Федеральное государственное образовательное автономное учреждение

Высшего профессионального образования

«Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и управления»

Отчёт

по дисциплине «Управление конфигурациями программного продукта»

Выполнил:

студент группы ИВТм-211

Елькин А. С.

Проверил:

Старший преподаватель

Убалехт И. П.

Омск 2022

Оглавление

[Введение 2](#_Toc120809689)

[1. Разработка программного продукта 3](#_Toc120809690)

[2. Ручная сборка проекта 6](#_Toc120809691)

[3. Система контроля версий 7](#_Toc120809692)

[Заключение 7](#_Toc120809693)

[Список использованной литературы 7](#_Toc120809694)

# Введение

Управление конфигурацией программного продукта — это процесс системного проектирования, предназначенный для отслеживания изменений в конфигурационных метаданных программных систем.

Целью данной работы является освоение управлением различными конфигурациями разработанного программного продукта.

В результате выполнения работы должны быть выполнены следующие задачи:

* разработан программный продукт, который включает в себя как минимум 2 проекта в рамках одного решения;
* произведена ручная сборка получившегося проектного решения;
* показана работа с системой контроля версий по текущему проектному решению.

Работа разделена на 3 главы. Каждая глава соответствует конкретной задаче.

# 1. Разработка программного продукта

Для разработки подходящего приложения был выбран язык программирования C#, на платформе .NET.

Цель существования приложения – редактировать текст в существующих текстовых файлах, либо же создавать новые.

Для того чтобы разработанное приложение больше подходило под цели данной работы – было решено сделать его из двух проектов внутри одного проектного решения.

На рисунке 1 представлено дерево файлов разработанного решения. Оно включает в себя 2 проекта: FileSaver и SaveLibrary.

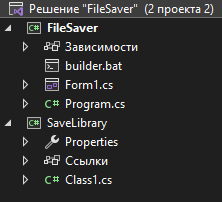


Рисунок 1 – Обозреватель решений

Есть рассматривать каждый проект по отдельности, то проект под названием FileSaver представляет из себя внешнюю оболочку разработанного приложения, которая включает в себя пользовательский интерфейс и вызовы написанной библиотеки SaveLibrary. Созданная библиотека состоит из одного класса, включающего в себя два метода: save и open.

Метод save принимает на вход текстовую строку, и предлагает пользователю создать текстовый файл, содержащий эту строку. Пользователь может сохранить получившейся файл куда захочет, также он может выбрать имя получившегося файла.

На рисунке 2 изображён внешний интерфейс программы.

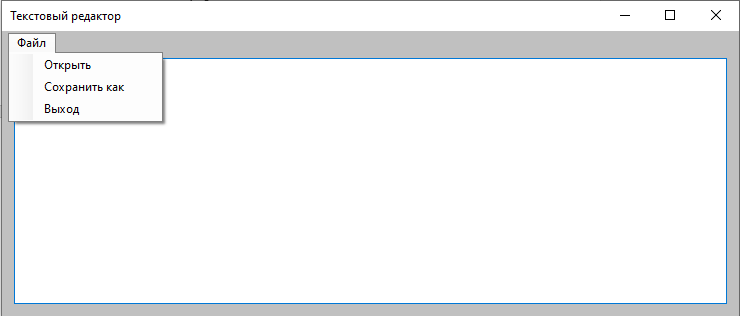


Рисунок 2 – Пользовательский интерфейс программы

Метод open наоборот, позволяет выбрать в обозревателе нужный файл из файловой системы, достать его содержимое, и вернуть это содержимое в пользовательский интерфейс. Далее это содержимое можно редактировать, а затем сохранять сколько угодно раз.

На рисунке 3 изображено окно выбора файла.

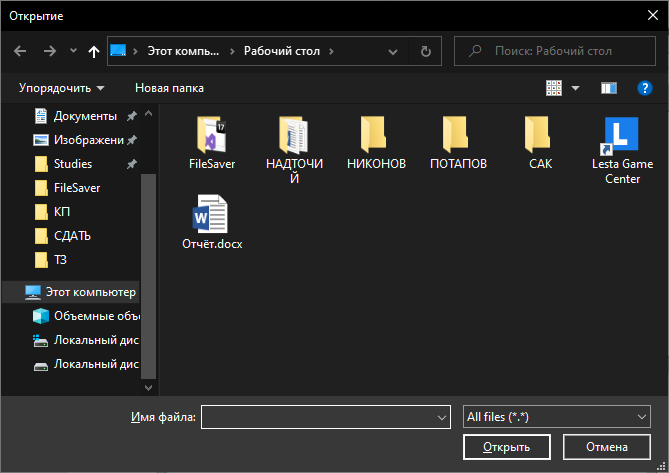


Рисунок 3 – Окно для выбора файла

С помощью этого окна происходит выбор файла при открытии, либо же выбор пути и имени файла, который будет создан при сохранении.

На рисунке 4 изображён разработанный код для метода save, который выполняется при сохранении файла.

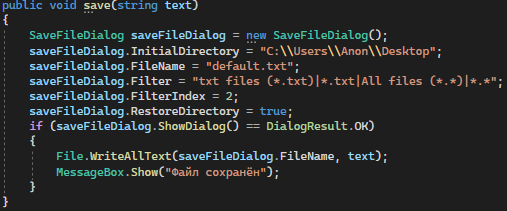


Рисунок 4 – Программный код метода save

На рисунке 5 проиллюстрирован программный код метода open, который выполняет функцию открытия текстового файла, и передачи его содержимого в пользовательский интерфейс.

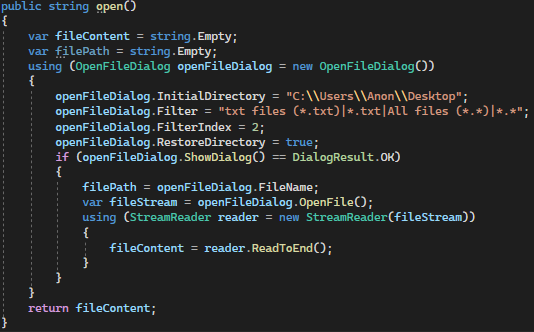


Рисунок 5 – Программный код метода open

Таким образом данную библиотеку можно использовать не только в разработанном решении, но и в любом другом, где надо открывать или сохранять текстовые файлы.

# 2. Ручная сборка проекта

Для сборки каждого проекта в рамках текущей работы будет использован bat скрипт, который выполняет ввод определённых команд в консоль Windows.

Содержимое bat файла представлено на рисунке 6.

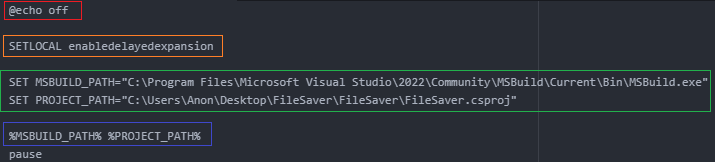


Рисунок 6 – Скрипт для сборки проекта

Разбор скрипта по строкам:

* красным цветом выделено «@echo off», это надо для того чтобы в консоль не выводились применённые команда, а выводились сразу результаты их применения;
* оранжевым цветом выделена команда «SETLOCAL enabledelayedexpansion». Эта строка состоит из двух слов, где «SETLOCAL» означает начало области задания для определения переменных среды в пакетном файле. А «enabledelayedexpansion» включает расширения переменной среды с задержкой до появления соответствующей команды endlocal, вне зависимости от состояния расширений командного процессора перед командой setlocal. Практическую пользу от отключения данного расширения назвать сложно, однако в процессе написания скрипта могут понадобиться некоторые функции, которые включает в себя это расширение, например, восклицательные знаки, которые не использовались конкретно в этом примере;
* зелёным цветом обозначен блок конфигураций, для каждого отдельного проекта этот блок будет свой, поскольку в нём идёт обозначение пути до самой программы MSBuild, а также пути для файла с расширением «.csproj», речь о котором пойдёт позже;
* синим цветом по сути идёт вызов сборщика MSBuild, которому на вход подаётся файл с «инструкциями» сборки.

Также в конце скрипта есть команда pause, для того чтобы успеть рассмотреть результаты ввода команд в консоль. Если не поставить скрипт на паузу – консоль после выполнения команд мгновенно закроется.

После пояснений, становится понятно, что скрипт всего лишь вызывает сборщик приложений, и даёт на вход xml файл, по которому собственно и идёт сборка. На рисунке 7 изображён код, который является инструкцией для сборки проекта FileSaver.

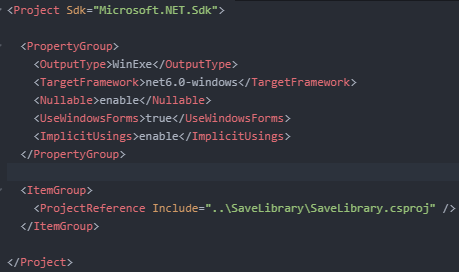


Рисунок 7 – Использование свойств сборки для проекта FileSaver

Данный файл подаётся на вход сборщику MSBuild, и его тоже следует немного разобрать:

* свойство «Project Sdk=“Microsoft.NET.Sdk”» означает подключение комплекта для разработки ПО к текущему проекту. К тому же, этот тег открывает область видимости для свойств конкретно этого проекта;
* свойство «PropertyGroup» является группировкой всех свойств, тег нужен для удобства чтения;
* свойство «OutputType» задаёт формат выходного файла;
* свойство «TargetFramework» определяет версию целевой платформы для приложения;
* свойство «Nullable» - это параметр компилятора C#;
* свойство «UseWindowsForms» определяет, предназначено ли приложение для Windows Forms;
* свойство «ImplicitUsings» можно использовать для включения и отключения неявных директив global using в проектах C#, предназначенных для .NET 6 или более поздней версии, и C# 10 или более поздней версии;
* в теге «ItemGroup» есть свойство «ProjectReference», которое указывает на раннее добавленную при разработки ссылку на проект SaveLibrary.

Как стало понятно – на представленных двух файлах работа скрипта не заканчивается. Как только MSBuild доходит до ссылки на проект, который является библиотекой для основного проекта, он переходит к следующе инструкции. Эта инструкция – точно такая же, как и первая, однако в ней намного больше свойств проекта.

На рисунке 8 изображена первая часть csproj файла в формате xml, который используется для сборки проекта SaveLibrary.

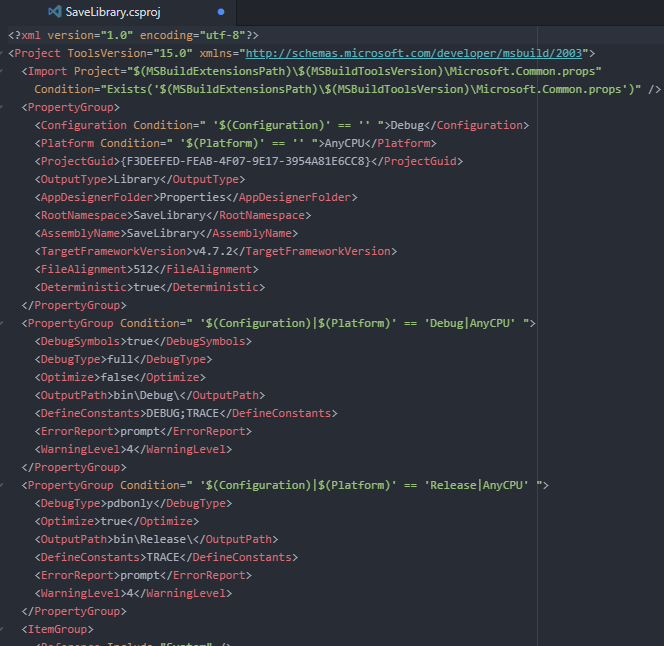


Рисунок 8 - Использование свойств сборки для проекта SaveLibrary (ч1)

На рисунке 9 изображена вторая часть этого же файла.



Рисунок 9 - Использование свойств сборки для проекта SaveLibrary (ч2)

Так как размеры этого файла достаточно большие для рассмотрения в рамках текущей работы – разбора каждого свойства не будет. В общем и целом, становится понятно, что с помощью таких инструкций MSBuild определяет какой именно проект требуется собрать, какие функции сборки включить и так далее.

На самом деле этих инструкций существует очень много, и далеко не все были представлены на проиллюстрированных рисунках. Все существующие инструкции (они же свойства сборки) описаны на официальном сайте Microsoft в разделе документации.

Так как в csproj файле указан режим сборки «Debug» - собранная версия в exe формате будет храниться именно в этой папке. Также существуют другие режимы сборки приложения, например, «Release».

На рисунке 10 изображён результат работы скрипта по сборке проектного решения.

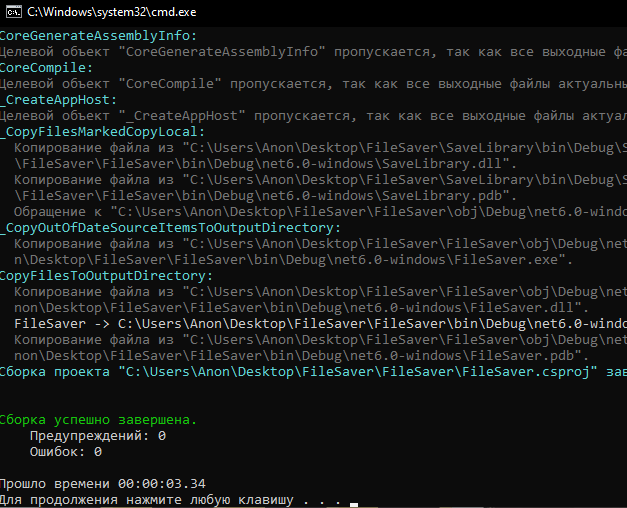


Рисунок 10 – Результат сборки приложения

Ну и в результате сборки появляется весь набор выходных файлов, в том числе файл с расширением executable, который и служит для запуска программы.

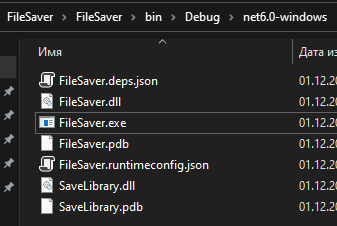


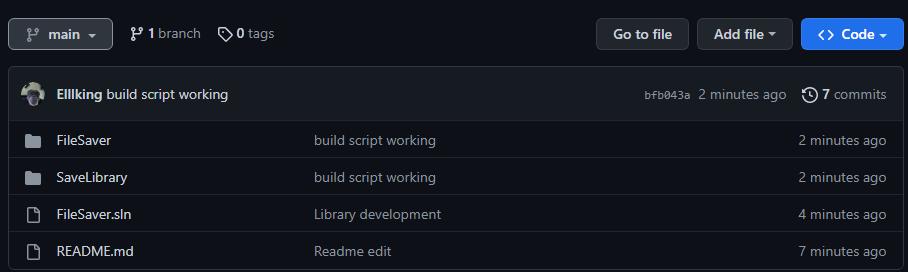
Рисунок 11 – Выходной файл сборки

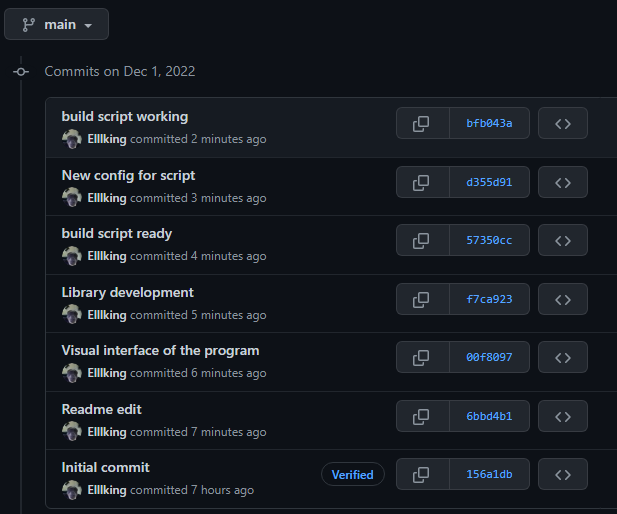
# 3. Система контроля версий

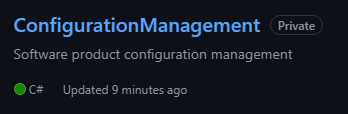
Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии.

В рамках текущей работы был создан отдельных репозиторий на github, а после этого была сымитирована поэтапная разработка рассматриваемого решения, состоящего из двух проектов. Это необходимо для лучшего представления работы с системой контроля версий, поскольку на момент написания отчёта приложение уже готово.









# Заключение

Таким образом, в рамках текущей работы было разработано небольшое

# Список использованной литературы

1. Model-View-Controller [Электронный ресурс] / Wikipedia. URL: [https://ru.wikipedia.org/wiki/Model-View-Controller,](https://ru.wikipedia.org/wiki/Model-View-Controller) свободный. Яз. Рус. (Дата обращения: 12.05.2020 г.)
2. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS / Дронов Владимир Александрович. - М.: БХВ-Петербург, 2018. - 228 c.
3. Конверс PHP 5 и MySQL. Библия пользователя / Конверс, др. Т. и. - М.: Вильямс, 2006. - 309 c.
4. PHP. Практика создания Web-сайтов / Кузнецов, М.В. и. - М.: БХВ-Петербург, 2008. - 963 c.
5. Каркас веб-приложений [Электронный ресурс] / Wikipedia. URL: [https://ru.wikipedia.org/wiki/Каркас\_веб-приложений,](https://ru.wikipedia.org/wiki/Каркас_веб-приложений) свободный. Яз. Рус. ((Дата обращения: 12.05.2020 г.)
6. Введение – Vue.js [Электронный ресурс] / Vue.js. URL: https://ru.vuejs.org/v2/guide/[,](https://ru.wikipedia.org/wiki/Интегрированная_среда_разработки) свободный. Яз. Рус. (Дата обращения 17.05.2020 г.)