Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Курганский государственный университет» (КГУ)

*Кафедра информационной безопасности автоматизированных систем*

**РАЗРАБОТКА ПРИЛОЖЕНИЯ  
«Разработка программных средств администрирования ОС Windows:** **Получение списка информационных ресурсов к которым имеет доступ на чтение задаваемый пользователь»  
на языке с#**

**Курсовая работа**

Дисциплина: Криптографические методы защиты информации

Студент: ИТ–30918 группы…………………………………. /Кандакова А.Н./

Направление:

Специальность: Информационная безопасность автоматизированных систем

Руководитель: к.п.н, доцент………………………………./Никифорова Т.А./

Комиссия: ….............................................................................../ /

….................................................................................../ /

….................................................................................../ /

Дата защиты:

Оценка:

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc315364790)

[1 Разработка эскизного и технического проектов программы «ADMIN» 4](#_Toc315364791)

[1.1 Назначение и область применения приложения «Admin» 4](#_Toc315364792)

[1.2 Технические характеристики 4](#_Toc315364793)

[1.2.1 Постановка задачи 4](#_Toc315364794)

[1.2.2 Описание алгоритма 7](#_Toc315364796)

[1.2.3 Организация входных и выходных данных 8](#_Toc315364797)

[1.2.4 Выбор состава технических и программных средств 8](#_Toc315364798)

[1.3 Разработка внешнего вида приложения 9](#_Toc315364799)

[2 Разработка рабочего проекта ПРИЛОЖЕНИЯ «Admin» 10](#_Toc315364800)

[2.1 Конструирование Windows-окна 10](#_Toc315364801)

[2.1.1 Визуальное проектирование приложения 10](#_Toc315364802)

[2.2 Написание кода программы 13](#_Toc315364803)

[2.3 Спецификация программы 21](#_Toc315364804)

[2.4 Описание программы и рекомендации по использованию 21](#_Toc315364805)

[2.5 Тестирование программы 21](#_Toc315364806)

[Заключение 22](#_Toc315364807)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 23](#_Toc315364808)

[ПРИЛОЖЕНИЕ 23](#_Toc315364809)

# ВВЕДЕНИЕ

Неотъемлемыми частями системного администрирование являются документирование всех произведенных процессов, учёт данных и пользовательских записей. В компаниях с большим штатом сотрудников следить за активностью каждого пользователя, а тем более документирования всех изменений, достаточно проблематично. Как правило, на штаб сотрудников приходится от 1-го до несколько системных администраторов.

Для облегчения работы с пользователями в большинстве случаях используются операционные системы семейства Windows NT. На это есть несколько причин:

1. ОС Windows наиболее распространена (в отличии от Unix-систем), а, следовательно, инстинктивно понятливее для рядового пользователя.
2. Также, по выше указанной причине, большинство программ создавались и создаются под Windows.
3. Современные версии Windows не уступают по уровню безопасности Linux.[1].

Целью курсовой работы является написание приложения, которое предоставляло бы пользователю с приоритетным правом (администратор, специалист по информационной безопасности) возможность получения списка файлов, ключей реестра, доступных для текущего пользователя Windows. В качестве дополнительной функции – сервер-приложение должно получать активность клиент-приложений.

Исходя из цели, можно сформулировать следующие задачи:

1. Проанализировать литературу по теме исследования.
2. Изучить и определить основные особенности Windows OS.
3. Спроектировать внешний вид приложения и написать код приложения.
4. Протестировать работу приложения.

Курсовая работа состоит из введения, двух глав, заключения и списка использованных источников.

# 1 Разработка эскизного и технического проектов программы «ADMIN»

## **1.1 Назначение и область применения приложения «Admin»**

Разрабатываемое приложение «Admin» предназначено для получения списка файлов и ключей реестра доступных для текущего пользователя OS Windows.

Разрабатываемое приложение может использоваться в плане информационной безопасности. Сервер отправляет запрос на получение данных выбранного клиента. Клиент свою очередь собирает необходимые данные и отправляет серверу результат.

## **1.2 Технические характеристики**

### *1.2.1 Постановка задачи*

Целью курсовой работы является разработка приложения, которое будет получать перечень доступных файлов и ключей реестра. Такая программа должна быть проста в использовании, иметь удобны «дружелюбный» интерфейс, чтобы пользователь приложения за несколько «кликов» мог получить список данных текущей учетной записи Windows.

Далее следует краткое описание файловой системы, реестра и Windows.

На данный момент наиболее распространёнными операционными системами семейства Windows NT: Windows 7, Windows 8.1, Windows 10. Все они являются операционными системами с многозадачностью и разработаны как для компьютеров с одним процессором, так и для компьютеров с несколькими процессорами.

Архитектура Windows NT имеет модульную структуру, состоящую из двух основных уровней: режим пользователя и режим ядра.

Пользовательский режим использует подсистемы (подсистема окружения и интегрированная подсистема), которые передают запросы ввода/вывода необходимому драйверу режима ядра. В свою очередь режим ядра имеет полный доступ к аппаратной части и ресурсам.

Интегрированная подсистема отвечает за службы безопасности, рабочей станции и сервера. Служба безопасности отвечает за уровни доступа, учетную запись пользователя и доступ к ней. Рабочая станция отвечает за доступ к сети, а служба Сервер – за сетевые сервисы.

Выделим основные плюсы Windows:

* Система проста в использовании;
* Большинство программ пишутся под Windows;
* Постоянная поддержка и обновления (касательно новых версий);
* Универсальность системы.

Однако стоит выделить и минусы:

* Достаточно требовательная система к ресурсам ПК.
* Так же, как и программы, большое количество вирусов пишутся для системы Windows;
* Высокая цена лицензии, несмотря на бесплатные версии, позже с них придётся перейти на платную;
* Несмотря на достаточно неплохую безопасность, система не защищена от критических ошибок. Такие ошибки часто сопровождают Windows (неверное обновление, нехватка памяти и другое);

Далее рассмотрим реестр. Реестр, по словарю корпорации   
Майкрософт [2], представляет собой иерархическую центральную базу данных используемы во многих версиях операционной системы Windows. База хранит в себе необходимую информацию о настройках ОС, приложений. Реестр содержит сведения о автозагрузки, параметрах и свойствах папок, приложений, профили пользователей. Следует вести следующее понятие – куст реестра. Куст реестра – группа разделов, подразделов, значений в реестре. Кроме того, куст содержит вспомогательные файлы с резервной копией информации.

*Таблица 1*

*Перечень кустов реестра*

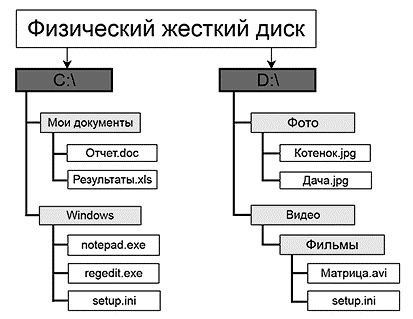
|  |  |
| --- | --- |
| Куст реестра | Вспомогательные файлы |
| HKEY\_LOCAL\_MACHINE\SAM | SAM, SAM. log, SAM. САВ |
| HKEY\_LOCAL\_MACHINE\Security | Безопасность, Security. log, Security. САВ |
| HKEY\_LOCAL\_MACHINE\Software | Программное обеспечение, Software. log, Software. САВ |
| HKEY\_LOCAL\_MACHINE\System | System, System. Alt, System. log, System. САВ |
| HKEY\_CURRENT\_CONFIG | System, System. Alt, System. log, System. САВ, NTuser. dat, NTuser. dat. log |
| HKEY\_USERS\DEFAULT | По умолчанию, Default. log, Default. САВ |

В целях безопасности управлять доступом к реестру Windows разрешено учетной записи с правами администратора. В следующей таблице (Таблица 2) перечислены стандартные ключи, используемые системой.

*Таблица 2*

|  |  |
| --- | --- |
| Папка/предварительно определенный раздел | Описание |
| HKEY\_CURRENT\_USER | Здесь хранятся папки пользователя, цвета экрана и параметры панели управления. |
| HKEY\_USERS | Содержит все активные загруженные профили пользователей на компьютере. |
| HKEY\_LOCAL\_MACHINE | Содержит сведения о конфигурации, определенные для компьютера (для любого пользователя). |
| HKEY\_CLASSES\_ROOT | Хранящиеся здесь сведения позволят убедиться в том, что при открытии файла с помощью проводника Windows откроется Правильная программа. |
| HKEY\_CURRENT\_CONFIG | Содержит сведения о профиле оборудования, используемом локальным компьютером при запуске системы. |

Теперь рассмотрим структуру файловой системы Windows. Система представляет собой древовидную структуру директорий, облегающих поиск и хранение информации. Внутри одной директории может храниться несколько других каталогов. Группировка элементов зависит от пользователя.



*Рисунок 1 – Разделение физического носителя.*

Для более простого хранения файлов принято разделять физический диск, как правило HDD –диски, на логические разделы. Обычно первый логический раздел используется для ОС, второй – хранение документов, программ.

### *1.2.2 Описание алгоритма*

При запуске сервер Сервер-приложения буду открыто главное с настройками IP-адреса, порта и ключа доступа. Ключ доступа является не обязательным параметром. Сервер, как и клиент включает все себя аналог чата, который может сообщать об ошибках операций. Так же доступен список подключенных клиентов.

Клиент-приложение включает в свою очередь чат и настройку подключения: IP-адрес, порт, ключ доступа. Имя клиента определяется учетной записью.

### *1.2.3 Организация входных и выходных данных*

Входными данными для сервера – строка “<GetDate>” содержащая параметры запроса:

* «-D» -диски
* «-I» - файлы
* «-R» - ключи

Пример отправляемого запроса с сервера на клиент:  
 ”<GetDate –D –I –R>”. Клиент отправляет результат с параметром “<GetDateResult>”, который считывается сервером в несколько файлов формата “.txt” в зависимости от количества полученных пакетов.

### *1.2.4 Выбор состава технических и программных средств*

Средой разработки приложения была выбрана интегрированная среда разработки ПО для Microsoft Windows – Visual Studio 2019. Данная среда разработки поддерживает множество языков, в том числе C# (си шарп) и C++.

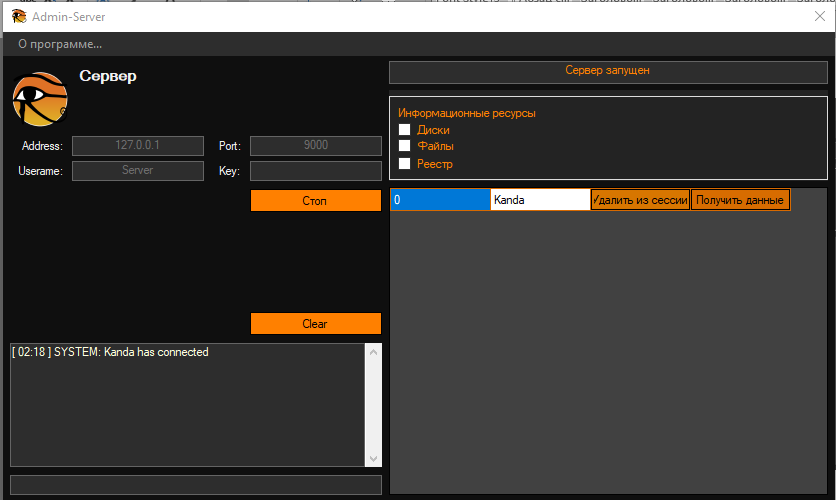
Выбор данной среды разработки обусловлен следующими причинами:

1. Поддержка SQL Server и SQL Client и других баз данных– более удобное и быстрое редактирование баз данных.
2. Визуальное построение приложений из программных прототипов, что значительно ускоряет разработку.
3. Наличие стандартных - для Windows Form, и нестандартных - для WPF (Windows Presentation Foundation), компонентов окна.
4. Полная поддержка таких кодировок, как Unicode и ANSI. Применение этих кодировок гарантирует, что приложения будут одинаково выглядеть и функционировать во всех языковых версиях Windows.

В качестве языка выбран C#, достаточно тонкая настройка интерфейса программы. Использование языка интегрированных LINQ даёт возможность писать структурированные безопасные в отношении типов запросы к локальным коллекциям объектов и удаленным данных.

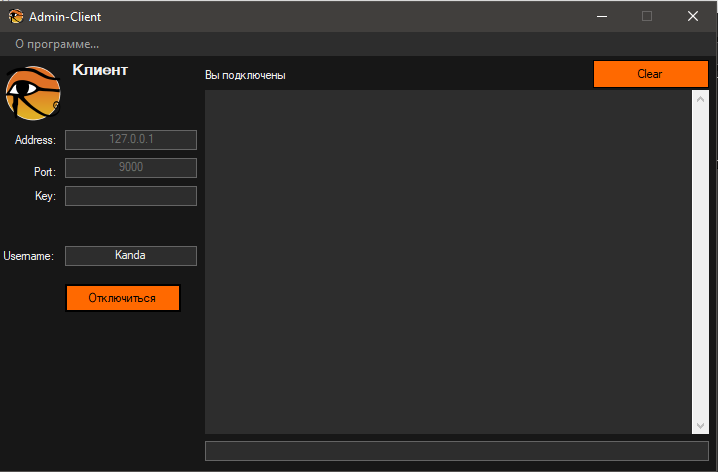
## **1.3 Разработка внешнего вида приложения**

Интерфейс приложения должен быть наиболее удобным и простым в использовании, поэтому на форме приложения компоненты будут располагаться компактно.



*Рисунок 2 - Внешний вид работающего приложения (Сервер)*

Форма сервера для удобства имеет выбор параметров для получения информационных ресурсов: «Диски», «Файлы», «Реестр». Программа сообщит в чат об ошибках подключение или ошибках выполнения самой программы в чат.

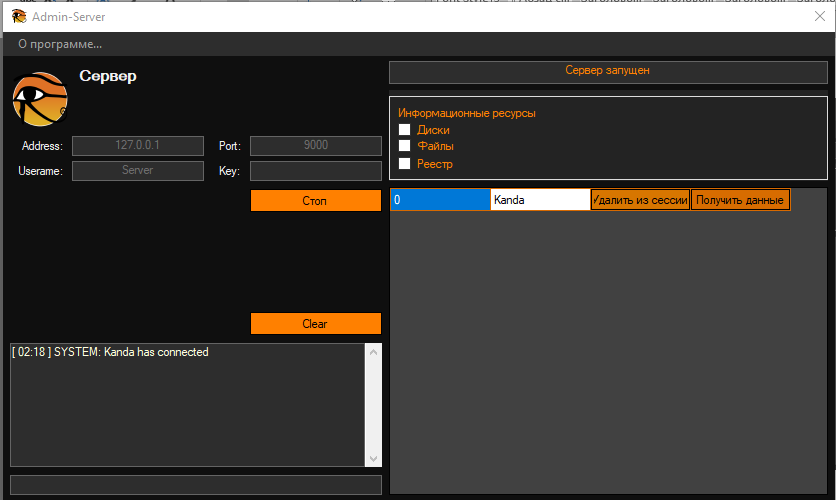
**

*Рисунок 3 - Внешний вид работающего приложения (Клиент)*

# 2 Разработка рабочего проекта ПРИЛОЖЕНИЯ «мЕТОДЫ ШИФРОВАНИЯ ИНФОРМАЦИИ»

## **2.1 Конструирование Windows-окна**

### 2.1.1 Визуальное проектирование приложения



statusBox

clientsDataGridView

DC и GetInfo

sendTextBox

startButton

*Рисунок 4 Визуальные и невизуальные компоненты главной формы приложения (Сервер)*

*Таблица 3*

*Назначение компонентов главной формы приложения(Сервер)*

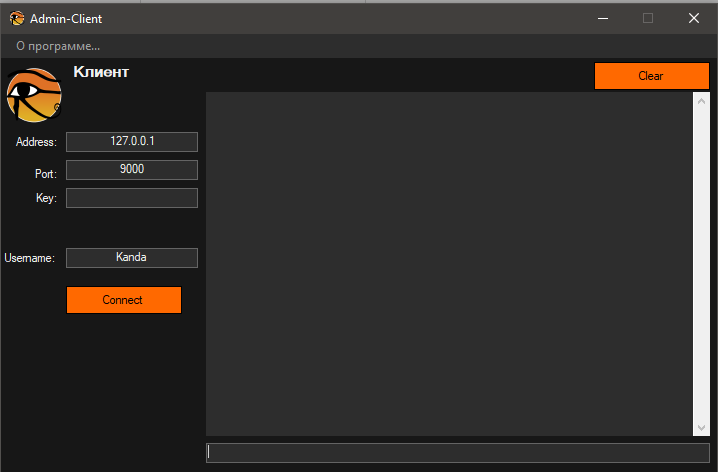
| ***№ п/п*** | ***Компонент*** | ***Назначение*** |
| --- | --- | --- |
| 1 | AdminServerForm | Главная форма приложения |
| 2 | оПрограммеToolStripMenuItem | Описание программы в отдельном окне |
| 3 | startButton | Запуск/Остановка Сервера |
| 4 | clearButton | Очистка области чата |
| 5 | clientsDataGridView | Таблица подключений |
| 6 | DC | Кнопка отключение текущего пользователя от сервера |
| 7 | GetInfo | Получить информацию о клиенте |
| 8 | logTextBox | Чат |
| 9 | sendTextBox | Ввод сообщений |
| 10 | addrTextBox | Ввод IP -адреса |

|  |  |  |
| --- | --- | --- |
| 11 | usernameTextBox | Имя сервера (используется только в чате) |
| 12 | portTextBox | Порт |
| 13 | keyTextBox | Ключ доступа к серверу |
| 14 | statusBox | Состояние сервера |
| 15 | disks | ComboBox. Выбор информации о дисках |
| 16 | files | ComboBox. Выбор информации о файлах |
| 17 | regedit | ComboBox. Выбор информации о реестре |
| 18 | logTextBox | Чат |
| 19 | sendTextBox | Ввод сообщений |

*Таблица 4*

*Изменённые свойства компонентов главной формы приложения (Сервер)*

| *№ п/п* | *Компонент* | *Свойства* |
| --- | --- | --- |
| 1 | startButton | Text = “Start” |
| 2 | statusBox | Text = “Сервер запущен” |

****

*Рисунок 5 Визуальные и невизуальные компоненты главной формы приложения (Клиент)*

*Таблицы 5*

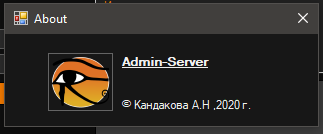
*Назначение компонентов главной формы приложения (Клиент)*

| ***№ п/п*** | ***Компонент*** | ***Назначение*** |
| --- | --- | --- |
| 1 | usernameTextBox | Имя сервера (используется только в чате) |
| 2 | portTextBox | Порт |
| 3 | keyTextBox | Ключ доступа к серверу |
| 4 | connectButton | Кнопка подключения к серверу |
| 5 | statuslabel | Статус клиента |
| 6 | оПрограммеToolStripMenuItem | Описание программы в отдельном окне |
| 7 | logTextBox | Чат |
| 8 | sendTextBox | Ввод сообщений |
| 9 | clearButton | Очистка области чата |
| 10 | addrTextBox | Ввод IP –адреса сервера |
| 11 | usernameTextBox | Имя пользователя |
| 12 | portTextBox | Порт |
| 13 | keyTextBox | Ключ доступа к серверу |

*Таблица 6*

*Изменённые свойства компонентов главной формы приложения (Клиент)*

| *№ п/п* | *Компонент* | *Свойства* |
| --- | --- | --- |
| 1 | connectButton | Text= “Connect” |
| 2 | statuslabel | Text=”Вы подключены” |

**

*Рисунок 6 Форма «о программе»*

## **2.2 Написание кода программы**

При работе программы использованы следующие процедуры:

*Таблица 5*

*Используемые процедуры и функции в главной форме приложения (Сервер)*

| *Процедура* | *Назначение* |
| --- | --- |
| Log(string msg = "") | Используется для вывода в logTextBox данных: ошибки, системные сообщения, сообщения с клиентов. |
| ErrorMsg(string msg) | Конструктор сообщений об ошибке |
| SystemMsg(string msg) | Конструктор системных сообщений. |
| Active(bool status) | Отвечает за запуск и остановку сервера. |
| AddToGrid(long id, string name) | Добавляет подключенного клиента в список clientsDataGridView. |
| RemoveFromGrid(long id) | Удаляет выбранного клиента из списка clientsDataGridView. Вызывается из события ClientsDataGridView\_CellClick с помощью кнопки dc. |
| ClientsDataGridView\_CellClick | Обработка и проверка двойных нажатий мышкой в ячейку. |
| ParseMessage(string message) | Проверка сообщения на команду. |
| Read(IAsyncResult result) | Чтение исходящих от клиентов данных. |
| ReadAuth(IAsyncResult result) | Авторизация клиента, читает данные о клиенте. Вызывается из Authorize |
| Authorize(MyClient obj) | Основная функция авторизации, вызывается из Connection. |
| Connection(MyClient obj) | Главная функция соединения клиентов с сервером. Отвечает за проверку соединения, результат выводить в logTextBox. Также добавляет или удаляет клиентов, если у тех произошёл сбой. |
| Listener(IPAddress ip, int port) | Проверка портов. В случает отсутствии Интернета выдаёт ошибку. |

| StartButton\_Click | Обработка события StartButton. |
| --- | --- |
| BeginWrite(string msg, MyClient obj) | Отправка сообщения определенному клиенту. |
| BeginWrite(string msg, long id = -1) | Отправка сообщения всем клиентам. |
| Write(IAsyncResult result) | Функция отправки сообщения используется в методах BeginWrite. |
| Send(string msg, MyClient obj) | Асинхронный метод отправки конкретному пользователю. Вызывает функции BeginWrite. Асинхронность методов необходима для работы с большим количеством клиентов. |
| SendTextBox\_KeyDown | Обработка ввода. Если клавиша “Enter”, то отправляет сообщения |
| AboutToolStripMenuItem\_Click | Вызывает форму с описанием программы. AboutAppForm() |
| ExitToolStripMenuItem\_Click | Выход из программы, при помощи контекстного меню. |
| Disconnect(long id = -1) | Завершение всех потоков. Вызывается при закрытии или остановки сервера. |
| AdminServerForm\_FormClosing | Закрытие главной формы |
| AboutAppForm() | Запуск формы с описанием. |
| ShowStatus(string status) | Устанавливает статус сервера |
| GetInformation(long id) | Создание команды по получению информационных ресурсов. Отправка её клиенту. |

Многие функции клиента повторяют функции сервера. Однако их стоит перечислить.Помимо этого, клиентское приложение содержит класс для работы с файловой системой.

*Таблица 6*

*Используемые процедуры и функции в главной форме приложения (Клиент)*

| *Процедура* | *Назначение* |
| --- | --- |
| Log(string msg = "") | Используется для вывода в logTextBox данных: ошибки, системные сообщения, сообщения с клиентов. |
| Active(bool status) | Отвечает за запуск и остановку сервера. |
| AdminClientForm\_Load | Используется для автоматического определения имени учетной записи |
| ParseMessage(string message) | Проверка сообщения на команду. |
| Read(IAsyncResult result) | Чтение исходящих от сервера данных. |
| ReadAuth(IAsyncResult result) | Авторизация клиента, читает данные о клиенте. Вызывается из Authorize |
| Authorize(MyClient obj) | Основная функция авторизации, вызывается из Connection. |
| Connection(MyClient obj) | Главная функция соединения клиентов с сервером. |
| ConnectButton\_Click | Отвечает за соединение с сервером. |

*Таблица 7*

*Класс GetDate (Клиент)*

| *Процедура* | *Назначение* |
| --- | --- |
| ParseCommand(string command) | Обработка команды, вызов функций работы с информационными ресурсами по параметрам команды. |
| GetKeys() | Получить ключи реестра |

| GetDisk() | Получить информацию о текущих дисках. |
| --- | --- |
| GetInformation() | Информация о файлах пользователя (выдаёт список). |
| SearchFile(string patch) | Поиск файлов по пути. |
| SearchDirectory(string patch) | Поиск вложенных каталогов по пути. |

Приведём описание некоторых из вышеописанных процедур главной формы приложения «Admin».

Список процедур главной формы приложения для более подробного описания:

Для начала рассмотрим работу серверно-клиентской части:

Запуск Сервера:

private void StartButton\_Click(object sender, EventArgs e)

{

if (active)

{ active = false; }

else if (listener == null || !listener.IsAlive)

{

string address = addrTextBox.Text.Trim();

string number = portTextBox.Text.Trim();

string username = usernameTextBox.Text.Trim();

bool error = false;

IPAddress ip = null;

if (address.Length < 1) // проверка адреса

{

error = true;

Log(SystemMsg("Необходимо ввести IP-адрес"));

}

else {

try

{

ip = Dns.Resolve(address).AddressList[0];

// Разрешает DNS-имя узла

}

сatch //в случае исключения выдаёт ошибку

{

error = true;

Log(SystemMsg("Не валидный адресс"));

}

}

int port = 0;

if (number.Length < 1 || port < 0 || port > 65535 || !int.TryParse(number, out port))

{

error = true;

Log(SystemMsg("Неверный порт"));

}

if (!error) // если все введенные данные верны, запускает проверку портов – асинхронно.

{

listener = new Thread(() => Listener(ip, port))

{

IsBackground = true

};

listener.Start();

}}}

Далее нужно показать пользователю сервера, что запуск произошёл:

private void Active(bool status)

{

if (!exit)

{

startButton.Invoke((MethodInvoker)delegate

{

active = status;

if (status)

{

addrTextBox.Enabled = false;

portTextBox.Enabled = false;

usernameTextBox.Enabled = false;

keyTextBox.Enabled = false;

startButton.Text = "Stop";

statusBox.Text ="Сервер запущен";

}

else

{

addrTextBox.Enabled = true;

portTextBox.Enabled = true;

usernameTextBox.Enabled = true;

keyTextBox.Enabled = true;

startButton.Text = "Start";

statusBox.Text = "Сервер остановлен";

}

});

}

}

*Примечание: Асинхронные методы просто необходимы для работы с формой, так последовательно выполнение может привести к ошибке. Возможен отказ сервера или критический сбой программы.*

Сейчас на данный момент в фоновом режиме циклично работает проверка портов. Как только клиент подключиться к серверу произойдёт его авторизация.

private void Listener(IPAddress ip, int port)

{

TcpListener listener = null;

try

{

listener = new TcpListener(ip, port);

listener.Start();

Active(true);

while (active)

{

if (listener.Pending())

{

try

{

MyClient obj = new MyClient();

obj.id = id;

obj.username = new StringBuilder();

obj.client = listener.AcceptTcpClient();

obj.stream = obj.client.GetStream();

obj.buffer = new byte[obj.client.ReceiveBufferSize];

obj.data = new StringBuilder();

obj.handle = new EventWaitHandle(false, EventResetMode.AutoReset);

Thread th = new Thread(() => Connection(obj))

{

IsBackground = true

};

th.Start();

id++;

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

else

{

Thread.Sleep(500);}}

Active(false);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

finally

{

if (listener != null)

{

listener.Server.Close();

}}}

Подключение клиента начинается со следующего события:

private void ConnectButton\_Click(object sender, EventArgs e)

{

if (connected)

{

obj.client.Close();

}

else if (client == null || !client.IsAlive)

{

string address = addrTextBox.Text.Trim();

string number = portTextBox.Text.Trim();

string username = usernameTextBox.Text.Trim();

bool error = false;

IPAddress ip = null;

if (address.Length < 1)

{

error = true;

Log(SysteMessage("Требуется IP-адрес"));

}

else

{

try

{

ip = Dns.Resolve(address).AddressList[0];

}

catch

{

error = true;

Log(SysteMessage("Неверный IP"));

}

}

int port = 0;

if (number.Length < 1)

{

error = true;

Log(SysteMessage("Требуется порт"));

}

else if (!int.TryParse(number, out port)|| port < 0 || port > 65535)

{

error = true;

Log(SysteMessage("Неверный порт"));

}

if (!error)

{

client = new Thread(() => Connection(ip, port, username, keyTextBox.Text))

{

IsBackground = true

};

client.Start();

}

}

}

Как и при запуске сервера, проверяются входные данные, но фоновым потоком запускается функция Connection, которая в свою очередь собирает информацию необходимую для подключения к серверу и отправляет данные для авторизации.

Клиент:

private void Connection(IPAddress ip, int port, string username, string key)

{

try

{

obj = new MyClient();

obj.username = username;

obj.key = key;

obj.client = new TcpClient();

obj.client.Connect(ip, port);

obj.stream = obj.client.GetStream();

obj.buffer = new byte[obj.client.ReceiveBufferSize];

obj.data = new StringBuilder();

obj.handle = new EventWaitHandle(false, EventResetMode.AutoReset);

if (Authorize())

{

while (obj.client.Connected)

{

try

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(Read), null);

obj.handle.WaitOne();

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

obj.client.Close();

Connected(false);

}

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

Функции сервера, которые авторизуют клиента:

Получает данные из Connection клиента, делает проверку соединения.

private bool Authorize(MyClient obj)

{

bool success = false;

while (obj.client.Connected)

{

try

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(ReadAuth), obj);

obj.handle.WaitOne();

if (obj.username.Length > 0)

{

success = true;

break;

}

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

return success;

}

Считывает данные для авторизации и сравнивает с данными сервера. Если данные не равны – запрет подключения:

private void ReadAuth(IAsyncResult result)

{

MyClient obj = (MyClient)result.AsyncState;

int bytes = 0;

if (obj.client.Connected)

{

try

{

bytes = obj.stream.EndRead(result);

}

catch (Exception ex)

{

Log(ErrorMsg(ex.Message));

}

}

if (bytes > 0)

{

obj.data.AppendFormat("{0}", Encoding.UTF8.GetString(obj.buffer, 0, bytes));

try

{

if (obj.stream.DataAvailable)

{

obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(ReadAuth), obj);

}

else

{

JavaScriptSerializer json = new JavaScriptSerializer();

Dictionary<string, string> data = json.Deserialize<Dictionary<string, string>>(obj.data.ToString());

if (!data.ContainsKey("username") || data["username"].Length < 1)

{

obj.client.Close();

}

else if (!data.ContainsKey("key") || !data["key"].Equals(keyTextBox.Text))

{

obj.client.Close();

}

else

{

obj.username.Append(data["username"].Length > 200 ? data["username"].Substring(0, 200) : data["username"]);

Send("{\"status\": \"authorized\"}", obj);

}

obj.data.Clear();

obj.handle.Set();

}

}

catch (Exception ex)

{

obj.data.Clear();

Log(ErrorMsg(ex.Message));

obj.handle.Set();

}

}

else

{

obj.client.Close();

obj.handle.Set();

}

}

Необходимо показать клиенту результат подключения – Connected(bool status)

После успешной проверки и авторизации результат будет выглядеть следующим образом.



*Рисунок 7 Результат подключения*

Теперь рассмотрим функции работы данных. Обработка информации начинается со следующего события ClientsDataGridView\_CellClick.

private void ClientsDataGridView\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex == clientsDataGridView.Columns["DC"].Index)

……..

//Если была нажата “Получить данные”

else if (e.RowIndex >= 0 && e.ColumnIndex == clientsDataGridView.Columns["GetInfo"].Index)

{

long.TryParse(clientsDataGridView.Rows[e.RowIndex].Cells["IDuser"].Value.ToString(), out long id);

GetInformation(id);

}

}

Далее вызывается функция GetInformation(id), которая получает данные из выбранных comboBox’ов и создаёт команду с необходимыми параметрами.

private void GetInformation(long id)

{

if (disks.Checked || files.Checked || regedit.Checked)

{

string command = "<GetDate ";

if (disks.Checked)

{

command += "-D ";

}

if (files.Checked)

{

command += "-I ";

}

if (regedit.Checked)

{

command += "-R";

}

command += "> ";

clients.TryGetValue(id, out MyClient obj);

Send(command, obj);

}

else

{

MessageBox.Show("Выберите тип ресурсов");

}

}

Далее эта команда отправляется к определённому клиенту пользователю, помощью функций Send. Со стороны клиента функция Read(IAsyncResult result) получает данное сообщение, при этом происходит проверка.

public List<string> information = new List<string>();

public bool ParseMessage(string message)

{

if (message.Contains("<GetDate"))

{

GetDate j = new GetDate();

j.ParseCommand(message);

List<string> report = j.information;

foreach (string g in report)

{

Send(g);

}

return true;

}

else

{

return false;

}

}

Если мы получили команду, то вызывается функция ParseCommand(string command), иначе это обычное сообщение.

public List<string> ParseCommand(string command)

{

if (command.Contains("-D"))

{

GetDisk();

}

if (command.Contains("-I"))

{

GetInformation();

}

if (command.Contains("-R"))

{

GetKeys();

}

// идёт обработка параметров

return information;

}

Стоит рассмотреть каждую функцию отдельно.

Получение ключей.

*Примечание: Пришлось искусственно ограничить получение ключей – большое количество ключей вызывало цикл обработки информации, что значительно долго влияло, вместо это мы получаем данные определённого куста реестра.*

public void GetKeys()

{

string ListKeys = String.Format("<GetDateResult> {0} ", Environment.UserName);

const string REGISTRY\_ROOT = @"SOFTWARE\";

using (RegistryKey rootKey = Registry.CurrentUser.OpenSubKey(REGISTRY\_ROOT, true))

{

if (rootKey != null)

{

string[] valueNames = rootKey.GetSubKeyNames();

StringBuilder sb = new StringBuilder(valueNames.Length);

foreach (string ch in valueNames)

{

sb.Append("\n");

sb.Append(ch);

}

ListKeys += sb.ToString();

}

rootKey.Close();

}

information.Add(ListKeys);

}

Получение информации о дисках текущего ПК.

public void GetDisk()

{

StringBuilder driveList = new StringBuilder();

foreach (DriveInfo d in DriveInfo.GetDrives())

{

if (d.IsReady) driveList.AppendLine(String.Format("Диск: {0}; метка тома: {1}; файловая система: {2}; тип: {3}; объем: {4} байт; свободно: {5} байт", d.Name, d.VolumeLabel, d.DriveFormat, d.DriveType, d.TotalSize, d.AvailableFreeSpace));

}

string temp = String.Format("<GetDateResult> {0} {1}",Environment.UserName, driveList.ToString());

information.Add(temp);

}

Получение информации о файлах и каталогов каждого диска на ПК. Исключение try-catch получает доступные для пользователя файлы и катологи.

public void GetInformation()

{

string ListPatch = String.Format("<GetDateResult> {0} ", Environment.UserName);

foreach (DriveInfo d in DriveInfo.GetDrives())

{

string[] S = SearchDirectory(d.Name);

foreach (string folderPatch in S)

{

try

{

string[] F = SearchFile(folderPatch);

foreach (string FF in F)

{

ListPatch += FF + "\n";

}

}

Catch{}}

}

information.Add(ListPatch);

}

Функции для рекурсивного поиска файлов, каталогов.

static string[] SearchFile(string patch)

{

string[] ReultSearch = Directory.GetFiles(patch, "\*", SearchOption.AllDirectories);

return ReultSearch;

}

static string[] SearchDirectory(string patch)

{

string[] ReultSearch = Directory.GetDirectories(patch);

return ReultSearch;

}

## **2.3 Спецификация программы**

Состав проекта

|  |  |
| --- | --- |
| *Наименование* | *Обозначение* |
| AdminServer | Исполняемый файл приложения (Windows-приложение). Программа сервер. |
| Admin | Исполняемый файл приложения (Windows-приложение). Программа клиент. |

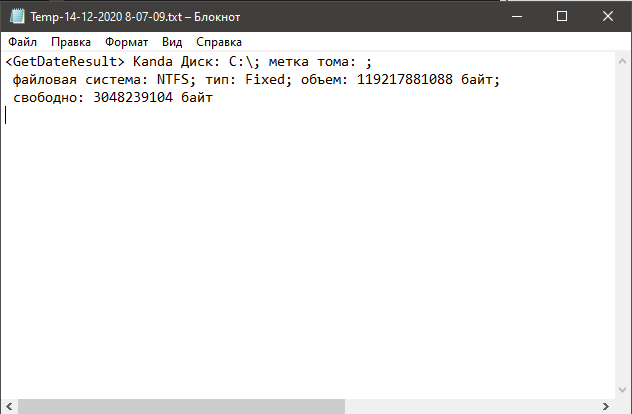
## **2.4 Описание программы и рекомендации по использованию приложения**

Данная программа проста в использовании. Для запуска достаточно запустить файлы «AdminServer.exe» на компьютере-сервере и «AdminServer.exe» на компьютере-клиенте.

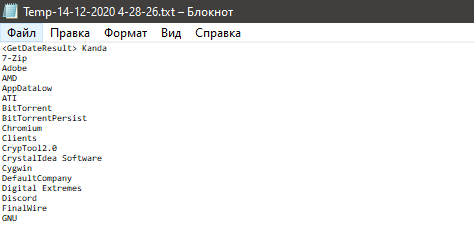
## **2.5 Тестирование программы**

Программа прошла тестирование в несколько этапов и было выявлено несколько уязвимостей связанных с безопасностью использования. Однако программа выполняет свои основные функции.

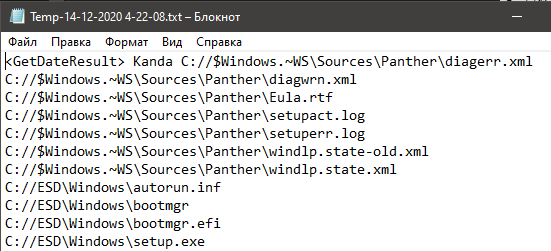
Ниже приведены примеры работы:



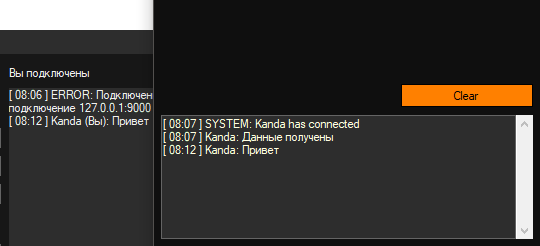
*Рисунок 8 Информация о дисках*



*Рисунок 9 Ключи*



*Рисунок 10 Результат подключения*



*Рисунок 11 Проверка работы чата.*

При тестировании программ были выделены следующие уязвимости:

* Скорость отправки и создание конечных файлов при большом объеме данных клиента – чем больше информации у клиентов, тем больше создаётся файлов.
* Конфиденциальность сервера и клиента – данные связанные с IP –адресом, портом и ключом никак не скрываются.

Основными способами решения проблем являются: добавление свойств или элементов для скрытия данных, при отправке команды можно использовать сторонние порты. Также для защиты от добавления зловредного кода следует пресечь попытки с помощью обфускации кода. При этом не помешает добавить криптографических функций при обработке данных: хэш-функции и так далее.

# **Заключение**

Целью курсовой работы явилось написание приложения, которое предоставлял список информационных ресурсов. Приложение выполняет основные функции. При этом дополнительно была добавлена серверно-клиентская часть программы. Цель была достигнута.

При написании курсовой работы были выполнены следующие задачи:

1. Проанализирована литературу по теме исследования.
2. Выявлены основные положительные и отрицательные черты Windows OS.
3. Спроектирован внешний вид приложения и написан код приложения.
4. Протестирована работу приложения.

Поставленные задачи выполнены.

Данная программа имеет возможность развития: добавление дополнительных функций для системного администрирования.

Практическая значимость данного приложения – возможность получать данные, к примеру, компьютеров офиса удаленно, что может улучшить производительность труда.

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Habr.com – блог о технологиях: «Веб-хостинг: Windows или Linux?»: офиц. сайт – URL <https://habr.com/ru/company/ruvds/blog/322650/> (дата обращения: 11.01.2020)
2. Домашняя страница документации и учебных ресурсов Майкрософт для разработчиков и технических специалистов: Сведения о реестре Windows для опытных пользователей: офиц. сайт – URL https://docs.microsoft.com/ru-ru/troubleshoot/windows-server/performance/windows-registry-advanced-users  
   (дата обращения: 5.10.2020)
3. Официальная Документация по Windows 10: офиц. сайт – URL https://docs.microsoft.com/ru-ru/windows/windows-10/ (дата обращения: 9.01.2020)
4. Юрий Белов. Администрирование серверных операционных систем семейства Windows. // МГТУ им Н.Э. Баумана 2014. - 325 стр.   
   (дата обращения 17.01.2020)
5. Петр Шетка. Microsoft Windows server 2003. Практическое руководство по настройки сети // СПб.: Наука и Техника. 2006. – 608 стр. (дата обращения: 24.10.2020)

# **ПРИЛОЖЕНИЕ**

Структура диска: