Miejsce na strone tytulowa

Spis treści

[1Temat 3](#__RefHeading___Toc106_972204239)

[2Analiza, projektowanie 3](#__RefHeading___Toc108_972204239)

[2.1Algorytmy, struktury danych, ograniczenia specyfikacji 3](#__RefHeading___Toc110_972204239)

[2.1.1Algorytmy 3](#__RefHeading___Toc248_972204239)

[2.1.2Struktury danych 3](#__RefHeading___Toc250_972204239)

[2.2Analiza obiektowa 3](#__RefHeading___Toc114_972204239)

[3Specyfikacja zewnętrzna 4](#__RefHeading___Toc116_972204239)

[3.1Obsługa programu 4](#__RefHeading___Toc118_972204239)

[3.2Format danych wejściowych 4](#__RefHeading___Toc120_972204239)

[3.3Komunikaty 5](#__RefHeading___Toc122_972204239)

[4Specyfikacja wewnętrzna 6](#__RefHeading___Toc124_972204239)

[4.1Zmienne 6](#__RefHeading___Toc126_972204239)

[4.2Funkcje 6](#__RefHeading___Toc128_972204239)

[5Wydruk 8](#__RefHeading___Toc132_972204239)

1. Temat

Temat tego sprawozdania to pobieranie I analiza danych meteorologicznych. Opisywany program będzie miał na celu okresowe pobieranie danych pogodowych z serwisu internetowego, zapis ich do pliku, proste przetwarzanie oraz wyświetlanie.

1. Analiza, projektowanie
   1. Algorytmy, struktury danych, ograniczenia specyfikacji
      1. Algorytmy

Algorytm użyty w zadaniu opiera się o odczyt danych z serwisu internetowego poprzez zapytania opisane w specyfikacji API. Zapytanie jest następujące:

*api.openweathermap.org/data/2.5/forecast/daily?id={city ID}&cnt={cnt} .*

*City ID* – identyfikator miasta – znaleźć można w udostępnionym przez openweathermap.org pliku json.

* + 1. Struktury danych

Strukturami danych odczytywanych z internetu jak i zapisywanych w plikach tekstowych są pliki json, które można w łatwy sposób parsować bibliotekami języka C#.

* 1. Analiza obiektowa

Aby zapewnić należyty porządek, kod podzielony został na klasy. Główną klasą będzie klasa MainClass, która zawierać będzie metodę Main() oraz pomocnicze metody przetwarzające, analizujące oraz dokonujące akwizycji danych.

Poza klasą główną istnieć będzie także klasa BackgroundTask odpowiedzialna za cykliczne czytanie w tle danych pogodowych z serwera, oraz klasy pomocnicze, potrzebne do deserializacji danych z json:

* Coord – klasa przechowująca współrzędne geograficzne danego miasta
* City – klasa przechowująca dane na temat danego miasta
* Temp – klasa przechowująca temperatury
* Weather – klasa przechowująca warunki pogodowe
* Days – klasa przechowująca dane pogodowe wielu dni
* WeatherData – korzeń powyższych klas, zawiera wszystkie dane pogodowe.

1. Specyfikacja zewnętrzna
   1. Obsługa programu

Program obsługuje się w następujący sposób:

* należy wpisać nazwę miasta
* należy wpisać ilość dni, jakie ma obejmować prognoza (zakres 1..16)
* należy wpisać co ile minut ma następować odczyt danych z serwera (≥1; zaleca się wpisywać w realnych zastosowaniach wartości ≥ 60, ponieważ przewidywania pogody zamieniają się bardzo powoli)
* po wykonaniu powyższych czynności program zacznie cyklicznie zapisywać dane pogodowe do plików (sygnowanych datą odczytu z serwera), oraz przetwarzać i wyświetlać dane już pobrane
* program kończy się poprzez wciśnięcie klawisza *enter*
  1. Format danych wejściowych

Format danych pobieranych z serwera, zapisywanych i odczytywanych z/do plikóœ jest następujący:

{"cod":"200","message":0.0032,

"city":{"id":1851632,"name":"Shuzenji",

"coord":{"lon":138.933334,"lat":34.966671},

"country":"JP"},

"cnt":10,

"list":[{

"dt":1406080800,

"temp":{

"day":297.77,

"min":293.52,

"max":297.77,

"night":293.52,

"eve":297.77,

"morn":297.77},

"pressure":925.04,

"humidity":76,

"weather":[{"id":803,"main":"Clouds","description":"broken clouds","icon":"04d"}],}

]}

Parametry:

* city – dane miasta
* city.id – identyfikator miasta
* city.name – nazwa miasta
* city.coord – współrzędne geograficzne miasta
  + city.coord.lat – szerokość geograficzna
  + city.coord.lon – długość geograficzna miasta
* city.country – kod kraju
* cod – kod zwrotny zapytania GET
* message – wiadomość zwrotna zapytania GET
* cnt – ilość linii (dni) zwrócona przez zapytanie
* list – zestawy danych pogodowych na kilka dni
  + list.dt – czas prognozy
  + list.temp - temperatura
    - list.temp.day – temperatura w dzień [K]
    - list.temp.min – dobowa temperatura minimalna [K]
    - list.temp.max – dobowa temperatura maksymalna [K]
    - list.temp.night – temperatura w nowy [K]
    - list.temp.eve – temperatura wieczorem [K]
    - list.temp.morn – temperatura nad ranem [K]
  + list.pressure – ciśnienie atmosferyczne [hPa]
  + list.humidity – wilgotność [%]
  + list.weather – dalsze dane pogodowe
    - list.weather.id – kod warunków pogodowych
    - list.weather.main – nazwa grupy warunków pogodowych
    - list.weather.description – opis warunków pogodowych
    - list.weather.icon – identyfikator ikony warunków pogodowych
  + list.speed – prędkość wiatru [m/s]
  + list.deg – kierunek wiatru [°]
  + list.clouds – stopień zachmurzenia [%]
  1. Komunikaty

Komunikaty generowane przez program:

* *Nie można znalezc miasta* – miast wpisane przez użytkownika nie figuruje w bazie miast, przez co jego kod nie może być znaleziony
* *Bledne dane* – użytkownik wpisał nieprawidłowe dane
* *Brak polaczenia z serwerem* – nie udało się pobrać danych pogodowych z powodu braku komunikacji z serwisem internetowym
* *Nie znaleziono bazy miast –* plik json zawierający listę miast i odpowiadające im kody miast nie został znaleziony pod daną ścieżką

1. Specyfikacja wewnętrzna
   1. Zmienne

Zmienne klasy MainClass metody Main():

* String city – miasto wpisane przez użytkownika
* int days – ilość dni, które ma obejmować prognoza
* int okresMin – okres pobierania nowych danych z serwera
* int cityId – identyfikator miasta, wydobyty z bazy json
* BackgroundTask bt1 – obiekt klasy BackgroundTask zawierający metodę, która ma działać w tle
* System.Threading.Thread t1 – obiekt nowego wątku działąjącego w tle

Zmienne klasy BackgroundTask metody keepChecking():

* String weatherDataJson – dane odczytane z serwera poprzez zapytanie
* String dataCzas – aktualna data i czas, służy do określenia kiedy pobrane zostały dane z serwera
* string[] fileArray – tablica zawierająca nazwy plików z danymi, które zawierają aktualnie interesujące nas dane
* string danePogodoweJson – dane json odczytane z danego pliku
* WeatherData danePogodowe – dane pogodowe utworzone z danych tekstowych json z pliku
* Temp srednieTemperatury – obiekt zawierający obliczone średnie temperatury z kilku dni
* double sredniaRoznica – srednia różnica między temperaturami maksymalnymi i minimalnymi na przestrzeni kilku dni
  1. Funkcje

Metody klasy MainClass:

* public static void Main(string[]) - metoda główna, w niej zaczyna się program
* public static int getCityID(String, String) – metoda ta pobierać będzie z pliku identyfikator miasta na podstawie nazwy wprowadzonej przez użytkownika
* public static String getForecast(int, int) – metoda pobierająca dane pogodowe w formacie json z serwera przy użyciu skonstruowanego zapytania
* public static WeatherData getWeatherDataFromJson(String) – metoda tworząca na podstawie danych json obiekt typu WeatherData zawierający wszystkie dane pobrane z serwera w danym zapytaniu, aby umożliwić do nich dostęp obiektowy
* public static Temp convertKelvinToCelsius(Temp) – temperatury są pobierane z serwera w Kelwinach, zatem konieczna jest ich konwersja na stopnie Celsjusza. Metoda tworzy nowy obiekt typu Temp, na podstawie podanego obiektu typu Temp (Zawartego w obiekcie typu WeatherData).
* public static Temp averageTempDays(WeatherData) – metoda zwraca obiekt typu Temp zawierający temperatury uśrednione z zakresu kilku dni.
* double AverageDelta(Temp) – metoda zwraca uśrednioną wartość różnic między temperaturami maksymalnymi i minimalnymi z zakresu kilku dni.

1. Wydruk

using System;

using System.IO;

namespace pogoda {

class MainClass {

public static void Main (string[] args) {

String city;

int days;

int okresMin;

int cityId;

BackgroundTask bt1;

System.Threading.Thread t1;

Console.WriteLine ("Program pobierajacy i analizujacy dane pogodowe pochodzace z serwisu openweathermap.org");

try {

Console.Write ("Prosze wpisac nazwe miasta (bez polskich znakow): ");

city = Console.ReadLine();

Console.Write ("Prosze wpisac liczbe dni, ktore ma obejmowac prognoza <1..16>: ");

days = Int32.Parse(Console.ReadLine());

Console.Write ("Prosze wpisac okres sprawdzania pogody [min]: ");

okresMin = Int32.Parse(Console.ReadLine());

} catch {

Console.WriteLine ("Bledne dane"); return;

}

try {

cityId = getCityID (city, "../../dane/city.list.json"); // lista pobrana z http://bulk.openweathermap.org/sample/city.list.json.gz

} catch {

Console.WriteLine ("Nie znaleziono bazy miast w folderze /dane/city.list.json"); return;

}

if (cityId == 0) {

Console.WriteLine ("Nie mozna znalezc miasta"); return;

}

Console.WriteLine ("ID miasta to: " + cityId);

Console.WriteLine ("Aby zakonczyc nacisnij enter...");

bt1 = new BackgroundTask(cityId, days, okresMin);

t1 = new System.Threading.Thread(new System.Threading.ThreadStart(bt1.keepChecking));

t1.Start();

//while (!t1.IsAlive);

Console.ReadLine();

t1.Abort (); t1.Join ();

Console.WriteLine ("Koniec programu.");

}

/////////// metody pomocnicze

public static double averageDelta(Temp tempAvg) {

//Console.WriteLine ("metoda averageDelta()");

//Console.WriteLine ("averageDelta: " + (tempAvg.max - tempAvg.min));

//Console.WriteLine ("koniec metody averageDelta()");

return tempAvg.max - tempAvg.min;

}

public static Temp averageTempDays(WeatherData wd1) { // liczenie sredniej

//Console.WriteLine ("metoda averageTempDays()");

Temp temp1 = new Temp();

//day

foreach (Days d in wd1.list)

temp1.day += d.temp.day;

temp1.day /= wd1.list.Length;

//Console.WriteLine ("avg day: " + temp1.day);

//min

foreach (Days d in wd1.list)

temp1.min += d.temp.min;

temp1.min /= wd1.list.Length;

//Console.WriteLine ("avg min: " + temp1.min);

//max

foreach (Days d in wd1.list)

temp1.max += d.temp.max;

temp1.max /= wd1.list.Length;

//Console.WriteLine ("avg max: " + temp1.max);

//night

foreach (Days d in wd1.list)

temp1.night += d.temp.night;

temp1.night /= wd1.list.Length;

//Console.WriteLine ("avg night: " + temp1.night);

//eve

foreach (Days d in wd1.list)

temp1.eve += d.temp.eve;

temp1.eve /= wd1.list.Length;

//Console.WriteLine ("avg eve: " + temp1.eve);

//morn

foreach (Days d in wd1.list)

temp1.morn += d.temp.morn;

temp1.morn /= wd1.list.Length;

//Console.WriteLine ("avg morn: " + temp1.morn);

//Console.WriteLine ("koniec metody averageTempDays()");

return temp1;

}

public static Temp convertKelvinToCelsius(Temp temp) {

//Console.WriteLine ("metoda convertKelvinToCelsius()");

Temp temp1 = new Temp ();

temp1.day = temp.day - 273.15;

temp1.min = temp.min - 273.15;

temp1.max = temp.max - 273.15;

temp1.night = temp.night - 273.15;

temp1.eve = temp.eve - 273.15;

temp1.morn = temp.morn - 273.15;

//Console.WriteLine ("koniec metody convertKelvinToCelsius()");

return temp1;

}

public static WeatherData getWeatherDataFromJson(String weatherJson) {

//Console.WriteLine ("metoda getWeatherDataFromJson()");

WeatherData wd1 = Newtonsoft.Json.JsonConvert.DeserializeObject<WeatherData> (weatherJson); // deserializacja linii json

//Console.WriteLine ("koniec metody getWeatherDataFromJson()");

return wd1;

}

public static String getForecast(int cityId, int days) { // api.openweathermap.org/data/2.5/forecast/daily?id={city ID}&cnt={cnt}; id: city ID; cnt: number of days returned (from 1 to 16)

//Console.WriteLine ("metoda getForecast()");

string htmlCode;

using (System.Net.WebClient client = new System.Net.WebClient ())

{

client.QueryString.Add("id", cityId.ToString());

client.QueryString.Add("cnt", days.ToString());

htmlCode = client.DownloadString("http://api.openweathermap.org/data/2.5/forecast/daily");

}

//Console.WriteLine (htmlCode);

//Console.WriteLine ("koniec metody getForecast()");

return htmlCode;

}

public static int getCityID(String cityName, String listPath) {

//Console.WriteLine ("metoda getCityID()");

int counter = 0;

string line; // znaleziona linia json zawierajaca szukane miasto

City c1 = null;

System.IO.StreamReader file = new System.IO.StreamReader(listPath); // wczytywanie pliku listy miast

while((line = file.ReadLine()) != null) {

line = line.Replace ("\_id", "id"); // specyfikacja API jest niescisla w sprawie \_id vs id

if ((line.ToLower()).Contains ("\"" + cityName.ToLower() + "\"")) {

Console.WriteLine (line);

break;

}

counter++;

}

file.Close();

if (line == null || line == String.Empty) {

Console.WriteLine ("line == null!");

return 0;

} else {

c1 = Newtonsoft.Json.JsonConvert.DeserializeObject<City> (line); // deserializacja linii json

}

//Console.WriteLine ("koniec metody getCityID()");

return c1.id;

}

}

/////////// klasa w tle

public class BackgroundTask {

int cityId;

int days;

int okresMinut;

public BackgroundTask(int cityId, int days, int okresMinut) {

this.cityId = cityId;

this.days = days;

this.okresMinut = okresMinut;

}

public void keepChecking()

{

while (true)

{

try {

String weatherDataJson = MainClass.getForecast (cityId, days);

String dataCzas = DateTime.Now.ToString (new System.Globalization.CultureInfo("en-GB")).Replace("/", "-").Replace(" ", "\_").Replace(":", "-");

System.IO.File.WriteAllText(cityId + "\_" + days + "\_" + dataCzas + ".json", weatherDataJson);

string[] fileArray = Directory.GetFiles(Directory.GetCurrentDirectory(), cityId + "\_" + days + "\_\*.json");

Console.Clear ();

foreach (string s in fileArray) {

string danePogodoweJson = File.ReadAllText(s);

WeatherData danePogodowe = MainClass.getWeatherDataFromJson (danePogodoweJson);

Temp srednieTemperatury = MainClass.averageTempDays (danePogodowe);

srednieTemperatury = MainClass.convertKelvinToCelsius (srednieTemperatury);

double sredniaRoznica = MainClass.averageDelta (srednieTemperatury);

string dataPliku = (new System.Text.RegularExpressions.Regex (@"/.\*/.+?\_.+?\_|\.json")).Replace(s, "");

Console.WriteLine (dataPliku);

Console.WriteLine ("srednie wartosci temperatur: " + srednieTemperatury.ToString());

Console.WriteLine ("srednia roznica min-max: " + sredniaRoznica);

Console.WriteLine ();

}

System.Threading.Thread.Sleep(okresMinut\*1000\*60);

} catch {

Console.WriteLine ("Brak polaczenia z serwerem");

System.Threading.Thread.Sleep(okresMinut\*1000);

}

}

}

};

/////////// klasy pomocnicze do parsowania json

public class Coord

{

public double lon { get; set; }

public double lat { get; set; }

}

public class City

{

public int id { get; set; }

public string name { get; set; }

public Coord coord { get; set; }

public string country { get; set; }

}

public class Temp

{

public double day { get; set; }

public double min { get; set; }

public double max { get; set; }

public double night { get; set; }

public double eve { get; set; }

public double morn { get; set; }

public override string ToString() {

return String.Format("rano: {0}, dzien: {1}, wieczor: {2}, noc: {3}, min: {4}, max: {5}", morn, day, eve, night, min, max);

}

}

public class Weather

{

public int id { get; set; }

public string main { get; set; }

public string description { get; set; }

public string icon { get; set; }

}

public class Days

{

public int dt { get; set; }

public Temp temp { get; set; }

public double pressure { get; set; }

public int humidity { get; set; }

public Weather[] weather { get; set; }

}

public class WeatherData

{

public string cod { get; set; }

public double message { get; set; }

public City city { get; set; }

public int cnt { get; set; }

public Days[] list { get; set; }

}

}