Ministerul Educației, Culturii și Cercetării al Republicii Moldova

IP Centrul de Excelență în Informatică și Tehnologii Informaționale

**RAPORTUL**

**STAGIULUI DE PRACTICĂ**

Elevul:  **Dubenco Alexandru**

Grupa: **B-2031**

Specialitatea: **Administrarea bazelor de date**

Conducătorul stagiului de practică

de la Centrul de Excelență

**Jumbei Olga**

Chişinău **2031**

**Cuprins:**

[**Conținutul activităților și sarcinilor de lucru: 2**](#_heading=h.gjdgxs)

[**Algoritmul realizat 2**](#_heading=h.1fob9te)

[1)Fisierul JSON stabilit dupa cheia API 2](#_heading=h.uzs01xddz60q)

[2)clasa in care declaram clasele meteo 3](#_heading=h.6joj7l5xeai)

[3)forma de logare si forma principala 4](#_heading=h.g5tuzqxn7sph)

[4)baza de date 8](#_heading=h.fp76tfzgfmq)

[5)Clasele folosite la costumizarea aplicatiei 8](#_heading=h.kcepqjex2owt)

[1)Custom Button 8](#_heading=h.h90a70c3rdd1)

[2)Custom Panel 11](#_heading=h.o0nvcd87fd3v)

[3)Custom Eclipse 12](#_heading=h.ri3e3z6mrmt7)

[6) Functionalitatea 13](#_heading=h.pc1tg6ef2zgc)

[**Concluzia 13**](#_heading=h.lci8hkkaluqq)

# Conținutul activităților și sarcinilor de lucru:

# Algoritmul realizat

## 1)Fisierul JSON stabilit dupa cheia API

{

"coord": {

"lon": 28.8575,

"lat": 47.0056

},

"weather": [

{

"id": 800,

"main": "Clear",

"description": "clear sky",

"icon": "01d"

}

],

"base": "stations",

"main": {

"temp": 293.39,

"feels\_like": 292.57,

"temp\_min": 293.39,

"temp\_max": 293.39,

"pressure": 1016,

"humidity": 42

},

"visibility": 10000,

"wind": {

"speed": 8.23,

"deg": 200

},

"clouds": {

"all": 0

},

"dt": 1684225441,

"sys": {

"type": 1,

"id": 7022,

"country": "MD",

"sunrise": 1684204122,

"sunset": 1684258401

},

"timezone": 10800,

"id": 618426,

"name": "Chișinău",

"cod": 200

}

## 2)clasa in care declaram clasele meteo

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practica

{

internal class InfoWeather

{

public class coord

{

public double lon { get; set; }

public double lat { get; set; }

}

public class weather

{

public string main { get; set; }

public string description { get; set; }

}

public class main

{

public double temp { get; set; }

public double temp\_min { get; set; }

public double temp\_max { get; set; }

public double pressure { get; set; }

public double humidity { get; set; }

}

public class wind

{

public double speed { get; set; }

public int deg { get; set;}

}

public class sys

{

public long sunrise { get; set; }

public long sunset { get; set; }

public string country { get; set; }

}

public class root

{

public coord coord { get; set; }

public List<weather> weather { get; set;}

public main main { get; set; }

public wind wind { get; set; }

public sys sys { get; set; }

}

}

}

## 

## 

## 

## 

## 

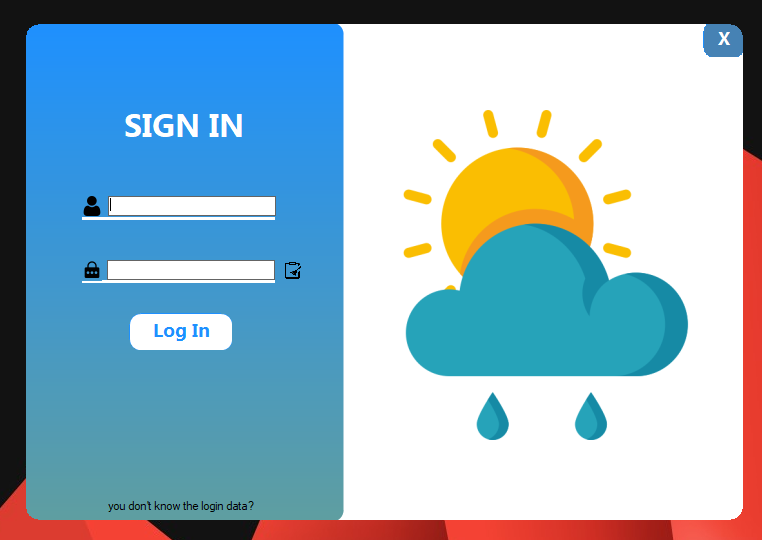
## 

## 

## 

## 

## 3)forma de logare si forma principala



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using static System.Net.Mime.MediaTypeNames;

namespace Practica

{

public partial class LogIn : Form

{

public LogIn()

{

InitializeComponent();

}

SqlConnection conn = new SqlConnection(@"Data Source=L08\_PC05\MSSQLSERVER02;Initial Catalog=LogIn\_Admin;Integrated Security=True");

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

{

}

private void btn\_closeL\_Click(object sender, EventArgs e)

{

this.Close();

}

private void artanButton1\_Click(object sender, EventArgs e)

{

String username, user\_password;

username = tb\_username.Text;

user\_password = tb\_password.Text;

try

{

String querry = "SELECT \* FROM Login\_new WHERE username = '"+tb\_username.Text+"' AND password = '"+tb\_password.Text+"' ";

SqlDataAdapter sda = new SqlDataAdapter(querry, conn);

DataTable dTable = new DataTable();

sda.Fill(dTable);

if(dTable.Rows.Count > 0 )

{

username = tb\_username.Text;

user\_password = tb\_password.Text;

//pagina ce trebuie pe care ne comutam

Form1 form2 = new Form1();

form2.Show();

this.Hide();

}

else

{

MessageBox.Show("Invalid login details", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

tb\_username.Clear();

tb\_password.Clear();

//to focus username

tb\_username.Focus();

}

}

catch

{

MessageBox.Show("error");

}

finally

{

conn.Close();

}

}

private void pic\_clear\_Click(object sender, EventArgs e)

{

tb\_username.Clear();

tb\_username.Clear();

tb\_username.Focus();

}

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

{

pic\_mem.Visible = true;

label\_mem.Visible=true;

}

protected override void OnMouseDown(MouseEventArgs e)

{

base.OnMouseDown(e);

if (e.Button == MouseButtons.Left)

{

this.Capture = false;

Message msg = Message.Create(this.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);

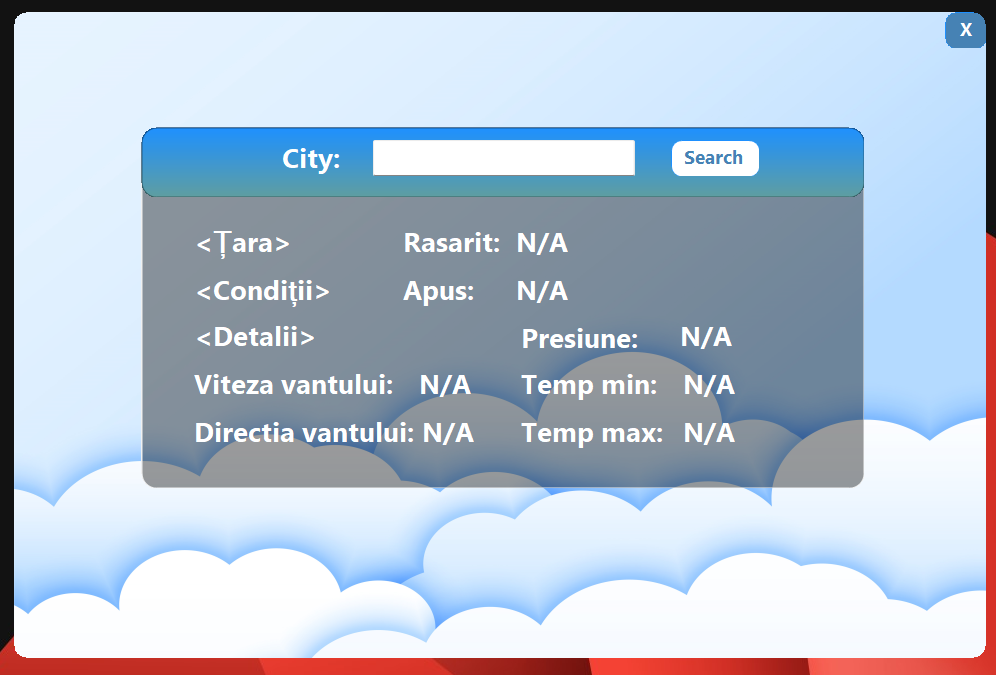
this.WndProc(ref msg);

}

}

}

}



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Newtonsoft.Json;

using System.Net;

namespace Practica

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void artanButton1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form1\_Load(object sender, EventArgs e)

{

artanPanel1.BackColor = Color.FromArgb(100,0,0,0);

}

string APIKey = "7d9ab23e7c45a245acd20efc72ec7acb";

private void btn\_search\_Click(object sender, EventArgs e)

{

getWeather();

}

private void getWeather()

{

using (WebClient web = new WebClient())

{

string url = string.Format("https://api.openweathermap.org/data/2.5/weather?q={0}&appid={1}", TbCity.Text, APIKey);

var json = web.DownloadString(url);

InfoWeather.root Info = JsonConvert.DeserializeObject<InfoWeather.root>(json);

lab\_condition.Text = Info.weather[0].main;

lab\_detail.Text = Info.weather[0].description;

lab\_sunset.Text = convertDateTime(Info.sys.sunset).ToString();

lab\_sunrise.Text = convertDateTime(Info.sys.sunrise).ToString();

lab\_windspeed.Text = Info.wind.speed.ToString();

lab\_pressure.Text = Info.main.pressure.ToString();

label\_country.Text = Info.sys.country.ToString();

lab\_deg.Text = Info.wind.deg.ToString();

lab\_temp\_min.Text = Info.main.temp\_min.ToString();

lab\_temp\_max.Text = Info.main.temp\_max.ToString();

}

}

DateTime convertDateTime(long millisec)

{

DateTime day = new DateTime(1970,1,1,0,0,0, DateTimeKind.Utc);

day = day.AddSeconds(millisec).ToLocalTime();

return day;

}

protected override void OnMouseDown(MouseEventArgs e)

{

base.OnMouseDown(e);

if (e.Button == MouseButtons.Left)

{

this.Capture = false;

Message msg = Message.Create(this.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);

this.WndProc(ref msg);

}

}

private void label4\_Click(object sender, EventArgs e)

{

}

}

}

## 4)baza de date

create database Login\_admin;

use Login\_admin;

create table Login\_new(

username varchar(50)

,password varchar(50)

)

insert into Login\_new(username,password)

values

('Admin','admin123')

////codul de conectare a BD cu aplicatia

SqlConnection conn = new SqlConnection(@"Data Source=L08\_PC05\MSSQLSERVER02;Initial Catalog=LogIn\_Admin;Integrated Security=True");

## 5)Clasele folosite la costumizarea aplicatiei

### 1)Custom Button

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ArtanButton

{

internal class ArtanButton:Button

{

//Fields

private int borderSize = 0;

private int borderRadius = 50;

private Color borderColor = Color.DodgerBlue;

//Properties

public int BorderSize

{

get => borderSize;

set

{

borderSize = value; Invalidate();

}

}

public int BorderRadius

{

get => borderRadius;

set

{

borderRadius = (value <= Height)? value : Height; Invalidate();

}

}

public Color BackgroundColor

{

get => BackColor;

set { BackColor = value; }

}

public Color TextColor

{

get => ForeColor;

set { ForeColor = value; }

}

//Constructor

public ArtanButton()

{

Size = new Size(200, 100);

FlatAppearance.BorderSize = 0;

FlatStyle = FlatStyle.Flat;

BackColor = Color.DodgerBlue;

ForeColor = Color.White;

Resize += new EventHandler(button\_Resize);

}

private void button\_Resize(object sender, EventArgs e)

{

if (borderRadius > Height) borderRadius = Height;

}

//Methods

private GraphicsPath GetFigurePath(RectangleF rectangle,float radius)

{

GraphicsPath graphicsPath = new GraphicsPath();

graphicsPath.StartFigure();

graphicsPath.AddArc(rectangle.X, rectangle.Y, radius, radius, 180, 90);

graphicsPath.AddArc(rectangle.Width - radius, rectangle.Y, radius, radius, 270, 90);

graphicsPath.AddArc(rectangle.Width - radius, rectangle.Height - radius, radius, radius, 0, 90);

graphicsPath.AddArc(rectangle.X, rectangle.Height - radius, radius, radius, 90, 90);

graphicsPath.CloseFigure();

return graphicsPath;

}

protected override void OnPaint(PaintEventArgs pevent)

{

base.OnPaint(pevent);

pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;

RectangleF rectangleSurface = new RectangleF(0, 0, Width, Height);

RectangleF rectangleBorder = new RectangleF(1, 1, Width - 0.5F, Height - 1);

if (borderRadius > 1)

{

using (GraphicsPath graphicsPathSurface = GetFigurePath(rectangleSurface, borderRadius))

using (GraphicsPath graphicsPathBorder = GetFigurePath(rectangleSurface, borderRadius - 1F))

using (Pen penSurface = new Pen(Parent.BackColor, 2))

using (Pen penBorder = new Pen(borderColor, borderSize))

{

penBorder.Alignment = PenAlignment.Inset;

Region = new Region(graphicsPathSurface);

pevent.Graphics.DrawPath(penBorder, graphicsPathSurface);

if (borderSize >= 1)

pevent.Graphics.DrawPath(penBorder, graphicsPathBorder);

}

}

else

{

Region = new Region(rectangleSurface);

if (borderSize >= 1)

using (Pen penBorder = new Pen(borderColor, borderSize))

{

penBorder.Alignment = PenAlignment.Inset;

pevent.Graphics.DrawRectangle(penBorder, 0 ,0 , Width-1, Height-1);

}

}

}

protected override void OnHandleCreated(EventArgs e)

{

base.OnHandleCreated(e);

Parent.BackColorChanged += new EventHandler(Container\_BackColorChanged);

}

private void Container\_BackColorChanged (object sender, EventArgs e)

{

if (DesignMode) Invalidate();

}

}

}

### 2)Custom Panel

g System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ArtanComponent

{

public class ArtanPanel : Panel

{

//Fields

private int borderRadius = 30;

private float gradientAngle = 90F;

private Color gradientTopColor = Color.DodgerBlue;

private Color gradientBottomColor = Color.CadetBlue;

//Constructor

public ArtanPanel()

{

this.BackColor = Color.White;

this.ForeColor = Color.Black;

this.Size = new Size(350, 200);

}

//Properties

public int BorderRadius

{

get => borderRadius;

set { borderRadius = value; this.Invalidate(); }

}

public float GradientAngle

{

get => gradientAngle;

set { gradientAngle = value; this.Invalidate(); }

}

public Color GradientTopColor

{

get => gradientTopColor;

set { gradientTopColor = value; this.Invalidate(); }

}

public Color GradientBottomColor

{

get => gradientBottomColor;

set { gradientBottomColor = value; this.Invalidate(); }

}

//Methods

private GraphicsPath GetArtanPath(RectangleF rectangle, float radius)

{

GraphicsPath graphicsPath = new GraphicsPath();

graphicsPath.StartFigure();

graphicsPath.AddArc(rectangle.Width - radius, rectangle.Height - radius, radius, radius, 0, 90);

graphicsPath.AddArc(rectangle.X, rectangle.Height - radius, radius, radius, 90, 90);

graphicsPath.AddArc(rectangle.X, rectangle.Y, radius, radius, 180, 90);

graphicsPath.AddArc(rectangle.Width - radius, rectangle.Y,radius, radius, 270, 90);

graphicsPath.CloseFigure();

return graphicsPath;

}

//Overriden methods

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e);

//gradient

e.Graphics.SmoothingMode= SmoothingMode.AntiAlias;

LinearGradientBrush brushArtan = new LinearGradientBrush(this.ClientRectangle, this.GradientTopColor, this.GradientBottomColor, this.gradientAngle);

Graphics graphicsArtan = e.Graphics;

graphicsArtan.FillRectangle(brushArtan, ClientRectangle);

//BorderRadius

RectangleF rectangleF = new RectangleF (0,0, this.Width , this.Height);

if (borderRadius > 2)

{

using (GraphicsPath graphicsPath = GetArtanPath(rectangleF, borderRadius))

using (Pen pen = new Pen(this.Parent.BackColor, 2))

{

this.Region = new Region(graphicsPath);

e.Graphics.DrawPath(pen, graphicsPath);

}

}else this.Region = new Region (rectangleF);

}

}

}

### 3)Custom Eclipse

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ElipseArtanAcademy1

{

public class ElipseControlArtan: Component

{

[DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]

private static extern IntPtr CreateRoundRectRgn(int nL, int nT, int nR, int nB, int nWidthEllipse, int nHeightEllipse);

private Control control;

private int cornerRadius = 25;

public Control TargetControl

{

get { return control; }

set

{

control = value;

control.SizeChanged += (sender, eventArgs) => control.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0, control.Width, control.Height, cornerRadius, cornerRadius));

}

}

public int CornerRadius

{

get { return cornerRadius; }

set

{ cornerRadius = value;

if (control != null)

control.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0, control.Width, control.Height, cornerRadius, cornerRadius));

}

}

}

}

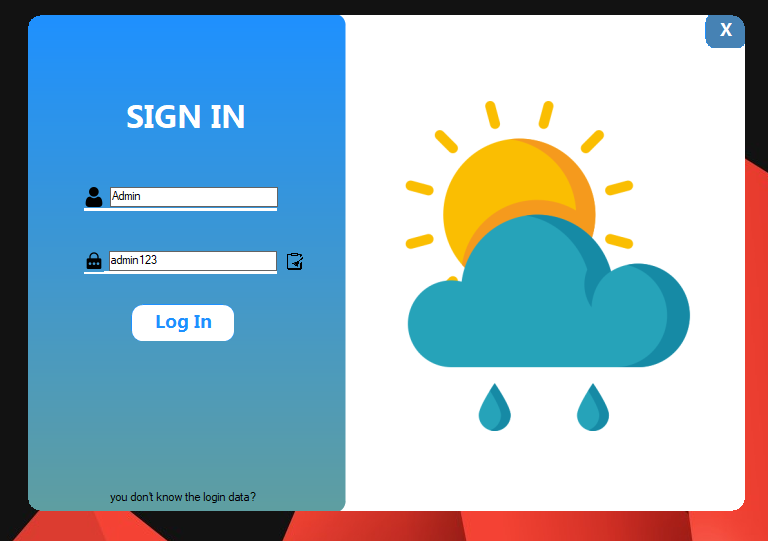
## 

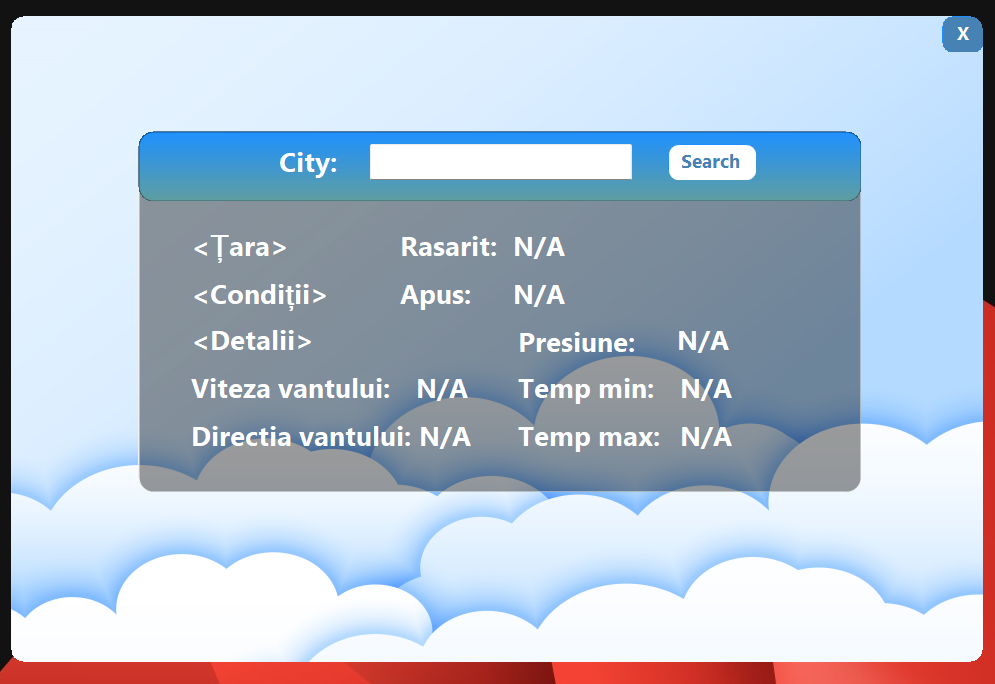
## 

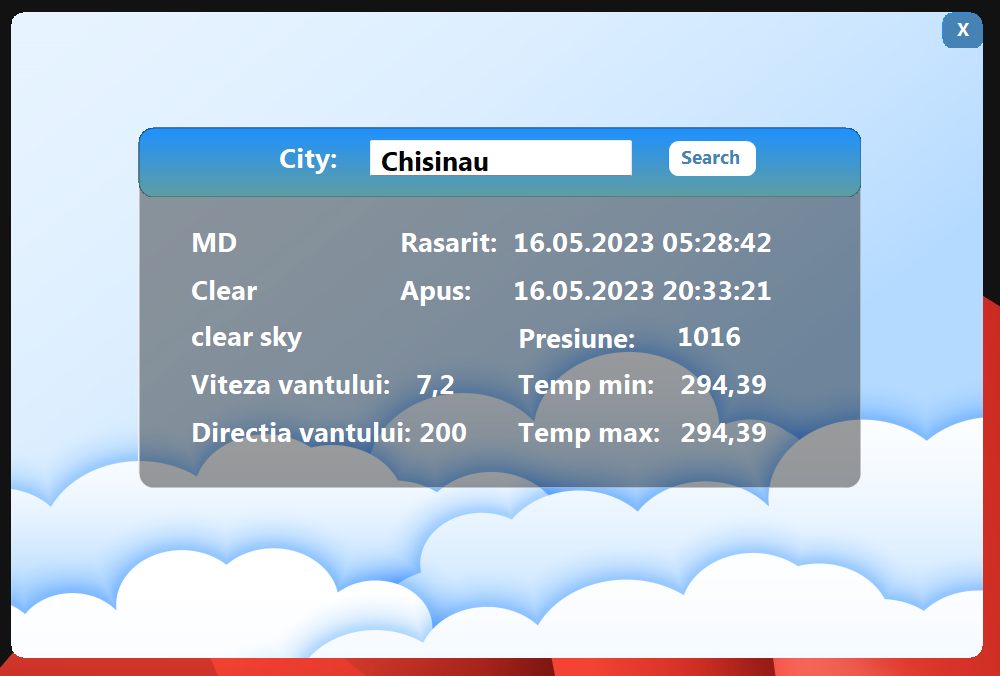
## 

## 

## 6) Functionalitatea







# Concluzia

În concluzie, procesul de creare a aplicației desktop meteo folosind API-ul cheie de pe OpenWeatherMap.org a fost un succes. Utilizând trei clase personalizate și integrând o bază de date pentru funcționalitatea de autentificare, am reușit să dezvolt o aplicație utilă și eficientă.

API-ul de la OpenWeatherMap.org ne-a oferit acces la datele meteorologice în timp real, permițând astfel afișarea informațiilor despre vreme în aplicație. Integrarea acestei funcționalități a fost esențială pentru a oferi utilizatorilor informațiile actualizate și precise despre condițiile meteorologice.

Prin crearea celor trei clase personalizate, am reușit să organizăm codul într-un mod structurat și modular. Aceste clase au furnizat instrumente esențiale pentru gestionarea cererilor API, manipularea datelor și afișarea informațiilor în interfața aplicației. Prin separarea responsabilităților în clase distincte, am obținut un cod mai ușor de întreținut și de extins pe viitor.

Integrarea bazei de date și conectarea ei la aplicație au adus un plus de securitate și flexibilitate. Am putut gestiona datele de autentificare ale utilizatorilor și am creat o interfață de logare pentru a permite accesul doar utilizatorilor autorizați. Această funcționalitate a contribuit la creșterea securității și confidențialității datelor utilizatorilor și a oferit o experiență personalizată în aplicație.

În final, aplicația noastră desktop meteo demonstrează un proces de dezvoltare eficient și o integrare reușită a resurselor externe. Prin utilizarea API-ului de la OpenWeatherMap.org, crearea claselor personalizate și integrarea bazei de date