МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка приложения чтения/записи XML и JSON-файлов

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 21-1  Пугачёв Игорь Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2024

Аннотация

Курсовая работа посвящена теме «Разработка приложения чтения/записи XML и JSON-файлов».

Данный отчёт состоит из введения, трёх основных глав, заключения, списка использованных в процессе выполнения работы источников, а также приложения, включающего в себя: листинг кода приложения, графические элементы приложения и руководство пользователя.

Введение включает в себя актуальность выбранной темы и описание рассматриваемой проблемы.

Первая глава содержит описание предметной области, рассмотрение ключевых терминов и понятий, анализ аналогичных продуктов, техническое задание на разработку программного продукта, а также описание структуры приложения.

Вторая глава представляет из себя описание разработки программы, где рассказывается о средствах и технологиях, которые были использованы при проектировании и создании приложения, а также описывается интерфейс приложения, логика работы и схема взаимодействия компонентов приложения.

Третья глава посвящена способу установки и началу работы с приложением, а также включает в себя проводимые тесты и результаты работы приложения.

В заключении изложены основные выводы и итоги, полученные в результате проведённого исследования.

Ключевые тезисы: C#, Xaml, XML, JSON, Windows, настольное приложение.

Ссылка на GitHub репозиторий, где размещено программное обеспечение и все материалы по курсовому проекту: <https://github.com/IPrizOn/Xml-Json-RW-Utility>.

Содержание

[Введение 4](#_Toc165534731)

[1. Описание предметной области 5](#_Toc165534732)

[1.1. Аналоги разрабатываемого приложения 5](#_Toc165534733)

[1.2. Техническое задание 9](#_Toc165534734)

[1.3. Описание структуры приложения 11](#_Toc165534735)

[2. Описание разработки приложения 12](#_Toc165534736)

[2.1. Обоснования средств разработки 12](#_Toc165534737)

[2.2. Разработка интерфейса 14](#_Toc165534738)

[2.3. Разработка логики работы приложения. Схема взаимодействия компонентов проекта 14](#_Toc165534739)

[2.4. Описание переменных, компонентов, классов и подпрограмм 24](#_Toc165534740)

[3. Тестирование и установка приложения 32](#_Toc165534741)

[Заключение 40](#_Toc165534742)

[Список использованной литературы 41](#_Toc165534743)

[Приложения А. Листинг кода приложения 42](#_Toc165534744)

[Приложение Б. Графические элементы приложения 48](#_Toc165534745)

[Приложение В. Руководство пользователя 53](#_Toc165534746)

# Введение

В наше время работа с файлами форматов XML и JSON является одной из ключевых задач в разработке программного обеспечения. Файлы в этих форматах широко используются для обмена данными между приложениями, хранения информации и конфигураций, а также для обеспечения функциональной совместимости различных систем.

В рамках работы будут рассмотрены основные характеристики XML и JSON, их отличия и области применения, а также подходы к работе с данными форматами.

В ходе выполнения курсовой работы будет разработано приложение, позволяющее проводить чтение и запись файлов в форматах XML и JSON. Рассмотрены будут основные этапы работы с данными форматами, включая анализ структуры файлов, обработку данных и сохранение результатов. Представленное приложение будет способствовать более глубокому пониманию работы с XML и JSON, а также позволит приобрести практические навыки в области разработки приложений для работы с ними.

# Описание предметной области

Форматы XML (Extensible Markup Language) и JSON (JavaScript Object Notation) являются широко используемыми структурированными форматами данных. XML представляет собой универсальный формат, для хранения и обмена данных между различными системами. JSON, в свою очередь, является легковесным форматом обмена и хранения информации, который использует для передачи данных пары "ключ-значений". [4]

Проблемой для пользователя может послужить отсутствие удобного и практичного инструмента для работы с приведёнными выше форматами.

Цель данного проекта: написание настольного приложения под Windows для просмотра и редактирования информации, содержащейся в XML и JSON файлах, при разработке которого нужно учесть особенности структуры и синтаксиса данных форматов, а также освоить навыки работы с необходимыми библиотеками и инструментами.

Пользователю должно быть достаточно открыть файл в приложении и уже иметь возможность просмотреть содержащуюся информацию или отредактировать её с последующим сохранением.

* 1. Аналоги разрабатываемого приложения

На рынке существует уже достаточное большое количество аналогов для работы с XML и JSON форматами, но можно выделить основные:

* Notepad++

Популярный и мощный текстовый редактор для Windows, который поддерживает множество языков программирования и форматов файлов, включая XML и JSON (Рис. 1).



Рисунок 1 – Логотип редактора Notepad++

Программа обладает возможностью подсветки синтаксиса, авто дополнения кода, функцией поиска и замены, а также поддерживает множество плагинов, позволяющих расширить функциональность редактора (Рис. 2).

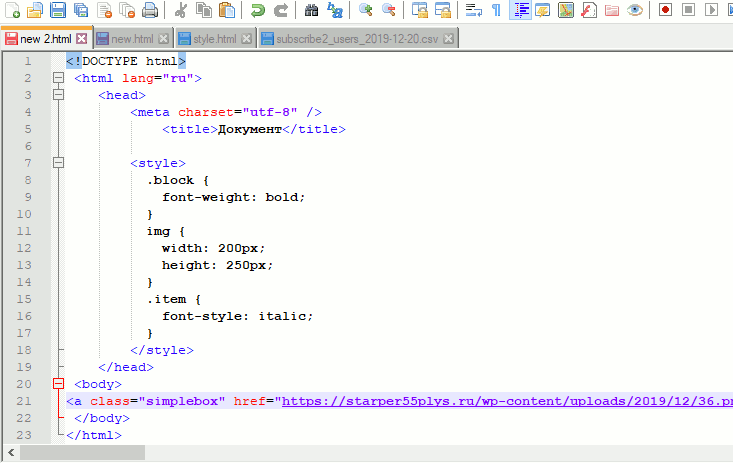


Рисунок 2 – Интерфейс редактора Notepad++

Из минусов можно отметить ограниченную поддержку языков программирования и функционал, а также менее активное сообщество.

От данного продукта можно взять меньшее потребление ресурсов за счёт малого функционала и простоту в использовании.

* Sublime Text
* Еще один популярный текстовый редактор, который широко используется разработчиками для работы с различными языками программирования, включая XML и JSON (Рис. 3).



Рисунок 3 – Логотип редактора Sublime Text

Программа отличается высокой производительностью, поддерживает подсветку синтаксиса, авто дополнение, множество горячих клавиш и возможность установки плагинов для расширения функциональности (Рис. 4).

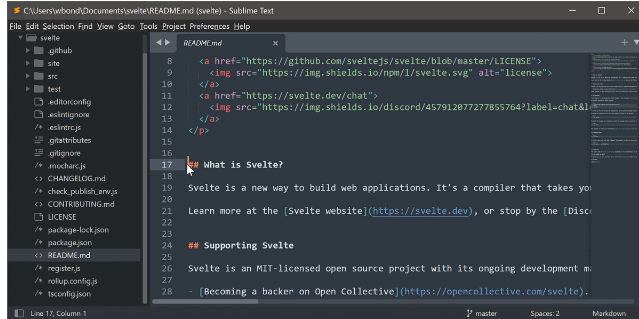


Рисунок 4 – Интерфейс редактора Sublime Text

Минусы у данного редактора: платная лицензия (есть бесплатная пробная версия) и отсутствие полной поддержки открытого исходного кода.

У продукта можно отметить скорость работы, простоту использования и широкие возможности кастомизации.

* Visual Studio Code

Современный и бесплатный редактор кода, который предоставляет широкие возможности для работы с различными языками программирования, включая поддержку форматов XML и JSON (Рис. 5).

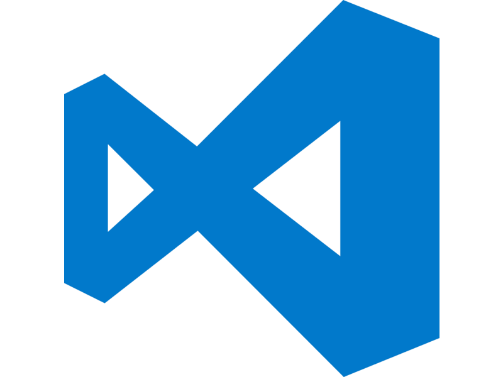


Рисунок 5 – Логотип редактора Visual Studio Code

Редактор обладает интегрированным редактором кода, подсветкой синтаксиса, авто дополнением кода, отладкой, интеграцией с системами контроля версий и множеством плагинов (Рис. 6).

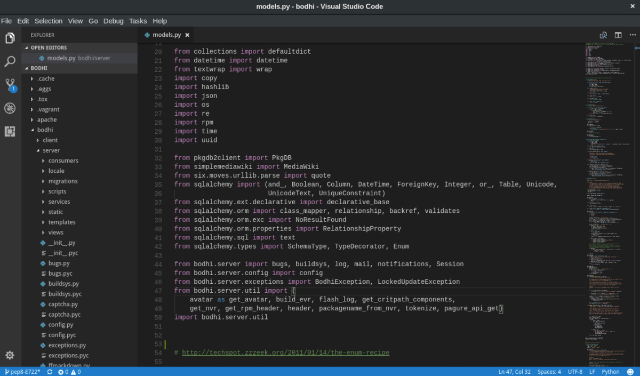


Рисунок 6 – Интерфейс редактора Visual Studio Code

Минусами данного редактора могут стать его большое потребление ресурсов, слишком обширный функционал и сложность настройки.

От продукта можно было бы взять обширный функционал и современный понятный интерфейс. [5]

* 1. Техническое задание

1. Общая информация

* Наименование программы: «Xml-Json RW Utility».
* Назначение и область применения: настольное приложение под Windows, которое позволяет считывать информацию или записывать её в XML и JSON файлы через графический интерфейс.

1. Требования к программе

Требования к функционалу:

* Приложение должно быть настольным и иметь графический интерфейс.
* Приложение должно запускаться через exe файл.
* В приложении должна быть возможность выбора типа файла перед тем, как с ним работать.
* В приложении должна быть возможность выбора файла пользователем с соответствующим форматом.
* Приложение должно отображать информацию из выбранного файла.
* В приложении должна быть возможность редактирования тегов и элементов тегов файла.
* В приложении должна быть возможность удаления и добавления тегов.
* В приложении должна быть возможность удаления и добавления элементов в теги.
* В приложении должна быть возможность сброса файла до момента изменений.
* В приложении должна быть возможность конвертации одного формата файла в другой.

Требования к надёжности:

* Приложение не должно вызывать ошибки в работе устройства после работы с файловой системой.
* Приложение не должно «ломать» файлы после работы с ними.

Требования к производительности:

* Приложение не должно сильно нагружать систему.
* Приложение должно быстро отображать информацию.

Требования к тестированию:

* Модульное тестирование. Должно быть проведено тестирование отдельных модулей программы, включая логику работы с файловой системой и обработка исключений.
* UI/UX тестирование. Должно быть проведено тестирование удобства пользователя взаимодействия с программой, включая масштабируемость окон, кнопок и правильное отображение контента в них.
* Системное тестирование. Должно быть проведено тестирование целой системы в составе всех модулей и компонентов программы.

Требования к программной документации:

* Техническое задание;
* Документация по программному продукту;
* Руководство пользователя.

1. Содержание и план работ

Содержание работ включает в себя:

* Постановка целей и задач;
* Анализ аналогичных продуктов и определение требований;
* Изучение необходимых технологий;
* Проектирование и разработка продукта.
* Тестирование продукта;
* Представление продукта.

В течение 1–2 месяцев с момента утверждения технического задания должен быть выполнен следующий план работ:

* Интерфейс программы: 7–14 дней.
* Функционал программы: 14–21 дней.
* Тестирование и разработка документации программы: 7 дней.
* Представление программы, правка недочётов: 1–2 дня.
  1. Описание структуры приложения

Продукт будет представлять из себя настольное приложение, запускаемое через exe файл на устройстве пользователя. В себе приложение будет совмещать логику обработки команд, файлов и вводимой информации от пользователя. [7]

Пользователь взаимодействует с приложением через графический интерфейс (представление), нажимая на кнопки, выбирая элементы и вводя информацию. Приложение в свою очередь обрабатывает команды от пользователя, сохраняет изменения и представляет пользователю обновлённый интерфейс (Рис. 7).

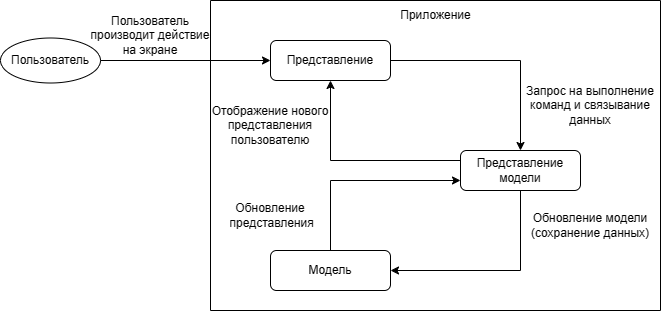


Рисунок 7 – Схема работы приложения

1. Описание разработки приложения
   1. Обоснования средств разработки

Для выбора языка программирования и сопутствующих ему средств разработки был проведён анализ трёх наиболее популярных языков программирования для создания настольных приложений: C#, C++ и Python. [3]

C# – язык, который используется для создания производительных и надёжных настольных приложений.

C++ – язык, который используется для создания высокопроизводительных настольных приложений.

Python – язык, который используется для быстрого создания простых настольных приложений.

Исходя из проанализированных данных, в качестве языка программирования был выбран C# благодаря своей популярности и эффективности в разработке настольных приложений. Также выбор C# обусловлен близостью и опытом работы с данным языком программирования (Рис. 8). [8]



Рисунок 8 – Значок языка программирования C#

В качестве среды для разработки была выбрана «Visual Studio 2022», являющаяся удобным и универсальным инструментом для любого разработчика на C# (Рис. 9). [1]

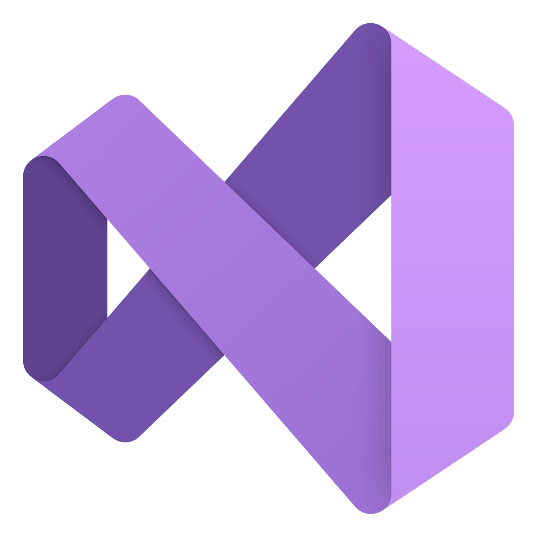


Рисунок 9 – Значок среды разработки Visual Studio

Также были рассмотрены такие среды разработки как:

* Project Rider: новый и удобный, но имеются баги, часть функционала в разработке, а также является платным;
* Eclipse: высокая функциональность, но может показаться слишком сложным для новичка, а также нет гарантий надёжности.

В качестве операционной системы была выбрана Windows за её популярность, а в качестве платформы (фреймворка) для создания графического интерфейса была выбрана WPF с языком разметки XAML, так как является хоть и не самой новой, но зато с большим сообществом и достаточной документацией для C# (Рис. 10). [9]



Рисунок 10 – Значок платформы WPF

Другими вариантами были:

* .NET MAUI: кроссплатформенный фреймворк, но слишком молодой, а также малое сообщество и документация;
* Windows Forms: устаревшая технология, на смену которой пришла WPF.
  1. Разработка интерфейса

У продукта будет свой уникальный интерфейс, а также при смене типа файла для работы будет меняться основная цветовая гамма приложения. Прилагаемые окна и страницы для приложения:

* При запуске приложения открывается главное окно, в котором размещается главная страница.
* Для чтения/записи файла предусмотрена специальная страница со всеми необходимыми кнопками, списками и формами.
* Для редактирования элементов предусмотрено специальное диалоговое окно с формой для редактирования элемента.
* Для подтверждения действий предусмотрено диалоговое окно подтверждения.
* Для конвертации файлов предусмотрено специальное диалоговое окно.
* Для обратной связи с пользователем будут предусмотрены диалоговые окна с сообщением об успехе/предупреждении/ошибке.

Все графические элементы, окна и страницы, предоставлены в Приложении Б.

* 1. Разработка логики работы приложения. Схема взаимодействия компонентов проекта
* Описание функционала (логики) приложения.

Для начала работы с приложением пользователю необходимо установить файлы приложение на своё устройство и запустить исполняемый файл. После чего откроется главное страница приложения, с которого уже можно начать работу (Рис. 11).



Рисунок 11 – Главная страница

Пользователь может выбрать, что ему в данный момент может быть нужно: переключить тип файла, с которым будет работать, сконвертировать один формат файла в другой или перейти на страницу чтения/записи файла (Рис. 12).

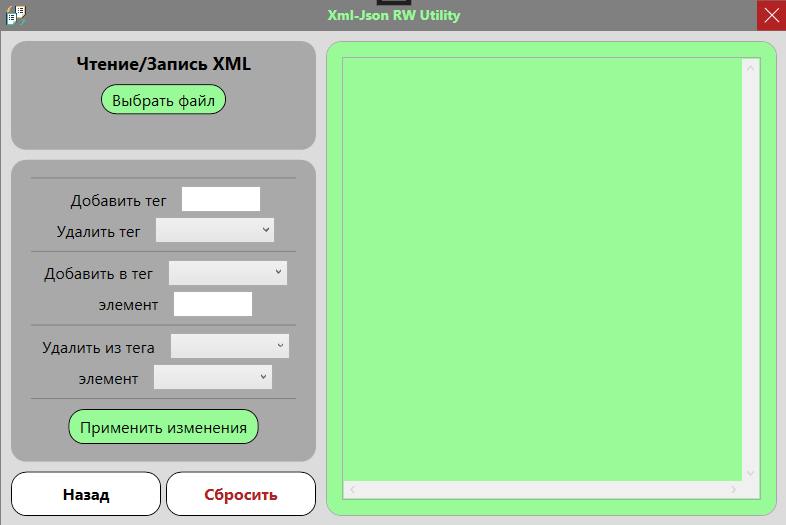


Рисунок 12 – Страница чтения/записи

На странице чтения/записи пользователь может выбрать файл с своего устройства, чтобы открыть его в программе. После чего у пользователя появляется возможность просмотреть информацию в файле и редактировать его содержимое.

Алгоритм загрузки файла в приложение включает в себя очистку всех полей и списков, создание словаря (ключ – строка, значение – список), после чего происходит преобразование строк из файла в объект документа соответствующего формата по средством его парсинга. Далее отпарсенный объект файла перебирается обычными циклами для заполнения словаря, тем самым отделить теги и элементы тегов, чтобы в последствии выстроить правильную структуру в прокручивающемся списке (Рис. 13).

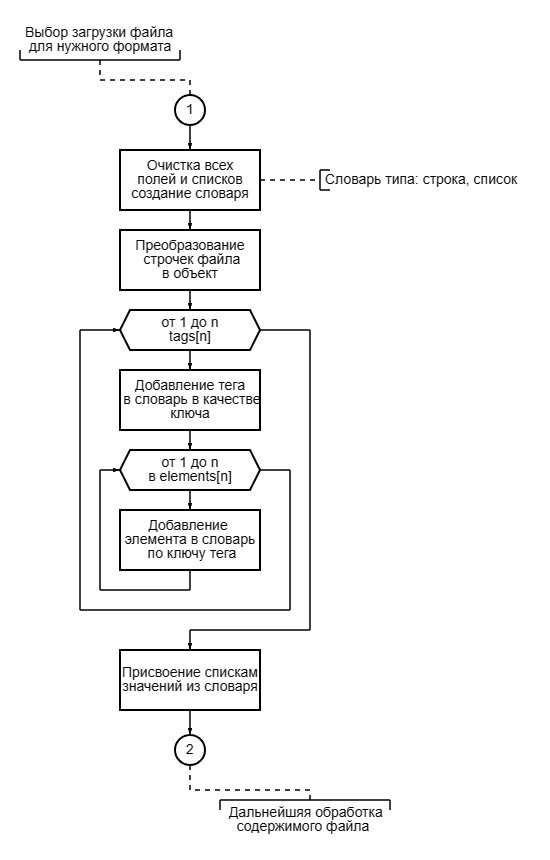


Рисунок 13 – Алгоритм загрузки файла в приложение

Правая часть страницы содержит прокручивающийся список, в котором отображаются теги и внутренние элементы выбранного файла. Левая часть страницы содержит элементы для редактирования файла (Рис. 14).

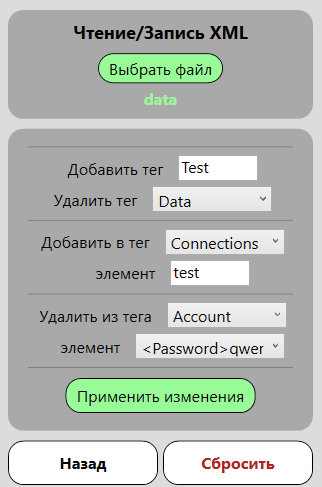


Рисунок 14 – Элементы редактирования

Назначение элементов редактирования:

«Добавить тег» – позволяет добавить тег с названием из поля ввода в файл.

«Удалить тег» – позволяет удалить выбранный тег из файла, тег выбирается посредство открывающего списка.

«Добавить в тег элемент» – позволяет добавить в выбранный тег элемент с названием из поля ввода.

«Удалить из тега элемент» – позволяет удалить выбранный элемент из выбранного тега, реализация посредством категорий и подкатегорий.

«Применить изменение» – после нажатия на кнопку берёт информацию из полей ввода и/или выпадающих списков, после чего обрабатывает её для соответствующего изменения информации в файле.

«Сбросить» – после нажатия на кнопку сбрасывает все изменения файла к моменту, когда он был выбран в системе.

«Назад» – после нажатия на кнопку возвращает на главную страницу приложения.

После любых действий выше, контент в прокручивающемся списке обновляется на новый со вступившими в силу изменениями.

Алгоритм применения изменений в файле включает в себя проверки полей ввода и списков на наличие контента в них (Рис. 15).

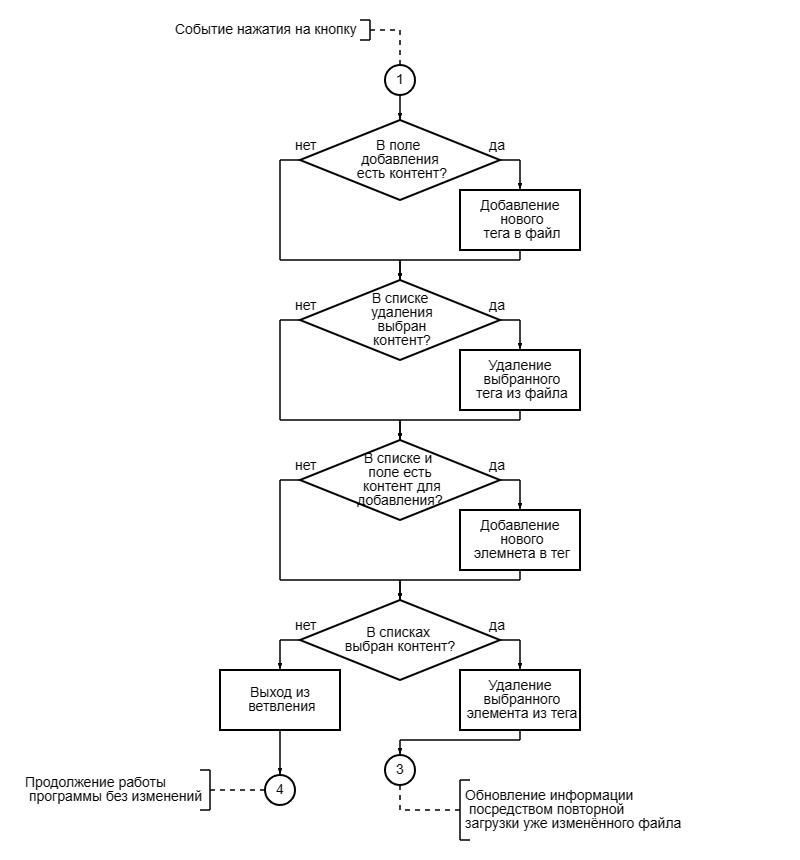


Рисунок 15 – Алгоритм применения изменений в файле

Также, значение любого элемента в теге можно редактировать, нажатием на кнопку «(изменить)» рядом с ним (Рис. 16).

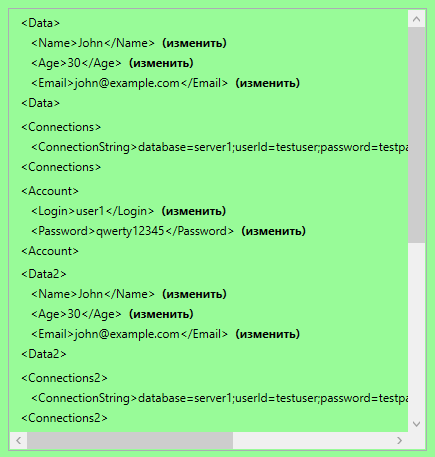


Рисунок 16 – Элементы списка, которые можно изменить

Для конвертации формата файла из XML в JSON или наоборот, необходимо на главной странице выбрать соответствующую кнопку, после чего откроется диалоговое окно, где выбирается файл с устройства и при нажатии на кнопку «Подтвердить» происходит конвертация файла.

* Разработка логики приложения

1. Настройка среды

Создание решения в IDE Visual Studio: выбираем «Приложение WPF (Майкрософт)», задаём имя решения «CourseProject» и имя для проекта WPF «Xml-Json RW Utility», в последствии будет добавлен ещё один проект для тестирования, поэтому мы разделяем имя решения и проекта WPF (Рис. 17).

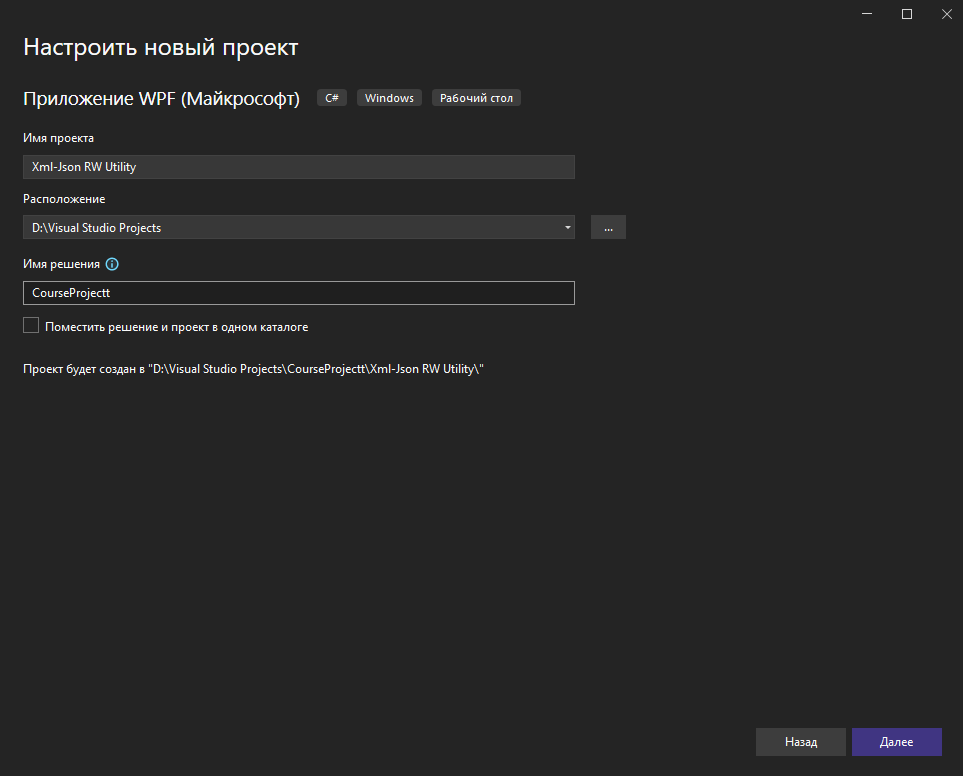


Рисунок 17 – Новое решение в Visual Studio

После создания проекта ожидаем подгрузки всех плагинов и окончания первоначальной сборки.

1. Установка пакетов

Для работы с форматом JSON потребуется установить специальный пакет в свой проект посредством пакетов NuGet, так как стандартные библиотеки языка поддерживают только работу с форматом XML. Для этого кликаем правой кнопкой мыши по проекту и выбираем «Управление пакетами NuGet» (Рис. 18)



Рисунок 18 – Переход в окно для управления пакетами NuGet

После чего откроется окно управления пакетами NuGet, где мы должны найти такой пакет как «Newtonsoft.Json» и установить его (Рис. 19).

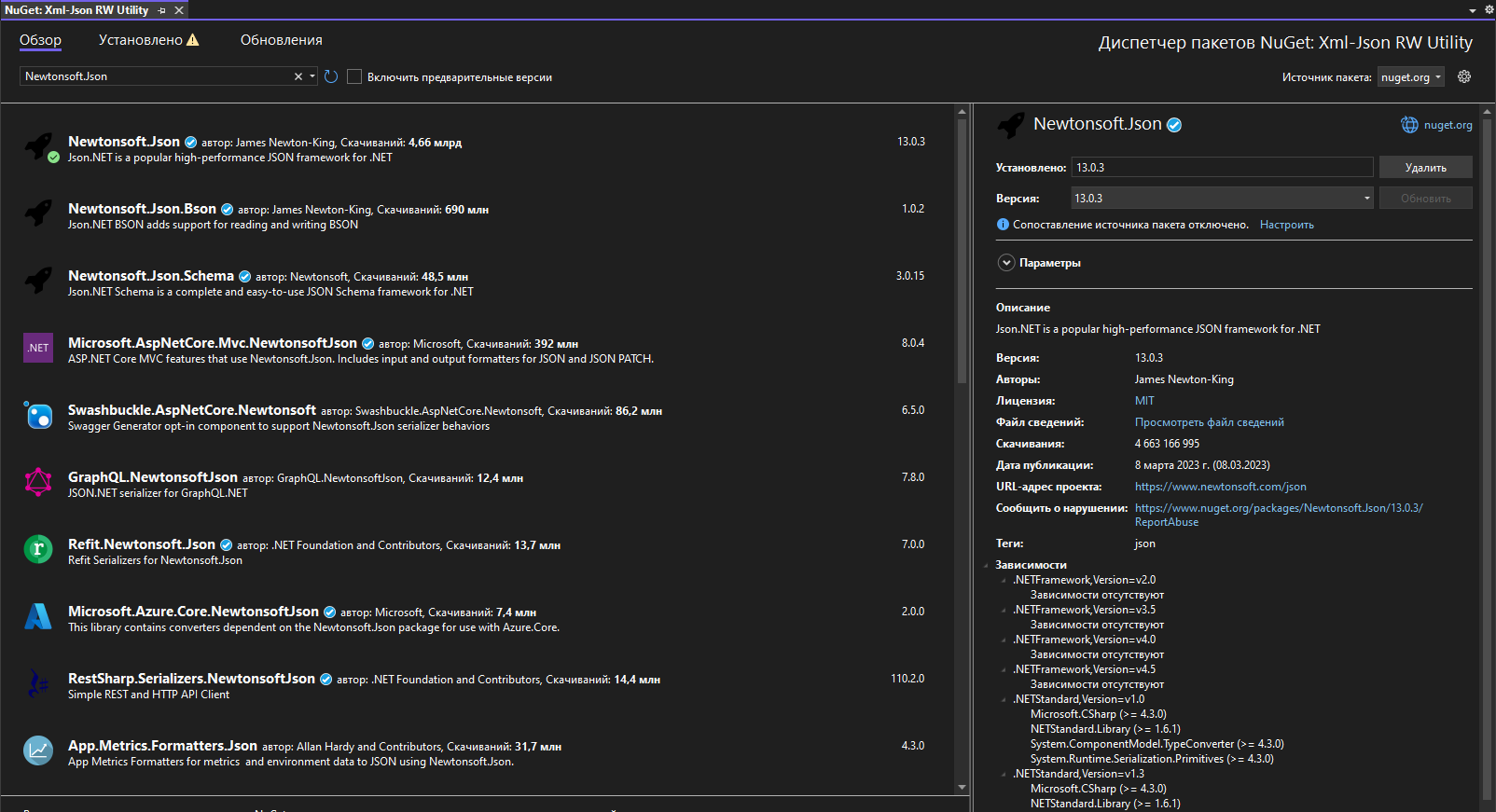


Рисунок 19 – Пакет «Newtonsoft.Json»

После установки пакета уже можно приступать к написанию кода для приложения.

1. Написание кода

Начать стоит с написания кода для главного окна, размещения в нём страниц, создания глобальных переменных для правильной работы приложения между страницами.

Далее стоит написать код для страницы чтения/записи файлов, в которой содержится весь основной функционал приложения, а также окна редактирования элементов файла.

В последнюю очередь стоит приступить к написанию кода для окна конвертации файлов, окна вывода информации об успехе/предупреждении/ошибке, а также окна подтверждения.

Завершить стоит написанием кода для обработки исключительных ситуаций и оформлению комментариев к коду, плавно переходя к составлению и написанию тестов.

* 1. Описание переменных, компонентов, классов и подпрограмм

Для описания основных классов приложения была сделана диаграмма классов (Рис. 20).

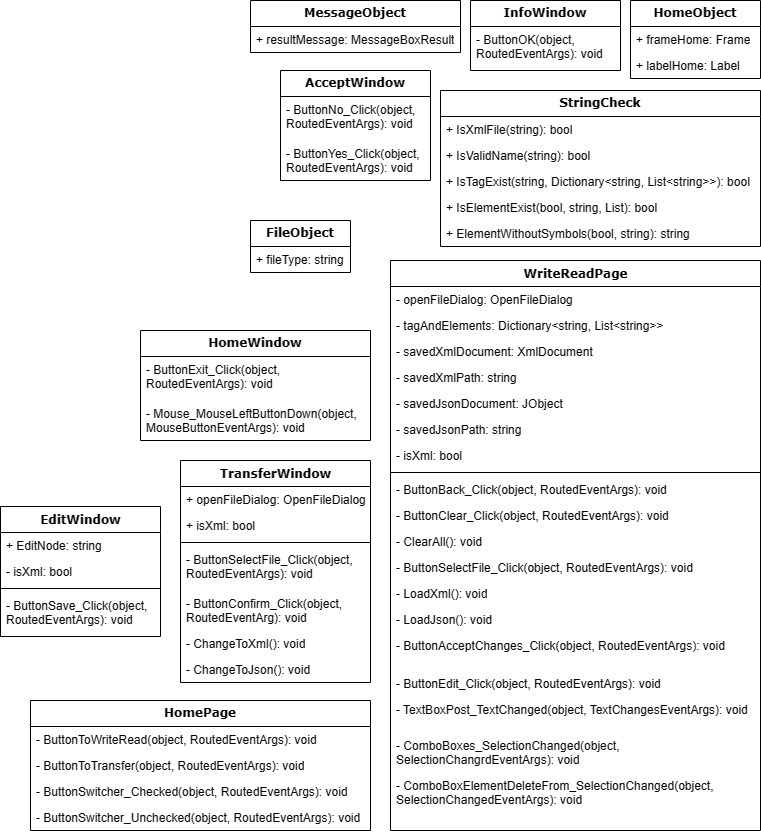


Рисунок 20 – Диаграмма основных классов

FileObject – класс, содержащий объекты используемого файла.

HomeObject – класс, содержащий объекты главного окна приложения.

MessageBoxObject – класс, содержащий объекты результата окна подтверждения.

StringCheck – класс, содержащий свои методы для обработки строк.

HomePage – класс, содержащий функционал главной страницы.

WriteReadPage – класс, содержащий функционал страницы чтения/записи.

AcceptWindow – класс, содержащий функционал окна подтверждения.

EditWindow – класс, содержащий функционал окна редактирования.

InfoWindow – класс, содержащий функционал окна с информацией.

TransferWindow – класс, содержащий функционал окна конвертации.

HomeWindow – класс, содержащий функционал главного окна приложения.

Для описания переменных, компонентов и подпрограмм была сделана Таблица 1.

Таблица 1 – Описание переменных, компонентов и подпрограмм.

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| Класс FileObject | |
| fileType | Хранит расширение файла, с которым в данный момент работа |
| Класс HomeObject | |
| frameHome | Хранит объект рамки, используется в качестве контейнера для страниц |
| labelHome | Хранит объект надписи в заголовке приложения |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| Класс MessageBoxObject | |
| resultMessage | Хранит результат окна подтверждения |
| Класс StringCheck | |
| IsXmlFile() | Метод, проверяющий, является ли файл XML или JSON |
| IsValidName() | Метод, проверяющий строку на наличие знаков препинания и если она начинается с цифры |
| IsTagExist() | Метод, определяющий, существует ли данный тег в ключах словаря |
| IsElementExist() | Метод, определяющий, существует ли данный элемент в списке по ключу словаря |
| ElementWithoutSymbols() | Метод, извлекающий подстроку имени элемента из строки с элементом |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| Класс HomePage | |
| ButtonToWriteRead() | Событие, отвечающее за нажатие кнопки «Чтение/запись», переходит на страницу чтения/записи |
| ButtonToTransfer() | Событие, отвечающее за нажатие кнопки «xml в json» («json в xml»), вызывает диалоговое окно конвертации файла |
| ButtonSwitcher\_Checked() | Событие, отвечающее за включённое состояние переключаемой кнопки, меняет тип файл для работы |
| ButtonSwitcher\_Unchecked() | Событие, отвечающее за выключенное состояние переключаемой кнопки, меняет тип файл для работы |
| Класс WriteReadPage | |
| openFileDialog | Хранит объект окна выбора файла в системе |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| tagAndElements | Словарь, хранящий в качестве ключей теги и в качестве списков элементы этого тега |
| savedXmlDocument | Хранит XML документ до изменения |
| savedXmlPath | Хранит путь к XML документу до изменения |
| savedJsonDocument | Хранит JSON документ до изменения |
| savedJsonPath | Хранит путь к JSON документу до изменения |
| isXml | Хранит результат булевого выражения, является ли файл xml |
| ButtonBack\_Click() | Событие, отвечающее за нажатие кнопки «Назад», переходит на главную страницу |
| ButtonClear\_Click() | Событие, отвечающее за нажатие кнопки «Сбросить», сбрасывает изменения в файле |
| ClearAll() | Метод, очищающий всю информацию на странице |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| ButtonSelectFile\_Click() | Событие, отвечающее за нажатие кнопки «Выбратб файл», вызывает окно выбора файла в системе |
| LoadXml() | Метод, отвечающий за загрузку XML файла |
| LoadJson() | Метод, отвечающий за загрузку JSON файла |
| ButtonAcceptChanges\_Click() | Событие, отвечающее за нажатие кнопки «Применить изменения», применяет изменения в файле в соответствии с контентом в полях ввода и выбранных списков |
| ButtonEdit\_Click() | Событие, отвечающее за нажатие на кнопку «(изменить)», открывает диалоговое окно изменения элемента |
| TextBoxPost\_TextChanged() | Событие, отвечающее за то, написан контент в полях ввода или нет |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| ComboBoxes\_SelectionChanged() | Событие, отвечающее за то, выбран контент в списках или нет |
| ComboBoxElementDeleteFrom\_SelectionChanged() | Событие, отвечающее за наполнение списка с подкатегорией элементов для выбранного тега в списке с категориями |
| Класс AcceptWindow | |
| ButtonNo\_Click() | Событие, отвечающее за нажатие на кнопку «Нет», закрывает диалоговое окно без изменений |
| ButtonYes\_Click() | Событие, отвечающее за нажатие на кнопку «Да», закрывает диалоговое окно с применением изменений |
| Класс EditWindow | |
| EditNode | Хранит новое значение для элемента тега |
| isXml | Хранит результат булевого выражения, является ли файл xml |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| ButtonSave\_Click() | Событие, отвечающее за нажатие на кнопку «Сохранить», закрывает диалоговое окно с сохранением результатов изменений |
| Класс InfoWindow | |
| ButtonOK\_Click() | Событие, отвечающее за нажатие на кнопку «ОК», закрывает диалоговое окно |
| Класс TransferWindow | |
| openFileDialog | Хранит объект окна выбора файла в системе |
| isXml | Хранит результат булевого выражения, является ли файл xml |
| ButtonSelectFile\_Click() | Событие, отвечающее за нажатие на кнопку «Выбрать файл», открывает окно выбора файла в системе |
| ButtonConfirm\_Click() | Событие, отвечающее за нажатие на кнопку «Подтвердить», закрывает окно и конвертирует файл |

Таблица 1 – продолжение

|  |  |
| --- | --- |
| Переменная, компонент, подпрограмма | Описание |
| ChangeToXml() | Метод, конвертирующий JSON в XML |
| ChangeToJson() | Метод, конвертирующий XML в JSON |
| Класс HomeWindow | |
| ButtonExit\_Click() | Событие, отвечающее за нажатие на кнопку закрытия приложения |
| Window\_MouseLeftButtonDown() | Событие, срабатывающее при нажатии ЛКМ по заголовку приложения, позволяет передвигать окно |

1. Тестирование и установка приложения

* Тестирование приложения.

1. Модульное тестирование.

Для модульного тестирования необходимо создать Unit-тесты. [6]

Для этой задачи был создан новый проект для тестирования «NUnit» в решении Visual Studio (Рис. 21).

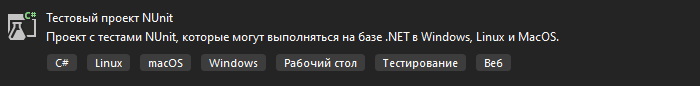


Рисунок 21 – Создание тестового проекта NUnit в Visual Studio

После добавления зависимости Xml-Json RW Utility в проект тестов, был написан Unit-тест для класса CheckString под названием StringCheckUnitTest, чтобы проверить, работают ли функции должным образом. StringCheckUnitTest включает в себя следующие тесты:

IsXmlFileTest – проверяет, является ли значение строки расширением для xml файла, в обратном случае это будет json расширение.

IsValidNameTest – проверяет, начинается ли входящая строка с цифры и содержит ли знаки препинания.

ElementWithoutSymbolsTest – выполняет извлечение подстроки из входящей строки для получения названия тега или элемента без дополнительных знаков.

После добавления тестовых данных и запуска тестов мы получили результат в виде пройденных всех трёх тестов (Рис. 22).

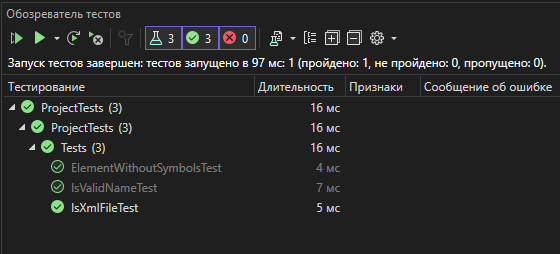


Рисунок 22 – Результат unit-теста

1. UI/UX тестирование.

Для тестирования удобства пользования попробуем максимально растянуть рамки окна (Рис. 23)

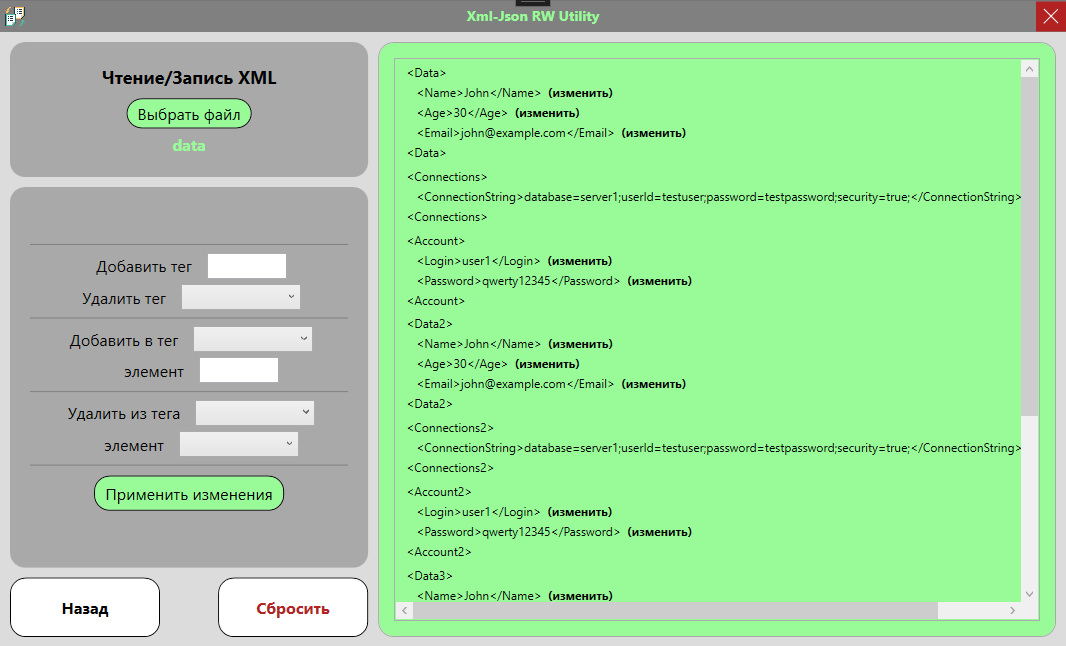


Рисунок 23 – Результат максимального расширения окна приложения

А также попробуем максимально сузить рамки окна (Рис. 24).

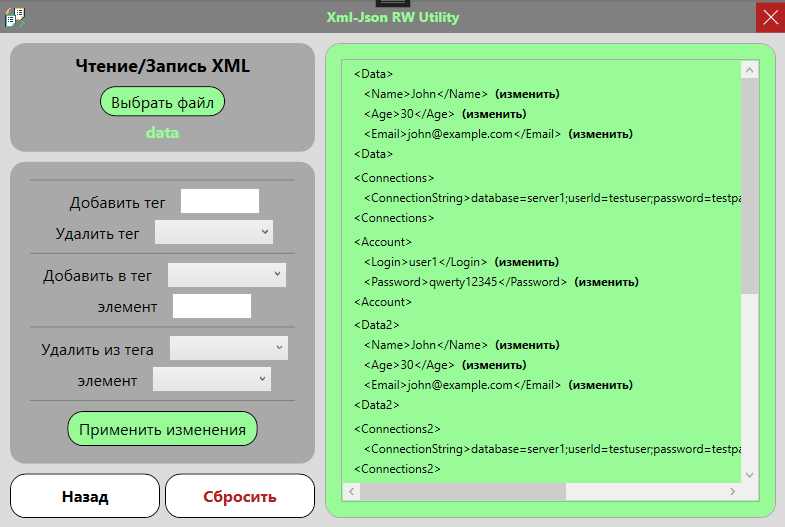


Рисунок 24 – Результат максимального сужения окна приложения

Как видно из результатов тестирования, контент и элементы на странице правильно отображаются и ничего не залазит друг на друга.

1. Системное тестирование.

Для системного тестирования приложения необходимо протестировать все компоненты и модули в целом. [2]

При запуске приложения нас успешно встречает главная страница приложения (Рис. 25).



Рисунок 25 – Главная страница приложения

Переходим на страницу чтения/записи и выбираем там файл data.xml (Рис. 26)

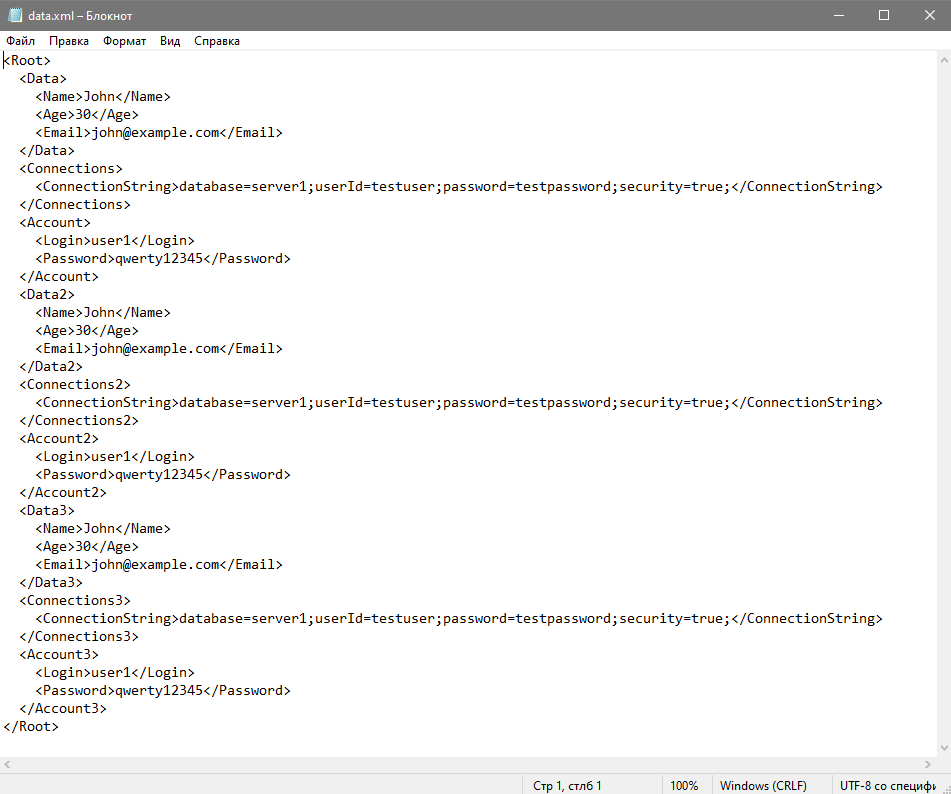


Рисунок 26 – Содержимое файла data.xml

После выбора файла приложение правильно отображает данные из файла в специальном прокручиваемом листе (Рис. 27)

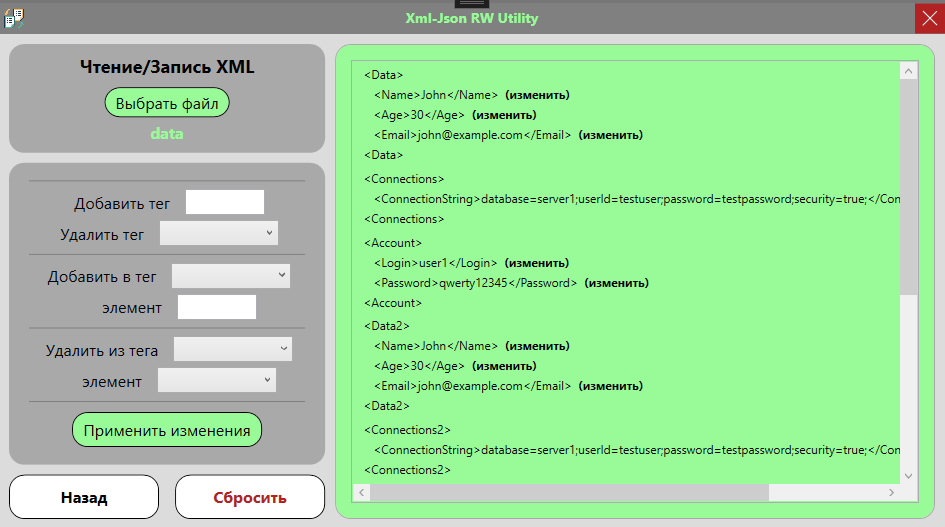


Рисунок 27 – Отображение содержимого файла

Попробуем изменить содержимое файла. Добавим тег «Test», удалим тег «Data», добавим элемент «NewElement» в тег «Test» и изменим его значение на value, удалим из тега «Account» элемент «Login» и нажмём на кнопку «Применить изменения» (Рис. 28).

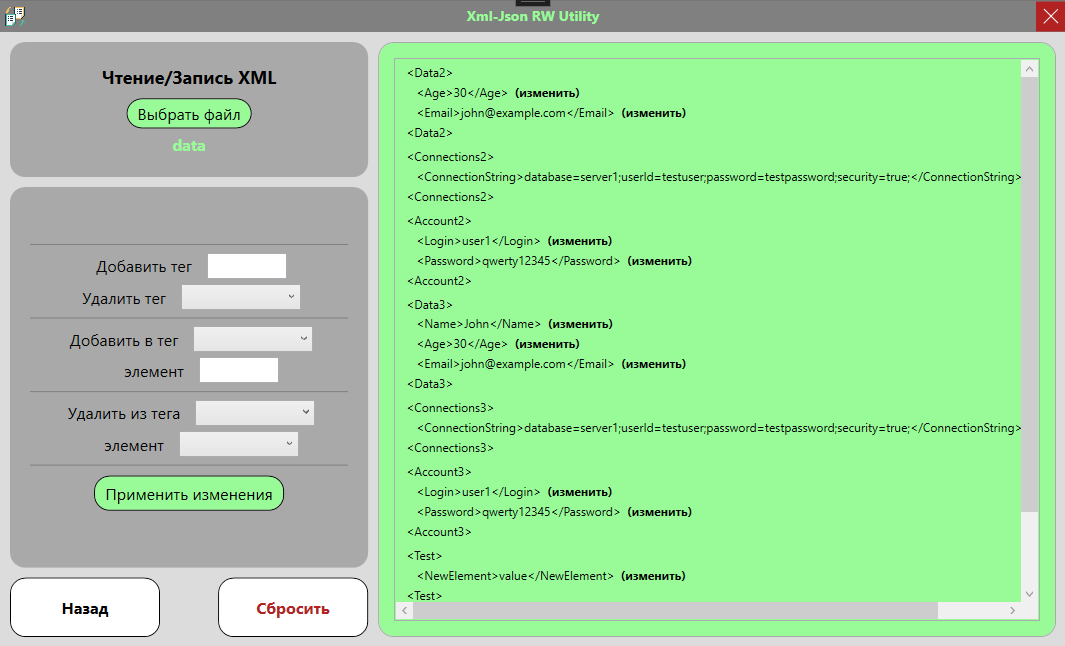


Рисунок 28 – Содержимое файла после произвольных операций

Попробуем сбросить все изменения, нажав на кнопку «Сбросить», после чего подтвердим действие в диалоговом окне (Рис. 29).

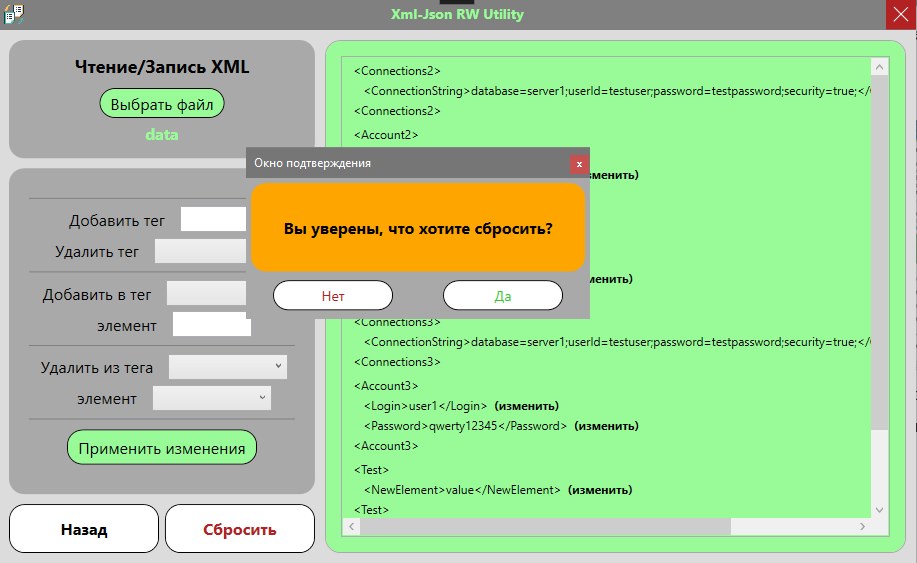


Рисунок 29 – Подтверждение сброса изменений

Как и ожидалось, все изменения были сброшены (Рис. 30).



Рисунок 30 – Сброшенные изменения

Вернёмся на главную страницу через кнопку «Назад» и нажмём на кнопку «xml в json», после чего в диалоговом окне выберем файл, с которым только что работали (Рис. 31).

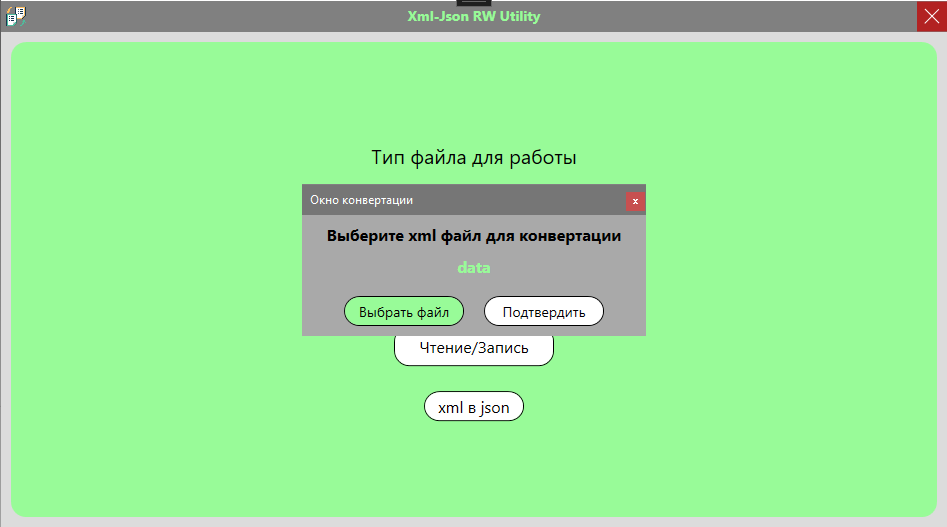


Рисунок 31 – Диалоговое окно конвертации файла

Теперь нажмём на кнопку «Подтвердить» и тем самым конвертируем xml файл в json (Рис. 32)

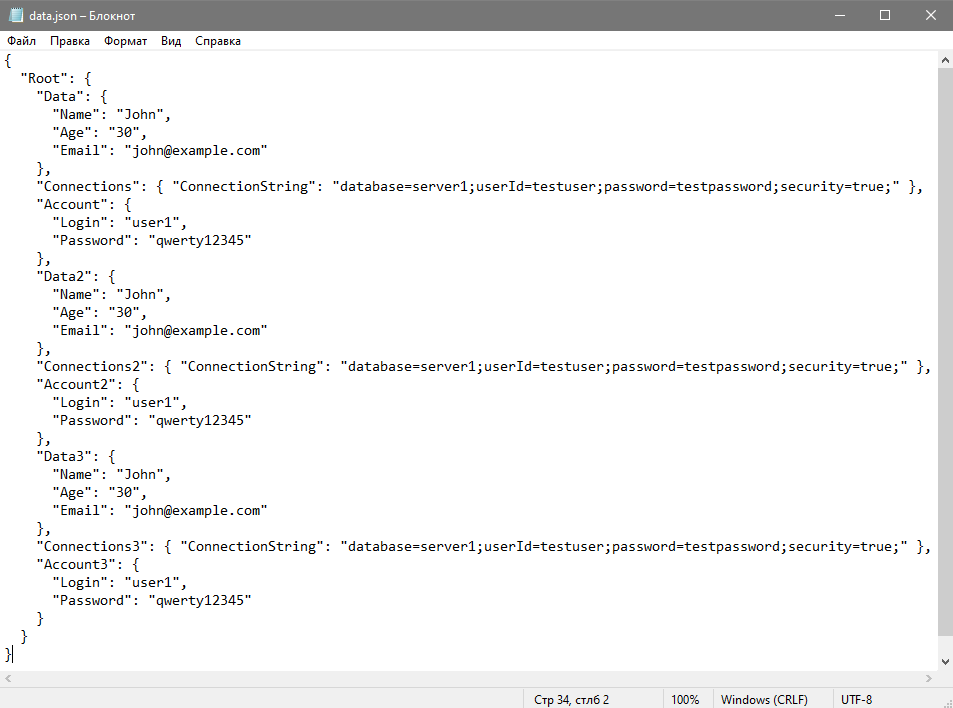


Рисунок 32 – Результат конвертации файла

По итогам системного тестирования приложение выполняет все свои функции правильно и работает безотказно. [10]

Листинг кода тестов приведён в приложении A.

* Установка приложения.

Приложение представляет из себя исполняемый файл, поэтому всего лишь необходимо скачать файл «Xml-Json RW Utility.exe» прямо на своё устройство. Потом достаточно запустить файл, после чего откроется приложение со всем предоставляемым функционалом.

Заключение

В результате выполнения данной курсовой работы было разработано приложение для чтения и записи файлов в форматах XML и JSON, что позволило более глубоко ознакомиться с особенностями работы с данными форматами. Изучение принципов работы с XML и JSON, их структуры и принципов кодирования данных позволило эффективно реализовать функциональность приложения и провести успешное тестирование его работы.

В рамках работы были изучены основные методы работы с файлами форматов XML и JSON, анализированы преимущества и недостатки каждого из форматов, а также рассмотрены способы обмена данными между различными системами. Разработанное приложение успешно справляется с задачей чтения и записи простых файлов, что делает его полезным инструментом для работы с данными в указанных форматах.

В целом, выполнение данной курсовой работы позволило углубить знания о работе с форматами XML и JSON, приобрести практические навыки разработки приложений для работы с данными форматами и расширить кругозор в области программирования. Полученный опыт и знания могут быть полезны как для разработки новых проектов, так и для оптимизации работы существующих систем, требующих обработки данных в форматах XML и JSON.

Список использованной литературы

1. Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. – СПб.: Питер, 2013. – 896 с.
2. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание – М.: «Вильямс», 2013. – 1024 с.
3. Герберт Шилдт. C# 4.0: полное руководство – М.: «Вильямс», 2010. – 1056 с.
4. Эндрю Троелсен. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание – М.: «Вильямс», 2013. – 1312 с.
5. Джозеф Албахари, Бен Албахари. C# 6.0. Справочник. Полное описание языка – М.: «Вильямс», 2018. – 1040 с.
6. Э. Стиллмен, Дж. Грин. Изучаем C#. 2-е издание, – СПб.: «Питер», 2012. – 704 с.
7. <https://stackoverflow.com> – решение ошибок и проблем.
8. <https://metanit.com/sharp/tutorial> – руководство по C# и .NET.
9. <https://metanit.com/sharp/wpf> – руководство по WPF.
10. <https://t.me/AIGPT4bot> – телеграмм бот с API GPT-4.

Приложения А. Листинг кода приложения

Код класса тестов StringCheckUnitTest.cs:

namespace ProjectTests

{

public class Tests

{

[SetUp]

public void Setup() {}

[Test]

public void IsXmlFileTest()

{

string fileType = ".xml";

Assert.AreEqual(true, StringCheck.IsXmlFile(fileType));

}

[Test]

public void IsValidNameTest()

{

string name = "f2).f";

Assert.AreEqual(false, StringCheck.IsValidName(name));

}

[Test]

public void ElementWithoutSymbolsTest()

{

string element = "<value>";

bool isXml = true;

Assert.AreEqual("value", StringCheck.ElementWithoutSymbols(isXml, element));

}

}

}

Код класса HomePage.xaml.cs:

using System.Windows.Controls;

using System.Windows.Media;

using Xml\_Json\_RW\_Utility.AppDataFile;

using Xml\_Json\_RW\_Utility.FunctionalWindows;

namespace Xml\_Json\_RW\_Utility.FunctionalPages

{

public partial class HomePage : Page

{

// Начальная страница программы

public HomePage(string fileType)

{

InitializeComponent();

// Определение типа файла для работы и соответствующего оформления окна для него

FileObject.fileType = fileType;

if (FileObject.fileType.Equals(".xml"))

{

DockPanel.SetDock(buttonSwitchType, Dock.Left);

buttonSwitchType.Background = Brushes.PaleGreen;

borderUI.Background = Brushes.PaleGreen;

buttonToTransfer.Content = "xml в json";

}

else

{

DockPanel.SetDock(buttonSwitchType, Dock.Right);

buttonSwitchType.Background = Brushes.Gold;

borderUI.Background = Brushes.Gold;

buttonToTransfer.Content = "json в xml";

}

}

// На страницу Чтения/Записи

private void ButtonToWriteRead(object sender, System.Windows.RoutedEventArgs e)

{

HomeObjects.frameHome.Navigate(new WriteReadPage(FileObject.fileType));

}

// Диалоговое окно конвертации

private void ButtonToTransfer(object sender, System.Windows.RoutedEventArgs e)

{

TransferWindow transferWindow = new TransferWindow(FileObject.fileType);

transferWindow.ShowDialog();

}

// Переключение на работу с JSON

private void ButtonSwitcher\_Checked(object sender, System.Windows.RoutedEventArgs e)

{

DockPanel.SetDock(buttonSwitchType, Dock.Right);

FileObject.fileType = ".json";

buttonSwitchType.Background = Brushes.Gold;

borderUI.Background = Brushes.Gold;

HomeObjects.labelHome.Foreground = Brushes.Gold;

buttonToTransfer.Content = "json в xml";

}

// Переключение на работу с XML

private void ButtonSwitcher\_Unchecked(object sender, System.Windows.RoutedEventArgs e)

{

DockPanel.SetDock(buttonSwitchType, Dock.Left);

FileObject.fileType = ".xml";

buttonSwitchType.Background = Brushes.PaleGreen;

borderUI.Background = Brushes.PaleGreen;

HomeObjects.labelHome.Foreground = Brushes.PaleGreen;

buttonToTransfer.Content = "xml в json";

}

}

}

Код класса WriteReadPage.xaml.cs:

using Microsoft.Win32;

using Newtonsoft.Json.Linq;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Xml;

using System.Xml.Linq;

using Xml\_Json\_RW\_Utility.AppDataFile;

using Xml\_Json\_RW\_Utility.FunctionalWindows;

namespace Xml\_Json\_RW\_Utility.FunctionalPages

{

public partial class WriteReadPage : Page

{

private OpenFileDialog openFileDialog;

private Dictionary<string, List<string>> tagAndElements;

private XmlDocument savedXmlDocument;

private string savedXmlPath;

private JObject savedJsonDocument;

private string savedJsonPath;

private bool isXml;

// Страница чтения/записи

public WriteReadPage(string fileType)

{

InitializeComponent();

openFileDialog = new OpenFileDialog();

tagAndElements = new Dictionary<string, List<string>>();

savedXmlDocument = new XmlDocument();

savedJsonDocument = new JObject();

// Определение типа файла для работы и соответствующего оформления окна для него

isXml = StringCheck.IsXmlFile(FileObject.fileType);

if (isXml)

{

openFileDialog.Filter = "XML Files (\*.xml)|\*.xml";

labelWriteRead.Content += " XML";

borderFileInfo.Background = Brushes.PaleGreen;

buttonSelectFile.Background = Brushes.PaleGreen;

buttonAcceptChanges.Background = Brushes.PaleGreen;

labelFileName.Foreground = Brushes.PaleGreen;

}

else

{

openFileDialog.Filter = "JSON Files (\*.json)|\*.json";

labelWriteRead.Content += " JSON";

borderFileInfo.Background = Brushes.Gold;

buttonSelectFile.Background = Brushes.Gold;

buttonAcceptChanges.Background = Brushes.Gold;

labelFileName.Foreground = Brushes.Gold;

}

}

// Возврат на главную

private void ButtonBack\_Click(object sender, RoutedEventArgs e)

{

HomeObjects.frameHome.Navigate(new HomePage(FileObject.fileType));

}

// Сброс информации в файле, если были изменения

private void ButtonClear\_Click(object sender, RoutedEventArgs e)

{

AcceptWindow acceptWindow = new AcceptWindow("Вы уверены, что хотите сбросить?", Brushes.Orange);

acceptWindow.ShowDialog();

if (MessageBoxObject.resultMessage == MessageBoxResult.Yes)

{

if (isXml)

{

try

{

savedXmlDocument.Save(savedXmlPath);

}

catch (Exception ex)

{

InfoWindow infoWindowError = new InfoWindow("Не удалось сбросить изменения.", Brushes.Firebrick);

infoWindowError.ShowDialog();

Console.WriteLine(ex.Message);

}

LoadXml();

}

else

{

try

{

File.WriteAllText(savedJsonPath, savedJsonDocument.ToString());

}

catch (Exception ex)

{

InfoWindow infoWindowError = new InfoWindow("Не удалось сбросить изменения.", Brushes.Firebrick);

infoWindowError.ShowDialog();

Console.WriteLine(ex.Message);

}

LoadJson();

}

textBoxTagPost.Text = null;

textBoxElementPost.Text = null;

buttonClear.IsEnabled = false;

buttonAcceptChanges.IsEnabled = false;

InfoWindow infoWindow = new InfoWindow("Информация успешно сброшена!", Brushes.LimeGreen);

infoWindow.ShowDialog();

}

MessageBoxObject.resultMessage = MessageBoxResult.No;

}

// Очистка содержимого всех полей на странице

private void ClearAll()

{

openFileDialog.FileName = "";

labelFileName.Content = "";

textBoxTagPost.Text = null;

textBoxElementPost.Text = null;

listBoxItemsList.Items.Clear();

tagAndElements.Clear();

comboBoxElementPostIn.ItemsSource = null;

comboBoxTagDelete.ItemsSource = null;

comboBoxElementDeleteFrom.ItemsSource = null;

comboBoxElementDelete.Items.Clear();

}

}

}

Код класса TransferWindow.xaml.cs:

using System.IO;

using System.Windows;

using System.Windows.Media;

using System.Xml;

using System.Xml.Linq;

using Microsoft.Win32;

using Newtonsoft.Json;

using Xml\_Json\_RW\_Utility.AppDataFile;

namespace Xml\_Json\_RW\_Utility.FunctionalWindows

{

public partial class TransferWindow : Window

{

private OpenFileDialog openFileDialog;

private bool isXml;

// Диалоговое окно конвертации файла

public TransferWindow(string fileType)

{

InitializeComponent();

openFileDialog = new OpenFileDialog();

// Определение типа файла для работы и соответствующего оформления для него

isXml = StringCheck.IsXmlFile(FileObject.fileType);

if (isXml)

{

buttonSelectFile.Background = Brushes.PaleGreen;

labelTypeTransfer.Content = "Выберите xml файл для конвертации";

openFileDialog.Filter = "XML Files (\*.xml)|\*.xml";

}

else

{

buttonSelectFile.Background = Brushes.Gold;

labelTypeTransfer.Content = "Выберите json файл для конвертации";

openFileDialog.Filter = "JSON Files (\*.json)|\*.json";

}

}

// Окно выбора файла и отображение его имени

private void ButtonSelectFile\_Click(object sender, RoutedEventArgs e)

{

if (openFileDialog.ShowDialog() == true)

{

if(isXml)

{

labelFileName.Content = openFileDialog.SafeFileName.Replace(".xml", "");

labelFileName.Foreground = Brushes.PaleGreen;

}

else

{

labelFileName.Content = openFileDialog.SafeFileName.Replace(".json", "");

labelFileName.Foreground = Brushes.Gold;

}

buttonAccept.IsEnabled = true;

}

}

// Подтверждение и выбор перевода файла

private void ButtonConfirm\_Click(object sender, RoutedEventArgs e)

{

if (isXml)

{

ChangeToJson();

}

else

{

ChangeToXml();

}

}

// Перевод JSON в XML

private void ChangeToXml()

{

try

{

string json = File.ReadAllText(openFileDialog.FileName);

XmlDocument doc = JsonConvert.DeserializeXmlNode(json, "Root");

string xmlFilePath = Path.ChangeExtension(openFileDialog.FileName, ".xml");

doc.Save(xmlFilePath);

XDocument xdoc = XDocument.Load(xmlFilePath);

XElement rootElement = xdoc.Element("Root");

xdoc.Element("Root").Remove();

rootElement.Elements().ToList().ForEach(e => xdoc.Add(e));

xdoc.Save(xmlFilePath);

File.Delete(openFileDialog.FileName);

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception ex)

{

InfoWindow infoWindow = new InfoWindow("Не удалось загрузить файл.", Brushes.Firebrick);

infoWindow.ShowDialog();

Console.WriteLine(ex.Message);

}

Close();

}

// Перевод XML в JSON

private void ChangeToJson()

{

try

{

XDocument xml = XDocument.Load(openFileDialog.FileName);

XDocument cleanedXml = new XDocument(xml.Root);

string cleanedXmlString = cleanedXml.ToString(SaveOptions.DisableFormatting);

string json = JsonConvert.SerializeXNode(XDocument.Parse(cleanedXmlString));

string jsonFilePath = Path.ChangeExtension(openFileDialog.FileName, ".json");

File.WriteAllText(jsonFilePath, json);

File.Delete(openFileDialog.FileName);

}

catch (ArgumentException ex)

{

Console.WriteLine(ex.Message);

}

catch (Exception ex)

{

InfoWindow infoWindow = new InfoWindow("Не удалось загрузить файл.", Brushes.Firebrick);

infoWindow.ShowDialog();

Console.WriteLine(ex.Message);

}

Close();

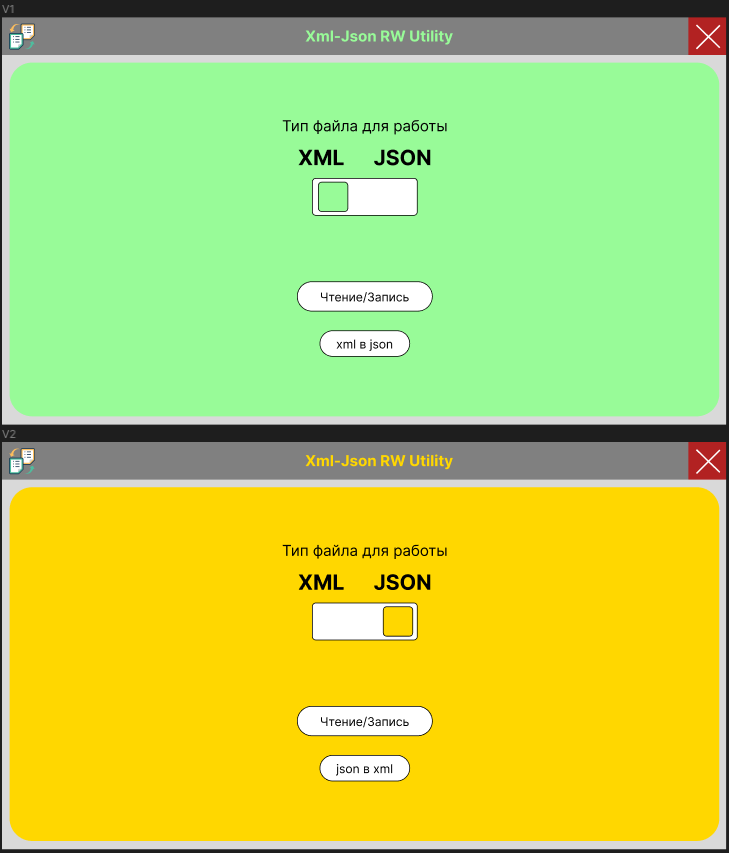
}

}

}

Приложение Б. Графические элементы приложения

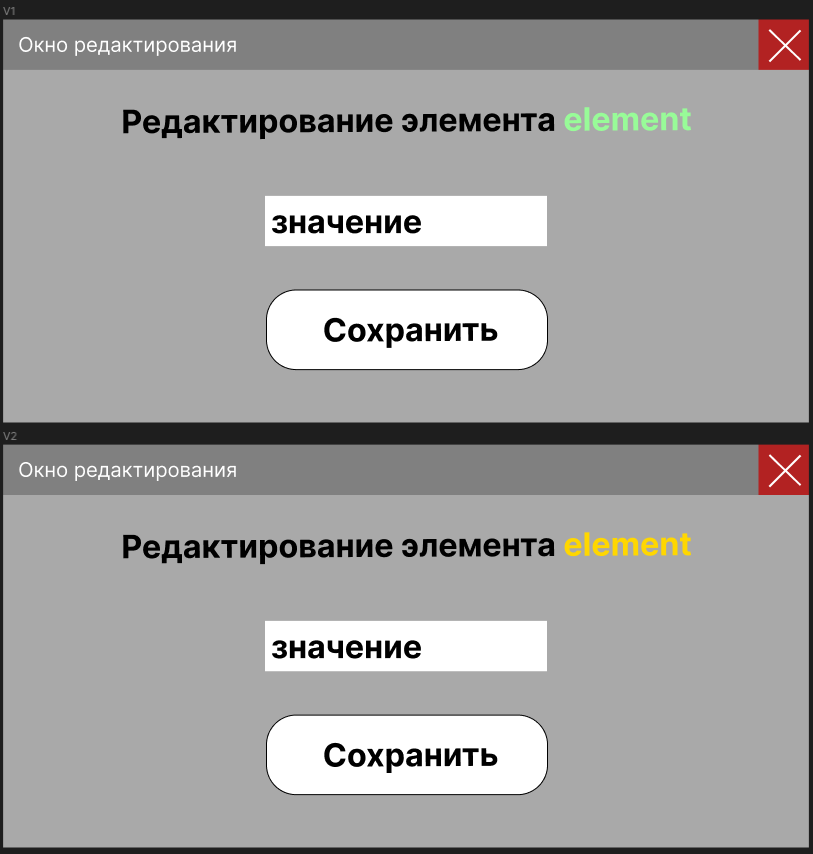
Главная страница приложения в 2 вариациях (PaleGreen и Gold):



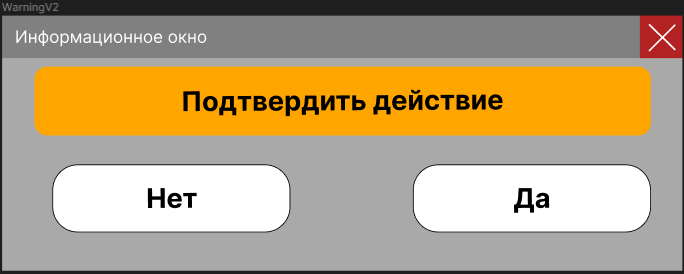
Страница чтения/записи файла в 2 вариациях (PaleGreen и Gold):



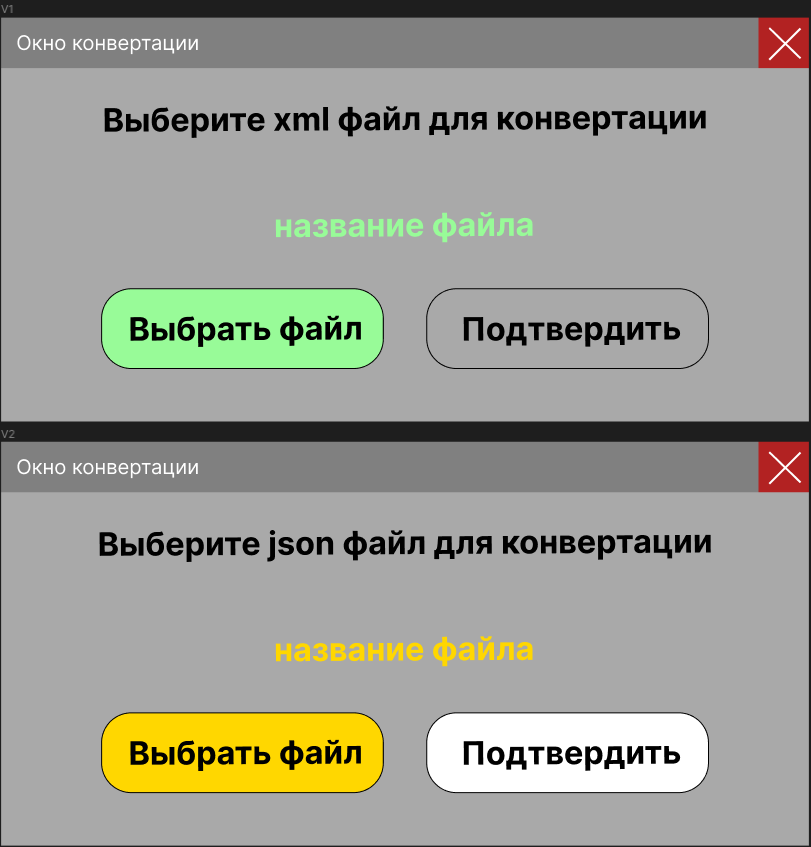
Окно редактирования элементов файла в 2 вариациях (PaleGreen и Gold):



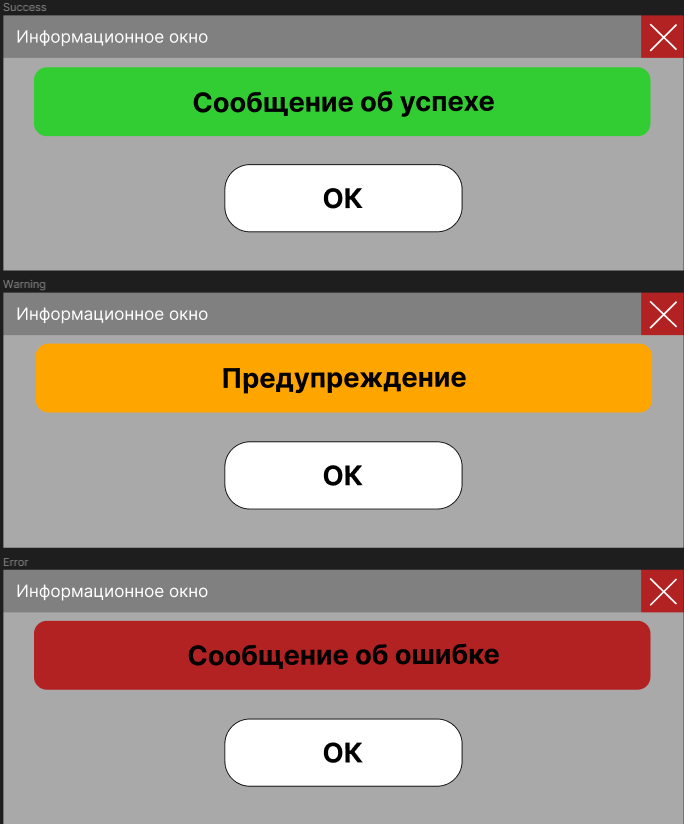
Диалоговое окно подтверждения действия (Orange):



Окно конвертации файла в 2 вариациях (PaleGreen и Gold):



Окна для обратной связи с пользователем (LimeGreen, Orange, FireBrick):



Приложение В. Руководство пользователя

1. Начало работы

Для начала работы с «Xml-Json RW Utility» необходимо скачать архив с файлами приложения и распаковать его на ваш компьютер с операционной системой Window 7/8/8.1/10. После чего достаточно просто запустить исполняемый файл «Xml-Json RW Utility» двойным кликом мышки по нему или через контекстное меню при нажатии по нему ПКМ.

1. Основные функции

* Переключение типа файла, с которым будем работать. На главной странице приложения есть специальный переключатель Xml-Json для того, чтобы поменять тип файла для работы.
* Отображение информации из xml/json файла. На странице чтения/записи после выбора файла, в левой части окна отображается информация из этого файла в виде списка.
* Редактирование узлов файла. После выбора файла на странице чтения/записи в левой части окна вы можете удалять узлы, добавлять узлы, добавлять элементы в узлы и удалять элементы из узлов с помощью специальных открывающихся списков и полей ввода.
* Редактирование элементов узлов. После выбора файла на странице чтения/записи в правой части окна вы можете нажать на кнопку «(изменить)» рядом с элементом и в диалоговом окне изменить значение этого элемента.

1. Дополнительные функции

* Конвертация одного типа файла в другой. На главной странице приложения есть кнопка для вызова диалогового окна конвертации файла, например, xml в json или наоборот.
* Сброс информации. На странице чтения/записи после выбора файла и вноса каких-либо изменений, вы можете сбросить их к состоянию до вноса изменений в файл, чтобы вернуть его в начальное состояние.
* Переключение темы. При переключении типа файла для работы меняется цветовая гамма приложения.
* Изменение размеров окна. При необходимости вы можете растянуть или сузить окно приложения для комфортного отображения контента в окне.

1. Советы и рекомендации

* Сохранение результатов. Обязательно нажимайте специальные кнопки для сохранения изменений или выполнения операций.
* Слабая универсальность. Приложение может обрабатывать не всё содержимое файлов, если присутствует сложная структура, не пытайтесь вносить изменения при виде неправильного отображения содержимого.

1. Помощь и поддержка

Если возникнут какие-то проблемы или вопросы при работе с приложением «Xml-Json RW Utility», вы можете обратиться за помощью или задать вопрос, написав на электронную почту разработчика: [ipygachev333@gmail.com](mailto:ipygachev333@gmail.com).