**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Доцент департамента  программной инженерии  факультета компьютерных наук  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Л. Макаров  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.02.07-01 12 01-1 | | **СИСТЕМА СЛЕЖЕНИЯ ЗА УДАЛЁННЫМ ПОМЕЩЕНИЕМ**  **НА ОСНОВЕ ИНТЕРНЕТА ВЕЩЕЙ**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.02.07-01 12 01-1-ЛУ** | | |
|  |  | |
| Исполнитель:  студент группы БПИ184  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /А. Д. Курылев /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |
|  | | |
|  | |  |

**Москва 2019**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.02.07-01 12 01-1 | | УТВЕРЖДЁН  RU.17701729.02.07-01 12 01-1-ЛУ  **СИСТЕМА СЛЕЖЕНИЯ ЗА УДАЛЁННЫМ ПОМЕЩЕНИЕМ**  **НА ОСНОВЕ ИНТЕРНЕТА ВЕЩЕЙ**  **Текст программы**  **RU.17701729.02.07-01 12 01-1**  **Листов 21** | | | | |
|  |  | | | |
|  | | | |
|  | | | | |
|  | | |  | |

**Москва 2019**

**СОДЕРЖАНИЕ**

[1. ТЕКСТ ПРОГРАММЫ 2](#_Toc9450064)

[1.1. Файл MainForm.cs 2](#_Toc9450065)

[1.2. Файл FilesForm.cs 10](#_Toc9450066)

[1.3. Файл SettingsForm.cs 15](#_Toc9450067)

[1.4. Файл Fieldset.cs 17](#_Toc9450068)

[1.5. Файл SettingsClass.cs 19](#_Toc9450069)

[ПРИЛОЖЕНИЕ 1 21](#_Toc9450070)

[2. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 21](#_Toc9450071)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 22](#_Toc9450072)

1. ТЕКСТ ПРОГРАММЫ
   1. Файл MainForm.cs

#define connection

using System;

using System.Net.Http;

using System.Windows.Forms;

namespace CourseWork

{

public partial class MainCameraForm : Form

{

public MainCameraForm()

{

InitializeComponent();

}

#region Fields

/// <summary>

/// Массивы кнопок для удобного к ним доступа

/// </summary>

Button[] buttons, cameraButtons;

/// <summary>

/// Адрес Raspberry Pi

/// </summary>

public static string uri = "http://192.168.0.101/";

/// <summary>

/// Адресс для FilesForm

/// </summary>

string fw = uri;

/// <summary>

/// Экземпляр класс HttpClient для связи с сервером и отправки GET запросов

/// </summary>

HttpClient client = new HttpClient();

/// <summary>

/// Счетчик для поворота сервомотора

/// </summary>

int flag = 1;

/// <summary>

/// Конечная позиция сервомотора

/// </summary>

public static int finalPosition = 250;

/// <summary>

/// Количество поворотов сервомотора в одну сторону до достижения конечной позиции

/// </summary>

int turnValue = (finalPosition - 50) / 25 - 1;

/// <summary>

/// Задержка между поворотами сервомотора в миллисекундах

/// </summary>

public static int turningTimerDelay = 10000;

#endregion

/// <summary>

/// Обработчик события загрузки формы

/// </summary>

/// <remarks>

/// Отвечает за начальное состояние кнопок и установку интервала таймера

/// </remarks>

private void MainCameraForm\_Load(object sender, EventArgs e)

{

buttons = new Button[] { startCameraButton, stopCameraButton, startRecordingButton,

stopRecordingButton, startMDButton, stopMDButton, takePictureButton, moveLeftButton, moveRightButton, startTurningButton };

cameraButtons = new Button[] { startRecordingButton,

stopRecordingButton, startMDButton, stopMDButton, takePictureButton, moveLeftButton, moveRightButton, startTurningButton };

Array.ForEach(buttons,

x => { x.Enabled = false;

x.TabStop = false; });

turningTimer.Interval = turningTimerDelay;

}

#region Buttons Click

/// <summary>

/// Обработчик события нажатия на кнопку "Connect"

/// </summary>

/// <remarks>

/// Делает кнопки активными и запускает таймер обновления изображения

/// </remarks>

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

{

Array.ForEach(buttons, x => x.Enabled = !x.Enabled);

connectButton.Visible = !connectButton.Visible;

cameraPictureTimer.Enabled = !cameraPictureTimer.Enabled;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Stop Camera"

/// </summary>

/// <remarks>

/// Меняет состояния кнопок и посылает запрос об остановке камеры на сервер

/// </remarks>

private async void stopCameraButton\_Click(object sender, EventArgs e)

{

stopCameraButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=ru%200");

#endif

stopCameraButton.Enabled = true;

Array.ForEach(cameraButtons, x => x.Enabled = !x.Enabled);

stopCameraButton.Visible = false;

startCameraButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Start Camera"

/// </summary>

/// <remarks>

/// Меняет состояния кнопок и посылает запрос о запуске камеры на сервер

/// </remarks>

private async void startCameraButton\_Click(object sender, EventArgs e)

{

startCameraButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=ru%201");

#endif

startCameraButton.Enabled = true;

Array.ForEach(cameraButtons, x => x.Enabled = !x.Enabled);

startCameraButton.Visible = false;

stopCameraButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Stop Recording"

/// </summary>

/// <remarks>

/// Меняет состояния кнопок и посылает запрос об остановке записи на сервер

/// </remarks>

private async void stopRecordingButton\_Click(object sender, EventArgs e)

{

stopRecordingButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=ca%200");

#endif

stopRecordingButton.Enabled = true;

startMDButton.Enabled = true;

stopCameraButton.Enabled = true;

stopRecordingButton.Visible = false;

startRecordingButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Start Recording"

/// </summary>

/// <remarks>

/// Меняет состояния кнопок и посылает запрос о начале записи на сервер

/// </remarks>

private async void startRecordingButton\_Click(object sender, EventArgs e)

{

startRecordingButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=ca%201");

#endif

startRecordingButton.Enabled = true;

startMDButton.Enabled = false;

stopCameraButton.Enabled = false;

startRecordingButton.Visible = false;

stopRecordingButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Stop Motion Detection"

/// </summary>

/// <remarks>

/// Меняет состояния кнопок и посылает запрос об остановке режима Motion Detection на сервер

/// </remarks>

private async void stopMDButton\_Click(object sender, EventArgs e)

{

stopMDButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=md%200");

#endif

stopMDButton.Enabled = true;

startRecordingButton.Enabled = true;

stopCameraButton.Enabled = true;

stopMDButton.Visible = false;

startMDButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Start Motion Detection"

/// </summary>

/// <remarks>

/// Меняет состояния кнопок и посылает запрос о запуске режима Motion Detection на сервер

/// </remarks>

private async void startMDButton\_Click(object sender, EventArgs e)

{

startMDButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=md%201");

#endif

startMDButton.Enabled = true;

startRecordingButton.Enabled = false;

stopCameraButton.Enabled = false;

startMDButton.Visible = false;

stopMDButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Taje Picture"

/// </summary>

/// <remarks>

/// Посылает запрос о съемки фотографии на сервер

/// </remarks>

private async void takePictureButton\_Click(object sender, EventArgs e)

{

takePictureButton.Enabled = false;

#if connection

await client.GetAsync(uri + "html/cmd\_pipe.php?cmd=im");

#endif

takePictureButton.Enabled = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Move Left Button"

/// </summary>

/// <remarks>

/// Посылает запрос о смене состояния сервомотора на сервер

/// </remarks>

private async void moveLeftButton\_Click(object sender, EventArgs e)

{

moveLeftButton.Enabled = false;

#if connection

await client.GetAsync(uri + "servo/turnleft.php");

#endif

moveLeftButton.Enabled = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Move Right Button"

/// </summary>

/// <remarks>

/// Посылает запрос о смене состояния сервомотора на сервер

/// </remarks>

private async void moveRightButton\_Click(object sender, EventArgs e)

{

moveRightButton.Enabled = false;

#if connection

await client.GetAsync(uri + "servo/turnright.php");

#endif

moveRightButton.Enabled = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Start Turning"

/// </summary>

/// <remarks>

/// Меняет интервал таймера и состояние кнопок, отправляет запрос на сервер об изменении положения сервомотора, запускает таймер

/// </remarks>

private async void StartTurningButton\_Click(object sender, EventArgs e)

{

turningTimer.Interval = turningTimerDelay;

startTurningButton.Enabled = false;

#if connection

await client.GetAsync(uri + "servo/initialposition.php");

#endif

startTurningButton.Enabled = true;

startTurningButton.Visible = false;

turningTimer.Enabled = true;

stopTurningButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Stop Turning"

/// </summary>

/// <remarks>

/// Останавливает таймер и меняет состояние кнопок

/// </remarks>

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

{

stopTurningButton.Visible = false;

turningTimer.Enabled = false;

startTurningButton.Visible = true;

}

#endregion

#region Menu Items Click

/// <summary>

/// Обработчик события нажатия на элемент "Settings" меню

/// </summary>

/// <remarks>

/// Открывает форму SettingsForm и меняет некоторые настройки, если была нажата кнопка "Apply"

/// </remarks>

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

{

SettingsForm settingsForm = new SettingsForm();

if (settingsForm.ShowDialog() == DialogResult.OK)

{

uri = settingsForm.settings.Ip;

turningTimerDelay = settingsForm.settings.Delay;

finalPosition = settingsForm.settings.FinalPosition;

fw = settingsForm.settings.Ip;

}

}

/// <summary>

/// Обработчик события нажатия на элемент "Files" меню

/// </summary>

/// <remarks>

/// Открывает форму FilesForm

/// </remarks>

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

{

FilesForm filesForm = new FilesForm();

filesForm.uri = fw;

filesForm.url = fw + "html/preview.php";

filesForm.ShowDialog();

}

/// <summary>

/// Обработчик события нажатия на элемент "About" меню

/// </summary>

/// <remarks>

/// Открывает форму AboutForm

/// </remarks>

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

{

AboutForm aboutForm = new AboutForm();

aboutForm.ShowDialog();

}

/// <summary>

/// Обработчик события нажатия на элемент "Information" меню

/// </summary>

/// <remarks>

/// Открывает форму AboutForm

/// </remarks>

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

{

AboutForm aboutForm = new AboutForm();

aboutForm.ShowDialog();

}

#endregion

#region Timer Ticks

/// <summary>

/// Обработчик события тика таймера

/// </summary>

/// <remarks>

/// Пробует загрузить изображение с камеры, если не удается, деактивирует кнопки и уведомляет пользователя

/// </remarks>

private void cameraPictureTimer\_Tick(object sender, EventArgs e)

{

try

{

#if connection

cameraPictureBox.Load(uri + "html/cam\_pic.php");

#endif

}

catch (Exception)

{

Array.ForEach(buttons, x => x.Enabled = !x.Enabled);

connectButton.Visible = !connectButton.Visible;

cameraPictureTimer.Enabled = !cameraPictureTimer.Enabled;

MessageBox.Show("Can't connect to the server.", "Error");

}

}

/// <summary>

/// Обработчик события тика таймера

/// </summary>

/// <remarks>

/// Отслеживает количество поворотов, отправляется запросы на сервер о повороте сервомотора, меняет направление поворота

/// </remarks>

private async void turningTimer\_Tick(object sender, EventArgs e)

{

#if connection

if(flag == turnValue + 1)

{

flag = -1;

}

if(flag == -(turnValue + 1))

{

flag = 1;

}

if(flag > 0)

{

await client.GetAsync(uri + "servo/turnright.php");

flag++;

}

if(flag < 0)

{

await client.GetAsync(uri + "servo/turnleft.php");

flag--;

}

#endif

}

#endregion

}

}

* 1. Файл FilesForm.cs

using AngleSharp.Html.Dom;

using AngleSharp.Html.Parser;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Windows.Forms;

namespace CourseWork

{

public partial class FilesForm : Form

{

#region Fields

HttpClient client = new HttpClient();

public string uri = "http://192.168.0.101/";

public string url = "http://192.168.0.101/html/preview.php";

GroupBox[] gbArray;

#endregion

public FilesForm()

{

InitializeComponent();

}

#region Methods

/// <summary>

/// Метод загрузки медиа-файлов и заполнения ими flowLayoutPanel1

/// </summary>

/// <remarks>

/// Получает элементы, содержащие данные о файлах, преобразует их в необходимые

/// поля даты, времени, превью и ссылки на файл, создает элементы GroupBox и добавляет их на flowLayoutPanel1

/// </remarks>

private async void FillThePanel()

{

HttpResponseMessage response = null;

string pageSource = "";

HtmlParser pageParser = new HtmlParser();

IHtmlDocument htmlPage = null;

try

{

response = await client.GetAsync(url);

pageSource = await response.Content.ReadAsStringAsync();

htmlPage = await pageParser.ParseDocumentAsync(pageSource);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

Close();

return;

}

Fieldset[] fieldsets = GetFieldsets(htmlPage);

GroupBox[] groupBoxes = new GroupBox[fieldsets.Length];

for (int i = 0; i < groupBoxes.Length; i++)

{

groupBoxes[i] = new GroupBox();

groupBoxes[i].Margin = new Padding(10, 10, 10, 10);

groupBoxes[i].Size = new Size(184, 131);

groupBoxes[i].Text = $" {fieldsets[i].Time} {fieldsets[i].Date} {fieldsets[i].Type}";

groupBoxes[i].Controls.Add(new CheckBox() { Size = new Size(15,15)});

PictureBox pictureBox = new PictureBox() { Location = new Point(2, 19), Size = new Size(180, 101), SizeMode = PictureBoxSizeMode.StretchImage };

try

{

pictureBox.Load(uri + fieldsets[i].Image);

}

catch

{

pictureBox.Load("../../../camerapic.jpg");

}

groupBoxes[i].Controls.Add(pictureBox);

flowLayoutPanel1.Controls.Add(groupBoxes[i]);

}

gbArray = flowLayoutPanel1.Controls.OfType<GroupBox>().ToArray();

for (int i = 0; i < gbArray.Length - 1; i++)

{

gbArray[i] = gbArray[i + 1];

}

Array.Resize(ref gbArray, gbArray.Length - 1);

if(gbArray.Length == 0)

{

flowLayoutPanel1.Controls.Add(new Label { Text = "No media to load." });

}

}

/// <summary>

/// Получает элементы Fieldset

/// </summary>

/// <param name="document">HTML документ</param>

/// <returns>

/// Массив элементов Fieldset

/// </returns>

private static Fieldset[] GetFieldsets(IHtmlDocument document)

{

List<Fieldset> list = new List<Fieldset>();

IEnumerable<AngleSharp.Dom.IElement> items = null;

try

{

items = document.QuerySelectorAll("fieldset");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

}

try

{

foreach (var item in items)

{

list.Add(new Fieldset((IHtmlFieldSetElement)item));

}

}

catch

{

MessageBox.Show("Try again later.", "Error");

}

return list.ToArray();

}

#endregion

/// <summary>

/// Обработчик события загрузки формы

/// </summary>

/// <remarks>

/// Вызывает метод FillThePanel, заполняющий форму

/// </remarks>

private void FilesForm\_Load(object sender, EventArgs e)

{

FillThePanel();

}

#region Buttons Click

/// <summary>

/// Обработчик события нажатия на кнопку "Select All"

/// </summary>

/// <remarks>

/// Помечает свойство Checked всех элементов CheckBox как true

/// </remarks>

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

{

Array.ForEach(gbArray, x => x.Controls.OfType<CheckBox>().First().Checked = true);

selectButton.Visible = false;

deselectButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку "Deselect All"

/// </summary>

/// <remarks>

/// Помечает свойство Checked всех элементов CheckBox как false

/// </remarks>

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

{

Array.ForEach(gbArray, x => x.Controls.OfType<CheckBox>().First().Checked = false);

deselectButton.Visible = false;

selectButton.Visible = true;

}

/// <summary>

/// Обработчик события нажатия на кнопку Download Selected"

/// </summary>

/// <remarks>

/// Выполняет загрузку выбранных файлов в указанный каталог

/// </remarks>

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

{

List<string> filesToDownload = new List<string>();

try

{

for (int i = 0; i < gbArray.Length; i++)

if (gbArray[i].Controls.OfType<CheckBox>().First().Checked)

{

string downloadFileAdress = gbArray[i].Controls.OfType<PictureBox>().First().ImageLocation;

filesToDownload.Add(downloadFileAdress.Substring(0, downloadFileAdress.Length - 13));

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

return;

}

if (!filesToDownload.Any())

{

MessageBox.Show("Files were not selected.", "Error");

return;

}

if (folderBrowserDialog1.ShowDialog() != DialogResult.OK)

return;

WebClient wc = new WebClient();

foreach (var i in filesToDownload)

{

try

{

wc.DownloadFile(i, folderBrowserDialog1.SelectedPath + "/" + i.Split(new string[] { "/media/" }, StringSplitOptions.None)[1]);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

return;

}

}

MessageBox.Show("Download completed successfully.", "Completed");

}

/// <summary>

/// Обработчик события нажатия на кнопку "Delete All"

/// </summary>

/// <remarks>

/// Отправляет POST запрос на сервер об удалении всех файлов и закрывает форму

/// </remarks>

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

{

if (gbArray.Length == 0)

Close();

WebRequest request = null;

try

{

request = WebRequest.Create(url);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

return;

}

request.Method = "POST";

string data = "action=deleteAll";

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

request.ContentType = "application/x-www-form-urlencoded";

request.ContentLength = byteArray.Length;

using (Stream dataStream = request.GetRequestStream())

{

try

{

dataStream.Write(byteArray, 0, byteArray.Length);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

return;

}

}

Close();

}

#endregion

}

}

* 1. Файл SettingsForm.cs

using System;

using System.IO;

using System.Runtime.Serialization.Json;

using System.Windows.Forms;

namespace CourseWork

{

public partial class SettingsForm : Form

{

public SettingsClass settings;

public SettingsForm()

{

InitializeComponent();

}

#region Buttons Click

/// <summary>

/// Обработчик события нажатия на кнопку "Import"

/// </summary>

/// <remarks>

/// Импортирует настройки из выбранного файла

/// </remarks>

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

{

var dialogResult = openFileDialog1.ShowDialog();

if (dialogResult == DialogResult.Abort || dialogResult == DialogResult.Cancel)

return;

DataContractJsonSerializer formatter = new DataContractJsonSerializer(typeof(SettingsClass));

SettingsClass settings = null;

using(FileStream fs = new FileStream(openFileDialog1.FileName, FileMode.Open))

{

try

{

settings = (SettingsClass)formatter.ReadObject(fs);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Import Error");

return;

}

}

MessageBox.Show("Import completed successfully.", "Completed");

delayTextBox.Text = settings.Delay.ToString();

ipTextBox.Text = settings.Ip;

servoTextBox.Text = settings.FinalPosition.ToString();

}

/// <summary>

/// Обработчик события нажатия на кнопку "Export"

/// </summary>

/// <remarks>

/// Экспортирует настройки в выбранный файл

/// </remarks>

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

{

SettingsClass exportSettings = null;

DataContractJsonSerializer formatter = new DataContractJsonSerializer(typeof(SettingsClass));

try

{

exportSettings = new SettingsClass(ipTextBox.Text, delayTextBox.Text, servoTextBox.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Export Error.");

return;

}

if (saveFileDialog1.ShowDialog() != DialogResult.OK)

return;

using (FileStream fs = new FileStream(saveFileDialog1.FileName, FileMode.Create))

{

try

{

formatter.WriteObject(fs, exportSettings);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

return;

}

}

MessageBox.Show("Export completed successfully.", "Completed");

}

/// <summary>

/// Обработчик события нажатия на кнопку "Apply"

/// </summary>

/// <remarks>

/// Инициализирует объект класса SettingsClass, возвращает OK в качестве DialogResult и закрывает форму

/// </remarks>

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

{

try

{

settings = new SettingsClass("http://" + ipTextBox.Text + "/", delayTextBox.Text, servoTextBox.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

return;

}

DialogResult = DialogResult.OK;

Close();

}

#endregion

}

}

* 1. Файл Fieldset.cs

using AngleSharp.Html.Dom;

using System;

namespace CourseWork

{

public class Fieldset

{

public Fieldset(IHtmlFieldSetElement element)

{

Type = GetType(element);

Date = GetDate(element);

Time = GetTime(element);

Image = GetImageSource(element);

}

private string GetType(IHtmlFieldSetElement element)

{

string content = element.TextContent;

string[] arr = content.Split(new string[] { " " }, StringSplitOptions.None);

if (arr[1].Length == 18)

return "Image";

return "Video";

}

private string GetDate(IHtmlFieldSetElement element)

{

string content = element.TextContent;

string[] arr = content.Split(new string[] { " " }, StringSplitOptions.None);

string dateAndTime = arr[1].Substring(arr[1].Length - 18, 18);

return dateAndTime.Substring(0, 10);

}

private string GetTime(IHtmlFieldSetElement element)

{

string content = element.TextContent;

string[] arr = content.Split(new string[] { " " }, StringSplitOptions.None);

string dateAndTime = arr[1].Substring(arr[1].Length - 18, 18);

return dateAndTime.Substring(10);

}

private string GetImageSource(IHtmlFieldSetElement element)

{

string source = ((IHtmlImageElement)(element.LastElementChild.FirstElementChild)).Source;

return "html/" + source.Substring(9);

}

public string Date { get; private set; }

public string Time { get; private set; }

public string Type { get; private set; }

public string Image { get; private set; }

}

}

* 1. Файл SettingsClass.cs

using System;

using System.Runtime.Serialization;

namespace CourseWork

{

[DataContract]

public class SettingsClass

{

/// <summary>

/// IP Raspberry Pi

/// </summary>

[DataMember]

public string Ip { get; private set; }

/// <summary>

/// Задержка между автоматическими поворотами сервомотора

/// </summary>

[DataMember]

public int Delay { get; private set; }

/// <summary>

/// Конечная позиция сервомотора

/// </summary>

[DataMember]

public int FinalPosition { get; private set; }

/// <summary>

/// Конструктор класса SettingClass

/// </summary>

/// <remarks>

/// Пытается инициировать свойства класса переданными значениями

/// </remarks>

/// <exception cref="ArgumentException">

/// Can't parse Delay or Final Position to integer.

/// </exception>

/// <exception cref="ArgumentException">

/// Delay was less than 0.

/// </exception>

/// <exception cref="ArgumentException">

/// Final Position was less than 75 or more than 250.

/// </exception>

/// <param name="ip">IP Raspberry Pi</param>

/// <param name="delay">Задержка между автоматическими поворотами сервомотора</param>

/// <param name="finalPosition">Конечная позиция сервомотора</param>

public SettingsClass(string ip, string delay, string finalPosition)

{

int intDelay, intFinalPosition;

if (!(int.TryParse(delay, out intDelay) && int.TryParse(finalPosition, out intFinalPosition)))

throw new ArgumentException("Can't parse Delay or Final Position to integer.");

if (intDelay <= 0)

throw new ArgumentException("Delay was less than 0.");

if (intFinalPosition < 75 || intFinalPosition > 250)

throw new ArgumentException("Final Position was less than 75 or more than 250.");

Ip = ip;

Delay = intDelay;

FinalPosition = intFinalPosition;

}

/// <summary>

/// Переопределенный метод ToString

/// </summary>

/// <returns>Экземпляр класса, приведенный к строке</returns>

public override string ToString()

{

return $"{Ip} {Delay} {FinalPosition}";

}

}

}

**ПРИЛОЖЕНИЕ 1**

1. **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**
2. ГОСТ 19.102-77 Стадии разработки. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.101-77 Виды программ и программных документов. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.104-78 Основные надписи. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.105-78 Общие требования к программным документам. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.603-78 Общие правила внесения изменений. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
12. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированых |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |