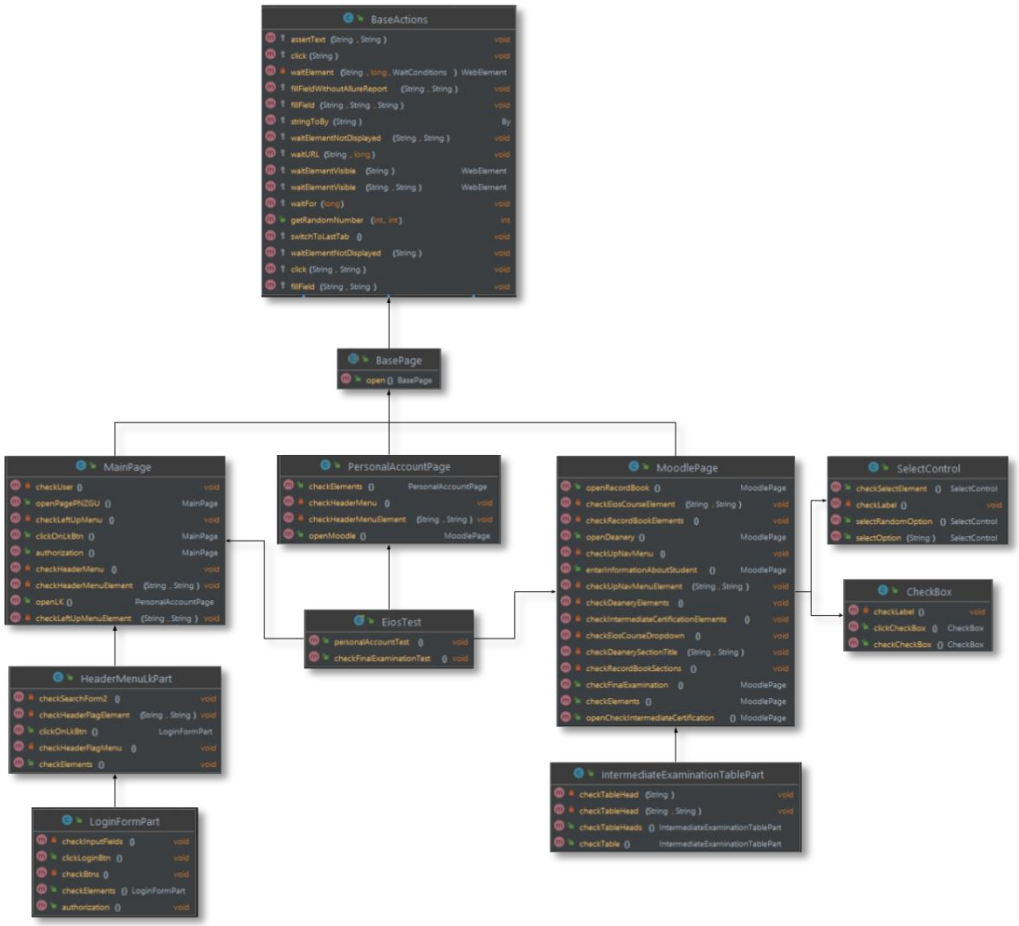


Рисунок 34 – Диаграмма классов проекта по паттерну *Steps*

Листинги классов, показанных на диаграмме, представлены в приложении Б.

1.2.3 Сценарии функционального тестирования

1.2.3.1 Требования к тест-кейсам для автоматизации тестирования

Тестировщики сами составляют тест-кейсы на основе требований к разрабатываемому продукту. Поэтому для них важен навык правильного написания тест-кейсов.

При создании тест-кейса важно учитывать следующие моменты.

Тест-кейсы должны быть легко читаемыми, понятными и хорошо структурированными, их должен понимать даже не знакомый с продуктом разработчик.

В предусловии важно описать состояние системы, которое нужно для

выполнения шагов тест-кейса. Например, там могут быть ссылки на среды или конкретные области, где проводятся тестирования, или идентификатор другого теста в случае, если текущий тест имеет зависимость от его результата.

В тест-кейсах должны быть чётко определены ожидаемые результаты. Тестирование должно строиться на сравнении фактических результатов работы тестируемого приложения с предполагаемыми результатами.

Главное, тест-кейсы должны быть корректными и доказать свою работоспособность на будущих релизах системы, поэтому повторное использование тестов для проверки стабильности приложения – это один из показателей общей эффективности автоматизированного тестирования.

1.2.3.2 Пример реализации сценария

На рисунках 35-38 представлены примеры тестовых сценариев.

[ПГУ ЭИОС-1] Проверка входа пользователя с существующими логином и паролем					
Проект:	ПГУ ЭИОС				
Тип:	Тест		**Приоритет:**	Высокий	
Тестировщик:	Павлов Алексей Александрович		**Ответственный:**	Павлов Алексей Александрович	
Тестовый сценарий:	**Шаги**		**Данные**	**Ожидаемый результат**	
	1	Перейти на сайт https://www.pnzgu.ru/		Открыта страница «Пензенский государственный университет». Отображено: 1) Активное лого с функциональной возможностью перехода «На главную». 2) Меню, содержащее: • гиперссылку «Сведения об образовательной организации» • кнопку «Личный кабинет» • Поисковую строку • Меню иконок с функциональной возможность смены версии сайта: o Для слабовидящих o Русская o Международная o Немецкая o Китайская 3) Меню навигации с заголовками: • Поступающим • Обучающимся • Выпускникам • Сотрудникам • Партнерам • Прессе • Институты • Подразделения 4) Меню навигации с заголовками: • Об университете • Образование • Наука • International • Воспитание • Здоровье • Спорт • Контакты 5) Слайдер новостей
	2	Нажать на кнопку "Личный кабинет" в верхнем правом углу экрана.		Открылась форма «Авторизация на сайте», на которой отображено: 1) Текстовая подсказка «Пожалуйста, заполните поля для входа» 2) Поля ввода:

Рисунок 35 – Тестовый сценарий

				• Логин • Пароль 3) Кнопки • ВОЙТИ • ЗАБЫЛИ ПАРОЛЬ? • РЕГИСТРАЦИЯ АБИТУРИЕНТА
	3	Ввести в поля ввода верные исходные данные. Нажать на кнопку «Войти».	Логин: Пароль:	Форма авторизации закрылась. В правом верхнем меню название кнопки «Личный кабинет» сменилось на фамилию и инициалы пользователя «Павлов А.А.».
	4	Нажать на кнопку с фамилией и инициалами пользователя «Павлов А.А.».		В новой вкладке открылся личный кабинет пользователя ЭИОС. Отображено: 1) Меню навигации: • Профиль • Учебный процесс • Портфолио • Оплата Online • Опросы • Электронная библиотека • Личный кабинет абитуриента • Сайт ПГУ • Административное меню 2) Заголовок «Портфолио. Алексей Александрович Павлов.» 3) Карточка с информацией о пользователе.
Описание	Проверка требований к реализации визуализации и управлению бизнес-функциями, реализованными на стороне Backend service в части экранной формы «Авторизация на сайте» на странице «Пензенский государственный университет»			

Рисунок 36 – Тестовый сценарий

[ПГУ ЭИОС-2] Просмотр «Промежуточной аттестации»				
Проект:	ПГУ ЭИОС			
Тип:	Тест	**Приоритет:**	Средний	
Тестировщик:	Павлов Алексей Александрович	**Ответственный:**	Павлов Алексей Александрович	
Предварительные действия:	Тест «ПГУ ЭИОС-1»			
Тестовый сценарий:				

	Шаги	**Данные**	**Ожидаемый результат**

1	В меню навигации нажать на кнопку «Учебный процесс»		В новой вкладке открылся кабинет moodle.
			Отображено меню навигации:
			• Выпадающий список «Курс по ЭИОС» с элементами:
			  ○ Курс по ЭИОС
			  ○ Инструкции
			  ○ Вопросы регистрации пользователей
			  ○ Проект ЭИОС и нормативные документы
			  ○ Форум для консультаций по загрузке ресурсов
			  ○ Форум для консультаций по работе со студентами
			  ○ Форум по тестам
			  ○ Категория ЭИОС
			• CHECKUP
			• Кабинет moodle
			• Эл. Деканат
			• Все курсы

2	В меню навигации нажать на кнопку «Эл. деканат»		Открылась страница «Электронный деканат». Отображены заголовки: • Преподавателям • Учащимся • Настройки • Администрирование Под заголовком «Учащимся» отображена гиперссылка «Зачетная книжка».
3	Нажать на гиперссылку «Зачетная книжка»		Открылась страница «Зачетная книжка». Отображены гиперссылки: • Просмотр рейтинга • Просмотр промежуточной аттестации • Настройка дисциплин по выбору

Рисунок 37 – Тестовый сценарий

	4	Нажать на гиперссылку «Просмотр промежуточной аттестации»		Открылась страница «Просмотр промежуточной аттестации». Отображены: 1) Чекбокс «Выбирать текущий курс и семестр автоматически» 2) Селект элементы: • Группы • Курс • Семестр • Дисциплина • Вид аттестации
	5	Деактивировать чекбокс «Выбирать текущий курс и семестр автоматически» и выбрать в полях селект элементов случайные опции		Чекбокс деактивирован. Поля заполнены. Кнопка «Показать» активна.
	6	Нажать на кнопку «Показать»		В нижней части экрана отобразился «Журнал аттестации [группа обучающегося]» с таблицей «Выставление промежуточной аттестации». Таблица имеет заголовки: • № п/п • ФИО студента • Контрольная точка 1 • Добор КТ1 • Контрольная точка 2 • Добор КТ2 • КМ после КТ • Доп. баллы • $R^{тек}$ • КМ сданы • Допуск деканат • $R^{атт}$ • $R^{дис}$ • Оценка
Описание	Проверка требований к реализации визуализации и управлению бизнес-функциями, реализованными на стороне Backend service в части экранной формы «Просмотр промежуточной аттестации» на странице «Электронного деканата»			

Рисунок 38 – Тестовый сценарий

1.2.4 Выводы

Во второй главе были рассмотрены следующие моменты:

– подготовка инструментов разработки тестового окружения;

– структура проекта автотестирования;

– использование паттернов проектирования в проекте;

– требования к тест-кейсам автотеста, а также их примеры.

1.3 Результаты работы системы автоматизированного тестирования

1.3.1 Результаты выполнения и структура отчета

Обзор (*Overview*) представляет собой главную страницу отчета *Allure*, обладающую блочной структурой [8]:

– блок *ALLURE REPORT* содержит информацию о дате и времени проведения теста, общем количестве выполненных запросов, а также график, отображающий процентное соотношение и количество успешных, неудачных и завершившихся с ошибкой запросов в процессе выполнения теста;

– блок *TREND* показывает тенденцию выполнения теста от сборки к сборке, иллюстрируя изменения в результате выполнения тестовых сценариев по мере развития проекта;

– блок *SUITES* отображает распределение результатов тестирования по группам тестов (наборам тестов), позволяя увидеть, какие части функциональности были протестированы и с каким результатом;

– блок *ENVIRONMENT* показывает информацию о тестовой среде, в которой были запущены тесты, информацию о чём впоследствии извлекается из специального файла проекта и включается в отчет;

– блок *CATEGORIES* демонстрирует распределение неудачных тестов по типам сбоев, позволяя определить, какие категории тестов испытывают наибольшие трудности или проблемы;

– блок *FEATURES BY STORIES* отображает распределение тестов по функциональным возможностям, которые они проверяют, что позволяет оценить покрытие тестами различных аспектов функциональности приложения;

– блок *EXECUTORS* показывает информацию о текущем исполнителе сборки. Если тесты были запущены с помощью инструмента непрерывной

интеграции (например, *Jenkins*), то здесь будет представлена информация о задаче и номере сборки, связанных с данным выполнением тестов. На рисунке 39 показан пример главной страницы сгенерированного отчёта.

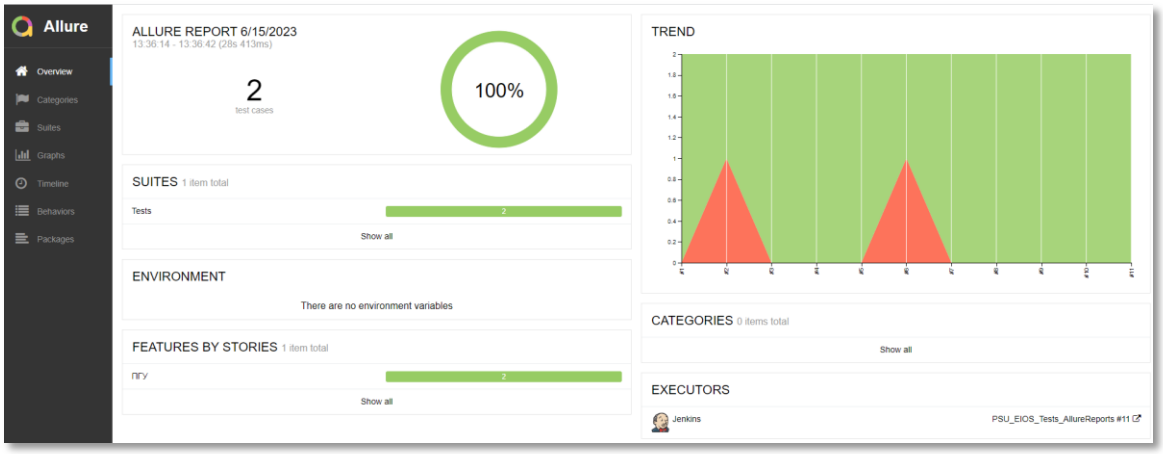


Рисунок 39 – Главная страница *Allure*-отчета

Во вкладке *Behaviors* отображены более подробные результаты на основе группирования тестов по функциональности. Успешно завершенные тесты выделены зеленой иконкой. В разделе *overview* можно проследить историю выполненных действий, а также увидеть затраченное на них время.

Шаги теста изначально отображены в свернутом состоянии. Для просмотра вложенных шагов необходимо раскрыть объемлющие шаги, нажав на соответствующие строки. Результат изображён на рисунке 40.

В случае нахождения проблем, полоса сценария будет выделяться характеризующей красной иконкой. При раскрытии вложенных шагов будет выделен шаг, на котором возникла ошибка. Также фреймворк создает скриншот рабочей области в момент падения теста, который также будет сохранён в описании сценария. На рисунке 41 представлен отчет сценария, завершённого ошибкой.

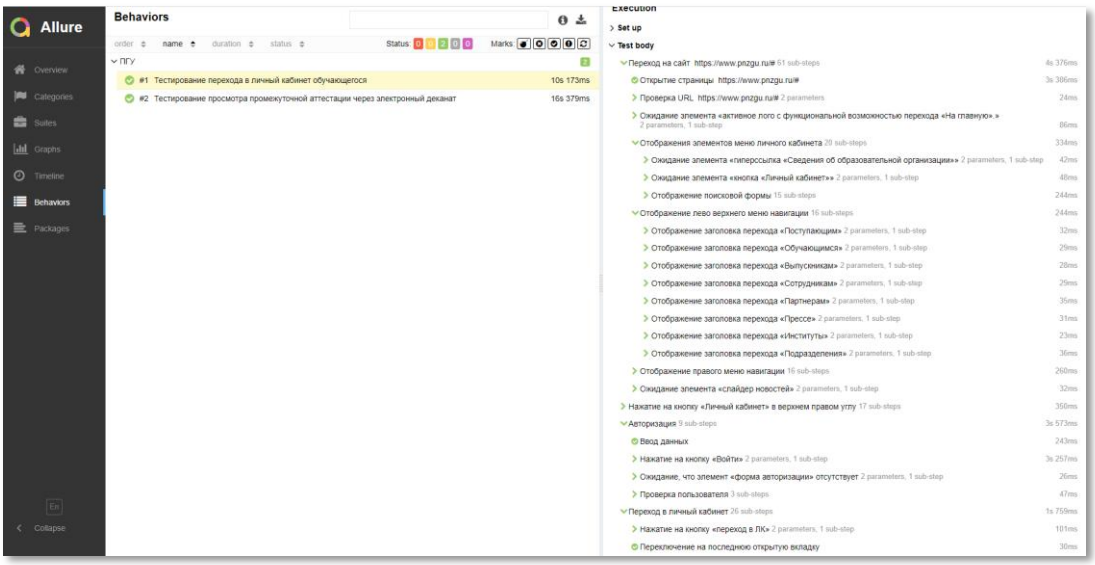


Рисунок 40 – Обзор результатов тестирования

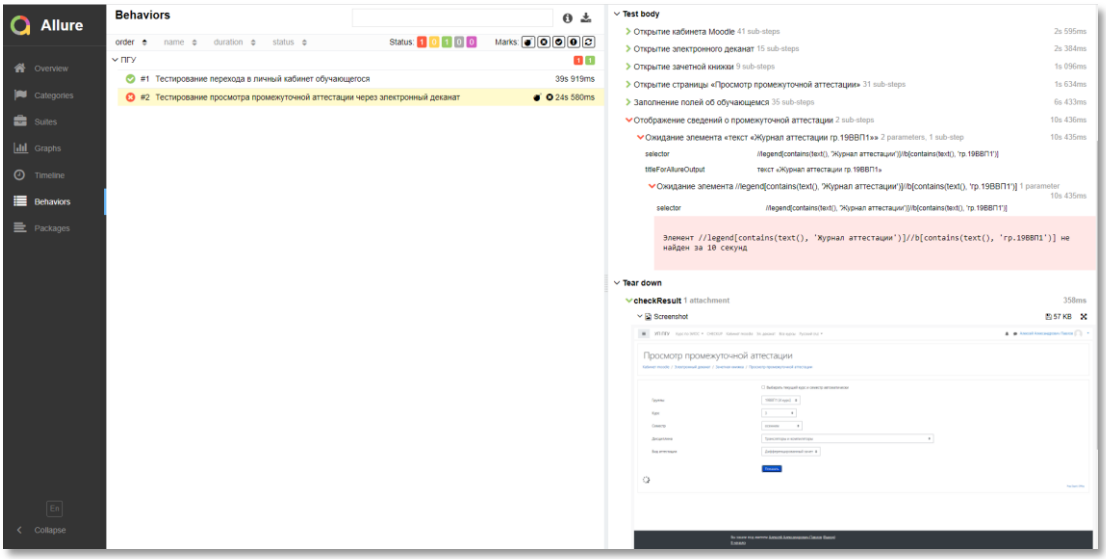


Рисунок 41 – Завершение с ошибкой тестового сценария

Как видно из описания, фреймворк за указанное время не смог найти ожидаемый объект на странице. Это подтверждается и скриншотом, по которому можно отследить затянувшуюся загрузку ожидаемого раздела страницы. Вложенный скриншот показан на рисунке 42.

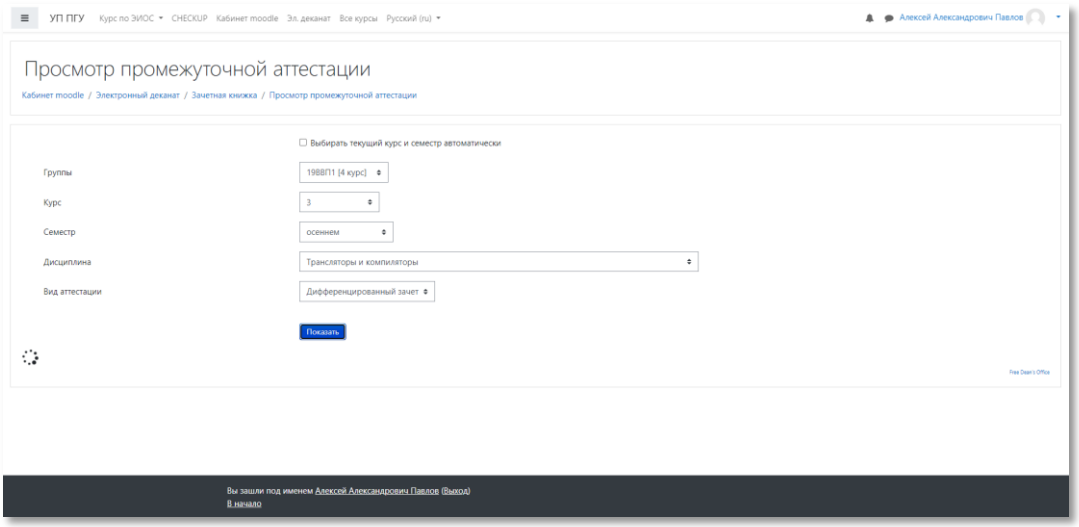


Рисунок 42 – Вложенный скриншот рабочей области в момент падения теста

1.3.2 Выводы

В третьей главе было представлены следующие моменты:

– результаты тестирования в виде сгенерированного отчета о прохождении тестов;

– примеры сборок успешно пройденных тестов;

– пример сборки теста, завершившегося с ошибкой.

2 Охрана труда

2.1 Загрязнение воздуха вредными веществами

В отличие от окружающей атмосферы на улице, внутреннее загрязнение воздуха в помещениях реже доступно для визуального, вкусового или обонятельного восприятия. При отсутствии визуальных сигналов, мы чаще пренебрегаем этим аспектом. Тем не менее, такое отношение является неправильным, поскольку известно, что загрязненный воздух оказывает негативное воздействие на здоровье человека. Он может вызывать респираторные инфекции и рассматривается как одна из ведущих причин рака лёгких и хронической обструктивной болезни легких (ХОБЛ).

В результате двухлетнего исследования, опубликованного в медицинском журнале *Lancet*, установлено, что низкое качество воздуха год за годом является причиной 6,5 миллиона преждевременных смертей. Кроме того, оно также снижает производительность труда.

Исследование, проведенное в 2014 году, выявило, что каждое увеличение концентрации вредных частиц *PM2,5* в воздухе на 10 микрограмм на кубический метр приводит к снижению производительности работников по сборке груш на 0,41 доллара в час. Мы часто стремимся укрыться в помещении, чтобы избежать загрязненного воздуха, однако такой подход довольно часто является ошибочным.

В соответствии с данными Агентства по охране окружающей среды США (*EPA*), уровень загрязнения воздуха в помещении обычно на два или даже пять порядков превышает уровень загрязнения воздуха на улице. В некоторых случаях качество воздуха внутри помещений может быть даже в 100 раз хуже, чем на открытой территории.

Мэтью С. Джонсон, исследователь компании *Airlabs*, специализирующейся на установке фильтров, способных удалить до 95% загрязнителей и вредных газов из воздуха, объясняет, что к загрязненному

воздуху с улицы в помещении добавляются собственные загрязняющие вещества в результате приготовления пищи, уборки и использования строительных материалов.

2.2 Повышенный уровень шума на рабочем месте

В соответствии с трудовым законодательством, каждый работодатель обязан обеспечивать приемлемый уровень шума на рабочем месте, в соответствии с требованиями, изложенными в СанПиН 2.2.4.3359-16. Этот нормативный документ, озаглавленный как "Санитарно-эпидемиологические требования к физическим факторам на рабочих местах", был утвержден постановлением Высшей государственной санитарной инспекции Российской Федерации от 21 июня 2016 года № 81 и вступил в силу с 1 января 2017 года. Раздел III данного документа посвящен уровню шума на производстве.

Стоит отметить, что представленные в нормативном документе нормы шума на рабочих местах основаны на научном подходе. Однако, формализация уровня шума на производстве и соответствующие законодательные требования представляют собой некоторую проблему для лиц, не являющихся профессионалами в данной области.

Существующий СанПиН не предоставляет конкретного перечня возможных источников шума на рабочем месте. Однако, важным фактором является степень негативного влияния шума на условия труда. Это определяет гигиенические аспекты шума на конкретном рабочем месте. При проведении специальной оценки условий труда шум обязательно учитывается как один из неблагоприятных факторов.

2.3 Недостаточная освещенность рабочей зоны

Недостаток освещения оказывает влияние на функционирование зрительной системы, определяя ее работоспособность, а также воздействует

на психологический и эмоциональный статус человека. Это вызывает утомление центральной нервной системы, которое возникает в результате усилий, направленных на восприятие нечетких или неоднозначных сигналов.

Исследования подтверждают, что свет, помимо своей роли в зрительном восприятии, воздействует на нервно-оптико-вегетативную систему, систему формирования иммунной защиты, рост и развитие организма, а также на множество других основных процессов жизнедеятельности. Он регулирует обмен веществ и способность организма справляться с неблагоприятными факторами окружающей среды. Сравнительный анализ естественного и искусственного освещения в отношении их влияния на производительность показывает преимущество естественного освещения.

Важно отметить, что на организм человека влияют не только интенсивность света, но и все аспекты его качества. Неравномерное освещение может вызвать проблемы с адаптацией из-за снижения видимости. Работа в условиях низкого качества или слабого освещения может привести к утомлению глаз и снижению работоспособности. В некоторых случаях это может вызывать головные боли. Факторами, способствующими возникновению таких проблем, являются недостаточное освещение, блики от источников света и неподходящая яркость. Головные боли также могут быть вызваны пульсирующим светом. Таким образом, неправильное освещение является значительным фактором риска для здоровья работников.

Рабочее освещение играет важную роль в оптимизации условий труда. Основная цель освещения рабочих мест состоит в обеспечении четкой видимости визуальных объектов и снижении зрительного напряжения и усталости.

Освещение должно быть равномерным и стабильным, с правильным направлением света, исключением бликов и предотвращением образования резких теней.

2.4 Повышенная яркость света

Известно, что практически 90% информации мы получаем благодаря нашим зрительным органам. Плохое освещение внутри помещений, наличие пульсирующих ламп, которые невозможно заметить невооруженным глазом, могут привести к различным заболеваниям глаз и ухудшению психического здоровья в течение нескольких лет. Негативное воздействие неприятного освещения ощущается не только нашим зрением, но и всем организмом. Это проявляется в форме усталости, сонливости, частых головных болей, повышения кровяного давления и, в результате, снижения работоспособности.

Также стоит отметить, что слишком высокая яркость также оказывает негативное воздействие на организм и может привести к ухудшению зрения. Поэтому рекомендуется использовать высококачественное светодиодное освещение, которое не оказывает негативного влияния на здоровье. Приятное освещение оказывает тонизирующий эффект на организм человека, способствует хорошему настроению и улучшению работы нервной системы.

Были проведены научные исследования с целью изучения влияния освещения на безопасность и производительность труда, в результате чего были сделаны следующие выводы:

– грамотное организация освещения в офисных помещениях способствует эффективной работе, снижению текучести кадров и увеличению производительности до 32%;

– улучшение условий освещения в производственных помещениях значительно повышает производительность и качество работы;

– статистика несчастных случаев на рабочих местах показывает, что в случаях, где система освещения была правильно спроектирована, количество несчастных случаев уменьшается в два раза;

– оптимальное освещение сокращает количество брака на 30%;

– качественное освещение в учебных аудиториях оказывает позитивное влияние на учеников и студентов, они легче воспринимают учебный материал, при этом меньше устают, а такое популярное заболевание как близорукость исключается.

Подводя итоги, в любом помещении освещение должно быть рациональным, то есть сочетать хороший световой поток, высокое качество, экономичность и безопасность.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана система автоматизированного тестирования, осуществляющая функциональную проверку учебного портала ЭИОС ПГУ. Данная система, предусматривающая возможность отложенного запуска, выполняет виртуально действия пользователя и генерирует отчет о проведенных проверках, а также оповещает пользователя о результатах по электронной почте.

Система разработана на базе фреймворков *Selenium, TestNG и Allure* на языке *Java* и осуществляет работу на сервере непрерывной интеграции *Jenkins* посредством автоматизированной сборки фреймворком *Maven*.

Одним из достоинств разработанной системы является эффективное разделение функциональности системы с помощью таких паттернов проектирования, как *Page Object* и *Steps*, что в дальнейшем позволит легко расширять объём тестируемого функционала в электронно-информационно образовательной среде ФГБОУ ВО ПГУ.

В дальнейшем можно усовершенствовать разработанную систему и добавить взаимодействие с базой данных, что значительно облегчит вопрос хранения и получения тестовых данных.

Список используемых источников

1. protesting.ru: [Электронный ресурс]. URL: http://www.protesting.ru/testing/types/smoke.html (дата обращения: 26.04.2023).
2. core.ac.uk: [Электронный ресурс]. URL: https://core.ac.uk/download/pdf/197460977.pdf (дата обращения: 28.04.2023)
3. ui.pnzgu.ru: [Электронный ресурс]. URL: https://ui.pnzgu.ru/files/ui.pnzgu.ru/doc/polozenie_eios.pdf (дата обращения: 28.04.2023)
4. jetbrains.com: [Электронный ресурс]. URL: https://www.jetbrains.com/idea/download/ (дата обращения: 29.04.2023)
5. chromedriver.chromium.org: [Электронный ресурс]. URL: https://chromedriver.chromium.org/downloads (дата обращения: 29.04.2023)
6. jenkins.io: [Электронный ресурс]. URL: https://www.jenkins.io/download/ (дата обращения: 01.05.2023)
7. pcnews.ru: [Электронный ресурс]. URL: https://pcnews.ru/blogs/%5Bperevod%5D_zapusk_testov_selenium_v_jenkins-1078801.html#gsc.tab=0 (дата обращения: 01.05.2023)
8. habr.com: [Электронный ресурс]. URL: https://habr.com/ru/companies/sberbank/articles/358836/ (дата обращения: 03.05.2023)
9. Дастин Э., Рэшка Дж., Пол Дж. Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. — Лори, 2019. — С.4-5.
10. Куликов С. Тестирование программного обеспечения. Базовый курс. — Четыре четверти, 2015. — С. 67-93.
11. Cem Kaner, James Bach, Bret Pettichord «Lessons Learned in Software Testing» — С.41-42.

12. Борис Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. — Питер, 2004. — С. 272-281.

13. Гленфорд Майерс, Том Баджетт, Кори Сандлер. Искусство тестирования программ. — Вильямс, 2012. — С. 25-31.

14. Джеймс Уиттакер, Джейсон Арбон, Джефф Каролло. Как тестируют в Google. — Питер, 2014. — С.295, 298, 305.

15. ui.pnzgu.ru: [Электронный ресурс]. URL: https://moodle.pnzgu.ru/course/view.php?id=374 (дата обращения: 26.05.2023)

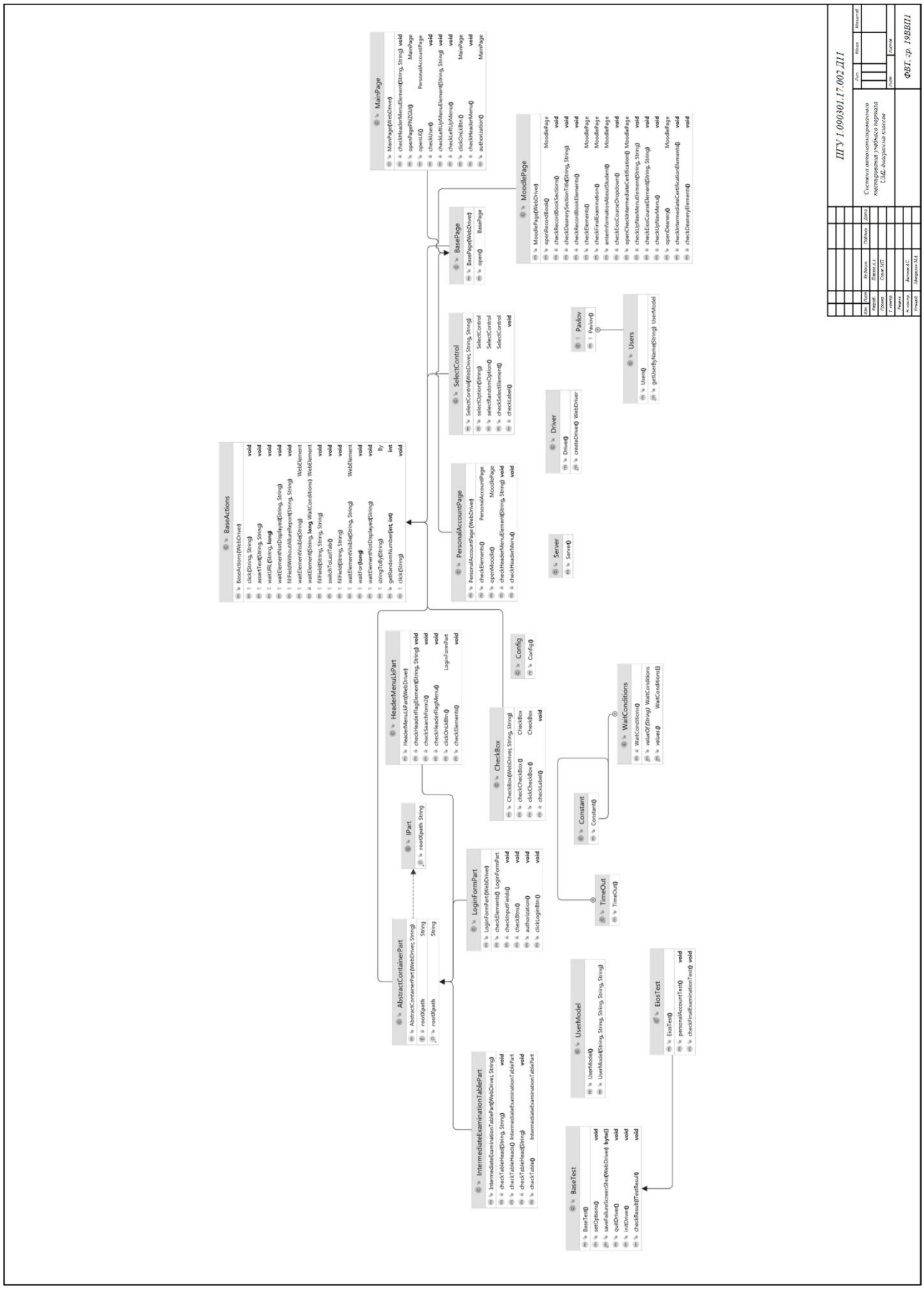
Приложение А

(обязательное)

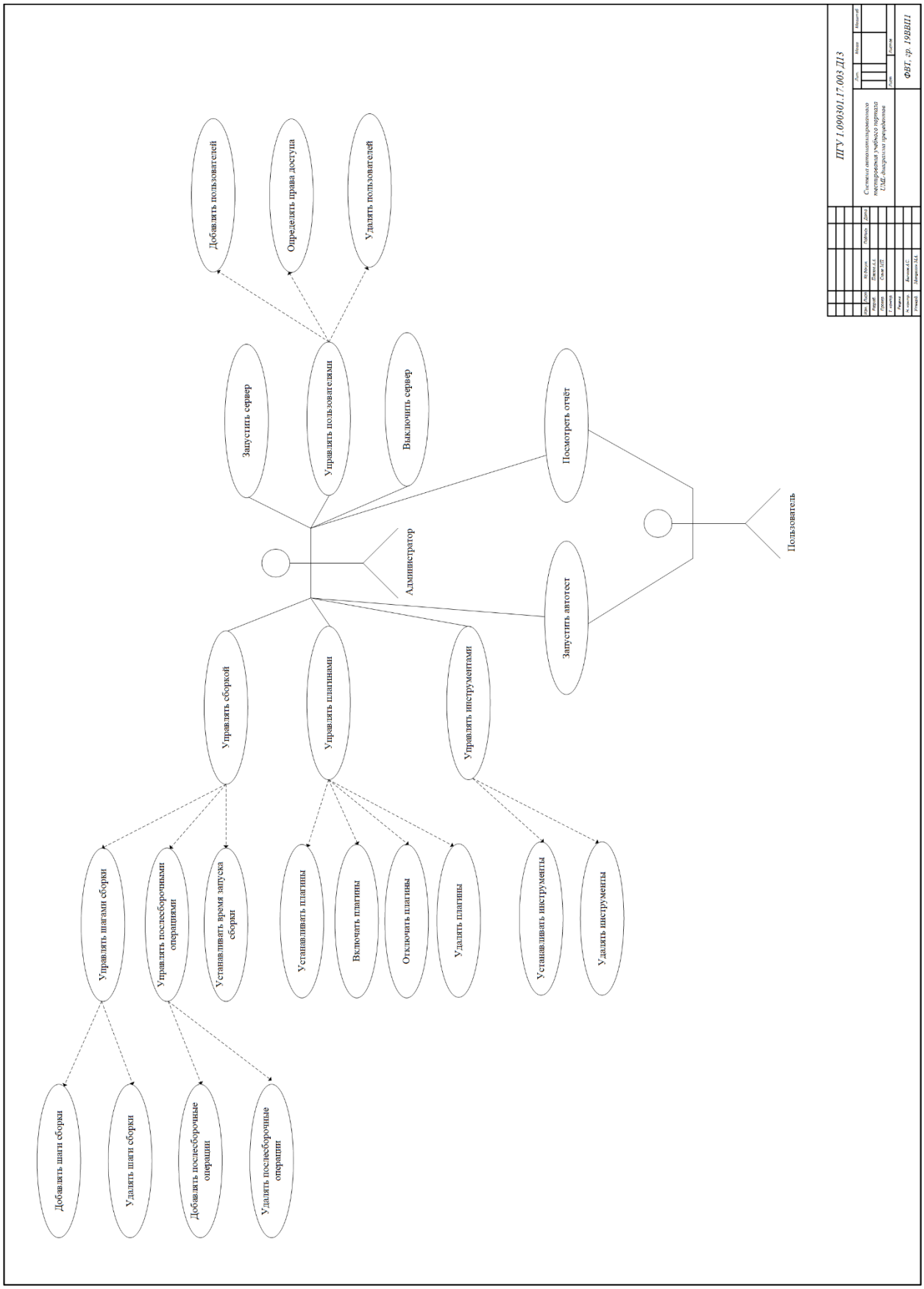
Система автоматизированного тестирования учебного портала

UML диаграммы

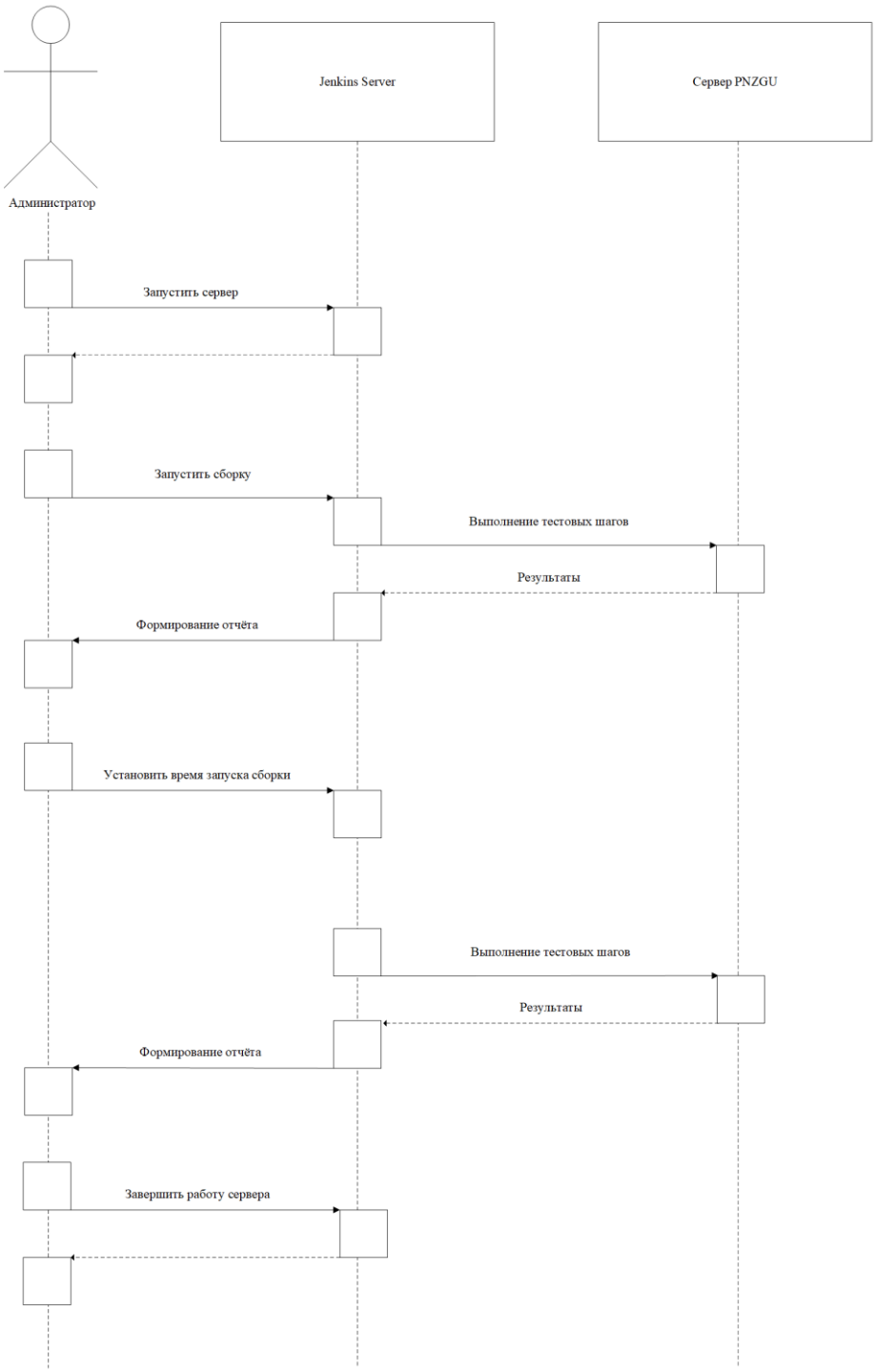
Приложение А.1 – Диаграмма классов



Приложение А.2 – Диаграмма прецедентов

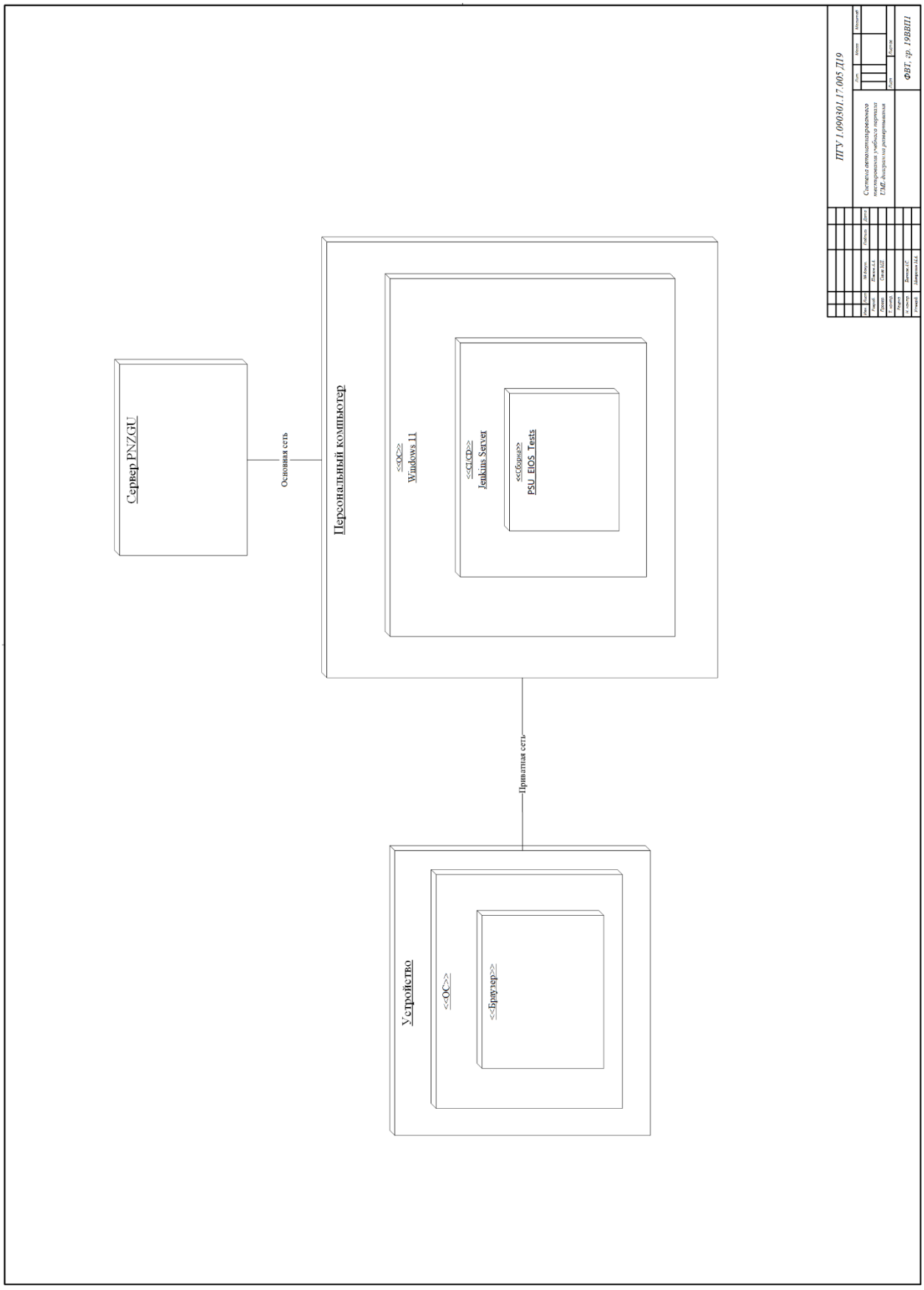


Приложение А.3 – Диаграмма последовательностей



					ПГУ 1.090301.17.004 Д14	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Лапин А.А.			Система автоматизированного тестирования учебного портала	Лит. / Масса / Масштаб
Провер.		Сенов М.П.			UML-диаграмма последовательностей	
Т. контр.						Лист / Листов
Реценз.						
Н. контр.		Витюк А.С.				ФВТ, гр. 19ВВП1
Утверд.		Макаров М.А.				

Приложение А.4 – Диаграмма развертывания



Приложение Б

(обязательное)

Система автоматизированного тестирования учебного портала

Листинг программы

Файл «*BaseActions.java*»

```
package org.psu.base;

import constant.Constant;
import constant.Constant.TimeOut;
import constant.Constant.WaitConditions;
import io.qameta.allure.Step;
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.testng.Assert;


import java.time.Duration;
import java.util.ArrayList;


public class BaseActions {

    protected WebDriver driver;
    private static Logger logger =
LoggerFactory.getLogger(BaseActions.class);

    public BaseActions(WebDriver driver) {
        this.driver = driver;
    }

    protected By stringToBy(String selector) {
```

```
            if (selector.charAt(0) == '#') {
                return By.id(selector.substring(1));
            }
            return By.xpath(selector);
        }

        @Step(value = "Сравнение текста на соответствие «{val}»
= «{expected}»")
        protected void assertText(String val, String expected) {
            Assert.assertEquals(val, expected);
        }

        @Step("Проверка URL {0}")
        protected void waitURL(String URL, long timeout) {
            try {
                new                          WebDriverWait(driver,
Duration.ofSeconds(timeout))

.until(ExpectedConditions.urlContains(URL));
            } catch (TimeoutException timeoutEx) {
                Assert.fail("URL " + URL + " не найден за " +
timeout + " секунд");
            }
        }

        @Step("Ожидание элемента «{titleForAllureOutput}»")
        protected WebElement waitElementVisible(String selector,
String titleForAllureOutput) {
            return waitElementVisible(selector);
        }
```

```
        @Step(value = "Ожидание элемента {0}")
        protected WebElement waitElementVisible(String selector)
{
            return     waitElement(selector,     TimeOut.SHORT,
Constant.WaitConditions.VISIBLE_ELEMENT);
        }

        @Step("Ожидание, что элемент {0} отсутствует")
        protected void waitElementNotDisplayed(String selector)
{
            By by = stringToBy(selector);
            try {
                new                          WebDriverWait(driver,
Duration.ofSeconds(Constant.TimeOut.SHORT))

.until(ExpectedConditions.invisibilityOfElementLocated(by));
            } catch (TimeoutException timeoutEx) {
                Assert.fail("Элемент " + selector + " найден на
странице через " + Constant.TimeOut.SHORT + " секунд");
            }
        }

        @Step("Ожидание,    что    элемент    «{titleForAllure}»
отсутствует")
        protected void waitElementNotDisplayed(String selector,
String titleForAllure) {
            waitElementNotDisplayed(selector);
        }

        private  WebElement  waitElement(String  selector,  long
timeout, WaitConditions condition) {
```

```
            WebElement element = null;
            By by = stringToBy(selector);
            if (condition ==
Constant.WaitConditions.VISIBLE_ELEMENT) {
                try {
                    element = new WebDriverWait(driver,
Duration.ofSeconds(timeout))

.until(ExpectedConditions.visibilityOfElementLocated(by));
                } catch (TimeoutException timeoutEx) {
                    Assert.fail("Элемент " + selector + " не
найден за " + timeout + " секунд");
                }
            } else if (condition ==
Constant.WaitConditions.PRESENCE_ELEMENT) {
                try {
                    element = new WebDriverWait(driver,
Duration.ofSeconds(timeout))

.until(ExpectedConditions.presenceOfElementLocated(by));
                } catch (TimeoutException timeoutEx) {
                    Assert.fail("Элемент " + selector + " не
найден в DOM за " + timeout + " секунд");
                }
            }
            return element;
        }

        @Step("Нажатие на кнопку {0}")
        protected void  click(String selector) {
            By by = stringToBy(selector);
```

```
            driver.findElement(by).click();
        }

        @Step("Нажатие на кнопку «{titleForAllureOutput}»")
        protected void click(String selector, String
titleForAllureOutput){
            click(selector);
        }


        @Step("Заполнение поля {0}")
        protected void fillField(String selector, String text) {
            By by = stringToBy(selector);
            driver.findElement(by).sendKeys(text);
        }

        @Step("Заполнение поля «{titleForAllureOutput}»")
        protected void fillField(String selector, String text,
String titleForAllureOutput) {
            fillField(selector, text);
        }


        protected void fillFieldWithoutAllureReport(String
selector, String text) {
            By by = stringToBy(selector);
            driver.findElement(by).sendKeys(text);
        }
        @Step("Переключение на последнюю открытую вкладку")
        protected void switchToLastTab() {
```

```
            ArrayList<String>         tabs         =         new
ArrayList<>(driver.getWindowHandles());
            driver.switchTo().window(tabs.get(tabs.size() - 1));
        }

        @Step("Ожидание {0} секунд")
        protected void waitFor(long timeOut) {
            if (timeOut > 3) {
                logger.info("Время ожидания больше 3 сек.");
            }
            timeOut *= 1000;
            try {
                Thread.sleep(timeOut);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }

        public int getRandomNumber(int min, int max) {
            return (int) ((Math.random() * (max - min)) + min);
        }
    }
```

Файл «*BasePage.java*»

```
package org.psu.base;

import common.Server;
import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;

public abstract class BasePage extends BaseActions {
```

```
    protected String URL = null;

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

    @Step("Открытие страницы " + Server.PSU)
    public BasePage open(){
        driver.get(Server.PSU);
        return this;
    }
}
```

Файл «*IPart.java*»

```
package org.psu.base;

public interface IPart {
    String getRootXpath();
}
```

Файл «*MainPage.java*»

```
package org.psu;

import common.Server;
import constant.Constant;
import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;
import org.psu.base.BasePage;
import parts.HeaderMenuLkPart;
import parts.LoginFormPart;
import users.Users;

public class MainPage extends BasePage {
    public MainPage(WebDriver driver) {
        super(driver);
    }
    HeaderMenuLkPart headerMenuLkPart = new
HeaderMenuLkPart(driver);
    LoginFormPart loginFormPart;

    @Step("Переход на сайт " + Server.PSU)
    public MainPage openPagePNZGU(){
        open();
        waitURL(Server.PSU, Constant.TimeOut.SHORT);
        waitElementVisible("//div[@class='header-
logo']/a[@title='На главную']",
                "активное лого с функциональной возможностью
перехода «На главную».");
        headerMenuLkPart.checkElements();
        checkLeftUpMenu();
```

```
            checkHeaderMenu();
            waitElementVisible("//div[@class='slider']",
"слайдер новостей");

            return this;
        }

        @Step("Отображение лево верхнего меню навигации")
        private void checkLeftUpMenu(){
            checkLeftUpMenuElement("Поступающим", "/Abitur");
            checkLeftUpMenuElement("Обучающимся", "/studentam");
            checkLeftUpMenuElement("Выпускникам", "/graduates");
            checkLeftUpMenuElement("Сотрудникам", "/employees");
            checkLeftUpMenuElement("Партнерам", "/partners");
            checkLeftUpMenuElement("Прессе",
"http://presscentr.pnzgu.ru/");
            checkLeftUpMenuElement("Институты",
"/portfolio/struct/dep/31429660");
            checkLeftUpMenuElement("Подразделения",
"/sveden/struct");
        }

        @Step("Отображение заголовка перехода «{title}»")
        private void checkLeftUpMenuElement(String title, String
href){

waitElementVisible("//div[@id='left_up_menu']//a[@href='"+   href
+"' and text()='"+ title +"']");
        }

        @Step("Отображение правого меню навигации")
```

```
        private void checkHeaderMenu(){
            checkHeaderMenuElement("Об университете",
"/about_pnzgu");
            checkHeaderMenuElement("Образование",
"/sveden/education");
            checkHeaderMenuElement("Наука",
"http://science.pnzgu.ru/");
            checkHeaderMenuElement("International",
"http://ims.pnzgu.ru/");
            checkHeaderMenuElement("Воспитание",
"http://svr.pnzgu.ru/");
            checkHeaderMenuElement("Здоровье",
"http://kmc.pnzgu.ru/");
            checkHeaderMenuElement("Спорт",
"http://dep_fizvos.pnzgu.ru/fvis_sport");
            checkHeaderMenuElement("Контакты", "/telspr");
        }

        @Step("Отображение заголовка перехода «{title}»")
        private void checkHeaderMenuElement(String title, String
href){
            waitElementVisible("//div[@class='header-
menu']//a[@href='"+ href +"' and text()='"+ title +"']");
        }

        @Step("Нажатие на кнопку «Личный кабинет» в верхнем
правом углу")
        public MainPage clickOnLkBtn() {
            loginFormPart = headerMenuLkPart.clickOnLkBtn();
            loginFormPart.checkElements();
            return this;
```

```
        }

        @Step("Авторизация")
        public MainPage authorization(){
            loginFormPart.authorization();
            loginFormPart.clickLoginBtn();

waitElementNotDisplayed(loginFormPart.getRootXpath(),      "форма
авторизации");
            checkUser();

            return this;
        }

        @Step("Проверка пользователя")
        private void checkUser(){
            String                    FIO                    =
waitElementVisible("//div[@class='header-menu-
lk']//a[@class='link-btn-lk']",
                    "кнопка перехода в ЛК").getText();
            assertText(FIO,
Users.getUserByName(Users.current).surnameAndInitials);
        }

        @Step("Переход в личный кабинет")
        public PersonalAccountPage openLK(){
            click("//div[@class='header-menu-
lk']//a[@class='link-btn-lk']", "переход в ЛК");
            switchToLastTab();
            waitURL("https://lk.pnzgu.ru/portfolio/my",
Constant.TimeOut.SHORT);
```

```
            return                                                  new
PersonalAccountPage(driver).checkElements();
        }
    }
```

Файл «*PersonalAccountPage.java*»

```
package org.psu;

import constant.Constant;
import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;
import org.psu.base.BasePage;
import users.Users;

public class PersonalAccountPage extends BasePage {
    public PersonalAccountPage(WebDriver driver) {
        super(driver);
    }

    @Step("Отображение элементов страницы")
    public PersonalAccountPage checkElements() {
        checkHeaderMenu();
        waitElementVisible("//h1[text()='Портфолио.       "+
Users.getUserByName(Users.current).fio +"']",
                "заголовок          «Портфолио.          "+
Users.getUserByName(Users.current).fio +".» ");
        waitElementVisible("//div[@class='user-
row']/../../div[contains(@class, 'card card-body')]",
                "карточка с информацией о пользователе");
```

```
            return this;
        }

        @Step("Отображение меню навигации")
        private void checkHeaderMenu(){
            checkHeaderMenuElement("Профиль", "/portfolio/my");
            checkHeaderMenuElement("Учебный процесс",
"http://moodle.pnzgu.ru/my/");
            checkHeaderMenuElement("Портфолио", "/portfolio");
            checkHeaderMenuElement("Оплата ONLINE", "/pay");
            checkHeaderMenuElement("Опросы", "/anketa");
            checkHeaderMenuElement("Электронная библиотека",
"http://elib.pnzgu.ru");
            checkHeaderMenuElement("Личный кабинет абитуриента",
"/cabinet");
            checkHeaderMenuElement("Сайт ПГУ",
"https://pnzgu.ru");

        }

        @Step("Отображение кнопки перехода «{title}»")
        private void checkHeaderMenuElement(String title, String
href){

            waitElementVisible("//div[@class='header-
menu']//a[text()='"+ title +"' and @href='"+ href +"']");

        }

        @Step("Открытие кабинета Moodle")
```

```
        public MoodlePage openMoodle(){
            waitElementVisible("//a[text()='Учебный процесс']",
                    "кнопка навигации \"Учебный процесс\"");
            click("//a[text()='Учебный   процесс']",   "Учебный
процесс");
            switchToLastTab();
            waitURL("https://moodle.pnzgu.ru/my/",
Constant.TimeOut.SHORT);
            return new MoodlePage(driver).checkElements();
        }
    }
```

Файл «*MoodlePage.java*»

```
package org.psu;

import constant.Constant;
import elements.CheckBox;
import elements.SelectControl;
import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.psu.base.BasePage;
import parts.IntermediateExaminationTablePart;
import users.Users;

import java.util.List;

public class MoodlePage extends BasePage {
```

```
        private String deaneryBtn = "//li[contains(@class, 'nav-
item')]//a[text()='Эл. деканат']";

        CheckBox autoChoice = new CheckBox(driver, "autoChoice",
"Выбирать текущий курс и семестр автоматически");
        SelectControl groups = new SelectControl(driver,
"group", "Группы");
        SelectControl course = new SelectControl(driver,
"course", "Курс");
        SelectControl semestr = new SelectControl(driver,
"semestr", "Семестр");
        SelectControl discipline = new SelectControl(driver,
"discipline", "Дисциплина");
        SelectControl attestation = new SelectControl(driver,
"attestation", "Вид аттестации");


        IntermediateExaminationTablePart finalExaminationTable =
new                    IntermediateExaminationTablePart(driver,
"//table[@id='vr_cntrlpoints_settings']");

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

        @Step("Отображение элементов страницы")
        public MoodlePage checkElements() {

            checkUpNavMenu();

            return this;
```

```
        }

        @Step("Отображение верхнего меню навигации")
        private void checkUpNavMenu(){
            checkEiosCourseDropdown();
            checkUpNavMenuElement("CHECKUP",
"https://moodle.pnzgu.ru/comp_checkup");
            checkUpNavMenuElement("Кабинет                moodle",
"https://moodle.pnzgu.ru/my/");
            checkUpNavMenuElement("Эл.                   деканат",
"https://moodle.pnzgu.ru/blocks/dof/im/standard/");
            checkUpNavMenuElement("Все                    курсы",
"https://moodle.pnzgu.ru/course/index.php");

        }

        @Step("Отображение элемента навинации «{title}»")
        private void checkUpNavMenuElement(String title, String
href) {
            waitElementVisible("//li[contains(@class,      'nav-
item')]//a[text()='"+ title +"' and @href='"+ href +"']");
        }

        @Step("Отображение выпадающего списка «Курс по ЭИОС» и
его элементов")
        private void checkEiosCourseDropdown(){
            waitElementVisible("//li[contains(@class,  'dropdown
nav-item')]/a[contains(text(), 'Курс по ЭИОС')]",
                    "выпадающий список «Курс по ЭИОС»");
```

```
            click("//li[contains(@class,      'dropdown      nav-
item')]/a[contains(text(), 'Курс по ЭИОС')]",
                        "раскрытие выпадающего списка «Курс по
ЭИОС»");

            checkEiosCourseElement("Курс по ЭИОС",
"https://moodle.pnzgu.ru/course/view.php?id=374");
            checkEiosCourseElement("Инструкции",
"https://moodle.pnzgu.ru/course/view.php?id=374#section-4");
            checkEiosCourseElement("Вопросы регистрации
пользователей",
"https://moodle.pnzgu.ru/mod/forum/view.php?id=4130");
            checkEiosCourseElement("Проект ЭИОС и нормативные
документы",
"https://moodle.pnzgu.ru/course/view.php?id=374#section-2");
            checkEiosCourseElement("Форум для консультаций по
загрузке ресурсов",
"https://moodle.pnzgu.ru/mod/forum/view.php?id=1861");
            checkEiosCourseElement("Форум для консультаций по
работе со студентами",
"https://moodle.pnzgu.ru/mod/forum/view.php?id=134739");
            checkEiosCourseElement("Форум по тестам",
"https://moodle.pnzgu.ru/mod/forum/view.php?id=458842");
            checkEiosCourseElement("Категория ЭИОС",
"https://moodle.pnzgu.ru/course/index.php?categoryid=192");

            click("//li[contains(@class,      'dropdown      nav-
item')]/a[contains(text(), 'Курс по ЭИОС')]",
                        "скрытие выпадающего списка «Курс по
ЭИОС»");
```

```
            waitElementNotDisplayed("//div[contains(@class,
'dropdown-menu show')]", "выпадающий список");


        }
        @Step("Отображение элемента выпадающего списка
«{title}»")
        private void checkEiosCourseElement(String title, String
href){
            waitElementVisible(
                    "//div[contains(@class,       'dropdown-menu
show')]/a[text()='"+ title +"' and @href='"+ href +"']");
        }

        @Step("Открытие электронного деканат")
        public MoodlePage openDeanery(){
            click(deaneryBtn, "Эл. деканат");
            checkDeaneryElements();

            return this;
        }

        @Step("Отображение элементов электронного деканата")
        private void checkDeaneryElements(){
            waitElementVisible("//div[contains(@class,    'page-
header-heading')]//h1[text()='Электронный деканат']",
                    "заголовок «Электронный деканат»");
            checkDeaneriySectionTitle("block_dof_section_my",
"Преподавателям");

checkDeaneriySectionTitle("block_dof_section_student","Учащимся")
;
```

```
checkDeaneriySectionTitle("block_dof_section_settings",
"Настройки");

checkDeaneriySectionTitle("block_dof_section_manager",
"Администрирование");

            waitElementVisible("//div[contains(@class,
'block_dof_section_student')]//a[text()='Зачетная книжка']",
                    "ссылка на раздел «Зачетная книжка»");

        }

        @Step("Отображение секции «{title}»")
        private void checkDeaneriySectionTitle(String className,
String title){
            waitElementVisible("//div[contains(@class,       '"+
className +"')]//strong[text()='"+ title +"']");
        }

        @Step("Открытие зачетной книжки")
        public MoodlePage openRecordBook(){
            click("//div[contains(@class,
'block_dof_section_student')]//a[text()='Зачетная книжка']",
                    "ссылка на раздел «Зачетная книжка»");
            checkRecordBookElements();

            return this;
        }

        @Step("Отображение элементов зачётной книжки")
```

```
    private void checkRecordBookElements(){
        waitElementVisible("//div[contains(@class, 'page-
header-heading')]//h1[text()='Зачетная книжка']",
                "заголовок «Зачетная книжка»");
        checkRecordBookSections();

    }

    @Step("Отображение секций зачётной книжки")
    private void checkRecordBookSections(){
        List<WebElement> sectionTitles =
driver.findElements(stringToBy("//div[@class='block_dof_maintabl
e_center']//a"));

        assertText(sectionTitles.get(0).getText(), "Просмотр
рейтинга");
        assertText(sectionTitles.get(1).getText(), "Просмотр
промежуточной аттестации");
        assertText(sectionTitles.get(2).getText(),
"Настройка дисциплин по выбору");
    }

    @Step("Открытие страницы «Просмотр промежуточной
аттестации»")
    public MoodlePage openCheckIntermediateCertification(){
        click("//a[contains(text(), 'Просмотр
промежуточной')]",
                "ссылка на раздел «Просмотр промежуточной
аттестации»");
        checkIntermediateCertificationElements();
```

```
            return this;
        }

        @Step("Отображение элементов раздела «Просмотр
промежуточной аттестации»")
        private void checkIntermediateCertificationElements(){
            waitElementVisible("//div[contains(@class, 'page-
header-heading')]//h1[text()='Просмотр промежуточной
аттестации']",
                    "заголовок «Просмотр промежуточной
аттестации»");

            autoChoice.checkCheckBox();

            groups.checkSelectElement();
            course.checkSelectElement();
            semestr.checkSelectElement();
            discipline.checkSelectElement();
            attestation.checkSelectElement();

            waitElementVisible("//input[@value='Показать']",
"кнопка «Показать»");


        }

        @Step("Заполнение полей об обучающемся")
        public MoodlePage enterInformationAboutStudent(){

            autoChoice.clickCheckBox();
```

```
            groups.selectRandomOption();
            waitFor(Constant.TimeOut.ONE_SEC);
            course.selectRandomOption();
            waitFor(Constant.TimeOut.ONE_SEC);
            semestr.selectRandomOption();
            waitFor(Constant.TimeOut.ONE_SEC);
            discipline.selectRandomOption();
            waitFor(Constant.TimeOut.ONE_SEC);
            attestation.selectRandomOption();
            waitFor(Constant.TimeOut.ONE_SEC);
            click("//input[@value='Показать']", "Показать");

            return this;

        }

        @Step("Отображение сведений о промежуточной аттестации")
        public MoodlePage checkFinalExamination(){

            waitElementVisible("//legend[contains(text(),
'Журнал аттестации')]//b[contains(text(), 'гр."+

Users.getUserByName(Users.current).group +"')]",
                    "текст «Журнал аттестации гр."+
Users.getUserByName(Users.current).group +"»");


waitElementVisible("//div[@class='div_edit_table']//h2[contains(
text(), 'Выставление промежуточной аттестации')]",
                    "заголовок «Выставление промежуточной
аттестации»");
```

```
            finalExaminationTable.checkTable();

            return this;
        }

    }
```

Файл «*AbstractContainerPart.java*»

```
    package parts;

    import org.openqa.selenium.WebDriver;
    import org.psu.base.BaseActions;
    import org.psu.base.IPart;

    public class AbstractContainerPart extends BaseActions
implements IPart {

        private String rootXpath = null;

        public AbstractContainerPart(WebDriver driver, String
rootXpath) {
            super(driver);
            this.rootXpath = rootXpath;
        }

        @Override
        public String getRootXpath() {
            return rootXpath;
        }
    }
```

Файл «*HeaderMenuLkPart.java*»

```
package parts;

import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;
import org.psu.base.BaseActions;

public class HeaderMenuLkPart extends AbstractContainerPart
{

        private static final String rootXpath =
"//div[@class='header-menu-lk']";

        private String educationalOrganizationInfo  = rootXpath
+ "//a[text()='Сведения об образовательной организации']";
        private String btnLK = rootXpath + "//div[@class='link-
btn-lk']";
        private String searchForm2 = rootXpath +
"//form[@name='serch_form2']"; //TODO: serch
        public HeaderMenuLkPart(WebDriver driver) {
            super(driver, rootXpath);
        }

        @Step("Отображения элементов меню личного кабинета")
        public void checkElements(){
            waitElementVisible(educationalOrganizationInfo,
"гиперссылка «Сведения об образовательной организации»");
            waitElementVisible(btnLK, "кнопка «Личный
кабинет»");
            checkSearchForm2();
```

```
        }

        @Step("Отображение поисковой формы")
        private void checkSearchForm2(){
            waitElementVisible(searchForm2 + "/input", "поле
ввода");
            waitElementVisible(searchForm2 + "/button", "кнопка
поиска");
            checkHeaderFlagMenu();

        }

        @Step("Отображение меню иконок с функциональной
возможностью смены версии сайта")
        private void checkHeaderFlagMenu() {
            waitElementVisible("//div[@class='header-menu-lk']"
+
                    "//div[@class='header-
flag']/a[@title='Версия сайта для слабовидящих' and @href='#']",
                    "иконка «Версия сайта для слабовидящих»");
            checkHeaderFlagElement("Русская версия сайта",
"http://pnzgu.ru");
            checkHeaderFlagElement("Международная версия сайта",
"http://international.pnzgu.ru");
            checkHeaderFlagElement("Немецкая версия сайта",
"http://de.pnzgu.ru");
            checkHeaderFlagElement("Китайская версия сайта",
"http://ch.pnzgu.ru");
        }

        @Step("Отображение иконки версии сайта «{title}»")
```

```
        private void checkHeaderFlagElement(String title, String
href){
            waitElementVisible("//div[@class='header-menu-
lk']//div[@class='header-flag']" +
                    "/a[@href='" + href + "']/img[@title='" +
title + "']");
        }

        @Step("Нажатие кнопки «Личный кабинет»")
        public LoginFormPart clickOnLkBtn() {
            click(btnLK);
            return new LoginFormPart(driver);
        }


    }
```

Файл «*LoginFormPart.java*»

```
    package parts;

    import constant.Constant;
    import users.Users;
    import io.qameta.allure.Step;
    import org.openqa.selenium.WebDriver;

    public class LoginFormPart extends AbstractContainerPart {

        private static final String rootXpath =
"//div[@class='login-form' and @style='display: block;']";

        private String loginInput = rootXpath +
"//input[@name='name']";
```

```
        private String passwordInput = rootXpath +
"//input[@id='password']";

        private String loginBtn = rootXpath +
"//input[@value='ВОЙТИ']";
        private String forgotPassBtn = rootXpath +
"//a[text()='Забыли пароль?' and
@href='https://lk.pnzgu.ru/users/recoveryform']";
        private String signUpBtn = rootXpath +
"//a[text()='РЕГИСТРАЦИЯ АБИТУРИЕНТА' and
@href='https://lk.pnzgu.ru/cabinet/registration']";

        public LoginFormPart(WebDriver driver) {
            super(driver, rootXpath);
        }
        @Step("Отображение элементов формы авторизации")
        public LoginFormPart checkElements(){
            waitElementVisible(rootXpath +
"/h1[text()='Авторизация на сайте']",
                    "текстовая подсказка «Пожалуйста, заполните
поля для входа»");
            checkInputFields();
            checkBtns();

            return this;
        }

        @Step("Отображение полей ввода")
        private void checkInputFields(){
            waitElementVisible(loginInput, "поле ввода логина");
```

```
            waitElementVisible(passwordInput,     "поле     ввода
пароля");

        }

        @Step("Отображение кнопок")
        private void checkBtns(){
            waitElementVisible(loginInput, "кнопка «ВОЙТИ»");
            waitElementVisible(forgotPassBtn,   "кнопка   «ЗАБЫЛИ
ПАРОЛЬ?»");
            waitElementVisible(signUpBtn,  "кнопка  «РЕГИСТРАЦИЯ
АБИТУРИЕНТА»");
        }
        @Step("Ввод данных")
        public void authorization(){

            fillFieldWithoutAllureReport(loginInput,
Users.getUserByName(Users.current).login);
            fillFieldWithoutAllureReport(passwordInput,
Users.getUserByName(Users.current).password);

        }

        public void clickLoginBtn(){

            click(loginBtn, "Войти");

        }

    }
```

Файл «*IntermediateExaminationTablePart.java*»

```
package parts;

import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;

public class IntermediateExaminationTablePart extends
AbstractContainerPart {
    public IntermediateExaminationTablePart(WebDriver
driver, String rootXpath) {
        super(driver, rootXpath);
    }
    @Step("Отображение таблицы промежуточной аттестации")
    public IntermediateExaminationTablePart checkTable() {

        checkTableHeads();

        return this;
    }

    @Step("Отображение заголовков таблицы")
    public IntermediateExaminationTablePart
checkTableHeads() {

        checkTableHead("№ п/п");
        checkTableHead("ФИО студента");
        checkTableHead("Контрольная точка 1");
        checkTableHead("Добор КТ1");
        checkTableHead("Контрольная точка 2");
        checkTableHead("Добор КТ2");
        checkTableHead("КМ после КТ");
```

```
            checkTableHead("Доп. баллы");
            checkTableHead("R", "тек");
            checkTableHead("КМ сданы");
            checkTableHead("Допуск деканат");
            checkTableHead("R", "атт");
            checkTableHead("R", "дис");
            checkTableHead("Оценка");

            return this;
        }

        @Step("Отображение заголовка таблицы «{text}»")
        private void checkTableHead(String text) {

waitElementVisible("//table//thead//*[contains(text(),  '"+  text
+"')]");

        }

        @Step("Отображение заголовка таблицы «{text}» с верхним
индексом «{supText}»")
        private void checkTableHead(String text, String supText)
{

waitElementVisible("//table//thead//*[contains(text(),  '"+  text
+"')]/sup[contains(text(), '"+ supText +"')]");

        }
    }
```

Файл «*SelectControl.java*»

```
package elements;


import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.psu.base.BaseActions;
import org.testng.Assert;


import java.util.List;


public class SelectControl extends BaseActions {


    private String rootXpath;
    private String id;
    private String label;



    public SelectControl(WebDriver driver, String id, String
label) {
        super(driver);
        this.id = id;
        this.label = label;
        this.rootXpath = "//div[@id='fitem_"+ id +"']";
    }


    @Step("Проверка селект элемента «{this.label}»")
    public SelectControl checkSelectElement(){
        checkLabel();
        waitElementVisible(rootXpath +  "//select");
```

```
        return this;
    }

    @Step("Отображение лейбла «{this.label}»")
    private void checkLabel(){
        waitElementVisible("//div[@id='fitem_"+ id
+"']//label[contains(text(), '"+ label +"')]");
    }

    @Step("Выбор в селект элементе «{this.label}» опции
«{option}»")
    public SelectControl selectOption(String option) {

        click(rootXpath + "//select");
        waitElementVisible(rootXpath +
"//option[contains(text(), '"+ option + "')]");
        click(rootXpath + "//option[contains(text(), '"+ option
+ "')]");
        return this;
    }

    @Step("Выбор в селект элементе случайной опции")
    public SelectControl selectRandomOption() {

        List<WebElement> options =
driver.findElements(stringToBy("//select[@id='"+ id
+"']//option"));
        int randomNum = getRandomNumber(2, options.size());
        String randomOption =
driver.findElement(stringToBy(rootXpath + "//select[@id='"+ id
+"']//option["+ randomNum +"]")).getText();
```

```
        if (options.size() == 1 &&
randomOption.contains("Выберите"))
            Assert.fail("В селект элементе отсутсвуют другие
опции кроме «"+ randomOption +"»");

        selectOption(randomOption);

        return this;
```

Файл «*CheckBox.java*»

```
package elements;
import io.qameta.allure.Step;
import org.openqa.selenium.WebDriver;
import org.psu.base.BaseActions;

public class CheckBox extends BaseActions {

    private String rootXPath;
    private String label;

    public CheckBox(WebDriver driver, String id, String label) {
        super(driver);

        this.rootXPath = "//input[@type='checkbox' and @id="'+ id +"']";
        this.label = label;
    }

    @Step("Проверка чекбокса «{this.label}»")
    public CheckBox checkCheckBox(){
```

```
        checkLabel();
        waitElementVisible(rootXPath);

        return this;
    }

    @Step("Отображение лейбла «{this.label}»")
    private void checkLabel(){
        waitElementVisible(rootXPath + "/..//label[contains(text(), '"+ label
+"')]");

    }

    @Step("Нажатие на чекбокс «{this.label}»")
    public CheckBox clickCheckBox(){
        waitElementVisible(rootXPath);
        click(rootXPath);
        return this;
```

Файл «*UserModel.java*»

```
package models;
public class UserModel {

    public UserModel() {

    }
```

```
    public String login = null;
    public String password = null;
    public String surnameAndInitials = null;

    public String fio = null;
    public String group = null;

    public UserModel(String login, String password, String surnameAndInitials, String fio, String group) {
        this.login = login;
        this.password = password;
        this.surnameAndInitials = surnameAndInitials;
        this.fio = fio;
        this.group = group;
    }
```

Файл «*Users.java*»

```
package users;

import models.UserModel;

public class Users {

    public static String current = null;
    public static final String pavlov = "Павлов А.А.";

    public static UserModel getUserByName(String userName) {

        switch (userName) {
```

```
                case pavlov:

                    return new UserModel(Pavlov.login,
                            Pavlov.password,
                            Pavlov.surnameAndInitials,
                            Pavlov.FIO,
                            Pavlov.group);


                default:
                    return new UserModel();
            }

        }
         protected class Pavlov {
            public static final String login = "*******";
            public static final String password = "*******";
            public  static  final  String  surnameAndInitials  =
"Павлов А.А.";
            public   static   final   String   FIO   =   "Алексей
Александрович Павлов";
            public static final String group = "19ВВП1";
        }

    }
```

Файл «*Constant.java*»

```
package constant;
public class Constant {
```

```
        public static class TimeOut {
            public static final long ONE_SEC = 1;
            public static final long SHORTEST = 5;
            public static final long SHORT = 10;
            public static final long MEDIUM = 15;
            public static final long HALF_MINUTE = 30;
            public static final long MINUTE = 60;
            public static final long LONG = 180;
        }

        public enum WaitConditions {
            VISIBLE_ELEMENT,
            PRESENCE_ELEMENT,
        }

    }
```

Файл «*Config.java*»

```
package common;

public class Config {

    public static final String webDriverPath =
"C:\\tools\\chromedriver\\chromedriver.exe";
    public static Boolean CLOSE_BROWSER = true;

}
```

Файл «*Driver.java*»

```
package common;
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class Driver {

    public static WebDriver createDriver() throws Exception {
        WebDriver driver;

        System.setProperty("webdriver.chrome.driver",
Config.webDriverPath);

        ChromeOptions options = new ChromeOptions();
        options.addArguments("--window-size=1920,1080");
        options.addArguments("--test-type");
        options.addArguments("--ignore-certificate-errors");
        options.addArguments("--remote-allow-origins=*");

        driver = new ChromeDriver(options);
        driver.manage().window().maximize();

        return driver;
    }
}
```

Файл «*Server.java*»

```
package common;

public class Server {
    public static final String PSU = "https://www.pnzgu.ru/#";

}
```

Файл «*BaseTest.java*»

```
package base;

import common.Config;
import common.Driver;
import io.qameta.allure.Attachment;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import users.Users;
public class BaseTest {
    protected WebDriver driver = null;

    @BeforeClass
    public void setOptions() {
        Users.current = Users.pavlov;
        Config.CLOSE_BROWSER = true;
    }

    @BeforeMethod
    public void initDriver() {
        if (driver == null) {
            try {
                driver = Driver.createDriver();
            } catch (Exception e) {
                e.printStackTrace();
```

```
            }
        }
    }

    @AfterMethod
    public void checkResult(ITestResult result){
        if (result.getStatus() == ITestResult.FAILURE) {
            saveFailureScreenShot(driver);

            if(Config.CLOSE_BROWSER){
                driver.quit();
                driver = null;
            }

        }
    }

    @AfterClass
    public void quitDriver(){
        if (driver != null) {
            if(Config.CLOSE_BROWSER) {
                driver.quit();
                driver = null;
            }
        }
    }

    @Attachment(value = "Screenshot", type = "image/png")
    public static byte[] saveFailureScreenShot(WebDriver driver)
{
        return
((TakesScreenshot)driver).getScreenshotAs(OutputType.BYTES);
```

Файл «*EiosTest.java*»

```
import base.BaseTest;

import io.qameta.allure.Epic;

import org.psu.MainPage;

import org.psu.MoodlePage;

import org.psu.PersonalAccountPage;

import org.testng.annotations.Test;


@Epic("ПГУ")

public class EiosTest extends BaseTest {

    @Test(description = "Тестирование перехода в личный кабинет
обучающегося", priority = 1)

    public void personalAccountTest(){

        MainPage mainPage = new MainPage(driver);

        mainPage.openPagePNZGU()

                .clickOnLkBtn()

                .authorization()

                .openLK();


    }


    @Test(description = "Тестирование просмотра промежуточной
аттестации через электронный деканат",

            dependsOnMethods={"personalAccountTest"})

    public void checkFinalExaminationTest(){

        PersonalAccountPage personalAccountPage = new
PersonalAccountPage(driver);

        MoodlePage moodlePage =
personalAccountPage.openMoodle();

        moodlePage.openDeanery()

                .openRecordBook()

                .openCheckIntermediateCertification()
```

```
                .enterInformationAboutStudent()
                .checkFinalExamination();
    }

}
```

Файл «*allure.properties*»

```
allure.results.directory=target/allure-results
```

Файл «*testing.xml*»

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">



<suite name="Tests" parallel="classes" thread-count="1">
    <test name="Test PSU" group-by-instances="true">
        <classes>
            <class name="EiosTest"></class>
        </classes>
    </test>
</suite>
```

Файл «*pom.xml*»

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>


    <groupId>org.psu</groupId>
    <artifactId>PSU_EIOS</artifactId>
```

```
    <version>1.0-SNAPSHOT</version>


    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>


        <aspectj.version>1.8.10</aspectj.version>
        <maven-surefire-plugin.version>2.20</maven-surefire-
plugin.version>
        <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
        <suiteXml>testng.xml</suiteXml>
    </properties>


    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.3.0</version>
        </dependency>
        <dependency>
            <groupId>io.qameta.allure</groupId>
            <artifactId>allure-testng</artifactId>
            <version>2.17.3</version>
        </dependency>
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.5</version>
        </dependency>
```

```
    </dependencies>


    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.20</version>
                <configuration>
                    <argLine>
                        -
javaagent:"${settings.localRepository}/org/aspectj/aspectjweaver
/${aspectj.version}/aspectjweaver-${aspectj.version}.jar"
                    </argLine>
                    <suiteXmlFiles>

<suiteXmlFile>src/main/resources/${suiteXml}</suiteXmlFile>
                    </suiteXmlFiles>
                </configuration>
                <dependencies>
                    <dependency>
                        <groupId>org.aspectj</groupId>
                        <artifactId>aspectjweaver</artifactId>
                        <version>${aspectj.version}</version>
                    </dependency>
                </dependencies>
            </plugin>
            <plugin>
                <groupId>io.qameta.allure</groupId>
                <artifactId>allure-maven</artifactId>
                <version>2.10.0</version>
            </plugin>
```

```
        </plugins>
    </build>

</project>
```

Приложение В

(Обязательное)

Система автоматизированного тестирования учебного портала

Презентация

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.17.001 ПЗ

Лист 117

тема автоматизированного тестирова
учебного портала

работы:
студент гр. 19ВВП1 Павлов Алексей Александрович
ий руководитель:
к.т.н., доцент каф. ВТ Синев Михаил Петрович

Пенза, 2023

Цель и объект работы

ы: разработать программную систему для реа
льных проверок учебного портала ЭИОС ПГУ

боты: система автоматизированного тестирова
яющая следующие возможности:

функционала электронно-информационно
ельной среды ФГБОУ ВО ПГУ;

ю отчёта с логом выполненных действий и их
ами;

времени автоматического запуска тестов;

ие по электронной почте о результатах сборки

ЬНОСТЬ

В случае системы уп
ЭИОС автотесты нуж
проверка устоявшего
критического функци

Такие сценарии кард
изменяются, но треб
постоянной оценки
работоспособности.

Используемые технологии

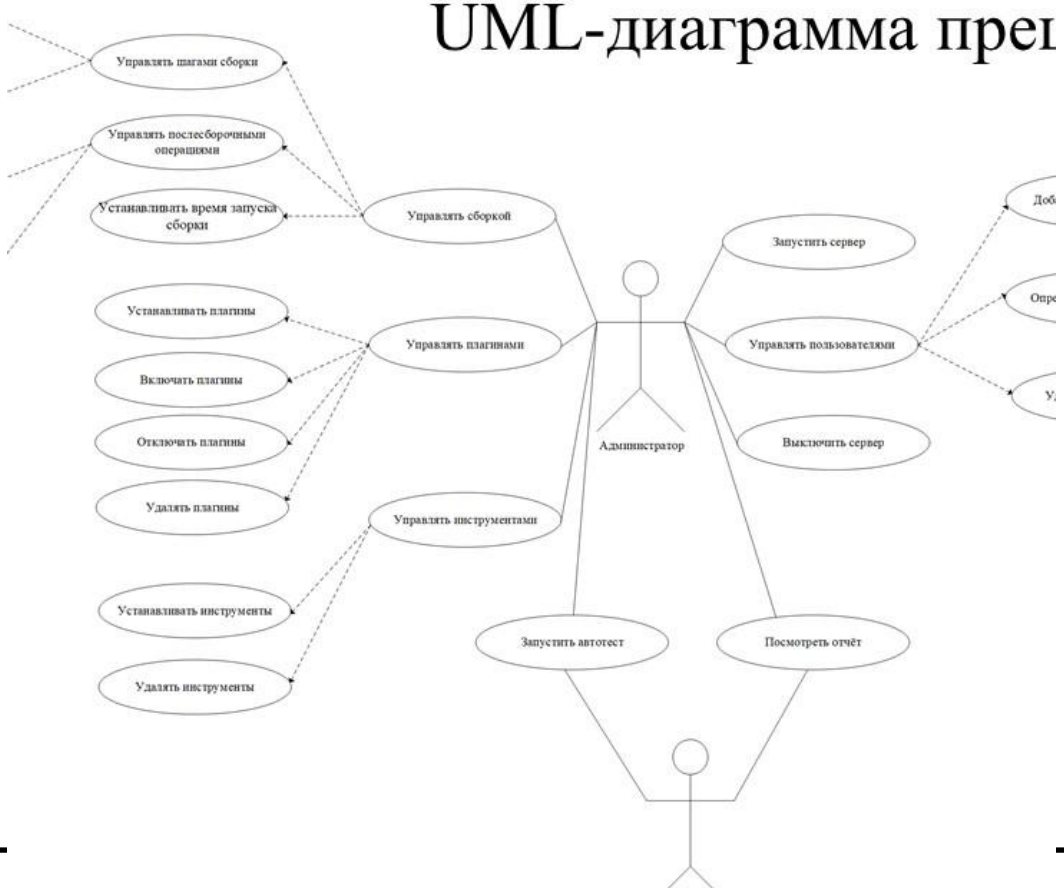
IntelliJ IDEA

Jenkins

Maven

SM

UML-диаграмма пре



ния по электронной почте

nail

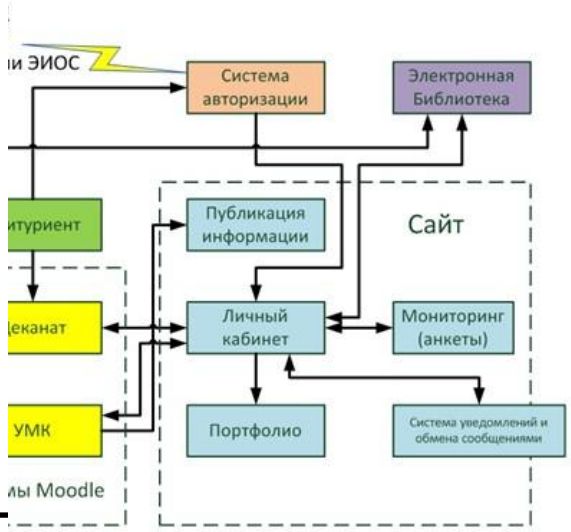
TP Server

- SMTP - *Simple Mail Transfer P*
  простой протокол передачи
- SSL расшифровывается как *Se*
  Layer, или уровень защищенн

- SMTP-сервер: smtp.gmail.com
- Порт: 465 (для SSL)

емая система

ЭИОС ПГУ



Составление плана теста
изучение требований и д

Требования к программн
обеспечению и другие до
связанные с ЭИОС ПГУ,
системе управления курс
разделе «Курс по ЭИОС»

В разделе приведены мат
проекте разработки ЭИС
целом, а также инструкц
эксплуатации.

реализации тестовых сценар

			• Логин • Пароль 3) Кнопки • ВОЙТИ • ЗАБЫЛ • РЕГИСТ АБИТУ
3	Ввести в поля ввода верные исходные данные. Нажать на кнопку «Войти».	Логин: Пароль:	Форма авториз правом верхне кнопки «Личн на фамилию и «Павлов А.А.»
4	Нажать на кнопку с фамилией и инициалами пользователя «Павлов А.А.».		В новой вклад кабинет польз Отображено: 1) Меню навиг • Профи • Учебн • Портф • Оплата • Опрос • Электр • Личны абитур • Сайт П • Админ 2) Заголовок « Александрович 3) Карточка с пользователе.

Описание	Проверка требований к реализации визуализации и управлени реализованными на стороне Backend service в части экранной

	я с существующими логином и паролем
Приоритет:	Высокий
Ответственный:	Павлов Алексей Александрович

ные	Ожидаемый результат
	Открыта страница «Пензенский государственный университет». Отображено: 1) Активное лого с функциональной возможностью перехода «На главную». 2) Меню, содержащее: • гиперссылку «Сведения об образовательной организации» • кнопку «Личный кабинет» • Поисковую строку • Меню иконок с функциональной возможность смены версии сайта: o Для слабовидящих o Русская o Международная o Немецкая o Китайская 3) Меню навигации с заголовками: • Поступающим • Обучающимся • Выпускникам • Сотрудникам • Партнерам • Прессе • Институты • Подразделения 4) Меню навигации с заголовками: • Об университете • Образование • Наука • International • Воспитание • Здоровье • Спорт • Контакты 5) Слайдер новостей
	Открылась форма «Авторизация на сайте», на которой отображено: 1) Текстовая подсказка «Пожалуйста,

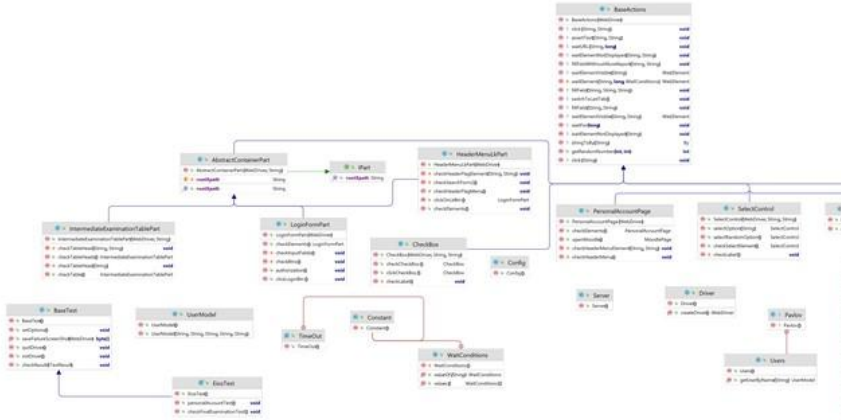
ы реализации тестовых сценар

тр «Промежуточной аттестации»		
	Приоритет:	Средний
сей ич	**Ответственный:**	Павлов Алексей Александрович
ОС-1»		

Шаги	Данные	Ожидаемый результат
навигации нажать на Учебный процесс»		В новой вкладке открылся кабинет moodle. Отображено меню навигации: • Выпадающий список «Курс по ЭИОС» с элементами: o Курс по ЭИОС o Инструкции o Вопросы регистрации пользователей o Проект ЭИОС и нормативные документы o Форумы для консультаций по загрузке ресурсов o Форумы для консультаций по работе со студентами o Форумы по тестам o Категории ЭИОС • CHECKUP • Кабинет moodle • Эл. Деканат • Все курсы
навигации нажать на Эл. деканат»		Открылась страница «Электронный деканат». Отображены заголовки: • Преподавателям • Учащимся • Настройки • Администрирование Под заголовком «Учащимся» отображена гиперссылка «Зачетная книжка».

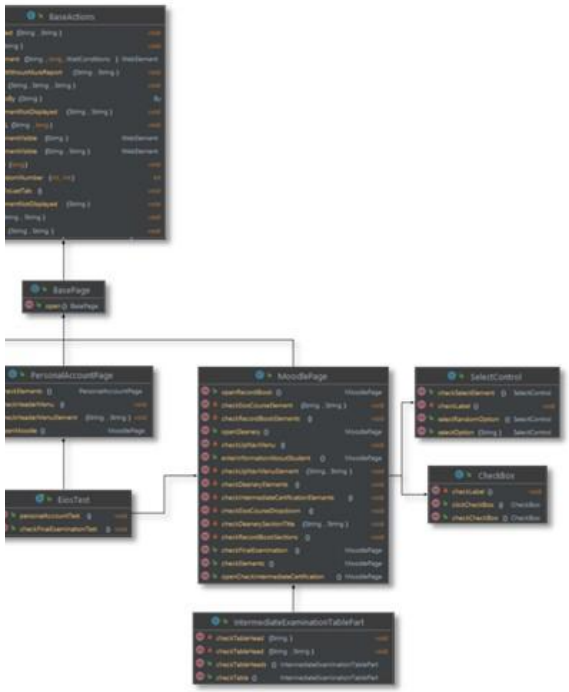
	3	Нажать на гиперссылку «Зачетная книжка»	Открылась страница «Зачет книжка». Отображены гиперссылки: • Просмотр рейтинга • Просмотр промежуточной аттестации • Настройка дисципл выбору
	4	Нажать на гиперссылку «Просмотр промежуточной аттестации»	Открылась страница «Просм промежуточной аттестации» Отображены: 1) Чекбокс «Выбирать текущ курс и семестр автоматичес 2) Селект элементы: • Группы • Курс • Семестр • Дисциплина • Вид аттестации
	5	Деактивировать чекбокс «Выбирать текущий курс и семестр автоматически» и выбрать в полях селект элементов случайные опции	Чекбокс деактивирован. Пол заполнены. Кнопка «Показат активна.
	6	Нажать на кнопку «Показать»	В нижней части экрана отоб «Журнал аттестации [группа обучающегося]» с таблицей «Выставление промежуточн аттестации». Таблица имеет заголовки: • № п/п • ФИО студента • Контрольная точка • Добор КТ1 • Контрольная точка • Добор КТ2 • КМ после КТ • Доп. баллы • R*** • КМ сданы • Допуск деканат • R** • R*** • Оценка
Описание	Проверка требований к реализации визуализации и управлению бизнес-функциями, реализованными на стороне Backend service в части экранной формы «Просмотр промежуточной аттестации» на странице «Электронного деканата»		

UML-диаграмма классов

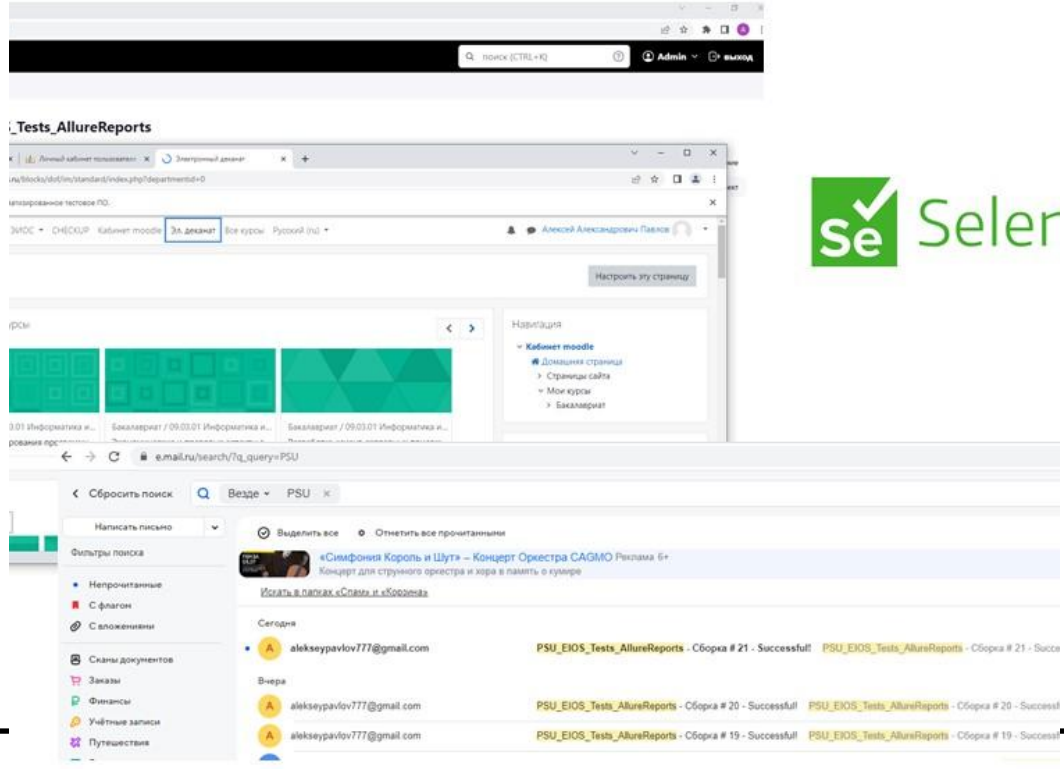


Реализация паттерна Page Object

ование паттерна Steps

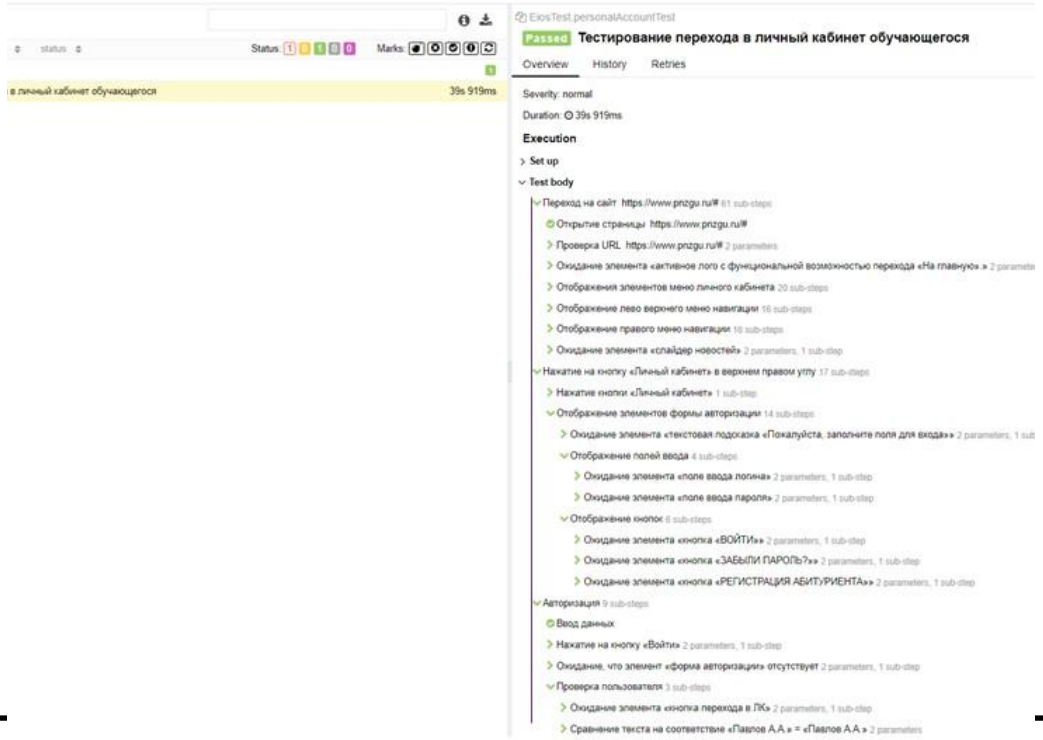


Процесс тестирования

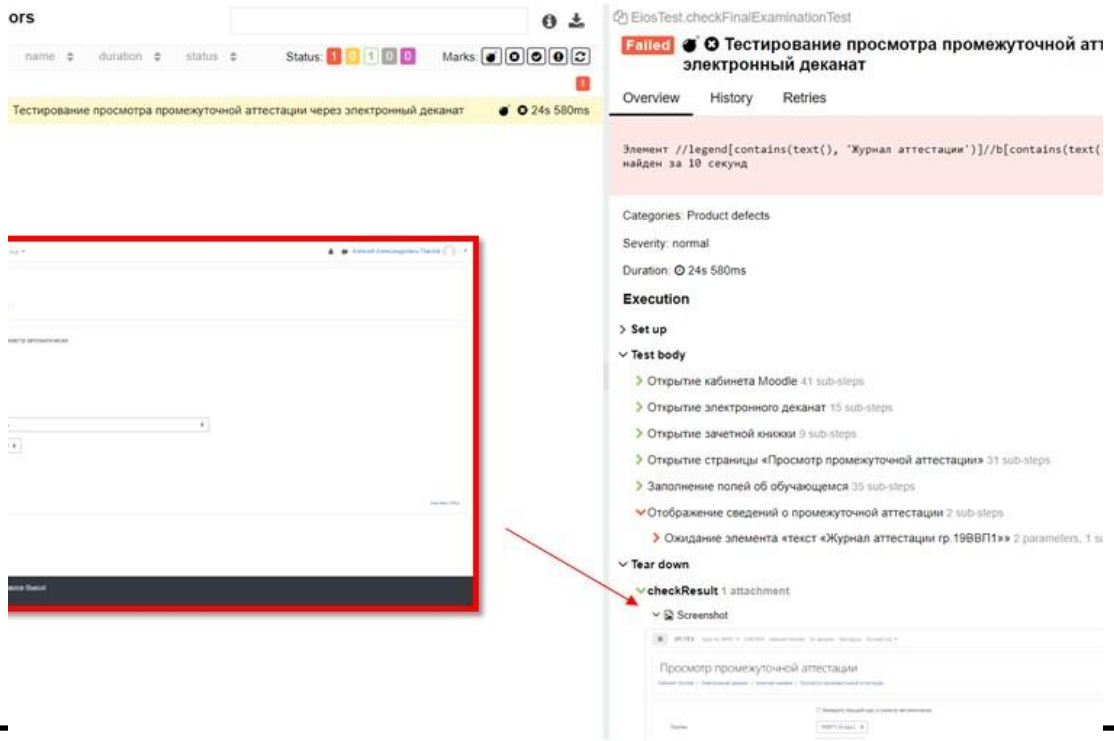


Seler

Успешно пройденный сценарий



Проваленный сценарий



Заключение

ла система автоматизированного тестирования,
ющая функциональную проверку учебного п
у.

редусматривает возможность отложенного зап
полняет виртуально действия пользователя и г
оведенных проверках, а также оповещает поль
к по электронной почте.

Благодарю за внимание!