**ÎNVĂŢĂM SĂ CALCULĂM**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Bogdan – Petru MATEŞ** |
|  |  |  |
|  | Coordonator ştiinţific: | **Conf. dr. Ing Emil Ştefan CHIFU** |

**2015**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DECAN, |  | | DIRECTOR DEPARTAMENT, | |
| **Prof. dr. ing. Liviu MICLEA** |  | **Prof. dr. ing. Rodica POTOLEA** | |

Absolvent: **Bogdan – Petru MATEŞ**

**ÎNVĂŢĂM SĂ CALCULĂM**

1. **Enunţul temei:**S-a urmărit realizarea unui produs software educaţional pentru învăţarea matematicii de către elevii claselor primare, într-un mod prietenos şi atractiv, folosind Windows Communication Foundation şi alte tehnologii .Net.
2. **Conţinutul lucrării:**Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză şi fundamentare teoretică, Proiectare de detaliu şi implementare, Testare şi validare, Manual de instalare şi utilizare, Concluzii, Bibliografie.
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanţi**:
5. **Data emiterii temei:** 1 noiembrie 2014
6. **Data predării:** 18 Iunie 2015

|  |  |
| --- | --- |
| Absolvent: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Coordonator ştiinţific: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul(a)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**, legitimat(ă) cu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ seria \_\_\_\_\_\_\_ nr. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, autorul lucrării \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ a anului universitar \_\_\_\_\_\_\_\_\_\_, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

|  |  |  |
| --- | --- | --- |
| Data  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Nume, Prenume  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
|  |  | Semnătura |

Cuprins

[Capitolul 1. Introducere – Contextul proiectului 1](#_Toc422050182)

[1.1. Contextul proiectului 1](#_Toc422050183)

[1.2. Motivaţia 1](#_Toc422050184)

[1.3. De ce s-a utilizat framework-ul .NET [1]? 2](#_Toc422050185)

[1.4. Structura documentului 2](#_Toc422050186)

[Capitolul 2. Obiectivele Proiectului 3](#_Toc422050187)

[Capitolul 3. Studiu Bibliografic 4](#_Toc422050188)

[3.1. Tehnologii back-end 4](#_Toc422050189)

[3.1.1. Microsoft SQL Server 4](#_Toc422050190)

[3.1.2. Servicii WCF 5](#_Toc422050191)

[3.1.3. LINQ 7](#_Toc422050192)

[3.2. Tehnologii front-end 9](#_Toc422050193)

[3.2.1. .NET 4.0 9](#_Toc422050194)

[3.2.2. Windows Form 9](#_Toc422050195)

[3.2.3. WPF 10](#_Toc422050196)

[Capitolul 4. Analiză şi Fundamentare Teoretică 11](#_Toc422050197)

[4.1. Arhitectura Client-Server 11](#_Toc422050198)

[4.2. Arhitectura Model View ViewModel (MVVM) 12](#_Toc422050199)

[4.3. Modelul Single Page Application 13](#_Toc422050200)

[4.4. Design-ul Arhitectural Singleton 13](#_Toc422050201)

[4.5. Design-ul bazei de date 15](#_Toc422050202)

[4.5.1. Entity Framework 15](#_Toc422050203)

[4.6. Iniţializare lazy 17](#_Toc422050204)

[4.7. Încărcare lazy 17](#_Toc422050205)

[4.8. Cuplare slabă 17](#_Toc422050206)

[4.9. Coeziune mare 18](#_Toc422050207)

[4.10. Considerente de securitate 19](#_Toc422050208)

[4.11. Protocoale utilizate 20](#_Toc422050209)

[Capitolul 5. Proiectare de Detaliu si Implementare 22](#_Toc422050210)

[5.1. Baza de date 22](#_Toc422050211)

[5.2. Diagrama de componente ale aplicaţiei 25](#_Toc422050212)

[5.3. Configurarea serviciului WCF 26](#_Toc422050213)

[5.4. Diagrama de clase 26](#_Toc422050214)

[5.4.1. Diagrama de clase a componentei Server 26](#_Toc422050215)

[5.4.2. Diagrama de clase a componentei Client 27](#_Toc422050216)

[5.5. Diagrama cazurilor de utilizare 29](#_Toc422050217)

[5.6. Diagrama de secvenţă pentru operaţia de Login 31](#_Toc422050218)

[5.7. Proiectarea interfeţei cu utilizatorul 32](#_Toc422050219)

[5.8. Generarea dinamică a termenilor din cadrul testelor şi jocurilor 32](#_Toc422050220)

[5.9. Comunicarea în cadrul aplicaţiei 33](#_Toc422050221)

[5.10. Implementarea şablonului Singleton 33](#_Toc422050222)

[5.11. Motivarea utilizării şablonului Model – View – ViewModel (MVVM) 33](#_Toc422050223)

[5.12. Implementarea modelului Single Page Application (SPA) 34](#_Toc422050224)

[5.13. Utilizarea delegaţilor 34](#_Toc422050225)

[Capitolul 6. Testare şi Validare 35](#_Toc422050226)

[6.1. Validarea datelor de intrare 35](#_Toc422050227)

[Capitolul 7. Manual de Instalare si Utilizare 39](#_Toc422050228)

[7.1. Ghid de instalare 39](#_Toc422050229)

[7.1.1. Resurse hardware necesare pentru utilizarea aplicaţiei 39](#_Toc422050230)

[7.1.2. Resurse software necesare pentru utilizarea aplicaţiei 39](#_Toc422050231)

[7.1.3. Manual de instalare al aplicaţiei 39](#_Toc422050232)

[7.2. Manualul de utilizare al aplicaţiei 39](#_Toc422050233)

[7.2.1. Înregistrarea unui utilizator nou 39](#_Toc422050234)

[7.2.2. Autentificarea unui utilizator 41](#_Toc422050235)

[7.2.3. Utilizarea aplicaţiei în calitate de utilizator tip „Guest” 41](#_Toc422050236)

[7.2.4. Pornirea şi oprirea melodiilor aplicaţiei 42](#_Toc422050237)

[7.2.5. Selectarea unei operaţii 43](#_Toc422050238)

[7.2.6. Selectare test nou, în funcţie de operaţia selectată anterior 45](#_Toc422050239)

[7.2.7. Selectarea unei ecuaţii, în funcţie de operaţia selectată anterior 47](#_Toc422050240)

[7.2.8. Selectarea unui joc tip „Puzzle” 48](#_Toc422050241)

[7.2.9. Jocul „Puzzle” 53](#_Toc422050242)

[7.2.10. Selectarea unui joc tip „Spânzurătoare” 55](#_Toc422050243)

[7.2.11. Jocul tip „Spânzurătoare” 57](#_Toc422050244)

[7.2.12. Selectarea unui joc tip „Blocks” 59](#_Toc422050245)

[7.2.13. Jocul tip „Blocks” 60](#_Toc422050246)

[7.2.14. Vizualizarea topului rezultatelor 62](#_Toc422050247)

[7.2.15. Vizualizarea premiilor obţinute de utilizatorul curent al aplicaţiei 63](#_Toc422050248)

[7.2.16. Administrarea elevilor de către utilizatorii de tip „Profesor” 65](#_Toc422050249)

[7.2.17. Logout utilizator 67](#_Toc422050250)

[7.2.18. Părăsirea aplicaţie 67](#_Toc422050251)

[Capitolul 8. Concluzii 68](#_Toc422050252)

[8.1. Realizari 68](#_Toc422050253)

[8.2. Concluzii 68](#_Toc422050254)

[8.3. Dezvoltari ulterioare 69](#_Toc422050255)

[Bibliografie 70](#_Toc422050256)

# Introducere – Contextul proiectului

## Contextul proiectului

Întrucât se doreşte informatizarea sistemului didactic, alături de afirmaţia „repetiţia este mama succesului”, am decis asupra implementării unei aplicatii de învăţare cu bază de date distribuită, in domeniul matematicii.

Aplicaţia este destinată elevilor din clasele primare, având scopul de a îmbina scopul de a stăpâni operaţiile primare cu faptul că un mediu de invăţare prietenos va captiva utilizatorii şi îi va indemna să revină la utilizarea aplicaţiei pe cât de des posibil.

Totodată, integrarea unor teste şi jocuri cunoscute, transpuse în teme matematice, va oferi cadrelor didactice posibilitatea de a evalua, urmări şi trage concluzii de pe urma progreselor făcute de utilizatorii aplicaţiei, funcţionalitate oferită de includerea unui clasament în structura aplicaţiei.

Exerciţiile din cadrul fiecărei operaţii oferă aplicaţii variate prin care elevii au posibilitatea să-şi fixeze şi să-şi aprofundeze cunoştinţele. Profesorul poate să evalueze continuu cunoştinţele, priceperile şi deprinderile elevilor precum şi să le trateze diferenţiat, în funcţie de capacităţile de înţelegere ale elevilor şi a obiectivelor urmărite; aceasta se realizează prin dispunerea graduală a sarcinilor didactice.

Situaţiile de autoevaluare permit realizarea feedback-ului, în special prin testele propuse şi jocurile puzzle.

## Motivaţia

În decursul studiilor gimnaziale, profesorul de engleză ne-a pus la dispoziţie si ne-a încurajat sa utilizăm o platformă asemanătoare. Acest proces a avut rezultate de-a dreptul încurajatoare pentru generaţiile ulterioare, întrucât s-a dovedit captivant si interesant pentru utilizatori.

Tendinţa mediului informatic de a reţine utilizatorii de la activităţile propuse poate fi stopată sau măcar redusă prin dezvoltarea unei aplicaţii care, prin utilizare, aduce utilizatorilor un aport de cunostinţe, poate chiar necesare în viaţa cotidiană sau în procesul de invăţare.

În prezent, trei din cele mai utilizate platforme de învăţare sunt Duoling, Polyglot si EduComm. Acestea se bucură de un real succes, atât în rândul elevilor din clasele primare sau din ciclul gimnazial, cât şi din partea persoanelor de vârstă medie care doresc să îşi îmbunătăţească cunoştinţele în unul din domeniile suportate de aceste platforme.

## De ce s-a utilizat framework-ul .NET?

Am decis asupra utilizării framework-ului .NET[5] întrucât majoritatea calculatoarelor din unităţile de invăţământ primar şi gimnazial utilizează versiuni ale sistemului de operare Windows.

Versiunea utilizată a fost 4.0 datorită faptului că o mare parte din unităţile de invăţământ utilizează în continuare versiunea XP a sistemului de operare Windows.

Figura 1. .NET Framework 4.0

Printre motivele alegerii acestui framework se numără şi faptul că recent am început sa lucrez cu acesta şi am considerat utilizarea sa o motivare în procesul de aprofundare a cunoştinţelor în această direcţie.

## Structura documentului

Acest document va prezenta abordările de implementare propuse în rezolvarea temei proiectului.

Vom începe cu prezentarea cerinţelor şi obiectivelor propuse în cadrul procesului de stabilire a temei, în cadrul capitolului 2.

În capitolul 3 vom continua cu prezentarea studiilor bibliografice, a tehnologiilor utilizate şi a fundamentelor de bază ale acestora. Capitolul a fost structurat în două părţi / subcapitole: tehnologii front-end şi tehnologii back-end.

În cadrul capitolelor 4 şi 5 vom prezenta abordările alese, şabloanele şi protocoalele utilizate şi modul de implementare a acestora. De asemenea, am inclus diagramele reprezentative ale aplicaţiei (diagrama de clase, diagrama de componente, diagrama bazei de date, diagrama de secvenţă, diagrama cazurilor de utilizare) şi prezentarea în detaliu a implementării anumitor aspecte / şabloane utilizate în implementarea aplicaţiei.

Capitolul 6 furnizează informaţii legate de validarea datelor introduse de utilizatori, mesajele de eroare afişate utilizatorilor şi modurile de testare a aplicaţiei.

În cadrul capitolului 7 am inclus un manual de instalare şi de utilizarea a aplicaţiei, incluzând capturi ale ecranului reprezentative. În ghidul de instalare vor fi incluşi paşii necesari instalării şi configurării aplicaţiei, iar în cadrul ghidului de utilizare al aplicaţiei am inclus paşi şi capturi ale ecranului ce oferă informaţii suficiente pentru o utilizare facilă a aplicaţiei. Întrucât aplicaţia este destinată utilizării de către persoane cu vârste cuprinse între 6 şi 12 ani, am pus accent sporit pe acest capitol al documentului.

În ultimul capitol, capitolul 8 am prezentat concluziile trase după implementarea aplicaţiei şi posibilele dezvoltări ulterioare ale acestui proiect.

În cadrul bibliografiei am inclus sursele de documentare utilizate în cadrul procesului de proiectare, implementare şi testare a acestei aplicaţii.

# Obiectivele Proiectului

S-a urmărit realizarea unui instrument pedagogic, modern şi util în învăţarea matematicii, în conformitate cu cerinţele pedagogice actuale.

Prin conţinutul său, programul va permite elevilor din clasele I-VI:

* dobândirea unor competenţe formativ-informative;
* dobândirea unui grad mai mare de autonomie a elevilor;
* dezvoltarea capacităţilor de a efectua operaţii logice de analiză, sinteză, comparaţie, abstractizare si generalizare;
* explicarea pe bază de argumente şi efectuarea raţionamentelor inductive, deductive si analogice;
* gândire divergentă de a transfera cunoştintele în situaţii diferite şi de a se autoevalua;
* exercţiile relativ independente şi foarte variate servesc scopurilor propuse, ţinând cont de capacităţile elevilor;
* eliminarea timpilor morţi din activitatea didactică;
* elevii sunt îndrumati să facă demersuri cognitive, de investigaţie, fiind implicaţi direct în activitatea de “descoperire”.

Principalele obiective ale aplicaţiei sunt următoarele:

* dezvoltă competenţele cheie de “a învăţa să înveţi”;
* dezvoltă competenţa cognitivă şi pragmatică;
* dezvoltă spiritul participativ, de interactivitate şi intercomunicare;
* dezvoltă capacitatea de evaluare, autoevaluare şi spiritual de competiţie;
* stimulează “triunghiul de aur” părinte-copil-profesor;
* dezvoltă o gândire logică;
* elevul învaţă în ritm propriu, fără emoţii şi perturbări ale comportamentului determinate de factorii de mediu;
* aprecierea rezultatelor şi progreselor obţinute se face în mod obiectiv;

Programul va fi compus din trei secţiuni:

I. activităţi de învăţare – această secţiune va permite învăţarea fiecărei operaţii, va reactiva sistemul de cunoştinţe asimilat anterior; va antrena elevii în descoperirea de noi cunoştinţe, tehnici de lucru, procedee, proprietăţi şi noţiuni.

II. activităţi de aprofundare a cunoştinţelor – va sistematiza noţiunile şi procedeele intuite, va realiza o sinteză riguroasă şi accesibilă din cadru fiecărei operaţii, va oferi aplicaţii variate prin care elevul are posibilitatea să-şi fixeze, să aprofundeze, să dezvolte şi să aplice creator cunoştinţele dobândite.

III. evaluarea – colecţii de teste şi jocuri care permit:

* profesorului să evalueze continuu cunoştinţele, priceperile şi deprinderile elevilor;
* elevilor să se autoevalueze şi corecteze dezvoltându-le astfel gândirea critică;

Toate aceste activitaţi se vor înregistra într-o bază de date.

Aplicaţia va avea două tipuri de utilizatori: profesori şi elevi.

1. Utilizatorii de tip „profesor” se vor înregistra în aplicaţie, îşi vor administra baza de date cu elevi asignaţi lor, vor urmări şi reiniţializa rezultatele obţinute de elevii asignaţi lor la testele si jocurile disponibile acestora.

De asemenea, acest tip de utilizator va avea posibilitatea de a explora aplicaţia, având astfel posibilitatea de a se ţine la curent cu gradul de dificultate al testelor şi cu conţinutul şi modul de expunere către elevi al materiei ce stă la baza componentelor de evaluare.

1. Utilizatorii de tip „elev” se vor înregistra în aplicaţie, îşi vor selecta dintr-o listă de profesori cel căruia vor fi asignaţi în procesul de urmărire a rezultatelor testelor, iar ulterior se vor putea autentifica şi utiliza aplicaţia.

Acest tip de utilizator va avea acces la un meniu în care va găsi lecţii de prezentare / predare a materiei, diferite ecuaţii pentru a înţelege mai bine rolul fiecărui termen al operaţiilor, jocuri cu teme matematice si teste / evaluări.

Ambelor tipuri de utilizatori le vor fi disponibile topuri ale rezultatelor obţinute de utilizatorii de tip „elev”.

Profesorii vor avea posibilitatea de a şterge elevii asignaţi lor în mod eronat şi vor putea reseta rezultatele obţinute de elevi în cadrul testelor şi jocurilor.

# Studiu Bibliografic

## Tehnologii back-end

### Microsoft SQL Server

SQL Server[2] este un sistem pentru gestiunea bazelor de date (DBMS – Data Base Management System) produs de Microsoft. Fiind un sistem relaţional de baze de date client-server, procesările de date ale programului nu au loc pe un singur calculator. Pe calculatorul server se crează partea din program legată de baza de date, iar cealaltă parte, ce conţine interfaţa client, se va găsi pe un calculator client. Componentele sistemului client-server comunică in reţea ca şi cum ar face parte din acelaşi program. Un sistem relaţional de baze de date este un tip de arhitectură de baze de date care este împărţită în mai multe tabele de date, care la rândul lor sunt împărţite în linii (înregistrări) şi coloane (câmpuri).

Figura 3. Microsoft SQL Server

Structured Query Language (SQL) este utilizat în comunicarea cu baza de date. Potrivit ANSI (American National Standards Institute), SQL este limbajul standard pentru sistemele de management al bazelor de date relaţionale. Propoziţiile SQL sunt utilizate pentru a efectual actualizări ale datelor sau retrageri de date dintr-o bază de date. Câteva dintre sistemele de management al bazelor de date relaţionale pe care SQL le utilizează sunt: Oracle, Microsoft SQL Server, Access, Ingres, Sybase. Deşi aproape toate sistemele de baze de date folosesc SQL, unele dintre ele au propriile extensii adiţionale care se folosesc de obicei doar în sistemul lor. Cu toate acestea, comenzile SQL standard ca: Select, Insert, Update, Delete, Create şi Drop sunt suficiente pentru a lucra cu baza de date.

Putem spune despre Microsoft SQL Server[2] că este o soluţie integrată de management şi analiză a datelor, care ajută organizaţiile de orice dimensiune să:

* Dezvolte, implementeze şi administreze aplicaţii la nivel de întreprindere mai sigure, scalabile şi fiabile
* Maximizeze productivitatea IT prin reducerea complexităţii creării, implementării şi administrării aplicaţiilor pentru baze de date
* Partajeze date pe mai multe platforme, aplicaţii şi dispozitive pentru a facilita conectarea sistemelor interne şi externe
* Controleze costurile fără a sacrifica performanţa, disponibilitatea, scalabilitatea sau securitatea

Gestionarea serverului se face foarte uşor prin aplicaţia SQL Server Management Studio.

### Servicii WCF

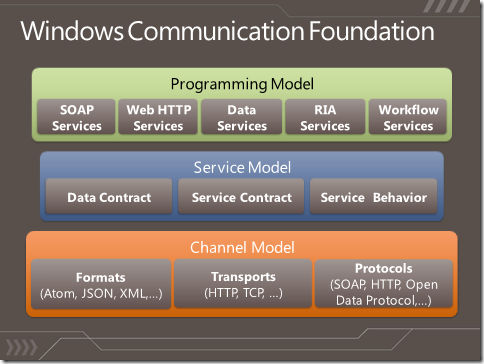
Windows Communication Foundation (WCF) [3] este un kit de dezvoltare software utilizat pentru dezvoltarea şi implementarea unor servicii. WCF[3] oferă un mediu de execuţie pentru serviciile noastre, fiind un Common Language Runtime (CLR) pentru servicii care nu permite sa utilizăm atât serviciile noastre ca servicii Windows, cât si alte servicii Windows, oferind astfel interoperabilitate între servicii.

Figura 3. Windows Communication Foundation overview diagram

Clienţii si serviciile interacţionează prin transmiterea si primirea de mesaje. Mesajele pot fi transferate fie direct de la client la serviciu, fie printr-un intermediar. WCF permite clientului sa comunice cu un serviciu în toate limitele de execuţie. Pe aceeaşi maşină, clientul poate consuma servicii în acelaşi domeniu sau în domenii diferite în cadrul aceluiaşi proces, sau între procese diferite. Fiecare serviciu este asociat cu o adresă unică. Adresa oferă două elemente importante: locul de serviciu şi protocolul de transport, utilizate in cadrul comunicării cu serviciul. WCF suportă următoarele protocoale de transport: HTTP, TCP, Peer Network, Inter-Process Communication (IPC) si MSMQ.

Comportarea unui serviciu afectează operaţia la runtime şi este definită de o serie de clase din .NET, invocate când se lansează WCF atât pe partea de server, cât şi pe partea de client.

Există trei tipuri primare pentru comportări:

* Service behaviors: rulează la nivel de serviciu şi au acces la toate endpoint-urie
* Endpoint behaviors: sunt pentru endpoint-urile serviciului
* Operation behaviors: acţionează la nivelul operaţiei şi sunt folosite în special pentru manipularea serializării, fluxului tranzactiilor şi gestionarea parametrilor pentru o operaţie a serviciului

Alături de aceste trei comportări, WCF defineşte comportare pentru callback, ce gestionează endpoint-urile create pe client pentru comunicarea bidirecţională.

În WCF, clienţii comunică cu serverul folosind protocolul SOAP. Toate comunicaţiile cu serviciile Windows Communication Foundation (WCF) se realizează prin intermediul endpoint-urilor. Endpoint-urile oferă clienţilor acces la funcţionalităţile oferite de serviciul WCF.

Fiecare endpoint conţine are în componenţă 4 proprietăţi:

* O adresă care indică unde poate fi găsit endpoint-ul
* Un binding care specifică cum poate comunica clientul cu endpoint-ul
* Un contract care identifică operaţiile disponibile
* Un set de comportamente care specifică detaliile de implementare locale ale endpoint-ului

Dintre aceste patru proprietăţi, primele trei sunt cele mai importante: adresa, binding-ul şi contractul.

Adresa identifică unic endpoint-ul şi indică potenţialilor consumatori ai serviciului unde îl pot găsi. Este reprezentată în modelul obiectului WCF de către clasa EndpointAddress.

Binding-ul specifică cum comunică cu endpoint-ul. Aceasta include:

* Protocolul de transport utilizat (de exemplu: TCP sau HTTP)
* Encoding-ul utilizat pentru mesaje (de exemplu: text sau binar)
* Cerinţele de securitate necesare (de exemplu: SSL sau SOAP Message Security)

Contractul repezintă funcţionalităţile expuse de endpoint către client. Un contract specifică:

* Ce operaţii pot fi apelate de către client
* Forma mesajului
* Tipul parametrilor de intrare sau datele necesare pentru a apela operaţia
* Ce tip de mesaj de procesare sau de răspuns este aşteptat de client

### LINQ

Language INtegrated Query[4] permite interograrea unor seturi de obiecte prin invocarea unor metode care “oglindesc” funcţionalitatea SQL.

În .Net Framework 3.5 este introdus LINQ[4], un mecanism pentru manipularea colecţiilor de obiecte (selectare, ordonare, filtrare, agregarea pentru afişarea, modificarea şi durabilitatea datelor).

LINQ[4] este o componentă a platformei Microsoft .NET care adaugă suport pentru interogări de date direct din sintaxa limbajelor .NET. Această componentă defineşte un set de operatori de interogare care pot fi folosiţi pentru chestionarea, proiectarea şi filtrarea datelor din vectori, clase numărabile, fisiere XML, baze de date relaţionale şi alte surse de date, cu condiţia ca informaţia să fie încapsulată în obiecte. Aşadar, daca sursa de date nu stocează datele sub formă de obiecte, este necesar să existe o corelare între aceasta şi modelul obiectual. Interogările scrise folosind operatorii de interogare sunt fie executate de motorul de procesare LINQ, fie prin intermediul unor extensii, transmise LINQ provider-ilor care fie implementează un motor de interogare separat, fie traduc interogarea într-un format diferit pentru a fi executat pe un suport de date diferit (cum ar fi un server de date SQL). Rezultatul unei interogări este o colecţie de obiecte stocate în memoria programului care pot fi enumerate folosind o funcţie de parcurgere standard cum ar fi funcţia foreach din C#.

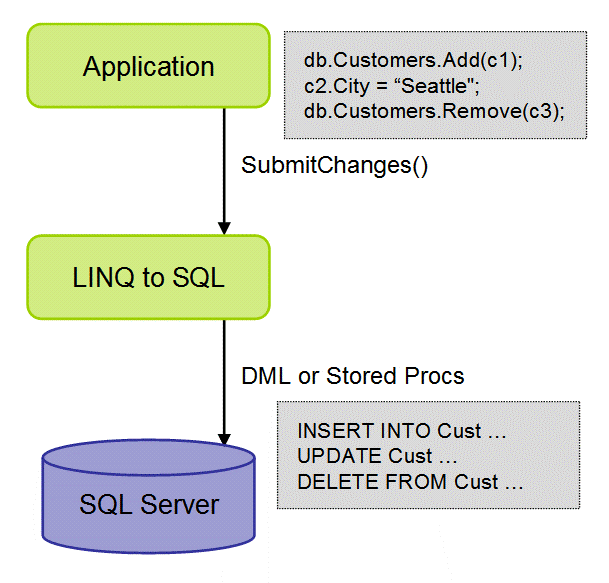
Pentu a folosi LINQ eficient în aplicaţii care stochează informaţii în baze de date SQL se foloseşte un provider special, LINQ to SQL. Această variantă de LINQ nu are nevoie de propriul motor de procesare a interogărilor ci traduce interogările în limbajul SQL şi le transmite serverului pentru procesare. În schimb, această variantă de LINQ are nevoie să stabilească o coreleţie între cele doua modele de date.

Figura 3. Exemplu LINQ to SQL

Obiectul DataContext este canalul prin care se obţin obiectele din baza de date şi prin care se transmit modificarile către baza de date. Se foloseste în acelaşi mod în care se foloseşte un obiect de tip conexiune în ADO.NET. De fapt, un obiect de tip DataContext se iniţializează cu un obiect de tip conexiune sau cu un şir de conectare. Scopul acestui obiect este de a traduce cererile de obiecte în interogari SQL adresate bazei de date şi de a crea obiectele din rezultatule interogărilor. Obiectul DataContext implementează operatorii de interogare specifici LINQ.

Specificaţii privind clasele auxiliare generate pentru LINQ to SQL, şi care corespund tabelelor din baza de date:

* Clasa este decorată cu atributul [Table];
* Câmpurile publice sau proprietăţile corespunzătoare coloanelor din tabele sunt decorate cu atributul [Column];
* Definirea relaţiilor între tabele cu atributul [Association];
* Suprascrierea comportamentelor de inserare, modificare şi ştergere se face prin definirea metodelor corespunzătoare şi decorarea acestora cu atributele [Insert] [Delete] şi [Update];
* Adaugarea la elementele corespunzătore procedurilor stocate, view-urilor şi funcţiilor se face adaugând atributele [StoredProcedure], [View] şi [Functions];
* Asigurarea declanşării evenimentelor PropertyChanging şi PropertyChanged.

Toţi operatorii de interogare sunt de fapt metode şi funcţii. O altă inovaţie este reprezentată de faptul că utilizarea punctului nu mai este neaparat necesară atunci când se foloseste sintagma „from” în interiorul limbajului de programare. De asemeni Lamda funcţiile şi un tip special de date, neprecizat, var, care este definit la runtime au făcut posibile interogările la nivelul nativ al limbajului de programare.

Dacă baza de date are relaţii definite între tabele, atunci ierarhia creată de acestea se reflectă în modelul obiectelor auxiliare cereate. Accesul la datele legate de o relaţie se face simplu, specificând tabela copil (ca proprietate a obiectului reprezentând tabela părinte. Această proprietate este o listă de obiecte copil), sau din tabela copil specificând tabela părinte (ca proprietate a obiectului reprezentând tabela copil).

Pentru a modifica o înregistrare din baza de date este suficient să încărcăm înregistrarea respectivă în memorie, să executăm modificările dorite şi apoi să apelăm metoda SubmitChanges. LINQ to SQL urmăreşte modificările şi generează automat cod SQL pentru a efectua modificările în baza de date. Acelaşi comportament este valabil şi pentru operaţiile de adăugare şi de ştergere a unor înregistrări din tabele. LINQ to SQL oferă posibilitatea de a suprascrie comportamentul în cazul acestor operaţii precum şi aceea de a urmări erorile, de a rectifica şi de a reîncerca efectuarea operaţiilor, prin intermediul tranzacţiilor. O altă problemă care poate fi urmărită şi controlată este modul în care sunt tratate problemele legate de concurenţă.

Adăugarea de înregistrări noi în baza de date este o simplă problemă de a crea instanţa unui obiect, adăugarea lui în colecţia potrivită şi apoi în apelarea metodei SubmitChanges pentru a înregistra noul obiect în baza de date. Pentru a adăuga înregistrări într-o tabelă care sunt legate de noua înregistrare, în condiţiile în care între cele două tabele este stabilită relaţia corespunzătoare, este suficient să se construiască obiectele, iar obiectul copil să se adauge în proprietatea reprezentând tabela copil din obiectul parinte şi apoi să se apeleze metoda SubmitChanges.

Apelul procedurilor stocate este cât se poate de simplu. Spre deosebire de ADO.NET –ul clasic, unde trebuiau construiţi parametri manual, conectarea la baza de date şi apoi apelul propriu-zis al procedurii, LINQ oferă posibiliatea creării de funcţii speciale care fac apelul pentru noi şi care returnează obiecte de tipuri specifice.

LINQ este un foarte bun ORM deoarece permite ca interogările asupra bazelor de date să beneficieze de verificarea erorilor de sintaxă, de autocomplete şi, nu în ultimul rând, generarea claselor auxiliare este foarte uşoară, datorită infrastructurii puse la dispoziţie. Pe de altă parte faptul că este foarte greu de folosit practic, cu alte servere de baze de date decât Microsoft SQL Server, dar mai ales pentru că platforma .NET are o nouă componentă ORM, numită Entity Framework face ca modelul LINQ să nu mai fie atât de folosit ca ORM ci mai degrabă ca limbaj pentru interogări, asa cum ii spune şi numele. Există şi o componentă care face legătura între Entity Framework şi LINQ, numită LINQ to EF.

## Tehnologii front-end

### .NET 4.0

.NET[5] este un framework de dezvoltare software care permite realizarea, rularea şi distribuirea atât a aplicaţiilor desktop Windows, cât şi a aplicaţiilor Web, constând dintr-o serie de tehnologii, precum ASP, XML, OOP, SOAP, WPF, WSDL, LINQ şi o serie de limbaje de programare, precum C#, C++, F#, VB. Acestor seturi li se adaugă un mediu runtime similar cu Java Virtual Machine şi un set de API-uri. Este asigurată atât portabilitatea codului între diferite calculatoare cu mediu de operare Windows, cât şi reutilizarea codului in programe, indiferent de limbajul de programare utilizat.

Figura 3. Microsoft .NET 4.0 Framework

Toate limbajele de programare utilizate în platforma .NET respectă specificaţii OOP numite Common Type System (CTS), având ca elemente de bază clase, interfeţe, delegăti, tipuri valoare si referinţă, iar ca mecanisme moştenire, polimorfism si tratarea excepţiilor.

### Windows Form

Windows Form[5] reprezintă numele dat interfeţei grafice de programare a aplicaţiilor (API). Ceea ce în programarea Windows se numeşte o fereastră, în programarea .NET se numeşte Form. Un Form este caracterizat de un titlu, conţine bare de navigare, un meniu, toolbars, iar ceea ce rămâne se numeşte zonă client.

Form-urile reprezintă obiecte care expun proprietăţi ce definesc modul de afişare şi metode care răspund evenimentelor ce definesc interacţiunea cu utilizatorul. Setând proprietăţile unui Form şi scriind codul ce răspunde la evenimente, în fapt realizăm obiecte care se comportă conform cerinţelor cerute de aplicaţie.

### WPF

Windows Presentation Foundation (WPF) [7] a fost dezvoltat de Microsoft ca un subsistem grafic ce interpretează elementele interfeţei utilizator. Spre deosebire de Windows Form, WPF foloseşte DirectX şi permite crearea de interfeţe mai bogate în elemente grafice. DirectX este o colecţie de interfeţe de programare (API) destinate manipulării sarcinilor legate de multimedia.

Un principal avantaj al acestui model este separarea completă dintre designeri si dezvoltatori. Designerii folosesc un limbaj numit eXtensible Application Markup Language (XAML). Specificaţia XAML defineşte reguli ce mapează spaţiile de nume din .NET, tipuri, proprietăţi şi evenimente in spaţii de nume XML, elemente şi atribute. XAML este bazat pe XML, existând o mapare a tipurilor din CLR la tag-urile din XML, de la atributele din XML la proprietăţile şi evenimentele din CLR. Tag-urile XML sunt definite în contextul unui namespace şi acel namespace determină ce tag-uri sunt valide. In XAML, se mapează namespace-uri XML la colecţii de assemblies şi namespace-uri CLR.

# Analiză şi Fundamentare Teoretică

## Arhitectura Client-Server

Arhitectura Client-Server[8] este o arhitectură de reţea în care fiecare calculator sau proces din reţea este un client sau un server. În mod normal, această arhitectură împarte o aplicaţie în trei componente de bază: clientul, infrastructura reţelei (marcată prin caracterul slash) şi server-ul, fiind o arhitectură stratificată.

Calculatorul client este acela care interacţionează cu un utilizator şi care în majoritatea tipurilor de arhitecturi client/server are două sarcini: logica prezentării (interfaţa cu utilizatorul) şi logica aplicaţiei (afacerii). Într-o aplicaţie tradiţională client/server, clientul conţine partea de prezentare (afişarea folosind o interfaţă prietenoasă), lucrul efectuat de aplicaţie (calcule, algoritmi) şi manipularea datelor (conectivitatea cu bazele de date).

Calculatorul server poate fi orice calculator de birou puternic, superservere specializate bazate pe microprocesoare Intel, minicalculatoare, calculatoare mainframe sau chiar supercalculatoare şi al cărui rol într-o reţea este acela de a furniza servicii şi resurse utilizatorilor.

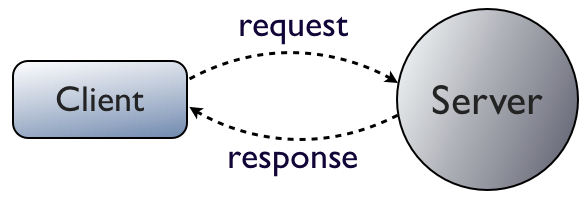
 O aplicaţie distribuită constă dintr-o parte server (back-end), unde sunt executate majoritatea procesărilor şi memorate bazele de date, şi o parte client (front-end) care furnizează o interfaţă utilizator, prin intermediul căreia utilizatorul poate executa cereri către server.

Figura 4. Comunicarea Client - Server

În modelul client/server, o aplicaţie este împărţită într-o componentă client şi o componentă server. Partea client a aplicaţiei rulează pe o staţie de lucru şi primeşte date care sunt introduse de utilizator. Componenta client pregăteşte datele pentru server, şi apoi trimite informaţiile procesate server-ului. Componenta server primeşte cererea clientului, o procesează şi returnează informaţia clientului. Clientul primeşte informaţia returnată de server şi o prezintă utilizatorului prin intermediul interfeţei utilizator.

O aplicaţie client/server este o aplicaţie distribuită alcătuită dintr-o parte server (back-end) şi o parte client (front-end). Prin aplicaţie distribuită se înţelege o aplicaţie care constă din două sau mai multe părţi care rulează pe calculatoare diferite dar care acţionează împreună.

„Client/Server” descrie relația dintre două programe, care sunt rulate pe două computere diferite. Un program, și anume Client-ul, primește anumite servicii precum date, prelucrări ale acestora, de la un alt program, care este executat pe un Server. Comunicarea dintre Client și Server, la nivel de programe, se face prin intermediul rețelei de calculatoare, așa numita LAN (Local Area Network) sau WAN (Wide Area Network).

De îndată ce un serviciu a fost efectuat de Server, conexiunea Client-ului cu Server-ul este terminată. Ea este refăcută dacă Client-ul dorește un nou serviciu de la Server. Aceste comunicații între programul de pe computerul pe care rulează Client-ul și programul care se execută pe computerul ce are funcția de Server, adică cel care oferă servicii, au loc într-un mod discret, cumva invizibil pentru utilizatorul de pe computerul Client.

## Arhitectura Model View ViewModel (MVVM)

Model-View-ViewModel (MVVM) [9] este un design pattern folosit în ingineria software, fiind introdus prima oară de către Microsoft ca o metodă de particularizare a modelului de prezentare introdus de Martin Fowler. Bazat în mare parte pe modelul Model-View-Controller (MVC), acesta se adresează în special dezvoltării interfeţei cu utilizatorii a platformelor moderne (HTML5, WPF – Windows Presentation Foundation şi Silverlight).

Ca structură, acesta este alcatuit din trei părți esențiale ce pot fi deduse și din denumirea design pattern-ului: un Model, un View și un ViewModel. Această structură seamană cu cea a MVC-ului, dar oferă în plus ușurința utilizării XAML-ului și a Windows Presentation Foundation, prin cuplarea datelor cât mai aproape de Model folosind XAML, View Model și orice verificare de date a nivelului Business pentru a valida datele înaintea afișării acestora pe interfață.

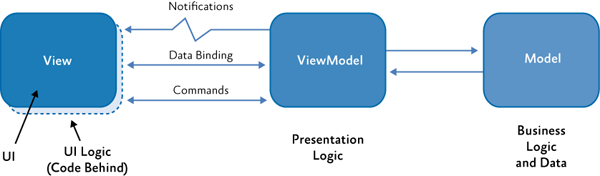
Modelul se referă la datele cu care se lucrează efectiv, dar şi la nivelul de acces la aceste date.

Figura 4. Modelul MVVM (Model – View – ViewModel)

View-urile, la fel ca și în cazul clasic se referă la partea vizuală care va fi afișată pe interfața grafică, cum ar fi butoanele, ferestrele, graficele și alte controale. Acestea nu conțin partea de logică de business. Marele avantaj în acest caz este că un designer poate să se ocupe de partea grafică a aplicației lucrând doar cu view-ul, în timp ce logica din spate rămâne neafectată.

ViewModel-urile reprezintă niște modele pentru view-uri, mai precis acestea se referă la o abstractizare a view-urilor care servesc și la binding-ul datelor între view și model. Pot fi privite și ca niște aspecte specializate ale Controalelor din design pattern-ul MVC (care acționează ca și data bindgins sau converters) în așa fel încât să schimbe informația din formatul modelului în formatul view-ului și să paseze comenzi din view în model. ViewModel-urile expun proprietățile publice, comenzile și abstractizările și au fost asemănate cu o stare conceptuală a datelor, spre depsebire de starea reală a datelor din model.

View-ul comunică doar cu ViewModel-ul, în timp ce ViewModel-ul este privit ca un punct intermediar între View și Model. De asemenea, Modelul este singurul care interacționează cu baza de date. Acest model are sens în practică doar dacă se folosește în combinație cu o bază de date. În caz contrar obiectele de date precum entitățile din EDMX și Linq nu au logică în acest context.

Alte două funcționalități care fac ca acesta să fie atât de des folosit sunt data template-urile și resursele de sistem. Template-urile aplică View-uri asupra obiectelor unui ViewModel. Întrucât aceast design pattern poate fi destul de greu de implementat, au fost create câteva platforme care să ajute programatorii în implementarea lui cum ar fi MVVM Light sau Caliburn.

MVVM oferă anumite avantaje precum separarea view-ului de logica business și ușurința în scrierea testelor unitare, însă programatorii trebuie să fie atenți dacă acest design pattern se pretează sau nu pentru proiectul lor.

## Modelul Single Page Application

Single Page Application (SPA) [10] defineşte un model de aplicaţie în care se urmăreşte evitarea reîncărcărilor succesive ale paginii afişate, optându-se pentru păstrarea paginii încărcate iniţial şi schimbarea conţinutului acesteia, prin paginile de destinaţie (landing pages). Acestea au rolul de a oferi utilizatorilor o experienţă vizuală mai fluidă, mai prietenoasă.

În general, interacţiunea cu o aplicaţie Single-Page presupune comunicare dinamică cu partea de server a acesteia. Deşi paginile nu sunt reîncarcate în toate punctele procesului, sau nu transferă controlul către alte pagini, tehnologiile moderne (ex. HTML5) oferă utilizatorilor această percepţie, chiar dacă resursele sunt încărcate dinamic, la momentul necesar, în funcţie de acţiunile utilizatorilor.

Acest concept este realizabil datorită separării datelor de prezentarea acestora prin intermediul unui nivel (layer) de model ce prelucrează datele, oferindu-le nivelului view pentru afişare.

## Design-ul Arhitectural Singleton

Pattern-ul Singleton[6] este utilizat pentru a restricţiona numărul de instanţieri ale unei clase la un singur obiect. La baza acestui pattern stă o metodă ce permite crearea unei noi instanţe a clasei dacă aceasta nu există deja. Dacă instanţa există deja, se întoarce o referinţă către acel obiect.

Diferenţa dintre o clasă cu atribute şi metode statice şi un Singleton este aceea că Singleton-ul permite instanţierea lazy, utilizând memoria doar în momentul în care acest lucru este necesar deoarece instanţa se crează atunci când se aplelează getInstance().

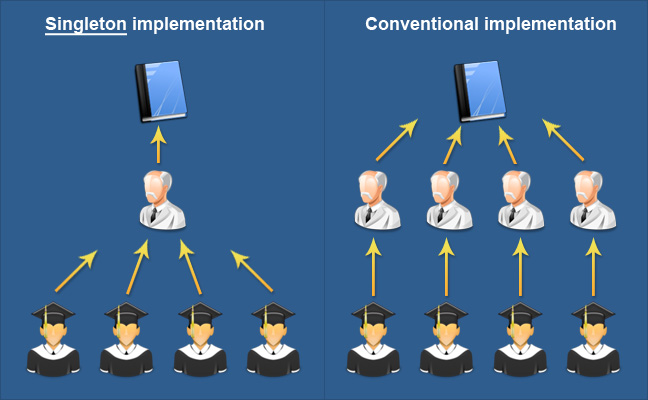
Şablonul Singleton se aplică în situaţia în care trebuie ca pentru o anumită clasă să existe doar o singură instanţă şi acesta trebuie să fie accesibilă clienţilor dintr-un punct de acces cunoscut sau în situaţia în care instanţa unică a unei clase trebuie să poată fi extinsă prin derivare, iar clienţii să poată utiliza instanţa extinsă fără a-şi modifica propriul cod.

Figura 4. Şablonul arhitectural Singleton

Şablonul Singleton are următoarele aventaje:

* controlează accesul la instanţa unică, deoarece clasa Singleton incapsulează propria instanţă
* conduce la reducerea spaţiului de nume, deoarece nu implică folosirea variabilelor globale
* permite detalierea operaţiilor şi a reprezentărilor; clasa Singleton poate fi derivată şi este uşor să se configureze o aplicaţie cu o instanţă a clasei derivate
* poate fi adaptat pentru a se crea nu doar o singură instanţă, ci un număr oarecare fixat, păstrându-se mereu controlul asupra acestui numar; pentru aceasta este necesar să se modifice doar operaţia Instance
* asigură o flexibilitate mai mare decât dacă s-ar utiliza metode statice; o alternativă la şablonul Singleton ar fi fost ca toate funcţiunile clasei Singleton să fie realizate cu ajutorul unor metode statice (practic in acest caz am fi avut de a face cu 0 instante, nu cu una)

Dezavantajele acestui şablon sunt următoarele:

* soluţia nu poate fi adaptată pentru situaţia în care dorim 'n' instanţe în loc de 1
* metodele statice nu pot fi virtuale, deci nu pot fi redefinite în subclase pentru a putea beneficia de polimorfism

## Design-ul bazei de date

### Entity Framework

Entity Framework (EF) [11] este un framework open source de mapare relaţional obiectuală pentru platforma .NET . Prin utilizarea acestuia, dezvoltatorilor li se permite să lucreze cu date sub forma unor obiecte şi proprietăţi specifice domeniului, cum ar fi clientul sau adresa clientului, fără să fie preocupaţi de tabelele din baza de date sau de coloanele unde vor fi stocate aceste date.

Arhitectura Entity Framework conţine următoarele elemente:

* Provider specific de surse de date
* Map provider – un provider specific bazei de date ce traduce command tree-ul EntitySQL într-o interogare nativă a bazei de date
* View mapping – din schema relaţională crează view-uri ale datelor corespondente modelului conceptual
* Query and update pipeline – procesează interogări, filtrează si face update cererilor
* Metadata services – gestionează metadatele legate de entităţi, relaţii si mapări
* Transactions
* Conceptual layer API – runtime-ul ce expune modelul de programare
* Embedded database – include o baza de date incorporata pentru interogarea datelor relationale
* Desing tools – simplifica task-ul de mapare a schemei conceptuale la schema relationala
* Programming layer – expune EDM-ul(Entity Data Model) ca unitati ce pot fi utilizate de limbajul de programare
* Object services – cod generat automat pentru clasele CLR ce detin aceleasi proprietati ca o entitate, dand astfel posibilitatea instantierii entitatilor ce obiecte .NET
* Web services – expun entitatile ca servicii web

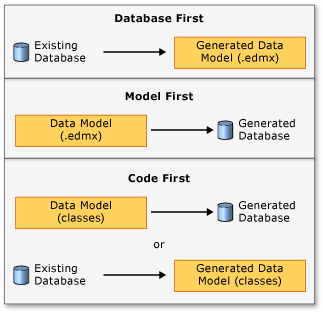
Entity Data Model (EDM) specifică modelul conceptual al datelor prin intermediul *Entity-Relationship model*, ce se ocupă în principal de entităţi şi asocierile din care acestea fac parte. În Visual Studio, EDM Wizard generează iniţial o mapare 1:1 a schemei bazei de date la schema conceptuală. In schema relaţională elementele sunt compuse din tabele, în care cheile primare şi cheile externe ţin tabelele corelate în strânsă legătură. In contrast, *Entity Types* defineşte schema conceptuală a datelor. Tipurile de entităţi sunt o agregare a unor tipuri de câmpuri multiple, fiecare câmp mapând o coloană din baza de date, câmpuri ce pot conţine informaţii din mai multe tabele fizice. Tipurile de entităţi pot fi corelate unele cu altele, independent de relaţiile din schema fizic.  Entităţile corelate sunt expuse similar prin intermediul unui câmp al cărui nume denotă relaţia în care acestea se află. Scema logică şi maparea ei pe schema fizică este reprezentată ca un EDM şi specificată ca un fisier XML.

Figura 4. Moduri utilizare Entity Framework

Entităţile sunt instanţe ale tipurilor de entităţi, reprezentând instanţe individuale ale obiectelor la care se referă informaţia. Tipul de entitate defineşte clasa de care aparţine o entitate, dar şi proprietăţile pe care o entitate le are. Proprietăţile descriu anumite aspecte ale entităţii dându-i acesteia un nume şi un tip. Oricare două tipuri de entităţi pot avea o legătură fie printr-o relaţie de tipul Association fie printr-una de tipul Contaninment, tipul de relaţie fiind specificat printr-un Relationship Type ce este caracterizat de numărul de tipuri de entitate pe care le leagă, dar şi de multiplicitatea acestora, reprezentând numărul de entităţi ce pot fi legate. Bazându-ne pe multiplicitate, relaţiile pot fi de unu-la-unu, unu-la-mulţi sau mulţi-la-mulţi. Un tip de relaţie poate avea ataşată o Acţiune sau o Operaţie. Unei relaţii i se poate specifica să efectueze o acţiune atunci când o operaţie este executată asupra unei entităţi cu care are legatură.

Baza de date pe care o folosim nu are impact asupra modelului. Putem folosi Oracle, SQLServer, Firebird, etc. Comunicarea cu baza de date se face prin furnizorii de ADO.NET pentru baza de date respectivă. Avem nevoie să identificăm providerul si stringul de conexiune la baza de date, iar cu aceste informaţii Entity Framework va reactualiza baza de date.

Entity Data Model (EDM) foloseste urmatoarele concepte pentru a descrie structura datei:

* tip entitate (entity type)
* tip asociere (association type)
* proprietate (property).

Tipul entitate descrie structura datei cu EDM. Fiecare entitate trebuie să aibă o cheie unică. O mulţime de entităţi este o colecţie a instanţelor tipului entităţii specificate. Mulţimile de entităţi sunt grupate în mod logic într-un container (entity container). Moştenirea este suportată între tipurile de entităţi.

Un tip asociere (association type) numit simplu şi asociere descrie relaţiile din EDM. Asocierea reprezintă o relaţie între două tipuri de entităţi. Multiplicitatea unei asocieri poate fi 0..1 sau mai multe. Entităţile ce folosesc asocieri pot fi accesate prin proprietăţile de navigare sau prin “foreign key”. Instanţele asocierilor sunt grupate în mod logic în “entity container”.

Tipurile entităţi conţin proprietăţi ce definesc structura acestora. O proprietate poate conţine tipuri primitive (string, int, Boolean, etc.) sau tipuri structurate (tipuri complexe).

## Iniţializare lazy

Acest şablon se feră la faptul că în majoritatea cazurilor când se apelează un obiect nu vom avea nevoie de toate dependenţele legate de acest obiect. Aceste valori le vom încărca prima dată în momentul în care vom avea nevoie de ele. În acest mod creşte performanţa aplicaţiei, întrucât nu se efectuează atâtea tranzacţii la baza de date.

Aplicaţia va avea multe procese costisitoare în ceea ce priveşte puterea de calcul, procese ce nu vor fi folosite de mare parte din utilizatori, sau vor fi folosite doar de puţine ori. Prin aplicarea iniţializării lazy[11] vom câştiga o performanţă totală asupra aplicaţiei.

## Încărcare lazy

Lazy loading[11] este un model de proiectare utilizat în mod obişnuit în programe cu scopul de a amâna iniţializarea unui obiect, până în punctul în care este necesar.

Acest model poate fi realizat prin mai multe mijloace, printre care se numără şi iniţializarea târzie (lazy initialization), descrisă anterior. Această tehnică poate contribui la eficienţa în funcţionare a programului, oferind o creştere de performanţă prin utilizarea datelor în momentul şi în modul corespunzător.

## Cuplare slabă

Elementele unui sistem cu cuplare slabă[6] folosesc puţine sau chiar niciuna dintre definiţiile componentelor cu care se cuplează. În programarea orientată pe obiecte aceasta se translatează la sistem, unde clasele au un grad minim de cunoştinţe unele despre altele.

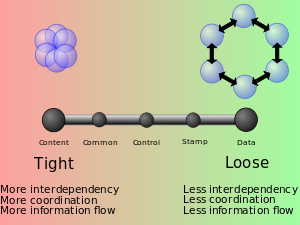
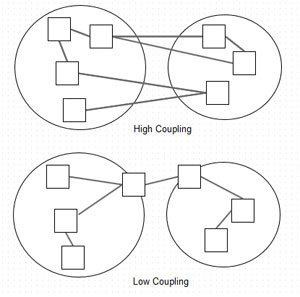
Acest concept poate fi atins prin dezvoltarea claselor care vor accesa interfeţe în locul claselor concrete. O interfaţă poate fi implementată de mai multe clase, iar schimbarea claselor şi modificarea conţinutului din spatele unei interfeţe nu va afecta clasele dependente.

Figura 4. Cuplare slabă

Cuplarea reprezintă forţa interconexiunilor dintre module. Atingerea scopului de cuplare slabă se poate realiza prin următoarele tehnici:

* Minimizarea numărului de interfeţe ale unui modul
* Minimizarea complexităţii interfeţelor (numărul de parametri)
* Trimiterea ca parametri numai a câmpurilor necesare, nu a întregului obiect
* Evitarea comunicaţiilor hibride



Cuplarea minimă are loc atunci când clasele derivate nu modifică metodele moştenite, ci doar adaugă noi câmpuri şi metode.

Figura 4. Cuplare puternică şi cuplare slabă

## Coeziune mare

Termenul de coeziune[6] descrie nivelul de interdependenţă a componentelor. Cu alte cuvinte, coeziunea reprezintă gradul în care un modul este „concentrat” pe responsabilităţile sale. Este de preferat un grad înalt de coeziune, intrucât astfel este asigurat un nivel mare (sau, cel puţin, mai ridicat) de flexibilitate în ceea ce priveşte lizibilitatea si posibilitatea de reutilizare a componentelor. Aşadar, mentenabilitatea codului şi coeziuna sunt direct proporţionale. În paradigma OOP, la nivelul claselor, este important ca, în vederea asigurării unui nivel ridicat de coeziune, metodele unei clase să fie cât mai „indispensabile” atingerii „scopului” acesteia. Pe scurt, metodele trebuie să servească un „scop comun”.

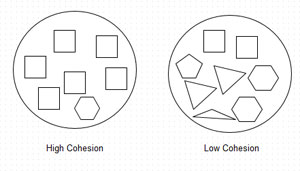
În general, o coeziune puternică este corelată cu o cuplare slabă. Coeziunea unui modul este maximul de coeziune aplicabil tuturor elementelor acestuia.

Figura 4. Coeziune mare

Coeziunea este mare dacă ierarhia urmăreşte generalizarea / specializarea codului şi este mică dacă scopul ierarhiei este partajarea codului.

## Considerente de securitate

Autentificarea este procesul de verificare a identității (digitale) a unui participant la comunicație, de obicei prin indicarea unui cod de utilizator mai mult sau mai puțin public și a unei parole secrete (*password*).

În astfel de situații "utilizatorul" este de cele mai multe ori o persoană fizică, dar uneori este foarte practic ca și un program sau chiar un calculator să primească un cod de utilizator propriu și o parolă, cu ajutorul cărora el se poate autentifica, de exemplu în cadrul unei rețele de calculatoare.

Autentificarea nu trebuie confundată cu autorizarea. În general, întâi are loc autentificarea cuiva, și numai dacă ea reușește, utilizatorului respectv i se acordă și drepturile de folosință (autorizarea) cuvenite lui, pe baza unor reguli prestabilite. La sistemele care necesită mai multe drepturi diferite de folosință este chiar și mai important ca autentificarea să funcționeze corect, astfel ca nimeni să nu-și depășească îndreptățirile primite pe baza ei.

Cea mai simplă metodă de autentificare este metoda de autentificare tradițională. În acest model, informațiile despre numele de utilizator și parola pentru fiecare utilizator autentificabil sunt stocate local pe un server. Utilizatorii trimit numele de utilizator și parola în clar (plain text) sistemului, care la rândul său compară informația de autentificare cu baza de date locală. În cazul în care numele de utilizator și parola se potrivesc, utilizatorul este autentificat.

Acesta este în principiu modelul folosit la autentificări cu login pe tradiționalele sisteme multiutilizator, unde a fost reprodus de nenumărate ori pe diverse pachete de aplicații.

Acest model are următoarele puncte slabe:

* În multe cazuri, parolele utilizatorilor sunt stocate în clar pe un server. Oricine are acces la baza de date a serverului are acces la suficientă informație pentru a impersona un utilizator autentificabil.
* În cazurile în care parolele utilizatorilor sunt stocate codificat (encrypted) pe server, parolele în clar sunt probabil trimise prin rețele nesigure de la client la server. Oricine are acces la această porțiune de rețea poate prelua aceste date personale (numele de utilizator și parola, chiar codificate) și să le utilizeze ulterior pentru a falsifica autentificarea în sistem.
* Fiecare sistem separat trebuie să dețină o copie a informației de autentificare a utilizatorului. Ca rezultat, utilizatorii trebuie să-și actualizeze parolele pe fiecare sistem la care se autentifică; pentru comoditate, utilizatorii aleg parole mai simple, ceea ce este riscant.
* Autentificarea nu este refolosibilă. Utilizatorii trebuie să se autentifice pe fiecare sistem sau aplicație pe care ei doresc să îl/o acceseze. Ca rezultat, utilizatorii trebuie să reintroducă parolele, iar aceasta are efectul că utilizatorii vor tinde să utilizeze o parolă mai simplă, pentru comoditate.
* Un sistem care impersonează sistemul real (prin IP spoofing), pe care utilizatorul nu-l poate distinge de serverul autentic, deschide posibilitatea pentru serverul impostor să colecteze informația personală (nume de utilizator și parola), care va putea fi apoi folosită la autentificare pe serverul real.

În baza de date, parolele sunt ţinute sub forma criptată, nu se decriptează nicăieri, nici măcar la login. În cadrul operaţiei de login, nu se decriptează parola salvată în baza de date, ci se criptează parola introdusă şi se compară cele 2 rezultate.

Criptarea este procesul de ascundere a informației pentru a o face ilizibilă fără cunoștințe speciale. Criptarea a fost folosită pentru protejarea comunicațiilor de secole, dar doar organizații sau indivizi cu necesități de intimitate extraordinare s-au preocupat de a o implementa. În prezent, este utilizată în protejarea unei mari varietăți de sisteme, precum e-comerț, rețele de telefonie mobilă și ATM-urile băncilor.

Criptarea reprezintă un mod de a îmbunătăți securitatea unui mesaj sau fișier prin amestecarea conținutului astfel încât să nu mai poată fi citit decât de cineva care are cheia de criptare corespunzătoare pentru a-l repune în ordine.

Criptarea poate fi folosită pentru a asigura discreția și/sau intimitatea, dar și alte tehnici sunt necesare pentru a face comunicațiile sigure, în mod particular verificarea integrității și autenticității unui mesaj; de exemplu, un cod de autentificare a mesajelor (CAM) sau semnături digitale. Altă considerentă este protecția împotriva analizei traficului.

Criptarea sau ascunderea codului de software este folosită în protecția copierii de software împotriva ingineriei inverse, analiza aplicațiilor neautorizate, crack-uri și pirateriei software.

## Protocoale utilizate

SOAP[1] este un protocol de mesaje extensibil bazat pe XML ce constituie fundamentul pentru serviciile web. SOAP ofera un mecanism simplu si consistent ce permite unei aplicaţii să trimită un mesaj XML unei alte aplicaţii.

SOAP oferă suport pentru comunicaţiile tip peer-to-peer. Un mesaj SOAP este o transmisie one-way de la un trasmiţător SOAP la un receptor SOAP, orice aplicaţie poate participa în acest schimb de mesaje fie ca transmiţător fie ca receptor. Mesajele SOAP pot fi combinate să suporte deverse tipuri de comunicaţii de ex: cerere/răspuns (request/response) cerere de răspuns (solicit response) sau notificare (notification).

SOAP conţine următoarele părţi:

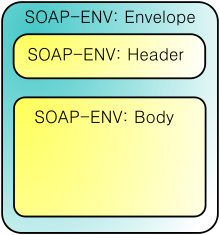
* SOAP Envelope –furnizează un mecanism de identificare a conţinutului mesajului şi de explicitare a modului în care trebuie procesat (tratat) mesajul. SOAP Envelope include un SOAP header si un SOAP body. Body poate conţine orice noduri copil este nevoie. Acesta este definit în aşa fel încât poate conţine orice XML valid şi foarte bine format ce a fost calificat şi nu poate conţine vreo instrucţiune de procesare sau Document Type Definition (DTD). Dacă Envelope conţine Header atunci numărul acestuia trebuie limitat la unul singur şi trebuie să apară ca fiind primul nod copil, înaintea Body-ului. La fel, header-ul poate conţine XML valid şi bine format pe care creatorul mesajului SOAP vrea să îl insereze.

Figura 4. Componente SOAP

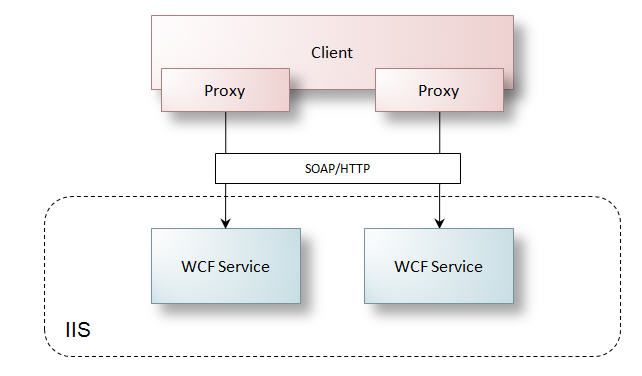
* SOAP Transport Binding Framework – furnizează un mecanism extensibil ce mapează SOAP Envelope la protocoalele de transport de nivel inferior. Specificaţia SOAP 1.2 defineşte legăturile de transport pentru HTTP si HTTP Extensions Framework. De asemenea, se pot dezvolta legături şi pentru alte protocoale de transport ca SMTP, JMS sau MQSeries.
* SOAP Enconding Rules- toate datele transmise printr-un mesaj SOAP sunt codate utilizând XML, dar nu există un mecanism implicit de serializare pentru maparea tipurilor de date definite de aplicaţie la elementele XML. Datele pot fi transmise literal sau ca valori codate. Utilizatorii pot defini propriul lor mecanism sau pot utiliza mecanismul de serializare definit de regulile de codare SOAP. Aceste reguli specifică în mod detaliat cum tipurile de date ale unei aplicaţii basic sunt mapate si encodate intr-un format XML. Stilurile de encoding sunt complet opţionale şi în foarte multe dintre situaţii nu sunt folositare. Envelope-urile SOAP sunt create pentru a putea conţine orice document XML indiferent cum poate arăta corpul mesajului sau se conformizeaza unui set specific de reguli.

Figura 4. Comunicarea SOAP

În esenţă, acest protocol este bazat pe limbajul XML, având următoarele caracteristici funcţionale:

* controlul transferului de pachete de date între furnizorul de servicii Web şi

utilizatorul acestora, folosind protocolul HTTP (metode GET sau POST)

pentru transferul pachetelor de date între server si utilizator;

* transferul parametrilor stabiliţi de utilizator şi specifici funcţiilor accesibile

prin intermediul serviciului Web;

* returnarea rezultatelor rulării funcţiilor pe serverul care furnizează serviciul

Web, aceste rezultate fiind în realitate seturi de date care au fost transpuse

în fişiere XML;

* să fie transmis prin cât mai multe căi posibile prin reţea. Acest lucru este

necesar deoarece multe companii doresc să folosească doar anumite

protocoale de transport pentru o mai bună securitate. Cea mai bună soluţie

este folosirea protocolului HTTP datorită posibilităţii de a nu fi blocat de

firewall;

# Proiectare de Detaliu si Implementare

## Baza de date

Pentru implementarea bazei de date am folosit utilitarul Microsoft SQL Server Management 2012.

După etapa în care am centralizat şi am înţeles cerinţele de implementare propuse, am trecut la proiectarea bazei de date, prin identificarea tabelelor necesare, câmpurilor acestora şi a relaţiilor dintre acestea.

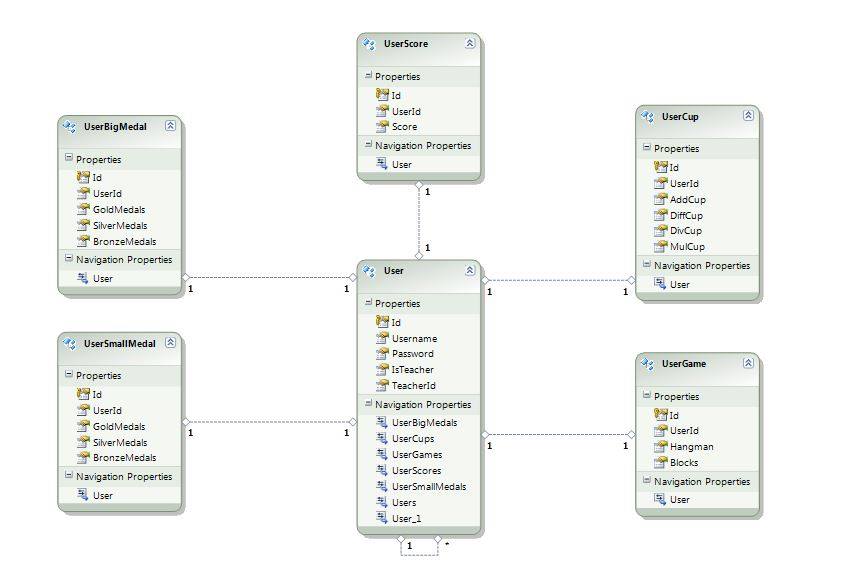
Figura următoare prezintă arhitectura bazei de date, cu tabelele şi relaţiile dintre ele:

Figura 5. Diagrama bazei de date

Tabela „User” conţine informaţiile principale despre utilizatori, precum credenţialele, rolul utilizatorului (elev sau profesor) şi ID-ul profesorului, în cazul în care tipul utilizatorului este „elev”.

Tabela „UserScore” conţine informaţii legate de scorul utilizatorilor.

Tabela „UserGame” conţine informaţii legate de rezultatele jocurilor din aplicaţie pentru fiecare utilizator. Chiar dacă această tabelă conţine informaţii legate de scorul obţinut de profesori în cadrul procesului de verificare a funcţionalităţii corecte a aplicaţiei, rezultatele obţinute de aceştia nu vor fi afişate în topurile din aplicaţie. Câmpurile "Hangman” şi „Blocks” conţin informaţii legate de scorurile obţinute în cadrul acestor jocuri.

Tabela „UserCup” conţine informaţii legate de numărul cupelor obţinute de utilizatorii aplicaţiei pentru fiecare operaţie. La fiecare test, dacă utilizatorul obţine nota 10, aceasta este corespondentă unei cupe de aur, nota 9 unei cupe de argint şi nota 8 unei cupe de bronz. În baza de date, toate cupele vor fi stocate în acelaşi loc şi nu separat, în funcţie de tipul lor (aur, argint, bronz). Chiar dacă această tabelă conţine informaţii legate de scorul obţinut de profesori în cadrul procesului de verificare a funcţionalităţii corecte a aplicaţiei, rezultatele obţinute de aceştia nu vor fi afişate în topurile din aplicaţie.

Tabela „UserSmallMedal” conţine informaţii legate de medaliile mici obţinute de utilizatorii aplicaţiei. Aceste medalii sunt stocate în baza de date în funcţie de tipul lor, spre deosebire de tabela descrisă anterior. Medaliile mici sunt obţinute în urma rezolvării puzzle-urilor mici, indiferent de tipul operaţiei pentru care a fost rezolvat puzzle-ul. Chiar dacă această tabelă conţine informaţii legate de scorul obţinut de profesori în cadrul procesului de verificare a funcţionalităţii corecte a aplicaţiei, rezultatele obţinute de aceştia nu vor fi afişate în topurile din aplicaţie.

Tabela „UserBigMedal” conţine informaţii legate de medaliile mari obţinute de utilizatorii aplicaţiei în urma rezolvării puzzle-urilor mari. Chiar dacă această tabelă conţine informaţii legate de scorul obţinut de profesori în cadrul procesului de verificare a funcţionalităţii corecte a aplicaţiei, rezultatele obţinute de aceştia nu vor fi afişate în topurile din aplicaţie.

În continuare am descris relaţiile dintre tabelele din baza de date:

Între tabelele „User” şi „UserScore” există relaţie de 1 la 1, fiecare utilizator poate avea un singur scor în cadrul aplicaţiei.

Între tabelele „User” şi „UserGame” există relaţie de 1 la 1, fiecare utilizator are un singur rezultat pentru fiecare joc de tip Hangman sau Blocks.

Între tabelele „User” şi „UserCup” există relaţie de 1 la 1, fiecare utilizator are o singură informaţie legată de cupe pentru fiecare tip de operaţie matematică.

Între tabelele „User” şi „UserSmallMedal” există relaţie de 1 la 1, fiecare utilizator are o singură informaţie pentru fiecare tip de medalie mică (aur, argint, bronz).

Între tabelele „User” şi „UserBigMedal” există relaţie de 1 la 1, fiecare utilizator are o singură informaţie pentru fiecare tip de medalie mare (aur, argint, bronz).

Tabela „User” are o relaţie recurentă, întrucât fiecare elev este asignat unui profesor, ambii fiind de tip user (utilizator al aplicaţiei).

Operaţiile pe baza de date se realizează pe baza Entity Framework 4.0 . În cadrul Entity Framework s-a utilizat utilitarul LINQ. Modelul Entity Framework mapeaza tabelele bazei de date în clase C#, transpunând astfel fiecare tabelă într-o colecţie, facilitând astfel utilizarea LINQ pe această colecţie. Modelul de Entity Framework se află pe Server, în cadrul serviciului WCF. Serverul conţine în fişierul de configurări string-ul conexiunii la baza de date:

<connectionStrings>

<add name="InvatamSaCalculamEntities" connectionString="metadata=res://\*/Models.DBModel.csdl|

res://\*/Models.DBModel.ssdl|res://\*/Models.DBModel.msl;

provider=System.Data.SqlClient;provider connection string=&quot;

data source= BOGDAN-PC;initial catalog=InvatamSaCalculam;

integrated security=True;multipleactiveresultsets=True;

App=EntityFramework&quot;"

providerName="System.Data.EntityClient" />

</connectionStrings>

## Diagrama de componente ale aplicaţiei

Aplicaţia este compusă pe baza arhitecturii Client-Server, comunicarea realizându-se prin transmiterea de mesaje între aceste două componente, prin serviciul WCF.

Componentele aplicaţiei sunt clientul, serverul şi baza de date. Conexiunea cu baza de date este realizată pe server.

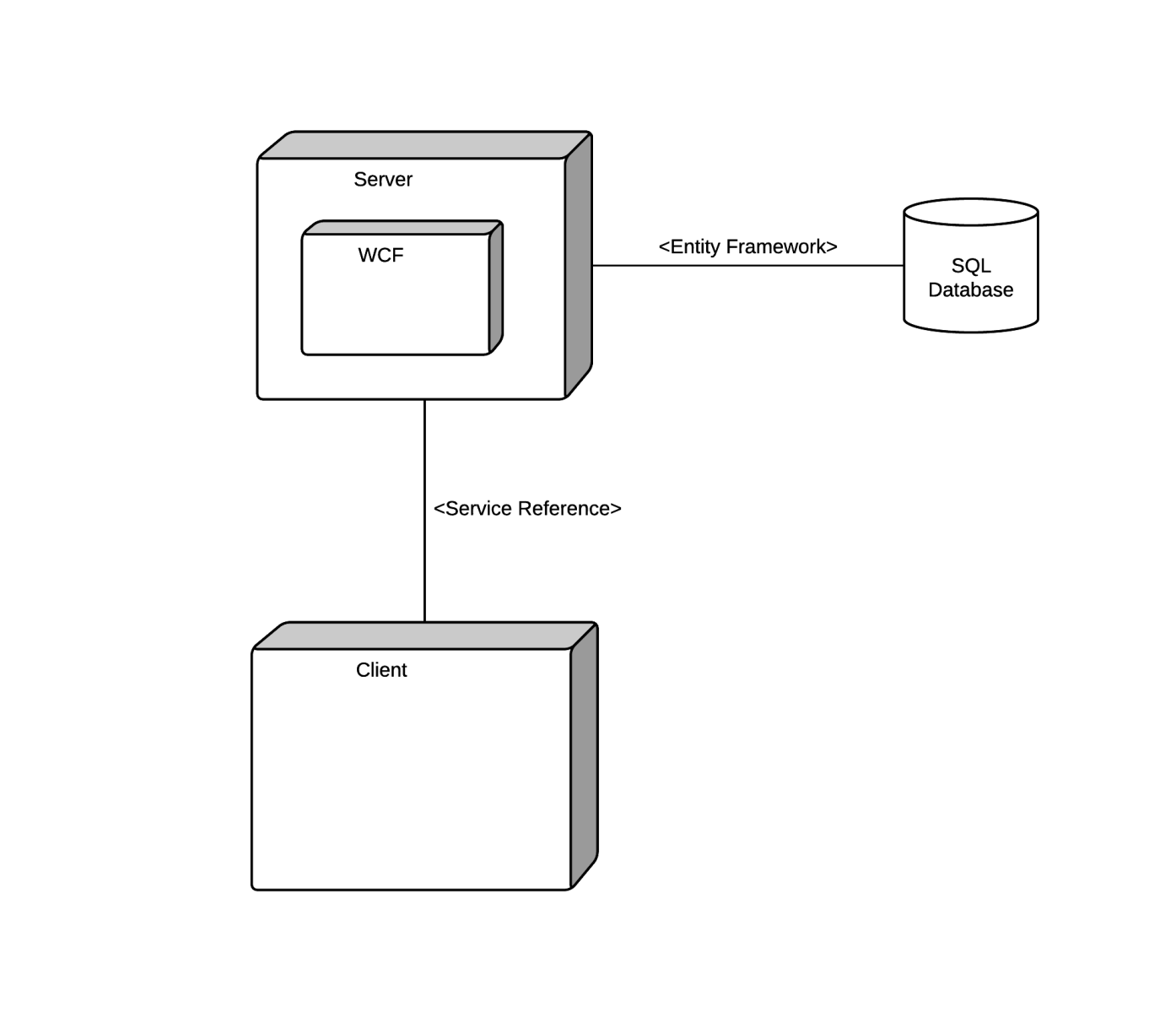
În figura următoare este ilustrată schema generală a aplicaţiei:

Figura 5. Diagrama de componente

Conexiunea serverului la baza de date este realizată prin intermediul Entity Framework 4.0, iar comunicarea dintre client şi server se realizează prin referinţă de serviciu.

## Configurarea serviciului WCF

Pentru ca cele două componente, clientul şi serverul să poată comunica fişierul de configurare trebuie setat corespunzător. Trebuie indicate informaţiile de adresă, binding şi contact pentru endpoint-urile serviciului (punctele de comunicare ale serviciului).

În continuare am ilustrat configurarea corespunzătoare de pe server:

<client>

<endpoint binding="basicHttpBinding" bindingConfiguration=""

contract="InvatamSaCalculam.Server.IServerService" />

</client>

<services>

<service name="InvatamSaCalculam.Server.ServerService">

<endpoint binding="basicHttpBinding" bindingConfiguration=""

contract="InvatamSaCalculam.Server.IServerService" />

</service>

</services>

<connectionStrings>

<add name="InvatamSaCalculamEntities" connectionString="metadata=res://\*/Models.DBModel.csdl|res://\*/Models.DBModel.ssdl|res://\*/Models.DBModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=BOGDAN-PC;initial catalog=InvatamSaCalculam;integrated security=True;multipleactiveresultsets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

</connectionStrings>

## Diagrama de clase

Acest subcapitol conţine structura de clase folosită în aplicaţie, pentru componentele client şi server.

### Diagrama de clase a componentei Server

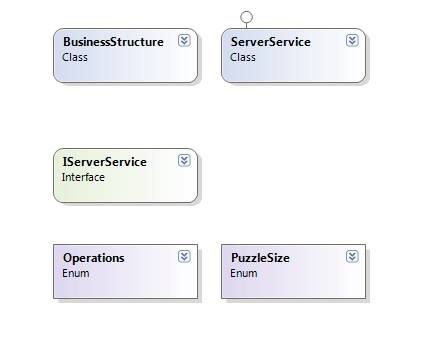


Figura 5. Diagrama UML a componentei Server

Clasa “BusinessStructure” adaptează tiparul Singleton şi conţine instanţa modelului bazei de date.

Interfaţa „IserverService” conţine header-ele operaţiilor oferite de serviciu.

Clasa „ServerService” implementează interfaţa „IServerService”. Aceasta conţine implementările funcţiilor. Această clasă reprezintă serviciul propriu-zis.

### Diagrama de clase a componentei Client

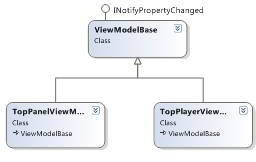




Figura 5. Diagrama UML a componentei Client

Clasa “Program” este una statică, are rolul de a porni aplicaţia, iniţializând fereastra principală.

Clasa “MainPage” este fereastra principală a aplicaţiei, extinde clasa “Form”. Aceasta conţine instanţe pentru toate paginile aplicaţiei (User Controale). În această fereastră se găsesc butoanele din meniul superior, precum butonul de părăsire a aplicaţiei, butoanele ce controlează melodiile din aplicaţie, Logout, Meniu şi Înapoi. În momentul în care este necesară afişarea altui conţinut, această fereastră rămâne deschisă, schimbându-se doar conţinutul ei.

Clasa „Login” extinde clasa „UserControl”. În aceasta se găsesc câmpurile de introducere a datelor de logare şi butoanele aferente.

Clasa „Register” extinde clasa „UserControl”. Aceasta se ocupă de înregistrarea unui utilizator nou în baza de date. Aceasta conţine şi un drop-down pentru selectarea profesorului de care aparţine elevul.

Clasa „MainMenu” extinde clasa „UserControl”. Aceasta con Aceasta conţine butoane pentru fiecare secţiune a aplicaţiei.

Clasele „Add”, „Div”, „Diff” şi „Mul” extind clasa „UserControl”. Acestea definiţiile operaţiilor şi probele operaţiilor. De asemenea, acestea conţin şi butoane pentru teste şi ecuaţii corespunzătoare fiecărei operaţii.

Clasele „AddEquation”, „DivEquation”, „DiffEquation” şi „MulEquation” extind clasa „UserControl”. Acestea conţin ecuaţiile cu doi termeni corespunzătoare fiecărei operaţii.

Clasele „AddTest”, „DivTest”, „DiffTest” şi „MulTest” extind clasa „UserControl”. Acestea conţin câte un test compus din câte 7 exerciţii pentru fiecare operaţie şi butoane pentru verificarea testului, un buton de Admin care completează rezultatele exerciţiilor şi un buton „Test Nou” care generează un nou test. În funcţie de rezultatul testului, la verificare, se obţine o cupă.

Clasa „PuzzleMenu” extinde clasa „UserControl”. La afişare, aceasta afişează 2 butoane pentru selectarea de puzzle mic sau mare, iar după selecţie afişează 23 de puzzle-uri din care utilizatorul alege unul.

Clasa „Puzzle” extinde clasa „UserControl”. Aceasta conţine un exerciţiu, butonul de veroificare a răspunsului, un buton de Admin care rezolvă exerciţiul automat, un scor şi un buton de reiniţializare a jocului.

Clasa „HallOfFame” extinde clasa „UserControl”. Aceasta conţine rezultatele obţinute de utilizatorul logat în cadrul testelor şi al jocurilor.

Clasa „Blocks” extinde clasa „UserControl”. Aceasta conţine scorul, un buton pentru joc nou şi blocurile ce reprezintă jocul.

Clasa „Hangman” extinde clasa „UserControl”. Aceasta conţine scorul, un buton pentru joc nou, scorul şi un exerciţiu de matematică.

Clasa „ManageStudents” extinde clasa „UserControl”. Aceasta este disponibilă numai utilizatorilor de tip „Profesor”, conţinând un tabel cu rezultatele elevilor asignaţi profesorului. Este disponibilă resetarea scorului fiecărui elev şi eliminarea elevului din clasa profesorului. Eliminarea unui elev îi va şterge contul din baza de date.

Clasele „Top”, „ViewModelBase”, „TopPanelViewModel”, „TopPlayerViewModel”, „TopPanelView”, „TopPlayerView” şi „TopHeaderView” sunt responsabile pentru afişarea topului. Toate, mai puţin „Top” sunt realizate în WPF, fiind folosite în cadrul clasei „Top”, în care sunt afişate.

Clasele „PasswordHandler”, „QuestionHandler”, „BusinessStructure” sunt clase care se ocupă cu administarea entităţilor pe care le reprezintă.

## Diagrama cazurilor de utilizare

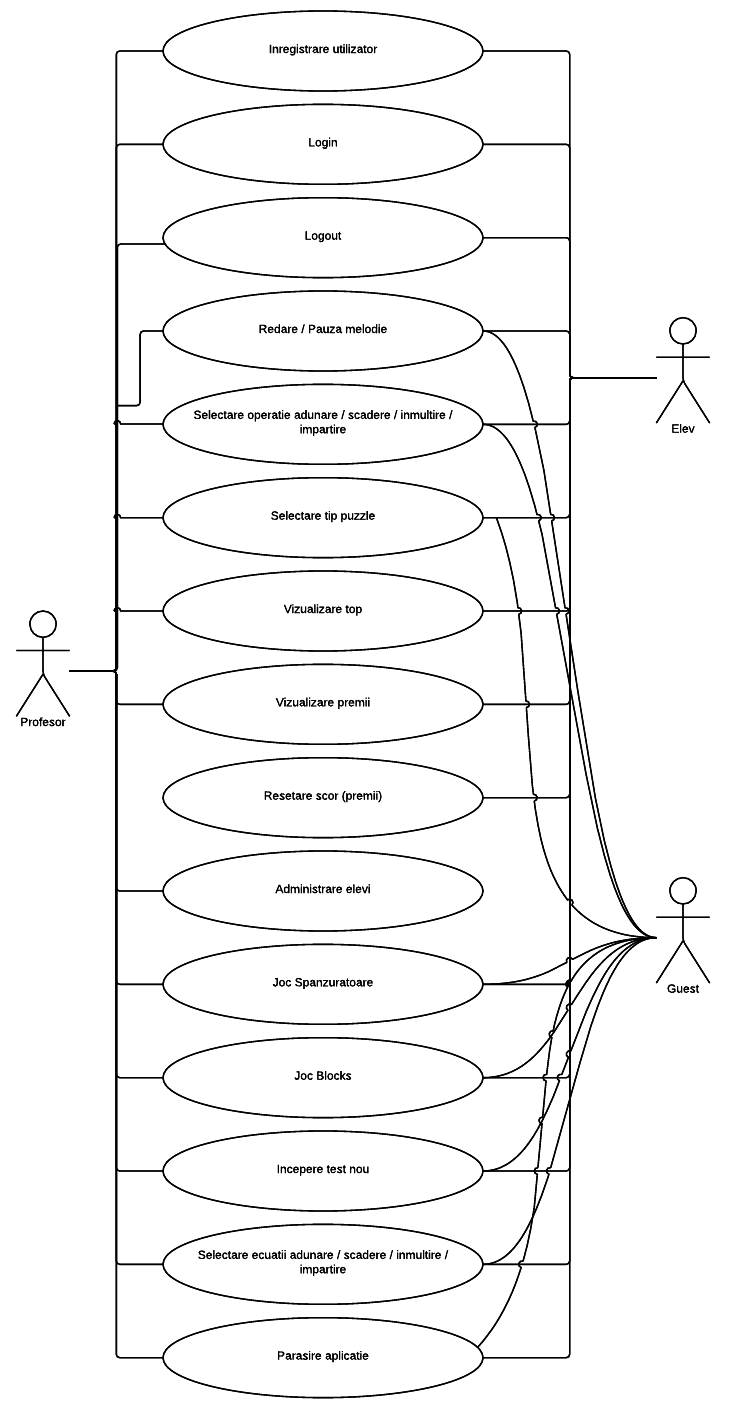


Figura 5. Diagrama cazurilor de utilizare

În cazul în care conexiunea la Server nu este disponibilă, utilizatorii se vor putea loga doar ca şi Guest. Acest tip de utilizator are posibilitatea de a accesa aplicaţia, de a porni sau opri muzica aplicaţiei, de a selecta operaţia matematică pe care doreşte să o înveţe / exerseze, de a selecta un tip de puzzle şi mai apoi un puzzle anume, de a selecta un joc Spânzurătoare sau Blocks, de a începe un test nou, de a selecta şi completa ecuaţii şi de a părăsi aplicaţia. Am decis asupra utilizării unui astfel de utilizator din pricina frecventelor probleme cu conexiunea la Internet sau a problemelor legate de componentele fizice ale infrastructurii pe care rulează componenta Server a aplicaţiei. Astfel, am reuşit apropierea de pragul de utilizare al aplicaţiei („Availability”) de 24 de ore din 24, presupunând că aplicaţia funcţionează în mod corespunzător.

Utilizatorul tip „Elev” va fi reprezentat de marea parte a persoanelor care accesează aplicaţia. Aplicaţia a fost proiectată şi construită în jurul acestui tip de utilizator. Astfel, utilizatorii de tip „Elev” au posibilitatea de a accesa aplicaaccesa aplicaţia, de a se înregistra în aplicaţie, de a se loga (login) şi de a se deloga (logout), de a porni sau opri muzica aplicaţiei, de a selecta o operaţie matematică din meniul aplicaţiei, de a selecta un tip de puzzle (mare sau mic) şi ulterior de a alege un puzzle pe care sa îl completeze, de a selecta un joc de tip „Spânzurătoare” sau „Blocks”, de a începe un test nou, de a selecta un tip de ecuaţie, de a vizualiza premiile sale şi topul şi de a părăsi aplicaţia. De asemenea, elevii îşi pot reseta scorul / punctajul obţinut în cadrul testelor şi jocurilor. Fiecare utilizator de acest tip trebuie să îşi aleagă un profesor în momentul înregistrării în aplicaţie.

Utilizatorul tip „Profesor” are rolul şi permisiunea de a testa buna funcţionare a aplicaţiei şi de a-şi administra elevii asignaţi. Astfel, acesta, pe lângă permisiunile unui elev, are dreptul de a administra elevii care îi sunt asignaţi. Are permisiunea de a le reseta scorurile / punctajele obţinute în cadrul testelor şi al jocurilor şi de a şterge elevii care i-au fost asignaţi din greşeală. Odată şters de către profesor, un elev trebuie să se înregistreze din nou în aplicaţie. De asemenea, ca urmare a jocurilor şi testelor pe care acest tip de utilizator le completează, sunt afişate rezultate / punctaje pe care acesta le poate vizualiza, însă aceste rezultate nu vor fi incluse în Topul Rezultatelor. Acest tip de utilizator are disponibil şi un buton „Admin”, care va completa automat rezultatele testelor, în cadrul operaţiunilor de verificare a bunei funcţionări a aplicaţiei.

## Diagrama de secvenţă pentru operaţia de Login

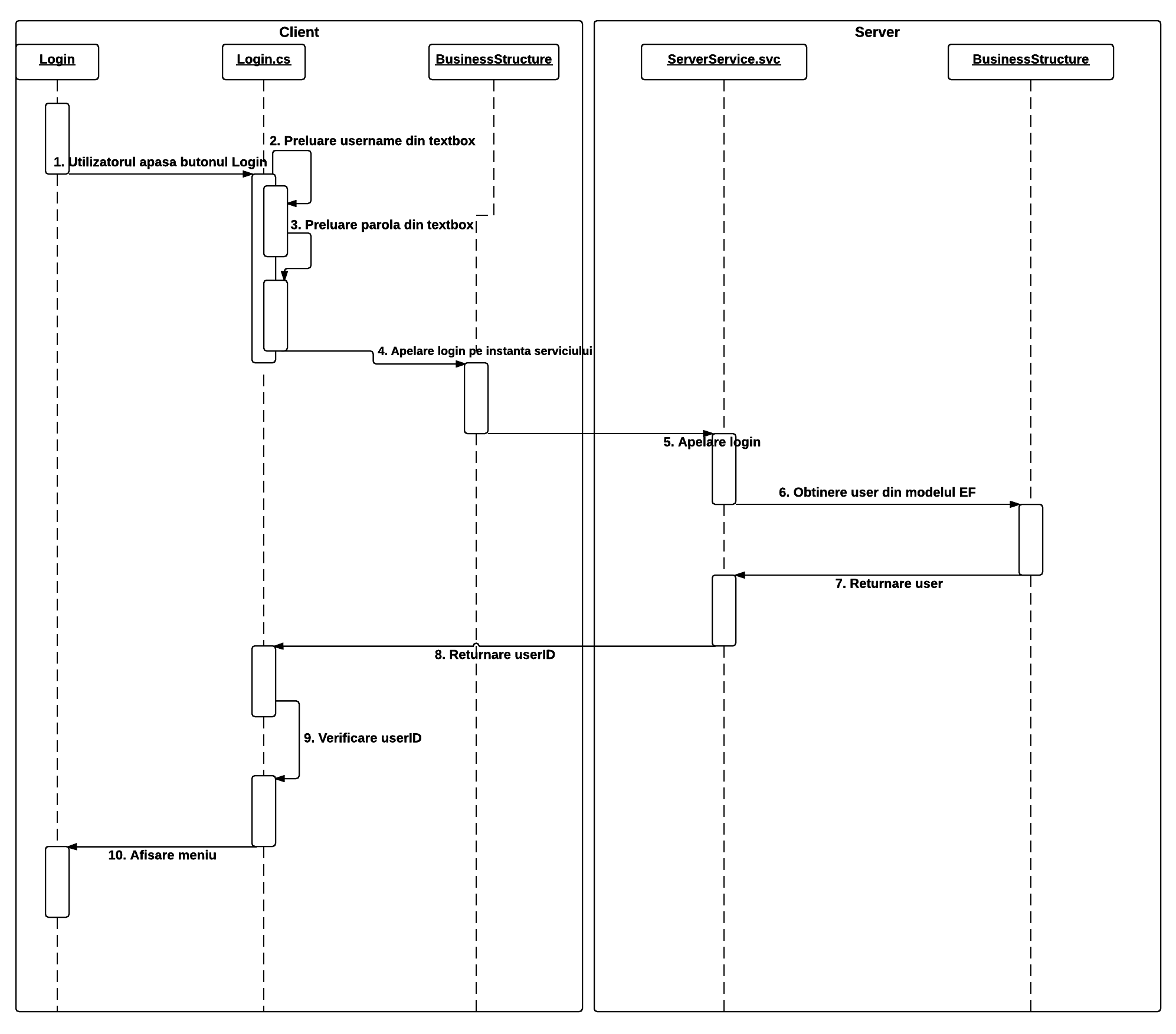


Figura 5. Diagrama de secvenţă a operaţiei Login

Operaţia de logare poate fi descrisă de următorii paşi:

* Utilizatorul îşi completează datele de logare
* Utilizatorul apasă butonul de Login
* Fereastra de Login preia datele din textBox-uri şi apelează cu ele funcţia login din serviciul WCF
* Serviciul preia datele de logare şi identifică utilizatorul corespunzător acestora şi returnează clientului ID-ul utilizatorului, dacă datele au fost corecte, returnează zero dacă doar numele de utilizator a fost corect şi NULL în cazul în care atât utilizatorul cât şi parola au fost greşite
* Clientul, în funcţie de răspunsul primit, afişează un mesaj de eroare sau fereastra de meniu

## Proiectarea interfeţei cu utilizatorul

În cadrul procesului de proiectare şi construire a interfeţei cu utilizatorul am utilizat WinForms şi WPF.

În WinForms am utilizat următoarele controale:

* Butoane
* Etichete (Label)
* TextBox-uri
* CheckBox-uri
* DataGridView
* DropDownList (ComboBox)
* PictureBox

În WPF am utilizat următoarele controale:

* Grid
* TextBox
* Border
* ScrollViewer
* StackPanel
* ItemsControl
* LinearGradientBrush
* Image

WPF a fost utilizat în construirea paginii care afişează topul rezultatelor. Am decis asupra utilizării sale pentru o personalizare mai bună a controalelor.

## Generarea dinamică a termenilor din cadrul testelor şi jocurilor

Pentru o experienţă diferită la fiecare utilizare şi pentru a obţine unicitatea datelor de test, am decis asupra generării dinamice a termenilor operaţiilor. Limitele intervalului de generare a numerelor au fost 0 şi 99, inclusiv.

În cadrul jocurilor „Spânzurătoare” şi „Blocks” exerciţiile, operaţiile şi termenii sunt generate în mod dinamic. Astfel, pentru selectarea tipului de operaţie se generează aleator un număr cuprins între 0 şi 3, inclusiv, iar operaţia este selectată în funcţie de acest număr.

## Comunicarea în cadrul aplicaţiei

Comunicarea dintre componentele Client şi Server se realizează utilizând protocolul SOAP, prin intermediul referinţelor serviciilor. Acest protocol este utilizat de către serviciul WCF.

Spre exemplu, în cadrul operaţiei de autentificare a unui utilizator (Login), aplicaţia funcţionează astfel:

* Se verifică validitatea câmpurilor cu numele de utilizator şi cu parola;
* Se preia numele de utilizator şi se caută în baza de date a aplicaţiei, prin apelarea metodei Login() din serviciul WCF intitulat ServerService;
* În cazul în care numele de utilizator este găsit în baza de date a aplicaţiei, se verifică parola prin compararea parolei corecte criptate cu parola introdusă, criptată prin intermediul PasswordHandler de pe Client;
* În cazul în care cele două parole corespund, serviciul WCF returnează ID-ul utilizatorului, iar în caz contrar este returnat ID-ul ‚0’, iar dacă utilizatorul nu a fost găsit în baza de date, este returnată valoarea NULL;

## Implementarea şablonului Singleton

Şablonul Singleton a fost utilizat în cadrul claselor de manipulare a datelor (a parolelor, respectiv a structurii de business a aplicaţiei), pentru a permite utilizarea acestor Handlere în mai multe locuri în aplicaţie (BusinessStructure este utilizat în toate locurile unde este nevoie de operaţii pe baza de date; PasswordHandler este utilizat în cadrul operaţiilor de logare şi înregistrare), fără a crea mai multe instanţe ale acestora şi pentru accesarea facilă a acestora, fără a fi necesară includerea referinţei în fiecare clasă în cadrul aplicaţiei. Instanţele acestor Handlere sunt statice.

## Motivarea utilizării şablonului Model – View – ViewModel (MVVM)

Şablonul Model – View – ViewModel a fost implementat doar în partea unde a fost utilizat WPF, în cadrul topului rezultatelor obţinute de utilizatorii aplicaţiei. Am utilizat acest şablon întrucât se mulează mult mai bine cu WPF.

Întrucât separarea între View şi partea de business din spatele View-ului se urmăreşte a fi cât mai mare, pentru a întruni cerinţele de cuplare slabă am decis asupra utilizării acestui şablon. Legătura dintre View şi ViewModel se face prin intermediul DataContext-ului, prin Binding-uri.

## Implementarea modelului Single Page Application (SPA)

Am decis asupra implementării acestui model întrucât se dovedeşte a fi mai optim din punct de vedere al resurselor necesare aplicaţiei. Astfel, avem o singură pagină principală, al cărei conţinut îl vom schimba prin afişarea unui conţinut personalizat, prin intermediul LandingPages-urilor.

Fiecare pagină a fost realizată într-un UserControl separat. Toate LandingPages-urile sunt instanţiate în pagina principală (MainPage), iar ulterior sunt încărcate în funcţie de acţiunile utilizatorului.

## Utilizarea delegaţilor

Un delegat reprezintă un tip de date care reprezintă referinţa unei metode cu anumiţi parametri şi un tip de returnat. Când instanţiem un delegat putem sa îi asociem instanţa cu orice metodă ce are acelaşi tip returnat, acelaşi număr de paramatri şi acelaşi tip de paramatri.

Am utilizat delegaţii în cadrul clasei QuestionHandler pentru a putea genera întrebări pentru orice tip de operaţie, fără a folosi cod duplicat.

# Testare şi Validare

În cadrul procesului de testare şi validare a aplicaţiei s-au utilizat tehnici de testarea manuală şi utilitarul ReSharper din Visual Studio 2012, pentru inspectarea codului. Testarea manuală a fost de tip „Exploratory” şi „Ad-hoc”, urmărind validarea algoritmilor din aplicaţie şi asigurarea asupra unei funcţionări corespunzătoare a aplicaţiei, prin urmărirea raspunsurilor din interfaţa aplicaţiei. Utilitarul pentru inspectarea codului a fost folosit pentru a obţine următoarele beneficii:

* Analizarea calităţii codului;
* Eliminarea erorilor;
* Modificarea sigură a codului;
* Traversarea instantă a întregii soluţii;
* Editarea facilă a codului;
* Transformarea instantă a codului;
* Menţinerea unui cod standardizat;

## Validarea datelor de intrare

În cazul câmpurilor care necesită numere drept date de intrare, se va efectua o validare şi în caz de incompatibilitate a datelor introduse, se va afişa un mesaj de eroare.

De asemenea, în cazul câmpurilor corespunzătoare datelor de înregistrare şi logare, se verifică o posibilă lipsă a conţinutului.

Astfel, în cazul necompletării sau completării cu date în format incorect, sunt afişate următoarele mesaje de eraore:

* În cazul în care utilizatorul nu completează atât câmpul destinat numelui de utilizator, cât şi cel destinat parolei, la apăsarea butonului „Login” este afişat următorul mesaj:

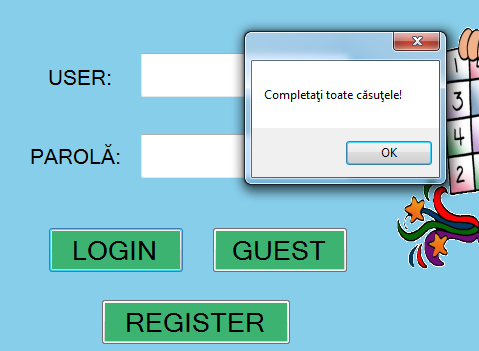


Figura 6. Mesaj eroare

* În cazul în care, la operaţia de înregistrare a unui utilizator nou, utilizatorul nu completează toate câmpurile (numele utilizatorului, parola, numele profesorului / bifare profesor) este afişat următorul mesaj:

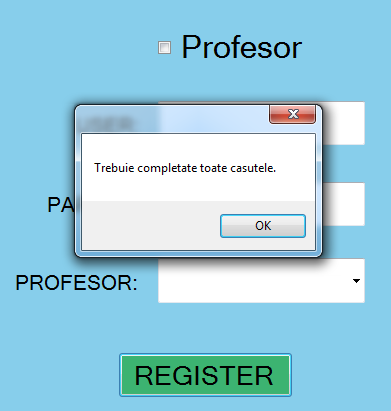


Figura 6. Mesaj eroare

* În cazul în care, în cursul operaţiei de logare a utilizatorului acesta îşi greşeşte parola, este afişat următorul mesaj de eroare:

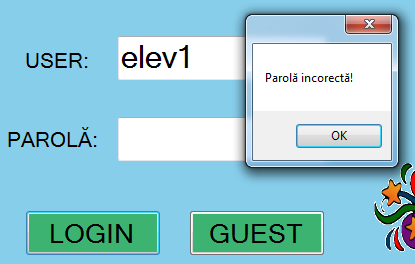


Figura 6. Mesaj eroare

* În cazul în care, în cadrul unui test, utilizatorul completează căsuţele destinate introducerii de numere cu litere sau alte caractere speciale, este afişat următorul mesaj:

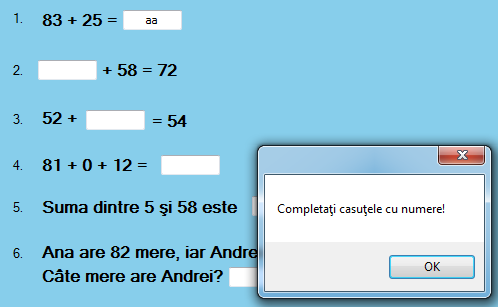


Figura 6. Mesaj eroare

* În cazul în care, în cadrul unui test, utilizatorul completează căsuţele destinate introducerii de numere cu litere sau alte caractere speciale, este afişat următorul mesaj:

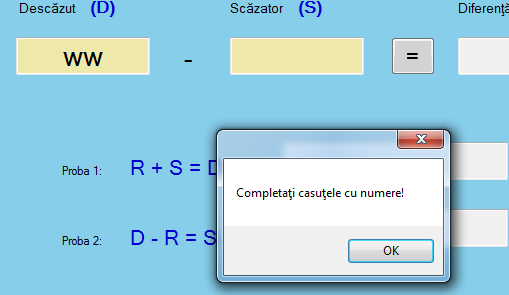


Figura 6. Mesaj eroare

* În cazul în care, în cadrul unui test, utilizatorul completează căsuţele destinate introducerii de numere cu litere sau alte caractere speciale, este afişat următorul mesaj:

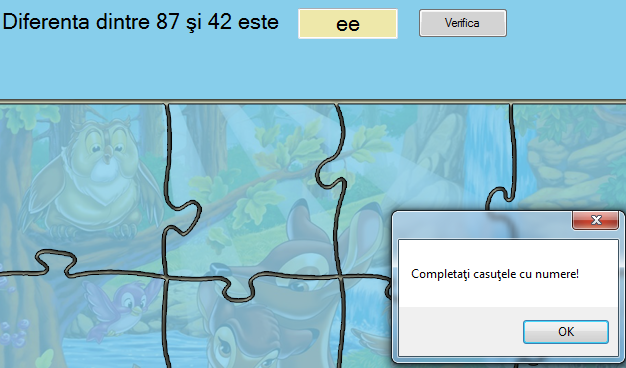


Figura 6. Mesaj eroare

* În cazul în care, în cadrul unui test, utilizatorul completează căsuţele destinate introducerii de numere cu litere sau alte caractere speciale, este afişat următorul mesaj:

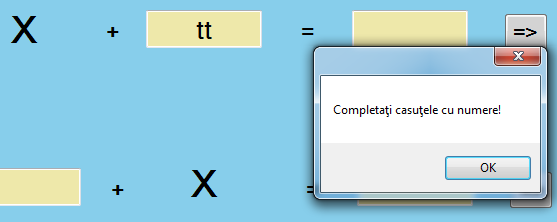


Figura 6. Mesaj eroare

# Manual de Instalare si Utilizare

## Ghid de instalare

### Resurse hardware necesare pentru utilizarea aplicaţiei

Pentru a utiliza în mod corespunzător aplicaţia „Învăţăm să calculăm” sunt necesare următoarele resurse hardware:

* Procesor cu unul sau mai multe nuclee, de preferat de minim 800MHz
* Memorie RAM de capacitate 512 MB sau mai mare
* Procesor video integrat sau dedicat, de cel puţin 32 MB
* Spaţiu de stocare liber pe disk de minim 300 MB
* Componente periferice (mouse, tastatură, difuzoare)

### Resurse software necesare pentru utilizarea aplicaţiei

Pentru a utiliza în mod corespunzător aplicaţia „Învăţăm să calculăm” sunt necesare următoarele resurse software:

* Sistem de operare Windows XP / Vista / 7 / 8 / 10
* Utilitar de management al bazelor de date SQL Server Management Studio 2012
* .Net Framework 4.0 sau mai nou
* Visual Studio 2010 sau mai nou
* Windows Media Player

### Manual de instalare al aplicaţiei

Pentru instalarea aplicaţiei trebuie urmaţi următorii paşi:

* Rulare SQL Server Management Studio 2012
* Importare bază de date
* Rulare Visual Studio 2010 sau mai nou
* Importare proiect Server
* Importare proiect Client
* Rulare proiect Server
* Rulare proiect Client

## Manualul de utilizare al aplicaţiei

### Înregistrarea unui utilizator nou

Pentru a înregistra un nou utilizator, trebuie urmaţi următorii paşi:

* Rularea aplicaţiei

1.PNG

Figura 7. Executabil

* Apăsarea butonului „Register”



Figura 7. Buton Register

* Selectare tip utilizator prin marcarea căsuţei „Profesor” în cazul utilizatorilor de tip „Profesor” (Implicit, utilizatorul este de tip „Elev”)

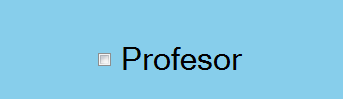


Figura 7. Checkbox Profesor

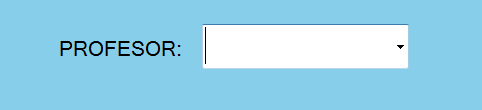


Figura 7. DropDown profesori

* Completare câmp „User” cu numele de utilizator

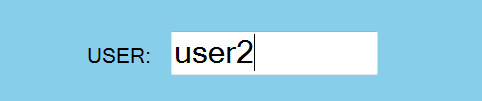


Figura 7. Câmp nume utilizator

* Completare câmp „Parolă” cu parola utilizatorului

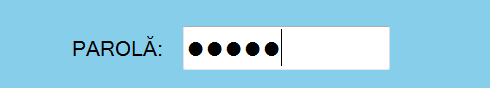


Figura 7. Câmp parolă

* Deschidere lista „Profesor” si selectare profesor, în cazul utilizatorilor de tip „Elev”



Figura 7. DropDown profesori

### Autentificarea unui utilizator

Pentru a autentifica un utilizator nou, trebuie urmaţi următorii paşi:

* Rularea aplicaţiei

1.PNG

Figura 7. Executabil

* Completarea numelui de utilizator

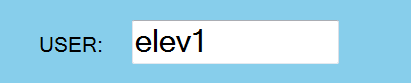


Figura 7. Câmp nume utilizator

* Completarea parolei

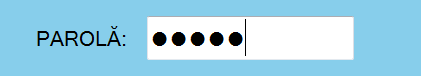


Figura 7. Câmp parolă

* Apăsarea butonului „Login”



Figura 7. Buton Login

### Utilizarea aplicaţiei în calitate de utilizator tip „Guest”

Pentru a utiliza aplicaţia în calitate de utilizator tip „Guest” trebuie urmaţi următorii paşi:

* Rularea aplicaţiei

1.PNG

Figura 7. Executabil

* Apăsarea butonului „Guest”



Figura 7. Buton Guest

În cazul utilizării aplicaţiei în calitate de utilizator tip „Guest”, butoanele „Top” şi „Premii” vor fi dezactivate:



Figura 7. Meniu aplicaţie

### Pornirea şi oprirea melodiilor aplicaţiei

Pentru a administra (porni / opri) melodiile din cadrul aplicaţiei este necesară apăsarea celor doua butoane din stânga butonului „Ieşire”. Astfel, paşii necesari pentru a realiza acestea sunt:

* Rularea aplicaţiei
* Autentificarea / Accesarea ca şi „Guest”
* Apăsarea butoanelor din stânga butonului „Ieşire”

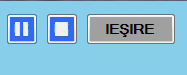


Figura 7. Butoane melodie şi părăsire aplicaţie

m1.PNG

Figura 7. Butoane melodie

m3.PNG

Figura 7. Butoane melodie

### Selectarea unei operaţii

Pentru a selecta o operaţie matematică trebuie să pargcurgem următorii paşi:

* Rularea aplicaţiei
* Autentificare / Accesare în calitate de utilizator „Guest”
* Apăsarea unuia dintre butoanele „Adunare”, „Scădere”, „Înmulţire”, „Împărţire”, în funcţie de dorinţa utilizatorului

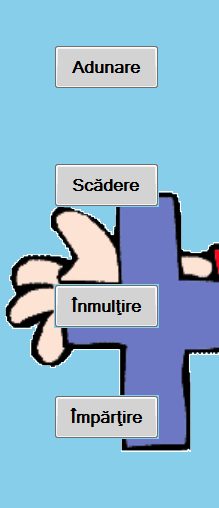


Figura 7. Butoane operaţii

o2.PNG

Figura 7. Buton adunare

o3.PNG

Figura 7. Buton scădere

o4.PNG

Figura 7. Buton înmulţire

o5.PNG

Figura 7. Buton impărţire

### Selectare test nou, în funcţie de operaţia selectată anterior

Pentru a începe un test, după urmarea paşilor descrişi la punctul anterior, trebuie urmaţi următorii paşi:

* Apăsare buton „Test”

t1.PNG

Figura 7. Buton test

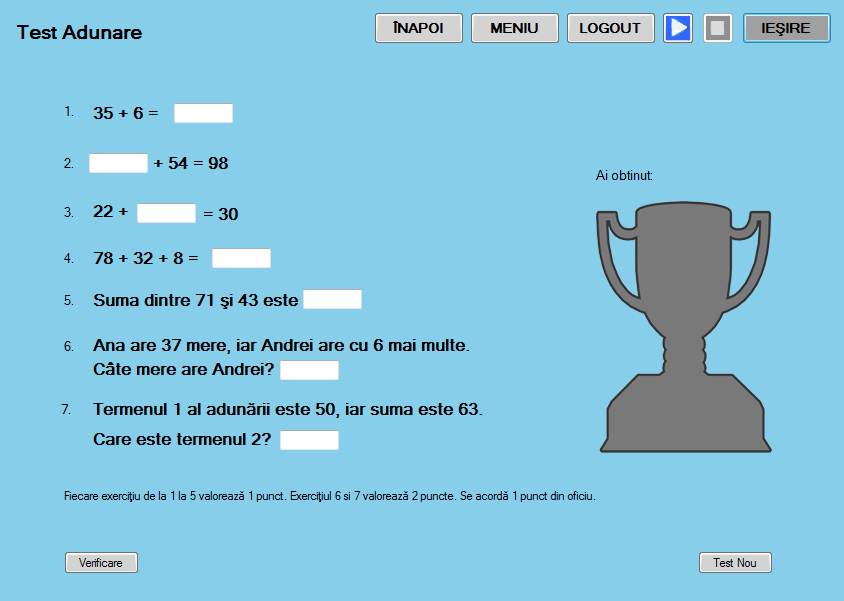


Figura 7. Test

Pentru a verifica rezultatele introduse, după paşii descrişi anterior, trebuie apăsat butonul „Verificare”



Figura 7. Buton verificare test

După apăsarea butonului verificare, sunt afişate etichete ce comunică modul în care au fost date răspunsurile (corect / greşit);



Figura 7. Verificare test

Pentru a începe un test nou, trebuie apăsat butonul „Test Nou”

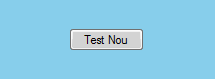


Figura 7. Buton test nou

### Selectarea unei ecuaţii, în funcţie de operaţia selectată anterior

Pentru a selecta o ecuaţie, dupa paşii corespunzători selectării unei operaţii, trebuie apăsat butonul „Ecuaţii”

e1.PNG

Figura 7. Buton ecuaţii

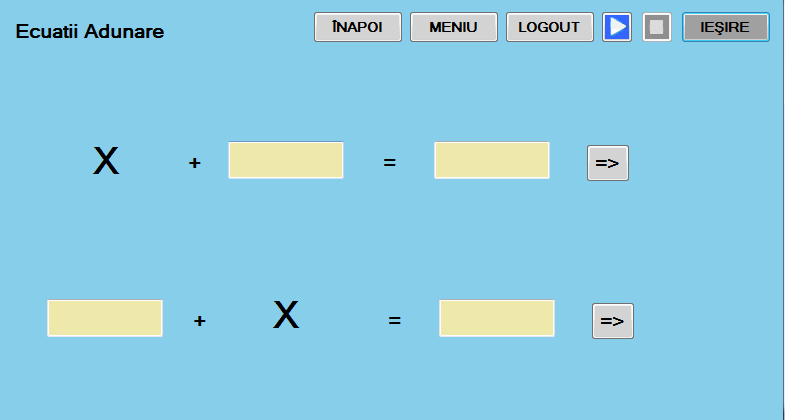


Figura 7. Ecuaţii adunare

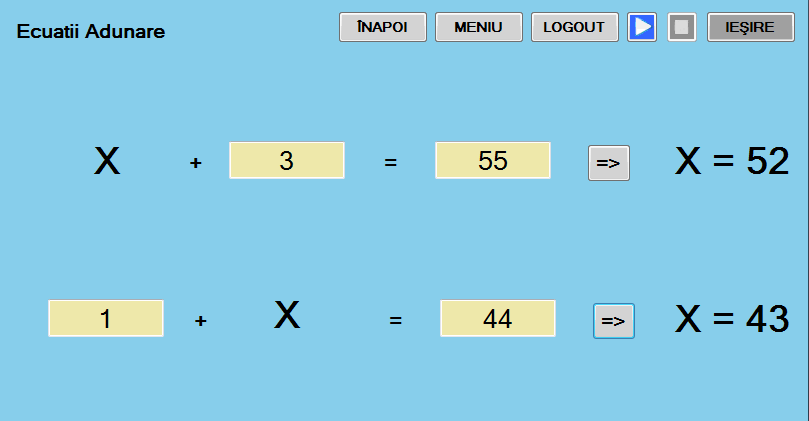


Figura 7. Ecuaţii adunare

### Selectarea unui joc tip „Puzzle”

Pentru a selecta un joc tip „Puzzle” sunt necesari următorii paşi:

* Rularea aplicaţiei
* Autentificare / Accesare în calitate de utilizator tip „Guest”
* Apăsarea unui buton „Puzzle adunare”, „Puzzle scădere”, „Puzzle înmulţire”, „Puzzle împărţire”

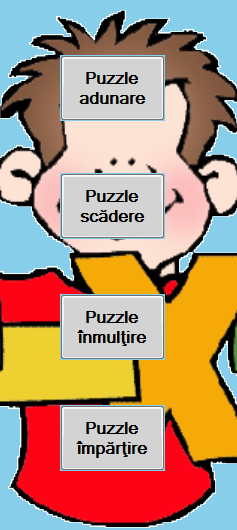


Figura 7. Butoane puzzle

p2.PNG

Figura 7. Puzzle adunare

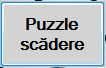


Figura 7. Puzzle scădere

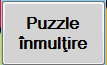


Figura 7. Puzzle înmulţire

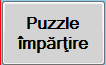


Figura 7. Puzzle împărţire

* Selectarea dimensiunii puzzle-ului (Puzzle Mic / Puzzle Mare)



Figura 7. Butoane selectare tip puzzle

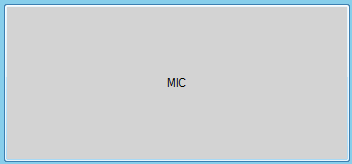


Figura 7. Buton puzzle mic



Figura 7. Buton puzzle mare

* Selectarea imaginii ce va reprezenta forma finală a puzzle-ului rezolvat

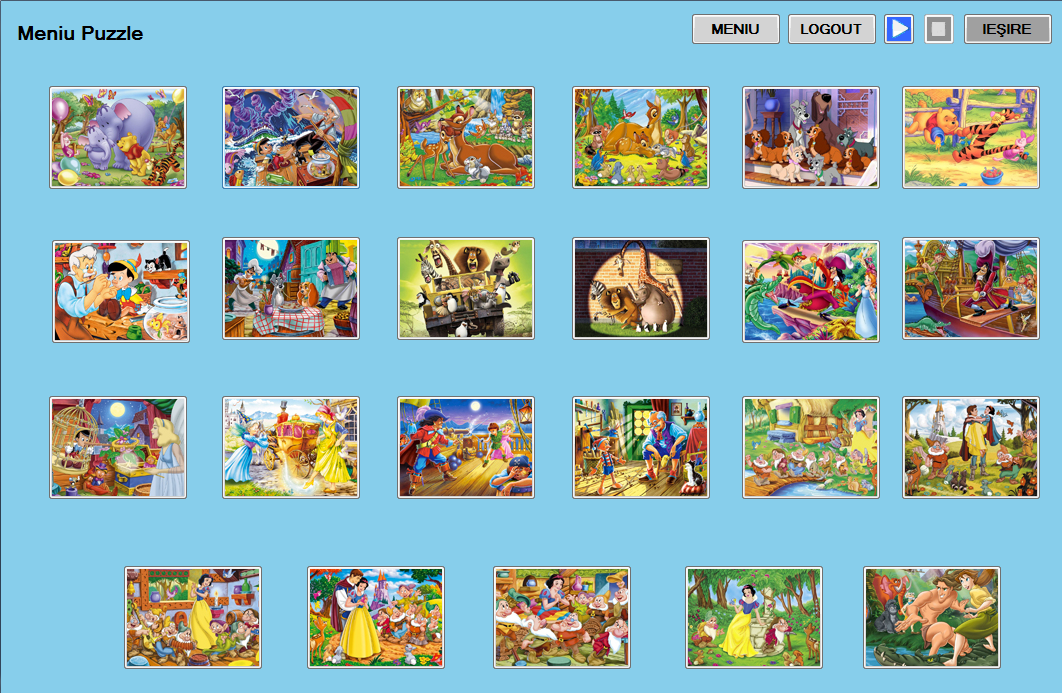


Figura 7. Imagini puzzle

### Jocul „Puzzle”

Pentru a juca un joc de tip „Puzzle”, după urmarea paşilor descrişi la punctul anterior, se introduce rezultatul în căsuţa din dreapta întrebării, iar apoi se apasă butonul „Verifica”



Figura 7. Joc Puzzle

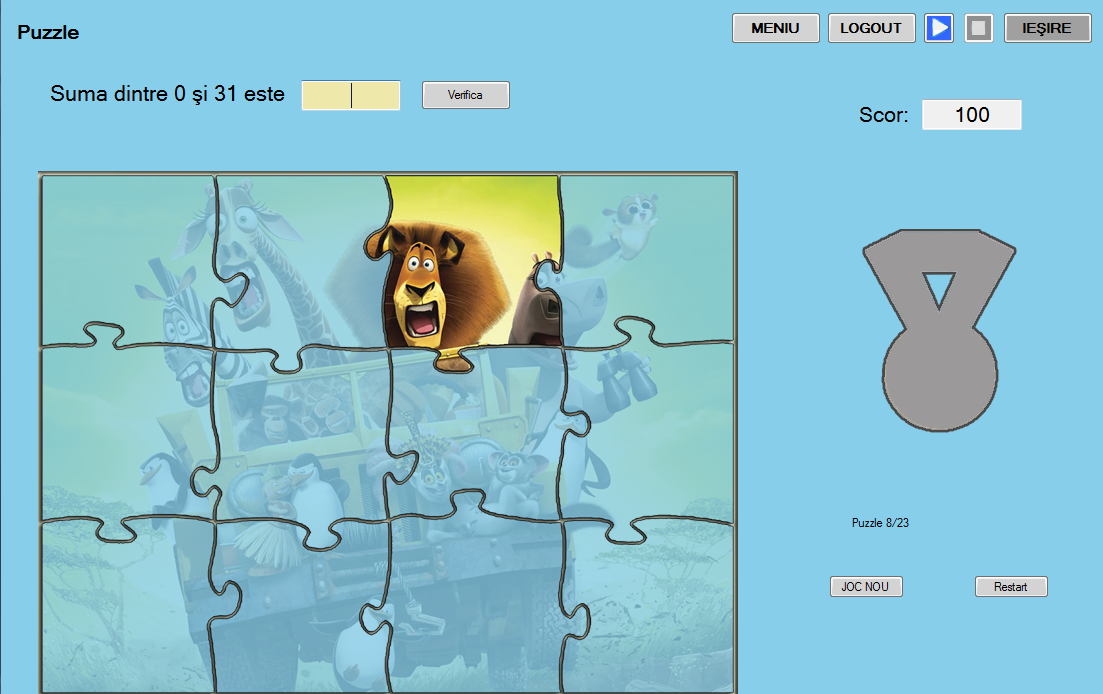


Figura 7. Joc Puzzle

În cazul introducerii unui răspuns greşit, scorul este decrementat cu 50 de unităţi

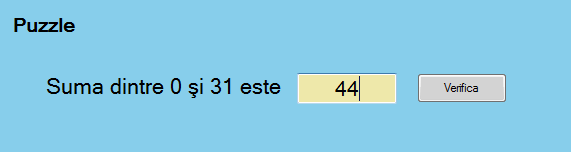


Figura 7. Răspuns puzzle

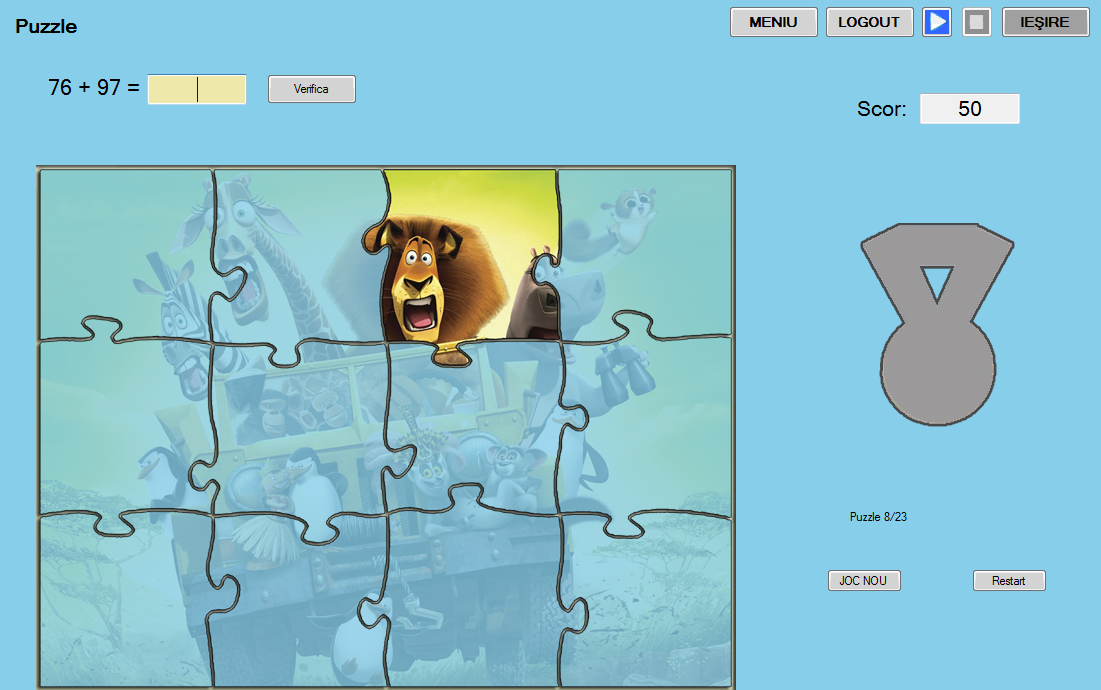


Figura 7. Puzzle

Pentru a reîncepe acelaşi puzzle, trebuie apăsat butonul „Restart”:



Figura 7. Restart puzzle

Pentru a selecta o dimensiune nouă şi un puzzle nou, trebuie apăsat butonul „JOC NOU”

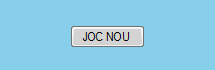


Figura 7. Puzzle nou

### Selectarea unui joc tip „Spânzurătoare”

Pentru a selecta un joc tip „Spânzurătoare” trebuie urmaţi următorii paşi:

* Rularea aplicaţiei
* Autentificarea / Accesarea aplicaţiei în calitate de utilizator tip „Guest”
* Apăsarea butonului „Spanzuratoare”

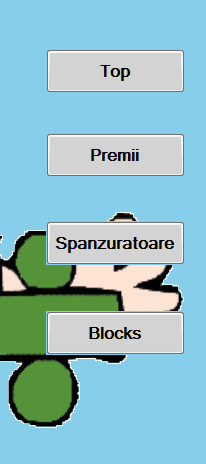


Figura 7. Meniu



Figura 7. Buton Spânzurătoare

După apăsarea butonului „Spânzurătoare” este afişată următoarea pagină:

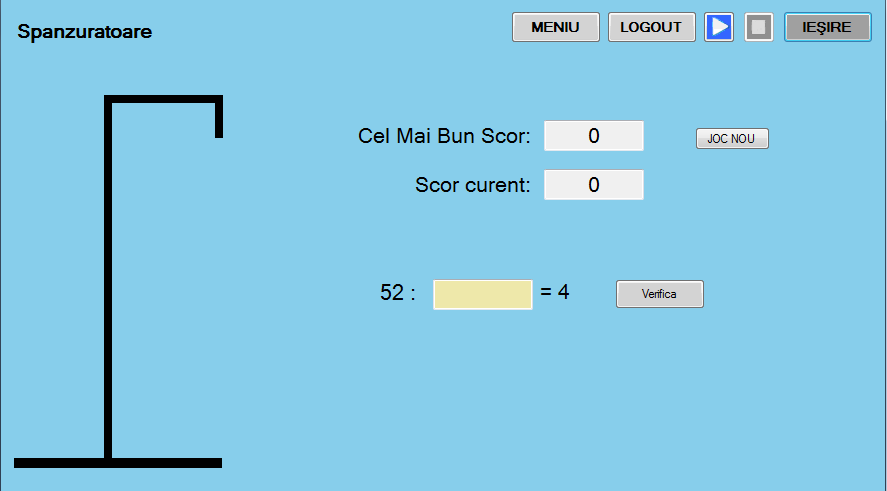


Figura 7. Joc Spânzurătoare

### Jocul tip „Spânzurătoare”

Pentru a juca un joc tip „Spânzurătoare”, după paşii descrişi în cadrul punctului anterior, trebuie compeltată căsuţa pentru rezultat şi apăsat butonul „Verifica”

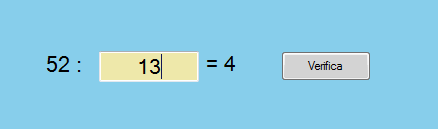


Figura 7. Joc Spânzurătoare

În cazul introducerii unui rezultat greşit, întrebarea ramâne pentru o nouă încercare, scorul rămâne neschimbat, însă imaginea se completează:



Figura 7. Joc Spânzurătoare

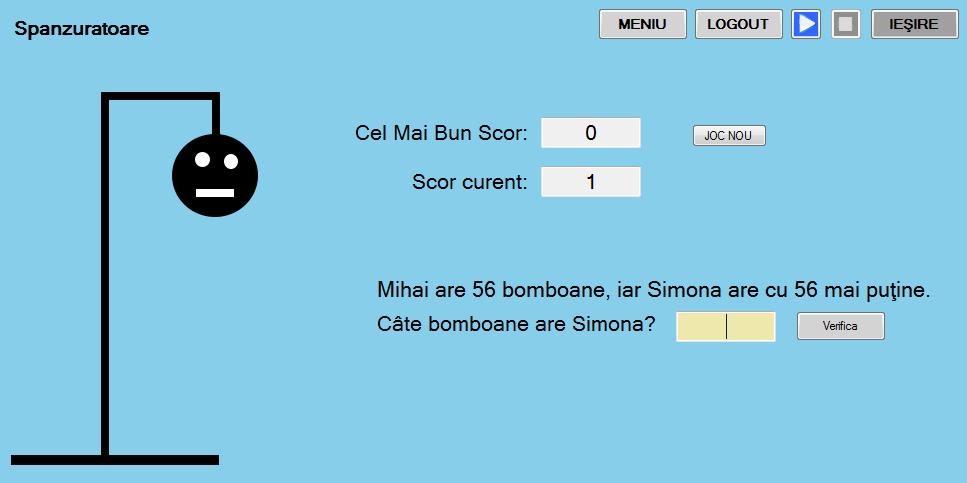


Figura 7. Joc Spânzurătoare

### Selectarea unui joc tip „Blocks”

Pentru a selecta un joc tip „Blocks” trebuie executaţi următorii paşi:

* Rularea aplicaţiei
* Autentificarea / Accesarea aplicaţiei cu un utilizator tip „Guest”
* Apăsarea butonului „Blocks” din meniul aplicaţiei

b1.PNG

Figura 7. Buton Blocks

### Jocul tip „Blocks”

Pentru a juca un joc de tip „Blocks”, după paşii descrişi la punctul anterior, se vor întoarce căsuţele prin Click, iar în momentul în care vor fi vizibile două căsuţe cu acelaşi rezultat al operaţiei afişate acestea vor dispărea:

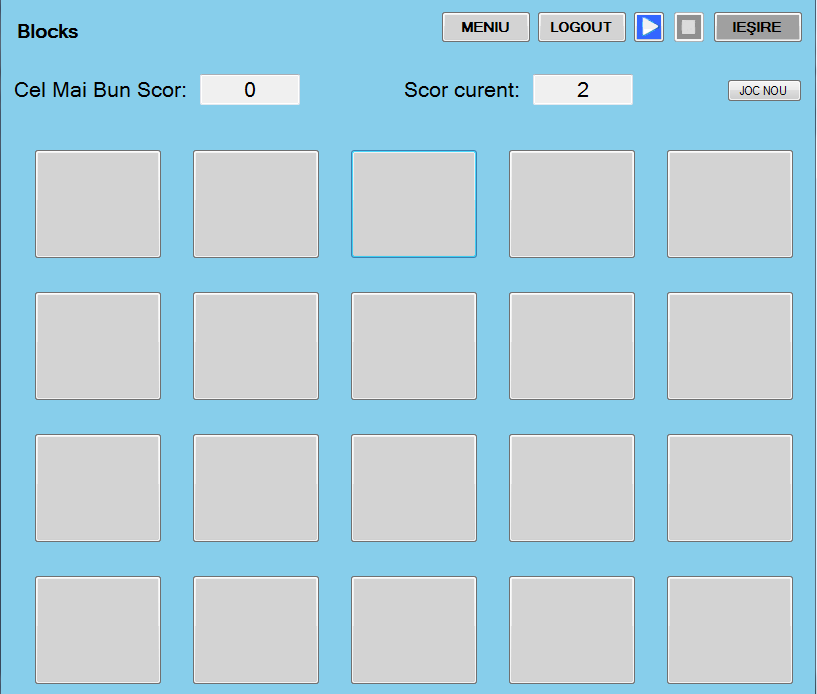


Figura 7. Joc Blocks

După întoarcerea a două căsuţe se aşteaptă un interval de o secundă până aplicaţia va primi o nouă comandă (click mouse). Acest interval de timp a fost setat cu scopul de a-i da utilizatorului un interval de timp suficient pentru a reţine conţinutul căsuţelor;

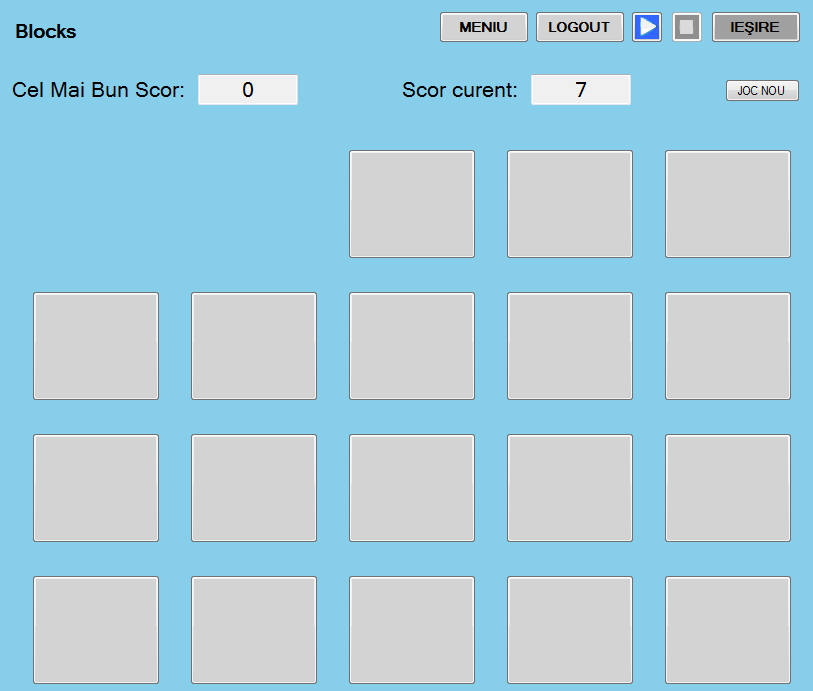


Figura 7. Joc Blocks

### Vizualizarea topului rezultatelor

Pentru a vizualiza topul rezultatelor obţinute de utilizatorii aplicaţiei, trebuie urmaţi următorii paşi:

* Rularea aplicaţiei
* Auentificarea cu un utilizator de tip „Elev” sau „Profesor”
* Apăsarea butonului „Top” din meniul aplicaţiei



Figura 7. Buton Top

### Vizualizarea premiilor obţinute de utilizatorul curent al aplicaţiei

Pentru a vizualiza premiile obţinute de utilizatorul curent al aplicaţiei este necesară realizarea următorilor paşi:

* Rularea aplicaţiei
* Autentificarea cu un utilizator de tip „Elev” sau „Profesor”
* Apăsarea butonul „Premii” din meniul aplicaţiei



Figura 7. Buton Premii

După apăsarea butonului „Premii” se va afişa pagina din figura următoare. Această pagină va conţine imagini ale tuturor tipurilor de cupe şi medalii pe care un utilizator le poate obţine în cadrul testelor şi al jocurilor şi numărul acestora, individual.

De asemenea, pagina va conţine şi o situaţie generală a numărului de cupe şi de medalii obţinute de utilizatorul curent şi valoarea acestora, exprimată printr-un scor.

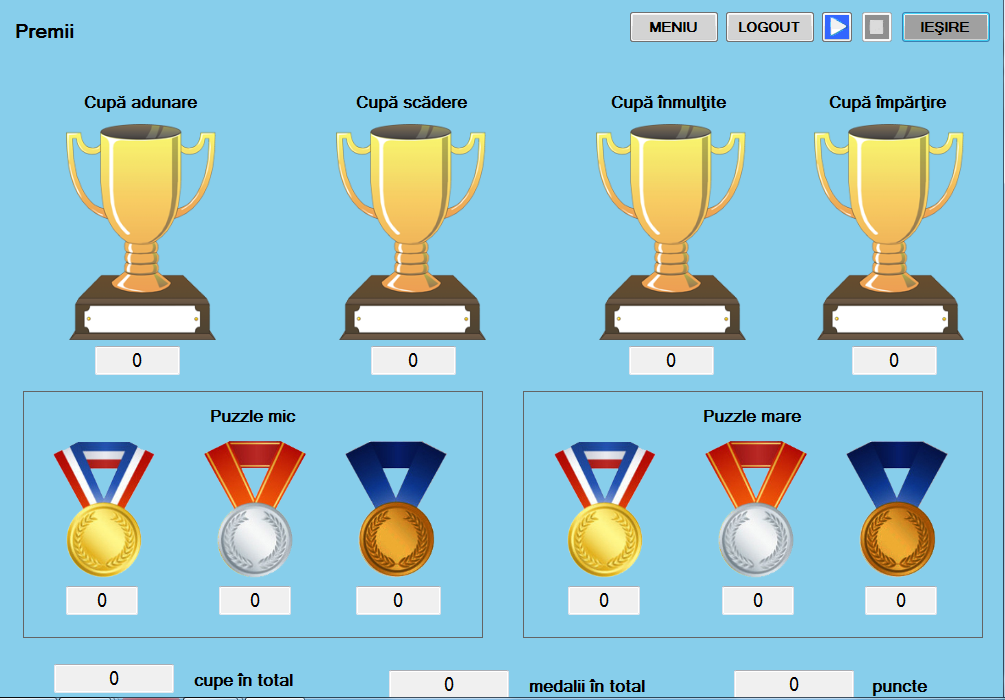


Figura 7. Premii

### Administrarea elevilor de către utilizatorii de tip „Profesor”

Pentru a administra elevii asignaţi unui profesor este necesară efectuarea următorilor paşi:

* Rularea aplicaţiei
* Autentificarea unui utilizator de tip „Profesor”
* Apăsarea butonului „Administrare Elevi” din meniul aplicaţiei

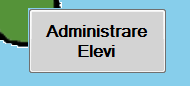


Figura 7. Buton Administrare Elevi

După apăsarea butonului „Administrare Elevi” se va afişa următoarea pagină. Aceasta va conţine un tabel cu toţi elevii asignaţi profesorului curent şi butoane pentru resetarea scorului acestora, precum şi pentru ştergerea conturilor acestora.

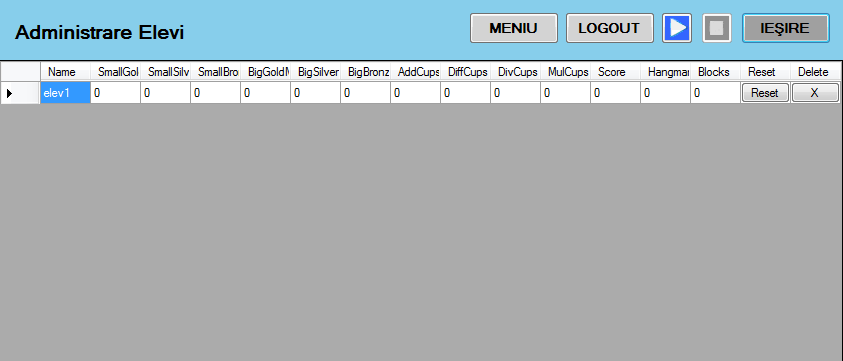


Figura 7. Administrare Elevi

* Apăsarea unuia dintre butoanele „Reset” sau „X”, pentru resetarea rezultatelor unui elev sau pentru a şterge contul unui elev

a4.PNG

Figura 7. Buton Reset rezultate elev

a5.PNG

Figura 7. Buton ştergere elev

### Logout utilizator

Pentru a deloga (Logout) un utilizator trebuie apăsat butonul „LOGOUT”

l1.PNG

Figura 7. Buton Logout

### Părăsirea aplicaţie

Pentru a părăsi aplicaţia trebuie apăsat butonul „IESIRE”

i1.PNG

Figura 7. Buton Părăsire aplicaţie

# Concluzii

## Realizari

Urmând fazele de alegere a unei teme, documentare în legătură cu domeniul aplicaţiei, proiectare a bazei de date, interfeţei cu utilizatorul şi a arhitecturii interne a aplicaţiei, implementare, testare şi fixare a problemelor descoperite în cadrul procesului de testare, am realizat o aplicaţie funcţională care atinge cerinţele stabilite iniţial.

Astfel, printre principalele provocări ale aplicaţiei se numără:

* Implementarea a 3 tipuri de utilizatori cu permisiuni specifice
* Implementarea de teste cu factori diferiţi, generaţi în mod aleator
* Implemantarea de aplicaţii tip “joc”, cu temă matematică, pentru captivarea utilizatorilor
* Realizarea de topuri în funcţie de rezultatele în cadrul testelor şi al jocurilor
* Realizarea unor jocuri de tip puzzle cu diferite dimensiuni şi imagini
* Realizarea unei interfeţe cu utilizatorul pe cât de uşor de folosit, pe atât de cuprinzătoare şi de captivantă

Am urmărit realizarea unei interfeţe estetice, cuprinzătoare şi uşor de utilizat, a cărei uzabilitate se bazează pe intuiţia utilizatorului, minimizând astfel timpul necesar pentru învăţarea aplicaţiei.

De asemenea, am insistat asupra proiectării şi implementării unor lecţii, teste şi jocuri captivante, uşor de înteles, cu un grav de dificultate suficient de uşor, însa suficient de cuprinzător pentru categoria de utilizatori căreia este destinată aplicaţia.

## Concluzii

Trecerea printr-un proces de propunere, documentare, proiectare, implementare şi corectare a unei aplicaţii cu un nivel de complexitate şi un număr de ore necesare implementării mai mare decât în cadrul activităţilor şi temelor din cadrul orelor de laborator a reprezentat o provocare ce m-a determinat să învăţ tehnici şi tehnologii cu care nu am avut tangenţă în cadrul facultăţii sau în cadrul locului de muncă.

Instruirea asistată de calculator poate fi privită sub mai multe aspecte. O dată, calculatorul este partener şi tutore în instruire, respectiv utilizat ca şi suport pentru soft-urile specifice educaţionale.

Pe de altă parte, calculatorul surprinde cele două paliere fundamentale ale educaţiei şi anume formarea şi informarea, ambele cuprinse în contextul acesta. În plus nu trebuie să omitem caracterul de autoeducaţie al acestui tip de instruire prin abordarea autoformării şi autoinformării. Caracterul activ al acestui tip de instruire face ca gradul de motivare a învăţării elevilor să fie unul ridicat datorită interesului lor pentru tot ce este nou şi dinamic.

Materialul care face obiectul acestei prezentări este util atât profesorului cât şi elevilor sau părinţilor acestora.

În cadrul lecţiilor profesorul poate utiliza programul în abordarea diferenţiată a elevilor, impunând grade diferite de dificultate. De asemenea, părinţii îl pot utiliza în verificarea temelor, iar elevii în autoevaluare şi corectare.

În concluzie, pot să afirm faptul că proiectarea, impelementarea şi corectarea ulterioară a acestei aplicaţii a reprezentat o provocare care a adus o considerabilă îmbunătăţire în colecţia de cunoştinţe din domeniul dezvoltării de aplicaţii software.

## Dezvoltari ulterioare

În categoria de dezvoltări ulterioare putem cuprinde următoarele îmbunătăţiri:

* Adăugarea unui utilizator de tip „Părinte”, cu permisiunea de a urmări rezultatele copilului / copiilor săi;
* Implementarea unui serviciu de recuperare a parolelor uitate;
* Implementarea unui sistem de mesagerie / forum, prin care utilizatorii de tip „Profesor”, „Elev” şi „Părinte” să comunice;
* Implementarea unui catalog care să conţină atât rezultatele obţinute la testele şi jocurile din cadrul aplicaţiei, cât şi notele obţinute la clasă de către elevi;
* Implementarea unui sistem de comunicare a rezultatelor obţinute de elevi către aceştia şi către părinţii acestora, prin intermediul unui sistem de mesagerie electronică şi prin SMS;
* Adăugarea de jocuri noi;
* Adăugarea de teste noi;
* Adăugarea de ilustraţii reprezentative în cadrul lecţiilor şi al ecuaţiilor;
* Realizarea unei variante a aplicaţiei destinată utilizării pe un dispozitiv mobil ce rulează sub un sistem de operarea Windows Phone, Android sau iOs;
* Implementarea unei versiuni cu tema limbilor străine, a geografiei, istoriei, sau a limbii române, în încercarea de a fixa, în cadrul ciclului primar, cunoşinţe necesare în cadrul ciclului gimnazial;
* Implementarea unor lecţii şi teste zilnice, scurte dar suficient de cuprinzătoare, pentru a menţine contactul utilizatorului cu aplicaţia şi cu materia studiată;
* Implementarea unei categorii de tutoriale, atât scrise, cât şi legături (linkuri) către lecţii video pe tema aleasă de utilizator;
* Implementarea unei transformări a punctelor obţinute în cadrul testelor şi jocurilor în puncte pe care utilizatorii le utilizează în cadrul diferitor jocuri (ex. FarmVille, Candy Crush etc.);

# Bibliografie

[1] The SOAP Protocol, <http://www.pearsonhighered.com/samplechapter/0672326418>

[2] Kevin Kline, Daniel Kline, “SQL in a Nutshell”, 2001

[3] Akshay Patel, „Learning WCF: Practical Implementation”, 2014

[4] Fabrice Marguerie, Steve Eichert and Jim Woole, „LINQ In Action”, 2008

[5] Joseph Albahari, „C# 4.0”, 2012

[6] Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra, „Head First Design Patterns”, 2004

[7] Pavan Podila, Kevin Hoffman, „WPF Control Development Unleashed”, 2009

[8] Dan Harkey, „Client/Server Survival Guide”, 1999

[9] Gary Hall, „Pro WPF and Silverlight MVVM”, 2010

[10] Gil Fink, „Pro Single Page Application Development”, 2014

[11] Julia Lerman, „Programming Entity Framework”, 2010