АНО «Ритм» является клубом для детей и подросков, который помогает детям выбрать свое профиссианальное направление в дальнейшем. При обучении, дети получают базовые знания и навыки в области схемотехники, электроники, программирования, а также точных наук — физики, геометрии, математики.

1) Структуры предприятия, и его процессы.
1. Управление предприятием:
- Бугалтерские работы;
- Процессы управления и регулирования внутренним распорядком;
- Решение разного рода вопросов, связанных как и с другими структурами, так и с внешними факторами;
2. Обучающая деятельность:
- Создание метадических пособий для детей;
- Занятия с детьми;
3. Работа с медиа-структурой:
- Обновление группы в социальной сети ВКонтакте и создение постов в ней;
- Ответы на задаваемые вопросы, как и в группе в ВКонтакте, так и на сотовые звонки;

2) Используемые системы автоматизации обмена информацией.
Передача информации на данном предприятии, осуществляется по средствам социальных сетей и электорнных почт, или воочию.
Данные методы используются по той причине, что предприятие не имеет крупных масштабов, необходимых для использования уже готовых решений для обмена крупной информацией, либо для создания собственных программ.

3) Тестовое задани.
Необходимо составить программу, которая способна по локальной сети, на других компьютерах, открывать файлы находящиеся на компьютере преподователя. Так-же иметь возможность, открывать файлы на определенных компьютерах, а не на всех разом.

```
using Newtonsoft.Json.Linq;
using System.Diagnostics;
using System.Formats.Tar;
using WebSocketSharp;

namespace practice
{
    internal static class Program
    {
        public const string PathDirectoryData = "Data";
        public const string PathDirectoryDataFile = PathDirectoryData + "\\File";
        public const string PathConfig = PathDirectoryData + "\\Config.json";
        public static FormMain form;

        [STAThread]
        static void Main()
        {
            if (!Directory.Exists(PathDirectoryData)) // проверка, есть ли папка для хранения информации
            {
                Directory.CreateDirectory(PathDirectoryData); // создание папки для хранения информации
            }
            if(!Directory.Exists(PathDirectoryDataFile)) // проверка, есть ли папка для хранения файлов
            {
                Directory.CreateDirectory(PathDirectoryDataFile); // создание папки для хранения файлов
            }
            Config.Load(PathConfig);


            form = new FormMain(Config.Type <= 0);
            if (Config.Type > 0)
            {
```

```
            Server.Start();
        }
        else
        {
            Client.Start();
        }
        Application.Run();
    }



    public static void Restart()
    {
        //Application.Restart(); - Не всегда срабатывает
        Process.Start(Application.ExecutablePath);
        Environment.Exit(0);
    }
    public static void Exit()
    {
        string[] files = Directory.GetFiles(PathDirectoryDataFile);
        foreach (string file in files)
        {
            try
            {
                File.Delete(file);
            }
            catch(Exception) { }
        }
        Environment.Exit(0);
    }

  }
}
using Newtonsoft.Json.Linq;
using practice;
using System.Net;
using WebSocketSharp;
using WebSocketSharp.Server;

class Server : WebSocketBehavior
{
  static Dictionary<string, string> files = new Dictionary<string, string>();
  public static int MaxClients = 0;
  public static Dictionary<int, ClientInfo> Clients = new Dictionary<int, ClientInfo>();
  static WebSocketServer webSocketServer;

  public static string GetFile(string path)
  {
    if(files.ContainsValue(path))
    {
      foreach(KeyValuePair<string, string> file in files)
      {
        if (file.Value == path)
        {
          return file.Key;
        }
      }
    }
    string name = Convert.ToString(DateTime.Now.Ticks, 16) + '.' + path.Split('.').Last();
    files.Add(name, path);
    return name;
  }
```

```
public static void ClientClose(CloseEventArgs e, WebSocket WebSocket)
{
    int IdClose = -1;
    foreach (KeyValuePair<int, ClientInfo> client in Server.Clients)
    {
        if (client.Value.WebSocket == WebSocket)
        {
            IdClose = client.Key;
            break;
        }
    }
    if(IdClose >= 0 && Clients.Remove(IdClose))
    {
        Program.form.UpdateUI();
    }
}
public static void ClientAdd(WebSocket WebSocket, string Name)
{
    if(MaxClients <= Clients.Count)
    {
        MaxClients = Clients.Count;
        Clients.Add(MaxClients++, new ClientInfo(WebSocket, Name));
    }
    else
    {
        for(int i = 0; i < MaxClients; i++)
        {
            if(!Clients.ContainsKey(i))
            {
                Clients.Add(i, new ClientInfo(WebSocket, Name));
                break;
            }
        }
    }
    Program.form.UpdateUI();
}
public static void Start()
{
    webSocketServer = new WebSocketServer($"ws://127.0.0.1:{Config.Port}");
    webSocketServer.AddWebSocketService<Server>($"/{Config.Service}");
    webSocketServer.Start();
}
public static void Stop() => webSocketServer.Stop();
public static void SendMessage(string msg) => webSocketServer.WebSocketServices.Broadcast(msg);
public static void SendMessage(int id, string msg)
{
    if(Clients.ContainsKey(id))
    {
        Clients[id].WebSocket.Send(msg);
    }
}
public static void SendMessage(int[] ids, string msg)
{
    foreach(int id in ids)
    {
        SendMessage(id, msg);
    }
}
protected override void OnMessage(MessageEventArgs e)
{
    try
    {
        JObject json = JObject.Parse(e.Data);
        switch ((string)json["Type"])
        {
```

```
            case "Connect":
                ClientAdd(Context.WebSocket, (string)json["Name"]);
                Context.WebSocket.OnClose += (sender, e) => ClientClose(e, Context.WebSocket);
                break;
            case "Download":
                byte[] data = File.ReadAllBytes(files[(string)json["File"]]);
                Send(data);
                break;
        }
      }
      catch (Exception) { }
   }
}
class ClientInfo
{
   public WebSocket WebSocket;
   public string Name;
   public ClientInfo(WebSocket WebSocket, string Name)
   {
      this.WebSocket = WebSocket;
      this.Name = Name;
   }
}
using Newtonsoft.Json.Linq;

public static class Config
{
   private static string Path;
   private static JObject json;
   private static JObject jsonFiles => (JObject)json["Files"];

   public static int Type = 0;
   public static string Service = "afarys";
   public static ushort Port = 10101;
   public static string Name = "None";
   public static Dictionary<string, string> Files = new Dictionary<string, string>();


   public static void Load(string Path)
   {
      Config.Path = Path;
      if (File.Exists(Path)) //проверка, есть ли файл с конфигом
         ReadConfig(); // чтение конфигов из файла
      else
         CreateConfig(); // создание файла с конфигами
   }
   private static void CreateConfig()
   {
      FileStream fileStream = File.Create(Path);
      fileStream.Close();
      json = new JObject
      {
         { "Service", Service },
         { "Name", Name },
         { "Type", Type },
         { "Files", new JObject() }
      };
      save();
   }
   private static void ReadConfig()
   {
      json = JObject.Parse(File.ReadAllText(Path));
      bool full = true; //полный ли файл
      #region Type
      if (!json.ContainsKey("Type"))
```

```
    {
        full = false;
        json.Add("Type", Type);
    }
    Type = (int)json["Type"];
    #endregion
    #region Service
    if (!json.ContainsKey("Service"))
    {
        full = false;
        json.Add("Service", Service);
    }
    Service = (string)json["Service"];
    #endregion
    #region Port
    if (!json.ContainsKey("Port"))
    {
        full = false;
        json.Add("Port", Port);
    }
    Port = (ushort)json["Port"];
    #endregion
    #region Name
    if (!json.ContainsKey("Name"))
    {
        full = false;
        json.Add("Name", Name);
    }
    Name = (string)json["Name"];
    #endregion
    #region Files
    if (!json.ContainsKey("Files"))
    {
        full = false;
        json.Add("Files", new JObject());
    }
    Files = ((JObject)json["Files"]).ToObject<Dictionary<string, string>>();
    #endregion
    if (!full)
    {
        save();
    }
}
private static void save()
{
    StreamWriter streamWriter = new StreamWriter(Path);
    streamWriter.Write(json);
    streamWriter.Close();
}
public static void FilesAdd(string key, string Path)
{
    if (!jsonFiles.ContainsKey(key))
    {
        jsonFiles.Add(key, Path);
        save();
    }
    else
        FilesUpdate(key, Path);
}
public static void FilesUpdate(string key, string Path)
{
    if (jsonFiles.ContainsKey(key))
    {
        jsonFiles[key] = Path;
        save();
```

```
        }
        else
            FilesAdd(key, Path);
    }
    public static void FilesRemote(string key)
    {
        if (jsonFiles.ContainsKey(key))
        {
            jsonFiles.Remove(key);
            save();
        }
    }
    public static void NameUpdate(string Name)
    {
        if (Name != null)
        {
            json["Name"] = Name;
            Config.Name = Name;
            save();
        }
    }
    public static void Update(JObject json)
    {
        if(json != null)
        {
            if(json.ContainsKey("Service"))
            {
                Service = (string)json["Service"];
                Config.json["Service"] = json["Service"];
            }
            if(json.ContainsKey("Name"))
            {
                Name = (string)json["Name"];
                Config.json["Name"] = json["Name"];
            }
            if (json.ContainsKey("Type"))
            {
                Type = (int)json["Type"];
                Config.json["Type"] = json["Type"];
            }
            if (json.ContainsKey("Files"))
            {
                Files = ((JObject)json["Files"]).ToObject<Dictionary<string, string>>();
                Config.json["Files"] = json["Files"];
            }
            save();
        }
    }
}
using Newtonsoft.Json.Linq;
using practice;
using System.Diagnostics;
using WebSocketSharp;

static class Client
{
    static WebSocket webSocket;
    static string file = null;
    public static void Start()
    {
        if(webSocket != null && webSocket.IsAlive)
        {
            webSocket.Close();
        }
        webSocket = new WebSocket($"ws://192.168.1.110:{Config.Port}/{Config.Service}");
```

```
        webSocket.OnMessage += (sender, e) => Task(e);
        webSocket.Connect();
        Send(new JObject() { { "Type", "Connect" }, { "Name", Config.Name } }.ToString());
    }
    public static void Send(string text)
    {
        if (webSocket != null)
            webSocket.Send(text);
        else
            Start();
    }
    static void Task(MessageEventArgs e)
    {
        if(e.IsText)
        {
            Task(e.Data);
        }
        else if(e.IsBinary)
        {
            File.WriteAllBytes(Program.PathDirectoryDataFile + "/" + file, e.RawData);
            Open();
        }
    }
    static void Open()
    {
        string type = file.Split('.').Last();
        string path = "\"" + AppDomain.CurrentDomain.BaseDirectory + Program.PathDirectoryDataFile + "\\" + file
+ "\"";
        if (Config.Files.ContainsKey(type))
        {
            try
            {
                Process.Start(Config.Files[type], path);
            }
            catch (Exception)
            {
                Process.Start("explorer", path);
            }
        }
        else
        {
            Process.Start("explorer", path);
        }
        file = null;
    }
    static void Task(string task)
    {
        try
        {
            JObject json = JObject.Parse(task);
            switch ((string)json["Type"])
            {
                case "Update":
                    Config.Update((JObject)json["JSON"]);
                    break;
                case "FileUpdate":
                    Config.FilesUpdate((string)json["Key"], (string)json["Path"]);
                    break;
                case "FileAdd":
                    Config.FilesAdd((string)json["Key"], (string)json["Path"]);
                    break;
                case "FileRemote":
                    Config.FilesRemote((string)json["Key"]);
                    break;
                case "NameUpdate":
```

```
                Config.NameUpdate((string)json["Name"]);
                break;
            case "AppRestart":
                Program.Restart();
                break;
            case "AppClose":
                Program.Exit();
                break;
            case "FileOpen":
                file = (string)json["File"];
                if (File.Exists(Program.PathDirectoryDataFile + "/" + file))
                    Open();
                else
                    Send(new JObject() { { "Type", "Download" }, { "File", file } }.ToString());
                break;
            case "FileDelete":
                File.Delete(Program.PathDirectoryDataFile + "/" + (string)json["File"]);
                break;
        }
    }
    catch(Exception) { }

  }
}
using Newtonsoft.Json.Linq;
using practice.Properties;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace practice
{
  public partial class FormMain : Form
  {
    Form formSetting;
    private readonly SynchronizationContext syncContext;
    public FormMain(bool isClient)
    {
      InitializeComponent();
      formSetting = new FormSetting();
      if (isClient)
      {
        Hide();
        notifyIcon.ContextMenuStrip = contextMenuStripClient;
      }
      else
      {
        Show();
        notifyIcon.ContextMenuStrip = contextMenuStripServer;
        syncContext = SynchronizationContext.Current;
      }
    }
    private void FormMain_FormClosing(object sender, FormClosingEventArgs e)
    {
      if (e.CloseReason == CloseReason.UserClosing)
      {
        e.Cancel = true;
      }
      Hide();
    }
    public void UpdateUI() => syncContext.Post(UpdateUI, null);

    private void UpdateUI(object? content)
    {
      for (; blocks.Count < Server.MaxClients;
        blocks.Add(new Block(PanelGroup))) ;
      for(int i = 0; i < Server.MaxClients; i++)
```

```
        {
            if(Server.Clients.ContainsKey(i))
            {
                blocks[i].StatusConnect(true);
                blocks[i].UpdateName(Server.Clients[i].Name);
            }
            else
            {
                blocks[i].StatusConnect(false);
                blocks[i].UpdateName("");
            }
        }
    }
    List<Block> blocks = new List<Block>();




    private void showToolStripMenuItem_Click(object sender, EventArgs e) => Show();
    private void hideToolStripMenuItem_Click(object sender, EventArgs e) => Hide();
    private void settingToolStripMenuItem_Click(object sender, EventArgs e) => formSetting.Show();
    private void reconnectToolStripMenuItem_Click(object sender, EventArgs e) => Client.Start();
    private void restartToolStripMenuItem_Click(object sender, EventArgs e) => Program.Restart();
    private void closeToolStripMenuItem_Click(object sender, EventArgs e) => Program.Exit();
    private void buttonSelectAll_Click(object sender, EventArgs e)
    {
        foreach(Block block in blocks)
        {
            block.Select(true);
        }
    }
    private void buttonRemoveSelection_Click(object? sender, EventArgs? e)
    {
        foreach (Block block in blocks)
        {
            block.Select(false);
        }
    }

    private void buttonOpenFile_Click(object sender, EventArgs e)
    {
        OpenFileDialog dialog = new OpenFileDialog();
        dialog.Title = "Select the file that will open on the selected computers";
        if (dialog.ShowDialog() == DialogResult.OK)
        {
            string file = Server.GetFile(dialog.FileName);
            Send(new JObject() { { "Type", "FileOpen" }, { "File", file } }.ToString());
        }
    }
    private void buttonRestart_Click(object sender, EventArgs e) =>Send(new JObject() { { "Type",
"AppRestart" } }.ToString());
    private void buttonClose_Click(object sender, EventArgs e) => Send(new JObject() { { "Type",
"AppClose" } }.ToString());
    private void Send(string msg)
    {
        foreach (KeyValuePair<int, ClientInfo> client in Server.Clients)
        {
            if (blocks[client.Key].isSelect)
            {
                Server.SendMessage(client.Key, msg);
            }
        }
        buttonRemoveSelection_Click(null, null);
    }
```

```
    }
}

class Block
{
    Panel Box;
    PictureBox Picture;
    Label Name;
    Label Status;
    bool select = false;
    public bool isSelect => select;
    public void Select() => Select(!select);
    public void Select(bool select)
    {
        this.select = select;
        if (select)
        {
            Box.BorderStyle = BorderStyle.Fixed3D;
        }
        else
        {
            Box.BorderStyle = BorderStyle.None;
        }
    }
    public Block(FlowLayoutPanel flow)
    {

        Box = new Panel();
        Box.Width = 100;
        Box.Height = 100;

        Picture = new PictureBox();
        Picture.Image = Resources.computer;
        Picture.SizeMode = PictureBoxSizeMode.Zoom;
        Picture.Width = 70;
        Picture.Height = 70;
        Box.Controls.Add(Picture);
        Picture.Left = 15;
        Picture.Top = 0;

        Name = new Label();
        Name.Text = "";
        Box.Controls.Add(Name);
        Name.ForeColor = Color.Black;
        Name.Font = new Font(Name.Font.FontFamily, 12);
        Name.Top = 75;
        Name.Left = 0;

        Status = new Label();
        Status.Text = "● ";
        Status.Font = Name.Font;
        StatusConnect(true);
        Box.Controls.Add(Status);

        flow.Controls.Add(Box);



        Box.MouseDown += (s, e) => Select();
        Status.MouseDown += (s, e) => Select();
        Picture.MouseDown += (s, e) => Select();
        Name.MouseDown += (s, e) => Select();
    }
    public void UpdateName(string name)
    {
```

```
        Name.Text = name;
    }
    public void StatusConnect(bool Connect)
    {
        if(Connect)
        {
            Status.ForeColor = Color.Green;
        }
        else
        {
            Status.ForeColor = Color.Red;
        }
    }
}
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace practice
{
    public partial class FormSetting : Form
    {



        public FormSetting()
        {
            InitializeComponent();
            inputPort.Controls.RemoveAt(0);
            UpdateInfo();
        }

        private void FormSetting_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (e.CloseReason == CloseReason.UserClosing)
                e.Cancel = true;
            Hide();
        }

        private void buttonDone_Click(object sender, EventArgs e)
        {
            JObject json = new JObject();
            if(inputName.Text != null && inputName.Text.Trim() != "")
            {
                json.Add("Name", inputName.Text.Trim());
            }

            json.Add("Port", (ushort)inputPort.Value);

            if (inputService.Text != null && inputService.Text.Replace(" ", "") != "")
            {
                json.Add("Service", inputService.Text.Replace(" ", ""));
            }

            json.Add("Type", radioButtonServer.Checked ? 1 : 0);

            JObject jsonFiles = new JObject();
            foreach(DataGridViewRow row in dataGridViewFiles.Rows)
```

```
    {
        if (row.Cells[1].Value != null && (string)row.Cells[1].Value != ""
            && row.Cells[0].Value != null && (string)row.Cells[0].Value != "")
        {
            jsonFiles.Add((string)row.Cells[0].Value, (string)row.Cells[1].Value);
        }
    }
    json.Add("Files", jsonFiles);
    Config.Update(json);
    DialogResult r = MessageBox.Show(
        "Have you changed some data, do you want to restart the app?",
        "Restart app?",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Question,
        MessageBoxDefaultButton.Button2);
    if(r == DialogResult.Yes)
    {
        Program.Restart();
    }
    Hide();
    UpdateInfo();
}



private void buttonCancel_Click(object sender, EventArgs e)
{
    Hide();
    UpdateInfo();
}

private void dataGridViewFiles_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == 1)
    {
        if (e.RowIndex < dataGridViewFiles.RowCount - 1 && e.RowIndex >= 0)
        {
            DataGridViewCell d = dataGridViewFiles[1, e.RowIndex];

            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "application (*.exe)|*.exe";
            if (d.Value != null && (string)d.Value != "")
            {
                dialog.InitialDirectory = Path.GetDirectoryName((string)d.Value);
            }
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                d.Value = dialog.FileName;
            }
        }
    }
}
private void dataGridViewFiles_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == 0)
    {
        if (e.RowIndex < dataGridViewFiles.RowCount - 1 && e.RowIndex >= 0)
        {
            DataGridViewCell d = dataGridViewFiles[0, e.RowIndex];
            if (d.Value == null || (string)d.Value == "")
            {
                dataGridViewFiles.Rows.RemoveAt(e.RowIndex);
            }
        }
```

```
            }
        }

        void UpdateInfo()
        {
            radioButtonClient.Checked = true;
            if(Config.Type > 0)
            {
                radioButtonServer.Checked = true;
            }
            inputName.Text = Config.Name;
            inputPort.Value = Config.Port;
            inputService.Text = Config.Service;
            dataGridViewFiles.Rows.Clear();
            foreach (KeyValuePair<string, string> file in Config.Files)
            {
                dataGridViewFiles.Rows.Add(new string[]{file.Key, file.Value });
            }
        }
    }
}
```