МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ   
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
имени И.С. ТУРГЕНЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки 01.03.02 Прикладная математика и информатика

направленность (профиль) Системное программирование   
и компьютерные технологии

Студента Тютюнова Игоря Андреевича шифр 150710

Факультет физико-математический

Тема выпускной квалификационной работы

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА И МЕНЕДЖМЕНТА КОМПЬЮТЕРОВ ПО ЛОКАЛЬНОЙ СЕТИ

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тютюнов И.А.

Научный   
руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Федяев Ю.С.

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дорофеева В.И.

Орёл 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ   
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
имени И.С. ТУРГЕНЕВА»

Факультет физико-математический

Кафедра информатики

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Системное программирование и компьютерные технологии

УТВЕРЖДАЮ:

Зав. кафедрой

\_\_\_\_\_\_\_\_ Дорофеева В.И.

«25» октября 2019 г.

**ЗАДАНИЕ**

на выполнение выпускной квалификационной работы

студента Тютюнова Игоря Андреевича шифр 150710

1. Тема ВКР Разработка приложения для мониторинга и менеджмента компьютеров по локальной сети

Утверждена приказом по университету от «25» октября 2019 года № 2–3218

2. Срок сдачи студентом законченной работы «06» июля 2020 года.

3. Исходные данные к работе: учебные пособия и курсы лекций по аспектам разработки программного обеспечения, электронные источники и ссылки по программированию, алгоритмизации, разработке приложений.

4. Содержание ВКР (перечень подлежащих разработке вопросов)

Введение

Глава 1. Теоретические сведения о проектировании приложения для мониторинга и менеджмента компьютеров по локальной сети

* 1. Теоретические сведения о способах программирования клиент-серверных приложений на языке программирования C#
  2. Описание логики взаимодействия приложения администратора с клиентским приложением

Глава 2. Проектирование интерфейса и программная реализация клиент-серверного приложения для мониторинга компьютеров по локальной сети

2.1. Проектирование интерфейса и программная реализация приложения администратора

2.1.1. Проектирование интерфейса

2.1.2. Программная реализация

2.2. Проектирование интерфейса и программная реализация клиентского приложения

Заключение

Список источников

5. Перечень графического материала: рисунков – 9, таблиц – 1

Дата выдачи задания «25» октября 2019 г.

Научный   
руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Федяев Ю.С.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тютюнов И.А.

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| Наименование этапов  ВКР | Срок выполнения  этапов работы | Примечание |
| Подбор и анализ источников и научных изданий в соответствии с темой исследования | ноябрь – декабрь 2019 | Выполнено |
| Написание введения | апрель 2020 г. | Выполнено |
| Написание главы 1 | май 2020 г. | Выполнено |
| Написание главы 2 | май 2020 г. | Выполнено |
| Написание заключения | май 2020 г. | Выполнено |
| Оформление ВКР | июнь 2020 г. | Выполнено |
| Сдача ВКР | июль 2020 г. | Выполнено |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тютюнов И.А.

Научный   
руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Федяев Ю.С.

# Аннотация

Выпускная квалификационная работа на тему «Разработка приложения для мониторинга и менеджмента компьютеров по локальной сети» содержит 50 страница текста, 9 рисунков, 6 листингов, 1 таблицу, 19 использованных источников, 7 приложений.

Данная работа посвящена разработке клиент-серверного приложения для удалённого мониторинга и менеджмента компьютеров по локальной сети. Приложение нацелено на использование в учебных заведениях. Позволяет удаленно с помощью главного компьютера, за которым должен сидеть преподаватель, производить мониторинг и менеджмент клиентских компьютеров, за которыми сидят обучающиеся. Под мониторингом понимается возможность наблюдения за тем, что происходит на экранах клиентских компьютеров, а также сбор информации о времени работы компьютера. Под менеджментом понимается возможность управления курсором мыши, включение и выключение монитора клиентских компьютеров.

*Ключевые слова*: локальная сеть, мониторинг, менеджмент, клиентское приложение, приложение администратора, логика взаимодействия.

*Предмет исследования*. Проектирование приложения в среде разработки Visual Studio для мониторинга и менеджмента компьютеров по локальной сети.

*Объект исследования*. Программирование приложений, работающих под управлением операционной системы Windows, в среде разработки Visual Studio.

*Цель работы*. Разработать приложение, для удаленного мониторинга и менеджмента компьютеров в локальной сети.

*Метод исследования.* При исследовании применялся теоретический анализ литературы и материалов сети Интернет, методы структурного и объктно-ориентированного программирования.

*Результаты работы*. В дипломном проекте разработано программное обеспечение для мониторинга и менеджмента компьютеров по локальной сети. Оно позволяет контролировать учебный процесс обучающихся во время их занятий за компьютерами.

*Работа имеет теоретическое и практическое значение*, т.к. разработанный программный продукт может использоваться в компьютерных аудиториях учебных заведениях для улучшения качества образовательного процесса.

# Abstract

The graduation project on the topic “Development of an application for monitoring and managing computers on a local network” contains 50 pages of text, 9 figures, 6 listings, 1 table, 19 sources used, 7 appendices.

This work is devoted to the development of a client – ​​server application for remote monitoring and management of computers on a local network. The application is intended for use in educational institutions. It allows you to remotely use the main computer, which the teacher should be sitting at, to monitor and manage the client computers that the students are sitting at. Monitoring refers to the possibility of monitoring what is happening on the screens of client computers, as well as collecting information about the time the computer is running and the frequency of the software launching. Management refers to the ability to control the mouse cursor, keyboard, and turn on and off the monitor of client computers.

*Keywords*: local network, monitoring, management, client application, administrator application, interaction logic.

*The subject of the study*. Designing an application in the Visual Studio development environment for monitoring and managing computers on a local network

*The object of study*. Programming applications running under the Windows operating system in the Visual Studio development environment.

*The purpose of the work*. Develop an application for remote monitoring and management of computers on a local network.

*Method of the study*. The study used a theoretical analysis of literature and materials on the Internet, methods of structural and object–oriented programming.

*The results of the work*. The graduation project developed software for monitoring and management of computers on a local network. It allows you to control the learning process of students during their classes at computers.

*The work has theoretical and practical importance*, because the developed software product can be used in computer audiences of educational institutions to improve the quality of the educational process.

Содержание

[Аннотация 5](#_Toc45049645)

[Abstract 7](#_Toc45049646)

[ВВЕДЕНИЕ 10](#_Toc45049647)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О ПРОЕКТИРОВАНИИ ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА И МЕНЕДЖМЕНТА КОМПЬЮТЕРОВ ПО ЛОКАЛЬНОЙ СЕТИ 14](#_Toc45049648)

[1.1. Теоретические сведения об инструментах разработки клиент-серверного приложения для мониторинга компьютеров по локальной сети 14](#_Toc45049649)

[1.2. Теоретические сведения о способах программирования клиент-серверных приложений на языке программирования C# c использованием библиотеки .NET Framework 15](#_Toc45049650)

[1.3. Описание логики взаимодействия приложения администратора с клиентским приложением 20](#_Toc45049651)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА КОМПЬЮТЕРОВ ПО ЛОКАЛЬНОЙ СЕТИ 33](#_Toc45049652)

[2.1. Проектирование интерфейса и программная реализация приложения администратора 33](#_Toc45049653)

[2.1.1. Проектирование интерфейса 33](#_Toc45049654)

[2.1.2. Программная реализация 36](#_Toc45049655)

[2.2. Проектирование интерфейса и программная реализация клиентского приложения 42](#_Toc45049656)

[ЗАКЛЮЧЕНИЕ 46](#_Toc45049657)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 48](#_Toc45049658)

[ПРИЛОЖЕНИЕ 51](#_Toc45049659)

# ВВЕДЕНИЕ

**Обоснование выбора темы и ее актуальность**

Проблема удаленного мониторинга и менеджмента, за каким-либо объектом без непосредственного физического контакта с ним, в век информационных и компьютерных технологий является очень актуальной. В частности, мониторинг и менеджмент работы обучающихся за компьютерами в учебных заведениях позволяет контролировать действия учеников, что в свою очередь положительно сказывается на образовательном процессе. Ученики, зная, что их действия, совершаемые во время работы за компьютером, видит преподаватель, с меньшей вероятностью будут отвлекаться от образовательного процесса.

Для решения этой проблемы необходимо спроектировать специальное компьютерное приложение, позволяющее удаленно с помощью главного компьютера, за которым должен сидеть преподаватель, производить мониторинг клиентских компьютеров, а также управлять клиентскими компьютерами. Под мониторингом понимается возможность наблюдения за тем, что происходит на экранах клиентских компьютеров. Под клиентскими компьютерами понимаются компьютеры, за которыми должны сидеть ученики. Программный продукт будет состоять из двух приложений. Приложение администратора, которое будет установлено на компьютере преподавателя и клиентское приложение, которое будет установлено на компьютеры обучающихся. При этом компьютер преподавателя должны быть связаны с клиентскими компьютерами проводной или беспроводной локальной сетью.

Таким образом, разрабатываемое приложение позволит:

1. Наблюдать за клиентскими компьютерами.

2. Транслировать изображения с экрана компьютера преподавателя на клиентские компьютеры.

3. Транслировать изображение экрана клиентского компьютера на экраны остальных клиентских компьютеров.

4. Собирать информацию о времени работы клиентских компьютеров.

5. Управлять курсором мыши клиентского компьютера.

6. Выключать и включать мониторы клиентских компьютеров.

**Степень разработанности проблемы**

В настоящее время уже имеется ряд программных решений, которые позволяют осуществлять мониторинг компьютеров, соединенных между собой локальной сетью. Были рассмотрены три специализированные программы для удаленного мониторинга компьютеров. Это Netop Vision, NetControl2 и Veyon. Программы Netop Vision и NetControl2 являются платными, Veyon бесплатным программным продуктом. Рассмотрим их подробнее.

Программа Netop Vision (http://www.netop.ru/) обладает широким спектром возможностей, например, просмотр экранов учащихся, трансляция экрана преподавателя, блокирование экрана, удалённое управление компьютером обучающегося, трансляция экрана одного из учащихся. Это основные возможности данного программного продукта. Цена версии на одного преподавателя и пятнадцать обучающихся составляет сорок три тысячи рублей.

Программа NetControl2 (http://www.netcontrol2.ru/) обладает схожими возможностями, наряду с программой Netop Vision, а также имеет такие возможности как, тестирование учеников, обмен файлами, текстовый чат. Стоимость версии на одного преподавателя и двадцать пользователей составляет семнадцать тысяч рублей.

И наконец приложение Veyon (https://veyon.io/). Данный программный продукт распространяется бесплатно, обладает схожими возможностями с предыдущими двумя приложениями, но имеет два существенных недостатка, а именно:

1. Отсутствует официальная документация на русском языке.
2. Сложная первоначальная настройка приложения.

Разрабатываемый в рамках выпускной квалификационной работы программный продукт будет распространяться бесплатно по лицензии freeware, будет прост в первоначальной настройке. Также в разрабатываемом программном продукте, будет реализована функция сбора информации о работе клиентских компьютеров. Будут собираться данные о частоте включения компьютера, продолжительность работы компьютера. Данной функцией не обладает ни одно приложение, которые были перечислены выше, что является отличительной чертой разрабатываемого программного продукта. Разработка приложения будет происходить на языке программирования C#, в среде разработки Visual Studio [1-3].

**Предмет исследования**

Проектирование приложения, в среде разработки Visual Studio, для мониторинга и менеджмента компьютеров по локальной сети.

**Объект исследования**

Программирование приложений, работающих под управлением операционной системы Windows, в среде разработки Visual Studio.

**Цель работы**

Разработать приложение, для удаленного мониторинга и менеджмента компьютеров в локальной сети.

**Основные задачи исследования**

1. Описать логику взаимодействия приложения администратора с клиентским приложением.
2. Спроектировать и реализовать интерфейс клиентского приложения, разработать программный код для управления интерфейсом и функциями клиентского приложения.
3. Спроектировать и реализовать интерфейс приложения администратора, разработать программный код для управления интерфейсом и функциями приложения администратора.

**Методы исследования**

При исследовании применялся теоретический анализ литературы и материалов сети Интернет, методы структурного и объектно-ориентированного программирования.

**Структура работы**

Работа состоит из введения, двух глав, заключения, списка источников и приложения.

Во введении рассматривается актуальность работы, ставится цель и формулируются задачи, необходимые для достижения поставленной цели.

Первая глава представляет собой совокупность теоретических данных об логике взаимодействия приложения администратора с клиентским приложением.

Во второй главе описывается программная реализация приложения для мониторинга и менеджмента компьютеров по локальной сети.

В заключении делаются выводы по проделанной работе и рассматриваются дальнейшие перспективы развития разработанного программного обеспечения.

В конце работы приводится список использованных источников.

В приложении приведен исходный код разработанного программного продукта.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О ПРОЕКТИРОВАНИИ ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА И МЕНЕДЖМЕНТА КОМПЬЮТЕРОВ ПО ЛОКАЛЬНОЙ СЕТИ

## Теоретические сведения об инструментах разработки клиент-серверного приложения для мониторинга компьютеров по локальной сети

Проектирование клиент-серверного приложения для мониторинга компьютеров по локальной сети будет реализовано с помощью языка программирования C# и программной платформы от компании Microsoft – .NET Framework.

C# является объектно-ориентированным языком программирования, относится к семье языков с C-подобным синтаксисом. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, исключения, комментарии в формате XML.

NET Framework – это программная платформа от компании Microsoft позволяющая выполнять разработку на различных языках программирования, так как для всех языков используется единая общеязыковая среда исполнения Common Language Runtime (CLR). Таким образом, основными возможностями платформы .NET являются:

1. Кроссплатформенность – .NET Framework поддерживается большинством современных операционных систем Windows, а также есть возможность создавать приложения поддерживаемые Linux системами, и даже мобильные приложения.
2. Мультиязычность – так как исходный код, написанный на используемом языке программирования, транслируется в общеязыковой Common Intermediate Language (CIL) код, появляется возможность вести разработку на любом поддерживаемом языке программирования, и даже использовать различные языки программирования в одном решении. Наиболее популярными поддерживаемыми языками являются C#, VB.NET, C++, F#.
3. Обширная библиотека классов и технологий – существует огромное количество готовых к использованию библиотек для решения необходимых задач. Зачастую совсем не приходится реализовывать низкоуровневую логику работы приложения, достаточно воспользоваться готовым решением, удобно поставляемым через менеджер пакетов nuget. Кроме того, платформа .NET предусматривает легкую разработку практически любого типа необходимого программного обеспечения.

## Теоретические сведения о способах программирования клиент-серверных приложений на языке программирования C# c использованием библиотеки .NET Framework

В разрабатываемом программном продукте, клиентское и серверное приложения будут соединены между собой с использованием протокола TCP.

TCP — это транспортный протокол, является частью стека протоколов TCP IP, он выполняет функции управления передачей данных и следит за их сохранностью. Расшифровывается как Transmission Control Protocol (протокол управления передачей). Это стандарт, который определяет, как нужно устанавливать связь и поддерживать ее, чтобы две программы могли обмениваться данными между собой.

Протокол TCP является надежным протоколом. Надёжность данного протокола передачи данных обусловлена следующими факторами:

1. Использует логическое соединение, благодаря чему обеспечивается надежная доставка данных.

2. Пронумеровывает передаваемые пакеты данных и проверяет их доставку, принимающая сторона высылает подтверждение о получении, в случае потери каких-либо пакетов создается повторная передача.

3. Проверяет контрольную сумму передаваемых пакетов, если она отличается — создается новая отправка.

4. Проверяет пакеты на дубликаты, в случае обнаружения таковых — уничтожает.

Соединение по TCP протоколу работает следующим образом:

1. Отправитель отсылает получателю специальный пакет, именуемый SYN, т.е. пригашает к соединению

2. Получатель отвечает уже пакетом SYN-ACK, т.е. соглашается

3. Отправитель отсылает спец. пакет ACK, т.е. подтверждает, что согласие получено.

Схему соединения с помощью протокола TCP можно увидеть на рисунке 1.

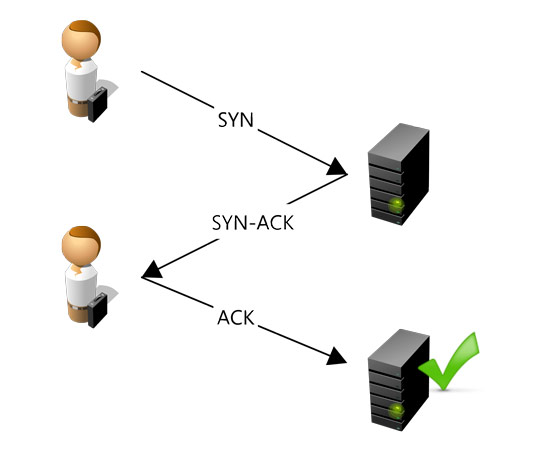


Рисунок 1. Схема соединения с помощью протокола TCP

На этом TCP-соединение успешно установлено, и получатель с отправителем могут спокойно обмениваться информацией. При передаче все пакеты данных нумеруются, отсылаются подтверждения о получении каждого из них, а потерянные пересылаются заново.

Для работы с протоколом TCP в .NET предназначены классы TcpClient и TcpListener. Эти классы строятся поверх класса System.Net.Sockets.Socket. Для создания клиентской программы, работающей по протоколу TCP, предназначен класс TcpClient. Для подключения к серверу TCP, в этом классе определен метод Connect(), которому передается название хоста и порт. Чтобы взаимодействовать с сервером TcpClient определяет метод GetStream(), который возвращает объект NetworkStream. Через данный объект можно передавать сообщения серверу или, наоборот, получать данные с сервера. После окончания работы с TcpClient его надо закрыть методом Close(). Для отправки данных у потока используется метод Write(). Для чтения используется метод Read() класса NetworkStream. Пример программного кода для клиента можно увидеть в листинге 1 [4][5].

|  |
| --- |
| using System;  using System.Net.Sockets;  using System.Text;  namespace TcpClientApp  {  class Program  {  private const int port = 8888;  private const string server = "127.0.0.1";  static void Main(string[] args)  {  try  {  TcpClient client = new TcpClient();  client.Connect(server, port);    byte[] data = new byte[256];  StringBuilder response = new StringBuilder();  NetworkStream stream = client.GetStream();  do  {  int bytes = stream.Read(data, 0, data.Length);  response.Append(Encoding.UTF8.GetString(data, 0, bytes));  }  while (stream.DataAvailable); // пока данные есть в потоке  Console.WriteLine(response.ToString());  // Закрываем потоки  stream.Close();  client.Close();  }  catch (SocketException e)  {  Console.WriteLine("SocketException: {0}", e);  }  catch (Exception e)  {  Console.WriteLine("Exception: {0}", e.Message);  }  Console.WriteLine("Запрос завершен...");  Console.Read();  }  }  } |

Листинг 1. Примр программного кода клиентского приложения

В данном примере предполагается, что по локальному адресу 127.0.0.1 на порту 8888 запущен TCP-сервер.

Мы ожидаем, что сервер будет передавать строку в кодировке UTF8, и для создания строки определяется объект StringBuilder.

Для считывания данных создаем массив байтов: byte[] data = new byte[256]. Однако поскольку данные могут иметь больший размер, то считываем их из потока в цикле do..while. При этом проверяется значение stream.DataAvailable на наличие данных в потоке.

## 1.3. Описание логики взаимодействия приложения администратора с клиентским приложением

Разрабатываемый программный продукт будет состоять из двух приложений: клиентского и приложения администратора. Связь между клиентским приложением и приложением администратора будет основано на принципе клиент-серверного взаимодействия, который подразумевает наличие приложения–сервера и приложения–клиента. В нашем случае, в качестве приложения–сервера будет выступать клиентская часть разрабатываемого программного продукта, так как приложение администратора будет отправлять запросы клиентскому приложению и получать ответ. Соответственно, в качестве приложения–клиента будет выступать приложение администратора.

Возможности данного приложения поделены на две группы – мониторинг и менеджмент.

Группа «Мониторинг» состоит из таких функций как:

1. Просмотр экранов обучающихся.
2. Трансляция экрана преподавателя.
3. Трансляция экрана обучающегося.
4. Сбор данных о времени работы учебных компьютеров.

Группа «Менеджмент» состоит из функций:

1. Управление курсором мыши обучающегося.
2. Включение/выключение монитора обучающегося.

Для того, чтобы клиентское приложение, понимало, какая команда поступила от приложения администратора, были придуманы сокращения для команд в виде строк, которые состоят строго из четырёх символов латинского алфавита. Так как один символ латинского алфавита имеет размер 1 байт, то при получении команды от приложения администратора, можно считать первые четыре байта и понять, какая команда поступила. В таблице 1 можно увидеть список команд, отправляемых приложением администратора на клиентские приложения.

Таблица 1 — Список команд, отправляемых приложением администратора на клиентские приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № | Группа | Команда | Расшифровка |
| 1. | Мониторинг | strm | Команда на отправку изображения экрана преподавателю. |
| 2. | Мониторинг | bron | Команда на получение изображения экрана преподавателя. |
| 3. | Мониторинг | brof | Команда на прекращение получения изображения экрана преподавателя. |
| 4. | Мониторинг | sttm | Команда на просмотр экрана обучающегося другими обучающимися и преподавателем. |
| 5. | Мониторинг | scmd | Команда на отправку информации о времени работы компьютера обучающегося на компьютер преподавателя. |
| 6. | Менеджмент | rwhm | Команда на отправку разрешения монитора, (ширины и высоты), на компьютер преподавателя. |
| 7. | Менеджмент | olbm | Команда на одно нажатие левой кнопки мыши. |
| 8. | Менеджмент | tlbm | Команда на двойное нажатие левой кнопки мыши. |
| 9. | Менеджмент | orbm | Команда на одно нажатие правой кнопки мыши. |
| 10. | Менеджмент | mlbm | Команда на перемещение курсора с зажатой левой кнопкой мыши. |
| 11. | Менеджмент | mnof | Команда на выключение монитора. |
| 12. | Менеджмент | mnon | Команда на включение монитора. |

Рассмотрим алгоритмы взаимодействия приложения администратора с клиентским приложением, то есть, какими командами и какими данными обмениваются эти два приложения в процессе работы. Так как передача команд и иной информации между компьютерами происходит по локальной сети, следовательно, вся передаваемая информация перед отправкой конвертируется в массив байт.

Рассмотрим алгоритмы работы функций, которые относятся к группе «Мониторинг».

Рассмотрим команду «strm». Это команда на отправку изображения экрана преподавателю. Она запускает функцию мониторинга клиентских компьютеров. То есть, преподаватель сможет видеть на своём компьютере всё что происходит на компьютерах обучающихся. Ученики, зная, что их действия под присмотром преподавателя, с меньшей вероятностью будут отвлекаться от учебного процесса, а если это и произойдёт, то преподаватель сразу увидит это на своём компьютере и сделает замечание. Рассмотрим алгоритм работы данной функции:

1. Приложение администратора отправляет строку «strm» клиентскому приложению.
2. Клиентское приложение, получив строку «strm», делает скриншот экрана.
3. Определяется размер скриншота экрана в байтах. Число, обозначающее размер изображения, имеет размер четыре байта.
4. Приложению администратора отправляется размер изображения и само изображение экрана обучающегося.
5. Приложение администратора, при получении ответа от клиентского приложения, считывает первые четыре байта, тем самым узнавая размер изобра­жения, которое ему предстоит получить. Далее, приложение администратора получает изображение.
6. После того, как изображение полностью получено и его размер совпадает с тем размером, который был отправлен клиентским приложением, полученное изображение выводится на монитор преподавателя, и отправка команды strm повторяется.

Схема работы функции мониторинга изображена на рисунке 2.



Рисунок 2 — Схема работы функции мониторинга

Рассмотрим команды «bron» и «brof». Отправка команды «bron» говорит клиентскому приложению о том, что нужно принять изображение экрана компьютера преподавателя. Команда «brof» говорит клиентскому приложению о том, что нужно прекратить отображать изображение экрана компьютера преподавателя. Данные команды относятся к функции трансляции экрана преподавателя. С помощью данной функции преподаватель может демонстрировать свою работу за компьютером ученикам, не прибегая к использованию проектора. Ученики видят действия преподавателя у себя на мониторах и тут же могут их повторять, что положительно сказывается на образовательном процессе. Рассмотрим алгоритм работы данной функции:

1. Приложение администратора делает скриншот экрана.
2. Далее определяется размер скриншота экрана.
3. Приложение администратора отправляет клиентскому приложению строку «bron», размер изображения и само изображение.
4. Клиентское приложение, получив данные от приложения администратора, считывает первые четыре байта которые являются командой «bron».
5. Далее, клиентское приложение считывает следующие четыре байта, которые являются размером принимаемого изображения.
6. После того, как размер принимаемого изображения стал известен, клиентским приложением принимается само изображение экрана компьютера преподавателя.
7. Как только изображение полностью получено, оно выводится на экране ученика и клиентское приложение отправляет приложению администратора строку «bron».
8. Приложение администратора, получив строку «bron», повторяет данный алгоритм снова до тех пор, пока преподаватель не выключит трансляцию, тем самым отправив клиентскому приложению команду «brof». После этого, отображение изображения экрана монитора преподавателя пропадает с экрана ученика.

Схема работы функции трансляции изображения компьютера преподавателя на клиентские компьютеры изображена на рисунке 3.

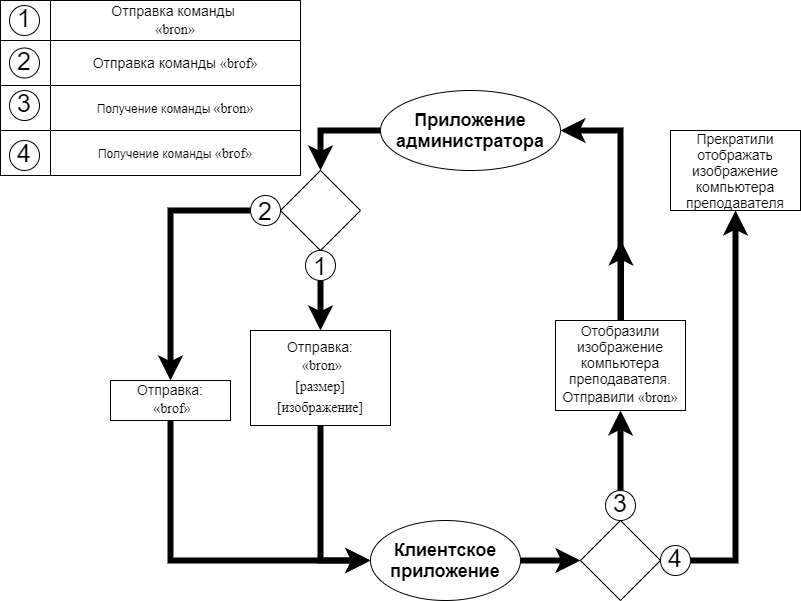


Рисунок 3 — Схема работы функции трансляции изображения компьютера преподавателя

Рассмотрим команду «sttm». Данная команда предназначена для трансляции экрана одного обучающегося на компьютеры остальных обучающихся, а также на компьютер преподавателя. Рассмотрим алгоритм работы данной функции.

1. Приложение администратора отправляет строку «sttm» клиентскому при­ложению.
2. Клиентское приложение, получив строку «sttm», делает скриншот экрана.
3. Определяет размер скриншота экрана в байтах.
4. Отправляет приложению администратора размер изображения и само изображение экрана обучающегося.
5. Приложение администратора, при получении ответа от клиентского приложения, считывает первые четыре байта, тем самым узнавая размер изобра­жения, которое ему предстоит получить. Далее, приложение администратора получает изображение.
6. После того, как изображение полностью получено и его размер совпадает с тем размером, который был отправлен клиентским приложением, полученное изображение выводится на монитор преподавателя.
7. Далее, на остальные клиентские компьютеры отправляется команда «bron», размер изображения и полученное в пункте 6 изображение.
8. Клиентские компьютеры, получив команду «bron», выводят полученное изображение на экран и отправляют приложению администратора строку «bron».
9. Приложение администратора, получив строку «bron», повторяет данный алгоритм снова до тех пор, пока преподаватель не выключит трансляцию, тем самым отправив клиентским приложениям команду «brof», после чего, отображение изображения экрана монитора одного ученика пропадает с экранов других учеников. Клиентскому компьютеру, изображение которого транслировалось всем остальным, будет прекращена отправка команды «sttm».

Схема работы функции трансляции изображения клиентского компьютера на остальные клиентские компьютеры изображена на рисунке 4.

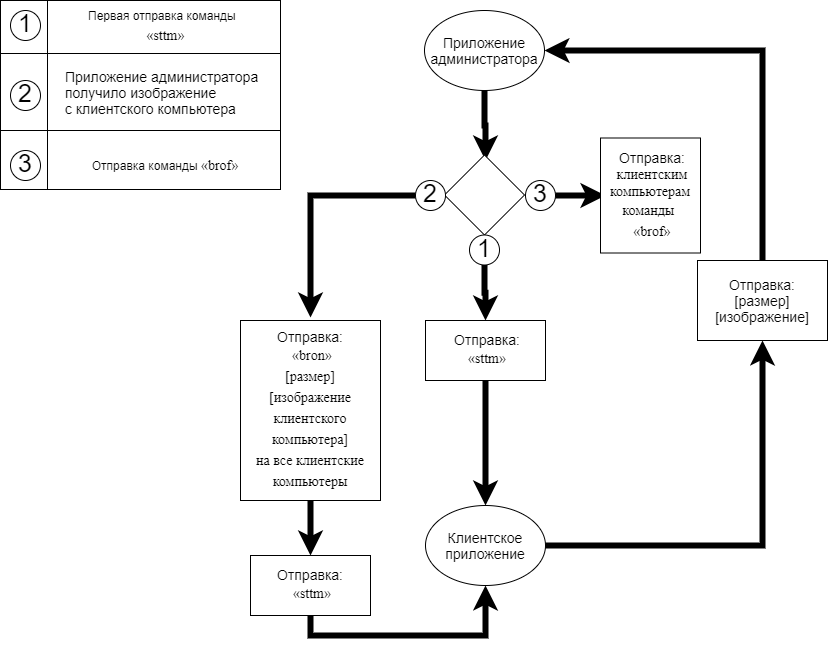


Рисунок 4 — Схема работы функции трансляции изображения клиентского компьютера на другие клиентские компьютеры

Рассмотрим команду «scmd». Команда «scmd» позволяет получить от клиентского компьютера информацию о времени его работы. Для хранения этих данных, на клиентских компьютерах была создана база данных для хранения информации о времени работы компьютера. В качестве СУБД была выбрана SQLite. Алгоритм выполнения команды «scmd».

1. Приложение администратора отправляет клиентскому приложению ко­манду «scmd», размер SQL запроса и сам SQL запрос к базе данных, например, «SELECT \* FROM tb\_time\_working».
2. Клиентское приложение, получив команду «scmd», считывает следующие четыре байта, тем самым узнавая размер SQL запроса, который будет предстоит принять.
3. Приняв SQL запрос от приложения администратора, клиентское приложение выполняет его и в ответ возвращает размер полученного результата и сам результат запроса в виде строки.
4. Приложение администратора, получив результат запроса, может его необходимым образом обработать и сформировать отчёт о времени работы клиентского компьютера.

Схема запроса информации о времени работы клиентского компьютера изображена на рисунке 5.



Рисунок 5 — Схема запроса информации о времени работы клиентского компьютера

Рассмотрим алгоритмы работы функций, которые относятся к группе «Менеджмент».

Рассмотрим команду «rwhm». Данная команда предназначена для запроса у клиентского компьютера ширины и высоты его экрана. Эти данные необходимы для сопоставления координат транслируемого изображения экрана компьютера ученика и реальных координат монитора ученика при клике кнопками мыши. Рассмотрим алгоритм получения разрешения экрана ученика:

1. Приложение администратора отправляет команду «rwhm».
2. Клиентское приложение, получив данную команду, определяет действительное разрешение экрана и отправляет эти данные приложению администратора.
3. Приложение администратора, получив разрешение монитора клиентского компьютера, сохраняет его в переменной для дальнейшего использования при отправке команд на управление кнопками мыши.

Рассмотрим команды управления кнопками мыши. «olbm» – команда на один клик левой кнопкой мыши, «tlbm» – команда на двойной клик левой кнопкой мыши, «orbm» – команда на один клик правой кнопкой мыши, «mlbm» – Команда на перемещение курсора с зажатой левой кнопкой мыши. Данные команды выполняются только в режиме мониторинга клиентского компьютера и имеют одинаковый алгоритм, разница лишь в количестве кликов или в кнопке мыши, а также при выполнении команды «mlbm» запоминаются координаты отпускания левой кнопки мыши. Рассмотрим алгоритм выполнения этих команд.

1. В приложении администратора запускается мониторинг клиентского компьютера и появляется изображение его экрана.
2. При клике мышью по транслируемому изображению вычисляются действительные координаты клика на клиентском компьютере. А в случае команды «mlbm» также ещё вычисляются действительные координаты отпускания левой кнопки мыши.
3. На клиентский компьютер отправляется одна из описываемых команд и действительные координаты клика.
4. Клиентский компьютер получив команду «olbm», «tlbm», «orbm» или «mlbm» считывает следующие четыре байта и получает координаты клика по оси OX. Далее считывает ещё четыре байта, тем самым получив коор­динату клика по оси OY. В случае команды «mlbm» считывание следующих четырёх байт происходит ещё два раза, для получения координат отпускания левой кнопки мыши.
5. После этого, по полученным клиентским приложением координатам происходит один клик левой кнопкой мыши, двойной клик левой кнопкой мыши, один клик правой кнопкой мыши или перемещение курсора с зажатой левой кнопкой мыши, в зависимости от отправленной команды. Для выполнения кликов использовался WinAPI.

Схема работы функции управление курсором мыши клиентских компьютеров изображена на рисунке 6.



Рисунок 6 — Схема работы функции управления курсором мыши

Рассмотрим оставшиеся две команды, которые относятся к группе функций «Менеджмент». Это команда «mnof» и команда «mnon», предназначенные для выключения и включения монитора клиентского компьютера. Данная функция полезна, когда необходимо отвлечь внимание учеников от мониторов компьютеров. Алгоритм выполнения данных команд следующий.

1. Приложение администратора отправляет команду «mnof» клиентскому приложению.
2. Клиентское приложение, получив команду «mnof» выключает монитор компьютера ученика с помощью WinAPI.
3. При отправке приложением администратора команды «mnon», монитор на клиентском компьютере снова включается, также с помощью WinAPI.

Схема работы функции выключения и включения монитора клиентского компьютера можно увидеть на рисунке 7.



Рисунок 7 — Схема работы функции выключения и включения монитора клиентского компьютера

Таким образом и осуществляется взаимодействие клиентского приложения, которое устанавливается на компьютерах обучающихся, и приложения администратора, которое устанавливается на компьютер преподавателя.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА КОМПЬЮТЕРОВ ПО ЛОКАЛЬНОЙ СЕТИ

## 2.1. Проектирование интерфейса и программная реализация приложения администратора

### 2.1.1. Проектирование интерфейса

Графический интерфейс будет сформирован с помощью технологии WPF. Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Преимущества WPF:

1. Использование традиционных языков .NET–платформы – C# и VB.NET для создания логики приложения.
2. Возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET.
3. Независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением.
4. Богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое.
5. Аппаратное ускорение графики – вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной [6–8].

Приложение администратора будет состоять из трёх окон, главное окно, окно добавления клиентских компьютеров и окно мониторинга отдельного клиентского компьютера.

При проектирование главного окна использовались следующие элементы управления:

1. Menu (панель меню).
2. ToolBar (панель инструментом).
3. TextBlock (элемент для вывода текстовой информации).
4. DataGrid (таблица).
5. WrapPanel (контейнер для элементов управления).
6. Image (контейнер для изображений).

Панель меню Menu позволит вызвать окно добавления клиентских компьютеров, производить сканирование сети на наличие включенных компьютеров, управлять функциями мониторинга и трансляции.

Панель инструментов ToolBar позволит управлять функциями мониторинга и трансляции, а также производить сканирование сети на наличие включенных компьютеров.

В элементе для вывода текста TextBlock будет отображаться выполненные преподавателем действия, а также количество включённых клиентских компьютеров.

В таблице DataGrid будет отображаться такая информация о клиентских компьютерах, как:

1. Имя компьютера
2. IP – адрес.
3. Включен или выключен.
4. Производиться мониторинг или нет.

В контейнере WrapPanel будут располагаться контейнеры для изображений типа Image. В Image будет отображаться миниатюрное изображения с экраном клиентских компьютеров, для которых включен мониторинг. Интерфейс главного окна приложения администратора можно увидеть на рисунке 8.

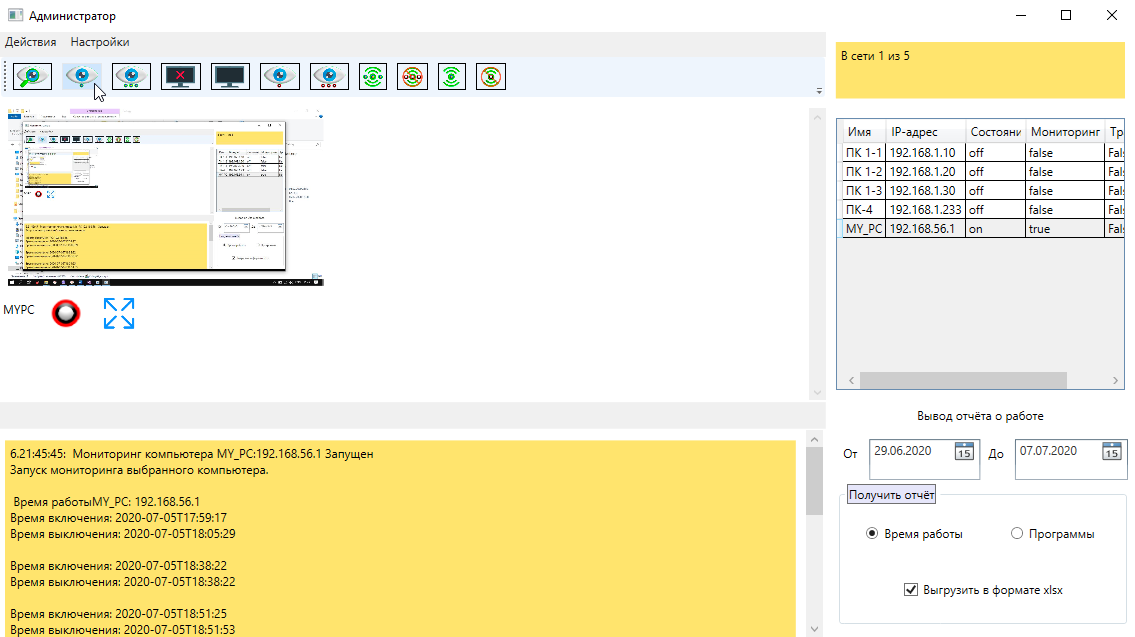


Рисунок 8 — Интерфейс главного окна приложения администратора

При проектировании окна добавления клиентских компьютеров использовались следующие элементы управления:

1. TextBox (текстовое поле).
2. DataGrid (таблица).
3. Button (кнопка).

В текстовые поля TextBox будут вводиться имя и ip–адрес клиентского компьютера, в таблицу DataGrid будет выводиться информация об уже имеющихся клиентских компьютерах, а также о добавленных. С помощью кнопок Button будет происходить добавление клиентского компьютера, управление сохранением и отменой изменений в таблице DataGrid, а также удаление клиентских компьютеров из списка. Интерфейс окна добавления клиентских компьютеров приложения администратора можно увидеть на рисунке 9.

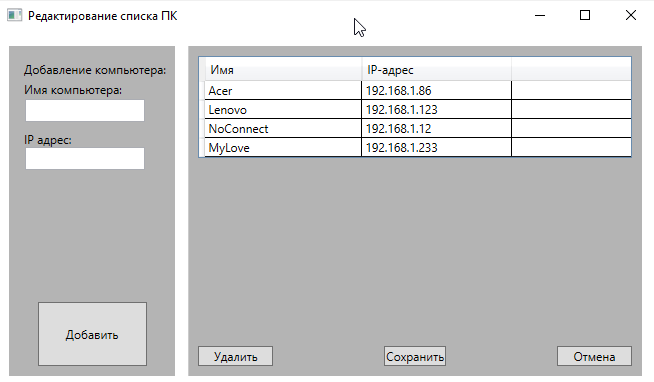


Рисунок 9 — Интерфейс окна добавления клиентских компьютеров

При проектировании окна мониторинга отдельного клиентского компьютера использовался только один элемент интерфейса – это контейнер для изображения Image, в который будет выводиться изображение клиентского компьютера в увеличенном размере.

### 2.1.2. Программная реализация

Программная реализация приложения администратора была выполнена с использование методологии объектно–ориентированного программирования на языке программирования C#. Данный подход позволяет создавать расширяемые системы, легко их модернизировать и в общем упрощает разработку программного обеспечения.

В основе программной разработки приложения администратора лежит класс ClientPC. Он состоит из следующих полей и методов:

1. namePC – имя клиентского компьютера.
2. ipPC – ip–адрес клиентского компьютера.
3. state – состояние (включен/выключен).
4. flagBroadCast – флаг трансляции экрана преподавателя (включена/выключена).
5. flagMonitoring – флаг мониторинга (включен/выключен).

И следующих методов:

1. ScreenShow – метод передачи команд клиентскому компьютеру.
2. PaintScreenshotToImage – метод отображения, полученного приложением администратора, изображения.
3. StopMonitoring – метод остановки мониторинга.
4. ScreenShot – метод создания скриншота экрана компьютера администратора.
5. CoorMonXY – метод сопоставления координат изображения клиентского компьютера и реальных координат дисплея клиентского компьютера.
6. MouseButtonClickToImageRetransToMonitor – метод определяет, какой клик произошёл по изображению, которое получает приложение администратора в процессе мониторинга клиентского компьютера.

Рассмотрим перечисленный список методов подробнее. Метод ScreenShow является рекурсивным. В качестве параметра принимает команду для клиентского компьютера, а также координаты клика курсора мыши. Координаты клика курсора мыши являются необязательными параметрами. Возможные команды были рассмотрены в параграфе 1.2. Внутри метода происходит соединение с клиентским компьютером и отправляется одна из команд, которая передаётся данному метода в качестве параметра. После отправки команды, приложение администратора дожидается ответа от клиентского приложения и в зависимости от ответа, происходит или повторный вызов данного метода или выход из метода. Программный код данного метода можно посмотреть в приложении 1 [9–10].

Метод PaintScreenshotToImage. Данный метод предназначен для конвертирования потока байт в изображение и помещения данного изображения в компонент Image. Принимает два аргумента. Первый аргумент – массив байт. Второй аргумент – компонент Image, в который будет помещаться изображение. Данный метод вызывается внутри метода ScreenShow. Программный код метода PaintScreenshotToImage можно увидеть в листинге 2 [11].

public void PaintScreenshotToImage(MemoryStream

memoryStreamScreenshot, System.Windows.Controls.Image image)

{

BitmapImage i = new BitmapImage();

memoryStreamScreenshot.Position = 0;

i.BeginInit();

i.CacheOption = BitmapCacheOption.OnLoad;

i.StreamSource = memoryStreamScreenshot;

i.EndInit();

image.Source = i;

memoryStreamScreenshot.Dispose();

}

Листинг 2. Программный код метода PaintScreenshotToImage

Метод StopMonitoring. Данный метод предназначен для удаления компонента Image, в котором отображается изображение с клиентского компьютера при включении мониторинга, из контейнера WrapPanel. Данный метод в качестве аргумента принимает контейнер WrapPanel, в котором находится компонент Image. Метод StopMonitoring вызывается внутри метода ScreenShow. Программный код данного метода можно увидеть в листинге 3.

public void StopMonitoring(WrapPanel wrapPanelImage) {

if (flagMonitoring==true)

{

wrapPanelImage.Children.Remove(stMain);

flagMonitoring = false;

stringFlag = "false";

countLog++;

mainWindow.tbLog.Text+= countLog+"."+ DateTime.Now.ToString("HH:mm:ss")+": Мониторинг компьютера " + namePC+":"+ipPC + " остановлен\n";

}

mainWindow.dataGridListPC.Items.Refresh();

}

Листинг 3. Программный код метода StopMonitoring

Метод ScreenShot. Данный метод предназначен для создания скриншота экрана. В качестве возвращаемого значения – поток байт. Данный метод используется внутри метода ScreenShow при запуске трансляции экрана компьютера преподавателя. Программный код данного метода можно увидеть в листинге 4 [12].

public MemoryStream ScreenShot()

{

MemoryStream memoryStream = new MemoryStream

Bitmap imageBitap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height

Graphics graphics = Graphics.FromImage(imageBitap as System.Drawing.Image);

graphics.CopyFromScreen(0, 0, 0, 0, imageBitap.Size

imageBitap.Save(memoryStream, System.Drawing.Imaging.ImageFormat.Jpeg);

graphics.Dispose();

imageBitap.Dispose();

return memoryStream;

}

Листинг 4. Программный код метода ScreenShot

Метод CoorMonXY. Данный метод предназначен для сопоставления координат изображения клиентского компьютера и реальных координат экрана клиентского компьютера. Делать это необходимо при управлении курсора мыши клиентского компьютера, так как разрешение изображения, полученного приложением администратора и реальное разрешение экрана клиентского компьютера различны. Перед вызовом данного метода, на клиентский компьютер отправляется запрос на разрешение его экрана. Программный код данного метода можно увидеть в Листинге 5.

[coorMonX,coorMonY,xImage,yImage,widthDifference,heightDifference], где:

private Dictionary<string, string> CoorMonXY(System.Windows.Controls.Image image, MouseButtonEventArgs e)

{

Dictionary<string, string> massCoorMonXY;

int xImage, yImage;

xImage = Convert.ToInt32(e.GetPosition(image).X);

yImage = Convert.ToInt32(e.GetPosition(image).Y);

double widthDifference = 0, heightDifference = 0;

widthDifference = widthDisplay / image.ActualWidth;

heightDifference = heightDisplay / image.ActualHeight;

int coorMonX, coorMonY;

coorMonX = Convert.ToInt32(xImage \* widthDifference);

coorMonY = Convert.ToInt32(yImage \* heightDifference);

massCoorMonXY = new Dictionary<string, string> {

{"coorMonX",coorMonX.ToString()},

{"coorMonY", coorMonY.ToString()},

{"xImage", xImage.ToString()},

{"yImage", yImage.ToString()},

{"widthDifference", widthDifference.ToString()},

{"heightDifference", heightDifference.ToString()}

};

return massCoorMonXY;

}

Листинг 5. Программный код метода CoorMonXY

Метод MouseButtonClickToImageRetransToMonitor вызывается в момент клика мышью по изображению клиентского компьютера в процессе мониторинга и опредеяет какой клик произошёл: один клик левой кнопкой мыши, двойной клик левой кнопкой мыши, один клик правой кнопкой мыши или перетаскивание курсора мыши при зажатой левой кнопки мыши. В зависимости от того, какой клик произошёл, на клиентский компьютер отправляется соответствующая команда. Программный код данного метода можно увидеть в приложении 4.

## 2.2. Проектирование интерфейса и программная реализация клиентского приложения

Клиентская часть приложения будет состоять из одного окна, на котором будут располагаться текстовая метка с ip–адресом компьютера и компонент Image, который будет отображаться только при включении трансляции экрана компьютера преподавателя и исчезать при выключении трансляции.

В основе программной разработки клиентского приложения находятся следующие методы:

1. Run – метод принятий и выполнения команд от приложения администратора.
2. ScreenShot – метод создания скриншота экрана компьютера.
3. PaintScreenshotToImage – метод отображения изображения.
4. LeftMouseClick, RightMouseClick, LeftMouseClickAndMove – методы эмуляции нажатия кнопок мыши.
5. TurnScreenOff, TurnScreenOn – методы включения и выключения монитора.
6. CreateDatabase, addOn, addOff – методы создания базы данных для хранения информации о работе клиентского компьютера и добавления записей в базу данных.

Рассмотрим данные методы подробнее. Метод Run осуществляет запуск TCP сервера, приём команд от приложения администратора и отправку ответов приложению администратора. В качестве аргументов принимает ip–адрес компьютера и порт. Программный код данного метода можно увидеть в приложении 5.

Метод ScreenShot. Данный метод предназначен для создания скриншота экрана. В качестве возвращаемого значения – поток байт. Данный метод используется внутри метода Run при получении команды от приложения администратора на запуск мониторинга. Программный код данного метода можно увидеть в приложении 3.

Метод PaintScreenshotToImage. Данный метод предназначен для конвертирования потока байт в изображение. После чего, полученное изображение помещается в компонент Image. Принимает два аргумента. Первый аргумент – поток байт из которого будет восстанавливаться изображение. Второй аргумент – компонент Image, в который будет помещаться изображение. Данный метод вызывается внутри метода Run при получении команды на запуск трансляции экрана компьютера преподавателя. Программный код метода PaintScreenshotToImage можно увидеть в приложении 2.

Методы LeftMouseClick, RightMouseClick, LeftMouseClickAndMove. Метод LeftMouseClick позволяет эмулировать нажатие левой кнопки мыши. Если вызвать этот метод два раза подряд, то произойдёт двойной клик левой кнопки мыши. Метод RightMouseClick позволяет эмулировать нажатие правой кнопки мыши. Метод LeftMouseClickAndMove позволяет эмулировать перемещение курсора мыши с зажатой левой кнопкой. Эмуляция нажатий кнопок мыши было реализовано с помощью WinApi. Программный код данных методов можно увидеть в листинге 6 [13–14].

[System.Runtime.InteropServices.DllImport("user32.dll")]

static extern bool SetCursorPos(int x, int y);

[System.Runtime.InteropServices.DllImport("user32.dll")]

public static extern void mouse\_event(int dwFlags, int dx, int dy, int cButtons, int dwExtraInfo);

public const int MOUSEEVENTF\_LEFTDOWN = 0x0002;

public const int MOUSEEVENTF\_LEFTUP = 0x0004;

public const int MOUSEEVENTF\_RIGHTDOWN = 0x0008;

public const int MOUSEEVENTF\_RIGHTUP = 0x0010;

public static void LeftMouseClick(int xpos, int ypos)

{

SetCursorPos(xpos, ypos);

mouse\_event(MOUSEEVENTF\_LEFTDOWN, xpos, ypos, 0, 0);

mouse\_event(MOUSEEVENTF\_LEFTUP, xpos, ypos, 0, 0);

}

public static void RightMouseClick(int xpos, int ypos)

{

SetCursorPos(xpos, ypos);

mouse\_event(MOUSEEVENTF\_RIGHTDOWN, xpos, ypos, 0, 0);

mouse\_event(MOUSEEVENTF\_RIGHTUP, xpos, ypos, 0, 0);

}

public static void LeftMouseClickAndMove(int xpos1, int ypos1, int xpos2, int ypos2)

{

SetCursorPos(xpos1, ypos1);

mouse\_event(MOUSEEVENTF\_LEFTDOWN, xpos1, ypos1, 0, 0);

SetCursorPos(xpos2, ypos2);

Thread.Sleep(500);

mouse\_event(MOUSEEVENTF\_LEFTUP, xpos2, ypos2, 0, 0);

}

Листинг 6. Программный код методов LeftMouseClick, RightMouseClick, LeftMouseClickAndMove

Метода TurnScreenOff, TurnScreenOn. Данные метода позволяют включать и выключать монитор компьютера. Включение и выключение монитора было реализовано с помощью WinApi. Программный код данных методов можно увидеть в приложении 6 [15–17].

Методы CreateDatabase, addOn, addOff. Метод CreateDatabase позволяет создать базу данных для хранения информации о работе компьютера, если она не создана. Метод addOn позволяет добавить запись в базу данных о включении компьютера. Метод addOff позволяет добавить запись в базу данных о выключении компьютера. Метод addOn вызывается при запуске клиентского приложения, а метод addOff вызывается каждые десять секунд при работе клиентского приложения, тем самым постоянно обновляя данные о времени выключения клиентского компьютера. Постоянное обновление времени выключения компьютера позволяет сохранить время выключения компьютера даже при внезапном исчезновении электричества и выключении компьютера. Программный код данных методов можно увидеть в приложении 7 [18–19].

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были решены следующие задачи:

1. Была описана логика взаимодействия приложения администратора с клиентским приложением. Разработана система команд для выполнения следующих функций: мониторинг клиентских компьютеров, трансляция экрана преподавателя, управление курсором мыши клиентских компьютеров, включение и выключение мониторов клиентских компьютеров, запрос данных о времени работы клиентских компьютеров.
2. С помощью технологии WPF и специального языка разметки XAML, основанном на xml, был спроектирован интерфейс клиентского приложения. В программном коде клиентского приложения были описаны следующие методы: метод принятия и выполнения команд от приложения администратора, метод создания скриншота экрана компьютера, метод отображения изображения из массива байт, методы эмуляции нажатия кнопок мыши, методы включения и выключения монитора, методы создания базы данных для хранения информации о работе клиентского компьютера и добавления записей в базу данных.
3. С помощью технологии WPF и специального языка разметки XAML, основанном на xml, был спроектирован интерфейс Приложения администратора. В программном коде приложения администратора были описаны следующие методы: метод передачи команд клиентскому компьютеру, метод отображения, полученного приложением администратора, изображения, метод остановки мониторинга, метод создания скриншота экрана компьютера администратора, метод сопоставления координат изображения клиентского компьютера и реальных координат дисплея клиентского компьютера.

В итоге разработан программный продукт, позволяющий производить мониторинг клиентских компьютеров, транслировать изображения с компьютера, на котором установлено приложение администратора, на клиентские компьютеры, собирать информацию о времени работы клиентских компьютеров, управлять курсором мыши клиентских компьютеров, выключать и включать мониторы клиентских компьютеров. Данный программный продукт состоит из двух приложений, которые устанавливаются на ученический компьютер и компьютер преподавателя.

Таким образом было разработано клиент-серверное приложение, для удаленного мониторинга и менеджмента компьютеров в локальной сети. Данный программный продукт может использоваться в компьютерных аудиториях учебных заведений для улучшения качества образовательного процесса и контроля работы учеников за компьютерами.

По результатам работы подготовлена статья «Алгоритм для программы управления компьютером по локальной сети» в Вестник науки физико-математического факультета Орловского государственного университета имени И.С. Тургенева.

В дальнейшем планируется развивать данный программный продукт и добавить такие функции, как:

1. Управление клавиатурой клиентского компьютера.
2. Сбор данных о работе программного обеспечения, установленного на клиентских компьютерах.
3. Обмен файлами между компьютерами.
4. Управление компьютерами с помощью командной строки.
5. Запись экрана компьютеров.
6. Выключение и перезагрузка компьютеров по таймеру.
7. Аннотирование экрана.
8. Удалённый запуск программ на всех или выбранных компьютерах.
9. Ограничение доступа в интернет.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Veyon [Электронный ресурс] Режим доступа: <https://veyon.io/> – дата доступа: 20.02.2020.
2. Netop Vision [Электронный ресурс] Режим доступа: http://www.netop.ru/– дата доступа: 20.02.2020.
3. Net Control 2 [Электронный ресурс] Режим доступа: http://www.netcontrol2.ru/ – дата доступа: 20.02.2020.
4. ProtocolType Перечисление [Электронный ресурс] Режим доступа: https://docs.microsoft.com/ru–ru/dotnet/api/system.net.sockets.protocoltype – дата доступа: 20.04.2020.
5. Socket Класс [Электронный ресурс] Режим доступа: https://docs.microsoft.com/ru–ru/dotnet/api/system.net.sockets.socket – дата доступа: 20.04.2020.
6. Назаркин О.А. Разработка графического пользовательского интерфейса в соответствии с паттерном Model–View–Viewmodel на платформе Windows Presentation Foundation. Основные средства WPF. Учебное пособие по дисциплине «Проектирование человеко–машинного интерфейса» [Электронный ресурс]: учебное пособие / О.А. Назаркин – Электрон. текстовые данные – Липецк.: Липецкий государственный технический университет, ЭБС АСВ, 2014. – 61 c. – 978–5–88247–679–2. – Режим доступа: https://clck.ru/PWXmA.
7. Введение в WPF: [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/1.php. – Дата доступа: 01.04.2020.
8. Казанский А.А. Объектно–ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум / А.А. Казанский. – Электрон. текстовые данные – М.: Московский государственный строительный университет, ЭБС АСВ, 2011. – 180 c. – 2227–8397. – Режим доступа: <http://www.iprbookshop.ru/16368.html>
9. Биллиг В.A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс]: учебное пособие / В.A. Биллиг. – Электрон. текстовые данные. – М.: Интернет–Университет Информационных Технологий (ИНТУИТ), 2017. – 583 c. – 978–5–4487–0145–0. – Режим доступа: http://www.iprbookshop.ru/72339.html
10. Нужный А.М. Разработка приложений на C# в среде Visual Studio [Электронный ресурс]: учебное пособие / А.М. Нужный, Н.И.Гребенникова, В.Ф. Барабанов, О.Б. Кремер. – Электрон. текстовые данные. – Воронеж.: Воронежский государственный архитектурно–строительный университет, ЭБС АСВ, 2019. – 89 c. – 978–5–7731–0776–7. – Режим доступа: <http://www.iprbookshop.ru/93286.html>
11. C# конвертировать Image в byte[] и обратно из byte[] в Image: [Электронный ресурс]. – Режим доступа: https://clck.ru/PHfCS. – Дата доступа: 11.4.2020.
12. Как делать скриншоты экрана на C# [Электронный ресурс]. – Режим доступа: https://clck.ru/PHhfT. – Дата доступа: 11.4.2020.
13. Автоматизация, управления мышью программно: [Электронный ресурс]. – Режим доступа: https://clck.ru/PHfZh. – Дата доступа: 20.4.2020.
14. Безруков В.А. Win32 API. Программирование. [Электронный ресурс]: учебное пособие / В.А. Безруков – Электрон. текстовые данные. – CПБ.: Университет ИТМО, 2009. – 96 c. – 2227–8397. – Режим доступа: <http://www.iprbookshop.ru/65752.html>
15. Win Api – Программно отключить монитор: [Электронный ресурс]. – Режим доступа: https://clck.ru/PHfnA. – Дата доступа: 20.4.2020.
16. Безруков В.А. Win32 API. Программирование. [Электронный ресурс]: учебное пособие / В.А. Безруков – Электрон. текстовые данные. – CПБ.: Университет ИТМО, 2009. – 96 c. – 2227–8397. – Режим доступа: <http://www.iprbookshop.ru/65752.html>
17. C# Глазами хакера. Урок 2. Использование WinAPI: [Электронный ресурс]. – Режим доступа: https://clck.ru/PHf48. – Дата доступа: 01.04.2020.
18. SQLite Учебное пособие: [Электронный ресурс]. – Режим доступа: https://www.w3big.com/ru/sqlite/default.html. – Дата доступа: 01.04.2020.
19. SQLite + C#: [Электронный ресурс]. – Режим доступа: https://devpractice.ru/sqlite–c/. – Дата доступа: 01.4.2020.

# ПРИЛОЖЕНИЕ

Приложение 1

Программный код метода ScreenShow

public async void screenShow(string commandToClient = "strm",int X1=0,int Y1=0, int X2 = 0, int Y2 = 0) {

byte[] data = Encoding.UTF8.GetBytes(commandToClient);

var tcpEndPoint = new IPEndPoint(IPAddress.Parse(ipPC), port);

var tcpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

MemoryStream memoryStream = new MemoryStream();

var bufferSizeImage = new byte[4];

int s = 0;

await Task.Run(() => {

try

{

tcpSocket.Connect(tcpEndPoint);

if (commandToClient == "scmd")

{

StringBuilder stringAnswer = new StringBuilder();

byte[] size = new byte[4];

byte[] byteAnswerSql;

byte[] sqlCmdByte = Encoding.UTF8.GetBytes(sqlCmd);

byte[] answer = new byte[data.Length + sqlCmdByte.Length + 4];

data.CopyTo(answer, 0); BitConverter.GetBytes(sqlCmdByte.Length).CopyTo(answer, 4);

sqlCmdByte.CopyTo(answer, 8);

tcpSocket.Send(answer);

tcpSocket.Receive(size, 0, 4, SocketFlags.None);

byteAnswerSql = new byte[BitConverter.ToInt32(size, 0)];

tcpSocket.Receive(byteAnswerSql, 0, byteAnswerSql.Length, SocketFlags.None);

stringAnswer.Append(Encoding.UTF8.GetString(byteAnswerSql, 0, byteAnswerSql.Length));

mainWindow.tbLog.Dispatcher.Invoke(() => {

char[] b = stringAnswer.ToString().ToCharArray();

int c = 0;

string l = "";

string[] words = stringAnswer.ToString().Split(new char[] {'#'});

string txt = "";

for (int i = 1; i < words.Length; i++)

{

string[] w = words[i].Split(new char[] {';'});

for (int j = 1; j < w.Length; j++)

{

if (j == 1)

txt += "Время включения: " + w[j] + "\n";

if (j == 2)

txt += "Время выключения: " + w[j] + "\n\n";

}

}

mainWindow.tbLog.Text = "\n Время работы" + namePC + ": " + ipPC + "\n" + txt + mainWindow.tbLog.Text;

});

Console.WriteLine(stringAnswer.ToString());

}

if (commandToClient == "rwhm") {

byte[] answer = new byte[8];

tcpSocket.Send(data);

tcpSocket.Receive(answer, 0, 8, SocketFlags.None);

widthDisplay = BitConverter.ToInt32(answer, 0);

heightDisplay = BitConverter.ToInt32(answer, 4);

System.Windows.MessageBox.Show(widthDisplay.ToString() );

}

if ((commandToClient == "olbm") || (commandToClient == "tlbm") || (commandToClient == "orbm")|| (commandToClient == "mlbm")) {

if (commandToClient == "mlbm") {

byte[] answer = new byte[20];

data.CopyTo(answer,0);

BitConverter.GetBytes(X1).CopyTo(answer, 4);

BitConverter.GetBytes(Y1).CopyTo(answer, 8);

BitConverter.GetBytes(X2).CopyTo(answer, 12);

BitConverter.GetBytes(Y2).CopyTo(answer, 16);

tcpSocket.Send(answer);

}

else {

byte[] answer = new byte[12];

data.CopyTo(answer, 0);

BitConverter.GetBytes(X1).CopyTo(answer, 4);

BitConverter.GetBytes(Y1).CopyTo(answer, 8);

tcpSocket.Send(answer);

}

}

if ((commandToClient == "bron")|| (commandToClient == "brof"))

{

byte[] answer = new byte[4];

StringBuilder stringAnswer = new StringBuilder();

if (commandToClient == "bron") {

byte[] bImage = ScreenShot().ToArray();

byte[] commandSizeImageByteImage = new byte[bImage.Length + 8];

data.CopyTo(commandSizeImageByteImage, 0);

BitConverter.GetBytes(bImage.Length).CopyTo(commandSizeImageByteImage, 4);

bImage.CopyTo(commandSizeImageByteImage, 8);

tcpSocket.Send(commandSizeImageByteImage);

tcpSocket.Receive(answer, 0, 4, SocketFlags.None);

stringAnswer.Append(Encoding.UTF8.GetString(answer, 0, 4));

Console.WriteLine("Ответ клиента на " + commandToClient + ": " + stringAnswer);

tcpSocket.Shutdown(SocketShutdown.Both);

tcpSocket.Close();

}

if (commandToClient == "brof") {

tcpSocket.Send(data);

tcpSocket.Receive(answer, 0, 4, SocketFlags.None);

stringAnswer.Append(Encoding.UTF8.GetString(answer, 0, 4));

Console.WriteLine("Ответ клиента на " + commandToClient + ": " + stringAnswer); tcpSocket.Shutdown(SocketShutdown.Both);

tcpSocket.Close();

}

if (Equals(stringAnswer.ToString(), "bron") && flagBroadCast == true)

{

screenShow("bron");

}

}

if (flagMonitoring && commandToClient == "strm")

{

tcpSocket.Send(data);

tcpSocket.Receive(bufferSizeImage, 0, 4, SocketFlags.None);

int sizeImage = BitConverter.ToInt32(bufferSizeImage, 0);

var buffersClientScreenShot = new byte[sizeImage];

for (int j = 0; j < sizeImage; j++)

{

s += tcpSocket.Receive(buffersClientScreenShot, s, 1, SocketFlags.None);

memoryStream.WriteByte(buffersClientScreenShot[j]);

}

tcpSocket.Shutdown(SocketShutdown.Both);

tcpSocket.Close();

image.Dispatcher.Invoke(() => {

PaintScreenshotToImage(memoryStream, image);

screenShow();

});

}

}

catch (SocketException e) {

countLog++;

if (flagFullScreen) {

System.Windows.MessageBox.Show("Компьютер " + namePC + ":" + ipPC + ".\nСоединение разорвано. Удалённый компьютер был выключен или была закрыта клиентская программа !");

mainWindow.tbLog.Text=countLog+"."+ DateTime.Now.ToString("HH:mm:ss")+": "+ DateTime.Now.ToString("HH:mm:ss")+": Компьютер " + namePC + ":" + ipPC + ".\nСоединение разорвано.Удалённый компьютер был выключен или была закрыта клиентская программа !\n"+mainWindow.tbLog.Text;

}else

image.Dispatcher.Invoke(()=> {

mainWindow.tbLog.Text = countLog + "." + DateTime.Now.ToString("HH:mm:ss") + ": Компьютер " + namePC + ":" + ipPC + ".\nСоединение разорвано. Удалённый компьютер был выключен или была закрыта клиентская программа !\n" + mainWindow.tbLog.Text;

if (flagFullScreen == false)

StopMonitoring(mainWindow.wrapPanelImage);

if (flagFullScreen == true)

{

flagFullScreen = false;

StopMonitoring(fullScreenPC.wrapPanelImage);

fullScreenPC.Close();

Приложение 2

Программный код метода PaintScreenshotToImage

public void PaintScreenshotToImage(MemoryStream

memoryStreamScreenshot, System.Windows.Controls.Image image)

{

BitmapImage i = new BitmapImage();

memoryStreamScreenshot.Position = 0;

i.BeginInit();

i.CacheOption = BitmapCacheOption.OnLoad;

i.StreamSource = memoryStreamScreenshot;

i.EndInit();

image.Source = i;

memoryStreamScreenshot.Dispose();

}

Приложение 3

Программный код метода ScreenShot

public MemoryStream ScreenShot()

{

MemoryStream memoryStream = new MemoryStream

Bitmap imageBitap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height

Graphics graphics = Graphics.FromImage(imageBitap as System.Drawing.Image);

graphics.CopyFromScreen(0, 0, 0, 0, imageBitap.Size

imageBitap.Save(memoryStream, System.Drawing.Imaging.ImageFormat.Jpeg);

graphics.Dispose();

imageBitap.Dispose();

return memoryStream;

}

Приложение 4

Программный код метода MouseButtonClickToImageRetransToMonitor

void MouseButtonClickToImageRetransToMonitor(MouseButtonEventArgs e)

{

Dictionary<string, string> coorMonXY = CoorMonXY(image, e);

if (e.LeftButton == MouseButtonState.Pressed)

{

corMonX1 = Convert.ToInt32(coorMonXY["coorMonX"]);

corMonY1 = Convert.ToInt32(coorMonXY["coorMonY"]);

if (e.ClickCount == 2)

{

fullScreenPC.Title = "Произошёл двойной клик ЛКМ";

screenShow("tlbm", corMonX1, corMonY1);

flag = 1;

corMonX1 = corMonX2 = corMonY1 = corMonY2 = 0;

}

}

if (e.LeftButton == MouseButtonState.Released)

{

corMonX2 = Convert.ToInt32(coorMonXY["coorMonX"]);

corMonY2 = Convert.ToInt32(coorMonXY["coorMonY"]);

if ((corMonX2 != corMonX1) || (corMonY1 != corMonY2))

{

fullScreenPC.Title = "Произошло перемещение";

screenShow("mlbm",corMonX1, corMonY1, corMonX2, corMonY2);

corMonX1 = corMonX2 = corMonY1 = corMonY2 = 0;

}

if ((corMonX2 == corMonX1) && (corMonY1 == corMonY2) && (flag == 0))

{

fullScreenPC.Title = "Произошёл один клик ЛКМ";

screenShow("olbm",corMonX2, corMonY2);

corMonX1 = corMonX2 = corMonY1 = corMonY2 = 0;

}

flag = 0;

Приложение 5

Программный код метода Run

async void run(string ip,int port) {

var tcpEndPoint = new IPEndPoint(IPAddress.Parse(ip), port);

try

{

tcpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

}

catch(SocketException e) {

}

tcpSocket.Bind(tcpEndPoint);

tcpSocket.Listen(3);

var i = 0;

await Task.Run(()=>

{

while (true)

{

try

{

listener = tcpSocket.Accept();

byte[] buffer = new byte[256];

int size = 0;

StringBuilder dataCommantStr = new StringBuilder();

byte[] dataCommand = new byte[4];

listener.Receive(dataCommand, 0, 4, SocketFlags.None);

dataCommantStr.Append(Encoding.UTF8.GetString(dataCommand, 0, 4));

Console.WriteLine("Приход " + dataCommantStr.ToString());

if (Equals(dataCommantStr.ToString(), "strm"))

{

byte[] bImage = ScreenShot().ToArray();

byte[] bByteImageAndSizeImage = new byte[bImage.Length + 4];

BitConverter.GetBytes(bImage.Length).CopyTo(bByteImageAndSizeImage, 0);

bImage.CopyTo(bByteImageAndSizeImage, 4);

Console.WriteLine(bByteImageAndSizeImage[0]); Console.WriteLine(bByteImageAndSizeImage[1]);

Console.WriteLine(bByteImageAndSizeImage[2]); Console.WriteLine(bByteImageAndSizeImage[3]);

Console.WriteLine();

listener.Send(bByteImageAndSizeImage);

Dispatcher.Invoke(() =>

{

i += 1;

labelCountScreen.Content = i.ToString() + "Размер: " + bByteImageAndSizeImage.Length.ToString();

});

}

else

if (Equals(dataCommantStr.ToString(), "bron"))

{

Dispatcher.Invoke(() =>

{

CheckVisibleOrHidden(true);

});

int s = 0;

byte[] databrgd = Encoding.UTF8.GetBytes("bron");

byte[] dataSizeImageToServer = new byte[4];

listener.Receive(dataSizeImageToServer, 0, 4, SocketFlags.None);

int sizeImage = BitConverter.ToInt32(dataSizeImageToServer, 0);

var buffersServerScreenShot = new byte[sizeImage];

MemoryStream memoryStream = new MemoryStream();

for (int j = 0; j < sizeImage; j++)

{

s += listener.Receive(buffersServerScreenShot, s, 1, SocketFlags.None);

memoryStream.WriteByte(buffersServerScreenShot[j]);

}

Dispatcher.Invoke(() =>

{

PaintScreenshotToImage(memoryStream, image);

});

listener.Send(databrgd);

Console.WriteLine("broadcastON");

}

else

if (Equals(dataCommantStr.ToString(), "brof"))

{

Dispatcher.Invoke(() =>

{

CheckVisibleOrHidden(false);

});

byte[] databrgd = Encoding.UTF8.GetBytes("brof");

listener.Send(databrgd);

Dispatcher.Invoke(() =>

{

image.Source = null;

});

Console.WriteLine("broadcastOFF");

}

else

if (Equals(dataCommantStr.ToString(), "tlbm") || Equals(dataCommantStr.ToString(), "olbm") || Equals(dataCommantStr.ToString(), "orbm") || Equals(dataCommantStr.ToString(), "mlbm"))

{

if (Equals(dataCommantStr.ToString(), "tlbm") || Equals(dataCommantStr.ToString(), "olbm") || Equals(dataCommantStr.ToString(), "orbm"))

{

byte[] XY = new byte[8];

int X1 = 0, Y1 = 0;

listener.Receive(XY, 0, 8, SocketFlags.None);

X1 = BitConverter.ToInt32(XY, 0);

Y1 = BitConverter.ToInt32(XY, 4);

if (Equals(dataCommantStr.ToString(), "olbm"))

{

Dispatcher.Invoke(() =>

{

Form1.Title = dataCommantStr.ToString() + "X1=" + X1.ToString() + " Y1=" + Y1.ToString();

});

LeftMouseClick(X1, Y1);

}

if (Equals(dataCommantStr.ToString(), "tlbm")) {

Dispatcher.Invoke(() =>

{

Form1.Title = dataCommantStr.ToString() + "X1=" + X1.ToString() + " Y1=" + Y1.ToString();

});

LeftMouseClick(X1, Y1);

LeftMouseClick(X1, Y1);

}

if (Equals(dataCommantStr.ToString(), "orbm")) {

Dispatcher.Invoke(() =>

{

Form1.Title = dataCommantStr.ToString() + "X1=" + X1.ToString() + " Y1=" + Y1.ToString();

});

RightMouseClick(X1, Y1);

}

}

else

if (Equals(dataCommantStr.ToString(), "mlbm")) {

byte[] XY = new byte[16];

int X1, Y1, X2, Y2;

listener.Receive(XY, 0, 16, SocketFlags.None);

X1 = BitConverter.ToInt32(XY, 0);

Y1 = BitConverter.ToInt32(XY, 4);

X2 = BitConverter.ToInt32(XY, 8);

Y2 = BitConverter.ToInt32(XY, 12);

Dispatcher.Invoke(() =>

{

Form1.Title = dataCommantStr.ToString() + "X1=" + X1.ToString() + " Y1=" + Y1.ToString() + "X2=" + X2.ToString() + " Y2=" + Y2.ToString();

});

LeftMouseClickAndMove(X1, Y1, X2, Y2);

}

}

else

if (Equals(dataCommantStr.ToString(), "rwhm")) {

byte[] widthHeigthDisplay = new byte[8];

System.Drawing.Size resolution = Screen.PrimaryScreen.Bounds.Size;

BitConverter.GetBytes(resolution.Width).CopyTo(widthHeigthDisplay, 0); BitConverter.GetBytes(resolution.Height).CopyTo(widthHeigthDisplay, 4);

listener.Send(widthHeigthDisplay);

}

else

if (Equals(dataCommantStr.ToString(), "scmd")){

byte[] sizeQuary = new byte[4];

listener.Receive(sizeQuary, 0, 4, SocketFlags.None);

byte[] quaryByte = new byte[BitConverter.ToInt32(sizeQuary, 0)];

listener.Receive(quaryByte, 0, quaryByte.Length, SocketFlags.None);

dataCommantStr.Clear();

dataCommantStr.Append(Encoding.UTF8.GetString(quaryByte, 0, quaryByte.Length));

m\_sqlCmd.CommandText = dataCommantStr.ToString();

string Sql = "id;time\_on;time\_off#";

try

{

SQLiteDataReader r = m\_sqlCmd.ExecuteReader();

string line = String.Empty;

timer.Stop();

while (r.Read())

{

Sql += r["id"].ToString() + ";" + r["time\_on"].ToString() + ";" + r["time\_off"].ToString() + "#";

}

r.Close();

timer.Start();

}

catch (SQLiteException ex)

{

Console.WriteLine(ex.Message);

}

string sAnswer = Sql;

byte[] answer = new byte[4 + sAnswer.Length];

BitConverter.GetBytes(sAnswer.Length).CopyTo(answer,0); Encoding.UTF8.GetBytes(sAnswer).CopyTo(answer, 4);

listener.Send(answer);

Console.WriteLine("QUestion " + dataCommantStr);

}

else

{

listener.Send(Encoding.UTF8.GetBytes("Пришла неизвестная команда"));

Console.WriteLine("Пришла неизвестная команда");

}

listener.Shutdown(SocketShutdown.Both);

listener.Close();

}

catch (Exception ex)

{

Console.WriteLine("Главный комьютер разорвал соединение");

}

}

});

}

Приложение 6

Программный код методов TurnScreenOff, TurnScreenOn

[DllImport("Coredll.dll", EntryPoint = "SetSystemPowerState", CharSet = CharSet.Unicode)]  
private extern static void \_SetSystemPowerState(uint systemState, uint stateFalgs, uint options);  
public const uint POWER\_STATE\_ON = 0x00010000;  
public const uint POWER\_STATE\_IDLE = 0x00100000;  
public void TurnScreenOff()  
{  
\_SetSystemPowerState(0, POWER\_STATE\_IDLE, 0);  
}  
public void TurnScreenOn()  
{  
\_SetSystemPowerState(0, POWER\_STATE\_ON, 0);  
}

Приложение 7

Программный код методов CreateDatabase, addOn, addOff

void addOn()

{

m\_sqlCmd.CommandText = "INSERT INTO tb\_working(time\_on,time\_off) VALUES('" + DateTime.Now.ToString("yyyy–MM–ddTHH:mm:ss") + "','" + DateTime.Now.ToString("yyyy–MM–ddTHH:mm:ss") + "')";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "SELECT last\_insert\_rowid() FROM tb\_working ";

id\_working = Convert.ToInt32(m\_sqlCmd.ExecuteScalar());

}

void addOff()

{

m\_sqlCmd.CommandText = "UPDATE tb\_working SET time\_off='" + DateTime.Now.ToString("yyyy–MM–ddTHH:mm:ss") + "' WHERE id=" + id\_working.ToString();

m\_sqlCmd.ExecuteNonQuery();

}

void CreateDatabase()

{

SQLiteConnection.CreateFile(dbFileName);

try

{

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

m\_sqlCmd.CommandText = "CREATE TABLE IF NOT EXISTS tb\_working (id INTEGER PRIMARY KEY AUTOINCREMENT, time\_on TEXT, time\_off TEXT)";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "CREATE TABLE IF NOT EXISTS tb\_software (id INTEGER PRIMARY KEY AUTOINCREMENT,id\_working INTEGER, name\_software TEXT INTEGER, time\_on TEXT, time\_off TEXT, FOREIGN KEY (id\_working) REFERENCES tb\_working(id))";

m\_sqlCmd.ExecuteNonQuery();

Console.WriteLine("Connected");

}

catch (SQLiteException ex)

{

Console.WriteLine("Disconnected");

Console.WriteLine("Error: " + ex.Message);

}

}