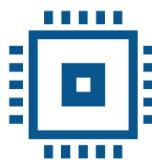




Universitatea *Transilvania* din Brașov  
Facultatea de Inginerie Electrică și Știința Calculatoarelor  
Departamentul Automatică și Tehnologia Informației



# PROIECT

## ORAR UNITBV

Profesor      îndrumător:      Conferențiar      universitar ,  
KRISTÁLY Dominic Mircea

*Grabovenco Bogdan-Iulian*

BRAȘOV, 2024

## Cuprins

1.	Orar UNITBV .....	2
1.1.	Enunt.....	2
1.2.	Rezolvare Proiect .....	2
1.3.	MS Excel.....	2
1.4.	Portarea și regulile de portare .....	3
1.5.	Aplicația Windows Form (C#).....	4
1.6.	XAMPP Service .....	36
1.7.	Dezvoltare WEB Frontend .....	37
1.8.	Manual de utilizare .....	58
1.9.	Posibilități de îmbunătățire.....	63

## 1. Orar UNITBV

### 1.1. Enunt

Să se realizeze o aplicație care să implementeze toate funcțiile necesare unei portări din mediul MS Excel în mediul de server MS SQL cu GUI web. Aplicația va oferi modulele:

1. Creare entități
2. Manipulare(vizualizare, alterare) entități
3. Folosirea unui GUI web pentru vizualizare

Aplicația poate conține orice alte module care sunt considerate a fi necesare.

### 1.2. Rezolvare Proiect

Prezentarea proiectului se va face etapizat în ordinea în care s-au abordat problemele, și anume ordinea ierarhizată care începe cu analizare structurii worksheet-ului MS Excel, apoi generarea unor reguli de extragere a datelor aflate în entitatea Excel pe baza cărora să fie posibilă portarea într-un mediu de server cu o bază de date reală reprezentată de MS SQL. Ultimul pas este crearea cu ajutorul managerul de fișiere .php XAMPP unei interfețe (GUI) pentru interogarea acestei baze de date în mod extern, fără a folosi softwareul dedicat, adică SQL Server Management Studio. Cu alte cuvinte, prima parte o constituie înțelegerea structurii, crearea unor reguli de extragere pentru generarea entităților, apoi vizualizarea rezultatelor printr-o interfață, acesta fiind a doua parte.

### 1.3. MS Excel

Majoritatea foilor de lucru (worksheet) sunt create pe aceleasi reguli, diferențe însă pot exista între diferitele facultăți din cadrul Universității "Transilvania". Orarul IESC este însă cel pe baza căruia s-au generat toate regulile de extragere a datelor. Menționez aici că orarul este salvat într-un fișier de extensie mai veche a MS Excel, fiind astfel necesară o conversie înainte de extragerea datelor. Aplicația de tip formular permite conversia fișierelor, dar permite și lucrul direct cu cea mai nouă versiune de Excel. Aici se va ataşa o imagine cu foaia de lucru pentru a evidenția aspectul acesteia.

FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR -												
LUNI												
1	Anul	P.S.	Grupa	Sgr.	8,00-9,50	10,00-11,50	12,00-13,50	14,00-15,50	16,00-17,50	18,00-19,50	20,00-21,50	8,00-9,50
4			A				Graf., L, NIIS, Musuroi_C_L					
5			4LF131						MatemS, S, NIIS, Dimitriu_M			
6			B				Graf., L, NIIS, Musuroi_C_L					ProgII., C, NIIS, Ghita_D_E
7	1	ET										
8			A					Graf., L, NIIS, Musuroi_C_L				
9			4LF132		Engleză, S, NIIS, Sasu_L_E			MatemS, S, NIIS, Dimitriu_M				ProgII., C, NIIS, Ghita_D_E
10			B		Engleză, S, NIIS, Sasu_L_E				Graf., L, I04_P_SC Motoasca_S_D			
11												

Fig 1.3 Exemplu de foaie de lucru (IESC)

#### 1.4. Portarea și regulile de portare

Portarea s-a realizat în cadrul unui Windows Form App pe baza unor reguli experimentale, care pot fi supuse greșelii pentru că orarul unei facultăți seamănă în linii mari cu orarul altei facultăți, dar pot exista diferențe majore care să împiedice aplicația concepută din a rula corect. O diferență majoră sesizată între câteva facultăți este prezența multiplă a titulaturii facultății respective în cadrul aceleiași foi de lucru.

Cea mai importantă regulă pe care o putem desprinde din foaia de lucru este că spațiile concatenate sunt, de fapt, un artificiu vizual pentru a putea mări claritatea, însă celule concatenate rămân în realitate nule, dintre mai multe celule concatenate celula cu indecșii mai mici este celula care conține informația, iar celelalte nu sunt decât ascunse pentru o vizibilitatea sporită. În imaginea de mai sus sirul de caractere "LUNI" este conținut doar în celula de linie 1 și coloană E, adică 5, celelalte coloane până la L, adică 12, sunt nule, lipsite de informație. Acest aspect este în mod evident valabil și pentru coloane.

Următoarea regulă ca importanță în performanța aplicației este închiderea instanțelor de Excel, fiecare citire de celulă se face prin instantierea unui obiect de tip Excel. După cum remarcam și în rezumatele concepute, supra-aglomerarea cu instanțe Excel pot face ca aplicația să cedeze, dar mai important, pot face ca sistemul să rămână fără memorie și să se prăbușească, rezultând într-un Force Shut Down de sistem, aspect cu totul indezirabil care licitează ca performanța regula de mai sus, dar care gestionat corespunzător, adică ștergerea instanțelor sau, după cum se mai numesc, "deinstantieri".

Altă regulă este numărarea liniilor și coloanelor de maxim. Obiectul de tip Excel permite utilizarea unei funcții de aflare a maximului pentru linii și coloane, însă metoda nu are cum să fie folosită în varianta menționată, o foaie de lucru poate avea sute de mii de linii și coloane, făcând imposibilă parcurgerea foii de lucru. În schimb pentru aflarea unor bariere de lucru s-a implementat un algoritm care numără distanța între celule, iar dacă o distanță este mai mare de 10 sau 15 celule cu informație, atunci acolo se poate impune o limită, caz valabil doar pentru linii, la coloane limitele se știu pentru că se respectă un plan de organizare săptămânal, iar de aici se deduce ușor numărul liniilor.

## 1.5. Aplicația Windows Form (C#)

Portarea s-a făcut după regulile menționate, regulile deduse empiric, de la particular la general, după cum am menționat, reguli care sunt supuse greșelii din moment ce facultățiile nu au o coeziune de redactarea a orarului în MS Excel. În continuare se vor prezenta secvențele de cod scrise în limbajul C# pentru aplicația de formular. Pentru inserare de date s-a folosit o clasă denumită Schedule.cs pentru evidențierea procesului de inserție. Codul conține și explicații în limba engleză pentru o înțelegere mai bună a liniilor de cod. Explicațiile au fost scrise de mine.

*Windows Form - Secvență de cod 1 Excel.cs*

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Diagnostics.Metrics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Office.Interop.Excel;
using _Excel = Microsoft.Office.Interop.Excel;

namespace OrarUnitbv
{
    class Excel
    {
        string path = "";
        _Application excel = new _Excel.Application();
        Workbook wb;
        Worksheet ws;
        public Excel(string path, int sheet)
        {
            this.path = path;
            wb = excel.Workbooks.Open(path);
            ws = wb.Worksheets[sheet];
        }
        public Excel()
        {

        }
        public string ReadCell(int i, int j)
        {
            try
            {
                return ws.Cells[i, j].Value2.ToString();
            }
        }
    }
}
```

```
    }
    catch
    {
        //Debug.WriteLine("\n" + "catch in Excel object via ReadCell method" + "\n");
        return "Null cell!";
    }
}

internal string ConvertXlsToXlsx(string xlsFilePath)
{
    try
    {
        excel.Visible = false;
        // opening the .xls file
        wb = excel.Workbooks.Open(xlsFilePath);
        // path for the new file
        string xlsxFilePath = System.IO.Path.ChangeExtension(xlsFilePath, ".xlsx");
        // save the file as .xlsx with additional parameters to avoid issues
        wb.SaveAs(xlsxFilePath,
            FileFormat: _Excel.XlFileFormat.xlOpenXMLWorkbook,
            AccessMode: _Excel.XlSaveAsAccessMode.xlNoChange,
            ConflictResolution: _Excel.XlSaveConflictResolution.xlLocalSessionChanges);
        //Close();
        return xlsxFilePath;
    }
    catch (Exception ex)
    {
        MessageBox.Show("A apărut o eroare de conversie!", "Eroare", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        Debug.WriteLine("Excel.class error: \n" + ex.Message + "\n");
        return "";
    }
}

public int MaxRows()
{
    return ws.Rows.Count;
}

public int MaxColumns()
{
    return ws.Columns.Count;
}

public void Close()
{
    wb.Close(false); // Close the workbook without saving changes
}
```

```
excel.Quit(); // Quit the Excel application

// Release the COM objects to fully kill the Excel process from running in the background
System.Runtime.InteropServices.Marshal.ReleaseComObject(ws);
System.Runtime.InteropServices.Marshal.ReleaseComObject(wb);
System.Runtime.InteropServices.Marshal.ReleaseComObject(excel);

ws = null;
wb = null;
excel = null;

GC.Collect(); // Force garbage collection to clean up the COM objects
GC.WaitForPendingFinalizers();
}

}

}
```

*Windows Form - Secvență de cod 2 Schedule.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OrarUnitbv
{
    internal class Schedule
    {
        // follows the setup of the entity to create
        private string tab, abvr, type, prof, day,
            time, place, stds;

        internal Schedule(string tabi, string abvri, string typei, string profi, string dayi, string timei, string placei,
            string stdsi)
        {
            // constructor
            this.tab = tabi;
            this.abvr = abvri;
            this.type = typei;
            this.prof = profi;
            this.day = dayi;
            this.time = timei;
            this.place = placei;
            this.stds = stdsi;
        }
    }
}
```

```
}

internal Schedule()
{

}

internal void setTab(string tabi) { this.tab = tabi; }
internal void setAbvr(string abvri) { this.abvr = abvri; }
internal void setType(string typei) { this.type = typei; }
internal void setProf(string profi) { this.prof = profi; }
internal void setDay(string dayi) { this.day = dayi; }
internal void setTime(string timei) { this.time = timei; }
internal void setPlace(string placei) { this.place = placei; }
internal void setStds(string stdsi) { this.stds = stdsi; }
internal String getTab() { return this.tab; }
internal String getAbvr() { return this.abvr; }
internal String getType() { return this.type; }
internal String getProf() { return this.prof; }
internal String getDay() { return this.day; }
internal String getTime() { return this.time; }
internal String getPlace() { return this.place; }
internal String getStds() { return this.stds; }
internal String To_String()
{
    return "Schedule " + "(tab: " + this.tab + ",\n" +
           "abvr: " + this.abvr + ",\n" +
           "type: " + this.type + ",\n" +
           "prof: " + this.prof + ",\n" +
           "day: " + this.day + ",\n" +
           "time: " + this.time + ",\n" +
           "place: " + this.place + ",\n" +
           "stds: " + this.stds + ")";
}
}
```

*Windows Form - Secvență de cod 3 Form1.Designer.cs*

```
using Microsoft.Azure.Amqp.Framing;
using System.Windows.Forms;

namespace OrarUnitbv
{
    partial class Form1
    {
        /// <summary>
```

```
/// Required designer variable.  
/// </summary>  
private System.ComponentModel.IContainer components = null;  
  
/// <summary>  
/// Clean up any resources being used.  
/// </summary>  
/// <param name="disposing">true if managed resources should be disposed; otherwise,  
false.</param>  
protected override void Dispose(bool disposing)  
{  
    if (disposing && (components != null))  
    {  
        components.Dispose();  
    }  
    base.Dispose(disposing);  
}  
  
#region Windows Form Designer generated code  
  
/// <summary>  
/// Required method for Designer support – do not modify  
/// the contents of this method with the code editor.  
/// </summary>  
private void InitializeComponent()  
{  
    components = new System.ComponentModel.Container();  
    System.ComponentModel.ComponentResourceManager resources = new  
System.ComponentModel.ComponentResourceManager(typeof(Form1));  
    load_path_btn = new Button();  
    open_path_btn = new Button();  
    path_tb = new TextBox();  
    above_cb_lbl = new Label();  
    insert_btn = new Button();  
    deps_cb = new ComboBox();  
    help_msg = new System.Windows.Forms.Timer(components);  
    del_btn = new Button();  
    above_path_lbl = new Label();  
    help_path = new System.Windows.Forms.Timer(components);  
    pnl = new Panel();  
    pictureBox1 = new PictureBox();  
    pcbClose = new PictureBox();  
    pcbMin = new PictureBox();  
    contact_lbl = new Label();  
    pnl.SuspendLayout();
```

```
((System.ComponentModel.ISupportInitialize)pictureBox1).BeginInit();
((System.ComponentModel.ISupportInitialize)pcbClose).BeginInit();
((System.ComponentModel.ISupportInitialize)pcbMin).BeginInit();
SuspendLayout();
//
// load_path_btn
//
load_path_btn.BackColor = Color.FromArgb(224, 224, 224);
load_path_btn.ForeColor = SystemColors.MenuHighlight;
load_path_btn.Location = new Point(254, 148);
load_path_btn.Name = "load_path_btn";
load_path_btn.Size = new Size(212, 37);
load_path_btn.TabIndex = 2;
load_path_btn.Text = "încarcă fișierul ales";
load_path_btn.UseVisualStyleBackColor = false;
load_path_btn.Click += load_path_btn_Click;
//
// open_path_btn
//
open_path_btn.BackColor = SystemColors.MenuHighlight;
open_path_btn.ForeColor = Color.FromArgb(224, 224, 224);
open_path_btn.Location = new Point(82, 148);
open_path_btn.Name = "open_path_btn";
open_path_btn.Size = new Size(167, 37);
open_path_btn.TabIndex = 12;
open_path_btn.Text = "alege fișierul Excel";
open_path_btn.UseVisualStyleBackColor = false;
open_path_btn.Click += open_path_btn_Click;
//
// path_tb
//
path_tb.Location = new Point(82, 109);
path_tb.Name = "path_tb";
path_tb.Size = new Size(385, 33);
path_tb.TabIndex = 13;
path_tb.MouseEnter += path_tb_MouseEnter;
path_tb.MouseLeave += path_tb_MouseLeave;
//
// above_cb_lbl
//
above_cb_lbl.AutoSize = true;
above_cb_lbl.BackColor = Color.White;
above_cb_lbl.BorderStyle = BorderStyle.Fixed3D;
above_cb_lbl.Location = new Point(82, 200);
above_cb_lbl.Name = "above_cb_lbl";
```

```
above_cb_lbl.Size = new Size(385, 28);
above_cb_lbl.TabIndex = 15;
above_cb_lbl.Text = "Alegeți un departament dintre cele înșiruite în listă.";
above_cb_lbl.Visible = false;
//
// insert_btn
//
insert_btn.BackColor = Color.LimeGreen;
insert_btn.ForeColor = Color.FromArgb(224, 224, 224);
insert_btn.Location = new Point(280, 280);
insert_btn.Name = "insert_btn";
insert_btn.Size = new Size(187, 37);
insert_btn.TabIndex = 41;
insert_btn.Text = "adaugă";
insert_btn.UseVisualStyleBackColor = false;
insert_btn.Click += insert_btn_Click;
//
// deps_cb
//
deps_cb.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
deps_cb.AutoCompleteSource = AutoCompleteSource.ListItems;
deps_cb.FormattingEnabled = true;
deps_cb.Location = new Point(82, 240);
deps_cb.Name = "deps_cb";
deps_cb.Size = new Size(385, 34);
deps_cb.TabIndex = 42;
deps_cb.MouseEnter += deps_cb_MouseEnter;
deps_cb.MouseLeave += deps_cb_MouseLeave;
//
// help_msg
//
help_msg.Enabled = true;
help_msg.Interval = 2500;
help_msg.Tick += help_msg_TimerTick;
//
// del_btn
//
del_btn.BackColor = Color.FromArgb(224, 224, 224);
del_btn.ForeColor = Color.LimeGreen;
del_btn.Location = new Point(82, 280);
del_btn.Name = "del_btn";
del_btn.Size = new Size(192, 37);
del_btn.TabIndex = 43;
del_btn.Text = "șterge";
del_btn.UseVisualStyleBackColor = false;
```

```
del_btn.Click += del_btn_Click;
//
// above_path_lbl
//
above_path_lbl.AutoSize = true;
above_path_lbl.BackColor = Color.White;
above_path_lbl.BorderStyle = BorderStyle.Fixed3D;
above_path_lbl.Location = new Point(168, 78);
above_path_lbl.Name = "above_path_lbl";
above_path_lbl.Size = new Size(219, 28);
above_path_lbl.TabIndex = 44;
above_path_lbl.Text = "Alegeți calea fișierului Excel.";
above_path_lbl.Visible = false;
//
// help_path
//
help_path.Interval = 1750;
help_path.Tick += help_path_TimerTick;
//
// pnl
//
pnl.BackColor = Color.CadetBlue;
pnl.Controls.Add(pictureBox1);
pnl.Controls.Add(pcbClose);
pnl.Controls.Add(pcbMin);
pnl.Dock = DockStyle.Top;
pnl.ImeMode = ImeMode.NoControl;
pnl.Location = new Point(0, 0);
pnl.Name = "pnl";
pnl.Size = new Size(550, 55);
pnl.TabIndex = 103;
pnl.MouseDown += Pnl_MouseDown;
pnl.MouseMove += Pnl_MouseMove;
//
// pictureBox1
//
pictureBox1.Image = Properties.Resources.logoUnitbv;
pictureBox1.Location = new Point(0, 1);
pictureBox1.Name = "pictureBox1";
pictureBox1.Padding = new Padding(25, 25, 0, 0);
pictureBox1.Size = new Size(201, 54);
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
pictureBox1.TabIndex = 5;
pictureBox1.TabStop = false;
//
```

```
// pcbClose
//
pcbClose.Image = Properties.Resources.closeing;
pcbClose.Location = new Point(488, 0);
pcbClose.Name = "pcbClose";
pcbClose.Padding = new Padding(0, 2, 2, 0);
pcbClose.Size = new Size(59, 58);
pcbClose.SizeMode = PictureBoxSizeMode.Zoom;
pcbClose.TabIndex = 100;
pcbClose.TabStop = false;
pcbClose.Click += pcbClose_Click;
//
// pcbMin
//
pcbMin.Image = Properties.Resources.minimize;
pcbMin.Location = new Point(423, 6);
pcbMin.Name = "pcbMin";
pcbMin.Padding = new Padding(0, 2, 2, 0);
pcbMin.Size = new Size(59, 49);
pcbMin.SizeMode = PictureBoxSizeMode.Zoom;
pcbMin.TabIndex = 101;
pcbMin.TabStop = false;
pcbMin.Click += pcbMin_Click;
//
// contact_lbl
//
contact_lbl.BackColor = SystemColors.Menu;
contact_lbl.Dock = DockStyle.Bottom;
contact_lbl.Location = new Point(0, 374);
contact_lbl.Name = "contact_lbl";
contact_lbl.Size = new Size(550, 26);
contact_lbl.TabIndex = 102;
contact_lbl.Text = "Portarea resursei Microsoft Excel în baza de date Microsoft SQL";
contact_lbl.TextAlign = ContentAlignment.MiddleCenter;
//
// Form1
//
AutoScaleDimensions = new SizeF(9F, 26F);
AutoScaleMode = AutoScaleMode.Font;
BackColor = Color.White;
ClientSize = new Size(550, 400);
Controls.Add(contact_lbl);
Controls.Add(pnl);
Controls.Add(above_path_lbl);
Controls.Add(del_btn);
```

```

        Controls.Add(deps_cb);
        Controls.Add(insert_btn);
        Controls.Add(above_cb_lbl);
        Controls.Add(path_tb);
        Controls.Add(open_path_btn);
        Controls.Add(load_path_btn);
        Font = new Font("UT Sans", 10.1999989F, FontStyle.Regular, GraphicsUnit.Point, 0);
        FormBorderStyle = FormBorderStyle.None;

        Icon = (Icon)resources.GetObject("$this.Icon");
        MaximumSize = new Size(550, 400);
        MinimumSize = new Size(550, 400);
        Name = "Form1";
        Text = "Portare";
        Load += MainFormLoad;
        pnl.ResumeLayout(false);
        ((System.ComponentModel.ISupportInitialize)pictureBox1).EndInit();
        ((System.ComponentModel.ISupportInitialize)pcbClose).EndInit();
        ((System.ComponentModel.ISupportInitialize)pcbMin).EndInit();
        ResumeLayout(false);
        PerformLayout();
    }

#endregion
private Button load_path_btn;
private Button open_path_btn;
private TextBox path_tb;
private Label above_cb_lbl;
private Button insert_btn;
private ComboBox deps_cb;
private System.Windows.Forms.Timer help_msg;
private Button del_btn;
private Label above_path_lbl;
private System.Windows.Forms.Timer help_path;
private Panel pnl;
private PictureBox pcbClose;
private PictureBox pcbMin;
private PictureBox pictureBox1;
private Label contact_lbl;
}
}

```

Urmează cea mai mare parte de cod care a fost scrisă într-un singur fișier, unde metodele multiple au rolul de a mai degreva din blocul de cod. Multiple linii de cod s-au impus din considerentul că baza de

date trebuie scrisă într-o singură instanță de formular, similar cu un panou de comandă, însă toate funcțiile sunt bine scrise și acoperă multe posibilități de funcționare eronată a aplicației, majoritatea situațiilor impunând un bloc try-catch sau mai multe.

*Windows Form - Secvență de cod 4 Form1.cs*

```
using System.Data.SqlClient;
using System.Diagnostics;
using System.IO;
using System.Linq.Expressions;
using System.Runtime.Intrinsics.X86;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace OrarUnitbv
{
    public partial class Form1 : Form
    {
        bool ok_excel = false;
        string G_PATH = "path";
        //string g_var = "";
        const string EXCEL_PATH = "D:\\unitbv\\practica\\an3\\OrarUnitbv\\orar_iesc_test.xlsx";
        string[] days = { "LUNI", "MARTI", "MIERCURI", "JOI", "VINERI" };
        const int MAX_DEP = 30; // maximum of 30 departments
        const int I = 10, J = 12; // indexes for measuring the worksheet
        const int CHECK_COLS = 30;
        const int CHECK_ROWS = 80;
        const int DEFAULT_SHEET = 1;
        Excel excel;
        bool excel_inst = false;
        Schedule schedule = new Schedule();
        Point mouse_location;
        int[] ones = new int[MAX_DEP];
        int[] nones = new int[MAX_DEP];
        int[] ones_nones = new int[2 * MAX_DEP];
        string[] deps = new string[MAX_DEP];
        int rows = 0;
        int cols = 0;
        //DB connection
        const string con_string = "Data Source=SERBANICA\\SQLEXPRESS03;Initial Catalog=IESC;Persist Security Info=True;User ID=SeanPaul;Password=securitycomesFIRST!!;TrustServerCertificate=True";
        SqlConnection con = new SqlConnection(con_string);
        //
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
// Excel
void ExcellInstantiation(string path, int sheet)
{
    if (path != "" && path != null)
    {
        if (!excel_inst)
        {
            excel = new Excel(path, sheet);
            excel_inst = true;
        }
    }
}
//
void ExcelDeinstantiation()
{
    if (excel_inst)
    {
        excel.Close();
        excel_inst = false;
    }
}
//
void load_path_btn_Click(object sender, EventArgs e)
{
    if (G_PATH != "" && G_PATH != null)
    {
        try
        {
            // counts rows and columns of the excel's sheet
            ExcellInstantiation(G_PATH, 1); // excel instance ready to use

            int i = 1, count = 0;
            bool chk = false;
            int test = 0;
            // check number of lines for the 12th columns
            while (i <= excel.MaxRows())
            {
                if (excel.ReadCell(i, J) == "Null cell!")
                {
                    //Debug.WriteLine("\n" + "first if" + "\n");
                    count++;
                    chk = true;
                    test++;
                    //label16.Text = test.ToString();
                }
            }
        }
    }
}
```

```
        else
        {
            //Debug.WriteLine("\n" + "first else" + "\n");
            count = 0;
            chk = false;
        }
        i++;
        // break if
        if (chk && count > CHECK_ROWS)
        {
            rows = i;
            i = excel.MaxRows() + 1;
        }
    }
    if (rows != 0)
    {
        //label14.Text = rows.ToString(); //rows!
    }
    else
    {
        //label14.Text = "failed to count properly the rows";
    }

    count = 0;
    chk = false;
    test = 0;
    i = 1;
    while (i <= excel.MaxColumns())
    {
        if (excel.ReadCell(i, i) == "Null cell!")
        {
            //Debug.WriteLine("\n" + "second if" + "\n");
            count++;
            chk = true;
            test++;
            // label17.Text = test.ToString();
        }
        else
        {
            //Debug.WriteLine("\n" + "second else" + "\n");
            count = 0;
            chk = false;
        }
        i++;
        if (chk && count > CHECK_COLS)
```

```

    cols = i;
    i = excel.MaxColumns() + 1;
}
}

if (cols != 0)
{
    //label15.Text = cols.ToString();
}
else
{
    // label15.Text = "failed to count properly the columns";
}

//SearchExcelByWord(excel_path, "LUNI", 1, 1, rows, cols);
//SearchExcelByWord(excel_path, "VINERI", 1, 1, rows, cols);
// vector with coordinates of department/specialization, blocked column is 1 for "ANUL"
// a maximum of 30 departments/specializations
//ones = new int[MAX_DEP]; no need to declare as it is because it'll be used in other methods
//and it won't get acknowledged properly due to it not being declared or having an instance
int first_1_i = SearchExcelByWordOnRow(G_PATH, "1", 1, rows, 1);
//int first_1_j = SearchExcelByWordOnRow(excel_path, "1", 1, rows, 1)[1];
//string msgbox = "";
ones[0] = first_1_i;
count = 0;
//Debug.WriteLine("Before trouble while.");
//Debug.WriteLine(first_1_i.ToString() + "\n");
while ((first_1_i <= rows) && (count < 30))//starts with first_1_i = 4
{
    try
    {
        //Debug.WriteLine("Entered in while!");
        if (ones[count] != 0)
        {
            //Debug.WriteLine("Entered in if");
            first_1_i++;
            //Debug.WriteLine(first_1_i.ToString());
            count++;
            ones[count] = SearchExcelByWordOnRow(G_PATH, "1", first_1_i, rows, 1);
            first_1_i = ones[count] + 1;
            //Debug.WriteLine(first_1_i.ToString() + "\n");
            //Debug.Write(ones[count]);
        }
    }
    else // break condition
    {

```

```
//Debug.WriteLine("Entered in else");
Array.Resize(ref ones, count);
first_1_i = rows + 1;
}
}
catch
{
    //Debug.WriteLine(exception);
    first_1_i = rows + 1;
}
}

// this is the only solution to close all the instances, to instantiate then to close it
// somewhere an instance persists and this method allows at the end to have no instance in
// the background
//Debug.WriteLine("After trouble while.");
/*
i = 0;
while (i < ones.Length)
{
    msgbox += ones[i].ToString() + "\n";
    i++;
}
i = 0;
*/
//MessageBox.Show(msgbox + "\n" + ones.Length.ToString());
Debug.WriteLine("Before SecondCheckForDeps(EXCEL_PATH)");
SecondCheckForDeps(G_PATH);
Debug.WriteLine("Before DepsFill()");
DepsFill();
Debug.WriteLine("Before CreateTables(EXCEL_PATH)");
CreateTables(G_PATH);
// data insertion:
// first ones[], then nones[], for each there is a reserved space in the sheet
// deps_cb.Items.Add("Option 1");
Debug.WriteLine("Before CombiBox(EXCEL_PATH, DEFAULT_SHEET)");
if (deps_cb.Items.Count == 0)
{
    CombiBox(G_PATH, DEFAULT_SHEET);
    Debug.WriteLine("CombiBox filled");
}
}
}
catch
{
    MessageBox.Show("A apărut o eroare în deschiderea fișierului Excel, verificați calea aleasă",
        "Cale incorectă/inexistentă",
    )
}
```

```
        MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
    }
}
//
void CombiBox(string path, int sheet)
{
    if (G_PATH != "" && G_PATH != null)
    {
        ExcellInstantiation(G_PATH, DEFAULT_SHEET);
        int i;
        for (i = 0; i < ones.Length; i++)
        {
            deps_cb.Items.Add(excel.ReadCell(ones[i], 2));
        }
        for (i = 0; i < nones.Length; i++)
        {
            deps_cb.Items.Add(excel.ReadCell(nones[i], 2));
        }
        ExcelDeinstantiation();
    }
}
//
void DepsFill()
{
    if (G_PATH != "" && G_PATH != null)
    {
        ExcellInstantiation(G_PATH, DEFAULT_SHEET);
        int c = 0, i;
        string s;
        for (i = 0; i < ones.Length; i++)
        {
            s = excel.ReadCell(ones[i], 2);
            deps[c] = s;
            ones_nones[c] = ones[i];
            c++;
        }
        for (i = 0; i < nones.Length; i++)
        {
            s = excel.ReadCell(nones[i], 2);
            deps[c] = excel.ReadCell(nones[i], 2);
            ones_nones[c] = nones[i];
            c++;
        }
    }
}
```

```
// resize for dep[]
int length = deps.Length;
for (i = 0; i < length; i++)
{
    if (deps[i] == null)
    {
        Array.Resize(ref deps, i);
        i = length;
    }
}
// resize for ones_nones[]
length = ones_nones.Length;
for (i = 0; i < length; i++)
{
    if (ones_nones[i] == 0)
    {
        Array.Resize(ref ones_nones, i);
        i = length;
    }
}

Array.Sort(ones_nones);
Array.Sort(deps);
//
string msg1 = "", msg2 = "";
for (i = 0; i < deps.Length; i++)
{
    msg1 += deps[i] + "\n";
}
//MessageBox.Show(msg1);

for (i = 0; i < ones_nones.Length; i++)
{
    msg2 += ones_nones[i].ToString() + "\n";
}
//MessageBox.Show(msg2);
//
ExcelDeinstantiation();
}
}
//
void OpenFileWithPath(string path, int row, int column)
{
    if (path != "" && path != null)
    {
```

```

Excel excel = new Excel(path, 1);
MessageBox.Show(excel.ReadCell(row, column));
ExcelDeinstantiation();
}
}
//
int[] SearchExcelByWord(string path, string word, int start_i, int start_j, int end_i, int end_j)
{
    // start_i = start index for rows and start_j = start index for columns
    // end_i = end index for rows and end_j = end index for columns
    if (path != "" && path != null)
    {
        ExcellInstantiation(path, DEFAULT_SHEET);
        int i, j;
        int[] a = new int[2];
        bool ok = false;
        for (i = start_i; i <= end_i; i++)
        {
            for (j = start_j; j <= end_j; j++)
            {
                if (excel.ReadCell(i, j) == word)
                {
                    a[0] = i; a[1] = j;
                    ok = true;
                }
            }
            //leaves the loop
            if (ok)
            {
                i = end_i + 1; j = end_j + 1;
            }
        }
        ExcelDeinstantiation();
        return a;
    }
    else
        return null;
}
//
int SearchExcelByWordOnRow(string path, string word, int start_i, int end_i, int static_j)
{
    if (path != "" && path != null)
    {
        // start_i = start index for rows and end_i = end index for rows
        ExcellInstantiation(G_PATH, DEFAULT_SHEET);
    }
}

```

```

int i;
int a = 0;
bool ok = false;

for (i = start_i; i <= end_i; i++)
{
    if (excel.ReadCell(i, static_j) == word)
    {
        a = i;
        ok = true;
    }

    //leaves the loop
    if (ok)
    {
        i = end_i + 1;
    }
}

ExcelDeinstantiation();
// if a = 0, the sheet doesn't have rows indexed with 0, then the
// word doesn't exists in the sheet
return a;
}

else
    return 0;
}

// 
string[] CellSplit(string str)
{
    // e. g.: Graf., L, NII8, Musuroi_C_L
    string[] arr = { str.Split(',')[0].Trim(), str.Split(',')[1].Trim(),
        str.Split(',')[2].Trim(), str.Split(',')[3].Trim() };
    return arr;
}

//
void open_path_btn_Click(object sender, EventArgs e)
{
    try
    {
        string extension = "";
        string newFilePath = "";
        // open file
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.Title = "Alegeți fișierul Excel:";
        openFileDialog.InitialDirectory = "D:\\";
    }
}

```

```
openFileDialog.Filter = "Excel Files (*.xlsx; *.xls)|*.xlsx; *.xls|All files (*.*)|*.*";
openFileDialog.FilterIndex = 1;

if (openFileDialog.ShowDialog() == DialogResult.OK)
{
    ok_excel = true;
    path_tb.Text = openFileDialog.FileName;
    G_PATH = openFileDialog.FileName;
    extension = System.IO.Path.GetExtension(openFileDialog.FileName).ToLower();
    if (extension == ".xls")
    {
        try
        {
            ExcelInstantiation(openFileDialog.FileName, DEFAULT_SHEET);
            // ConvertXlsToXlsx() also closes the instance
            newFilePath = excel.ConvertXlsToXlsx(openFileDialog.FileName);
            ExcelDeinstantiation();
            if (!string.IsNullOrEmpty(newFilePath))
            {
                // update the path text box with the new file path
                path_tb.Text = newFilePath;
                G_PATH = newFilePath;
                ok_excel = true;
                MessageBox.Show("Conversie reușită!", "Succes", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        }
        catch
        {
            path_tb.Text = "Conversia eșuată.";
            G_PATH = "";
            ok_excel = false;
        }
    }
    else
    {
        path_tb.Text = "Nu s-a putut deschide fișierul.";
        G_PATH = "";
        ok_excel = false;
    }
    //MessageBox.Show(newFilePath);
}
catch (Exception ex)
{
```

```

        MessageBox.Show("Eroare de conversie.", "Eroare", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        Debug.WriteLine("\n" + ex.ToString() + "\n");
        ok_excel = false;
    }
}

// 
void insert_btn_Click(object sender, EventArgs e)
{
    if (G_PATH != "" && G_PATH != null && deps_cb.Items.Count != 0)
    {
        //nothing for now !!!!!
        // panel text and forward it
        bool chk = false;
        string dep = deps_cb.Text;
        string sgr, day, time, nsgr, typ;
        string debug = "";
        int c = 0, w;
        // i = 0 and j = 0 for the exception case
        int i = 0, j = 0, semigroup, years;
        for (i = 0; i < deps.Length; i++)
        {
            if (deps[i] == dep)
            {
                chk = true;
                i = deps.Length;
            }
        }
        if (!chk)
        {
            MessageBox.Show("Departamentul ales nu există!\nAlegeți unul dintre " +
                "departamentele înșiruite în listă.");
        }
        else
        {
            try
            {
                ExcellInstantiation(G_PATH, DEFAULT_SHEET);
                int dep_i = 0;
                // insert data with method
                for (i = 0; i < ones_nones.Length - 1; i++) // the last won't be read and used
                {
                    if (excel.ReadCell(ones_nones[i], 2) == dep)
                    {
                        // retain the index
                }
            }
        }
    }
}

```

```
dep_i = i;
// break condition
i = ones_nones.Length;
}
}
if (dep_i != 0)
{
    schedule.setTab(dep);
    debug += dep + "\n";
    // group = semigroup - 1 on column
    semigroup = SearchExcelByWord(G_PATH, "Sgr.", 1, 1, rows, cols)[1];
    ExcellInstantiation(G_PATH, DEFAULT_SHEET);
    years = SearchExcelByWord(G_PATH, "Anul", 1, 1, rows, cols)[1];
    ExcellInstantiation(G_PATH, DEFAULT_SHEET);
    for (i = ones_nones[dep_i]; i <= ones_nones[dep_i + 1] - 1; i++) // dep_i
    {
        c++;
        for (j = 5; j <= 39; j++)
        {
            // another value for dep different from the previous one
            dep = excel.ReadCell(i, j);
            // here we take each relevant cell and insert it into the table
            if (dep != "Null cell!")
            {
                // this is what is in a cell
                schedule.setAbvr(CellSplit(dep)[0]); // course, e.g. PCLP2
                debug += CellSplit(dep)[0] + "\n";

                // if this is C then it must be for every semigroup
                typ = CellSplit(dep)[1];
                schedule.setType(typ); // type, e.g. L or C
                debug += typ + "\n";

                schedule.setPlace(CellSplit(dep)[2]); // e.g. NII8
                debug += CellSplit(dep)[2] + "\n";

                schedule.setProf(CellSplit(dep)[3]); // e.g. Musuroi_C_L
                debug += CellSplit(dep)[3] + "\n";

                time = excel.ReadCell(ones_nones[0] - 1, j);
                schedule.setTime(time); // time(hours)
                debug += time + "\n";

                //FromTimeToDay(time); // day based on time refference
                day = excel.ReadCell((ones_nones[0] - 2), j - FromTimeToDay(time));
            }
        }
    }
}
```

```

        schedule.setDay(day);
        debug += day + "\n";

        // semigroups always on column 4, which is p(var) for us
        sgr = excel.ReadCell(i, semigroup);
        nsgr = excel.ReadCell(i + 1, semigroup);
        //Debug.WriteLine("sgr:" + sgr + "\nnsg: " + nsgr);
        if (sgr == "A")
        {
            sgr = excel.ReadCell(i, semigroup - 1) + "A1";
            schedule.setStds(sgr);
            debug += sgr + "\n";
        } // PROBLEMA CAND SGR nu e A!! deci pe a doua linie
        else if (sgr == "B")
        {
            sgr = excel.ReadCell(i - 2, semigroup - 1) + "B1";
            schedule.setStds(sgr);
        }
        else if (sgr == "Null cell!" && nsgr == "A")
        {
            sgr = excel.ReadCell(i - 3, semigroup - 1) + "B2";
            schedule.setStds(sgr);
        }
        else if (sgr == "Null cell!" && nsgr == "B")
        {
            sgr = excel.ReadCell(i - 1, semigroup - 1) + "A2";
            schedule.setStds(sgr);
        }
        // here we must try to insert data into the table
        InsertDataFromCellIntoTab(
            schedule.getTab(),
            schedule.getAbvr(),
            schedule.getType(),
            schedule.getProf(),
            schedule.getDay(),
            schedule.getTime(),
            schedule.getPlace(),
            schedule.getStds());

        // insert courses for every semigroup
        // the years are always on column 1, but we will use years
        if (typ == "C" && excel.ReadCell(i + 1, j) == "Null cell!")
        {
            w = i + 1;
            // semigroups always on column 4, which is p(var) for us

```

```

        sgr = excel.ReadCell(w, semigroup);
        nsgr = excel.ReadCell(w + 1, semigroup);
        //Debug.WriteLine("sgr:" + sgr + "\nnsg: " + nsgr);
        if (sgr == "A")
        {
            sgr = excel.ReadCell(w, semigroup - 1) + "A1";
            schedule.setStds(sgr);
            debug += sgr + "\n";
        } // PROBLEMA CAND SGR nu e A!! deci pe a doua linie
        else if (sgr == "B")
        {
            sgr = excel.ReadCell(w - 2, semigroup - 1) + "B1";
            schedule.setStds(sgr);
        }
        else if (sgr == "Null cell!" && nsgr == "A")
        {
            sgr = excel.ReadCell(w - 3, semigroup - 1) + "B2";
            schedule.setStds(sgr);
        }
        else if (sgr == "Null cell!" && nsgr == "B")
        {
            sgr = excel.ReadCell(w - 1, semigroup - 1) + "A2";
            schedule.setStds(sgr);
        }
        InsertDataFromCellIntoTab(
            schedule.getTab(),
            schedule.getAbvr(),
            schedule.getType(),
            schedule.getProf(),
            schedule.getDay(),
            schedule.getTime(),
            schedule.getPlace(),
            schedule.getStds());
    }
}
}
}
}
else
{
    // redundant, but might be caused by problems outside the soft
    MessageBox.Show("Departamentul ales nu există!\nAlegeti unul dintre " +
        "departamentele înșiruite în listă.");
}
}

```

```

        catch (SqlException ex)
    {
        MessageBox.Show($"Acțiune de inserare a datelor în tabelul {schedule.getTab()} a " +
            $"eșuat!");
        MessageBox.Show("line: " + i + " col: " + j + "\n" + ex.ToString());
    }
    // no need to reset c1 and/or c2 because this method executes on button click
    //InsertDataFromCellIntoTab();
    ExcelDeinstantiation();
    //Debug.WriteLine(debug); in my_file.txt
    debug = "exception encountered!";
    File.WriteAllText("D:\\unitbv\\practica\\an3\\OrarUnitbv\\my_file.txt", debug);
}
}
}
// 
int FromTimeToDay(string time)
{
    // the columns that must be decreased in order to check the day
    // also the line is always i - 1, so one line back
    switch (time)
    {
        case "8,00-9,50":
        {
            return 0;
        }
        case "10,00-11,50":
        {
            return 1;
        }
        case "12,00-13,50":
        {
            return 2;
        }
        case "14,00-15,50":
        {
            return 3;
        }
        case "16,00-17,50":
        {
            return 4;
        }
        case "18,00-19,50":
        {
            return 5;
        }
    }
}

```

```

        }
        case "20,00-21,50":
        {
            return 6;
        }
    default:
    {
        return -1;
    }
}
//
string CreateTables(string s)
{
    if (G_PATH != null && G_PATH != "")
    {
        // Run in a loop, search for "1", then the next column is the department
        // Array ones[]
        // ANUL always on column 1, Department is on the next column, column 2
        // Tables creation is a one time process
        bool chk1 = false, chk2 = false;
        string ent;

        try
        {
            con.Open();
        }
        catch (Exception ex)
        {
            chk1 = true;
            Debug.WriteLine("\n" + "Exception met in first try-catch (problems in opening a connection to the
server)! Read below:" + "\n" + ex.ToString() + "\n");
        }
        // First, we must check if the entities were already created
        try
        {
            // 1
            ent = deps[0];
            SqlCommand check_cmd_1 = new SqlCommand($"select * from {ent}", con);
            check_cmd_1.CommandTimeout = 150;
            SqlDataReader check_reader_1 = check_cmd_1.ExecuteReader();
            if (!check_reader_1.HasRows)
            {
                chk2 = true;
                Debug.WriteLine("\n" + "Entities already created!" + "\n");
            }
        }
    }
}

```

```

        con.Close();
        return "entities already created";
    }
    check_reader_1.Close();

    // 2
    ent = deps[deps.Length / 2];
    check_cmd_1 = new SqlCommand($"select * from {ent}", con);
    check_cmd_1.CommandTimeout = 150;
    check_reader_1 = check_cmd_1.ExecuteReader();
    if (!check_reader_1.HasRows)
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Entities already created!" + "\n");
        con.Close();
        return "entities already created";
    }
    check_reader_1.Close();

    // 3
    ent = deps[deps.Length - 1];
    check_cmd_1 = new SqlCommand($"select * from {ent}", con);
    check_cmd_1.CommandTimeout = 150;
    check_reader_1 = check_cmd_1.ExecuteReader();
    if (!check_reader_1.HasRows)
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Entities already created!" + "\n");
        con.Close();
        return "entities already created";
    }
    check_reader_1.Close();
}
catch (SqlException ex)
{
    if (ex.Number == 208)
    {
        // in this case we can't find the entities
        // chk2 stays false
    }
    else
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Error in checking entities existence! Read below:" + "\n" + ex.ToString()
+ "\n");
    }
}

```

```
        con.Close();
        return "fail-0 (in checking existence)";
    }
}

if (!chk1 && !chk2)
{
    for (int i = 0; i < deps.Length; i++)
    {
        try
        {
            SqlCommand create_cmd;
            create_cmd = new SqlCommand(
                $"create table {deps[i]}" +
                "(" +
                "ID int identity primary key not null, " +
                "MaterieAbreviere nvarchar(15) not null, " +
                "Materie nvarchar(15), " +
                "Tip nchar(1) check (Tip = 'C' or Tip = 'L' or Tip = 'S' or Tip = 'P') not null, " +
                "CadruDidactic nvarchar(150) not null, " +
                "Ziua nvarchar(15) check (Ziua = 'LUNI' or Ziua = 'MARTI' or Ziua = 'MIERCURI' or Ziua = 'JOI' or
Ziua = 'VINERI') not null, " +
                "IntervalOrar nvarchar(15) check (IntervalOrar = '8,00-9,50' or IntervalOrar = '10,00-11,50' or
IntervalOrar = '12,00-13,50' or IntervalOrar = '14,00-15,50' or IntervalOrar = '16,00-17,50' or IntervalOrar =
'18,00-19,50' or IntervalOrar = '20,00-21,50') not null, " +
                "Locatie nvarchar(50) not null, " +
                "Semigrupa nvarchar(15) not null " +
            ");", con);
            //create_cmd.Parameters.AddWithValue("@Dep", dep);
            create_cmd.CommandTimeout = 70;
            create_cmd.ExecuteNonQuery();

        }
        catch (Exception ex)
        {
            i = ones.Length;
            con.Close();
            Debug.WriteLine("\n" + "Exception met in second try-catch! Read below:" + "\n" +
ex.ToString() + "\n");
            return "fail-1";
        }
    }
}
else
{
```

```

        con.Close();
        return "fail-2";
    }

    con.Close();
    return "success";
}
else
{
    return "Global path is null or empty!";
}
}

// 
string InsertDataFromCellIntoTab(string tab, string abvr, string type, string prof, string day, string time,
string place, string stds)
{
    // con.Open();
    // SqlCommand cmd = new SqlCommand("insert intol", con);
    // con.Close();
    // e.g. Graf., L, NII8, Musuroi_C_L
    // using CellSplit(str)[] method
    con.Open();
    SqlCommand insert_cmd = new SqlCommand("insert into {tab} (MaterieAbreviere, Tip, CadruDidactic, " +
        $"Ziua, IntervalOrar, Locatie, Semigrupa) values " +
        $"(@abvr, @type, @prof, @day, @time, @place, @stds);", con);
    //insert_cmd.Parameters.AddWithValue("@tab", tab);
    insert_cmd.Parameters.AddWithValue("@abvr", abvr);
    insert_cmd.Parameters.AddWithValue("@type", type);
    insert_cmd.Parameters.AddWithValue("@prof", prof);
    insert_cmd.Parameters.AddWithValue("@day", day);
    insert_cmd.Parameters.AddWithValue("@time", time);
    insert_cmd.Parameters.AddWithValue("@place", place);
    insert_cmd.Parameters.AddWithValue("@stds", stds);
    insert_cmd.CommandTimeout = 45;
    insert_cmd.ExecuteNonQuery();
    con.Close();
    return "ceva";
}
//
void SecondCheckForDeps(string path)
{
    if (path != null && path != "")
    {
        // Departments are always on the second column, only the line changes
    }
}

```

```
ExcelInstantiation(path, DEFAULT_SHEET);
string main1, main2, cell, old1, old2 = "";
int count = 0;
for (int j = 0; j < ones.Length - 1; j++)
{
    for (int i = ones[j] + 1; i < ones[j + 1]; i++)
    {
        cell = excel.ReadCell(i, 2);
        main1 = excel.ReadCell(ones[j], 2);
        main2 = excel.ReadCell(ones[j + 1], 2);
        if (cell != "Null cell!" && cell != main1 && cell != main2)
        {
            old1 = cell;
            if (old1 != old2)
            {
                nones[count++] = i;
            }
            old2 = old1;
        }
    }
}
for (int i = 0; i < nones.Length; i++)
{
    if (nones[i] == 0)
    {
        Array.Resize(ref nones, i);
        i = nones.Length;
    }
}
/*
string msg = "";
for (int i = 0; i < nones.Length; i++)
{
    msg += nones[i].ToString() + "\n";
}
MessageBox.Show(msg);
*/
ExcelDeinstantiation();
// ExcelInstantiation(excel_path, DEFAULT_SHEET);
// excel.Close();
}
}
// deps_cb
void help_msg_TimerTick(object sender, EventArgs e)
{
```

```
above_cb_lbl.Visible = true;
help_msg.Stop();
}

// 
void deps_cb_MouseEnter(object sender, EventArgs e)
{
    help_msg.Start();
}
// 
void deps_cb_MouseLeave(object sender, EventArgs e)
{
    above_cb_lbl.Visible = false;
    help_msg.Stop();
}
// path_tb
void help_path_TimerTick(object sender, EventArgs e)
{
    above_path_lbl.Visible = true;
    help_path.Stop();
}
// 
void path_tb_MouseEnter(object sender, EventArgs e)
{
    help_path.Start();
}
// 
void path_tb_MouseLeave(object sender, EventArgs e)
{
    above_path_lbl.Visible = false;
    help_path.Stop();
}
// 
void del_btn_Click(object sender, EventArgs e)
{
    if (deps_cb.Items.Count != 0)
    {
        string tab = deps_cb.Text;
        bool chk = false;
        try
        {
            con.Open();
            SqlCommand drop_cmd = new SqlCommand($"truncate table {tab}", con);
            drop_cmd.ExecuteNonQuery();
            chk = true;
        }
    }
}
```

```
        catch (SqlException sql_ex)
        {
            Debug.WriteLine("\n" + "Exception met in first try-catch! " +
                "Read below:" + "\n" + sql_ex.ToString() + "\n");
        }
        if (chk)
        {
            con.Close();
        }
        MessageBox.Show($"Datele tabelului {tab} au fost șterse.");
    }
}
//  

void MainFormLoad(object sender, EventArgs e)
{
    this.Paint += new PaintEventHandler(this.CustomForm_Paint);
    this.Resize += (s, ev) => this.Invalidate();
}
//  

void CustomForm_Paint(object sender, PaintEventArgs e)
{
    Pen pen = new Pen(Color.Gray, 2);
    e.Graphics.DrawRectangle(pen, 0, 0, this.ClientSize.Width - 1, this.ClientSize.Height - 1);
}
//  

void pcbMin_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}
//  

void pcbClose_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Doriți să ieșiți din aplicație?", "Confirmăți acțiunea",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (result == DialogResult.Yes)
    {
        Application.Exit();
    }
}
//  

void Pnl_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
```

```
Point mouse_pose = Control.mousePosition;
mouse_pose.Offset(mouse_location.X, mouse_location.Y);
Location = mouse_pose;
}
}
//
void Pnl_MouseDown(object sender, MouseEventArgs e)
{
    mouse_location = new Point(-e.X, -e.Y);
}
}
}
```

Mai multe detalii se pot găsi și în rezumatele întocmite până în momentul redactării documentației de față. Adițional o să se insereze la final acestui subcapitol și o imagine cu aplicația rulând.

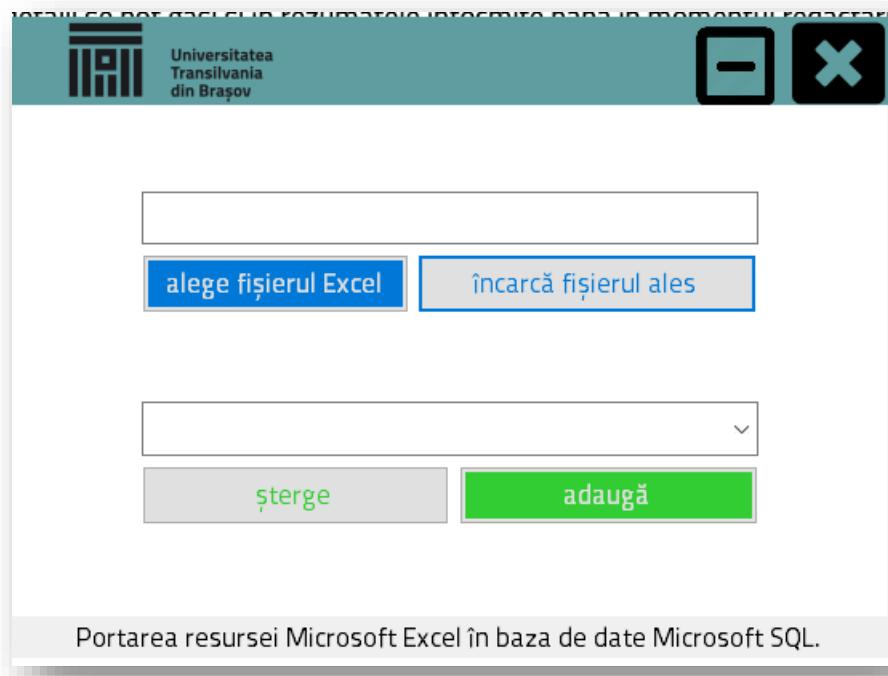


Fig 1.5.1

## 1.6. XAMPP Service

XAMPP este o aplicație care permite gestionarea paginilor non-statice de tip .php care rulează doar în browser nu ar funcționa. Aceasta permite folosirea de metodelor de comunicație bazate pe HTTP, HyperText Transfer Protocol, GET, POST, PUT, etc. În cazul aplicației de față s-a utilizat doar GET și POST, posibilitatea de ștergere fiind o funcționalitate ulterior dezvoltabilă.

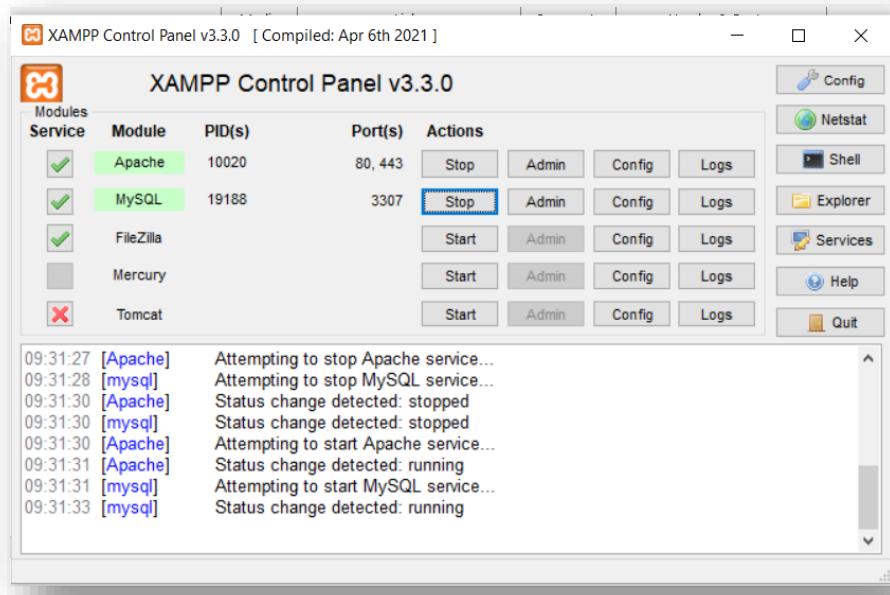


Fig 1.6  
Panoul de comandă XAMPP

## 1.7. Dezvoltare WEB Frontend

Partea finală și cea mai anevoie de programat este partea de interfață, denumită și API, care permite utilizatorului să altereze direct baza de date și, în general, să se deservească de aceasta pentru a citi rezultatele interogărilor scrise în formularul pus la dispoziție.

*Web App - Secvență de cod 1 index.html*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Meniu</title>
    <link rel="icon" href="/favicon.ico" type="image/x-icon">
    <link rel="stylesheet" href="../css/global.css">
    <link rel="stylesheet" type="text/css" href="../css/index.css">
</head>

<body>
    <div>
        <label>Alegeti facultatea:</label>
        <form action="../php/login.php" method="post">
            <button type="submit" name="ImgBtn" value="DPM">
                
    </button>
    <button type="submit" name="ImgBtn" value="IESC">
        
    </button>
    <button type="submit" name="ImgBtn" value="DMIL">
        
    </button>
    <button type="submit" name="ImgBtn" value="Mecanică">
        
    </button>
    <button type="submit" name="ImgBtn" value="ITMI">
        
    </button>
    <button type="submit" name="ImgBtn" value="Silvic">
        
    </button>
    <button type="submit" name="ImgBtn" value="SIM">
        
    </button>
    <button type="submit" name="ImgBtn" value="Drept">
        
    </button>
    <button type="submit" name="ImgBtn" value="Sport">
        
    </button>
    <button type="submit" name="ImgBtn" value="Litere">
        
    </button>
    <button type="submit" name="ImgBtn" value="MateInfo">
        
    </button>
    <button type="submit" name="ImgBtn" value="Muzică">
        

```

```
</button>
<button type="submit" name="ImgBtn" value="Medicină">
    
</button>
<button type="submit" name="ImgBtn" value="PsihoEdu">
    
</button>
<button type="submit" name="ImgBtn" value="Socio">
    
</button>
<button type="submit" name="ImgBtn" value="ECON">
    
</button>
<button type="submit" name="ImgBtn" value="AT">
    
</button>
<button type="submit" name="ImgBtn" value="Construcții">
    
</button>
</form>
</div>
<div>
    <label></label>
</div>
</div>
</body>

</html>
```

*Web App - Secvență de cod 2 global.css*

```
@font-face {
    font-family: 'UT Sans';
    src: url('../fonts/UTSans-Regular.ttf') format('truetype');
    font-weight: normal;
    font-style: normal;
}

@font-face {
```

```
font-family: 'UT Sans';
src: url('../fonts/UTSans-Bold.ttf') format('truetype');
font-weight: bold;
font-style: normal;
}

@font-face {
    font-family: 'UT Sans';
    src: url('../fonts/UTSans-Light.ttf') format('truetype');
    font-weight: 300;
    font-style: normal;
}

@font-face {
    font-family: 'UT Sans';
    src: url('../fonts/UTSans-Medium.ttf') format('truetype');
    font-weight: 500;
    font-style: normal;
}

/*apply for every part of the display*/
* {
    font-family: 'UT Sans', Arial, sans-serif;
    box-sizing: border-box;
}

body {
    background-color: cadetblue;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}
```

*Web App - Secvență de cod 3 index.css*

```
label {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    background-color: whitesmoke;
    width: 100%;
    margin-bottom: 7px;
}
```

```
form {  
    border: 3px solid whitesmoke;  
    display: flex;  
    flex-wrap: wrap;  
    justify-content: center;  
    gap: 7px;  
    max-width: 1200px;  
}  
  
button {  
    background-color: transparent;  
    border: 0px;  
    border-style: solid;  
    padding: 0;  
    cursor: pointer;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
    transition: transform 0.2s ease-in-out;  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
button:hover {  
    transform: scale(1.05);  
}  
  
img {  
    width: 250px;  
    height: auto;  
    display: block;  
    transition: filter 0.3s ease-in-out, transform 0.3s ease-in-out;  
    background-color: whitesmoke;  
    padding: 5px;  
}  
  
@media (max-width: 600px) {  
    img {  
        width: 100%;  
    }  
  
    form {  
        gap: 7px;  
    }  
}
```

Web App- Secvență de cod 4 login.php

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Autentificare</title>
    <link rel="icon" href="/favicon.ico" type="image/x-icon">
    <link rel="stylesheet" href="../css/global.css">
    <link rel="stylesheet" type="text/css" href="../css/login.css">
    <script>
        function goBack() {
            window.history.back();
        }
    </script>
</head>

<body>
    <div>
        <form id="login_form" action="connection.php" method="post">
            <h2>Datele de autentificare</h2>
            <label>Server: </label>
            <input type="text" name="Server" placeholder="numele serverului"><br>
            <label>Facultate: </label>
            <?php
                if ($_SERVER['REQUEST_METHOD'] == 'POST') {
                    echo $_POST['ImgBtn'];
                } ?>
            <br>
            <label>Nume de utilizator: </label>
            <input type="text" name="Username" placeholder="numele adminului"><br>
            <label>Parolă: </label>
            <input type="password" name="Password" placeholder="parola adminului"><br>
            <input type="submit" value="Trimite">
        </form>
        <div id="btn-div">
            <button onclick='goBack()'>Înapoi</button>
        </div>
    </div>
</body>

</html>
```

```
<?php
session_start();
$_SESSION['ImgBtn'] = $_POST['ImgBtn'];
?>
```

*Web App- Secvență de cod 5 login.css*

```
h2 {
    text-align: center;
    margin-bottom: 50px;
}

form {
    width: 500px;
    border-width: 2px solid #ccc;
    padding: 30px;
    background: #fff;
    border-radius: 15px;
}

input {
    display: block;
    border: 2px solid #ccc;
    width: 50%;
    padding: 10px;
    margin: 10px auto;
    border-radius: 5px;
}

label {
    color: #888;
    font-size: 18px;
    padding: 10px;
}

button {
    display: block;
    border: 2px solid #ccc;
    width: 45%;
    padding: 10px;
    margin: 1px auto;
    border-radius: 5px;
}
```

```
}
```

*Web App- Secvență de cod 6 connection.php*

```
<?php
session_start();
header('Content-Type: text/html; charset=utf-8');
//phpinfo();
//$srv = "SERBANICA\SQLEXPRESS03";
$check = 'no';
$con_opt = array(
    'Database' => $_SESSION['ImgBtn'],
    'Uid' => $_POST['Username'],
    'PWD' => $_POST['Password']
);
// establishes the connection
$con = sqlsrv_connect($_POST['Server'], $con_opt);

if ($con === false) {
    //die(print_r(sqlsrv_errors(), true));
    $check = 'no';
} else {
    //echo "Connection established.<br>";
    $check = 'yes';
    $_SESSION['Server'] = $_POST['Server'];
    //$_SESSION['ImgBtn'] = $_POST['ImgBtn'];
    $_SESSION['Username'] = $_POST['Username'];
    $_SESSION['Password'] = $_POST['Password'];
}

if ($check != 'yes') {
    echo "
    <!DOCTYPE html>
    <html lang='en'>
    <head>
        <meta charset='UTF-8'>
        <meta name='viewport' content='width=device-width, initial-scale=1.0'>
    </head>
    <body>
        <h1>Welcome to the Agenda App!</h1>
        <p>This is a simple agenda application built with Spring Boot. You can add, edit, and delete events here.</p>
        <table border='1'>
            <thead>
                <tr>
                    <th>Event ID</th>
                    <th>Event Name</th>
                    <th>Event Description</th>
                    <th>Event Date</th>
                    <th>Edit / Delete</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>Meeting with John</td>
                    <td>Discuss project progress</td>
                    <td>2023-10-15</td>
                    <td><a href='edit/1'>Edit</a> <a href='delete/1'>Delete</a></td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>Gym Session</td>
                    <td>Workout at the gym</td>
                    <td>2023-10-17</td>
                    <td><a href='edit/2'>Edit</a> <a href='delete/2'>Delete</a></td>
                </tr>
            </tbody>
        </table>
        <button>Add New Event</button>
    </body>
</html>
"
}
```

```
<title>Conectare eșuată</title>
<link rel='icon' href='/favicon.ico' type='image/x-icon'>
<link rel='stylesheet' href='..css/global.css'>
<link rel='stylesheet' type='text/css' href='..css/connect.css'>
<script>
function goBack() {
window.history.back();
}
</script>
</head>
<body>
<div>
<h2></h2>
<p>Nu s-a putut stabili o conexiune cu serverul!</p>
<p>Vă rugăm să reintroduceți datele.</p>
<button onclick='goBack()'>Înapoi</button>
</div>
</body>
</html>
";
exit();
} else {
echo
"
<!DOCTYPE html>
<html lang='en'>
<head>
<meta charset='UTF-8'>
<meta name='viewport' content='width=device-width, initial-scale=1.0'>
<title>Conectare reușită</title>
<link rel='icon' href='/favicon.ico' type='image/x-icon'>
<link rel='stylesheet' href='..css/global.css'>
<link rel='stylesheet' type='text/css' href='..css/connect.css'>
</head>
<body>
<form action='command.php'>
<label>Conexiune stabilită cu succes!</label><br>
<label>Apăsați înainte pentru a ajunge la centrul de comandă</label><br>
<input type='submit' value='Înapate'>
</form>
</body>
</html>
";
//header("");
sqlsrv_close($con);
```

```
    exit();  
}
```

*Web App- Secvență de cod 7 connect.css*

```
div {  
    width: 500px;  
    border: 2px solid #ccc;  
    padding: 30px;  
    background: #fff;  
    border-radius: 15px;  
}  
  
form {  
    width: 500px;  
    border-width: 2px solid #ccc;  
    padding: 30px;  
    background: #fff;  
    border-radius: 15px;  
}  
  
button {  
    padding: 10px 20px;  
    background-color: #007bff;  
    color: white;  
    border: none;  
    cursor: pointer;  
}  
  
button:hover {  
    background-color: #0056b3;  
}
```

```
<!DOCTYPE html>
<html lang='en'>

    <head>
        <meta charset='UTF-8'>
        <meta name='viewport' content='width=device-width, initial-scale=1.0'>
        <title>Caută</title>
        <link rel="icon" href="/favicon.ico" type="image/x-icon">
        <link rel="stylesheet" href="../css/global.css">
        <link rel="stylesheet" type="text/css" href="../css/cmd.css">
        <script>
            function goBack(url) {
                window.location.href = url;
            }
        </script>
    </head>

    <body>
        <div class="form-container">
            <form action="command.php" method="get">
                <h2>Informațiile vor fi redate în funcție de câmpurile completate</h2>
                <div class="form-group">
                    <label>Departament:</label>
                    <input type="text" name="Dep" placeholder="TI, CALC, etc."><br>
                </div>
                <div class="form-group">
                    <label>Materie(abreviere):</label>
                    <input type="text" name="Abv" placeholder="MS, GAC, etc."><br>
                </div>
                <div class="form-group">
                    <label>Materie(nume complet):</label>
                    <input type="text" name="NAbv" placeholder="funcție în lucru" disabled id="mat"><br>
                </div>
                <div class="form-group">
                    <label>Tip:</label>
                    <input type="text" name="Type" placeholder="C(curs), S(seminar), etc."><br>
                </div>
                <div class="form-group">
                    <label>Cadru didactic:</label>
                    <input type="text" name="Prof" placeholder="Moldoveanu_F_D, Beldianu_I, etc."><br>
                </div>
                <div class="form-group">
                    <label>Ziua:</label>
```

```

<input type="text" name="Day" placeholder="LUNI, VINERI, etc."><br>
</div>
<div class="form-group">
    <label>Interval orar: </label>
    <input type="text" name="Time" placeholder="8,00-9,50, 14,00-15,50, 20,00-21,50, etc."><br>
</div>
<div class="form-group">
    <label>Locație: </label>
    <input type="text" name="Loc" placeholder="VIV7, PIII1, AULA_NT, etc."><br>
</div>
<div class="form-group">
    <label>Semigrupă: </label>
    <input type="text" name="Sgr" placeholder="4LF431A1, 4LF421A1, 4LF422B2, etc."><br>
</div>
<div class="form-group">
    <input type="submit" value="Caută">
</div>
</form>
<button onclick="goBack('http://localhost/html/')">Înapoi</button>
</div>
<div id="resultModal" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <div id="modal-body">
            <?php
            if (
                $_SERVER['REQUEST_METHOD'] == 'GET' && !isset($_GET['Abv']) && !isset($_GET['Type'])
                && !empty($_GET['Abv']) && !empty($_GET['Type'])
            ) {
                session_start();
                $con_opt = array(
                    'Database' => $_SESSION['ImgBtn'],
                    'Uid' => $_SESSION['Username'],
                    'PWD' => $_SESSION['Password']
                );
                $con = sqlsrv_connect($_SESSION['Server'], $con_opt);
                if ($con === false) {
                    die("Connection failed: " . print_r(sqlsrv_errors(), true));
                }
                $dep = $_GET['Dep'];
                $abv = $_GET['Abv'];
                $type = $_GET['Type'];
            }
        </div>
    </div>
</div>

```

```
$sql = "select CadruDidactic, Ziua, IntervalOrar, Locatie, Semigrupa
        from {$dep}
        where MaterieAbreviere = ? and Tip = ?;";

$params = array($abv, $type);
$query = sqlsrv_query($con, $sql, $params);

if ($query == false) {
    die(print_r(sqlsrv_errors(), true));
} else {
    if (sqlsrv_has_rows($query)) {
        while ($row = sqlsrv_fetch_array($query, SQLSRV_FETCH_ASSOC)) {
            echo "<p>Cadru didactic: " . htmlspecialchars($row['CadruDidactic']) . "<br>".
                  "Ziua: " . htmlspecialchars($row['Ziua']) . "<br>" .
                  "Interval orar: " . htmlspecialchars($row['IntervalOrar']) . "<br>" .
                  "Locație: " . htmlspecialchars($row['Locatie']) . "<br>" .
                  "Semigrupă: " . htmlspecialchars($row['Semigrupa']) . "<br></p>";
        }
    } else {
        echo "<p>Nu există rezultat pentru căutarea inițiată.</p>";
    }
}

sqlsrv_free_stmt($query);
sqlsrv_close($con);
} elseif (
    $_SERVER['REQUEST_METHOD'] == 'GET' && !isset($_GET['Dep']) && !isset($_GET['Type']) &&
    !isset($_GET['Sgr'])
    && !empty($_GET['Dep']) && !empty($_GET['Type']) && !empty($_GET['Sgr'])
) {
    session_start();
    $con_opt = array(
        'Database' => $_SESSION['ImgBtn'],
        'Uid' => $_SESSION['Username'],
        'PWD' => $_SESSION['Password']
    );
}

$con = sqlsrv_connect($_SESSION['Server'], $con_opt);
if ($con === false) {
    die("Connection failed: " . print_r(sqlsrv_errors(), true));
}

$dep = $_GET['Dep'];
$type = $_GET['Type'];
$sgr = $_GET['Sgr'];
```

```
$sql = "select CadruDidactic, Ziua, IntervalOrar, Locatie
from {$dep}
where Tip = ? and Semigrupa = ?;";

$params = array($type, $sgr);
$query = sqlsrv_query($con, $sql, $params);

if ($query == false) {
    die(print_r(sqlsrv_errors(), true));
} else {
    if (sqlsrv_has_rows($query)) {
        while ($row = sqlsrv_fetch_array($query, SQLSRV_FETCH_ASSOC)) {
            echo "<p>Cadru didactic: " . htmlspecialchars($row['CadruDidactic']) . "<br> .
"Ziua: " . htmlspecialchars($row['Ziua']) . "<br> .
"Interval orar: " . htmlspecialchars($row['IntervalOrar']) . "<br> .
"Locație: " . htmlspecialchars($row['Locatie']) . "<br></p>";
        }
    } else {
        echo "<p>Nu există rezultat pentru căutarea inițiată.</p>";
    }
}

sqlsrv_free_stmt($query);
sqlsrv_close($con);
} elseif (
    $_SERVER['REQUEST_METHOD'] == 'GET' && !isset($_GET['Dep']) && !isset($_GET['Sgr'])
    && !empty($_GET['Dep']) && !empty($_GET['Sgr'])
) {
    session_start();
    $con_opt = array(
        'Database' => $_SESSION['ImgBtn'],
        'Uid' => $_SESSION['Username'],
        'PWD' => $_SESSION['Password']
    );
}

$con = sqlsrv_connect($_SESSION['Server'], $con_opt);
if ($con === false) {
    die("Connection failed: " . print_r(sqlsrv_errors(), true));
}

$dep = $_GET['Dep'];
$sgr = $_GET['Sgr'];

$sql = "select MaterieAbreviere, CadruDidactic, Tip, Ziua, IntervalOrar, Locatie
```

```
from {$dep}
where Semigrupa = ?;";

$params = array($sgr);
$query = sqlsrv_query($con, $sql, $params);

if ($query == false) {
    die(print_r(sqlsrv_errors(), true));
} else {
    if (sqlsrv_has_rows($query)) {
        while ($row = sqlsrv_fetch_array($query, SQLSRV_FETCH_ASSOC)) {
            echo "<p>Materie: " . htmlspecialchars($row['MaterieAbreviere']) . "<br> .
                "Cadru didactic: " . htmlspecialchars($row['CadruDidactic']) . "<br> .
                "Tip: " . htmlspecialchars($row['Tip']) . "<br> .
                "Ziua: " . htmlspecialchars($row['Ziua']) . "<br> .
                "Interval orar: " . htmlspecialchars($row['IntervalOrar']) . "<br> .
                "Locație: " . htmlspecialchars($row['Locatie']) . "<br></p>";
        }
    } else {
        echo "<p>Nu există rezultat pentru căutarea inițiată.</p>";
    }
}

sqlsrv_free_stmt($query);
sqlsrv_close($con);
}

?>
</div>
</div>
</div>

<script>
var modal = document.getElementById("resultModal");

var span = document.getElementsByClassName("close")[0];

if (modal.querySelector('#modal-body').innerHTML.trim() !== "") {
    modal.style.display = "block";
}

span.onclick = function() {
    modal.style.display = "none";
}

window.onclick = function(event) {
```

```
if (event.target == modal) {  
    modal.style.display = "none";  
}  
}  
</script>  
  
</body>  
  
</html>
```

*Web App- Secvență de cod 9 cmd.css*

```
/* Center the header */  
h2 {  
    text-align: center;  
    margin-bottom: 30px;  
    /* Reduced margin for a cleaner look */  
}  
  
/* Style the form container */  
form {  
    width: 500px;  
    border: 2px solid #ccc;  
    padding: 20px;  
    /* Reduced padding for a more compact form */  
    background: #fff;  
    border-radius: 10px;  
    /* Slightly rounded corners */  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
    /* Subtle shadow for depth */  
    margin: 0 auto;  
    /* Center the form horizontally */  
}  
  
/* Flex container for form groups */  
.form-group {  
    display: flex;  
    align-items: center;  
    /* Align items vertically centered */  
    margin-bottom: 15px;  
    /* Space between form groups */  
}
```

```
/* Style the labels */
label {
    color: #555;
    /* Darker color for better readability */
    font-size: 14px;
    flex: 0 0 200px;
    /* Fixed width for labels */
    margin-right: 15px;
    /* Space between label and input */
    text-align: right;
    /* Align text to the right for better alignment with input */
}

/* Style the input fields */
input[type="text"] {
    flex: 1;
    /* Grow to fill remaining space */
    border: 2px solid #ccc;
    padding: 10px;
    border-radius: 5px;
    font-size: 14px;
    /* Consistent font size */
}

/* Focus state for input fields */
input[type="text"]:focus {
    border-color: #007bff;
    /* Blue border on focus */
    outline: none;
}

/* Style the form container */
.form-container {
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    width: 100%;
    max-width: 600px;
    /* Maximum width for larger screens */
    margin: 0 auto;
    /* Center the container horizontally */
}

input[type="submit"] {
```

```
display: block;
width: 50%;
border-radius: 5px;
flex: 1;
/* Grow to fill remaining space */
border: 2px solid #ccc;
padding: 10px;
border-radius: 5px;
font-size: 14px;
/* Consistent font size */
margin: 0 auto;
/* Center the submit button */
}

#mat {
background-color: #c3c3c3;
}

/* Modal Styles */
.modal {
display: none;
/* Hidden by default */
position: fixed;
/* Stay in place */
z-index: 1;
/* Sit on top */
left: 0;
top: 0;
width: 100%;
/* Full width */
height: 100%;
/* Full height */
overflow: auto;
/* Enable scroll if needed */
background-color: rgba(0, 0, 0, 0.4);
/* Black w/ opacity */
}

.modal-content {
background-color: #fff;
margin: 15% auto;
/* 15% from the top and centered */
padding: 20px;
border: 1px solid #888;
width: 80%;
```

```
/* Could be more or less, depending on screen size */
border-radius: 10px;
}

.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}

button {
    display: block;
    border: 2px solid #ccc;
    width: 45%;
    padding: 10px;
    margin: 1px auto;
    border-radius: 5px;
}
```

De remarcat aici sunt variabilele super-globale `$_POST['']`, `$_GET['']` și `$_SESSION['']` care sunt vitale pentru funcționarea corectă a interfeței. Cu ajutorul acestora se pot transmite date între paginile aplicației web, `$_POST['']` este bună pentru lucrul cu date sensibile deoarece modalitatea de transmitere a acestora este învăluită spre deosebire de `$_GET['']` care transmite prin intermediul URL-ului, iar, spre exemplu, în cazul transmiterii de parole acest lucru este de mare risc, datele fiind la vedere pentru orice soft de tip malware. `$_SESSION['']` permite reținerea variabilelor pe baza unei sesiunii, în proiect s-a folosit pentru a putea reține o variabilă transmisă prin metoda POST, care trebuia ulterior refolosită, însă trebuia memorată într-o variabilă de sesiune. Variabilele în discuție sunt cele necesare stabilirii unei conexiuni cu baza de date deoarece conexiunea nu poate rămâne (nu este recomandat) deschisă pe durata transferului între mai multe pagini, aşadar fiecare interogare închide și deschide conexiunea pentru maximum de siguranță, diminuând astfel performanța pentru securitate. În continuare vor fi atașate imagini cu toate paginile aplicației web dezvoltate.

# Departamentul Automatică și Tehnologia Informației

## Proiect Agendă Spring Boot

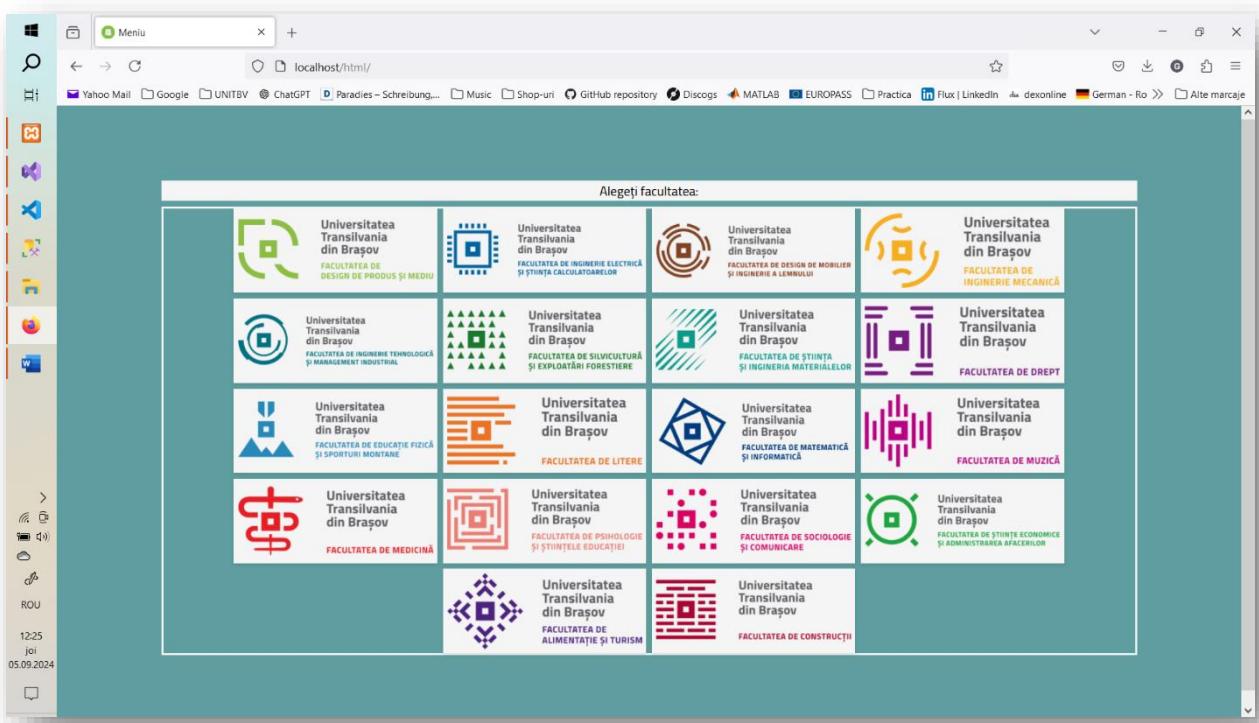


Fig 1.7.1  
Meniu

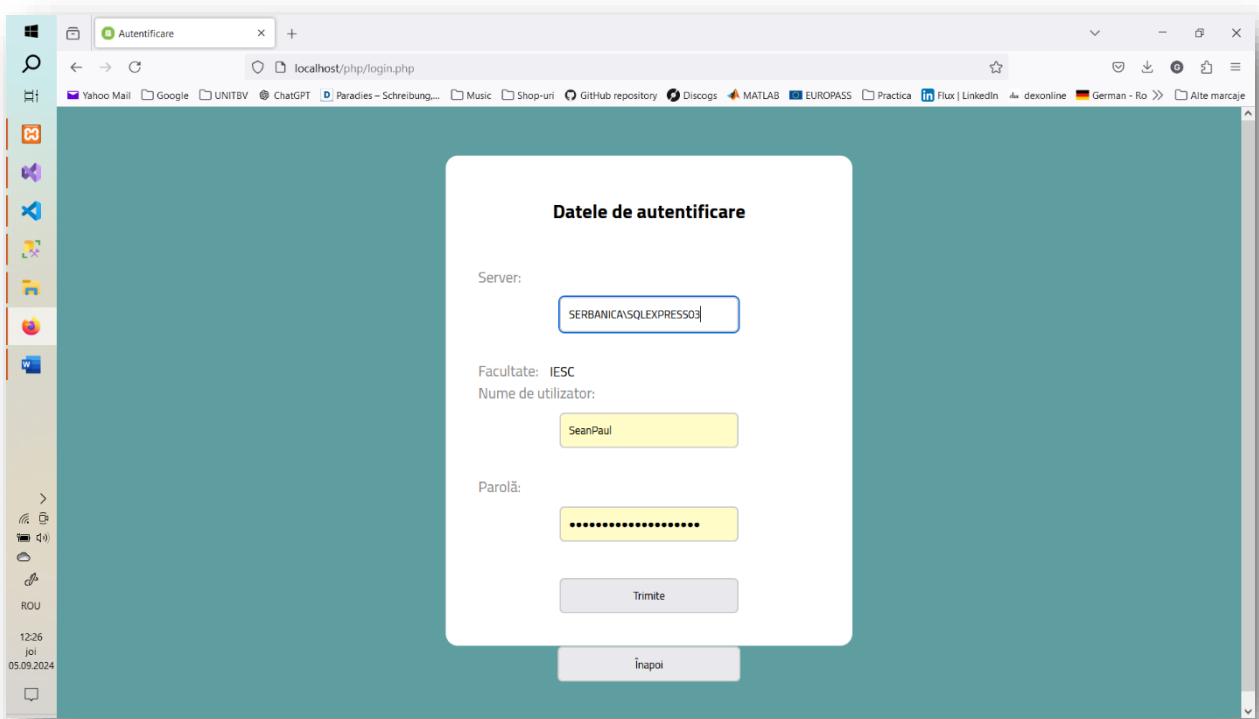


Fig 1.7.2  
Formular de autentificare

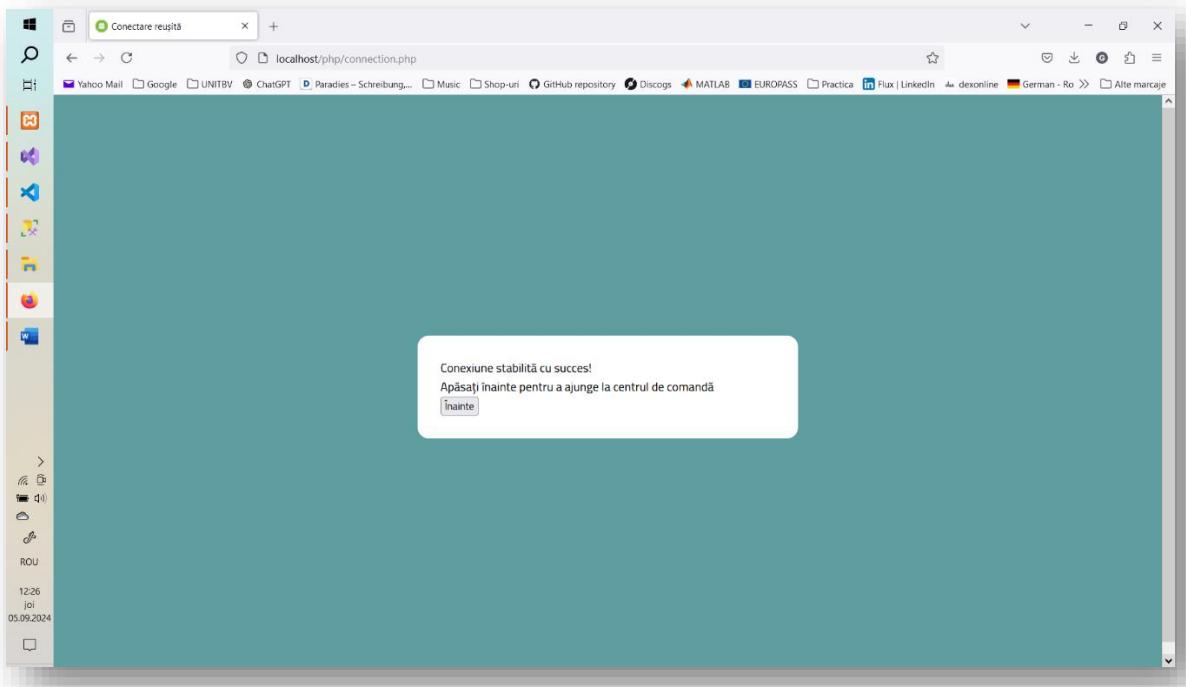


Fig 1.7.3  
Confirmarea autentificării

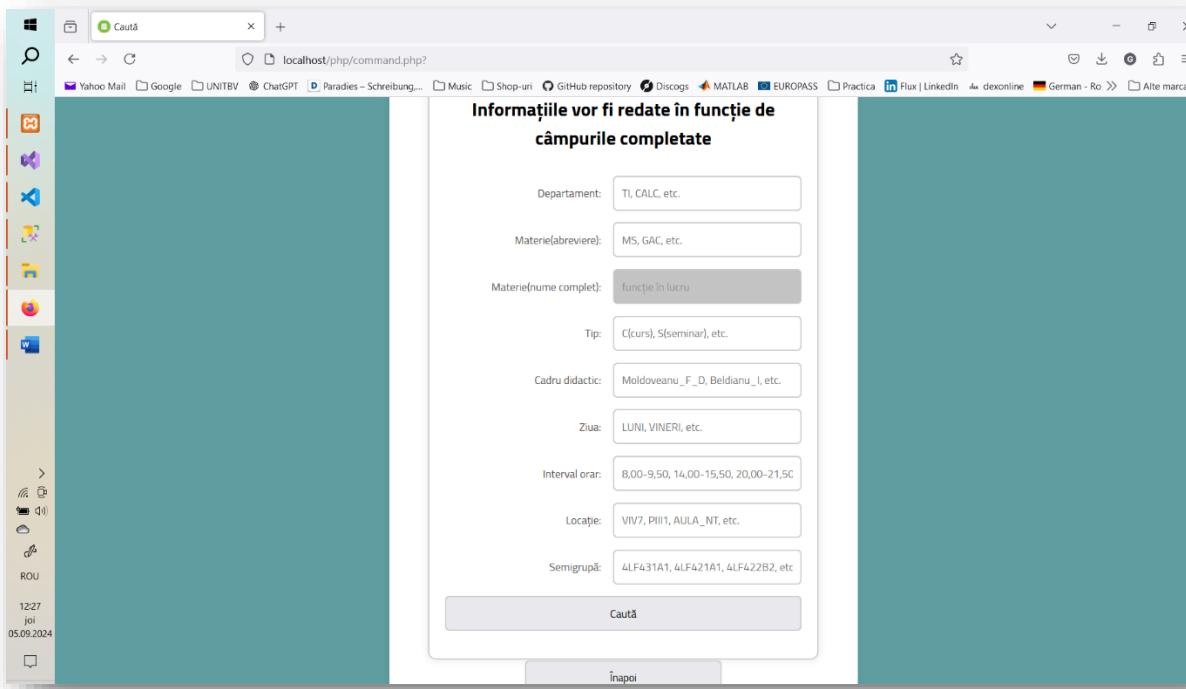


Fig 1.7.4  
Panoul de comandă

## 1.8. Manual de utilizare

Manual de utilizare va prezenta, după cum am realizat în filmarea demonstrativă, același aspect, crearea unei baze de date prin intermediul unei aplicații de tipul Windows Form, apoi interogarea ei prin intermediul aplicației web cu ajutorul XAMPP, videoclipul surprinde mai bine modul de utilizare, însă imaginile de mai jos vor surprinde aceeași metodă doar că într-un mod mai succint fără conexiuni evidente. Evenimentele decurg la fel, și anume

- Selectarea unui fișier Excel de format vechi (.xls) (fig 1.8.1 și fig 1.8.2),
- Convertirea lui în format mai nou de Excel, și anume .xlsx (fig 1.8.3),
- Adăugarea unui departament în baza de date selectată, AIA este adaugat în IESC (fig 1.8.4),
- Confirmare în SSMS a adăugării în baza de date (fig 1.8.5),
- Meniul paginii web, de unde se poate selecta facultatea pe baza de butoane (fig 1.8.6),
- Formularul de autentificare (fig 1.8.7),
- Confirmarea/infirmarea autentificării (fig 1.8.8),
- Panoul de comenzi, unde se interoghează baza de date (fig 1.8.9 și fig 1.8.10),
- Confirmarea în SSMS a interogării alese (fig 1.8.11)



Fig 1.8.1

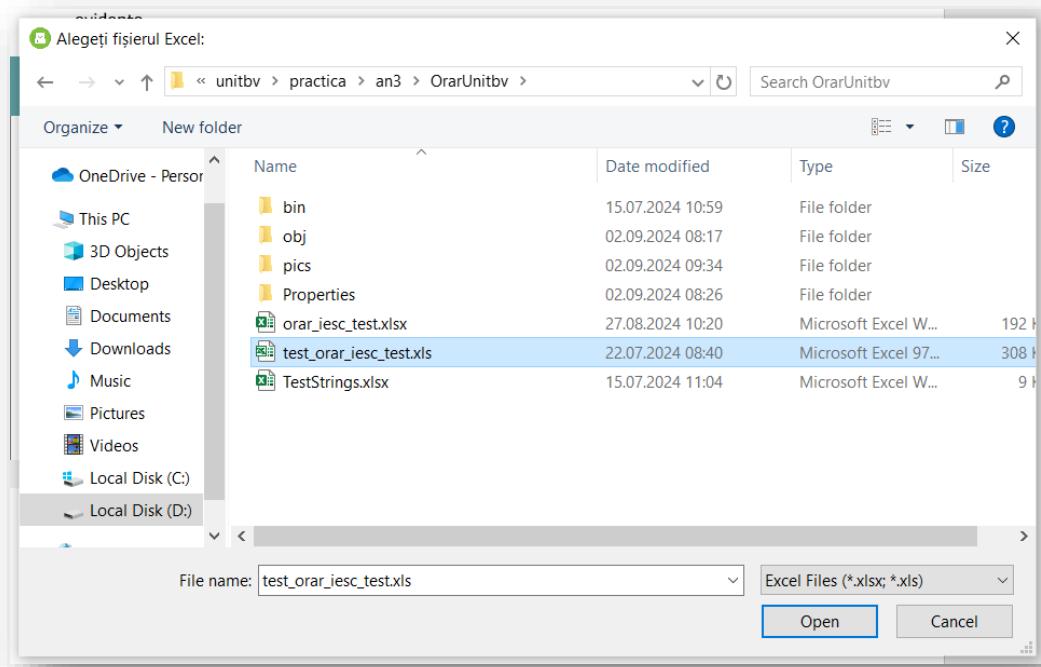


Fig 1.8.2

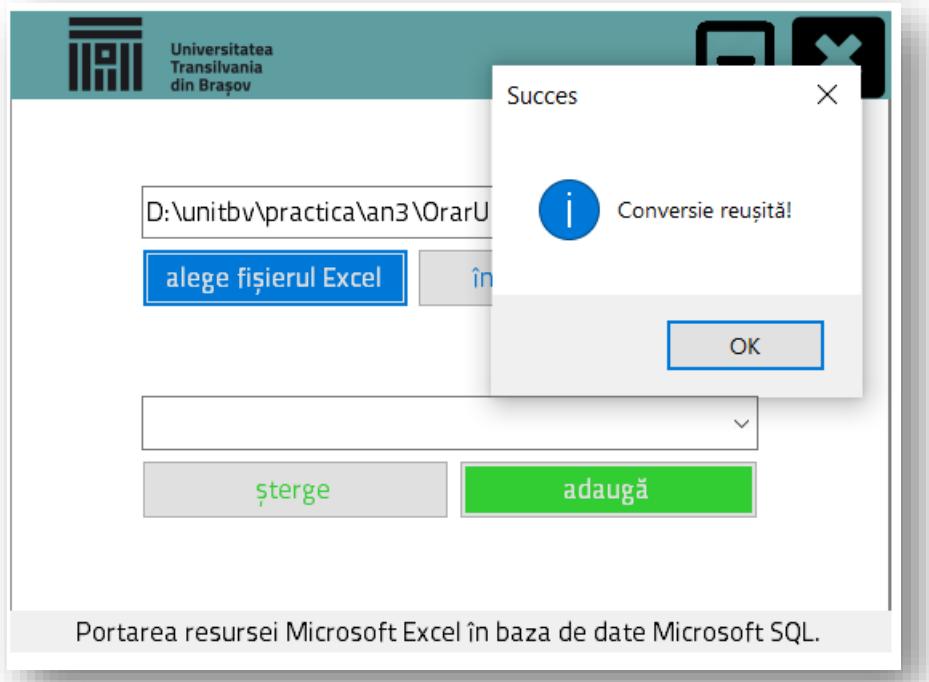


Fig 1.8.3

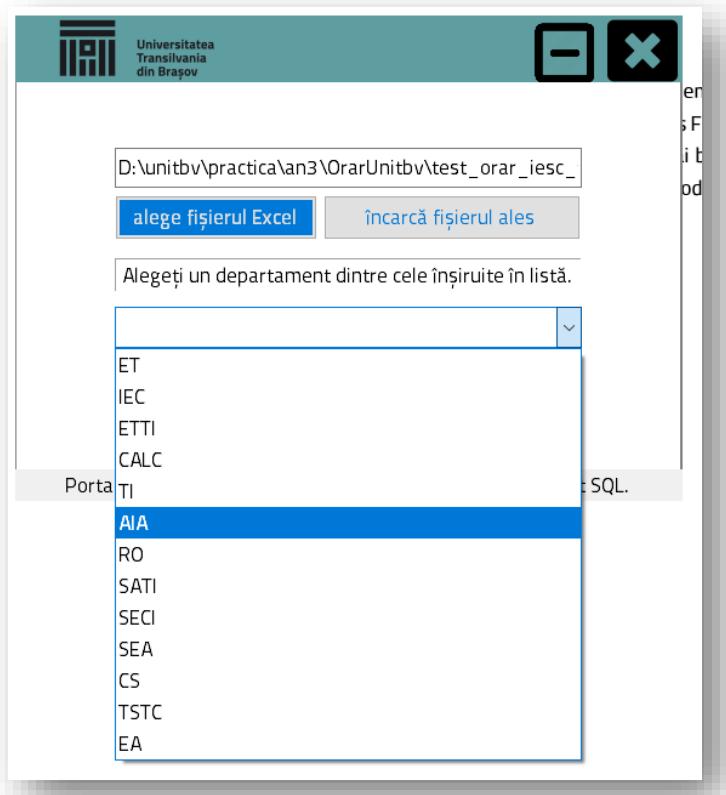


Fig 1.8.4

The screenshot shows a SQL query results table. The query is:

```
use TLSU;
select * from AIA;
select CadruDidactic, Ziua, IntervalOrar, Locatie, Semigrupa from AIA where
select MaterieAbreviere, CadruDidactic, Tip, Ziua, IntervalOrar, Locatie, Semigrupa
```

The results table has columns: ID, MaterieAbreviere, Materie, Tip, CadruDidactic, Ziua, IntervalOrar, Locatie, Semigrupa. The data is as follows:

ID	MaterieAbreviere	Materie	Tip	CadruDidactic	Ziua	IntervalOrar	Locatie	Semigrupa
121	Engleza	NULL	C	Pirnuta_O_A	VINERI	12.00-13.50	VIV4	4LF433B2
122	Fizica	NULL	L	Florin_L	LUNI	14.00-15.50	L06_P_OS	4LF433B2
123	Fizica	NULL	C	Florin_L	MIERCURI	16.00-17.50	VIV7	4LF433B2
124	GAC	NULL	C	Popa_Lumi	VINERI	8.00-9.50	VIV3	4LF433B2
125	GAC	NULL	C	Popa_Lumi	VINERI	8.00-9.50	VIV3	4LF434A1
126	GAC	NULL	L	Popa_Lumi	LUNI	10.00-11.50	VIV5A	4LF434A1
127	ProcDate	NULL	L	Manea_C_Ad	LUNI	12.00-13.50	VIII8	4LF434A1
128	Engleza	NULL	S	Pirnuta_O_A	LUNI	14.00-15.50	VP20	4LF434A1
129	Fizica	NULL	L	Florin_L	LUNI	16.00-17.50	L06_P_OS	4LF434A1
130	MS	NULL	S	Chirila_A	MARTI	8.00-9.50	VIV3	4LF434A1
131	Electro	NULL	C	Barote_L	MARTI	12.00-13.50	VIV4	4LF434A1
132	Electro	NULL	C	Barote_L	MARTI	12.00-13.50	VIV4	4LF434A2
133	Electro	NULL	C	Barote_L	MARTI	14.00-15.50	VIV4	4LF434A1
134	Electro	NULL	C	Barote_L	MARTI	14.00-15.50	VIV4	4LF434A2
135	Pedag	NULL	S	Florescu_C	MARTI	18.00-19.50	TII2	4LF434A1
136	PCLP2	NULL	L	Bratu_D	MARTI	20.00-21.50	VIII12	4LF434A1
137	ProcDate	NULL	C	Manea_C_Ad	MIERCURI	8.00-9.50	VIV7	4LF434A1
138	ProcDate	NULL	C	Manea_C_Ad	MIERCURI	8.00-9.50	VIV7	4LF434A2

Fig 1.8.5

Departamentul Automatică și Tehnologia Informației  
Proiect Agendă Spring Boot

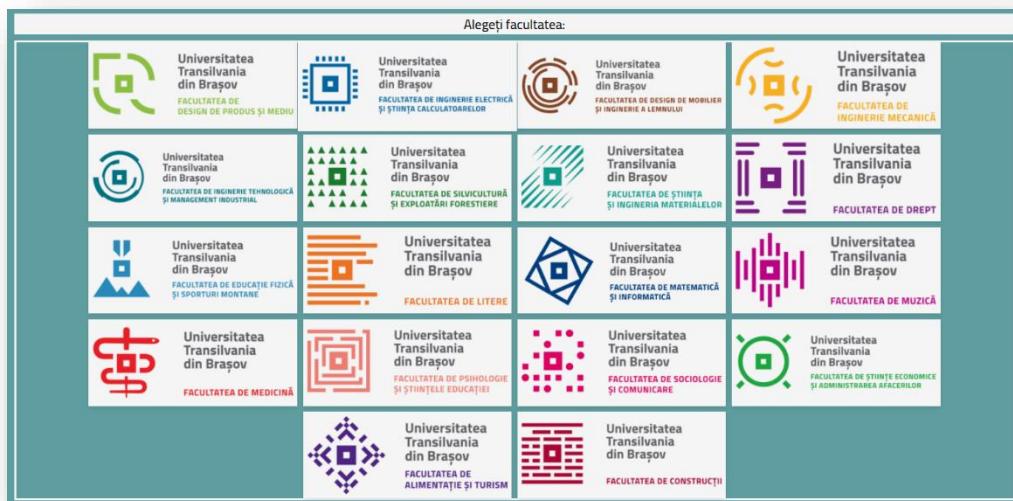


Fig 1.8.6

### Datele de autentificare

Server:

Facultate: IESC

Nume de utilizator:

Parolă:

Fig 1.8.7

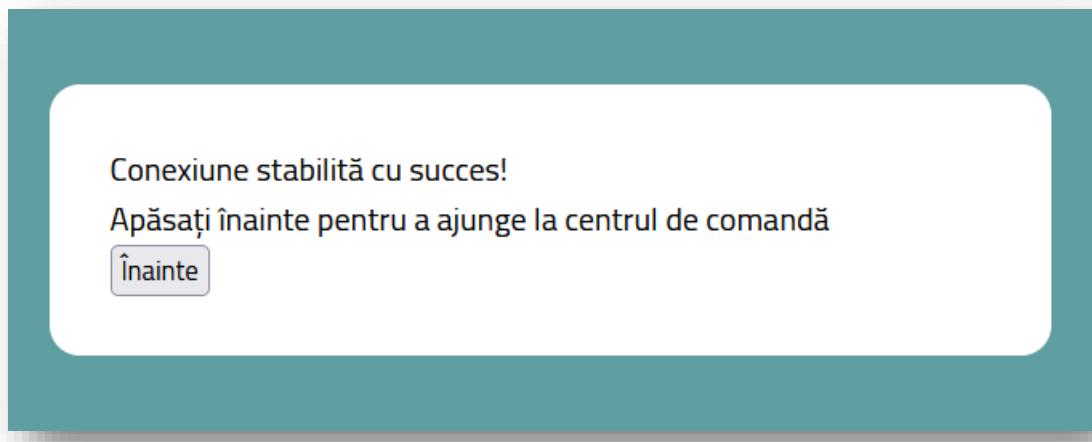


Fig 1.8.8

**Informațiile vor fi redate în funcție de câmpurile complete**

Departament:	AIA
Materie(abreviere):	MS
Materie(nume complet):	funcție în lucru
Tip:	S
Cadru didactic:	Moldoveanu_F_D, Beldianu_I, etc.
Ziua:	LUNI, VINERI, etc.
Interval orar:	8,00-9,50, 14,00-15,50, 20,00-21,50
Locație:	VIV7, PIII1, AULA_NT, etc.
Semigrupă:	4LF431A1, 4LF421A1, 4LF422B2, etc

**Caută**

**Înapoi**

Fig 1.8.9

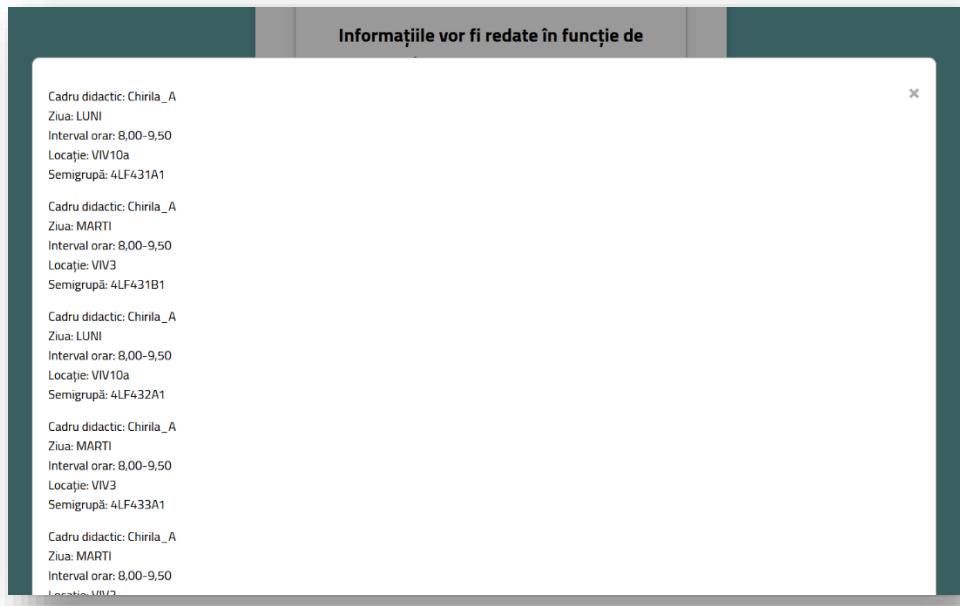


Fig 1.8.10

```

select * from AIA;
select CadruDidactic, Ziua, IntervalOrar, Locatie, Semigrupa from AIA where MaterieAbreviere = 'MS' and Tip = 'S';
select MaterieAbreviere, CadruDidactic, Tip, Ziua, IntervalOrar, Locatie from AIA where Tip = 'C' and Semigrupa = '4LF431B1';

```

Results

	CadruDidactic	Ziua	IntervalOrar	Locatie	Semigrupa
1	Chirila_A	LUNI	8:00-9:50	VIV10a	4LF431A1
2	Chirila_A	MARTI	8:00-9:50	VIV3	4LF431B1
3	Chirila_A	LUNI	8:00-9:50	VIV10a	4LF432A1
4	Chirila_A	MARTI	8:00-9:50	VIV3	4LF433A1
5	Chirila_A	MARTI	8:00-9:50	VIV3	4LF434A1

Fig 1.8.11

## 1.9. Posibilități de îmbunătățire

- Realizare unei aplicații care să poată să detecteze toate varianțele de orare ale facultăților, testarea s-a făcut pe orarul IESC, însă toate celelalte orare similare (făcute după aceeași structură) vor funcționa,
- Îmbunătățirea timpului de încărcare și de rularea a aplicației tip formular,
- În partea de aplicație Web diversificarea comenzilor pe care le poate îndeplini utilizatorul, însemnând mai mult decât afișare de informații filtrate.