**COLEGIUL NAŢIONAL „MIHAIL KOGĂLNICEANU” GALAŢI**

**MY WEATHER**

PROIECT PENTRU OBŢINEREA ATESTATULUI PROFESIONAL ÎN INFORMATICĂ

profesor coordonator realizat de

POPESCU MĂDĂLINA ANTON SILVIU CONSTANTIN

clasa a XII-a E

sesiunea mai 2018

Cuprins

[Tema proiectului 1](#_Toc513407283)

[Argument 1](#_Toc513407284)

[Capitolul I – Programare Orientată Obiect 2](#_Toc513407285)

[I.1. Despre Microsoft Visual Studio Community 4](#_Toc513407286)

[I.2. Limbajul C# – generalități 5](#_Toc513407287)

[Capitolul II – Arhitectura aplicației 7](#_Toc513407288)

[II.1. Paginile proiectului 9](#_Toc513407289)

[II.2. Metodele aplicației 12](#_Toc513407290)

[II.3. Ghid de utilizare a aplicației 19](#_Toc513407291)

[Concluzii 21](#_Toc513407292)

[Bibilografie 21](#_Toc513407293)

Index imagini

[Figură 1 – Principiu funcționare XAML 6](#_Toc513421410)

[Figură 2 – Pagină START 8](#_Toc513421411)

[Figură 3 – Meniu aplicație 9](#_Toc513421412)

[Figură 4 – Pagină întâmpinare 9](#_Toc513421413)

[Figură 5 – Pagină feedback imagine 10](#_Toc513421414)

[Figură 6 – Pagină interacţiune „checkboxes” 10](#_Toc513421415)

[Figură 7 – Pagină interacţiune „radio buttons” 11](#_Toc513421416)

[Figură 8 – Pagină interacţiune „text expunere” 11](#_Toc513421417)

[Figură 9 – Funcţie „Porneşte muzica/Opreşte muzica” 20](#_Toc513421418)

[Figură 10 – Funcţie „Temă Noapte/Temă Zi” 20](#_Toc513421419)

# Tema proiectului

Nu este niciun secret pentru nimeni: vremea ne poate influența atât fizic, cât și psihic.

Dovadă în acest sens este apariția unei noi discipline: biometeorologia, care studiază impactul temperaturilor exterioare și al climei asupra sănătății umane.

Schimbarea bruscă a stării vremii atrage atenția asupra capacității de adaptare a omului. În primul rând, vorbim despre adaptarea fiziologică, iar ulterior despre adaptarea psihică, fără a putea preciza care dintre cele două o determină pe cealaltă. Ambele variante sunt adevărate.

Cum se joacă vremea cu sănătatea şi cu psihicul nostru? Este esenţial ca fiecare dintre noi să avem măcar idee despre ceea ce suntem, să ne cunoaştem corpul, să înţelegem care este impactul schimbărilor de vreme asupra noastră şi cum ne putem „împrieteni” cu ea. „Vremea mea” este tema acestui atestat la informatică realizat în limbajul C#.

# Argument

Adolescența, perioadă de mari transformări datorate dezvoltării fizice și psihice, este etapa care începe în timpul pubertății și durează până la vârsta legală de adult (majoratul). Ca urmare a transformărilor biologice, psihologice şi emoţionale, adolescenţa reprezintă, fără îndoială, perioada cea mai tumultoasă din întreaga viață. Putem spune că adolescentul este schimbător ca şi vremea.

Pe de altă parte, vremea, fie că este frumoasă, fie că este urâtă, are un efect profund asupra vieţii adolescenţilor. Le influenţează organismul, voinţa şi emoţiile. Ce pot face ei pentru a nu fi dependenți de mișcarea sinusoidală, colerică a vremii? A fi informați niciodată nu ar putea avea consecințe negative. A fi la curent cu fenomenele care pot fi predictibile înseamnă a avea ocazia de a îşi adapta viața și stilul de viață.

Proiectul „My Weather” dezvoltă subiecte tratate în biometeorologie şi psihologie, adaptate nevoii de cunoaştere şi înţelegerii adolescentului. În 16 pagini sunt prezentate informaţii utile lui, cu privire la problema influenţei vremii asupra tuturor şi la soluţia pentru această problemă. Layout-ul aplicaţiei, imaginile, cromatica şi fonturile folosite creează un design plăcut ochiului. Meniul este simplu de utilizat, făcând posibilă o navigare rapidă printre paginile aplicaţiei.

# Capitolul I Programare Orientată Obiect

Programarea orientată pe obiect (Programare Orientată Obiectual) este unul dintre cei mai importanți pași făcuți în evoluția [limbajelor de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) spre o mai puternică abstractizare în implementarea programelor. Ea a apărut din necesitatea exprimării problemei într-un mod mai natural ființei umane. Astfel, unitățile care alcătuiesc un program se apropie mai mult de modul nostru de a gândi decât modul de lucru al calculatorului.

Până la apariția programării orientate pe obiect, programele erau implementate în [limbaje de programare](https://ro.wikipedia.org/wiki/Limbaj_de_programare) procedurale (C, Pascal) sau în limbaje care nici măcar nu ofereau o modalitate de grupare a instrucțiunilor în unități logice (funcții, proceduri) cum este cazul limbajului de asamblare (asembler). Altfel spus, o problemă preluată din natură trebuia fragmentată în repetate rânduri astfel încât să se identifice elementele distincte, implementabile într-un limbaj de programare.

O mare problemă a [programării procedurale](https://ro.wikipedia.org/wiki/Programare) era separarea datelor de unitățile care prelucrau aceste date (subrutinele), ceea ce făcea foarte dificilă extinderea și întreținerea unui program. Astfel, s-a pus problema ca cele două [entități](https://ro.wikipedia.org/wiki/Entitate) (date și subrutine) să fie grupate într-un anumit mod, astfel încât subrutinele să "știe" în permanență ce date prelucrează și, mai mult decât atât, ele să formeze un modul, adică o unitate care separă implementarea de interfață, ceea ce implică posibilitatea refolosirii codului.

A apărut, astfel, conceptul de [clasă](https://ro.wikipedia.org/wiki/Clas%C4%83). [Clasa](https://ro.wikipedia.org/wiki/Clasa) realizează, în speță, ceea ce am văzut mai înainte: grupează datele și unitățile de prelucrare a acestora într-un modul, unindu-le astfel într-o entitate mult mai naturală. Deși tehnica se numește "Programare Orientată Obiectual", conceptul de bază al ei este [Clasa](https://ro.wikipedia.org/wiki/Clasa). Clasa, pe lângă faptul că abstractizează foarte mult analiza/sinteza problemei, are proprietatea de generalitate, ea desemnând o mulțime de obiecte care împart o serie de proprietăți.

De exemplu: Clasa "floare" desemnează toate plantele care au flori, precum clasa "Fruct" desemnează toate obiectele pe care noi le identificam ca fiind fructe.

Bineînțeles, în implementarea efectivă a programului nu se lucrează cu entități abstracte, precum clasele ci se lucrează cu obiecte, care sunt "instanțieri" ale claselor.

Altfel spus, plecând de la exemplul de mai sus, dacă se construiește un program care să lucreze cu fructe, el nu va prelucra entitatea "fruct", ci va lucra cu entități concrete ale clasei "fruct", adică "măr", "pară", "portocală" etc.

Apare însă următoarea problemă: "cum se trece de la o structură generală la una particulară, mai precis ce înseamnă efectiv procesul de instanțiere?". Instanțierea (trecerea de la clasă la obiect) înseamnă atribuirea unor proprietăți specifice clasei, astfel încât aceasta să indice un obiect anume, care se diferențiază de toate celelalte obiecte din clasă printr-o serie de atribute.

Dacă vom considera ca "fruct\_exotic", care desemnează clasa tuturor fructelor exotice ce conține proprietatea "culoare", atunci atribuind acesteia valoarea "galben" noi vom crea o nouă mulțime (clasa fructelor exotice care au culoarea galbenă), care este o subclasă a clasei "fruct\_exotic", deci realizăm astfel o particularizare. Mai mult decât atât, dacă vom adăuga noi și noi atribute vom individualiza clasa astfel încât să ajungem la un caz concret, care este Obiectul.

PRINCIPII DE BAZĂ

**Abstractizarea**

Este posibilitatea ca un program să ignore unele aspecte ale informației pe care o manipulează, adică posibilitatea de a se concentra asupra esențialului. Fiecare obiect în sistem are rolul unui “actor” abstract, care poate executa acțiuni, își poate modifica și comunica starea și poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilitați. Procesele, funcțiile sau metodele pot fi de asemenea abstracte, și în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea.

**Încapsularea**

Nnumită și ascunderea de informații, asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziție de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfață pentru celelalte obiecte care specifică modul cum acele obiecte pot interacționa cu el.

**Polimorfismul**

Este abilitatea de a procesa obiectele în mod diferit, în funcție de tipul sau de clasa lor. Mai exact, este abilitatea de a redefini metode pentru clasele derivate. De exemplu pentru o clasă Figura putem defini o metodă arie. Dacă Cerc, Dreptunghi etc. ce vor extinde clasa Figura, acestea pot redefini metoda arie.

**Moștenirea**

Organizează și facilitează polimorfismul și încapsularea, permițând definirea și crearea unor clase specializate plecând de la clase (generale) deja definite - acestea pot împărtăși (și extinde) comportamentul lor, fără a fi nevoie de a-l redefini. Aceasta se face de obicei prin gruparea obiectelor în clase și prin definirea de clase ca extinderi ale unor clase existente.

Conceptul de moștenire permite construirea unor clase noi, care păstrează caracteristicile și comportarea, deci datele și funcțiile membru, de la una sau mai multe clase definite anterior, numite clase de bază, fiind posibilă redefinirea sau adăugarea unor date și funcții noi. Se utilizează ideea: ”Anumite obiecte sunt similare, dar în același timp diferite”. O clasă moștenitoare a uneia sau mai multor clase de bază se numește clasă derivată. Esența moștenirii constă în posibilitatea refolosirii lucrurilor care funcționează.

## I.1. Despre Microsoft Visual Studio Community

Visual Studio include un set complet de instrumente de dezvoltare pentru generarea de aplicații ASP.NET, Servicii Web XML, aplicații desktop și aplicații mobile. Visual Basic, Visual C++, Visual C# și Visual J# toate folosesc același mediu de dezvoltare integrat (IDE) care le permite partajarea instrumentelor și facilitează crearea de soluții folosind mai multe limbaje de programare.

Aceste limbaje permit să beneficieze de caracteristicile .NET Framework care oferă acces la tehnologii cheie care simplifică dezvoltarea de aplicații web ASP și XML Web Services cu Visual Web Developer.

Microsoft Visual Studio Comunity este versiunea gratuită a mediului de programare, pusă la dispoziție de către Microsoft şi actualizată regulat.

## I.2. Limbajul C# – generalități

Ca parte a ansamblului strategiei .NET, dezvoltată de Microsoft, la finele anilor ’90 a fost creat limbajul C#. C# este direct înrudit cu C, C++ şi Java.

“Bunicul” limbajului C# este C-ul. De la C, C# moşteneşte sintaxa, multe dintre cuvintele cheie şi operatorii. De asemenea, C# construieşte peste modelul de obiecte definit în C++.

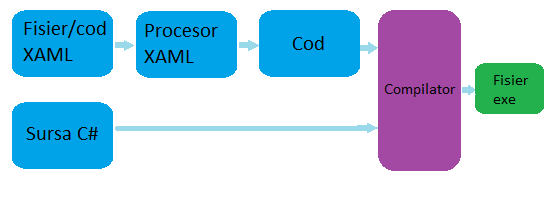
Relaţia dintre C# şi Java este mai complicată. Java derivă la rândul său din C şi C++. Ca şi Java, C# a fost proiectat pentru a produce cod portabil. Limbajul C# nu derivă din Java.

Între C# şi Java există o relaţie similară celei dintre “veri”, ele derivă din acelaşi strămoş, dar se deosebesc prin multe caracteristici importante.

Limbajul C# conţine mai multe facilităţi novatoare, dintre care cele mai importante se referă la suportul încorporat pentru componente software. C# dispune de facilităţi care implementează direct elementele care alcătuiesc componentele software, cum ar fi proprietăţile, metodele şi evenimentele. Poate cea mai importantă facilitate de care dispune C# este posibilitatea de a lucra într-un mediu cu limbaj mixt.

 XAML este un limbaj declarativ bazat pe XML, în sensul că definește CE și CUM doriți să faceți. Principalul avantaj al XAML este separarea modului în care arată programul și modul în care acesta se comportă. Principalul dezavantaj al WinForms era faptul că atât codul funcțional, cât și interfața vizuală, erau ambele create în C# sau VB.NET de către programator, și nu erau interschimbabile. În plus, era nevoie de mai mult efort din partea programatorului, pentru a crea atât funcționalitatea aplicației, cât și designul acesteia, pe care mulți coderi îl displăceau, deoarece nu erau designeri.

Odată cu sosirea XAML, această problemă a fost în cele din urmă rezolvată. XAML decupla interfața vizuală de codul din fundal, făcându-le două procese separate. Acest lucru însemna că un designer calificat se putea ocupa de proiectarea GUI în același timp în care un coder dezvolta funcționalitatea, și că cele două componente vor interacționa între ele. O altă consecință benefică a acestui lucru a fost posibilitatea creării de diferite GUI-uri pentru același cod funcțional.



Figură 1 – Principiu funcționare XAML

Exemplu cod XAML preluat din aplicație:



# Capitolul II Arhitectura aplicației

Aplicaţia este structurată în patru componente:

1. Hai să aflăm ce e cu vremea;
2. O problemă şi soluţia ei;
3. La scară mare – despre noi şi climă;
4. Vremea lui ”Cum să fiu bine”.

SECŢIUNEA „HAI SĂ AFLĂM CE E CU VREMEA”

Aplicația debutează cu o secțiune care abordează într-un mod aparte relația dintre om și

vreme. Astfel, utilizatorul este întâmpinat de o interfață personalizată, iar pe întregul parcurs al celor 5 pagini care compun secţiunea, informația este prezentată în raport cu opțiunile acestuia.

SECŢIUNEA „O PROBLEMĂ ȘI SOLUȚIA EI”

În această secţiune, conținutul, structurat în patru pagini, este unul interactiv și intuitiv deopotrivă, permițând utilizatorului să-și însușească cunoștințele fără dificultate, printr-o abordare axată pe punerea acestuia față în față cu problema în cauză.

Problematizarea informației îndrumă utilizatorul pas cu pas către soluțiile pe care acesta le are la îndemână pentru a contracara efectele negative ale schimbărilor climatice: practicarea sporturilor, alimentația sănătoasă etc.

SECŢIUNEA „LA SCARĂ MARE – DESPRE NOI ŞI CLIMĂ”

Cea de-a treia secțiune propusă este structurată în 5 pagini, fiecare pagină conținând câte o sinteză atractivă (infografic) cu privire la vreme și influențele acesteia asupra mediului înconjurător. Vremea declanșează reacții în lanț care afectează ulterior și oamenii în moduri adesea neașteptate.

Astfel, se evidențiază vulnerabilitățile societății umane cotidiene în raport cu schimbările climatice constante și din ce în ce mai relevante în ziua de azi: creșterea temperaturii, fenomenele meteorologice extreme, poluarea aerului și răspândirea bolilor.

SECŢIUNEA „VREMEA LUI ”CUM SĂ FIU BINE”

O altă componentă a aplicaţiei – structurată într-o singură pagină – este reprezentată de ultima sa secțiune, care oferă o serie variată de sugestii adolescentului vis-a-vis de reflectarea pe care o poate face asupra sănătății sale, serie denumită sugestiv “10 practici pentru a mă stresa mai puțin”.

Aceste indicații au ca scop îmbunătățirea stării de spirit cotidiene a adolescentului în moduri simple, care nu necesită un efort prea mare, dar care reușesc să îi echilibreze dispoziția și să îl binedispună.

La click pe butonul „START” se lansează aplicaţia, lăsându-se la latitudinea utilizatorului să îşi aleagă ritmul şi ordinea în care va studia informaţiile prezentate în fiecare dintre componentele aplicaţiei.



Figură 2 – Pagină START

Astfel, după lansarea aplicaţiei, odată cu afişarea paginii iniţiale, utilizatorul are la dispoziţie un meniu.

Secţiunile aplicaţiei pot fi accesate dând click pe unul dintre butoanele meniului din partea stângă a ecranului.



Figură 3 – Meniu aplicație

## II.1. Paginile proiectului

Secțiune1\_Pagină1

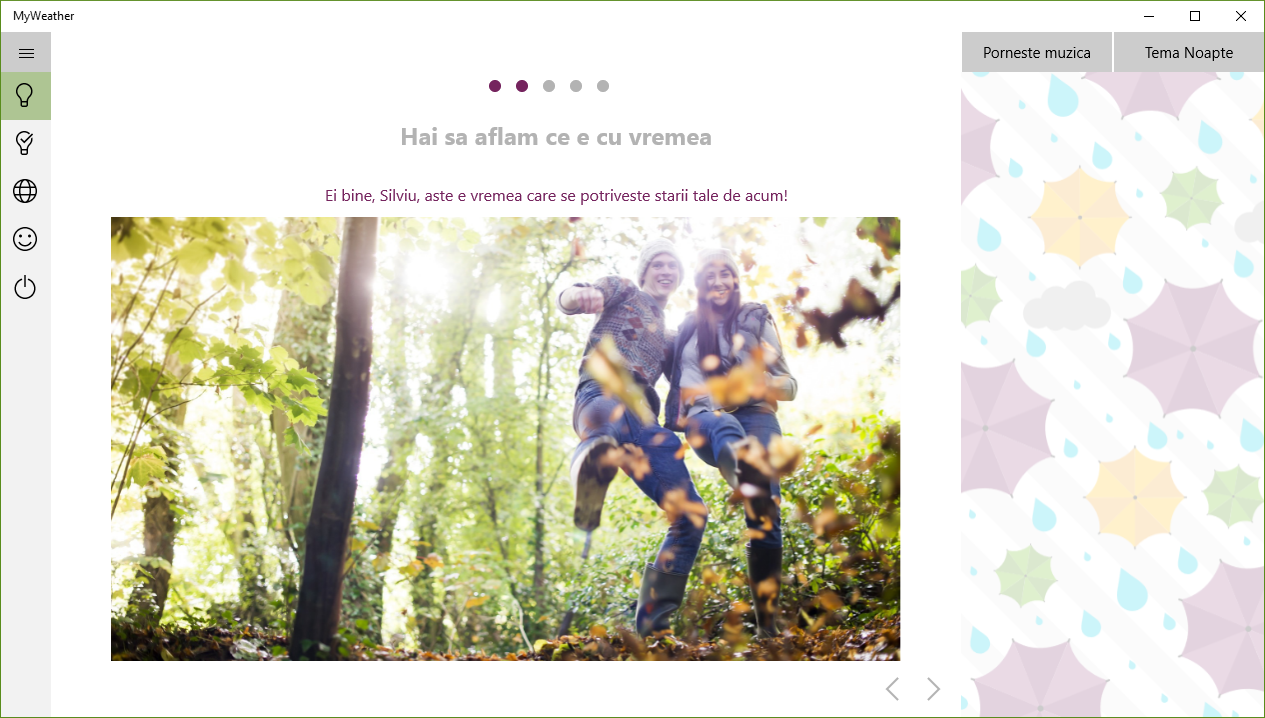
Utilizatorul completează un câmp cu numele său și selectează dintr-o listă numerică valoarea corespunzătoare stării sale de spirit.



Figură 4 – Pagină întâmpinare

Secțiune1\_Pagină2

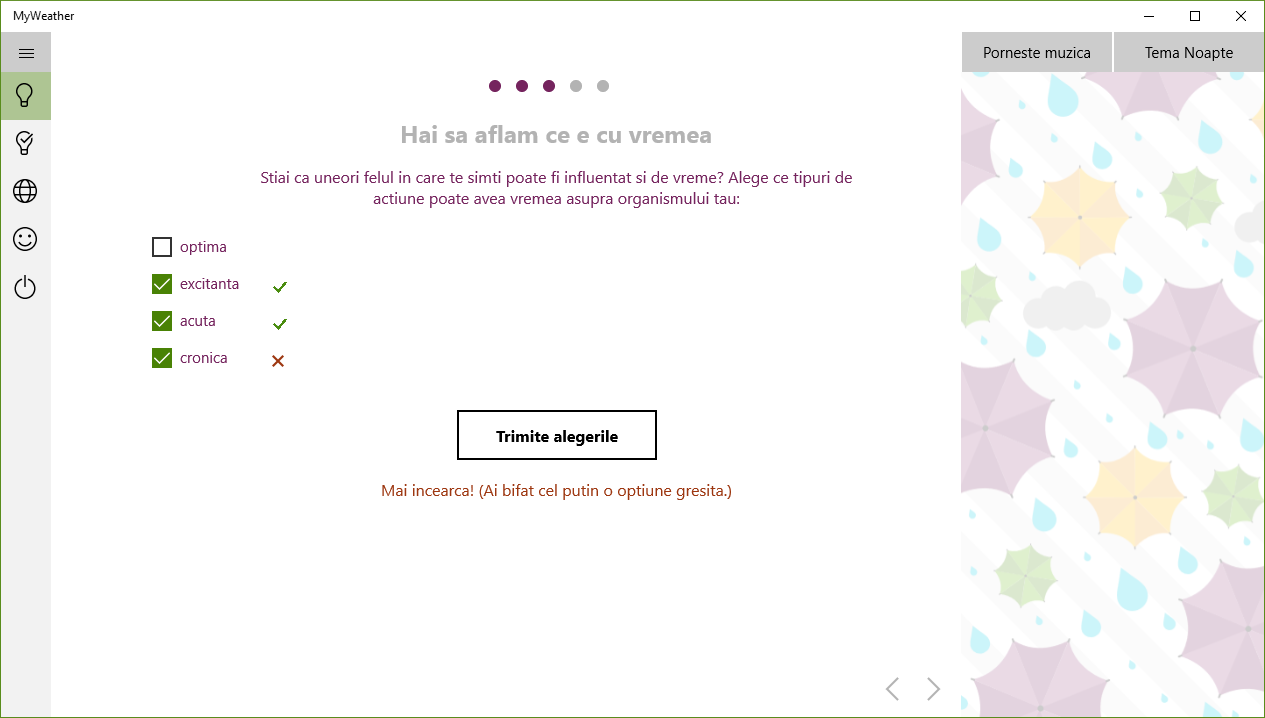
Utilizatorului i se prezintă o imagine sugestivă, potrivit alegerii făcute în pagina anterioară, în modul de adresare fiind folosit numele introdus anterior.



Figură 5 – Pagină feedback imagine

Secțiune1\_Pagină3

Utilizatorul trebuie să rezolve un exercițiu de tip checkboxes (alegere multiplă), primind feedback la apăsarea butonului “Trimite alegerile”.



Figură 6 – Pagină interacţiune „checkboxes”

Secțiune1\_Pagină5

Utilizatorul trebuie să rezolve un exercițiu de tip radio buttons (alegere singulară), primind feedback la apăsarea butonului “Trimite alegerile”.



Figură 7 – Pagină interacţiune „radio buttons”

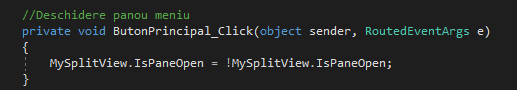
Secțiune2\_Pagină1

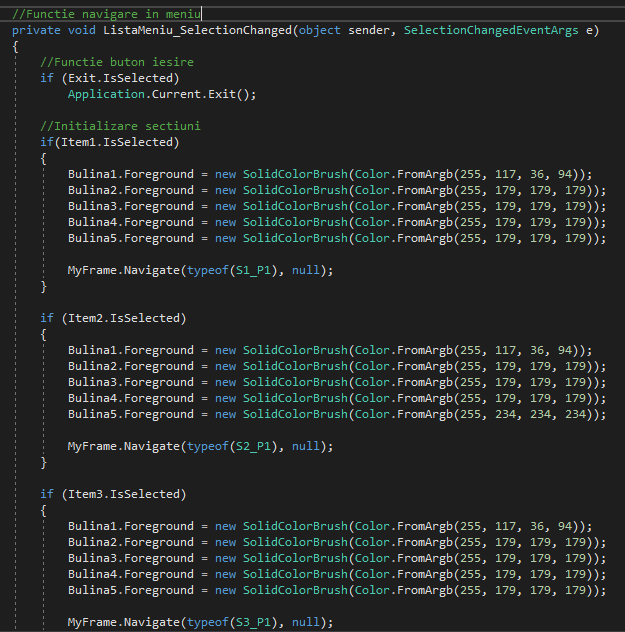
Pagina constă într-o interactivitate de tip expozitiv, la apăsarea pe una dintre cele trei buline fiind afișat câte un text asociat.

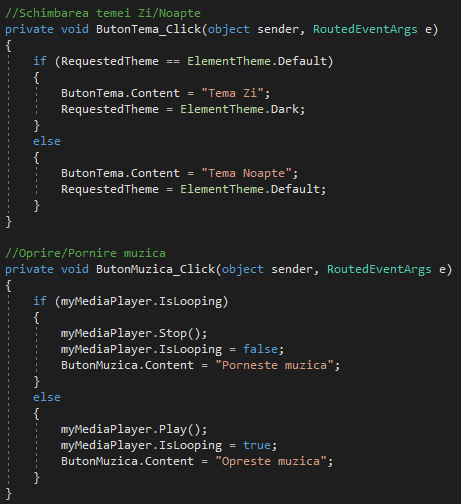


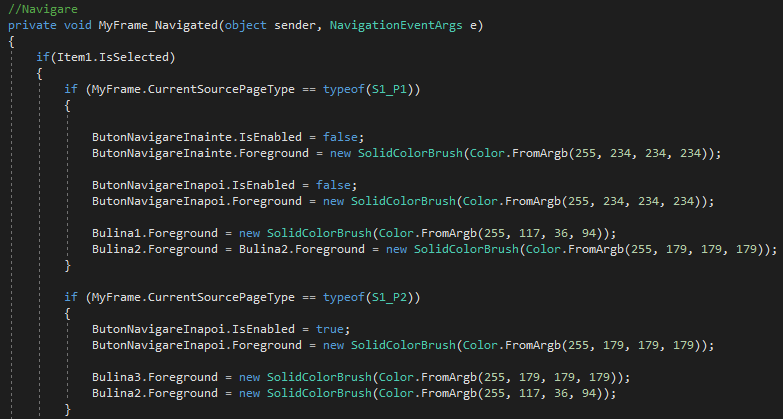
Figură 8 – Pagină interacţiune „text expunere”

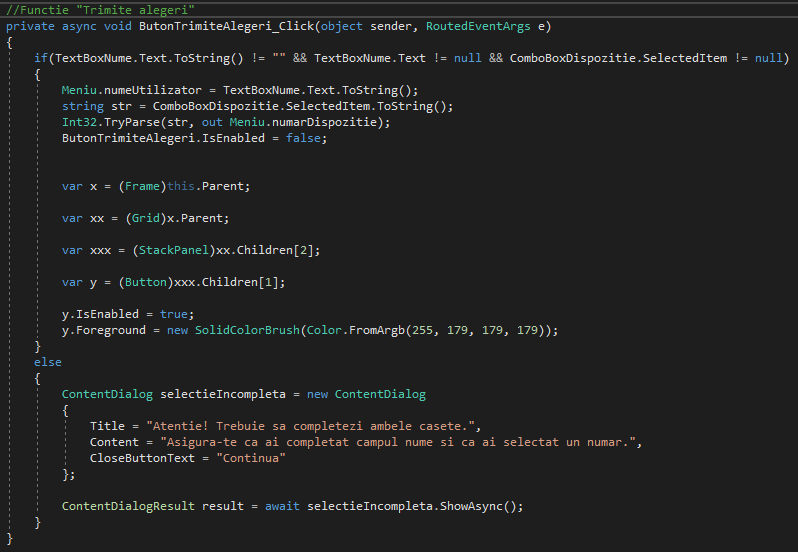
## II.2. Metodele aplicației

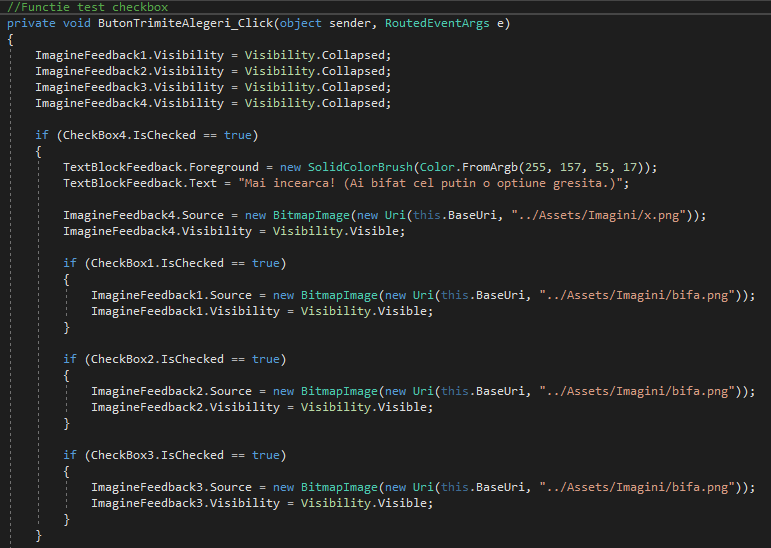


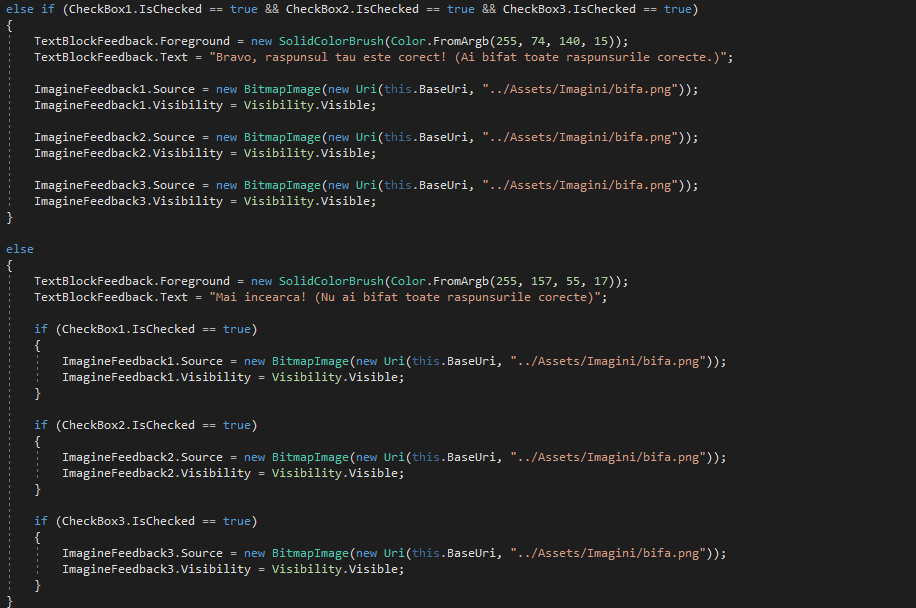
******

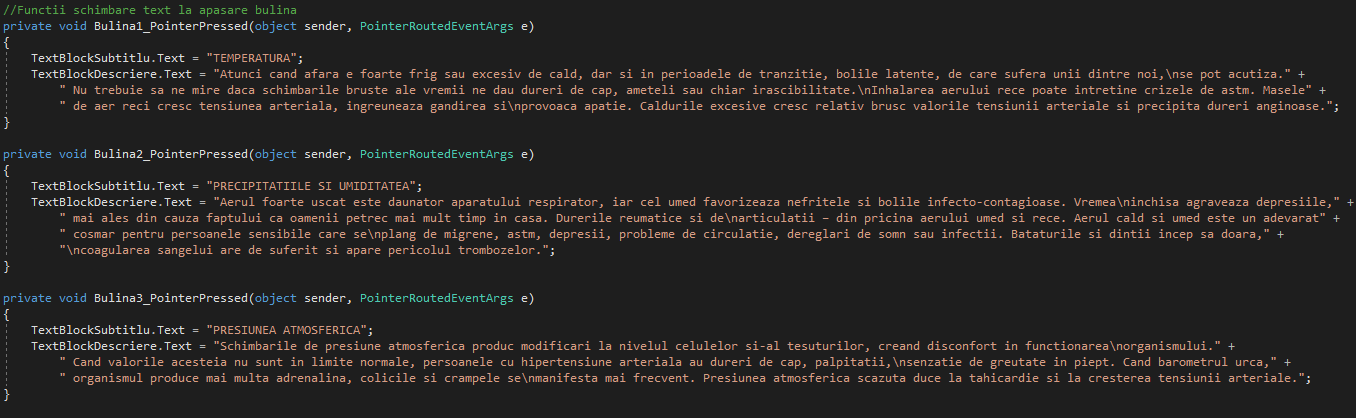












## II.3. Ghid de utilizare a aplicației

CONFIGURAŢIE HARDWARE

Configuraţia hardware minimă necesară pentru ca produsul realizat să ruleze în condiţii optime este următoarea:

* frecvenţa procesorului: 1,5 GHz
* memorie RAM: 512 MB
* rezoluţie ecran: 1024 x 768 pixeli
* sistem de operare: Windows 10

APLICAŢII UTILIZATE

Proiectul de faţă a fost dezvoltat în mediul de programare Microsoft Visual Studio Community şi implementat cu ajutorul limbajului C#. Pentru designul aplicaţiei, s-a folosit un limbaj de tip markup numit XAML, limbaj specific aplicaţiilor de tipul „Universal Windows Application”. Editarea imaginilor s-a realizat cu ajutorul aplicaţiilor Microsoft Paint şi Microsoft Office Picture Manager.

ACCESARE A APLICAŢIEI

Pentru a putea accesa aplicaţia pe calculatorul dumneavoastră, este necesar un CD cu versiunea demo a proiectului.

BUTOANE DE NAVIGARE SECVENŢIALĂ

Amplasate centrat (butonul „Start”) în ecranul iniţial, în meniul din partea stângă şi în partea dreaptă jos a celorlalte ecrane, cu următoarele funcţii:

* lansarea aplicaţiei ;
* încărcarea paginilor anterioară şi ulterioară ale paginii curente ;
* închiderea aplicaţiei .

FUNCŢIONALITĂŢI SUPLIMENTARE

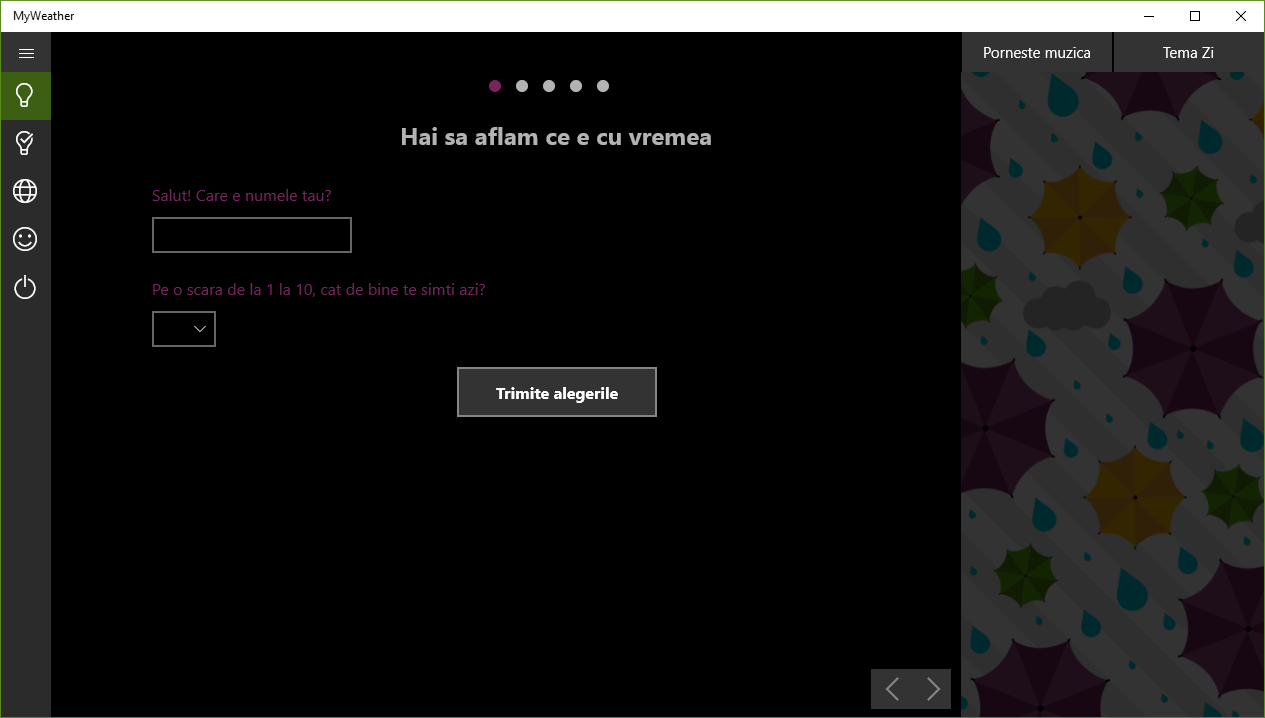
Aplicaţia pune la dispoziţia utilizatorului 2 funcţionalităţi adiţionale: „Porneşte muzica/Opreşte muzica” şi „Temă noapte/Temă zi”.

Astfel, utilizatorul va putea asculta sau nu muzică în fundal, pe parcursul navigării în aplicaţie.



Figură 9 – Funcţie „Porneşte muzica/Opreşte muzica”

Şi va reduce intensitatea luminoasă a ecranului, pentru a nu obosi ochii pe timpul nopţii.



Figură 10 – Funcţie „Temă Noapte/Temă Zi”

# Concluzii

Consider că, prin realizarea acestui proiect pentru primirea atestatului, am avut prilejul de a-mi aprofunda cunoștiințele în domeniul programării obiectuale.

De asemenea, am explorat o abordare diferită, alegând să creez un proiect de tipul „Universal Windows Application”, ce aduce aplicației un aspect modern și aerisit, dar și posibiliatea portării acesteia pe telefoanele ce au Windows-ul ca sistem de operare. Astfel, am reușit să mă dezvolt ca viitor programator, învățând cum să proiectez aspectul aplicației prin folosirea unui limbaj nou și intuitiv, ce aduce mai multă flexibilitate în proiectare, și anume XAML.

# Bibilografie

1. Herbert Schildt, C#: A Beginner’s Guide, 2001
2. Herbert Schildt, C#, Ed.Teora (traducere), 2002.
3. Karli Watson et al., Beginning Visual C#, Wrox Press Ltd., 2002.
4. Philip Syme, Peter Aitken, SAMS Teach Yourself the C# Web Programming in 21 Days, 2002.
5. Ion Smeureanu, Marian Dardala, Adriana Reveiu, Visual C# .NET, Bucureşti, 2004.
6. Bradley L. Jones, SAMS Teach Yourself the C# Language in 21 Days, 2004.
7. Karli Watson, Beginning C# 2005 Databases, Wiley Publishing, Inc., 2006.
8. Kris Jamsa, Lars Klander, Totul despre C şi C++ Manualul fundamental de

programare în C şi C++, Ed. Teora (traducere), 2007.

1. <http://www.terapeutic.ro/informatii-sanatate/boli-influentate-de-vreme/>
2. <https://www.apha.org/news-and-media/multimedia/infographics/how-climate-change-affects-your-health>