Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра ИСиТ

Дисциплина: ОСиСП

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ПС Многопоточной загрузки файлов из сети Интернет

БГУИР КП 1-40.01.01.310 ПЗ

Студент: гр. 981063 Ефименко П.В.

Руководитель: Матвеев А.В.

Минск 2021

СОДЕРЖАНИЕ

[Введение 4](#_Toc69389867)

[1 Постановка задачи и составление технического задания 5](#_Toc69389868)

[1.1 Анализ существующих аналогов 5](#_Toc69389869)

[1.2 Обоснование выбора языка программирования 7](#_Toc69389870)

[1.3 Обоснование выбора среды разработки 7](#_Toc69389871)

[1.4 Постановка задачи 8](#_Toc69389872)

[1.5 Техническое задание 9](#_Toc69389873)

[1.6 Определение данных и их представление 9](#_Toc69389874)

[2 Проектирование программного средства 10](#_Toc69389875)

[2.1 Разработка спецификации требований к ПС 10](#_Toc69389876)

[2.2 Разработка структурной схемы ПС 11](#_Toc69389877)

[2.3 Разработка алгоритма работы ПС 11](#_Toc69389878)

[2.4 Обоснование выбора языка и среды программирования 12](#_Toc69389879)

[2.5 Программная реализация разработанного алгоритма работы ПС 13](#_Toc69389880)

[3 Тестирование и отладка ПС 14](#_Toc69389881)

[3.1 Тестовые примеры 14](#_Toc69389882)

[4 Руководство пользователя 17](#_Toc69389883)

[Заключение 19](#_Toc69389884)

[Список использованных источников 20](#_Toc69389885)

[Приложение А 21](#_Toc69389886)

[Приложение Б графическая часть проекта 27](#_Toc69389887)

Введение

На сегодняшний день существует множество программы для скачивания файлов из сети Интернет, они имеют широкий и узкий набор инструментов, работают на разных сетевых протоколах и по-разному могут нагружать компьютер. Этим программные средства могут быть разработаны так, что их можно внедрить в другое программное средство.

Целью курсового проекта является разработка библиотеки и программного средства, которое будет использовать данную библиотеку. Чтобы скачивать файлы по ссылке из сети Интернет в многопоточном режиме, с возможностью отмены, загрузки для кроссплатформенного использования.

Также в рамках курсового проектирования была разработана пояснительная записка.

Пояснительная записка содержит следующие разделы:

1) Раздел «Постановка задачи и составление технического задания» содержит формулировку задания с указанием всех функций проекта, определение круга задач, которые должны быть автоматизированы или модифицированы в данном курсовом проекте. Описание входных, выходных данных в проекте.

2) Раздел «Проектирование программного средства» содержит описание внешнего пользовательского интерфейса, описание функций и алгоритмов и структуры справочной системы и средств её разработки.

3) Раздел «Тестирование программного средства» содержит тестовые примеры проверки работы программы в нормальных и экстремальных условиях, анализ ожидаемых и полученных результатов.

4) Раздел «Руководство по использованию программного средства» содержит сведения о назначении программного средства, требования к необходимым для данного программного средства техническим и программным ресурсам для работы программы, описание установки и запуска программы.

В «Заключении» описывается краткая формулировка проблемы и пути ее решения, использованные методы и средства, возможность дальнейшей модификации проекта.

Приложение содержит код программы и разработанные тесты, а графическая часть состоит из созданных для курсового проекта диаграмм

1. Постановка задачи и составление технического задания
   1. Анализ существующих аналогов

На сегодняшний день имеется множество менеджеров закачки файлов, которые работают с разными протоками. Менеджеры закачки (часто называемые и менеджерами загрузки) значительно облегчают процесс скачивания файлов с серверов в интернете на локальный компьютер. Пользователь, посещая различные сайты, может добавить в менеджер ссылки на файлы, которые он хотел бы скачать и запустить их закачку после того, как сёрфинг закончен. Таким образом, закачка файлов при помощи менеджера может быть выполнена в то время, когда пользователь не работает в интернете. Многие менеджеры закачки позволяют указать время, когда необходимо скачать файлы, что бывает удобно при использовании диал-ап доступа в интернет: список файлов для закачки можно сформировать днём и запланировать закачку на ночное время, когда действует более дешёвый тариф на услуги доступа в сеть. В этом случае, менеджер сам дозвонится до провайдера, скачает файлы и выключит компьютер. Помимо этого, менеджеры могут докачивать файлы, то есть, если закачка была прервана и файл не был скачан полностью, то менеджер после повторного соединения с сервером продолжит скачивание файла с того места, где была прервана закачка, а не будет скачивать весь файл заново. На этом список возможностей менеджеров закачки не заканчивается, они могут интегрироваться с браузером, автоматически снижать скорость закачки, когда пользователь работает в интернете, хранить учётные данные для доступа к определённым серверам и так далее.

Сайт Download Accelerator Plus (speedbit.com)

Размер дистрибутива 3 мегабайта, бесплатная версия с ограничением по количеству потоков (10), с рекламой, отключенными возможностями по работе с ZIP-архивами и другими ограничениями. Полная версия стоит 30$, русский интерфейс.

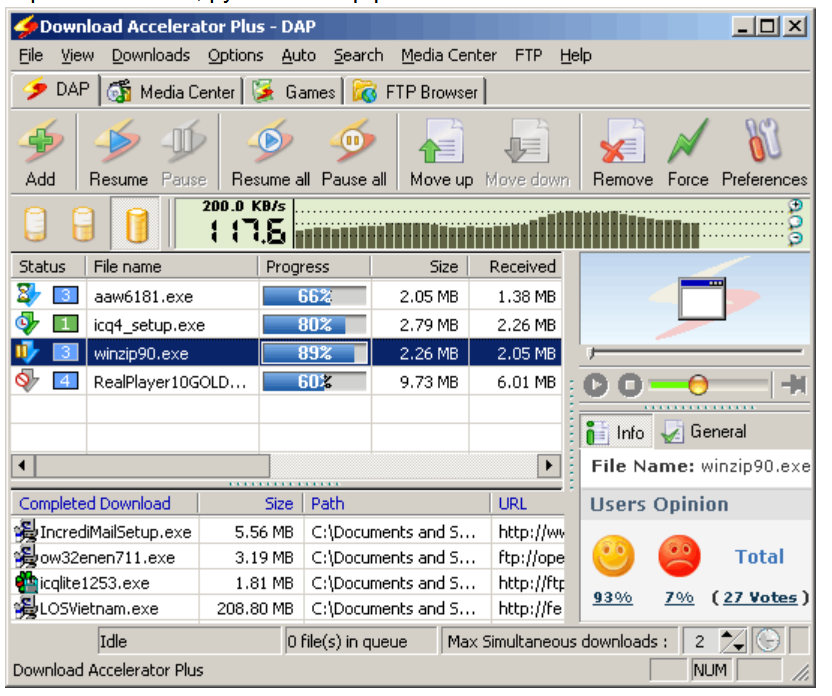


Рисунок 10.1 – Download Accelerator Plus.

Сайт Download Master (westbyte.com)

Размер дистрибутива 2 мегабайта, бесплатный, русский интерфейс.

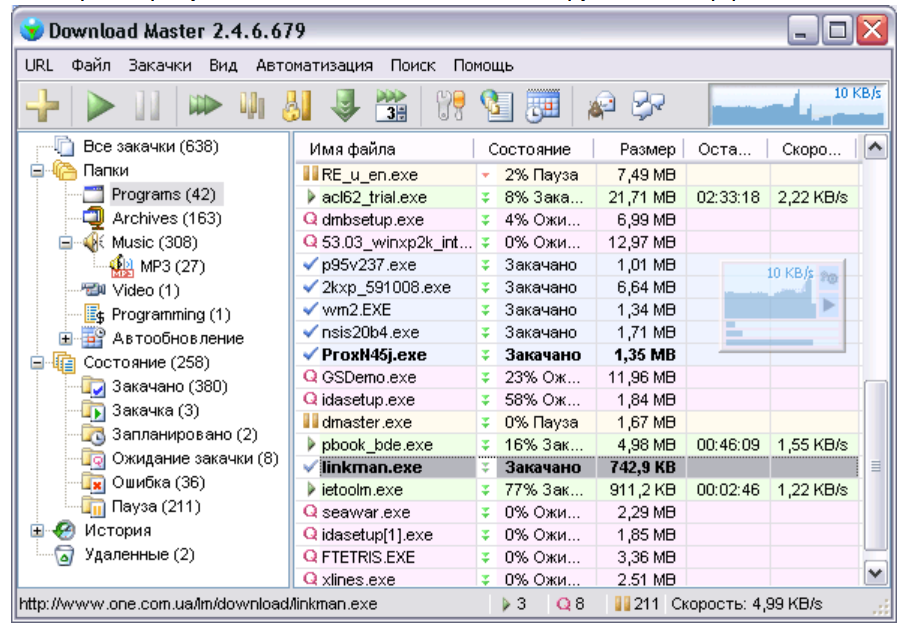


Рисунок 10.2 – Download Master.

* 1. Обоснование выбора языка программирования

Целевой технологией разработки модуля является .NetCore. Данная технология позволяет уменьшить затраты на разработку программного обеспечения, рассчитанного на разные платформы, так как является гибридной технологий, тем самым сократить количество разработчиков в команде. NetCore поддерживает исключительно С#. Соответственно целевым языком программирования является C#.

В качестве языка программирования для разрабатываемого модуля выбран язык программирования C#. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящее время на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей. C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. C# является объектно-ориентированным, и в этом плане он многое перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

При разработке использована гибридная технология NetCore. NetCore – это фреймворк, или же гибридная технология для кроссплатформенной с использованием языка C#. NetCore по заявлениям разработчиков является средством кроссплатформенной разработки, то есть. ожидаемо, что приложение, написанное один раз, может быть запущено на различных платформах.

* 1. Обоснование выбора среды разработки

В качестве среды программирования для разрабатываемого проекта выбрана среда программирования Visual Studio 2019. Microsoft Visual Studio 2019 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio. Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных.

Плюсами данной среды являются, что с каждой версией инструментов Microsoft старается учесть пожелания разработчиков и сделать их удобнее для создания приложений практически для любой платформы. Результатом является огромный интерес и более 21 миллион установок инструмента. Данная среда распространяется абсолютно бесплатно по сравнению с другими программными решениями. Продукт постоянно обновляется и добавляются новые возможности для разработки приложений и другие языки программирования.

Минусом будет является отладка в Visual Studio. При удаленной сборке приложения, процесс отладки может прекратить работу по истечению время ожидания. Хотя, опять же, стоит отдать должное разработчикам – они исправляют ошибки достаточно интенсивно. Хотя стоит заметить, что на текущий момент, времени между запуском приложения и появлением его на экране симулятора при использовании Visual Studio требуется несколько больше, чем при использовании Visual Studio на Mac.

* 1. Постановка задачи

Необходимо разработать программное средство для скачивания файлов из сети. Программное средство должно иметь графический, интуитивный интерфейс. Менеджер закачек должен работать на операционной системе Windows семейства Windows NT.

Интерфейс должен состоять из одного окна с полем ввода ссылки на файл и кнопкой скачать. Ниже от данного блока должен отображаться список загруженных файлов и текущих загрузок.

Блок активной загрузки должен состоять из наименования файла, его размера в байтах, источник публикации (хост, с которого скачивается файл), дату и время загрузки файла, а также кнопку чтобы перейти к загруженному файлу и панель степени загрузки файла.

* 1. Техническое задание

Данное программное средство должно быть создано для пользователей Windows 10 с поддержкой 32-битных систем. А также должен быть реализован на языке C# с поддержкой .NetCore. Графический пользовательский интерфейс необходимо реализовать на основе WPF. Программное средство должно занимать места на жестком диске не более 200 мегабайт. Для проекта необходимо провести функциональное тестирование и составить для него пояснительную записку, описывающее назначение, функционал и общее описание программы. В случае необходимо, к проекту разрешено подключать сторонние библиотеки

* 1. Определение данных и их представление

Входные данные – это данные, получаемые из первичной информации и создающие информационный образ предметной области. Они могут быть представлены действиями пользователя, ввод предложений выбор пунктов. Входными данными пользователя будут являться:

− ссылка на файл;

− путь для сохранения;

− отмена скачивания.

Выходные данные – это то, что отобразится, или то, что мы получим в результате манипуляций при работе с программой.

Выходными данными программного средства будет состояние загрузки и скаченный файл.

1. Проектирование программного средства
   1. Разработка спецификации требований к ПС

Программное средство разработано для .NET Core приложений по спецификации .NET Core, а именно Linux, UWP.

Спецификация .NET Core представляет собой официальную спецификацию интерфейсов API .NET, которые должны быть доступны во всех реализациях .NET. Спецификация .NET Core создана для того, чтобы повысить согласованность экосистемы .NET. Стандарт ECMA 335 продолжает обеспечивать единообразие для реализации .NET.

Спецификация .NET Core позволяет унифицировать набор API-интерфейсов для реализации алгоритма, а разработчикам создавать переносимые библиотеки, которые могут использоваться в разных реализациях .NET, с помощью одного набора api-интерфейсов.

Различные реализации .NET реализуют конкретные версии .NET Core. Каждая версия реализации .NET ориентирована на использование максимальной поддерживаемой ею версии .NET Core. Это также означает, что она поддерживает и предыдущие версии.

Алгоритм должен учитывать особенности работы данных операционных систем, выполнять свои действия как можно быстрее.

Также алгоритм должен быть спроектирован по закону Диметры. Это набор правил проектирования при разработке программного обеспечения, в частности объектно-ориентированных программ, накладывающий ограничения на взаимодействия объектов (модулей). Обобщенно, закон Деметры является специальным случаем слабой связанности. Преимуществами закона Деметры является то, что код, разработанный с соблюдением данного закона, делает написание тестов более простым, а разработанное программное обеспечение менее сложно при поддержке и имеет большие возможности повторного использования кода. Так как объекты являются менее зависимыми от внутренней структуры других объектов, контейнеры объектов могут быть изменены без модификации вызывающих объектов (клиентов).

* 1. Разработка структурной схемы ПС

Физическая структура проекта имеет следующие библиотеки:

1. Microsoft .NET Core App – целевой проект.
2. Syncfusion – набор инструментов кроссплатформенного пользовательского интерфейса.
3. .NET Core – представляет собой официальную спецификацию интерфейсов API .NET, которые должны быть доступны во всех реализациях .NET. .NET Core создана для того, чтобы повысить согласованность экосистемы .NET. ECMA 335 продолжает обеспечивать единообразие для реализации .NET.
4. Newtonsoft.Json– набор API для взаимодействия с форматом Json.
5. CN\_and\_SYS\_Downloader– разработанный модуль приложения.

Логическая структура демонстрационного приложения состоит из:

1. IContainer – интерфейс позволяющих получить доступ к сервисам программного средства.
2. IMemoryManager – интерфейс позволяющих управлять памятью приложения.
3. IConfig –интерфейс позволяющий получить доступ к настройкам приложения.
4. IRequestService –интерфейс с помощью, которого осуществляется запрос на скачивание файла.
5. IApiService – интерфейс с помощью, которого происходит загрузка файла.
   1. Разработка алгоритма работы ПС

При запуске программы происходит определение настроек приложения, где указывается количество попыток соединения с хостом, тайм-аут подключения и время ожидания ответа от сервера.

Далее производит инициализация сервис-локатора – IContainer. С помощью него получаем лицензионный ключ библиотеки интерфейса из IConfig, регистрируется библиотека.

Затем происходит инициализация графического интерфейса, создается модель визуализации, в которой идет запыление данных.

При вводе в поле URL, значение переменной присваивается в модели полю LoadFromUrl. При нажатии на кнопку загрузки срабатывает обработчик LoadFileCommand. Идет проверка на том, что данные не пустые, создается элемент загрузки, добавляется в очередь загрузки и происходит скачивание. В случае возникновения проблем при скачивании, система производит 3 попытки чтобы загрузить файл, в противном случае загрузка останавливается.

При успешной загрузке файл сохраняется в место, куда указал пользователь при загрузке файла.

* 1. Обоснование выбора языка и среды программирования

В качестве языка программирования для разрабатываемого модуля выбран язык программирования C#. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящее время на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей. C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. C# является объектно-ориентированным, и в этом плане он многое перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

В качестве среды программирования для разрабатываемого проекта выбрана среда программирования Visual Studio 2019. Microsoft Visual Studio 2019 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio. Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных.

Плюсами данной среды являются, что с каждой версией инструментов Microsoft старается учесть пожелания разработчиков и сделать их удобнее для создания приложений практически для любой платформы. Результатом является огромный интерес и более 21 миллион установок инструмента. Данная среда распространяется абсолютно бесплатно по сравнению с другими программными решениями. Продукт постоянно обновляется и добавляются новые возможности для разработки приложений и другие языки программирования.

* 1. Программная реализация разработанного алгоритма работы ПС

Сервис ProgressHandler – позволяет отслеживать прогресс загрузки файла.

Сервис IRequestService – отправляет запрос на загрузку файла, там указывается адрес файла, хост, заголовки запроса, время отклика от сервера и жетон отмены действия.

Сервис RequestBuilder – помогает построить запрос на основе входных данных и настроек приложения.

Сервис IApiService – скачивает файл по заданной ссылке.

Сервис IMemoryManager – позволяет добавить файл в хранилище, удалить файл, получить максимальный размер хранилища, получить путь к файлу, сохранить состояние сервиса и задать максимальный размер хранилища.

1. Тестирование и отладка ПС
   1. Тестовые примеры

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий две различные цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* –выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

При анализе качества выполняемого программного обеспечения, в компании, где разрабатывался модуль, придерживаются описанных ниже критериев.

Качество программного обеспечения имеет внешние и внутренние характеристики. К внешним характеристикам относятся свойства, которые осознает пользователь программы. Они описаны ниже.

Корректность –отсутствие/наличие дефектов в спецификации, проекте и реализации системы.

Практичность –легкость изучения и использования системы.

Эффективность –степень использования системных ресурсов. Эта характеристика учитывает такие факторы, как быстродействие приложения и требуемый им объем памяти.

Надежность –способность системы выполнять необходимые функции в предопределенных условиях; средний интервал между отказами.

Адаптируемость –возможность использования системы без ее изменения в тех областях или средах, на которые она не была ориентирована непосредственно.

Правильность –степень безошибочности системы, особенно в отношении вывода количественных данных. Правильность характеризует выполнение системой ее функций, а не то, создана ли она корректно. Этим правильность отличается от корректности.

Внутренние характеристики качества, которые перечислены ниже.

Удобство сопровождения –легкость изменения программной системы с целью реализации дополнительных возможностей, повышения быстродействия, исправления дефектов и т. д.

Гибкость –возможный масштаб изменения системы с целью использования ее в тех областях или средах, на которые она не была непосредственно ориентирована.

Портируемость –легкость изменения системы с целью использования в средах, на которые она не была ориентирована непосредственно.

Возможность повторного использования –масштабность и легкость использования частей системы в других системах.

Удобочитаемость –легкость чтения и понимания исходного кода системы, особенно на детальном уровне отдельных операторов.

Тестируемость –возможная степень выполнения блочного и системного тестирования программы и проверки ее соответствия требованиям.

Понятность –легкость понимания системы и на уровне общей организации, и на детальном уровне отдельных операторов. Понятность характеризует согласованность системы на более общем уровне, чем удобочитаемость.

Для того чтобы произвести проверку разработанного программного модуля, необходимо сформулировать перечень принципов тестирования модуля системы и в соответствии с ними произвести тестирование. Так как вариантов тестирования множество, то разработанный программный модуль будет протестирован по следующим классификациям:

– тестирование производительности. Это тестирование проводится с целью определения того, как быстро работает система, или её часть, под определенной нагрузкой. Также оно может служить для проверки и подтверждения других атрибутов качества системы, таких, как масштабируемость, надёжность и потребление ресурсов. Данный вид тестирования так же включает в себя и стресс-тестирование, которое используется для понимания пределов пропускной способности системы, нагрузочное тестирование, которое проводится для того, чтобы оценить поведение приложения под заданной ожидаемой нагрузкой;

– тестирование совместимости. Это метод, основной целью которого является обеспечение качественной работы конечного продукта с другим программным обеспечением;–ручное тестирование. Это тестирование системы производится лично без использования каких-либо запрограммированных процедур.

Для положительной заключительной оценки о готовности программного обеспечения система должна в обязательном порядке пройти все тесты.

Для проведения тестирования разработанного в рамках курсового проекта программного средства был выполнен ряд тестирующих заданий, которые позволяют выявить корректность работы графического интерфейса и приложения в целом.

Список тестирующих заданий представлен в таблице 4.1.

Таблица 4.1 – Результаты выполнения тестовых заданий.

|  |  |  |  |
| --- | --- | --- | --- |
| Название операции | Событие | Ожидаемый результат | Фактический результат |
| 1 | 2 | 3 | 4 |
| Ввод ссылки на файл | Загрузка файла по ссылке | Загруженный файл | Успешная загрузка файла |
| Открытие файла после успешной загрузки | Открытие файла | Открытие проводника с указание на файл | Открытый проводник с указание на файл |
| Отмена загрузки файла большого размера | Отмена загрузки | Остановка загрузки файла | Остановленная загрузка. Сохранен на диске недоконченный файл |
| Загрузка файла при отсутствии интернета | Загрузка файла по ссылке | Остановка загрузки файла. Файл отсутствует в казанной папке | Прогресс загрузки на нуле. Загрузка остановлена. Файл не сохранен на диске. |

Все тесты производились методом черного ящика. После проведения тестовых заданий над программным средством была выявлена некорректная работа системы. Все недостатки были исправлены, система имеет стабильное состояние.

1. Руководство пользователя

В поле URL, которое изображено на рисунке 10.3, необходимо указать ссылку на файл, которое на конце имеет расширение. Затем нажать на кнопку загрузки, которая изображена на рисунке 10.4.

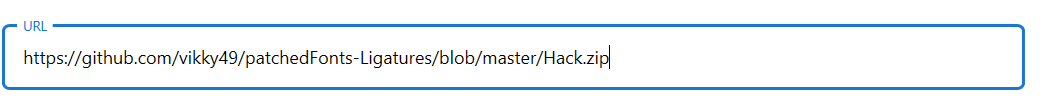


Рисунок 10.3 – Поле ввода URL на файл.



Рисунок 10.4 – Кнопка загрузки.

После нажатия на кнопку загрузки откроется окно сохранения файла, необходимо выбрать место сохранения и указать имя файла, окно указано на рисунке 10.5.

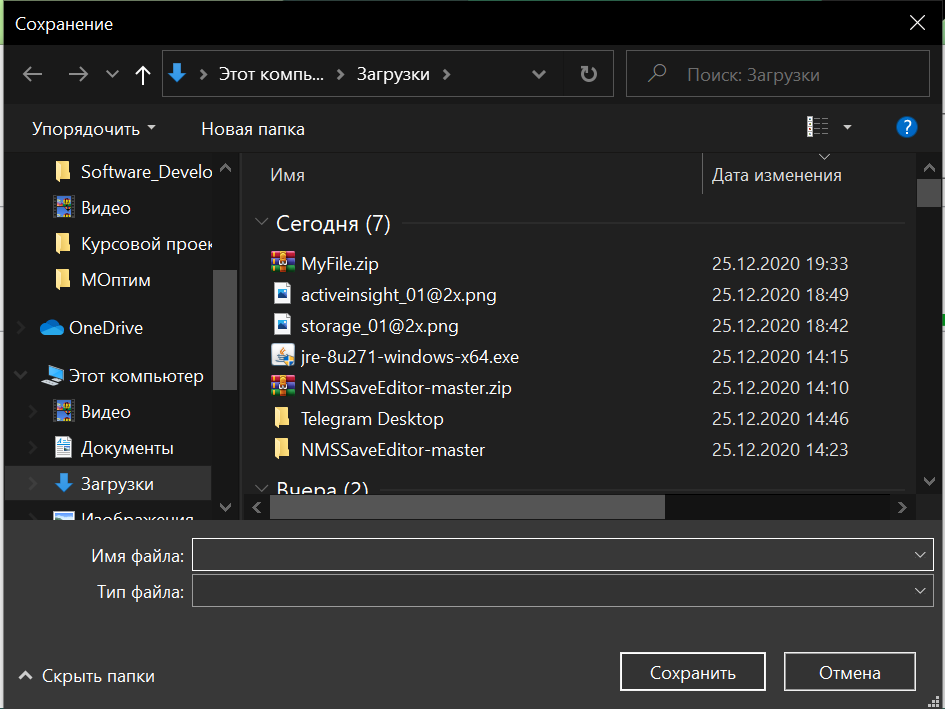


Рисунок 10.5 – Окно сохранения файла.

После успешной загрузки появится похожий блок, как на рисунке 10.6.

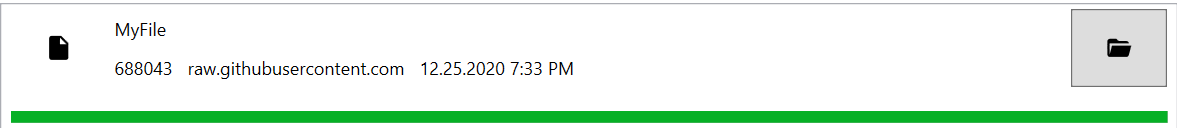


Рисунок 10.6 – Блок загрузки файла.

Чтобы перейти к файлы необходимо нажать кнопку открыть в проводнике, которая указана на рисунке 10.7.



Рисунок 10.7 – Кнопка открытия файла.

После этого откроется окно проводника Windows с указателем на данный файл, проводник указан на рисунке 10.8.

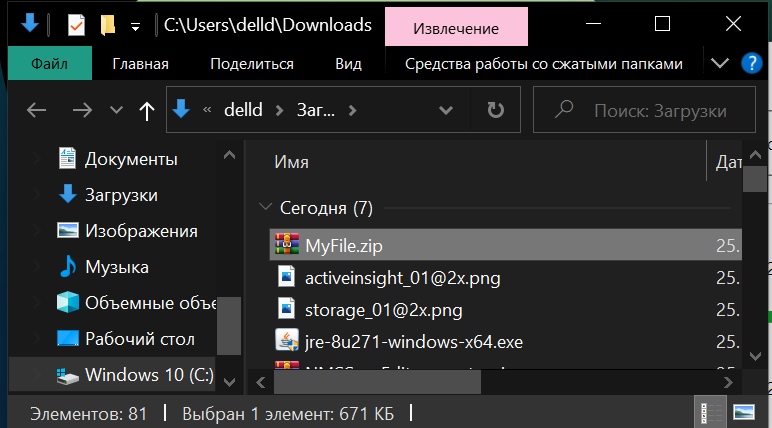


Рисунок 10.8 – Проводник с файлом.

После закрытия окно список текущих загрузок очищается.

ЗАКЛЮЧЕНИЕ

В ходе курсового проекта было разработано программное средство позволяющая выполнять многопоточные загрузки файлов. Также были учтены особенности операционных систем и их архитектурные нюансы. Поэтому программное средство было разработано на платформе .NetCore, которое позволяет использовать стандарт .NetCore для создания программ, которые будут работать в операционных системах: Windows, Linux и использовать любую операционную систему для компиляции программы.

Во время разработки использовалась среда разработки Microsoft Visual Studio 2017

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Компьютерные сети. Учебный курс: Официальное пособие Microsoft для самостоятельной подготовки: Пер. с англ. – 2-е издание, испр. и доп. – М.: Издательско-торговый дом «Русская редакция», 1999. – 568с.: ил.
2. Олифер, В.Г., Олифер, Н.А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. Стандарт третьего поколения.-Спб.: Питер, 2012.-944с.
3. Бройдо,В.Л., Ильина,О.П.Вычислительные системы,сети и телекоммуникации: учебник для вузов.4-е изд.-СПб.:Питер,2011.-560с.
4. Проскурин,В.Г.Защита программ и данных.2-е изд.-М.:Академия,2012.-208с.
5. Пролетарский, А.В. Баскаков И.В., Черков,Д.Н.Беспроводные сети Wi-Fi:учебное пособие .-М.:БИНОМ ЛЗ,ИНТУИТ,2012.-215с.
6. Таненбаум, Э. Компьютерные сети. – Спб.: Питер,2003.-992с.
7. Щерба,В.К., Киреичев,В.М., Самойленко,С.И. Стандарты по локальным вычис-лительным сетям: Справочник/ Под ред. С.И. Самойленко. – М.: Радио и связь, 1990.
8. Блэк,Ю. Сети ЭВМ: протоколы, стандарты, интерфейсы/ Пер. с англ. – М.:Мир,1990.
9. Куини Л., Рассел Р. Fast Ethernet/ - BHV – Киев, 1998.
10. Семенов,А.Б. Волоконная оптика в локальных и корпоративных сетях связи. АйТи. – М.: Компьютер-пресса,1998.

Приложение А

**Текст ПС**

namespace CN\_and\_SYS\_Downloader.Services

{

public interface IContainer

{

IMemoryManager GetMemoryManager();

IRequestService GetRequestService();

IApiService GetApiService();

IConfig GetConfig();

}

}

namespace CN\_and\_SYS\_Downloader.Services

{

public interface IConfig

{

string SyncfusionLicenseKey { get; set; }

byte ActiveMonthsCount { get; set; }

byte AttemptsCount { get; set; }

int ConnectionTimeoutMillis { get; set; }

int DelayBetweenAttemptsMillis { get; set; }

int ReachabilityRequestTimeoutMillis { get; set; }

int ReachabilityWatingForInternetAfterFailureMillis { get; set; }

int ReachabilityWatingForInternetBeforeCommunicationMillis { get; set; }

}

}

namespace CN\_and\_SYS\_Downloader.Services.Request

{

public interface IRequestService

{

Task<HttpResponseMessage> RequestAsync(Func<HttpRequestMessage> requestMessageFactory, int timeoutMillis, CancellationToken token);

}

}

namespace CN\_and\_SYS\_Downloader.Services.Network

{

public class ProgressHandler

{

private readonly ProgressChangedDelegate progressChangedDelegate;

public delegate void ProgressChangedDelegate(long? totalStepsCount, long passedStepsCount);

public ProgressHandler(ProgressChangedDelegate progressChangedDelegate)

{

this.progressChangedDelegate = progressChangedDelegate ?? throw new ArgumentNullException(nameof(progressChangedDelegate));

}

public void ProgressChanged(long? totalStepsCount, long passedStepsCount)

{

progressChangedDelegate.Invoke(totalStepsCount, passedStepsCount);

}

}

}

namespace CN\_and\_SYS\_Downloader.Services.MemoryManagement

{

public interface IMemoryManager

{

void AddFile(string path, string hash);

void DeleteFile(string hash);

long GetAllowedMaxSize();

string GetFilePath(string hash);

bool IsFileExist(string hash, bool lockFile);

void SaveState();

void SetAllowedMaxSize(long totalSize);

void UnlockFile(string hash);

}

}

namespace CN\_and\_SYS\_Downloader.Services.Api

{

public interface IApiService

{

Task GetFile(ILoadInfo loadInfo, CancellationToken token, ProgressHandler progressHandler = null);

delegate void FileMethodContainer(object sender, IApiLoadEventArg eventArg);

event FileMethodContainer DownloadCompleted;

event FileMethodContainer StatusChanged;

}

}

namespace CN\_and\_SYS\_Downloader.Presentation.ViewModels

{

public interface IMainWindowViewModel : IBaseViewModel

{

IObservableRangeCollection<ILoadItemViewModel> LoadItemCollection { get; set; }

IAsyncDelegateCommand LoadFileCommand { get; }

bool IsCommandActive { get; set; }

IObservableObject<string> LoadFromUrl { get; set; }

}

}

namespace CN\_and\_SYS\_Downloader.Presentation.ViewModels

{

public interface ILoadItemViewModel : ILoadItem, ILoadInfo

{

IAsyncDelegateCommand OpenFileInFolderCommand { get; set; }

IAsyncDelegateCommand CancelLoadCommand { get; set; }

IObservableObject<double> Progress { get; set; }

CancellationTokenSource CancellationToken { get; set; }

}

}

namespace CN\_and\_SYS\_Downloader

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

var assembly = System.Reflection.IntrospectionExtensions.GetTypeInfo(typeof(App)).Assembly;

foreach (var res in assembly.GetManifestResourceNames())

{

System.Diagnostics.Debug.WriteLine("found resource: " + res);

}

IContainer container = new Container();

var syncfusionLicenseKey = container.GetConfig().SyncfusionLicenseKey;

Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense(syncfusionLicenseKey);

InitializeComponent();

IMainWindowViewModel viewModel = new MainWindowViewModel(container);

DataContext = viewModel;

}

}

}

public MainWindow()

{

var assembly = System.Reflection.IntrospectionExtensions.GetTypeInfo(typeof(App)).Assembly;

foreach (var res in assembly.GetManifestResourceNames())

{

System.Diagnostics.Debug.WriteLine("found resource: " + res);

}

IContainer container = new Container();

var syncfusionLicenseKey = container.GetConfig().SyncfusionLicenseKey;

Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense(syncfusionLicenseKey);

InitializeComponent();

IMainWindowViewModel viewModel = new MainWindowViewModel(container);

DataContext = viewModel;

}

}

}

Приложение Б графическая часть проекта