Всероссийский конкурс исследовательских и проектных работ школьников «Высший пилотаж» Всероссийский конкурс-конференция школьников «Авангард»

Дистанционное управление компьютером со смартфона, в рамках ОС Windows

Исследовательская работа

Автор: Седов Дмитрий,

Учащийся 10 класса,

ЧОУ «Первая гимназия Максимум»

Г. Пенза

2023 г.

Оглавление.

Введение.

Глава 1 Теоретическая часть.

1.1Xamarin.

1.2 Среда разработки

1.3. Немного о C#

Глава 2. Практическая часть.

2.1 Администраторское приложение

2.1.1 Создаем консольное приложение

2.1.2 Добавляем имена.

2.1.3 Добавляем IP адрес вашего компьютера.

2.1.4 Создаем порт 123

2.1.5 Создаем соединение с пользователем.

2.1.6 Пишем метод Sleep

2.2. Пользовательское приложение.

2.2.1 **Поработаем с Xamarin.Forms**

2.2.2 Создаем класс соединения.

2.2.3 Создание пользовательского интерфейса.

2.2.4 Описываем метод соединения.

2.2.5 Обновление класса приложения

2.2.6 Добавление функции снимка экрана ученика.

Заключение.

Введение.

Неправильное использование компьютеров в образовательном процессе, например в школе, является обычной практикой среди пользователей, где отсутствуют определенные системы контроля. Приводит это к тому, что на уроках, где используются ноутбуки или персональные компьютеры, школьники могут потратить свое учебное время на игры или что-то другое. Это приводит к напрасной трате ресурсов и времени, снижает скорость достижения целей урока. Таким образом, для достижения желаемых результатов необходим эффективный механизм, с помощью которого учитель может отслеживать и контролировать рабочие компьютеры школьников.

Целью данного проекта является разработка приложения на основе XamarinForms, которое позволит учителю, проводить удаленное наблюдение на своем рабочем месте.

В ходе проекта будет создано два приложения. Одно из которых будет функционировать на рабочих компьютерах школьников, а второе на смартфоне учителя.

Результатом работы будет функциональная система контроля за персональными компьютерами и ноутбуками школьников, что повысит результативность урока.

Глава 1 Теоретическая часть.

1.1 Xamarin.

Xamarin – это инструмент для создания приложений на языках семейства .NET (C#, F#, Visual Basic), который позволяет создавать единый код, работающий на Android, iOS и Windows. Это xaml-подобная технология, то есть интерфейс описывается декларативно в формате xml, вы сразу видите, как элементы расположены на форме и какие свойства имеют.

Xamarin.Forms – это кроссплатформенный набор инструментов пользовательского интерфейса, который позволяет разработчикам легко создавать нативные макеты пользовательского интерфейса, которые можно использовать совместно на Android, iOS и Windows Phone.

Возможности Xamarin.Forms

- Несколько различных макетов страниц – включая навигационную страницу, которая управляет навигационным стеком других страниц. Страница с вкладками, содержащая другие страницы, доступ к которым осуществляется с помощью вкладок, и главная страница с подробной информацией.
- Xamarin.Forms предоставляет средства для компоновки элементов управления на страницах с помощью так называемых макетов, включая Stack, Grid, Absolute и Relative.
- Механизм привязки – свойство класса может быть «привязано» к свойству элемента управления, например, свойство Text для Label. Уже одно это значительно ускоряет время разработки.
- Отправка сообщений через класс MessagingCenter, позволяющий различным классам и компонентам общаться, ничего не зная друг о друге.
- Существует множество утилит для доступа к базовым проектам платформы, чтобы разработчик мог добавить специфический для платформы функционал в основной или общий проект Xamarin.Forms.
- Класс DependencyService – одна из таких утилит, указатель служб, который позволяет приложениям Xamarin.Forms вызывать собственные функции платформы из общего кода.
- Эффекты – это средство, с помощью которого вы можете создавать небольшие специфические для платформы изменения пользовательского интерфейса элементов управления и применять их в общем проекте.
- Пользовательские рендереры (Custom Renders) позволяют полностью контролировать отображение элемента управления в Xamarin.Forms, таким образом вы можете добавить любой дополнительный внешний вид или функциональность, которая вам может понадобиться.

1.2 Среда разработки

Для написания программ с Xamarin.Forms используется Visual Studio. Для мобильной разработки имеется целый ряд встроенных эмуляторов мобильных устройств, а также возможность подключить реальный девайс для отладки.

1.3. Немного о C#

C# живет по принципу «всякая сущность есть объект». Его причисляют к объектно-ориентированным, а точнее объектным, языкам программирования. «Язык основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода»

– так принято характеризовать его. Однако скептики сомневаются как минимум в его безопасности.

Сторонники C# называют его самым мультипарадигменным, универсальным, продвинутым и удобным в использовании языком программирования.

Название «Си шарп» (от англ. sharp — диез) несет «сакральный» смысл. Знак «#» (в музыкальной нотации читается как «диез») означает повышение высоты звука на полтона. С другой стороны, название «C#» получается путем следующей «эволюционной цепочки»: C → C++ → C++++(C#), так как символ «#» можно составить из 4-х знаков «+».

Авторами этого языка программирования стали Скотт Вилтамут и Андерс Хейльсберг — создатель Турбо Паскаля и Дельфи, перешедший в 1996 году в Microsoft. C# поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм.

Синтаксис языка программирования C#

Идентификатор – имя сущности. Идентификатор может только начинаться с символа {_}

Ключевые слова - это предварительно определенные зарезервированные идентификаторы, имеющие особое синтаксическое значение. Язык программирования C# имеет два типа ключевых слов — зарезервированные в любой части кода и контекстные. Ключевые слова, которые потребуются при создании игры:

- Namespace – ключевое слово указывает название области выполнения скрипта для видеоигры
- Static – ключевое слово используется для объявления статического члена, принадлежащего собственно типу, а не конкретному объекту, также static можно использовать для объявления классов static
- Void – метод возврата, который не возвращает значение
- Private – ключевое слово является модификатором доступа к переменным. Private является уровнем доступа с минимальными правами. Доступ к закрытым переменным можно получить только внутри тела класса или структуры, в которой они объявлены
- String – класс, который предоставляет множество методов для безопасного создания, обработки и сравнения строк
- Float – тип данных, представляющее действительное число с плавающей запятой в диапазоне от ±1,5 x 10−45 до ±3,4 x 1038
- Int – тип данных, представляющее 32-разрядное целое число со знаком в диапазоне от -2 147 483 648 до 2 147 483 647
- Using – ключевое слово позволяет обращаться скрипту к другим скриптам или библиотекам
- Public – ключевое слово является модификатором доступа к переменным. Public является уровнем доступа с максимальными правами. Ограничений доступа к общим членам не существует

Ключевые слова в среде Unity

- GameObject тип объектов, которые могут существовать в Сцене
- MonoBehaviour базовый класс, от которого по умолчанию наследуется каждый скрипт Unity

- Object – базовый класс для всех объектов, на которые Unity может ссылаться в редакторе

Глава 2.

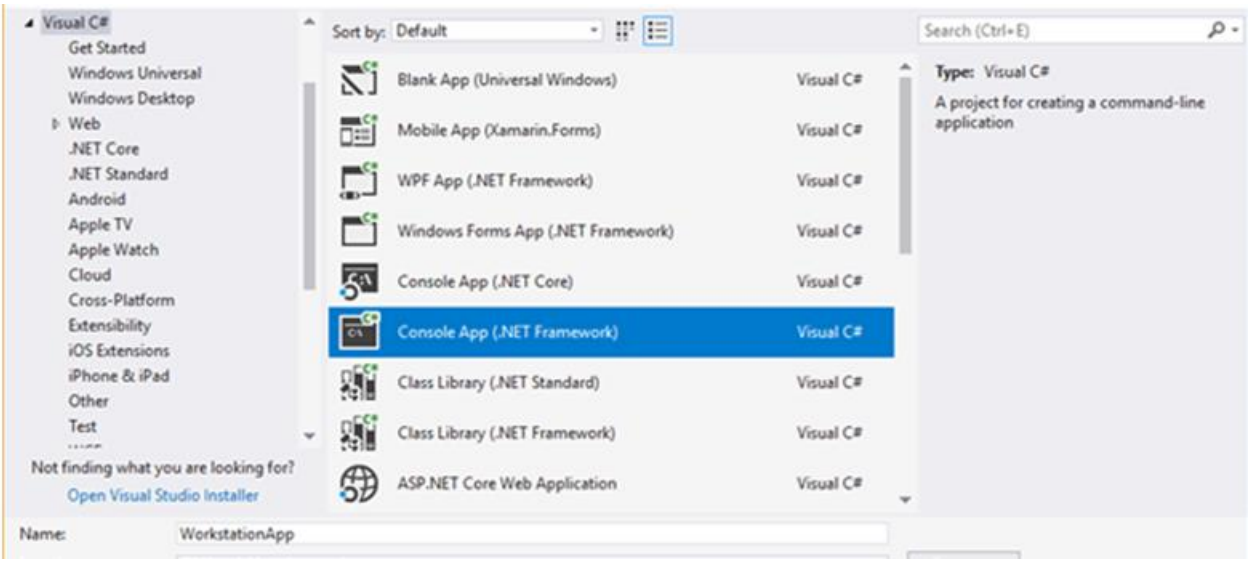
Практическая часть.

В процессе работы над проектом мы создаем два приложения: администраторское, для учителя, и пользовательское, для учеников.

2.1. Администраторское приложение

2.1.1. Создаем консольное приложение

Начнём с того, что создадим консольное приложения с NET.FRAEMWORK .



2.1.2. Добавляем имена.

```
public static TcpClient client;
private static TcpListener listener;
private static string ipString;
```

2.1.3. Добавляем IP адрес вашего компьютера.

[redacted]

2.1.4. Создаем порт 123

```
IPEndPoint ep = new IPEndPoint(IPAddress.Parse(ipString), 123);
listener = new TcpListener(ep);
listener.Start();
Console.WriteLine(@"
==================================================
Started listening requests at: {0}:{1}
==================================================",
ep.Address, ep.Port);
client = listener.AcceptTcpClient();
```

```
Console.WriteLine("Connect" + " \n");
```

2.1.5. Создаем соединение с пользователем.

```
while (client.Connected) {
    try {
        const int bytesize = 1024 * 1024;
        byte[] buffer = new byte[bytesize];
        string x = client.GetStream().Read(buffer, 0, bytesize).ToString();
        var data = ASCIIEncoding.ASCII.GetString(buffer);
        if (data.ToUpper().Contains("SLP2")) {
            Console.WriteLine("Pc is going to Sleep Mode!" + " \n");
            Sleep();
        }
    } catch (Exception exc) {
        client.Dispose();
        client.Close();
    }
}
```

2.1.6. Пишем метод Sleep

```
using System.Windows.Forms;
void Sleep() {
    Application.SetSuspendState(PowerState.Suspend, true, true);
}
```

2.2. Пользовательское приложение.

2.2. 1. Поработаем с Xamarin.Forms

Открываем Visual Studio и переходим в New Project-> Cross-platform-> Xamarin.Forms-> Blank app. Даем ему имя, XamarinFormsPOL.

2.2.2. Создаем класс соединения.

```
using System;
using System.Collections.Generic;
using System.Net.Sockets;

namespace XamarinForms.Client
{
    public class Connection
    {
        private static Connection _instance;
        public static Connection Instance
        {
            get
            {
                if (_instance == null) _instance = new Connection();
```

```
            return _instance;
        }
    }
    public TcpClient client { get; set; }
  }
}
```

2.2.3 Создание пользовательского интерфейса.

```
  <?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
          xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
          xmlns:local="clr-namespace:XamarinForms.Client"
          x:Class="XamarinForms.Client.MainPage">
  <StackLayout>
  <Label Text="Connect to Server"
        FontSize="Medium"
        HorizontalOptions="Center" />
  <Entry x:Name="IPAddress" Placeholder="IP Address"/>
  <Entry x:Name="Port" Placeholder="Port Number"/>
  <Button x:Name="Connect" Text="Connect" Clicked="Connect_Clicked"/>
  </StackLayout>
</ContentPage>
```

2.2.4. Описываем метод соединения.

```
using System;
using System.Net.Sockets;
using Xamarin.Forms;


namespace XamarinForms.Client
{
  public partial class MainPage : ContentPage
  {
    public MainPage()
    {
      InitializeComponent();
    }

    private async void Connect_Clicked(object sender, EventArgs e)
    {
      try
      {
        TcpClient client = new TcpClient();
          await client.ConnectAsync(IPAddress.Text, Convert.ToInt32(Port.Text));
```

```
            if (client.Connected)
            {
                Connection.Instance.client = client;
                Application.Current.MainPage = new NavigationPage(new OperationsPage());

                await DisplayAlert("Connected", "Connected to server successfully!", "Ok");
            }
            else
            {
                await DisplayAlert("Error", "Connection unsuccessful!", "Ok");
            }
        }
        catch (Exception ex)
        {
            await DisplayAlert("Error", ""+ex.ToString(), "Ok");
        }
    }
}
```

2.2.5. Обновление класса приложения

```
MainPage = new NavigationPage(new MainPage());
```

2.2.6. Добавление функции снимка экрана ученика.

Подправим ранее написанный код в файле program.cs

```
 else if (data.ToUpper().Contains("SHTD3")) {
  Console.WriteLine("Pc is going to Shutdown!" + " \n");
  Shutdown();
} else if (data.ToUpper().Contains("TSC1")) {
  Console.WriteLine("Take Screenshot!" + " \n");
  var bitmap = SaveScreenshot();
  var stream = new MemoryStream();
  bitmap.Save(stream, ImageFormat.Bmp);
  sendData(stream.ToArray(), client.GetStream());
}
```

Добавим в серверное приложение следующие пространства имен. Иначе Visual Studio не распознает ключевое слово graphics функции снимка экрана.

```
 using System.Drawing;
using System.Drawing.Imaging;
```

Вернемся к program.cs и поместите эти функции в основной класс.

```
// Функция выключения рабочей станции
void Shutdown() {
  System.Diagnostics.Process.Start("Shutdown", "-s -t 10");
}
// Функция сохранения скриншота
```

```
Bitmap SaveScreenshot() {
    var bmpScreenshot = new Bitmap(Screen.PrimaryScreen.Bounds.Width,
Screen.PrimaryScreen.Bounds.Height, PixelFormat.Format32bppArgb);
    // Создание графического bitmap-объекта
    var gfxScreenshot = Graphics.FromImage(bmpScreenshot);
    // Берем скриншот из Take the screenshot от верхнего левого до нижнего правого угла
    gfxScreenshot.CopyFromScreen(Screen.PrimaryScreen.Bounds.X,
Screen.PrimaryScreen.Bounds.Y, 0, 0, Screen.PrimaryScreen.Bounds.Size,
CopyPixelOperation.SourceCopy);
    return bmpScreenshot;
}
// Преобразуем изображение в байтовый код.
void sendData(byte[] data, NetworkStream stream) {
    int bufferSize = 1024;
    byte[] dataLength = BitConverter.GetBytes(data.Length);
    stream.Write(dataLength, 0, 4);
    int bytesSent = 0;
    int bytesLeft = data.Length;
    while (bytesLeft > 0) {
        int curDataSize = Math.Min(bufferSize, bytesLeft);
        stream.Write(data, bytesSent, curDataSize);
        bytesSent += curDataSize;
        bytesLeft -= curDataSize;
    }
```

Теперь внесем изменения в пользовательском приложении.

Возвращаемся в клиентское приложение (Xamarin.Forms) и добавляем новый ContentPage с именем OperationsPage. Внутри этого макета добавим следующий код, чтобы создать больше кнопок.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:XamarinForms.Client"
             x:Class="XamarinForms.Client.OperationsPage">
    <ContentPage.Content>
        <StackLayout Orientation="Vertical">
            <Button x:Name="Screenshot" Text="Screenshot" Clicked="Screenshot_Clicked"/>
            <Button x:Name="Sleep" Text="Sleep" Clicked="Sleep_Clicked"/>
            <Button x:Name="Shutdown" Text="Shutdown" Clicked="Shutdown_Clicked"/>
            <Image x:Name="imageView"/>

        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Открываем файл OperationPage.xamlи переносим следующий код.

```
using System;
```

```
using System.IO;
using System.Net.Sockets;
using System.Text;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace XamarinForms.Client
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class OperationsPage : ContentPage
    {
        public OperationsPage ()
        {
            InitializeComponent ();
        }

        // Команда для кнопки Sleep
        private void Sleep_Clicked(object sender, EventArgs e)
        {
            var client = Connection.Instance.client;
            NetworkStream stream = client.GetStream();
            String s = "SLP2";
            byte[] message = Encoding.ASCII.GetBytes(s);
            stream.Write(message, 0, message.Length);
        }

        // Команда для кнопки Shutdown
        private void Shutdown_Clicked(object sender, EventArgs e)
        {
            var client = Connection.Instance.client;
            NetworkStream stream = client.GetStream();
            String s = "SHTD3";
            byte[] message = Encoding.ASCII.GetBytes(s);
            stream.Write(message, 0, message.Length);
        }

        // Команда для снимка экрана
        private void Screenshot_Clicked(object sender, EventArgs e)
        {
            var client = Connection.Instance.client;
            NetworkStream stream = client.GetStream();
            String s = "TSC1";
            byte[] message = Encoding.ASCII.GetBytes(s);
            stream.Write(message, 0, message.Length);
            var data = getData(client);
            imageView.Source = ImageSource.FromStream(() => new MemoryStream(data));
```

```
        }

        // Сбор данных с сервера
        public byte[] getData(TcpClient client)
        {
            NetworkStream stream = client.GetStream();
            byte[] fileSizeBytes = new byte[4];
            int bytes = stream.Read(fileSizeBytes, 0, fileSizeBytes.Length);
            int dataLength = BitConverter.ToInt32(fileSizeBytes, 0);

            int bytesLeft = dataLength;
            byte[] data = new byte[dataLength];

            int buffersize = 1024;
            int bytesRead = 0;

            while (bytesLeft > 0)
            {
                int curDataSize = Math.Min(buffersize, bytesLeft);
                if (client.Available < curDataSize)
                    curDataSize = client.Available;
                bytes = stream.Read(data, bytesRead, curDataSize);
                bytesRead += curDataSize;
                bytesLeft -= curDataSize;
            }
            return data;
        }
    }
}
```

Заключение.

Данное приложение, а точнее тандем приложений позволяет отключить ПК ученика или сделать снимок экрана в реальном времени, что достаточно удобно. У учителя не всегда есть время контролировать всех учеников одновременно .

Говоря о практическом применении этих приложений не только в школе, можно рассмотреть их использование в различных организациях и компаниях.

Сейчас приложения находятся на дальнейшей разработке. Есть место, куда расти и развивать их.

Рецензия

на проектную работу ученика 10 класса

Седова Дмитрия на тему:

«Дистанционное управление компьютером со смартфона в рамках ОС Windovs»

Данный проект на тему «Дистанционное управление компьютером со смартфона в рамках ОС Windovs» соответствует требованиям ФГОС, предъявляемым к содержанию, оформлению индивидуального проекта.

Содержание работы соответствует целям и задачам проектной работы. Выдержаны требования к структуре проекта.

В процессе работы над проектом учащийся самостоятельно выбрал тему, сформулировал цель и задачи, подобрал теоретический материал, создал презентацию и сами приложения.

Теоретическая часть содержит информацию о Xamarin, выбранном языке программирования и необходимость использования именно их. Теоретическая часть соответствует выбранной теме.

Экспериментальная часть выполнена логично, подробно. Представлены необходимые фотографии, скрины, текст кода, с объяснением.

Замечательно, что есть часть, которая содержит актуальность данной темы в наши дни, когда профессии из it-сферы так востребованы.

В заключении работы сформулирован вывод и представлен готовый продукт – приложение.

Обобщая все сказанное, работу Седова Дмитрия можно считать интересной и актуальной.

Научный руководитель: Фахретдинова Л.А.  (подпись)