СОДЕРЖАНИЕ

[Введение 4](#_Toc1310110)

[1 Экономическая характеристика объекта 6](#_Toc1310111)

[2 Постановка задачи 8](#_Toc1310112)

[2.1 Экономическое содержание задачи 8](#_Toc1310113)

[2.2 Входная информация 8](#_Toc1310114)

[2.3 Выходная информация 9](#_Toc1310115)

[3 Технические и инструментальные средства обеспечения задачи 10](#_Toc1310116)

[3.1 Обоснование выбора языка программирования 10](#_Toc1310117)

[3.2 Технические требования к компьютеру для решения задачи 13](#_Toc1310118)

[4 Информационное обеспечение задачи 14](#_Toc1310119)

[4.1 Классификаторы, используемые для решения задачи 14](#_Toc1310120)

[5 Программное обеспечение задачи 15](#_Toc1310121)

[5.1 Описание процедур, составляющих программу 15](#_Toc1310122)

[5.2 Описание блок-схемы 18](#_Toc1310123)

[5.3 Описание интерфейса программы 18](#_Toc1310124)

[Заключение 24](#_Toc1310125)

[Список использованных источников 25](#_Toc1310126)

[Приложение А 26](#_Toc1310127)

[Приложение Б 27](#_Toc1310128)

# Введение

Темой курсового проекта является «Разработка облачного хранилища». XXI век – это период информационных технологий. То, что еще совсем недавно казалось новым и неизведанным, сегодня уже неактуально. Мы используем различные гаджеты повсеместно, а программное обеспечение к ним обновляется практически ежедневно. Программирование это та самая важная составляющая любого гаджета, без которого выше устройство является простой коробкой с интегральной схемой.

На сегодняшний день общими главными проблемами для всех облачных приложения является обязательным доступ к сети интернет и ограниченное место на диске, но это лишь общие проблемы облачных приложений, если рассматривать каждое приложение в отдельности то возникает большое количество минусов, начиная с неудобного интерфейса и заканчивая безопасностью личных файлов пользователей.

Целью данного проекта является написание безопасного облачного приложения, не требующего доступ к сети интернет для передачи файлов между пользователями при этом способного отправлять файлы больших объемов за максимально короткое время.

Задачами являются:

* Разграничение прав доступа пользователей;
* Безопасный (зашифрованный) обмен файлами;
* Способность передачи фалов, как по локальной сети, так и по сети интернет.

Актуальность данного проекта заключается в востребованности безопасного, бесплатного, удобного и простого приложения выполняющего обмен файлами, как по локальной сети, так и по сети интернет программный продукт может применяться в бюджетных муниципальных и коммерческих предприятиях как для простого обмена фалами, так и ведения электронного документооборота.

# 1 Экономическая характеристика объекта

Волгодонский техникум – это образовательное учреждение, имеющее славную 20-летнюю историю, которая официально началась с 1990 года как профессиональное училище № 105. Быстро развиваясь, в 1996 году училище было преобразовано в лицей, а в 2005 году ему присвоено имя первого директора Владимира Васильевича Самарского, в 2010 году образовательное учреждение было преобразовано и получило статус техникума [7].

Сегодня техникум – это современное и престижное образовательное учреждение системы среднего профессионального образования Ростовской области, прочно занявшее одно из ведущих мест среди средних профессиональных образовательных учреждений региона, вошедшее в национальный реестр «Ведущие образовательные учреждения России».

Техникум, сохраняя и совершенствуя лучшие традиции отечественной системы подготовки высококвалифицированных специалистов, в процессе подготовки профессионального, компетентностного работника активно применяет инновационные и информационные технологии в обучении, основываясь на компетентностном подходе и практико-ориентированном обучении. Подготовка в техникуме ведётся по современным, востребованным на рынке труда специальностям и профессиям.

Сегодня техникум обладает развитой образовательной инфраструктурой: здание техникума удобно расположено в центральной части города, красивый и просторный трёхэтажный учебный корпус с учебными кабинетами, лабораториями и цехами, оснащенными современным учебным и производственным оборудованием, объединёнными локальной сетью, сетью Интернет и Интранет, позволяют эффективно решать учебные и воспитательные задачи. В образовательном процессе широко используются программные и инструментальные средства, с помощью которых техникум успешно реализует профессиональную подготовку выпускников и является одним из лучших в Южном Федеральном округе. В образовательном учреждении создана мощная компьютерная база, включающая 12 лабораторий с современными компьютерными и мультимедийными программами,  60 терминалов, имеющих доступ к глобальной сети Internet, в техникуме введена в действие система электронного обучения и система электронных коммуникаций.

Приложение «Cloud» решает ряд важнейших экономических проблем существующих в техникуме. Одной из таких проблем является ведение электронного документооборота, которого на данный момент в техникуме нет. Благодаря электронному документообороту значительно упрощается деятельность ведения бухгалтерского учёта, а именно увеличивается скорость обмена документами, между пользователями, имеющими к ним доступ. Также благодаря резервированию риск потери документов практически исчезает, а благодаря закрытой локальной сети взлом сервера за пределами техникума не является возможным [6].

Ещё одной не маловажной проблемой является отсутствие толкового обмена файлами между преподавателем и студентами. В техникуме лишь в нескольких лабораториях есть обмен файлами, но он настроен посредством администрирования операционной системы, что является не удобным для многих преподавателей т.к., после переустановки операционной системы в лаборатории компьютеры требуют длительной настрой для обмена файлами. Приложение «Cloud» решает эту проблему простой установкой, быстрой и безопасной авторизацией, а также удобным и интуитивно понятным интерфейсом.

# 2 Постановка задачи

# 2.1 Экономическое содержание задачи

Проект разработан для «Волгодонского техникума информационных технологий, бизнеса и дизайна имени В. В. Самарского» чтобы упростить и оптимизировать обмен файлами между преподавателем студентами. ВТИТБиД – это современное и престижное образовательное учреждение системы среднего профессионального образования Ростовской области, прочно занявшее одно из ведущих мест среди средних профессиональных образовательных учреждений региона, вошедшее в национальный реестр «Ведущие образовательные учреждения России» [9].

На данный момент программа «Cloud» тестируется в Волгодонском техникуме информационных технологий, бизнеса и дизайна имени В. В. Самарского в нескольких лабораториях техникума программа выполняет обмен файлами, благодаря чему преподаватель отправляет задание на сервер студенты в свою очередь скачивают это задание и после выполнения отправляют обратно на сервер по локальной сети с использованием HTTP протокола [10]. Данный способ решает проблему с передачей файлов на USB накопители, и потерю этих файлов в случаи неправильного извлечения, помимо этого риск подхватить вирус также уменьшается, нежели при передачи файлов с помощью флешки.

# 2.2 Входная информация

Входной информацией в данном проекте являются файлы загружаемые пользователями на сервер, такие как размер файла, имя файла, имя пользователя который загружает файл и дата загрузки. Помимо этого есть запросы отправляемые пользователем на сервер для получения той или иной информации, например запрос на получения пароля подтверждения или вывода удачного или неудачного результата выполнения операции.

# 2.3 Выходная информация

Выходной информацией в данном проекте являются файлы скачиваемые пользователями, а также ответы на запросы удаления, получения списка файлов с сервера, запрос на получения пароля подтверждения и вывода удачного или неудачного результата выполнения операции.

# 3 Технические и инструментальные средства обеспечения задачи

# 3.1 Обоснование выбора языка программирования

Для написания приложения был выбран объектно-ориентированный язык программирования C#. Одним из его преимуществ является кроссплатформенность на этом языке можно писать практически все, что угодно игры, мобильные приложения, программирование контроллеров, WEB сайты, приложения для Desktop не зависимо от операционной системы. Также большим плюсом является огромный стек технологий, таких как WindowsForms, Windows Presentation Foundation, Windows Communication Foundation, Xamarin, ADO.NET, Entity Framework, ASP.NET MVC, ASP.NET Core и многие другие, большее количество из которых являются кроссплатформенными, то есть используются в других языках программирования. Для решения данной задачи преимущественно использовались технологии Windows Presentation Foundation, Windows Communication Foundation, Language Integrated Query и Entity Framework [5].

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998 - 2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA - 334 и ISO/IEC23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [5].

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ и некоторых других языков, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR [1]. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др [4].

Windows Communication Foundation (WCF) — программный фреймворк, используемый для обмена данными между приложениями, входящий в состав .NET Framework. До своего выпуска в декабре 2006 года в составе .NET Framework 3.0, WCF был известен под кодовым именем Indigo.

WCF делает возможным построение безопасных и надёжных транзакционных систем через упрощённую унифицированную программную модель межплатформенного взаимодействия. Комбинируя функциональность существующих технологий .NET по разработке распределённых приложений (ASP.NET XML Web Services — ASMX, WSE 3.0, .NET Remoting, .NET Enterprise Services и System.Messaging), WCF предоставляет единую инфраструктуру разработки, при умелом применении повышающую производительность и снижающую затраты на создание безопасных, надёжных и транзакционных Web-служб нового поколения. Заложенные в неё принципы интероперабельности позволяют организовать работу с другими платформами, для чего используются технологии взаимодействия платформ, например WSIT, разрабатываемые на базе открытого исходного кода [2].

Windows Presentation Foundation (WPF) (произносится — «даб-пи-эф») — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в браузере приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Window Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX [3].

Безопасность это важная составляющая любого программного продукта в приложении «Cloud» для обеспечения безопасности применяется WCF сервис с режимом безопасности TransportWithMessageCredential транспортный уровень используется для защиты передачи сообщений. При этом каждое сообщение включает расширенные учетные данные, необходимые другим службам. Этот способ сочетает повышение производительности механизма безопасности транспорта с наличием различных учетных данных механизма безопасности сообщений. Его можно использовать со следующими привязками: BasicHttpBinding, WSFederationHttpBinding, NetPeerTcpBinding и WSHttpBinding [8].

# 3.2 Технические требования к компьютеру для решения задачи

Back end:

* Процессор - с частотой 1 гигагерц (ГГц) или быстрее или система на кристалле SoC;
* Операционная система - Windows Server 2003 и выше;
* Оперативная память - 1 гигабайт (ГБ) для 32-разрядных систем или 2 ГБ для 64-разрядных систем;
* Пространство на жестком диске - 16 ГБ для 32-разрядных систем или 32 ГБ для 64-разрядных систем;
* Программная платформа - .NET Framework 4.0.

Front end:

* Процессор - с частотой 1 гигагерц (ГГц) или быстрее;
* Операционная система - Windows XP и выше;
* Оперативная память - 2 гигабайт (ГБ) для 32-разрядных систем или 4 ГБ для 64-разрядных систем;
* Пространство на жестком диске - 32 ГБ для 32-разрядных систем или 64 ГБ для 64-разрядных систем;
* Программная платформа - .NET Framework 4.0;
* Видеоадаптер - DirectX 9 или более поздней версии с драйвером WDDM 1.0 [10].

# 4 Информационное обеспечение задачи

# 4.1 Классификаторы, используемые для решения задачи

Классификатор представляет собой директорию папок, где хранятся файлы пользователей после отправки их на сервер. Папки в свою очередь имеют определённые название в зависимости от электронной почты пользователя, которая была указана при входе в программу. Если пользователь входит с логином (электронной почтой) папка которого уже существует на сервере, то пользователь будет иметь доступ к ресурсам в этой папки [4].

Файлы клиентской составляющей приложения представляют собой папку «Download», которая расположена в одной директории с «exe» файлом в ней хранятся файлы скаченные пользователем с сервера. Также в одной директории с «exe» файлом хранится текстовый документ «Code.txt» хранящий логин (электронную почту) пользователя введённую при авторизации это необходимо для решения проблемы с повторной авторизацией.

# 5 Программное обеспечение задачи

# 5.1 Описание процедур, составляющих программу

Back end (Сервер):

Отправка потока байт клиенту:

public RemoteFileInfo DownloadFile(DownloadRequest request)

{

str = request.mail;

string filePath = System.IO.Path.Combine("Upload/" + request.mail, request.FileName);

System.IO.FileInfo fileInfo = new System.IO.FileInfo(filePath);

if (!fileInfo.Exists) throw new System.IO.FileNotFoundException("Файл не найден", request.FileName);

System.IO.FileStream stream = new System.IO.FileStream(filePath, System.IO.FileMode.Open, System.IO.FileAccess.Read);

RemoteFileInfo result = new RemoteFileInfo();

result.FileName = request.FileName;

result.Length = fileInfo.Length;

result.FileByteStream = stream;

return result;

}

Принятие потока байт на сервер:

public void UploadFile(RemoteFileInfo request)

{

if (!System.IO.Directory.Exists("Upload/" + request.mail)) System.IO.Directory.CreateDirectory("Upload/" + request.mail);

string filePath = System.IO.Path.Combine("Upload/" + request.mail, request.FileName);

if (System.IO.File.Exists(filePath)) System.IO.File.Delete(filePath);

int chunkSize = 2048;

byte[] buffer = new byte[chunkSize];

using (System.IO.FileStream writeStream = new System.IO.FileStream(filePath, System.IO.FileMode.CreateNew, System.IO.FileAccess.Write))

{

do{

int bytesRead = request.FileByteStream.Read(buffer, 0, chunkSize);

if (bytesRead == 0) break;

writeStream.Write(buffer, 0, bytesRead);

} while (true);

writeStream.Close();

}

}

Front end (Клиент):

Отправка потока байт на сервер:

public void dovnload(string s){

if (!System.IO.Directory.Exists("Download")) System.IO.Directory.CreateDirectory("Download");

System.IO.FileInfo fileInfo = new System.IO.FileInfo(s);

using (System.IO.FileStream stream = new System.IO.FileStream(s, System.IO.FileMode.Open, System.IO.FileAccess.Read))

{

using (StreamWithProgress uploadStreamWithProgress = new StreamWithProgress(stream))

{

облако.ServiceReference1.Service1Client client = new облако.ServiceReference1.Service1Client();

client.UploadFile(fileInfo.Name, fileInfo.Length, File.ReadAllText(currDir), uploadStreamWithProgress);

service client

client.Close();

}

}

Ok o = new Ok("Отправлено");

o.Show();

}

Принятие потока байт с сервера:

private void dwn(){

string filePath = System.IO.Path.Combine("Download", FolderView.SelectedItem.ToString());

if (System.IO.File.Exists(filePath)) System.IO.File.Delete(filePath);

System.IO.Stream inputStream;

string fileName = FolderView.SelectedItem.ToString();

string s = File.ReadAllText(currDir);

long length = client.DownloadFile(ref fileName, ref s, out inputStream);

using (System.IO.FileStream writeStream = new System.IO.FileStream(filePath, System.IO.FileMode.CreateNew, System.IO.FileAccess.Write)){

int chunkSize = 2048;

byte[] buffer = new byte[chunkSize];

do{

int bytesRead = inputStream.Read(buffer, 0, chunkSize);

if (bytesRead == 0) break;

writeStream.Write(buffer, 0, bytesRead);

} while (true);

writeStream.Close();

}

}

# 5.2 Описание блок-схемы

Шаг 1. Запуск программы.

Шаг 2. Вопрос: Загрузка или скачивание файла? Если загрузка переход к ШАГУ 3, если скачивание переход к ШАГУ 6.

Шаг 3. Выбор загружаемого файла и преобразование его в массив байт.

Шаг 4. Шифрование массива байт и отправка на сервер.

Шаг 5. Дешифрование массива байт и создание фала на сервере.

Шаг 6. Отправка запроса на получение списка фалов на сервере.

Шаг 7. Выбор файла для скачивания из списка.

Шаг 8. Отправка запроса на скачивание файла.

Шаг 9. Поиск файла на сервере по запросу пользователя и запись его массив байт.

Шаг 10. Шифрование массива байт и отправка клиенту.

Шаг 11. Дешифрование массива байт и создание фала.

Шаг 12. Конец.

Блок-схема приведена в приложении А.

# 5.3 Описание интерфейса программы

Cloud – это нативно понятное пользователю приложение главное окно которого представляющее собой виджет как показано на рисунке 1. Для загрузки файлов на сервер достаточно перенести выбранные файлы на главное окно программы или воспользоваться контекстным меню для вызова диалогового окна загрузки файлов.

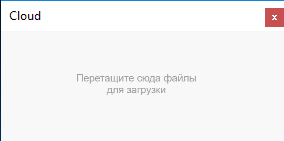


Рисунок 1 – Главное окно приложения Cloud

Но для того, чтобы выше упомянутое окно появилось нужно пройти простую авторизацию, как показано на рисунке 2, где необходимо указать свой адрес электронной почты, на который придет код подтверждения рисунок 3, после чего необходимо ввести код в окно на рисунке 4 такая авторизация необходима для разграничения прав пользователей, таким образом, один пользователь не сможет зайти в хранилище другого.

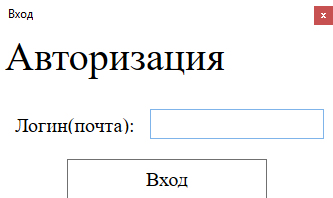


Рисунок 2 – Окно авторизации

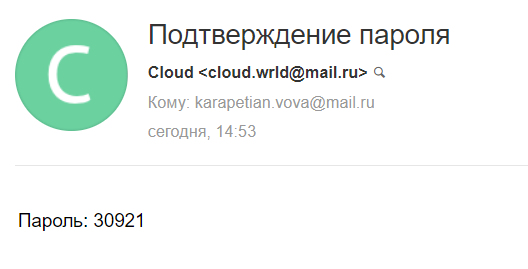


Рисунок 3 – Письмо с паролем для подтверждения электронной почты

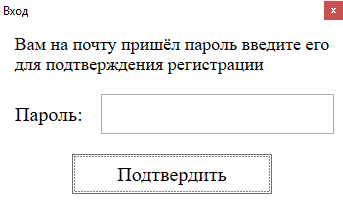


Рисунок 4 – Окно подтверждения электронной почты

Чтобы скачать ранее загруженные файлы достаточно перейти в окно администрирования фалов выбрав в контекстном меню пункт «Cloud» как показано на рисунке 5.

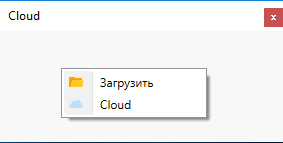


Рисунок 5 – Контекстное меню приложения Cloud

В появившемся окне администрирования рисунок 6 для загрузки нужного файла достаточно кликнуть по нему 2 раза.

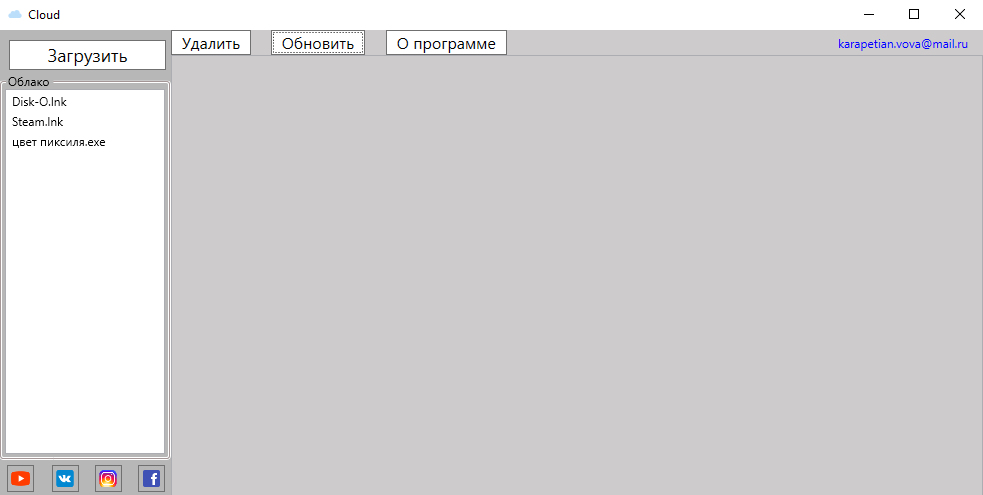


Рисунок 6 – Окно администрирования файлов приложения Cloud

После чего в окне появится процесс скачивания выбранного файла рисунок 7, по окончанию загрузки появится уведомление о результате загрузки как показано на рисунке 8.

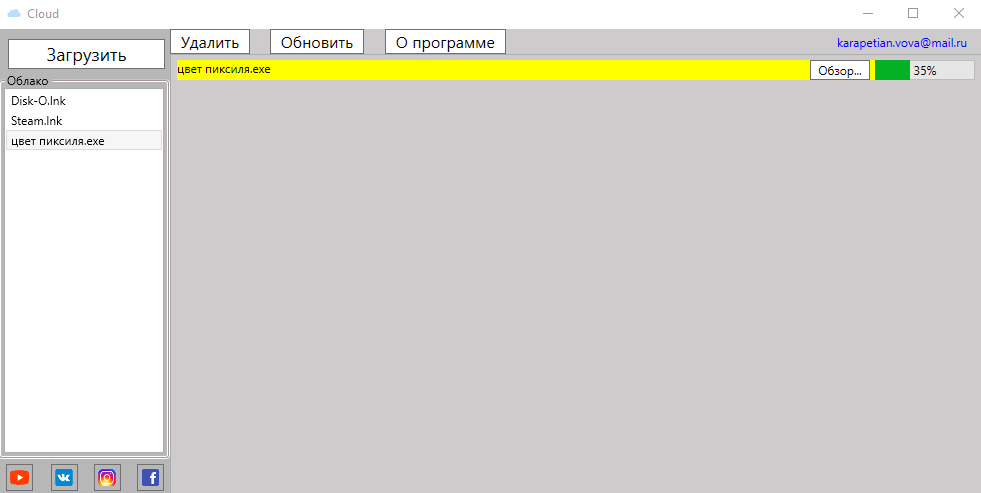


Рисунок 7 – Процесс скачивания выбранного файла

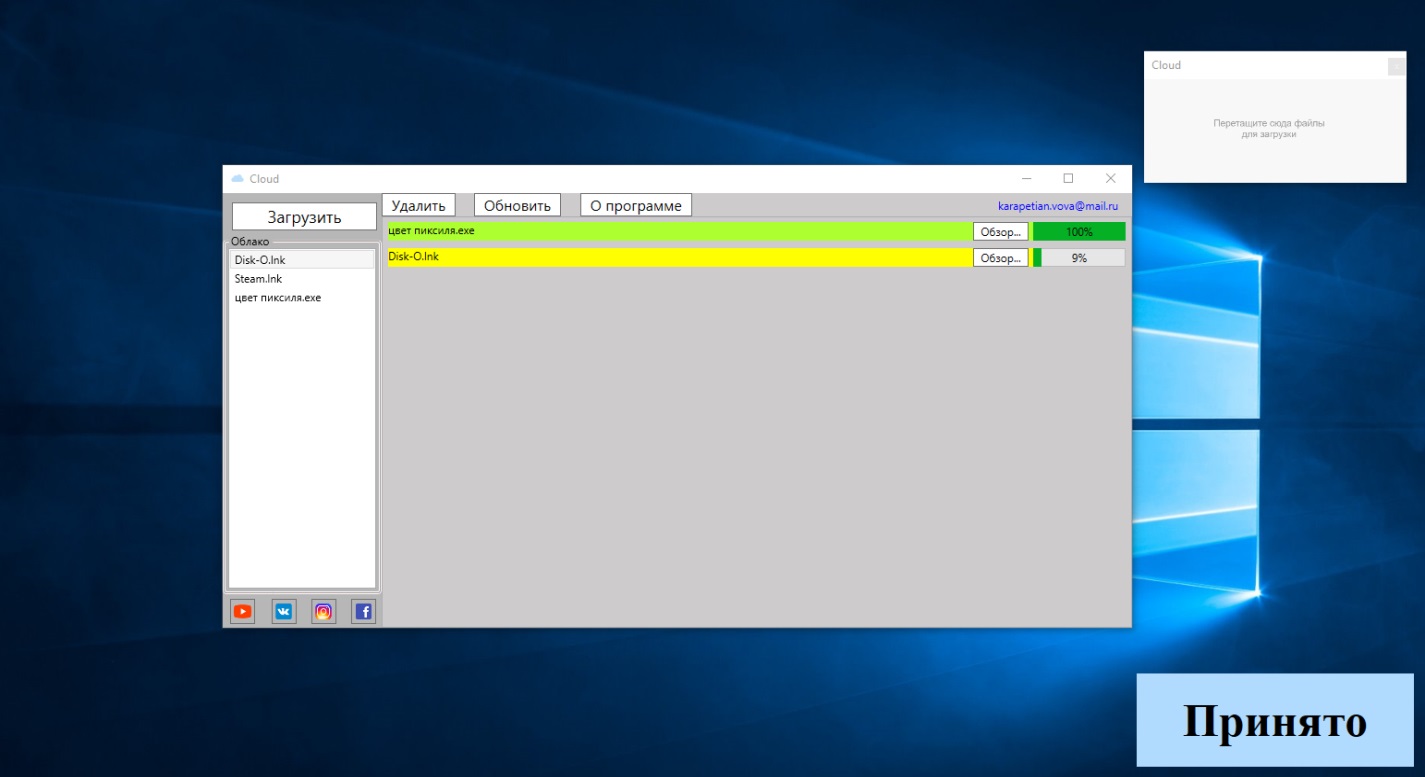


Рисунок 8 – Уведомление о результате загрузки

Для удаления файла с сервера нужно выбрать его в списке и нажать кнопку удалить по окончанию процесса файл удалится из списка и появится уведомление об окончании операции удаления рисунок 9.

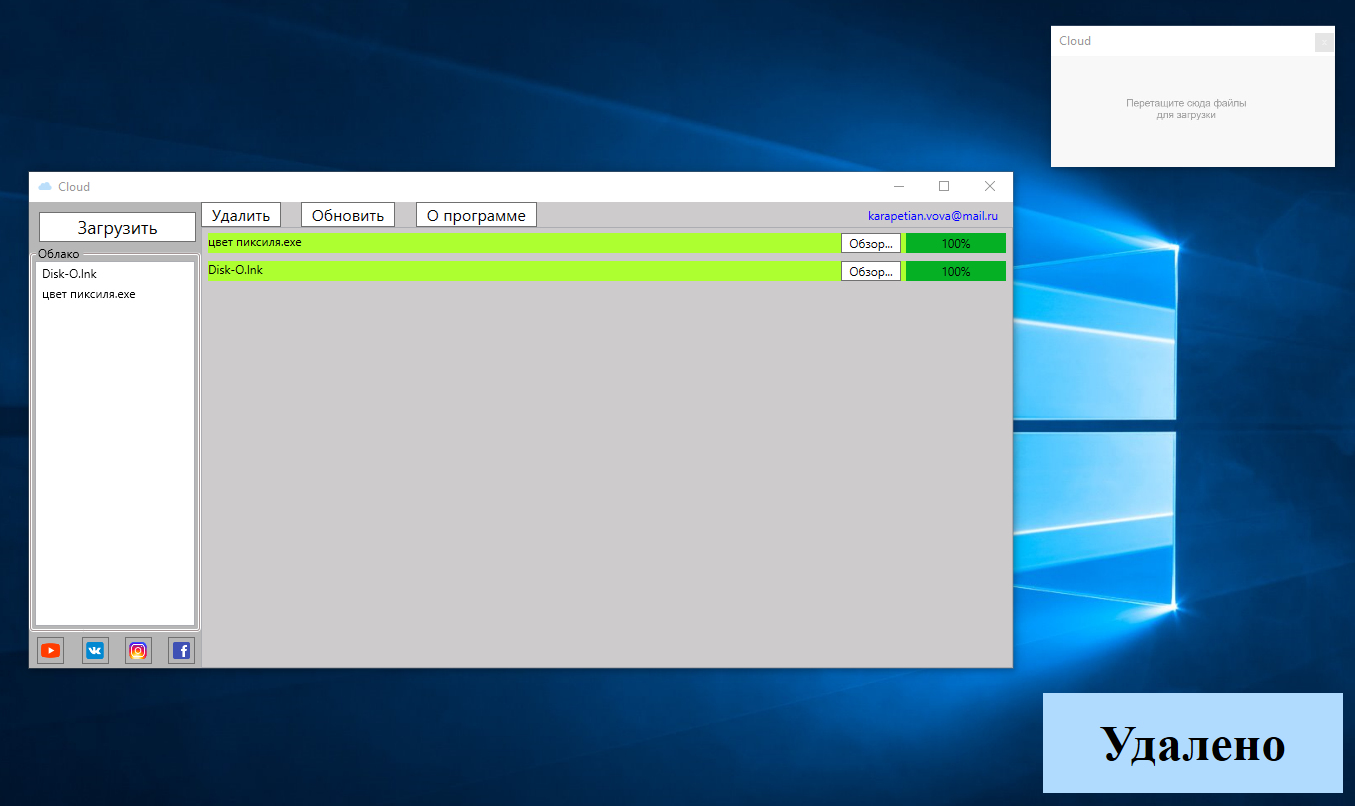
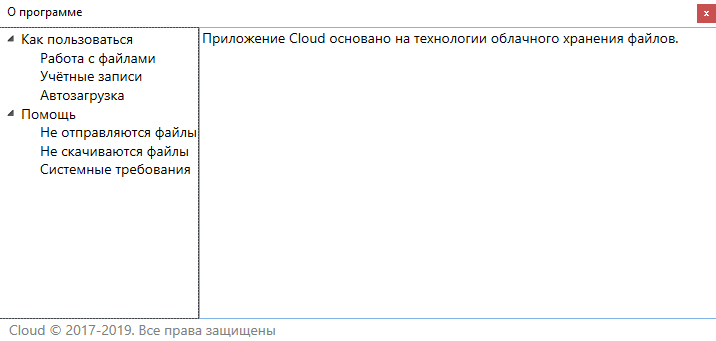


Рисунок 9 – Окончание операции удаления файла

В случае если по окончанию той или иной операции, пользовательская информация не обновиться есть кнопка «Обновить», которая принудительно обновит информацию. Для того чтобы ознакомится с программой можно кликнуть на кнопку «О программе» после чего появится окно где можно ознакомиться со всеми возможностями программы рисунок 10.

Рисунок 10 – Окно «О программе»

# Заключение

В ходе выполнения курсового проекта были выполнены следующие задачи:

* Разграничение прав доступа пользователей;
* Безопасный (зашифрованный) обмен файлами;
* Способность передачи фалов, как по локальной сети, так и по сети интернет.

Приложение Cloud значительно облегчает обмен файлами и ведение электронного документооборота внутри организации и за её приделами. Объем памяти не имеет ограничений, а файлы на сервере автоматически могут проверяться антивирусом, установленном на персональном компьютере. Пользователю не обязательно обладать углубленными знаниями в компьютерной технике, чтобы пользоваться этой программой.

Программа, написана на языке программирования C#, не содержит сложного интерфейса, но при этом имеет весь необходимый функционал. Содержит в себе такие паттерны проектирования как Windows Presentation Foundation и Windows Communication Foundation.

В процессе работы над курсовым проектом были закреплены практические навыки работы в среде программирования Visual Studio.

# Список использованных источников

Список использованной литературы

1. М.Е. Фленов, Библия С#. – 3 - е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2016.г - 544 с.: ил.
2. Албахари, Джозеф, Албахари, Бен, С# 6.0. Справочник. Полное описание языка, 6-е изд.: Пер. с англ. - М.: ООО "И.Д. Вильямс", 2016. – 1040 с.: ил.
3. Эндрю Стиллмен, Дженнифер Грин, Изучаем C#. 3-е изд. – СПб.: Питер, 2014. – 816 с.: ил.
4. Мак-Дональд Мэтью, WPF: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 – СПб.: Вильямс, 2011. - 1020с.
5. Мак-Дональд Мэтью, Фримен Адам, Шпушта Марио, Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов, 4-е издание.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2011 – 1424с.: ил.
6. Н.П. Котерова Экономика организации: учебник для студ. Учреждений сред. проф. образования / Н.П. Котерва.- М.: Издательский центр «Академия», 2014г.

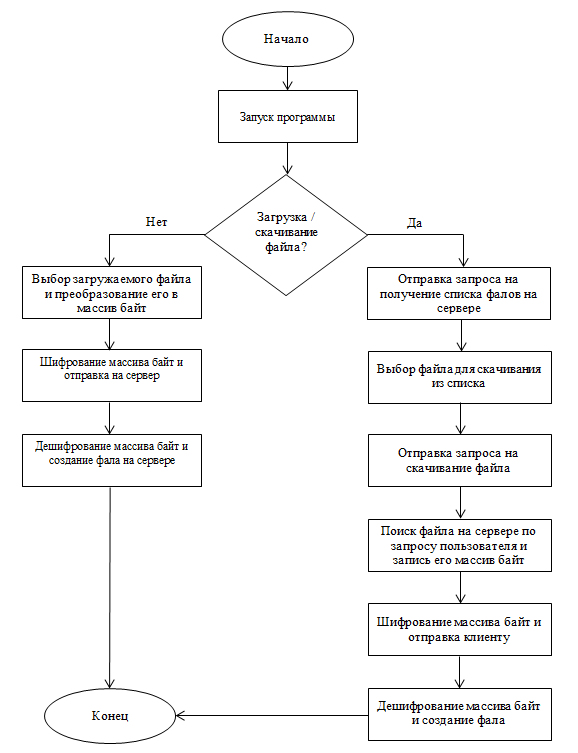
Интернет-ресурсы

1. Энциклопедия Экономиста [Электронный ресурс] - Режим доступа: http://www.grandars.ru
2. METANIT.COM - Сайт о программировании [Электронный ресурс] - Режим доступа: https://metanit.com
3. MSDN - информационный сервис для разработчиков [Электронный ресурс] - Режим доступа: https://msdn.microsoft.com
4. Хабр - социальное СМИ об IT [Электронный ресурс] - Режим доступа: https://habr.com

# Приложение А

(обязательное)

Блок - схема описывающая алгоритм работы приложения «Cloud» при взаимодействия пользователя с программой.

Рисунок А.1 – Блок-схема программы

# Приложение Б

(обязательное)

Листинг программы

Back end:

Класс – интерфейс

[ServiceContract]

public interface IService1

{

[OperationContract]

void UploadFile(RemoteFileInfo request);

[OperationContract]

RemoteFileInfo DownloadFile(DownloadRequest request);

[OperationContract]

FileInfo[] Masage(string request);

[OperationContract]

void Delit(string filname, string request);

}

[MessageContract]

public class DownloadRequest

{

[MessageBodyMember]

public string FileName;

[MessageBodyMember]

public string mail;

}

[MessageContract]

public class RemoteFileInfo : IDisposable

{

[MessageHeader(MustUnderstand = true)]

public string mail;

[MessageHeader(MustUnderstand = true)]

public string FileName;

[MessageHeader(MustUnderstand = true)]

public long Length;

[MessageBodyMember(Order = 1)]

public System.IO.Stream FileByteStream;

public void Dispose()

{

if (FileByteStream != null)

{

FileByteStream.Close();

FileByteStream = null;

}

}

}

Реализация класса – интерфейса

public class Service1 : IService1

{

string str;

public void Delit(string filname, string request)

{

string currDirк = Environment.CurrentDirectory.ToString() + @"\Upload\" + request + @"\" + filname;

File.Delete(currDirк);

}

public RemoteFileInfo DownloadFile(DownloadRequest request)

{

str = request.mail;

string filePath = System.IO.Path.Combine("Upload/" + request.mail, request.FileName);

System.IO.FileInfo fileInfo = new System.IO.FileInfo(filePath);

if (!fileInfo.Exists) throw new System.IO.FileNotFoundException("Файл не найден", request.FileName);

System.IO.FileStream stream = new System.IO.FileStream(filePath, System.IO.FileMode.Open, System.IO.FileAccess.Read);

RemoteFileInfo result = new RemoteFileInfo();

result.FileName = request.FileName;

result.Length = fileInfo.Length;

result.FileByteStream = stream;

return result;

}

public FileInfo[] Masage(string request)

{

string currDir = Environment.CurrentDirectory.ToString() + @"/Upload/" + request;

DirectoryInfo dinfo = new DirectoryInfo(currDir);

return dinfo.GetFiles();

}

public void UploadFile(RemoteFileInfo request)

{

if (!System.IO.Directory.Exists("Upload/" + request.mail)) System.IO.Directory.CreateDirectory("Upload/" + request.mail);

string filePath = System.IO.Path.Combine("Upload/" + request.mail, request.FileName);

if (System.IO.File.Exists(filePath)) System.IO.File.Delete(filePath);

int chunkSize = 2048;

byte[] buffer = new byte[chunkSize];

using (System.IO.FileStream writeStream = new System.IO.FileStream(filePath, System.IO.FileMode.CreateNew, System.IO.FileAccess.Write))

{

do

{

int bytesRead = request.FileByteStream.Read(buffer, 0, chunkSize);

if (bytesRead == 0) break;

writeStream.Write(buffer, 0, bytesRead);

}

while (true);

writeStream.Close();

}

}

}

Front end:

Разметка главного окна программы

<Window x:Class="облако.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:облако"

mc:Ignorable="d"

Title="Cloud" Closed="Window\_Closed" ShowInTaskbar="False" Background="White" AllowDrop="True" Height="150" Width="300" DragOver="Window\_DragOver" DragEnter="Window\_DragEnter" Drop="Window\_Drop" DragLeave="Window\_DragLeave" Icon="Resourse/Clouds.ico" Loaded="Window\_Loaded" WindowStyle="ToolWindow" Activated="Window\_Activated">

<Grid>

<Image Name="pictureBox1" Source="Resourse\открыть.jpg" Grid.Row="0" Grid.Column="1" Stretch="Fill" >

<Image.ContextMenu>

<ContextMenu Background="White">

<MenuItem Header="Загрузить" Click="Open\_MenuItem">

<MenuItem.Icon>

<Image Height="20" Source="Resourse\Open Folder\_48px.png"/>

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="Cloud" Click="OpenWin\_MenuItem">

<MenuItem.Icon>

<Image Height="20" Source="Resourse\Clouds\_96px\_1.png"/>

</MenuItem.Icon>

</MenuItem>

</ContextMenu>

</Image.ContextMenu>

</Image>

</Grid>

</Window>

Исходный код главного окна программы

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

[DllImport("user32.dll")]

private static extern int SetWindowLong(IntPtr window, int index, int value);

[DllImport("user32.dll")]

private static extern int GetWindowLong(IntPtr window, int index);

private const int GWL\_EXSTYLE = -20;

private const int WS\_EX\_TOOLWINDOW = 0x00000080;

public static void HideFromAltTab(IntPtr Handle)

{

SetWindowLong(Handle,

GWL\_EXSTYLE,

GetWindowLong(Handle, GWL\_EXSTYLE) | WS\_EX\_TOOLWINDOW);

}

private IntPtr Handle

{

get

{

return new WindowInteropHelper(this).Handle;

}

}

[DllImport("user32.dll")]

public static extern bool SetWindowPos(int hWnd, int hWndInsertAfter, int X, int Y, int cx, int cy, uint uFlags);

public const int HWND\_BOTTOM = 0x1;

public const uint SWP\_NOSIZE = 0x1;

public const uint SWP\_NOMOVE = 0x2;

public const uint SWP\_SHOWWINDOW = 0x40;

private void ShoveToBackground()

{

SetWindowPos((int)this.Handle, HWND\_BOTTOM, 0, 0, 0, 0, SWP\_NOMOVE | SWP\_NOSIZE | SWP\_SHOWWINDOW);

}

private void Window\_Closed(object sender, EventArgs e)

{

Process.GetCurrentProcess().Kill();

}

HeaderToImageConverter g = new HeaderToImageConverter();

private void Window\_DragOver(object sender, DragEventArgs e)

{

this.pictureBox1.Source = new BitmapImage(new Uri(@"Resourse\воронка.png", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

}

private void Window\_DragEnter(object sender, DragEventArgs e)

{

e.Effects = DragDropEffects.All;

}

private void Window\_Drop(object sender, DragEventArgs e)

{

this.pictureBox1.Source = new BitmapImage(new Uri(@"Resourse\открыть.jpg", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

string[] f = (string[])e.Data.GetData(DataFormats.FileDrop, false);

foreach (string sl in f)

g.dovnload(sl);

}

private void Window\_DragLeave(object sender, DragEventArgs e)

{

this.pictureBox1.Source = new BitmapImage(new Uri(@"Resourse\открыть.jpg", UriKind.Relative)) { CreateOptions = BitmapCreateOptions.IgnoreImageCache };

}

private void Open\_MenuItem(object sender, RoutedEventArgs e)

{

OpenFileDialog op = new OpenFileDialog();

op.Title = "Выберите файл";

op.Filter = "Все файлы (\*.\*)|\*.\*";

if (op.ShowDialog() == true)

{

string put = op.FileName;

g.dovnload(put);

}

}

private void OpenWin\_MenuItem(object sender, RoutedEventArgs e)

{

glav g = new glav();

g.Show();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

HideFromAltTab(Handle);

double screeHeight = SystemParameters.FullPrimaryScreenHeight;

double screeWidth = SystemParameters.FullPrimaryScreenWidth;

this.Top = (screeHeight - this.Height) / 10;

this.Left = (screeWidth - this.Width) / 1.01;

g.SetAutorunValue(true, Assembly.GetExecutingAssembly().Location);

}

string text { get; set; }

private void Window\_Activated(object sender, EventArgs e)

{

ShoveToBackground();

}

}

}

Разметка окна администрирования программы

<Window x:Class="облако.glav"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfTreeView"

mc:Ignorable="d"

Title="Cloud" Height="505" Width="1000" Loaded="Window\_Loaded" WindowStartupLocation="CenterScreen" Icon="Resourse/Clouds.ico">

<Grid Margin="0,0,0,0">

<Grid Background="#FFCDCBCB" Margin="172,0,0,0">

<Button FontSize="16" Content=" Удалить " Background="White" HorizontalAlignment="Left" VerticalAlignment="Top" Click="delit\_Click"/>

<Button Margin="100,0,0,0" FontSize="16" Content=" Обновить " Background="White" HorizontalAlignment="Left" VerticalAlignment="Top" Click="up\_Click"/>

<Button Margin="215,0,0,0" FontSize="16" Content=" О программе " Background="White" HorizontalAlignment="Left" VerticalAlignment="Top" Click="info\_Click"/>

<Label Name="Label1" Content="11111111@mail.ru" Foreground="Blue" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,0,10,0">

<Label.ContextMenu>

<ContextMenu Background="White">

<MenuItem Header="Выход из учётной записи" Click="Close\_MenuItem">

<MenuItem.Icon>

<Image Height="20" Source="Resourse\Close Window\_48px.png"/>

</MenuItem.Icon>

</MenuItem>

</ContextMenu>

</Label.ContextMenu>

</Label>

<ListBox Background="Transparent" x:Name="ListBox1" HorizontalContentAlignment="Stretch" Margin="0,25,0,0">

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Margin="0,2" Background="{Binding BackColor}">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="100" />

</Grid.ColumnDefinitions>

<TextBlock x:Name="TextBlock1" Text="{Binding Title}" />

<ProgressBar Name="ProgressBar1" Width="100" HorizontalAlignment="Right" Grid.Column="1" Minimum="0" Maximum="100" Value="{Binding Progress}"/>

<Button Background="White" Name="Button1" Width="60" Content="Обзор..." HorizontalAlignment="Right" Margin="0,0,5,0" Click="btnobzor\_Click"/>

<TextBlock x:Name="myTextBlock" Grid.Column="1" Text="{Binding ElementName=ProgressBar1, Path=Value, StringFormat={}{0:0}%}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Grid>

<Grid Background="#FFB8B7B7" Width="172" HorizontalAlignment="Left">

<GroupBox Foreground="Black" Header="Облако" Margin="0,42,0,36" Width="172" BorderBrush="#FFA49696">

<ListBox x:Name="FolderView" MouseDoubleClick="FolderView\_MouseDoubleClick"/>

<GroupBox.OpacityMask>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black" Offset="0"/>

<GradientStop Color="Black" Offset="1"/>

</LinearGradientBrush>

</GroupBox.OpacityMask>

</GroupBox>

<Button Background="#FFB8B7B7" Margin="-100,0,29,4" Height="27" Width="27" VerticalAlignment="Bottom">

<Image Source="Resourse\YouTube 2\_48px.png"/>

</Button>

<Button Background="#FFB8B7B7" Height="27" Margin="75,0,29,4" Width="27" VerticalAlignment="Bottom">

<Image Source="Resourse\Instagram\_48px.png"/>

</Button>

<Button Background="#FFB8B7B7" Height="27" Margin="139,0,6,4" Width="27" VerticalAlignment="Bottom">

<Image Source="Resourse\Facebook\_48px.png"/>

</Button>

<Button Background="#FFB8B7B7" Height="27" Margin="0,0,40,4" Width="27" VerticalAlignment="Bottom">

<Image Source="Resourse\Vk.com\_48px.png"/>

</Button>

</Grid>

<Button Content="Загрузить" HorizontalAlignment="Left" Height="30" Margin="10,10,0,0" VerticalAlignment="Top" Width="157" Background="White" FontSize="18" Click="download"/>

</Grid>

</Window>

Исходный код окна администрирования программы

public partial class glav : Window

{

string put;

private Thread \_thread;

HeaderToImageConverter g = new HeaderToImageConverter();

private class ListBoxDataItem

{

public SolidColorBrush BackColor { get; set; }

public string Title { get; set; }

public int Progress { get; set; }

public ListBoxDataItem(string title, Color backColor, int progress = 0)

{

this.Title = title;

this.BackColor = new SolidColorBrush(backColor);

this.Progress = progress;

}

public ListBoxDataItem(ListBoxDataItem item, int progress, Color backColor)

: this(item.Title, backColor, progress)

{

}

public ListBoxDataItem(ListBoxDataItem item, Color backColor)

: this(item.Title, backColor, item.Progress)

{

}

}

public glav()

{

InitializeComponent();

n();

}

string currDir = Environment.CurrentDirectory.ToString() + @"\Code.txt";

private void n()

{

Label1.Content = File.ReadAllText(currDir);

}

private void FillListBox()

{

ListBox1.Items.Add(new ListBoxDataItem(put, Colors.Transparent));

}

private void Work()

{

var random = new Random();

for (int i = 0; i < this.ListBox1.Items.Count; i++)

{

var lbItem = this.ListBox1.Items[i] as ListBoxDataItem;

if (lbItem == null)

{

return;

}

do

{

lbItem = new ListBoxDataItem(lbItem, lbItem.Progress + 1, Colors.Yellow);

lbItem.BackColor.Freeze();

Application.Current.Dispatcher.BeginInvoke(

DispatcherPriority.DataBind,

new Action(() => this.ListBox1.Items[i] = lbItem));

Thread.Sleep(random.Next(0, 50));

}

while (lbItem.Progress < 100);

lbItem = new ListBoxDataItem(lbItem, Colors.GreenYellow);

lbItem.BackColor.Freeze();

Application.Current.Dispatcher.BeginInvoke(

DispatcherPriority.DataBind,

new Action(() => this.ListBox1.Items[i] = lbItem));

Thread.Sleep(50);

}

}

private void Close\_MenuItem(object sender, RoutedEventArgs e)

{

File.WriteAllText(currDir, "");

registr l = new registr();

l.Show();

Process.GetCurrentProcess().Kill();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

try

{

foreach (var drive in client.Masage(File.ReadAllText(currDir)))

{

FolderView.Items.Add(drive.Name);

}

}

catch (Exception ex)

{

this.Hide();

MessageBox.Show("Не удалось подключится к серверу", "Сервер не отвечает");

}

}

ServiceReference1.Service1Client client = new ServiceReference1.Service1Client();

private void download(object sender, RoutedEventArgs e)

{

try

{

OpenFileDialog op = new OpenFileDialog();

op.Title = "Выберите файл";

op.Filter = "Все файлы (\*.\*)|\*.\*";

if (op.ShowDialog() == true)

{

string put = op.FileName;

g.dovnload(put);

Ok o = new Ok("Отправлено");

o.Show();

FolderView.Items.Clear();

foreach (var drive in client.Masage(File.ReadAllText(currDir)))

{

FolderView.Items.Add(drive.Name);

}

}

}

catch (Exception ex)

{

Ok o = new Ok("Ошибка");

o.Show();

}

}

private void dwn()

{

try

{

string filePath = System.IO.Path.Combine("Download", FolderView.SelectedItem.ToString());

if (System.IO.File.Exists(filePath)) System.IO.File.Delete(filePath);

System.IO.Stream inputStream;

string fileName = FolderView.SelectedItem.ToString();

string s = File.ReadAllText(currDir);

long length = client.DownloadFile(ref fileName, ref s, out inputStream);

using (System.IO.FileStream writeStream = new System.IO.FileStream(filePath, System.IO.FileMode.CreateNew, System.IO.FileAccess.Write))

{

int chunkSize = 2048;

byte[] buffer = new byte[chunkSize];

do

{

int bytesRead = inputStream.Read(buffer, 0, chunkSize);

if (bytesRead == 0) break;

writeStream.Write(buffer, 0, bytesRead);

}

while (true);

writeStream.Close();

}

}

catch (Exception ex)

{

Ok o = new Ok("Ошибка");

o.Show();

}

}

private void FolderView\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

dwn();

Ok o = new Ok("Принято");

o.Show();

put = FolderView.SelectedItem.ToString();

this.FillListBox();

this.\_thread = new Thread(this.Work);

this.\_thread.Start();

}

string currDirк = Environment.CurrentDirectory.ToString() + @"/Download";

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

{

Process.Start(currDirк);

}

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

{

client.Delit(FolderView.SelectedItem.ToString(), File.ReadAllText(currDir));

FolderView.Items.RemoveAt(FolderView.SelectedIndex);

Ok o = new Ok("Удалено");

o.Show();

}

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

{

FolderView.Items.Clear();

foreach (var drive in client.Masage(File.ReadAllText(currDir)))

{

FolderView.Items.Add(drive.Name);

}

}

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

{

info i = new info();

i.Show();

}

}

}

Класс программы

[ValueConversion(typeof(string), typeof(BitmapImage))]

public class HeaderToImageConverter

{

облако.ServiceReference1.Service1Client client = new облако.ServiceReference1.Service1Client();

string currDir = Environment.CurrentDirectory.ToString() + @"\Code.txt";

public static HeaderToImageConverter Instance = new HeaderToImageConverter();

public string s { get; set; }

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

public void dovnload(string s)

{

try

{

if (!System.IO.Directory.Exists("Download")) System.IO.Directory.CreateDirectory("Download");

System.IO.FileInfo fileInfo = new System.IO.FileInfo(s);

using (System.IO.FileStream stream = new System.IO.FileStream(s, System.IO.FileMode.Open, System.IO.FileAccess.Read))

{

using (StreamWithProgress uploadStreamWithProgress = new StreamWithProgress(stream))

{

облако.ServiceReference1.Service1Client client = new облако.ServiceReference1.Service1Client();

client.UploadFile(fileInfo.Name, fileInfo.Length, File.ReadAllText(currDir), uploadStreamWithProgress);

client.Close();

}

}

Ok o = new Ok("Отправлено");

o.Show();

}

catch (Exception ex)

{

Ok o = new Ok("Ошибка");

o.Show();

}

}

public bool SetAutorunValue(bool autorun, string path)

{

const string name = "Cloud";

string ExePath = path;

RegistryKey reg;

reg = Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run\\");

try

{

if (autorun)

reg.SetValue(name, ExePath);

else

reg.DeleteValue(name);

reg.Close();

}

catch

{

return false;

}

return true;

}

}

Класс программы

class StreamWithProgress:Stream

{

public string maill { get; set; }

private readonly FileStream file;

private readonly long length;

public class ProgressChangedEventArgs : EventArgs

{

public long BytesRead;

public long Length;

public ProgressChangedEventArgs(long BytesRead, long Length)

{

this.BytesRead = BytesRead;

this.Length = Length;

}

}

public event EventHandler<ProgressChangedEventArgs> ProgressChanged;

private long bytesRead;

public StreamWithProgress(FileStream file)

{

this.file = file;

length = file.Length;

bytesRead = 0;

if (ProgressChanged != null) ProgressChanged(this, new ProgressChangedEventArgs(bytesRead, length));

}

public double GetProgress()

{

return ((double)bytesRead) / file.Length;

}

public override bool CanRead

{

get { return true; }

}

public override bool CanSeek

{

get { return false; }

}

public override bool CanWrite

{

get { return false; }

}

public override void Flush() { }

public override long Length

{

get { throw new Exception("Метод или операция не реализована."); }

}

public override long Position

{

get { return bytesRead; }

set { throw new Exception("Метод или операция не реализована."); }

}

public override int Read(byte[] buffer, int offset, int count)

{

int result = file.Read(buffer, offset, count);

bytesRead += result;

if (ProgressChanged != null) ProgressChanged(this, new ProgressChangedEventArgs(bytesRead, length));

return result;

}

public override long Seek(long offset, SeekOrigin origin)

{

throw new Exception("Метод или операция не реализована.");

}

public override void SetLength(long value)

{

throw new Exception("Метод или операция не реализована.");

}

public override void Write(byte[] buffer, int offset, int count)

{

throw new Exception("Метод или операция не реализована.");

}

}