**ВВЕДЕНИЕ**

На данный момент в сфере тестирования большинство операций осуществляется вручную специалистами по тестированию программного обеспечения (ПО). Процесс функционального тестирования затрагивает анализ требований, составление тестовой документации, осуществление тестирования, обнаружение и описание дефектов.

Такой жизненный цикл тестирования занимает много времени у исполнителей и требует сильной концентрации внимания. Для решения этой проблемы на современных проектах вводится автоматизация тестирования. Роль специалиста по автоматизации тестирования относительно новая и инновационная в индустрии информационных технологий.

Автоматизация тестирования играет ключевую роль в обеспечении качества программных продуктов. Она позволяет проводить тесты быстрее, эффективнее и точнее, чем это возможно при ручном тестировании, также обеспечивает избавление от человеческого фактора в процессе поиска дефектов, возможность переиспользования действий без затрат сил человека, а также гарантирует наглядную отчетность.. Это особенно важно в условиях современных методологий разработки, таких как Agile (гибкая методика разработки) и DevOps (методология разработки и эксплуатации), где скорость и гибкость являются критическими факторами успеха.

Также проблемой в этой области является  
отсутствие реализации тестовых сценариев, выходящих за рамки браузера. Например, отсутствуют инструменты для работы с файловыми менеджерами, проверки корректной загрузки файлов, их названий и расширений, отсутствие возможности визуального сравнения изображений и т.д.

Дипломный проект, темой которого является разработка фреймворка для автоматизации тестирования и его эргономическое обеспечение, будет выполнен в виде удобного и современного фреймворка, решающего вышеуказанные проблемы, который можно подключить в проект в качестве внешней библиотеки.

При разработке фреймворка был использован язык программирования C# версии 12.0, платформа для разработчиков .NET 8 и Selenium WebDriver в качестве основного ядра.

Пояснительная записка имеет следующую структуру:

Здесь будет описана каждая глава диплома. Пока что это не пишу, т.к. не все главы завершены.

1 ИНФОРМАЦИОННЫЕ СИСТЕМЫ В СФЕРЕ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ

**1.1 Анализ предметной области**

Автоматизация тестирования – это мощный инструмент, который может оказать положительное влияние на бизнес. Рассмотрим, как автоматизация тестирования способствует целям бизнеса:

1. Улучшение качества продукта: Автоматизированные тесты позволяют выявлять дефекты и ошибки в программном продукте более быстро и точно. Это помогает улучшить качество продукта, что в свою очередь способствует удовлетворенности клиентов и повышению лояльности.
2. Сокращение времени релиза: Автоматизация позволяет проводить тестирование быстрее и эффективнее. Это сокращает время, необходимое для выпуска новых версий продукта на рынок. Быстрый релиз может быть критически важным для бизнеса, особенно в быстро меняющейся среде.
3. Экономия ресурсов: Ручное тестирование требует большого количества времени и участия специалистов. Автоматизация позволяет сократить затраты на тестирование, так как тесты могут выполняться автоматически без постоянного вмешательства человека.
4. Повышение надежности: Автоматизированные тесты могут выполняться в любое время, что позволяет выявлять проблемы даже до того, как они повлияют на пользователей. Это способствует повышению надежности продукта.
5. Снижение рисков: Автоматизация позволяет проводить тестирование в различных сценариях, включая крайние случаи и нагрузочное тестирование. Это помогает выявить потенциальные проблемы и снизить риски для бизнеса.
6. Быстрое обнаружение проблем: Автоматизированные тесты могут запускаться после каждого изменения кода. Это позволяет быстро обнаруживать проблемы и устранять их на ранних этапах разработки.

В конечном итоге, автоматизация тестирования помогает бизнесу достичь своих целей, обеспечивая высокое качество продукта, сокращение времени релиза и экономию ресурсов

Разработка фреймворка для автоматизации тестирования – это важный процесс, который обеспечивает эффективное и надежное тестирование программного обеспечения. Вот основные этапы и объекты, которые следует учесть при создании такого фреймворка:

1. Определение объема тестирования:

* Анализ требований и функциональности продукта.
* Определение, какие части продукта будут подвергаться автоматизированному тестированию.

1. Выбор инструментов автоматизированного тестирования:

* Выбор языка программирования (например, C#, Java).
* Использование фреймворков (например, Selenium, XUnit, NUnit).

1. Планирование автоматизированных тестов:

* Создание плана тестирования, включая сценарии и тест-кейсы.
* Определение приоритетов и распределение ресурсов.

1. Разработка дизайна фреймворка:

* Создание архитектуры фреймворка.
* Разработка модулей для управления тестами, данных, отчетами и логами.

1. Разработка тестовых сценариев:

* Написание автоматизированных тестовых сценариев.
* Включение проверок, ассертов и ожиданий.

1. Настройка среды запуска автоматизированных тестов:

* Установка и настройка инструментов (например, Selenium WebDriver).
* Создание конфигурационных файлов.

1. Запуск разработанных автоматизированных тестов и анализ результатов:

* Запуск тестов в автоматическом режиме.
* Анализ результатов, выявление ошибок и дефектов.

1. Поддержка автоматизированных тестов:

* Обновление тестов при изменениях в продукте.
* Поддержка и оптимизация фреймворка

При разработке фреймворка для автоматизации тестирования, максимум внимания требуется уделить взаимодействию каждого модуля между собой, потому что основная потребность пользователей подобной системы – это объединить возможности нескольких инструментов, предназначенных для обеспечения качества через тестирование, в одном единственном месте – в данном случае, в едином цельном фреймворке.

В этом, как раз, и заключается цель данного дипломного проекта – разработка фреймворка для автоматизации тестирования и осуществление анализа преимуществ, которые он может принести как специалистам по тестированию, так и жизненному циклу разработки ПО в целом.

Таким образом, существует целесообразность разработки нового фреймворка для автоматизации тестирования веб интерфейсов с целью совершенствования рабочего процесса тестировщиков, повышения удобства оценки качества ПО, простоты и скорости проведения функционального и интеграционного тестирования в нужных предметных областях, предоставления возможности тестировщикам передавать более полную отчётность о проведении тестирования, при условии обеспечения основных показателей надежности, безопасности и соответствия ожидаемым требованиям на современном этапе развития информационных технологий.

**1.2 Основные аналоги информационной системы для автоматизации тестирования**

В данный момент на коммерческих проектах часто вводится автоматизация тестирования. Бизнесу выгодно доверить информационной системе действия, которые вручную выполняются специалистами по тестированию ПО. В связи с этим остро стоит вопрос по выбору инструмента для автоматизации тестирования.

Перед рассмотрением аналогов важно определить, какие категории фреймворков существуют на рынке. Определение категорий позволит подобрать наиболее подходящий для проекта вариант. Особое внимание уделялось таким параметрам, как мультиплатформенность и поддержка языков программирования, сложность использования, особые функциональные возможности, популярность и совместимость.

На рынке существует большое количество фреймворков по автоматизации тестирования, каждый из них глобально выполняет одну и ту же задачу – автоматизировать ручные процессы при тестировании, но аналоги уникальны и имеют как свои преимущества, так и недостатки.

Каждый фреймворк нацелен на конкретную сферу тестирования. Некоторые созданы для автоматизации тестирования пользовательского интерфейса веб-приложений, некоторые оптимизированы для работы с настольными приложениями, некоторые имеют функционал для симуляции действий на мобильных устройствах, другие могут тестировать Application Programming Interface (API) или базы данных.

В результате информационного поиска, были выделены 6 основных категорий, описание каждой из них, их предназначение и примеры инструментов автоматизации, относящихся к этим категориям.

*Инструменты для веб-тестирования*

Эти инструменты предназначены для автоматизации тестирования веб-приложений, симулируют взаимодействие пользователя с веб-элементами страницы. Наиболее популярными и востребованными являются:

* *Selenium:* Наиболее популярный фреймворк для веб-автоматизации;
* *Cypress:* Современный инструмент с уникальными возможностями, работающий напрямую с версткой веб-приложения;
* *WebDriverIO*: Node.js-фреймворк для веб-тестирования;

*Инструменты для мобильного тестирования*

Эти инструменты предназначены для автоматизации тестирования мобильных приложений, взаимодействие пользователя с мобильным устройством. Например:

* *Appium:* Кросс-платформенный инструмент для мобильной автоматизации;
* *XCUITest* (для iOS) и *Espresso* (для Android);

*Инструменты для API-тестирования*

Эти инструменты помогают автоматизировать тестирование API для клиент-серверной и других архитектур. Примеры:

* *Postman:* Инструмент для создания и отправки HTTP-запросов;
* *RestAssured:* Java-библиотека для тестирования REST API;
* *RestSharp:* C#-библиотека для тестирования REST API;

*Инструменты для нагрузочного тестирования*

Эти инструменты позволяют оценить производительность системы под нагрузкой. Примеры:

* *JMeter:* Самый популярный инструмент для нагрузочного тестирования;
* *Gatling:* Scala-инструмент для тестирования производительности;

*Инструменты для тестирования на уровне базы данных*

* Эти инструменты помогают автоматизировать проверку данных в базах данных. Примеры:
* *DbUnit:* Java-библиотека для тестирования баз данных;
* *SQLUnit:* Инструмент для запуска SQL-запросов в тестах;

*Инструменты для функционального тестирования*

* Эти инструменты помогают автоматизировать функциональное тестирование. Примеры:
* *NUnit:* Фреймворк для тестирования на C#;
* *JUnit и TestNG:* Популярные фреймворки для тестирования на Java;

Для определения категории, необходимой для сравнительного анализа был проведён детальный анализ актуальности и популярности на современных проектах каждой из категорий.

В последние время набирают популярность zero-code решения, которые не требуют для своего использования знаний языков программирования, достаточно просто записать действия и иметь возможность воспроизводить их в качестве автоматизированного теста. Примером является платформа TestGrid (Рисунок 1).



Рисунок 1. Интерфейс TestGrid

Но такие решения не будут рассматриваться для сравнения, так как решения, поддерживающие языки программирования более стабильны и эффективны для решения задач по тестированию.

По результатам анализа и опроса был сделан вывод, что самой популярной и важной на данный момент категорией является категория “Инструменты для веб-тестирования”. Далее предоставлено краткое обоснование выбора этой категории.

Веб-приложения продолжают оставаться востребованными и актуальными в современном мире. Вот несколько ключевых аспектов, подчеркивающих их популярность:

Рост числа веб-приложений. Каждый день появляется все больше веб-приложений – от интернет-магазинов до социальных сетей. Это связано с ростом онлайн-бизнеса, цифровой трансформацией и повышенным спросом на онлайн-сервисы;

1. Множество устройств и браузеров: Пользователи заходят на веб-сайты с разных устройств (компьютеры, смартфоны, планшеты) и используют разные браузеры. Веб-приложения должны быть адаптированы для всех сценариев;
2. Безопасность и защита данных: Веб-приложения подвергаются угрозам безопасности, таким как взломы, утечки данных и мошенничество. Тестирование и обеспечение безопасности – важные аспекты веб-разработки;
3. Постоянные обновления и новые требования: Веб-приложения постоянно обновляются, добавляются новые функции и исправляются ошибки. Разработчики должны следить за последними тенденциями и адаптироваться к новым требованиям;
4. Пользовательский опыт: Качество веб-приложения напрямую влияет на удовлетворенность пользователей. Отзывчивость, интуитивный интерфейс и быстрая загрузка – ключевые факторы успеха;

После определения точной категории был составлен список самых часто используемых на практике инструментов автоматизации, входящих в эту категорию, которые выполняют свои задачи на современных проектах по разработке ПО и широко используются в сфере тестирования пользовательских интерфейсов и внутренней логики веб-приложений.

Список основных инструментов для автоматизации тестирования интерфейса веб-приложений, начиная от самого популярного к менее популярному:

* Selenium;
* Cypress;
* Playwright;
* Katalon Studio;
* TestCafe;
* WebDriverIO;
* Ranorex;
* Jest;
* Puppeteer;
* SikuliX;
* Applitools Eyes;
* UFT;
* Robot Framework;
* Screenster;
* CasperJS;
* Watir;
* Ghost Inspector;
* TestCraft;
* Cucumber;
* NightwatchJS;
* TestRigor;
* Rapise;
* Aquality framework;
* Testim;
* Subject7;
* Selenide;
* ACCELQ;
* iMacros;
* IBM Rational Functional Tester;
* Watir;
* Telerik Test Studio;
* Virtuoso;

Для сравнительного анализа необходимо определить тройку наиболее подходящих инструментов. Чтобы выбрать наилучшие фреймворки из списка были определены самые влиятельные критерии выбора.

Основные критерии для выбора исследуемых аналогов могут включать следующие аспекты:

* *Функциональность:* Какие функции и возможности предоставляют системы. Особое внимание уделялось на автоматизацию тестов, интеграцию с другими инструментами, отчетность и уникальные особенности инструмента.
* *Возможности:* Оценивается, насколько фреймворк соответствует потребностям проекта. Рассмотрена поддержка различных типов тестов (функциональные, нагрузочные, UI-тесты), интеграция с другими инструментами и возможности взаимодействия с элементами страницы.
* *Производительность:* Как быстро системы выполняют тесты. Есть ли ограничения по объему тестов.
* *Поддержка и сообщество:* Каков уровень поддержки со стороны разработчиков. Есть ли активное сообщество пользователей.
* *Сложность использования:* Насколько легко развернуть и настроить систему. Требуется ли специальная экспертиза.
* *Язык программирования:* Нужно убелиться, что фреймворк поддерживает язык программирования, востребованный для проекта.
* *Отчетность и логирование:*Какие отчеты генерируются после выполнения тестов. Хорошая отчетность поможет анализировать результаты.
* *Совместимость с инфраструктурой:*Как фреймворк интегрируется с существующей инфраструктурой (CI/CD-пайплайны, контейнеры и т. д.).
* *Бюджет и лицензия:*Некоторые фреймворки могут быть платными или иметь ограничения по использованию. Фреймворк должен соответствовать проектному бюджету и требованиям.

На основе выделенных критериев были определены три наиболее влиятельных фреймворка для сравнения: Selenium WebDriver, Cypress и Playwright. Разбор фреймворка Selenium WebDriver представлен в таблице 1.

Таблица 1 – Описание Selenium WebDriver

|  |  |
| --- | --- |
| **Критерий** | **Описание** |
| **Название и разработчик** | Selenium WebDriver. Разработан командой разработчиков и сообществом, без единого авторства. |
| **Описание и назначение** | Это инструмент для автоматизации тестирования веб-приложений и симуляции действий с браузером.  Он позволяет создавать и запускать тесты, взаимодействуя с веб-страницами и веб-элементами как это делает пользователь.  Особенностью является открытый исходный код фреймворка и бесплатное использование. |
| **Основные компоненты** | *Selenium WebDriver.* Это основной компонент, который позволяет взаимодействовать с браузерами. Есть возможность написать тесты на разных языках программирования (Java, Python, C#, Ruby или JavaScript) и выполнять их с помощью WebDriver.  *Selenium IDE*. Это расширение для браузера Chrome и Firefox, которое позволяет записывать и воспроизводить действия пользователя на веб-страницах.  *Selenium Grid*. Этот компонент позволяет распределить тесты на несколько машин, чтобы ускорить выполнение. |
| **Производительность** | Selenium обеспечивает эффективное выполнение тестов, но производительность может зависеть от объема тестов и настроек. |
| **Поддержка и сообщество** | Selenium имеет активное сообщество разработчиков и пользователей, что обеспечивает поддержку и решение проблем.  Также он является устоявшимся инструментом с обширной документацией и большим количеством примеров использования. |
| **Сложность использования** | Настройка Selenium может быть сложной, особенно для тестирования в разных браузерах и для запуска тестов в параллельных потоках. |
| **Язык программирования** | Поддерживает широкий спектр языков программирования: Java, Python, C#, Ruby, JavaScript. |
| **Отчетность и логирование** | Selenium предоставляет возможности для генерации отчетов после выполнения тестов.  Также имеет большое количество плагинов для генерации отчетов. |
| **Совместимость с инфраструктурой** | Selenium интегрируется с CI/CD-пайплайнами, контейнерами и другими инструментами. |
| **Бюджет и лицензия** | Selenium – это бесплатный и открытый инструмент, доступный для всех. С открытым исходным кодом |

Разбор фреймворка Cypress представлен в таблице 2.

Таблица 2. Описание Cypress

|  |  |
| --- | --- |
| **Критерий** | **Описание** |
| **Название и разработчик** | Cypress разрабатывается Cypress.io, некоммерческой организацией, которая создает инструменты для тестирования веб-приложений. |
| **Функциональность и возможности** | Cypress – это инструмент для автоматизации веб-приложений, специализирующийся на тестировании фронтенда. Он обеспечивает простоту в создании и запуске тестов, а также взаимодействие с веб-страницами.  Особенностью является простой синтаксис и интегрированный тестовый раннер. |
| **Производительность** | Cypress обладает высокой производительностью благодаря своей архитектуре и способности выполнять тесты непосредственно в браузере. |
| **Поддержка и сообщество** | У Cypress есть активное сообщество, которое предоставляет поддержку, обучающие материалы и решения проблем. |
| **Сложность использования** | Cypress обладает простым и интуитивным синтаксисом, что упрощает его использование даже для новичков. |
| **Язык программирования** | Cypress поддерживает только JavaScript, что делает его доступным для менее широкого круга разработчиков. |
| **Отчетность и логирование** | Cypress предоставляет гибкие возможности для создания отчетов и логирования результатов тестов. |
| **Совместимость с инфраструктурой** | Cypress интегрируется с CI/CD-пайплайнами, Docker, и другими инструментами. |
| **Бюджет и лицензия** | Cypress – это бесплатный и открытый инструмент, доступный для всех. |

Разбор фреймворка Playwright практически аналогичен разбору Cypress, поэтому выделены отличительные особенности в таблице 3. А также проанализированы преимущества и недостатки в таблице 4.

Таблица 3. Описание особенностей Playwright

|  |  |
| --- | --- |
| **Критерий** | **Описание** |
| **Функциональность и возможности** | Playwright – это библиотека для автоматизации веб-приложений, специализирующаяся на тестировании фронтенда. |
| **Отличительные особенности** | Она обладает кросс-платформенной, кросс-браузерной и кросс-языковой поддержкой, а также включает полезные функции, такие как автоматическое ожидание.  Она специально разработана для современного веба и обычно выполняет тесты очень быстро, даже для сложных проектов тестирования |

Таблица 4. Преимущества и недостатки Playwright

|  |  |
| --- | --- |
| **Преимущества** | **Недостатки** |
| **Кросс-браузерная поддержка**: Playwright работает с несколькими браузерами, включая Chromium (Chrome, Edge), Firefox и WebKit (Safari). Эта совместимость с разными браузерами позволяет проводить последовательное тестирование в разных средах. | **Сложность использования**: Настройка Playwright может быть сложной, требует документации и опыта работы с подобными системами |
| **Кросс-платформенная поддержка**: Вы можете использовать Playwright для тестирования приложений на разных платформах, таких как мобильные устройства (Android), веб и настольные компьютеры (MacOS, Linux, Windows). | **Отчетность и логирование:** Playwright не предоставляет гибкие возможности для создания отчетов и логирования результатов тестов.Для этого требуются дополнительные плагины. |
| **Эмуляция мобильных устройств**: Playwright может эмулировать мобильные устройства, включая геолокацию, размер экрана и другие характеристики устройств. | **Совместимость с инфраструктурой**: Playwright плохо интегрируется с CI/CD-пайплайнами, Docker и другими инструментами. |
| **Поддержка нескольких языков программирования**: Изначально созданный для Node.js, Playwright теперь предлагает привязки для JavaScript, TypeScript, Python, Java и C#/.NET, что делает его доступным для широкого круга разработчиков и тестировщиков. |  |
| **Режимы без графического интерфейса и с графическим интерфейсом**: Он может запускать браузеры в режиме без графического интерфейса (headless) для более быстрого выполнения в тестовых средах и в режиме с графическим интерфейсом для разработки и отладки. |  |

Исходя из изложенного, каждый из существующих инструментов предоставляет слишком абстрактные и сложные интерфейсы, также они были разработаны до выхода современных сред программирования, поэтому есть необходимость разработки упрощенного фреймворка под конкретную современную платформу, которым смогут пользоваться специалисты по функциональному тестированию без большого опыта в программировании.

1.3 Выводы и постановка задач на дипломное проектирование

Целью дипломного проекта является проектирование набора классов, методов и интерфейсов, разработанных с учетом потребностей специалистов по тестированию.

С его помощью специалисты должны иметь возможность автоматизировать свои тестовые сценарии, соблюдая TDD (Test-driven development) и BDD (Behavior-driven development) подходы. Также он должен позволить производить автоматический запуск тестов для регрессионного тестирования, не тратя рабочее время на повторное выполнение однотипных задач. Подключаемый код фреймворка должен обладать простыми и понятными названиями переменных, методов и классов, своевременно обновлять подключенные пакеты-зависимости и совершенствоваться сообществом тестировщиков. При этом, фреймворк должен иметь возможность решить ряд конкретных задач функционального и интеграционного тестирования, формулируемых на основе требований тестировщиков. Важным является исправление слабых сторон, выявленных при рассмотрении аналогичных инструментов автоматизации.

Объект исследования – применение автоматизации процессов с помощью информационных технологий для повышения качества продукта, ускорения процесса разработки, снижения затрат и ресурсов на тестирование, уменьшения рисков появления ошибок в ПО и быстрого обнаружения дефектов.

Предмет исследования – разработка информационной системы для содействия специалистам по тестированию в автоматизации тестовых сценариев.

Исходя из изложенного материала, при разработке дипломного проекта, необходимо решить следующие задачи:

* реализовать метод поиска элемента по Xpath локатору, который позволяет хранить найденный элемент в виде переменной для дальнейшего переиспользования;
* создать классы для элементов разного типа, содержащих уникальный набор функций для симуляции действий над конкретным типом элемента;
* разработать метод симуляции клика и подобных действий, которые совершаются с элементом;
* предусмотреть подсветку активного элемента;
* предоставить возможность получения текста элемента, его HTML атрибутов и CSS значений, состояния отображения;
* разработать алгоритм работы с явными и неявными ожиданиями, а также ожиданий с условием;
* спроектировать функционал для работы с файловой системой для возможности загрузки файлов;
* определить основные функции пользователей в ролях «Пользователя» и «Контрибьютора»;
* обеспечить возможность внесения дополнительного или изменения существующего функционала фреймворка;
* оценить целесообразность финансовых вложений в разработку фреймворка и провести анализ конкурентной среды на коммерческом рынке;
* протестировать каждый метод и класс фреймворка, включая проверку работоспособности всех языковых операторов, ключевых слов и полей, правильность отображения данных при логировании, корректность работы симуляции браузера и т.д., а также совместимость фреймворка с различными браузерами, их версиями, и кроссплатформы, включая проверку корректной работы на разных операционных системах и мобильных устройствах разных версий;
* проанализировать эргономические показатели системы и удобство использования библиотеки с кодом фреймворка для пользователей, включая оценку удобства навигации, доступности, описания функций и др.
* предусмотреть рабочие условия и разработать меры безопасности для снижения рисков возникновения травм, оценить условия работы с компьютером, включая обоснование выбора системы искусственного освещения в помещении, вентилируемости, использования особых дисплеев и других мер безопасности

2 ЭРГОНОМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Разработка алгоритмов работы пользователя

Основываясь на функциональных возможностях информационной системы фреймворк для автоматизации тестирования, были выделены все возможные роли пользователей, а именно роли “Контрибьютор” и “Пользователь”.

Описание выделенных ролей пользователей системы представлено в таблице 2.1.

Таблица 2.1 – Роли пользователей системы

|  |  |  |
| --- | --- | --- |
| № | Роль | Описание |
| 1 | Контрибьютор | Имеет доступ к исходному коду фреймворка, может его модифицировать и предоставлять новую версию пользователям. |
| 2 | Пользователь | Использует фреймворк в качестве библиотеки для автоматизации тестирования своих сценариев. Может использовать фреймворк для своих нужд, но не имеет права изменять его реализацию. |

Для каждой выделенной роли составлен перечень полномочий пользователей в рамках системы, перечень предоставлен в таблице 2.2.

Таблица 2.2 – Полномочия пользователей системы

|  |  |  |
| --- | --- | --- |
| № | Полномочие | Описание полномочий |
| 1 | Получить доступ к исходному коду фреймворка | Имеет возможность склонировать репозиторий с исходным кодом на локальную машину |
| 2 | Создание Pull Request (PR) с изменением исходного кода | Создание предложения на внесение изменений в исходный код фреймворка |
| 3 | Код-ревью PR исходного кода | Возможность оставлять комментарии на предложенные изменения в исходный код, а также отвечать на них |
| 4 | Слияние изменений в основную ветку фреймворка | Возможность применить предложенные изменения в основную версию кода фреймворка и обновить его |
| 5 | Просмотр описания и журнала обновлений фреймворка | Доступ к документации фреймворка |
| 6 | Установить фреймворк в проект или решение | Доступ к добавлению фреймворка в виде зависимости или внешней библиотеки |
| 7 | Использовать доступные классы и методы фреймворка | Доступ к классам и методам фреймворка, не помеченным как private или internal. |
| 8 | Локально расширять фреймворк | Возможность добавлять методы расширения, классы наследники для проекта с использованием фреймворка, не меняя его исходный код |

Составлена матрица ролей и полномочий, на основании таблиц 2.1 и 2.2, отображающая доступность тех или иных возможностей системы в зависимости от роли пользователя. Матрица предоставлена в таблице 2.3.

Таблица 2.3 – Полномочия пользователей системы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Полномочие | Контрибьютор | Пользователь |
| 1 | Получить доступ к исходному коду фреймворка | + | - |
| 2 | Создание Pull Request (PR) с изменением исходного кода | + | - |
| 3 | Код-ревью PR исходного кода | + | - |
| 4 | Слияние изменений в основную ветку фреймворка | + | - |
| 5 | Просмотр описания и журнала обновлений фреймворка | + | + |
| 6 | Установить фреймворк в проект или решение | + | + |
| 7 | Использовать доступные классы и методы фреймворка | + | + |
| 8 | Локально расширять фреймворк | + | + |

Более наглядное представление функций информационной системы и взаимодействие пользователей отображено на диаграмме вариантов использования информационной системы, изображенной на рисунке 2.1



Рисунок 2.4 – Диаграмма вариантов использования

Содержание функций информационной системы можно описать следующим образом:

1. Поиск элемента по Xpath локатору

Каждый элемент на странице имеет своё уникальное положение в иерархии DOM страницы. Для того, чтобы автоматизированный тест взаимодействовал с нужным элементом, используется специальный “путь” на языке Xpath.

1. Создание элементов разного типа

Для эргономичности фреймворка элементы разделены на типы, схожие на типы веб-элементов фронтенд разработчиков. Такие как Кнопка, Текстовое поле, Выпадающий список, Ячейка таблицы и т.д.

1. Клик по элементу

Основное действие над элементом – клик. Встречается в каждом тестовом сценарии, автоматизация позволяет кликать по элементам без задержки, что позволяет находить дополнительные дефекты.

1. Подсветка активного элемента

Во время выполнения автоматизированного теста действия могут происходить со слишком большой скоростью для восприятия человеком, также не всегда заметно, с каким именно элементом взаимодействует автотест. Подсветка решает эти проблемы.

1. Получение текста элемента

Получение текста, который появился после взаимодействия пользователя или уже был на странице нужно для сравнения его с тестовыми данными.

1. Получение HTML атрибутов элемента

Такие HTML аттрибуты как name, class, id, value могут пригодиться в процессе тестирования, т.к. являются динамическими и меняются после действий пользователя

1. Получение CSS значений элемента

Позволяет протестировать визуальную часть элемента, например: цвет элемента, границу, шрифт, размер и т.д.

1. Установка явных ожиданий и ожиданий с условием

Некоторые действия происходят с задержкой по разным причинам, чтобы автоматизированный тест не расценивал эту задержку как провал – необходимо устанавливать ожидания определенных условий, с гибкими настройками времени.

1. Заполнение текстовых полей

Текстовые поля тестируются тщательнее всего в веб-приложениях. Т.к. именно они обычно имеют особые условия, граничные значения и принимаемые паттерны. Заполнение текстовых полей помогает протестировать все возможные варианты.

1. Возможность загрузки файлов

Некоторые тестовые сценарии требуют загрузки файлов либо со страницы на ПК пользователя, либо наоборот на сервер. Специальный функционал поможет упростить этот процесс.

1. Получение состояния видимости и существования в DOM элемента

Даже если элемент нашелся в структуре страницы. Нужно убедиться, что он не перекрыт другими элементами и находится в зоне видимости пользователя. Проверки на видимость элемента решают эту задачу.

1. Прокрутка страницы до определенного элемента

Не всегда элемент находится в видимости пользователя. Также не всегда приложением пользуются со стандартным разрешением сторон. Появляется необходимость симулировать прокрутку колесиком мыши до появления искомого элемента.

1. Наследование базового класса страницы

Принятый в автоматизации паттерн Page Object довольно сложный для реализации с нуля. Поэтому во фреймворке уже реализован этот шаблон в базовом классе страницы, пользователю остается только наследовать его функционал и использовать в своем проекте.

1. Взаимодействие с выпадающим списком

Выпадающий список – особый тип элемента. Для работы с ним нужно реализовать такие функции как: раскрытие, закрытие, выбор элемента, получение всех элементов, получение элемента по умолчанию, получение выбранного элемента.

1. Сравнение ожидаемого и настоящего результата

Основой любого тестирование является сопоставление ожидаемого и актуального состояния. Во фреймворке пользователю предоставлены самые частоиспользуемые проверки, которые встречаются в популярных техниках тест-дизайна.

Для эффективного использования ресурсов разрабатываемого приложения необходимо оптимально распределить функции в системе между пользователем и ПК. Поскольку каждая система обладает своим собственным набором функций и особенностей управления, универсального способа распределения нет. Следовательно, требуется проведение анализа для определения оптимального распределения этих функций в рамках каждой конкретной системы. Основное внимание уделяется уменьшению действий, требуемых от пользователя при взаимодействии с системой. Этот подход обусловлен необходимостью предотвращения возможной негативной реакции пользователя на систему, вызванной излишним взаимодействием. Результат распределения функций представлен в таблице 2.5.

Таблица 2.5 – Распределение функций между человеком и компьютером в проектируемой системе

|  |  |  |
| --- | --- | --- |
| Название функции | Что делает пользователь в системе | Что выполняет в системе компьютер |
| 1. Поиск элемента по Xpath локатору | Составляет путь к необходимому элементу на языке Xpath | Находит необходимый элемент и подготавливает к взаимодействию |
| 1. Создание элементов разного типа | Выбирает логически подходящий тип элемента | Инициализирует объект класса подходящего типа |
| 1. Клик по элементу | Вызывает метод “.Click()” у созданного элемента | Симулирует клик мыши пользователя по элементу |
| 1. Подсветка активного элемента | Вызывает метод “.Highlight()” у созданного элемента | Выделяет границу активного элемента красным цветом |
| 1. Получение текста элемента | Вызывает метод “.GetText()” у созданного элемента | Получает текст внутри HTML тега элемента и возвращает пользователю |
| 1. Получение HTML атрибутов элемента | Вызывает метод “.GetAttribute(string attributeName)” у созданного элемента | Получает HTML атрибут, выбранный пользователем |
| 1. Получение CSS значений элемента | Вызывает метод “.GetCssValue(string valueName)” у созданного элемента | Получает CSS значение элемента, выбранное пользователем |
| 1. Установка явных ожиданий и ожиданий с условием | Составляет условие для ожидания и значения тайм-аута | Устанавливает ожидание по заданному условию на указанное количество секунд |
| 1. Заполнение текстовых полей | Вызывает метод “.SendText(string text)” у созданного элемента | Заполняет текстовое поле выбранным пользователем текстом |
| 1. Возможность загрузки файлов | Вызывает метод “.Upload(string filePath)” у созданного элемента с тегом input | Находит файл на ПК по заданному пути, загружает его на страницу в необходимый элемент |
| 1. Получение состояния видимости и существования в DOM элемента | Обращается к свойству “State” элемента, выбирает необходимое состояние | Проверяет необходимое состояние элемента и возвращает true или false |
| 1. Прокрутка страницы до определенного элемента | Вызывает метод “.ScrollTo()” у созданного элемента | Симулирует прокрутку мыши, пока необходимый элемент не будет в области видимости |
| 1. Наследование базового класса страницы | Создает класс тестируемой страницы, наследует от базового класса “Form” | Передает базовые методы класса “Form” пользователю |
| 1. Взаимодействие с выпадающим списком | Вызывает метод “.SelectOption(string optionName)” у созданного элемента с тегом select | Симулирует открытие выпадающего списка и выбор определенного варианта |
| 1. Сравнение ожидаемого результата | Указывает ожидаемый результат и настоящий результат | Производит сравнение и останавливает тест в случае провала проверки, выводит сообщение с результатом |

Несмотря на различия в выполняемых функциях, человек и компьютер часто находятся в тесной взаимосвязи в рамках современных систем. Человек предоставляет инструкции и контекст для выполнения задач, в то время как компьютер обеспечивает быструю и точную обработку информации. Такое сотрудничество подчеркивает важность сбалансированного распределения функций между человеком и компьютером для достижения оптимальной эффективности и результативности в работе системы.

Фреймворк должен быть легко масштабируемым и расширяемым для добавления нового и поддержки уже существующего функционала. Он не должен содержать грамматических ошибок и неправильных функциональных переходов.

2.2 Разработка эргономических требований и сценария информационного взаимодействия

Деятельность пользователя зависит от шагов теста, который он хочет автоматизировать. Тест может включать в себя различные тестируемые страницы и веб-элементы. Деятельность зависит от необходимых действий над элементом для симуляции: клик, получение текста , подсветка, прокрутка, ожидание, получение состояния.

Необходимо разработать алгоритмы работы человека в подсистеме “тестировщик-фреймворк-среда” в виде таблиц и блок-схем. Алгоритм работы пользователей представлен в таблице 2.6

Таблица 2.6 – Алгоритм работы пользователя в процессе взаимодействия с приложением в системе «тестировщик-фреймворк-среда»

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Содержание операции | Обращение к средствам отображения информации | Обращение к органам управления |
| 1 | Подключение фреймворка | Окно менеджера пакетов | Нажатие кнопки “Добавить пакет” |
| 2 | Запуск автоматизированного теста | Окно обозревателя тестов | Нажатие кнопки “Run” |
| 3 | Выбор браузера | Конфигурационный json файл | Добавление значения к полю “browserName” |
| 4 | Дополнительные настройки браузера | Конфигурационный json файл | Добавление значения к полю “browserVersion” |
| 5 | Выбор тестируемой страницы | Класс наследник базовой страницы | Ввод названия выбранной страницы |
| 6 | Добавление элемента | У драйвера метод добавления элемента | Ввод пути Xpath в параметр “locator” |
| 7 | Симуляция клика | У необхоимого элемента метод “Click()” | Вызов метода у объекта наследника базового элемента |
| 8 | Подсветка элемента | У необхоимого элемента метод “Highlight()” | Вызов метода у объекта наследника базового элемента |
| 9 | Получение текста | У необхоимого элемента метод “GetText()” | Вызов метода у объекта наследника базового элемента |
| 10 | Получение HTML атрибута | У необхоимого элемента метод “GetAttribute()” | Ввод необходимого аттрибута в параметр “attribute” |
| 11 | Получение CSS значения | У необхоимого элемента метод “GetCSSValue()” | Ввод необходимого значения в параметр “value” |
| 12 | Установка ожидания с условием | Объект класса ожидания, метод | Передача необходимого условия в параметр “condition” |
| 13 | Заполнение текстового поля | У необхоимого элемента метод “SendText()” | Передача необходимого текста в параметр “text” |
| 14 | Загрузка файла | У необхоимого элемента метод “UploadFile()” | Передача пути к файлу в параметр “filePath” |
| 15 | Получение состояния элемента | У необхоимого элемента свойство “Condition” | Выбор необходимого состояния |

Алгоритмы, приведенные выше, описывают работу пользователя-тестировщика со всеми функциями фреймворка для автоматизации тестирования на платформе .NET 8.

Эргономические требования (ЭТ) – это требования к системе в целом, ее отдельным подсистемам, оборудованию, рабочей среде, а также человеку-пользователю компьютера, определяемые свойствами человека и устанавливаемые для обеспечения его эффективной и безопасной деятельности. Задачей эргономического проектирования является проектирование деятельности пользователей. В процессе ее решения определяются различные эргономические требования, такие как: объем памяти и внимания, скорость реакции, эмоциональной устойчивости и другие, производится проверка выполнения предельно допустимых норм деятельности пользователя.

Основываясь на рекомендациях и требованиях по включению психофизиологических и эргономических характеристик человека в дизайн архитектуры модулей фреймворка, приведенных в нормативной, справочной и научной литературе, были разработаны спецификации эргономических требований, проведена их группировка.

Поскольку перечень ЭТ может быть достаточно обширным, его разделяют на 6 групп в соответствии с принятой в инженерной психологии и эргономике классификацией. При этом выделяют антропометрические, физиологические, психофизиологические, психологические, социально-психологические и гигиенические требования. Количество учитываемых групп может быть различным у систем разного типа. Поскольку в нашем случае ставится задача обеспечить эргономичность пользовательского интерфейса системы «человек-компьютер-среда», то можно ограничиться учетом только психофизиологических, психологических и социально-психологических групп ЭТ.

Групповые эргономические показатели определяют эргономические свойства системы. В общем случае принято выделять 4 таких свойства, а именно: «управляемость», «обслуживаемость», «освояемость» и «обитаемость». В нашем случае можно ограничиться только 2 свойствами: «управляемость» и «освояемость».

Дальнейший анализ представляет собой определение общих эргономических требований. От результатов выполнения этой части работы зависит эффективность взаимодействия пользователей с проектируемой системой. Эргономические требования к системе приведены в таблице 2.7

Таблица 2.7 – Эргономические требования

|  |  |  |
| --- | --- | --- |
| Эргономические свойства | Эргономические показатели | |
| Управляемость | Группа ЭТ | Номенклатура ЭТ |
| Психофизиологические | * Оптимальный контраст элементов интерфейса, чтобы обеспечить хорошую видимость; * оптимальный размер и толщина знаков на экране с учетом порога чувствительности зрения человека; * оптимальное расстояние между буквами, словами и строками с учетом порога чувствительности зрения человека; * соответствие количества одновременно предъявляемых стимулов объему восприятия человека; * соответствие объемов информации, требующей запоминания, возможностям памяти человека. |
| Психологические | * Соответствие цветов надписей и знаков стереотипам восприятия; * пространственное расположение информации на экране должно соответствовать оптимальному порядку ее восприятия; * отсутствие неоднозначного толкования требований, инструкций и команд; * соответствие количества одновременно предъявляемых сигналов возможностям внимания человека; * изменение размеров подобных элементов в диапазоне, позволяющем воспринимать их как однотипные; * один и тот же характер команд на протяжении всего периода работы в системе в схожих ситуациях; * наличие подсказок о следующих шагах работы в системе; * наличие предупреждений о нежелательных последствиях некоторых действий. |
| Социально-психологические | * Наличие ограничений доступа к определенным функциям в зависимости от роли в конкретной ситуации. |
| Освояемость | Психологические | * Соответствие сложности инструкций времени, отводимому на их усвоение; * логичная и интуитивно понятная структура фреймворка. |

Разработанный фреймворк должен полностью соответствовать техническому заданию и позволять успешно решать поставленные перед ним цели и задачи, а в процессе использования – надежно и устойчиво функционировать.

Рассмотрим подробнее эргономические требования к информации, представляемой на экране дисплея. Для точного считывания информации и комфортных условий ее восприятия работа с дисплеями должна проводиться при таких характеристиках, которые входят в оптимальные или предельно допустимые (при кратковременной работе) диапазоны.

Контраст изображения должен быть не менее 3:1. Ширина контура знака должна быть в пределах от 0,25 до 0,5 мм.

При необходимости точной идентификации цвета знака в рядах буквенно-цифровых символов высота знака не должна быть менее 20` и 30` для обособленных знаков при проектном расстоянии наблюдения.

Поля восприятия графической информации должны иметь следующие размеры:

* поле точного восприятия: 3⁰ вверх-вниз (или 2,6-2,7 см), 7⁰ вправо-влево (или 4,8-5,2 см) от оси зрения;
* поле опознания расположения: вверх 25⁰ (или 24-28 см), вниз 35⁰ (или 34 ˗ 40 см), вправо и влево по 32⁰ (или 31-37 см) от оси зрения;
* поле высокозначимой информации: 15⁰ (или 14-16 см) во все стороны от оси зрения;
* поле главного объекта: 10⁰ (или 9-10 см) во все стороны от оси зрения.

В поле главного объекта не должно находиться более 4-6 второстепенных объектов. Надписи, обозначающие объекты или органы управления, должны быть краткими, однозначно воспринимаемыми и читаться слева направо. В них допускается использование только тех слов, которые хорошо известны пользователю.

Параметры предъявляемого на экране текста должны удовлетворять следующим требованиям:

* высота знака – не менее 3 мм;
* отношение ширины буквы, цифры к высоте – в пределах 0,76-0,8;
* толщина линии обводки в прямом контрасте – в пределах 10-15% от высоты знака, в обратном контрасте – в пределах 12-16% от высоты знака;
* расстояние между буквами от 20 до 50 % ширины знака, а между словами от 1 до 1,5 ширины знака;
* расстояние между строками – 1,5-2 высоты знака;
* длина строки – 40-80 знакомест.

С учётом приведенных параметров, в результате эргономического проектирования, был разработан пользовательский интерфейс, который удовлетворяет условиям технического задания и рассмотренным выше эргономическим требованиям.

Здесь нужно будет добавить изображения различных частей интерфейса.

2.3 Эргономическая оценка проектируемой системы и выводы

Заключительным этапом эргономического проектирования системы является эргономическая оценка проекта. Эргономическая оценка представляет собой определение степени реализации эргономических требований. В данном случае она выполняется экспертным методом, при этом в качестве эксперта выступает сам разработчик. Сущность этого метода заключается в проведении интуитивно-логического анализа с количественной оценкой суждений и обработкой результатов.

Эффективность функционирования системы «тестировщик-фреймворк- среда» определяется взаимовлияющими показателями работы технического звена и работоспособности человека-оператора, а также эффективностью взаимодействия человека и машины в процессе функционирования системы.

Для полной эргономической оценки исходными материалами служат составленное техническое задание на разработку систем, проектная техническая документация, описывающая результаты эргономического проектирования, публичные конструкторские документы, прототипы и образцы системы «тестировщик – фреймворк – среда».

Последующий анализ включает в себя выявление единичных эргономических показателей, которые составляют каждую из перечисленных групп показателей.

На основе сформулированных ранее эргономических требований составим спецификацию единичных эргономических показателей (ЕЭП) (таблица 2.8)

Таблица 2.8 – Общие эргономические требования к проектируемой системе и соответствующие им единичные эргономические показатели для эргономичных свойств «Управляемость» и «Освояемость» .

|  |  |  |
| --- | --- | --- |
| Эргономическое свойство «Управляемость» | | |
| Группа ЭТ | Эргономическое требование | Единичный эргономический показатель |
| Психофизиологические | 1. Возможность настройки контрастности кода, ключевых слов и операторов для обеспечения необходимой видимост | Шаблоны контрастности разработаны с учетом оптимального контраста, что обеспечивает четкость различения между определенными частями кода |
| 1. Оптимальный размер и толщина знаков на экране с учетом порога чувствительности зрения человека | Размер и толщина шрифтов на экране регулируется пользователем с учетом личного уровня зрения |
| 1. Оптимальное расстояние между буквами, словами и строками с учетом порога чувствительности зрения человека | Расстояние между буквами, словами и строками в текстовых элементах настроено так, чтобы обеспечить комфортное чтение кода для различных категорий пользователей |
| 1. Соответствие количества одновременно предъявляемых стимулов объему восприятия человека | Пользователь лично контролирует одновременное количество стимулов – при желании может разбить большие конструкции на несколько методов или файлов, уменьшив количество стимулов |
| 1. Соответствие объемов информации, требующей запоминания, возможностям памяти человека | Пользователь в любой момент может использовать подсказки, описывающий каждый используемый компонент или использовать письменные комментарии |
| Психологические | 1. Соответствие цветов надписей и знаков стереотипам восприятия | Использование зеленого цвета для уведомления о успешном прохождении теста и красного при наличии ошибки в коде или неуспешном завершении теста. Синий цвет используется как цвет активных элементов или для выделения кода |
| 1. Пространственное расположение информации на экране должно соответствовать оптимальному порядку ее восприятия | Описание элементов в начале класса, а действий над элементов в конце позволяет быстро анализировать страницу. |
| 1. Отсутствие неоднозначного толкования требований, инструкций и команд | Однозначность формулировки требований, инструкций и команд |
| 1. Соответствие количества одновременно предъявляемых сигналов возможностям внимания человека | Количество предъявляемых сигналов контролируется пользователем, имеется возможность уменьшать |
| 1. Один и тот же характер команд на протяжении всего периода работы в системе в схожих ситуациях | Использование однообразных структур и команд в схожих сценариях |
| 1. Наличие подсказок о следующих шагах работы в системе | Наличие подсказок, предоставляющих все возможные действия пользователю |
| 1. Изменение размеров подобных элементов в диапазоне, позволяющем воспринимать их как однотипные | Размеры подобных элементов варьируются в диапазоне 0-5%, что позволяет воспринимать их как однотипные |
| 1. Наличие предупреждений о нежелательных последствиях некоторых действий | При возникновении возможности нежелательных последствий, компилятор выделяет опасный код желтым цветом |
| Социально-психологические | СП-1. Наличие ограничений доступа к определенным функциям в зависимости от роли в конкретной ситуации | Предоставление прав доступа к внутреннему коду фреймворка только для соучастников репозитория |
| Эргономическое свойство «Освояемость» | | |
| Психологические | 1. Соответствие сложности инструкций времени, отводимому на их усвоение | Пользователю представлена инструкция по пользованию фреймворком с примерами, время контролирует пользователь |
| 1. Логичная и интуитивно понятная структура приложения | Реализация структуры фреймовка и названий переменных, классов, методов в соответствии с общепринятыми конвенциями чистого кода |

Проведем оценку значений единичных эргономических показателей и занесем результат в таблицу 2.9.

Таблица 2.9 – Оценка единичных и групповых эргономических показателей проектируемой системы

|  |  |  |
| --- | --- | --- |
| Эргономическое свойство «Управляемость» | | |
| Группа ЭП | Значение единичных ЭП | Значение групповых ЭП |
| Психофизиологические | ПФ-1, ПФ-2, ПФ-3, ПФ-4,  ПФ-5 = 1 | 5 ∙ 1 / 5 = 1 |
| Психологические | П-1, П-2, П-3, П-4, П-5, П-6, П-7,  П-8 = 1 | 8 ∙ 1 / 8 = 1 |
| Социально-психологические | СП-1 = 1 |  |
| Эргономическое свойство «Освояемость» | | |
| Психологические | П-1, П-2 = 1 | 1 ∙ 1 / 1 = 1 |

Проведем оценку эргономических свойств, значимых для проектируемой системы. Эргономические свойства системы определяются как совокупность групповых эргономических показателей. Различные групповые эргономические показатели играют разную роль в реализации конкретного эргономического свойства, поэтому групповые эргономические показатели имеют свои весовые коэффициенты. Сумма весовых коэффициентов по каждому эргономическому свойству должна равняться единице. Значения весовых коэффициентов групп ЭП проектируемой системы представлены в таблице 2.10.

Таблица 2.10 – Распределение значений весовых коэффициентов групп ЭП

|  |  |
| --- | --- |
| Эргономическое свойство «Управляемость» | |
| Групповой ЭП | Значение весового коэффициента |
| Психофизиологический | 0,3 |
| Психологический | 0,6 |
| Социально-психологический | 0,1 |
| Эргономическое свойство «Освояемость» | |
| Психологический | 1 |

Для расчета эргономических свойств воспользуемся формулой:

ЭСв = ∑ 𝛼𝑖 ∙ ГЭПi,

где:

ЭСв – эргономическое свойство;

𝛼𝑖–весовые коэффициенты, присвоенные различным групповым эргономическим показателям;

ГЭПi – групповые эргономические показатели.

ЭСвуправляемость = (0,3 ∙ 1) + (0,6 ∙ 1) + (0,1 ∙ 1) = 0,9

ЭСвосвояемость = 1 ∙ 1 = 1

Различные эргономические свойства играют разную роль в формировании эргономичности интерфейса проектируемой системы, поэтому эргономическим свойствам также присваиваются весовые коэффициенты, соответствующие их вкладу в эргономичность. Значения весовых коэффициентов эргономических свойств проектируемой системы представлены в таблице 7.4.

Таблица 2.11 – Распределение значений весовых коэффициентов эргономических свойств

|  |  |
| --- | --- |
| Эргономическое свойство | Значение весового коэффициента |
| Управляемость | 0,7 |
| Освояемость | 0,2 |

Эргономичность вычисляется по формуле:

Э = ∑ 𝛽𝑖 ∙ ЭСвi,

где:

Э – эргономичность;

𝛽𝑖–весовые коэффициенты, присвоенные различным эргономическим свойствам;

ЭСв – эргономическое свойство.

Э = (0,7 ∙ 1) + (0,2 ∙ 1) = 0,9

На основании полученного значения показателя эргономичности можно сделать вывод о высокой степени реализации эргономических требований.

Возможно здесь стоит добавить Рекомендации по улучшению эргономичности проектируемой системы

3 РАЗРАБОТКА ФРЕЙМВОРКА ДЛЯ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ НА ПЛАТФОРМЕ .NET 8.

3.1 Разработка структуры программного обеспечения фреймворка

Согласно разделу 1.3 «Выводы и постановка задач на дипломное проектирование», на этапе проектирования необходимо указать взаимодействие модулей фреймворка между собой. Для отображения и детальной визуализации необходимо создать следующие диаграммы:

* Структурная схема системы в виде диаграммы классов фреймворка;
* диаграмма деятельности пользователей фреймворка;
* диаграмма вариантов использования.

Структурная схема системы в виде диаграммы классов фреймворка отображает наличие различных модулей системы и представлена на рисунке 3.1.

Рисунок 3.1 - Структурная схема системы в виде диаграммы классов фреймворка. ВОЗМОЖНО ТРЕБУЕТСЯ ПЕРЕРАБОТКА В КОНЦЕ



Модуль класса «Browser» является основной частью фреймворка и содержит в себе объект самого веб-драйвера, который открывает необходимый браузер заданной версии и является средой для симуляции действий пользователя в тестируемой системе. Также этот класс предоставляет публичные методы для взаимодействия с браузером или конфигурации активного в текущем потоке браузера. Предоставлена возможность перехода по заданной ссылке, закрытия активного браузера и завершения процесса драйвера, получение объекта для инъекции скриптов на языке JavaScript в автоматизированный тест, получения списка текущих вкладок, возможность сделать и сохранить скриншот экрана.

Класс «BrowserFactory» обеспечивает корректное создание экземпляра браузера и доступ к нему в других частях фреймворка. При этом контролируется несколько экземпляров, при использовании асинхронного запуска тестов и работе в нескольких потоках. Реализован по популярному порождающему паттерну «фабрика». Также предоставляет доступ к заданным конфигурациям браузера при необходимости внесения в них изменений по ходу теста.

Модуль «BrowserCapabilities» предоставляет параметры и настройки браузера, которые можно изменить. Основным параметрами являются: язык браузера, инкогнито режим, путь к папке загрузок, подсвечивание активного элемента.

Классы «ChromeCaps» и «FirefoxCaps» – типы данных с особыми параметрами для конкретного вида браузера.

Утилитарный класс «SettingReader» предоставляет возможность считывать тестовые или конфигурационные данные с файлов типом json. Предоставляет возможность получить конкретного значения по названию ключа, либо список объектов по названию ключа, либо коллекцию типа ключ-значения со всеми данными из файла.

Модуль «Logger» реализует логику логирования и сохранения результата автоматизированного теста. Предоставляет пользователю возможность добавления текстовых данных в стандартный файл-лог, который генерируется автоматически, при каждом запуске теста. Имеет возможность добавления информационного сообщения, сообщения отладки, предупреждения и ошибки.

Абстрактный класс «BaseElement» является вторым по важности модулем в системе. Базовый класс для каждого типа элемента, который описывает базовую логику работы с элементами, такую как клик, нахождение элемента на странице, получение текста, получение состояния видимости элемента, получения аттрибута и т.д.. Хранит в себе Xpath локатор, по которому можно найти элемент на странице, а также имя элемента, которое будет отображаться в итоговом логе.

Наследники «BaseElement» в виде «Button», «Label», «TextBox», «CheckBox» предоставляют пользователю дополнительные возможности взаимодействия с конкретными типами элементов. Например, ввод текста в элемент типа TextBox.

Статический класс «ConditionalWait» предоставляет возможность создания ожидания с условием. Содержит в себе объект веб-драйвера, необходимый для проверки выполнения условия ожидания, числовое значения количества секунд для ожидания, интервал повторной попытки совершения действия перед проверкой на ожидаемый результат. Также предоставляет пользователю методы для ожидания видимости элемента на экране, ожидания состояния возможности кликнуть на элемент, ожидания любого условия, возвращающего true или false, а также самый гибкий способ ожидания любого события, переданного пользователем через аргументы.

Абстрактный класс «BaseForm» представляет собой базовый класс для любого созданного класса страницы или формы в проекте по автоматизированному тестированию. Создан для соблюдения паттерна «Page Object», широко используемого в тестировании. Содержит в себе уникальный локатор формы, название и методы проверки на успешное открытие конкретной страницы, также дает возможность получения названия страницы пользователю.

Конфигурационный файл «SettingsFile», в котором хранятся выбранные настройки и параметры драйвера, которые считываются вышеописанным классом «SettingsReader» и применяются при создании экземпляра драйвера конкретного браузера или его изменении.

На диаграммах деятельности будет отражен функционал, доступный каждому из участников веб-сайта по ролям.

Алгоритм работы пользователя в роли «Пользователь» с системой представлен на рисунке 3.2 и и предполагает следующую последовательность действий:

* пользователь выбирает браузер и его версию;
* если требуются дополнительные настройки браузера, то заполняются необходимые параметры браузера в конфигурационном файле проекта;
* выбирается тестируемая страница или форма, на которой будут симулироваться действия тестировщика;
* на выбранную страницу добавляется уникальный элемент для проверки открытия нужной страницы в тесте;
* при необходимости, добавляются дополнительные элементы страницы, над которыми нужно произвести определенные действия, в рамках тестового сценария;
* после добавления элементов выбираются те, для котороых необходима симуляция действия в тесте;
* затем предоставляется выбор действия для симуляции над текущим элементом;
* если пользователь не добавляет больше элементов для симуляции, то взаимодействие с фреймворком заканчивается и пользователь может запускать созданный автоматизированный тест.



Рисунок 3.2 – Алгоритм работы пользователя в роли «Пользователь»

Одной из самых важных частей алгоритма является момент выбора действия для симуляции над текущим элементом. Алгоритм работы пользователя в роли «Пользователь» с выбором действия для симуляции предоставляет обширные возможности для взаимодействия с элементом и вынесен в отдельную блок-схему, представленную на рисунке 3.3, и предполагает следующую последовательность действий:

* пользователь заполняет путь XPath, необходимый для нахождения элемента на веб-странице;
* пользователю предоставлены варианты действия для симуляции, из которого он выбирает подходящий. У пользователя есть возможность выбрать один вариант из списка: «Действие с элементом», «Ввести текст в элемент», «Получить CSS значение», «Получить аттрибут», «Получить состояние элемента»;
* при выборе «Действие с элементом» имеется возможность выбрать один из вариантов: «Клик», «Подсветить элемент», «Прокрутить экран до элемента», «Получить текст», «Ожидать состояние элемента», «Навести мышь на элемент», «Удалить элемент». После выбора пользователь переходит к симуляции выбранного действия;
* при выборе «Ввести текст в элемент» указывается текст для ввода и происходит симуляция ввода текста;
* при выборе «Получить CSS значение» пользователем указывается название необходимого CSS свойства и он получает его значение;
* при выборе «Получить аттрибут» указывается название необходимого аттрибута и пользователю предоставляется его значение;
* при выборе «Получить состояние элемента» пользователю предоставляется выбор одного из трёх возможных состояний элемента: «Видимый», «Существует в верстке», «Активный». После чего получает в ответ true или false, в зависимости значения элемента;
* когда пользователь симулировал выбранное действие или получил запрошенные данные, то предлагается добавить ещё один элемент или завершить выполнение выбора действия для симуляции.



Рисунок 3.3 – Алгоритм работы пользователя в роли «Пользователь» с выбором действия для симуляции

НУЖНО ТАКЖЕ СДЕЛАТЬ АЛГОРИТМ ДЛЯ РОЛИ КОНТРИБЬЮТОР

Данные алгоритмы работы пользователя дают общее представление о том, каким образом происходит автоматизация тестирования с помощью фреймворка. Описываются основные шаги и действия пользователей-тестировщиков, которые необходимо выполнить, чтобы получить доступ к основным функциональным возможностям фреймворка и автоматизировать тестовый сценарий.

Поскольку фреймворк содержит двух актеров – для каждого из них будет составлено подробное описание вариантов использования. Каждый вариант использования определяет определенный набор действий, которые система выполняет при взаимодействии с актером.

Актеру в роли пользователя «Пользователь» доступны возможности:

1. Выбор одного из действий для симуляции над элементом. Для этого пользователю необходимо выбрать один из вариантов: «Кликнуть», «Подсветить», «Прокрутить», «Получить текст», «Ожидать состояние», «Навести мышь», «Удалить»;
2. Симулировать выбранное действие над элементом;
3. Получить состояние выбранного элемента;
4. Найти элемент на странице по заданному пути XPath;
5. Заполнить путь XPath для нахождения элементов;
6. Добавить уникальный элемент страницы;
7. Выбрать страницу для произведения тестирования.

Актеру в роли «Контрибьютор», доступен такой функционал, как:

1. Создание Pull Request с внесенными изменениями для обновления функционала фреймворка;
2. Получение доступа к исходному коду фреймворка для его изучения и внесения изменений;
3. Произвести слияние в основную ветку фреймворка для публикации новой версии библиотеки;
4. Добавить Code-review на чужой Pull Request с исправлениями.

Диаграмма вариантов использования вынесена на чертеж ГУИР 508900.002 ПД.

ДОБАВИТЬ ОПИСАНИЕ БД

3.2 Разработка алгоритма работы фреймворка

ДОДЕЛАТЬ (ВСТАВИТЬ КАК ФРЕЙМВОРК СОЗДАЕТ ДРАЙВЕР И БРАУЗЕР И КАК РАБОТАЕТ СЕЛЕНИУМ)

3.3 Выводы и оценка результатов разработки

В рамках выполнения разработки фреймворка для автоматизации тестирования были разработаны программный модуль системы, структура классов фреймворка, диаграмма деятельности пользователя и контрибьютора, диаграмма вариантов использования. Был описан алгоритм работы фреймворка с подробной схемой взаимодействия пользователя с фреймворком, фреймворка с драйвером и драйвера с браузером. При разработке фреймворка был использован язык программирования C# 12 версии, а также инструменты платформы .NET 8. Для подключения фреймворка в качестве библиотеки был использован nuget packet manager. В качестве среды разработки (IDE) была использована Visual Studio 2022. Для создания проекта с автоматизированными тестами использовалась библиотека NUnit и встроенный Test Explorer. Как основа фреймворка для взаимодействия c браузером использовался Selenium WebDriver. Для осуществления проверки ожидаемых результатов тестовых сценариев использовался класс Assert и дополнительная библиотека FluentAssertions. Для взаимодействия с файловой системой используются встроенные в .NET библиотеки пространства имен System.

Дополнительно был реализован функционал взаимодействия фреймворка с API. Для отправки HTTP запросов и получения ответов использовалась библиотека RestSharp. Для поддержки составления BDD сценариев подключено расширение SpecFlow. Для осуществления логирования действий, выполняемых во время работы автоматического теста использовалась библиотека Serilog. В качестве CI/CD системы для запусков тестов использовался сервер AzureDevOps. Для доступа к некоторым функциям браузера был использован язык JavaScript, который позволил создать необходимые пользователю скрипты взаимодействия с элементами, недоступным при использовании Selenium. Для реализации дополнительной логики работы с клавиатурой и мышью, такой как одновременное нажатие нескольких клавиш, нажатие правой кнопки мыши, взаимодействия с колесиком мыши, была использована библиотека Actions.

Разработка фреймворка была выполнена с учетом требований к качеству кода по соответствию принципам чистого кода и ООП, соблюдением принципов SOLID, а также с поддержкой многопоточности и параллельного запуска тестов.

Фреймворк создан для обеспечения удобного и эффективного способа специалистам по тестированию ПО автоматизировать свои тестовые сценарии, ускорить процесс обеспечения качества ПО, без участия человека запускать проект с тестами и получать результаты с историей прошлых запусков.

4 ТЕСТИРОВАНИЕ ФРЕЙМВОРКА ДЛЯ АВТОМАТИЗАЦИИ

5 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ФРЕЙМВОРКА ДЛЯ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ

5.1 Характеристика информационной системы

Информационная система, разработанная в дипломном проекте, представляет собой фреймворк для автоматизации тестирования веб-интерфейсов на платформе .NET 8. Помимо основной функции симуляции действий над элементами веб-страницы, фреймворк также позволяет внедрять в сценарии тестов взаимодействие с API ресурсом тестируемой системы и предоставляет отчёты о пройденных тестах и динамике положительных или отрицательных результатов в показателях качества ПО.

Информационная система – фреймворк для автоматизации тестирования разработана для обеспечения удобного и эффективного способа сократить затраты человеческих сил специалистов по тестированию ПО и проектных ресурсов на осуществление ручного тестирования при помощи автоматизации процессов.

Для создания информационной системы использовалась операционная система Windows 11, такие языки программирования, как C# и JavaScript. Разработка происходила в таких средах, как VisualStudio и GitHub, для хранения данных используются конфигурационные JSON файлы и система управления базами данных класса NoSQL с типом Key-value.

Преимуществом фреймворка является его универсальные возможности использования, ведь он совмещает в себе функционал тестирования интерфейса веб-приложений, клиент-серверной архитектуры приложения, позволяет автоматизировать как функциональные и интеграционные тесты, так и end-to-end тестовые сценарии.

Его можно использовать как разработчику приложения, так и специалисту по тестированию ПО, не имеющему большого опыта в программировании.

Целевая аудитория – все люди, вовлеченные в сферу тестирования: QA инженеры, автоматизаторы тестирования, разработчики ПО, DevOps инженеры.

Экономическим эффектом для разработчика будет чистая прибыль, полученная от комиссии на лицензионное использование фреймворка в коммерческих проектах.

В данном разделе необходимо определить все затраты, произведенные на каждом из этапов создания описанной информационной системы. Также необходимо рассчитать экономию основных видов ресурсов в связи с использованием данного фреймворка.

5.2 Расчет затрат на разработку ПО

5.2.1 Затраты на основную заработную плату команды разработчиков

Для расчета затрат на разработку фреймворка были учтены система налогообложения и различных расходов на дополнительные нужды.

Для исполнителей, занятых разработкой основная заработная плата () определяется исходя из численности и состава команды проекта, а также величины месячной заработной платы у каждого из участников команды, также учитывается общая трудоемкость разработки программного обеспечения.

В результате анализа предметной области и оценки рынка труда, в команду проектировщиков были привлечены следующие специалисты:

* разработчик, трудоемкость выполнения работ 80 часов, месячный оклад 850 рублей.

– тестировщик, трудоемкость 20 часов, месячный оклад 720 рублей.

Значение основной заработной платы команды получим по формуле:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | (5.1) |

где n – количество исполнителей, занятых разработкой фреймворка;

= коэффициент премий участников (составляет 1,2);

– часовая заработная плата i-го исполнителя, руб.;

– трудоемкость работ i-го исполнителя, ч.

Трудоёмкость проекта была оценена в 2,5 рабочие недели, или 100 часов.

Часовая оплата определяется путем деления месячного оклада на количество рабочих часов в месяце (для апреля 175 ч).

Расчет затрат на основную заработную плату команды представлен   
в таблице 5.2.

Таблица 5.2 – Расчет затрат на основную заработную разработчиков фреймворка

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Разработ-чик | Вид работы | Оклад месячный, руб. | Оклад часовой,  руб. | Трудоем-  кость работ, ч. | Зарплата по тарифу, руб. |
| Разработ-чик | Разработка фреймворка, исправление дефектов | 850 | 4,86 | 80 | 388,8 |
| Тестиров-щик | Тестирование фреймворка | 720 | 4,11 | 20 | 82,2 |
| Итого | | | | | 471 |
| Премия и иные стимулирующие выплаты (20%) | | | | | 94,2 |
| Всего затрат на основную заработную плату разработчиков, руб. | | | | | 565,2 |

5.2.2 Затраты на дополнительную заработную плату команды разработчиков

Дополнительная заработная плата исполнителей проекта определяется по формуле:

|  |  |
| --- | --- |
|  | (5.3) |

где, Зо – затраты на основную заработную плату, руб.;

Нд – норматив дополнительной заработной платы (примем норматив 10%).

Дополнительная заработная плата исполнителей составит:

Согласно формуле (5.3), затраты на дополнительную заработную плату команды составляют рублей.

5.2.3 Отчисления в фонд социальной защиты населения и обязательное страхование

Отчисления в фонд социальной защиты населения и на обязательное страхование определяются в соответствии с действующим с действующими законодательными актами Республики Беларусь по формуле:

|  |  |
| --- | --- |
|  | (5.4) |

где – норматив отчислений в фонд социальной защиты населения и обязательное страхования, где отчисления в фонд социальной защиты составляют 34%, а обязательное страхование – 0,6%.

Итого отчисления в ФСЗН и на обязательное страхование составят:

Согласно формуле (5.4), размер отчислений в фонд социальной защиты населения и на обязательное страхование составляет рублей.

5.2.4 Прочие затраты

Расходы по статье прочих затрат включают в себя затраты на приобретение специальной научно-технической информации, литературы, а также использование лицензионного платного программного обеспечения в ходе разработки, определяются по формуле:

|  |  |
| --- | --- |
|  | (5.5) |

где, Нp – норматив прочих затрат, (примем значение равным 30%).

Прочие затраты составят:

Общая сумма инвестиций на разработку считается по формуле:

|  |  |
| --- | --- |
|  | (5.6) |

Полная сумма затрат на разработку программного обеспечения находится путем суммирования всех рассчитанных статей затрат. Расчет общей суммы затрат представлен в таблице 5.7.

Таблица 5.7 – Затраты на разработку веб-сайта

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Основная заработная плата команды разработчиков ( |  |
| Дополнительная заработная плата команды разработчиков (д) |  |
| Отчисления на социальные нужды ( |  |
| Прочие затраты ( |  |
| Общая сумма затрат на разработку | 1006,39 |

Таким образом, полная себестоимость разрабатываемого фреймворка для автоматизации тестирования составит 1006,39 руб. Основные статьи расходов это – основная заработная плата программистов, а также отчисления на социальные нужды.

5.3 Расчет экономического эффекта от реализации информационной системы

Для проведения оценки экономического эффекта от реализации информационной системы необходимо определить предполагаемый объем специалистов по тестированию ПО, которые будут использовать фреймворк на коммерческих проектах и платить лицензионную комиссию.

|  |  |
| --- | --- |
|  | (5.8) |

где – прибыль, руб.;

Ц ком – цена комиссии за использование на 1 проекте,

*N* – количество пользователей, подключивших библиотеку с фреймворком.

НДС – сумма налога на добавленную стоимость, р.;

Рпр – рентабельность продаж (80%);

Нп – ставка налога на прибыль, согласно законодательству   
(стандартная составляет 18%).

За первый год ожидается привлечь 20 компаний по разработке коммерческих проектов, у каждой из которых по 3 проекта в год, следовательно за год общее число лицензий составит 60.

Комиссия за лицензионное использование фреймворка составляет 180 рублей.

Налог на добавленную стоимость при таком подходе к ценообразованию рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (5.9) |

где Нд.с. – ставка налога на добавленную стоимость согласно действующему законодательству РБ..

Налог на добавленную стоимость составит:

Прирост чистой прибыли составит:

5.4 Расчет эффективности показателей информационной системы

Для расчета экономической эффективности разработки и использования фреймворка для автоматизации необходимо провести сравнение затрат на его разработку и полученный годовой прирост чистой прибыли.

Для проведения расчетов эффективности показателей информационной системы использовалась формула расчета рентабельности инвестиций (Return on Investment, *ROI*):

|  |  |
| --- | --- |
|  | (5.10) |

где – прирост чистой прибыли, полученной от реализации фреймворка на коммерческих проектах, р;

– затраты на разработку и реализацию фреймворка, р.

Рентабельность инвестиций будет равна:

На разработку фреймворка для автоматизации тестирования было задействовано 100 человеко-часов и затраты составили рублей. За год планируется привлечь 60 лицензионных пользователей, которые оплачивают комиссию 180 руб. При таких условиях прирост чистой прибыли за год составит рублей. Сумма затрат намного меньше, чем сумма годового экономического эффекта, рентабельность инвестиций составляет очень высокий показатель 614%, следовательно, при успехе привлечения нужного количества клиентов, инвестиции будут весьма эффективными и целесообразно разрабатывать данный фреймворк.

Актуальность разработки объясняется тем, что современные проекты по разработке ПО все больше автоматизирует процессы, которые занимают время у специалистов, что позволяет экономить ресурсы бизнеса, перенаправить сэкономленное время на более важные задачи, а также уменьшить человеческий фактор в процессе обеспечения качества ПО.

При создании эффективного фреймворка уникальность продукта достигнута за счет определенных факторов. В первую очередь, это возможность использовать новейшую версию языка программирования C# и современную платформу разработки .NET 8, чего не могут предложить альтернативные системы. Это позволяет использовать современные подходы программирования и улучшить скорость прохождения автоматических тестов за счёт усовершенствованной среды разработки. Во вторую очередь, фреймворк покрывает сразу несколько областей в процессе тестирования. В-третьих, это низкая стоимость лицензии по сравнению с конкурентами, что является определяющим фактором при выбора инструмента для бизнеса по разработке ПО.

Таким образом, фреймворк для автоматизации тестирования на платформе .NET 8 среди специалистов по тестированию ПО является актуальным продуктом, который может быть внедрен в проект, благодаря совершенствованию технических возможностей для повышения удобства, простоты и скорости выполнения тестовых сценариев и анализа соответствия требованиям разрабатываемого программного средства.