**СОДЕРЖАНИЕ**

Введение 5

1 Назначение системы «человек-компьютер-среда» (СЧКС), ее задачи и основные функции 6

2 Анализ содержания основных функций веб-приложения для поиска и публикации вакансий 8

3 Распределение функций в системе между человеком и компьютером и определение структуры СЧКС 13

4 Разработка алгоритмов работы пользователей веб-приложения 18

5 Обоснование и разработка эргономических требований к веб-приложению для поиска и публикации вакансий 25

6 Разработка сценария информационного взаимодействия человека-пользователя и приложения 29

7 Оценка эргономичности веб-приложения 44

Заключение 49

Литература 50

Приложение А 51

Приложение Б 52

Приложение В 53

Введение

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

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

Основная задача автоматизации тестирования, как и любой другой автоматизации процессов, это экономия времени (человеко-часов), избавление от человеческого фактора в процессе поиска дефектов, возможность переиспользования без затрат сил человека, интеграция с системами постоянной интеграции и доставки кода и наглядная отчетность.

Для автоматизации можно использовать практически любой язык программирования, для данной курсовой работы выбран язык программирования C#, среда разработки Visual Studio и платформа .NET 8. Каждый из этих 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.

Основная цель данной курсовой работы — это предоставление пользователю удобного инструмента для автоматизации тестирования, основанного на Selenium WebDriver и на самой современной платформе .NET 8, предоставление эргономичного интерфейса для написания собственных автоматизированных тестов пользовательского интерфейса и API.

# Назначение системы «человек-компьютер-среда» (СЧКС), ее задачи и основные функции

Назначение данного фреймворка – представить набор классов, методов и интерфейсов, разработанных с учетом потребностей специалистов по тестированию.

С его помощью инженеры должны иметь возможность автоматизировать свои тестовые сценарии, соблюдая TDD и BDD подходы. Также производить автоматический запуск тестов для регрессионного тестирования, не тратя рабочее время на повторное выполнение однотипных задач.

Таким образом, можно выделить следующие задачи системы:

* Поиск и хранение веб-элементов в интерфейсе веб-приложений и сервисов
* Симуляции взаимодействия с элементами интерфейса
* Обеспечение доступа к состоянию элементов и метаданным страницы
* Автоматизированный запуск и анализ тестов

Проанализировав вышеназванные задачи, можно сделать вывод о том, какие функции должны быть разработаны для их выполнения:

1. Поиск элемента по Xpath локатору

2. Создание элементов разного типа

3. Клик по элементу

4. Подсветка активного элемента

5. Получение текста элемента

6. Получение HTML атрибутов элемента

7. Получение CSS значений элемента

8. Установка явных ожиданий и ожиданий с условием

9. Заполнение текстовых полей

10. Возможность загрузки файлов

11. Получение состояния видимости и существования в DOM элемента

12. Прокрутка страницы до определенного элемента

13. Наследование базового класса страницы

14. Взаимодействие с выпадающим списком

15. Сравнение ожидаемого и настоящего результата

# Анализ содержания основных функций веб-приложения для поиска и публикации вакансий

Каждая из функций проектируемой системы, включенная в разработанную спецификацию, может выполняться различными способами, поэтому необходимо определить их содержание, т.е. выбрать конкретный способ реализации с учетом назначения проектируемой системы.

Содержание функций системы можно описать следующий 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.

2. 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.

3. 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.

4. 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.

5. Получение текста элемента

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

6. 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

7. Получение CSS значений элемента

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

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

# Распределение функций в системе между человеком и компьютером и определение структуры СЧКС

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 3.1.

Таблица 3.1 – 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.

В любой момент времени система имеет компьютер, среду и человека-оператора с той или иной ролью. Разберем каждую из частей:

1. Человек. Оператор системы, взаимодействующий с технической частью системы через представленный функционал и тем самым выполняющий задачи, возложенные на систему. На работу пользователя влияют его мотивы, цели деятельности, психические и физиологические особенности и др. Пользователь может быть авторизован как соискатель, работодатель или администратор.
2. Компьютер. Программное и аппаратное обеспечение системы, в том числе облачное хранилище данных. Представляет из себя ПК пользователя с операционной системой. Состояние технической составляющей определяется уровнем развития программных и аппаратных технологий на момент пользования системой, версией установленной операционной системы и др.
3. Среда. Среда рабочего места представляет собой совокупность физических, социально-психологических, эстетических факторов внешней среды, которые воздействуют на оператора (освещенность, шум и т.п.).

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



Рисунок 3.1 – Структурная схема системы

# Разработка алгоритмов работы пользователей веб-приложения

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 4.1

Таблица 4.1 – 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 |

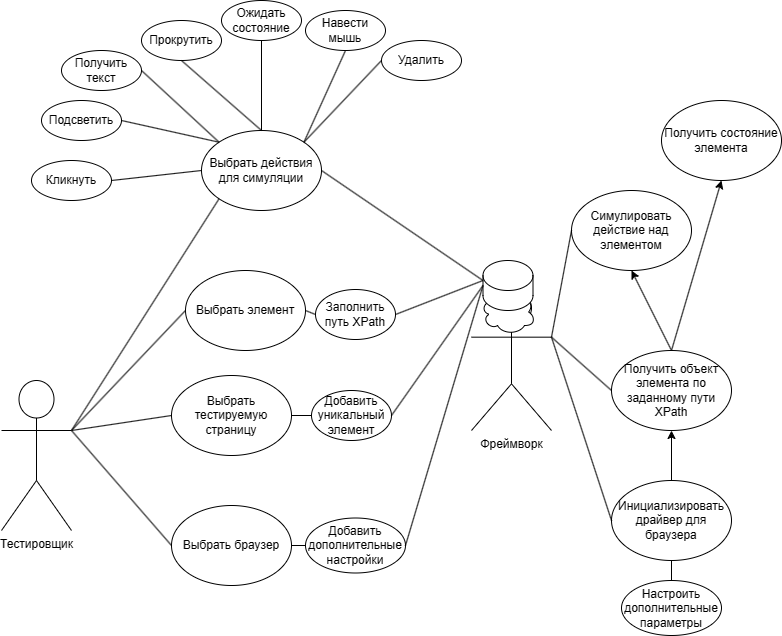


Рисунок 4.1. – Диаграмма вариантов использования в системе «тестировщик – фреймворк – среда»



Рисунок 4.2 Алгоритм работы пользователей, представленный в виде UML-диаграммы



Рисунок 4.3 Дополнение к алгоритму пользователей

Данный алгоритм работы пользователя дает общее представление о том, каким образом происходит автоматизация тестирования с помощью фреймворка. Описываются основные шаги и действия пользователей-тестировщиков, которые необходимо проделать, чтобы получить доступ к основным функциональным возможностям фреймворка и автоматизировать тест-кейс.

# Обоснование и разработка эргономических требований к веб-приложению для поиска и публикации вакансий

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.

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.

Таблица 5.1 – 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 приложения. |

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.

# Разработка сценария информационного взаимодействия человека-пользователя и приложения

Данный раздел курсовой работы представляет собой глубокий анализ и

описание конкретных сценариев, в которых взаимодействуют человек-

пользователь и ПК, а также предоставляет подробные описания процессов,

происходящих в каждом сценарии.

Первый шаг для пользователя – добавления фреймворка в проект в

качестве дополнительной зависимости или библиотеки. Самым удобным

способом является использование компонента nuget package manager.

Пример подключения фреймворка представлен на рисунке 6.1.

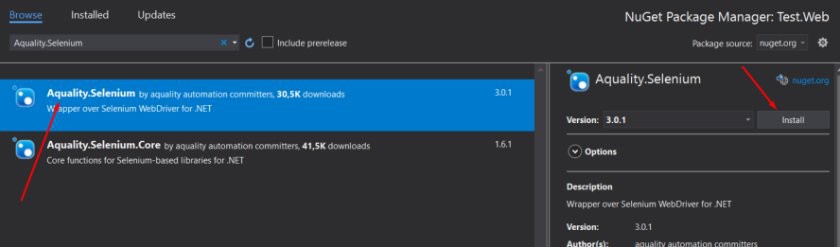


Рисунок 6.1. Подключение фреймворка

После подключения пользователь может ознакомиться со структурной схемой доступных ему классов, методов, свойств, параметров и интерфейсов. Структурная схема фреймворка представлена на рисунке 6.2



Рисунок 6.2. Базовая структура фреймворка

После того, как пользователь ознакомился со структурой фреймворка и нашёл подходящий для себя компонент – он может посмотреть подробную информацию с о любом классе или методе. Для этого нажимается Ctrl + Левая кнопка мыши для отображения описания метода и списка всех параметров метода. Или пользователь может щелкнутль правой кнопкой мыши и перейти к определению. Пример перехода к определению метода представлен на рисунке 6.3. А также пример описания метода и его параметров на рисунке 6.4

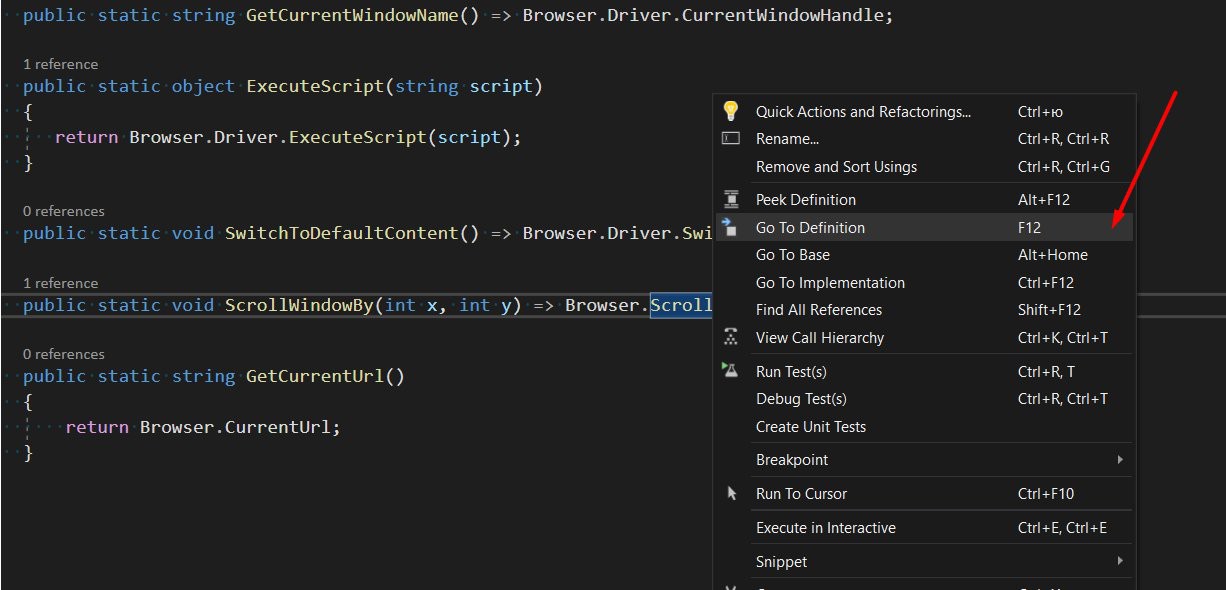


Рисунок 6.3. Переход к определению метода

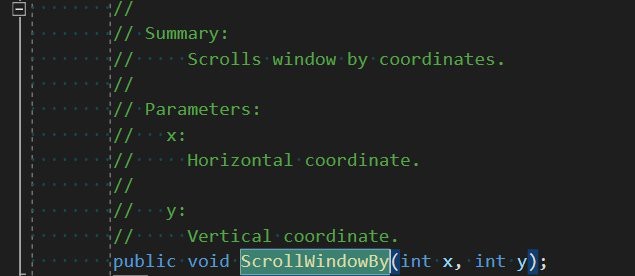


Рисунок 6.4. Описание метода и его параметров

После изучения необходимых классов и методов пользователь должен подготовить настройки запуска своих автоматизированных тестов с помощью специального файла. Для настройки используется файл settings.json, который находится в папке Resources. Пример расположения файла настройки предоставлен на рисунке

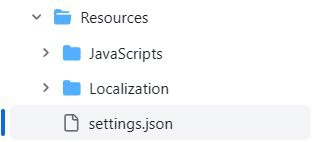


Рисунок 6.5. Расположение файла настройки

После этого пользователь заходит в параметры этого файла и изменяет значение на “Копировать всегда” для его успешной работы. Пример изменения параметра представлен на рисунке 6.6.

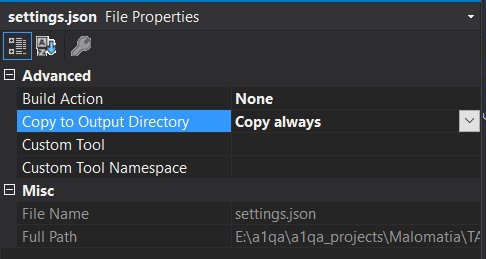


Рисунок 6.6. Изменение параметра файла

Далее пользователь составляет собственный автоматизированный тест. Для этого в первую очередь создаётся класс необходимой страницы или формы, который наследуется от базового класса для всех страниц и форм – от класса Form. Пример создания класса формы на примере FooterForm представлен на рисунке 6.7



Рисунок 6.7. Создание формы для тестирования

После чего пользователь заполняет базовый конструктор данного класса, используя локатор элемента и название страницы. Пример заполнения представлен на рисунке 6.8.

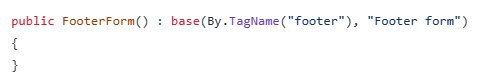


Рисунок 6.8. Заполнение базового конструктор

Когда страница создана, пользователь наполняет ее необходимыми элементами. Элементы располагаются в самой верхней части класса. Для этого используется класс фреймворка FormElement и метод FindChildElement, также указывается тип элемента и путь к нему. Пример создания элемента ContactsLabel представлен на рисунке 6.9.

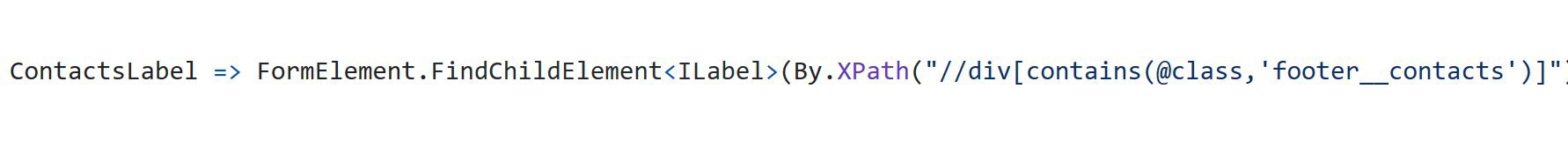


Рисунок 6.9. Элемент, готовый для взаимодействия

Затем пользователь может использовать действия над элементом, предоставленные фреймворком. Посмотреть список доступных действий пользователь может через символ точки “.” после названия своего элемента. Пример просмотра доступных действий представлен на рисунке 10.

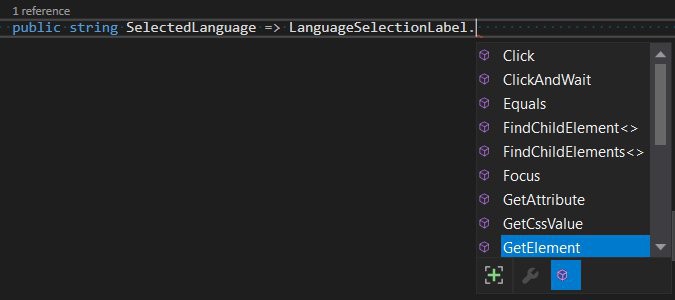


Рисунок 6.10. Просмотр действий над элементом

Таким же образом можно увидеть список доступных свойств элемента. Пример на рисунке 6.11.

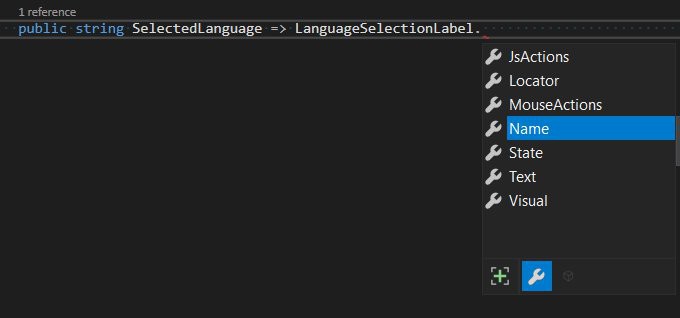


Рисунок 6.11. Просмотр доступных свойств элемента

После всей подготовки пользователь создает сценарий автотеста с использованием созданных элементов. Сценарий создаётся на языке Gherkin. Пример сценария для тестирования калькулятора представлен на рисунке 12.

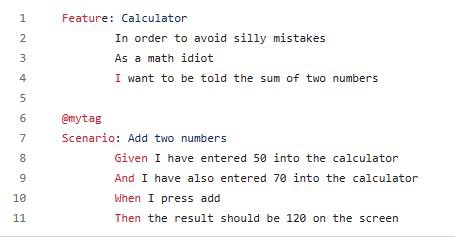


Рисунок 6.12. Сценарий автоматизированного теста

# Оценка эргономичности веб-приложения

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.

На основе сформулированных ранее эргономических требований составим спецификацию единичных эргономических показателей (ЕЭП) (таблица 7.1).

Таблица 7.1 – Эргономические требования к проектируемой системе и соответствующие им единичные эргономические показатели

|  |  |  |
| --- | --- | --- |
| 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 7.2.

Таблица 7.2 – 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 7.3.

Таблица 7.3 – 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 7.4 – 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.

Заключение

В ходе выполнения курсового проекта был разработан фреймворк для автоматизации тестирования на платформе .NET 8, предназначенный для тестировщиков. Информационная система создана в формате фреймворка, тем самым предоставляя доступ к необходимому функционалу в любое время при наличии интернета и установленного пакета с фреймворком.

В ходе проектирования фреймворка были рассмотрены некоторые существующие аналоги, в ходе чего были выявлены их достоинства и недостатки, которые были учтены в разработке. На основе выполненного анализа были сформулированы требования к программному продукту.

В конце разработки были протестированы основные функции и возможности программного продукта.

Все цели были достигнуты, разработанный фреймворк выполняет поставленную задачу.

Литература

[1] Что такое автоматизированное тестирование? Гайд по основам [Электронный ресурс] - https://testengineer.ru/chto-takoe-avtomatizirovannoe-testirovanie

[2] Naming guildelines Microsoft [Электронный ресурс] - <https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-1.1/xzf533w0%28v=vs.71%29?redirectedfrom=MSDN>

[3] Best Design Pattern for Data Access in C# [Электронный ресурс] - <https://iqunlock.com/best-design-pattern-for-data-access-in-csharp/>

[4] Для чего нужен фреймворк и как его выбрать [Электронный ресурс] - <https://practicum.yandex.ru/blog/chto-takoe-framework/>

[5]Документация Selenium [Электронный ресурс] - <https://www.selenium.dev/documentation/>

[6] Описание .NET 8 [Электронный ресурс] - https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8

Приложение А

**(обязательное)**

**Структурная схема системы**



Рисунок А.1 – Структура информационной системы

Приложение Б

**(обязательное)**

**Блок-схема алгоритма работы пользователя веб-приложения**

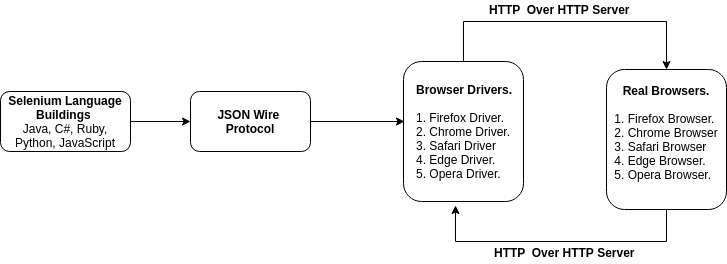


Рисунок Б.1 – Схема архитиктуры работы фреймворка

Приложение В

**(обязательное)**

**Диаграмма вариантов использования системы**



Рисунок В.1 – Алгоритм работы пользователя и администратора



Рисунок В.2 – Продолжение алгоритма работы пользователя и администратора