updateWorkLoads - обновляет переменные из json

ShowListViewGroup - отображает список групп

FormShedule\_Load -

ShowAudiences - отображает список аудиторий

listViewGroup\_MouseClick - при нажатии на группу заполняется таблица с расписанием и отображаюся нагрузки преподователей в листе преподователей

Save - сохраняем изменения расписания у данной группы

SaveFileToolStripMenuItem\_Click - сохраняем в эксель

DisciplineCheck(не нужна пока) - убирает из списка предметов те которые были перетянуты в расписание

ShowSubjects - показ списка преподов и предметов

ShowShedule - показ расписания

ListViewItemToString - ту стринг

dataGridViewShedule\_SelectionChanged

HasLectorFreeHours - проверяет есть ли у препода свободные часы

ShowAudienceDescription - показ описания данной аудитории

.........куча функций отвечающих за drag and drop

dataGridViewAudience\_MouseDown - отображение описания аудитории и dad

dataGridViewAudience\_DragEnter

dataGridViewAudience\_DragDrop

listViewSubjects\_MouseDown -

listViewSubjects\_DragDrop -

listViewSubjects\_DragEnter -

dataGridViewShedule\_MouseDown

dataGridViewShedule\_DragDrop -

dataGridViewShedule\_DragEnter

.............

AudiencesForm\_Click - переключение на вкладку с аудиториями

ShedulesForm\_Click - переключение на вкладку с аудиториями

AddLoadsToolStripMenuItem\_Click - подгружаем нагрузки

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Reflection;

using System.Windows.Forms;

//using System.Windows.Media;

namespace Schedule\_Editor

{

public partial class ScheduleEditor : Form

{

ListTeachers AllTeachers;

ListSubgroupSchedule AllScheduleGroup;

AudienceGroup AllAudiences;

ListGroups AllGroups;

string ActiveGroup;

string activeDiscipline = "";

int formwidth;

int formheight;

string curDir = Environment.CurrentDirectory;

int CountOfWeeksInYear = 39;

int CountOfWeeksIn1Semester = 18;

public ScheduleEditor()

{

InitializeComponent();

this.WindowState = FormWindowState.Maximized;

dataGridViewShedule.Height = formheight = this.Height - 40;

dataGridViewShedule.Width = formwidth = this.Width;

dataGridViewShedule.RowTemplate.Height = 46;

dataGridViewShedule.RowCount = 20;

dataGridViewShedule.ColumnHeadersHeight = 40;

dataGridViewShedule.ColumnCount = 2;

dataGridViewShedule.Columns[1].Width = 100;

dataGridViewShedule.Columns[1].HeaderText = "Аудитория";

string[] weekDays = { "Пн", "Вт", "Ср", "Чт", "Пт" };

for (int i = 0; i < 20; i += 4)

{

dataGridViewShedule.Rows[i].HeaderCell.Value = weekDays[i / 4];

}

for (int i = 0; i < dataGridViewShedule.Columns.Count; i++)

{

dataGridViewShedule.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;

}

for (int col = 0; col < dataGridViewShedule.Columns.Count; col++)

{

for (int row = 0; row < dataGridViewShedule.Rows.Count; row++)

{

dataGridViewShedule[col, row].Value = "";

}

}

//dataGridViewShedule.DefaultCellStyle.SelectionBackColor = dataGridViewShedule.DefaultCellStyle.BackColor;

listViewSubjects.Columns.Add("Дисциплина");

listViewSubjects.Columns.Add("Преподователь");

listViewSubjects.Columns.Add("Тип занятия");

listViewSubjects.Columns.Add("Кол-во часов");

listViewSubjects.Columns[0].Width = 220;

listViewSubjects.Columns[1].Width = 150;

listViewSubjects.Columns[2].Width = 150;

listViewSubjects.Columns[3].Width = 150;

listViewSubjects.Font = new Font(FontFamily.GenericSansSerif, 12);

//listViewGroup.Font = new System.Drawing.Font(FontFamily.GenericSansSerif, 12);

//убираем мерцание и свойства выделения

listViewSubjects.HoverSelection = false;

listViewSubjects.FullRowSelect = true;

Type type = listViewSubjects.GetType();

PropertyInfo propertyInfo = type.GetProperty("DoubleBuffered", BindingFlags.NonPublic | BindingFlags.Instance);

propertyInfo.SetValue(listViewSubjects, true, null);

foreach (DataGridViewRow row in dataGridViewAudience.Rows)

{

row.Height = 40;

}

dataGridViewAudience.RowCount = 5;

dataGridViewAudience.ColumnCount = 10;

dataGridViewAudience.BackgroundColor = Color.White;

listViewAudienceDescription.Height = 350;

foreach (var item in new string[] { "Номер", "Количество мест", "Меловая доска", "Маркерная доска", "Количество компьютеров", "Проектор" })

{

listViewAudienceDescription.Columns.Add(item, 120);

}

}

void updateWorkLoads()

{

using (StreamReader file = new StreamReader(curDir + @"\..\..\Files\subgroupShedule.json"))

{

AllScheduleGroup = JsonConvert.DeserializeObject<ListSubgroupSchedule>(file.ReadToEnd());

}

using (StreamReader file = new StreamReader(curDir + @"\..\..\Files\loads.json"))

{

AllTeachers = JsonConvert.DeserializeObject<ListTeachers>(file.ReadToEnd());

}

using (StreamReader file = new StreamReader(curDir + @"\..\..\Files\groups.json"))

{

AllGroups = JsonConvert.DeserializeObject<ListGroups>(file.ReadToEnd());

}

using (StreamReader file = new StreamReader(curDir + @"\..\..\Files\audienceGroup.json"))

{

AllAudiences = JsonConvert.DeserializeObject<AudienceGroup>(file.ReadToEnd());

}

}

void ShowListViewGroup()

{

listViewGroup.Items.Clear();

int year = DateTime.Now.Year - 2000;

listViewGroup.View = View.Tile;

for (int i = 1; i < 5; i++)

{

listViewGroup.Groups.Add(new ListViewGroup($"{i} курс"));

}

listViewGroup.HeaderStyle = ColumnHeaderStyle.Nonclickable;

foreach (var item in AllGroups.Groups)

{

int groupInd = year - 1 - Convert.ToInt32(item.Name.Split('-')[1]);

ListViewItem group = new ListViewItem(item.Name);

listViewGroup.Items.Add(group);

listViewGroup.Groups[groupInd].Items.Add(group);

}

}

// считываем данные с файлов и заполняем лист групп

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

{

var infFileShGroup = new FileInfo(curDir + @"\..\..\Files\subgroupShedule.json");

if (infFileShGroup.Length == 0)

AddLoads.GenerateNewLoads();

updateWorkLoads();

ShowListViewGroup();

dataGridViewAudience.Hide();

listViewAudienceDescription.Hide();

ShowAudiences();

}

// при нажатии на группу заполняется таблица с расписанием и отображаюся нагрузки преподователей в листе преподователей

private void listViewGroup\_MouseClick(object sender, MouseEventArgs e)

{

try

{

ActiveGroup = listViewGroup.SelectedItems[0].Text;

dataGridViewShedule.Columns[0].HeaderText = ActiveGroup;

ShowShedule();

ShowSubjects();

//DisciplineCheck();

}

catch (Exception)

{ }

}

void Save()

{

if (ActiveGroup == null) return;

foreach (var subGrouSchedule in AllScheduleGroup.Shedule)

{

if (subGrouSchedule.Name == ActiveGroup)

{

for (int i = 0; i < dataGridViewShedule.Rows.Count; i++)

{

subGrouSchedule.ScheduleFieldsSubjects[i] = dataGridViewShedule.Rows[i].Cells[0].Value.ToString();

subGrouSchedule.ScheduleFieldsAudiences[i] = dataGridViewShedule.Rows[i].Cells[1].Value.ToString();

}

break;

}

}

using (StreamWriter sw = new StreamWriter(curDir + @"\..\..\Files\subgroupShedule.json"))

sw.WriteLine(JsonConvert.SerializeObject(AllScheduleGroup));

//DisciplineCheck();

}

}

public string ShowDialog1(string text, string caption)

{//тут

Form prompt = new Form()

{

Width = 500,

Height = 150,

FormBorderStyle = FormBorderStyle.FixedDialog,

Text = caption,

StartPosition = FormStartPosition.CenterScreen

};

Label textLabel = new Label() { Left = 50, Top = 20, Text = text };

TextBox textBox = new TextBox() { Left = 50, Top = 50, Width = 400 };

Button confirmation = new Button() { Text = "Ok", Left = 350, Width = 100, Top = 70, DialogResult = DialogResult.OK };

confirmation.Click += (sender, e) => { prompt.Close(); };

prompt.Controls.Add(textBox);

prompt.Controls.Add(confirmation);

prompt.Controls.Add(textLabel);

prompt.AcceptButton = confirmation;

return prompt.ShowDialog() == DialogResult.OK ? textBox.Text : "";

}

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

{

if (!AllScheduleGroup.IsScheduleFilled()) ShowDialog1("Lana", "Stas");

SaveSchedule.Save();

}

private void DisciplineCheck()

{

for (int i = 0; i < dataGridViewShedule.Rows.Count; i++)

{

string s = dataGridViewShedule.Rows[i].Cells[0].Value.ToString();

for (int j = 0; j < listViewSubjects.Items.Count; j++)

{

string t = ListViewItemToString(listViewSubjects.Items[j]);

if (s == t)

{

listViewSubjects.Items.RemoveAt(j);

}

}

}

}

// функция отображения аудиторий в его датагриде

void ShowAudiences()

{

for (int ind = 0, row = 0, col = 0; ind < AllAudiences.Count; ind++)

{

dataGridViewAudience[col, row].Value = AllAudiences.Audiences[ind].Number;

col++;

if (col == dataGridViewAudience.Columns.Count)

{

col = 0;

row++;

}

}

}

private void ShowSubjects()

{

listViewSubjects.Items.Clear();

foreach (var teacher in AllTeachers.Teachers)

{

foreach (var sub in teacher.Subjects.Items)

{

if (sub.Group == ActiveGroup)

{

ListViewItem lvi = new ListViewItem(sub.Name);

lvi.SubItems.Add(teacher.LastName + " " + teacher.FirstName);

lvi.SubItems.Add(sub.ClassForm);

lvi.SubItems.Add(sub.NumberOfHours.ToString());

listViewSubjects.Items.Add(lvi);

}

}

}

}

private void ShowShedule()

{

foreach (var item in AllScheduleGroup.Shedule)

{

if (item.Name == ActiveGroup)

{

for (int i = 0; i < dataGridViewShedule.Rows.Count; i++)

{

dataGridViewShedule.Rows[i].Cells[0].Value = item.ScheduleFieldsSubjects[i];

dataGridViewShedule.Rows[i].Cells[1].Value = item.ScheduleFieldsAudiences[i];

}

break;

}

}

}

private string ListViewItemToString(ListViewItem item)

{

return $"{item.SubItems[0].Text} {item.SubItems[1].Text} {item.SubItems[2].Text}";

}

private void dataGridViewShedule\_SelectionChanged(object sender, EventArgs e)

{

this.dataGridViewShedule.ClearSelection();

}

public bool HasLectorFreeHours(string lector)

{

foreach (ListViewItem item in listViewSubjects.Items)

{

if (ListViewItemToString(item) == lector)

{

return Convert.ToInt32(item.SubItems[3].Text) >= CountOfWeeksIn1Semester;//Тут

}

}

return false;

}

void ShowAudienceDescription(string aud)

{

if (!string.IsNullOrEmpty(aud))

{

foreach (var item in AllAudiences.Audiences)

{

if (item.Number.ToString() == aud)

{

listViewAudienceDescription.Items.Clear();

ListViewItem listViewItem = new ListViewItem(aud);

listViewItem.SubItems.Add(item.CountOfSeats.ToString());

listViewItem.SubItems.Add(item.ChalkBoard ? "Есть" : "Нет");

listViewItem.SubItems.Add(item.MarkerBoard ? "Есть" : "Нет");

listViewItem.SubItems.Add(item.NumberOfComputers.ToString());

listViewItem.SubItems.Add(item.Projector ? "Есть" : "Нет");

listViewAudienceDescription.Items.Add(listViewItem);

}

}

}

}

private void dataGridViewAudience\_MouseDown(object sender, MouseEventArgs e)

{

try

{

DataGridView.HitTestInfo info = dataGridViewAudience.HitTest(e.X, e.Y);

string aud = dataGridViewAudience[info.ColumnIndex, info.RowIndex].Value.ToString();

ShowAudienceDescription(aud);

dataGridViewShedule.DoDragDrop(aud, DragDropEffects.Copy);

}

catch (Exception)

{ }

}

private void dataGridViewAudience\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.Text))

{

e.Effect = DragDropEffects.Copy;

}

}

private void dataGridViewAudience\_DragDrop(object sender, DragEventArgs e)

{

try

{

string cellvalue = e.Data.GetData(typeof(string)) as string;

Point cursorLocation = this.PointToClient(new Point(e.X, e.Y));

DataGridView.HitTestInfo hittest = dataGridViewAudience.HitTest(cursorLocation.X, cursorLocation.Y - 20);

if (hittest.ColumnIndex != -1

&& hittest.RowIndex != -1)

dataGridViewAudience[hittest.ColumnIndex, hittest.RowIndex].Value = cellvalue;

Save();

}

catch

{ }

}

private void listViewSubjects\_MouseDown(object sender, MouseEventArgs e)

{

try

{

int indexSource = listViewSubjects.Items.IndexOf(listViewSubjects.GetItemAt(e.X, e.Y));

string s = ListViewItemToString(listViewSubjects.Items[indexSource]);

listViewSubjects.DoDragDrop(s, DragDropEffects.Copy);

}

catch

{ }

}

private void listViewSubjects\_DragDrop(object sender, DragEventArgs e)

{

activeDiscipline = "";

ShowSubjects();

//DisciplineCheck();

}

private void listViewSubjects\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.Text))

{

e.Effect = DragDropEffects.Copy;

}

}

private void dataGridViewShedule\_MouseDown(object sender, MouseEventArgs e)

{

try

{

DataGridView.HitTestInfo info = dataGridViewShedule.HitTest(e.X, e.Y);

string s = dataGridViewShedule[info.ColumnIndex, info.RowIndex].Value.ToString();

if (!string.IsNullOrEmpty(s))

{

dataGridViewShedule.DoDragDrop(s, DragDropEffects.Copy);

dataGridViewShedule[info.ColumnIndex, info.RowIndex].Value = activeDiscipline;

activeDiscipline = "";

listViewSubjects.DoDragDrop(s, DragDropEffects.Copy);

dataGridViewAudience.DoDragDrop(s, DragDropEffects.Copy);

Save();

}

}

catch (Exception)

{ }

}

private void dataGridViewShedule\_DragDrop(object sender, DragEventArgs e)

{

try

{

string cellvalue = e.Data.GetData(typeof(string)) as string;

Point cursorLocation = this.PointToClient(new Point(e.X, e.Y));

DataGridView.HitTestInfo hittest = dataGridViewShedule.HitTest(cursorLocation.X, cursorLocation.Y - 24);

//MessageBox.Show(HasLectorFreeHours(cellvalue).ToString());

if (hittest.RowIndex != -1)

{

bool IsAudience = int.TryParse(cellvalue, out int \_) && hittest.ColumnIndex == 1;

bool IsLector = !int.TryParse(cellvalue, out int \_) && hittest.ColumnIndex == 0;

if (IsAudience && AllScheduleGroup.IsAudienceEmpty(cellvalue, hittest.RowIndex) ||

IsLector && AllScheduleGroup.IsLectorFree(cellvalue, hittest.RowIndex) && HasLectorFreeHours(cellvalue))

{

activeDiscipline = dataGridViewShedule[hittest.ColumnIndex, hittest.RowIndex].Value.ToString();

dataGridViewShedule[hittest.ColumnIndex, hittest.RowIndex].Value = cellvalue;

}

}

ShowSubjects();

//DisciplineCheck();

Save();

}

catch

{ }

}

private void dataGridViewShedule\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.Text))

{

e.Effect = DragDropEffects.Copy;

}

}

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

{

listViewSubjects.Hide();

dataGridViewAudience.Show();

listViewAudienceDescription.Show();

}

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

{

listViewSubjects.Show();

dataGridViewAudience.Hide();

listViewAudienceDescription.Hide();

}

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

{

AddLoads.GenerateNewLoads();

updateWorkLoads();

ShowListViewGroup();

}

}

}