Documentul de Proiectare a Soluţiei Aplicaţiei Software

(Software Design Document)

Version 1.0

17 Mai, 2023

Aplicație de simulare teste de admitere AC

Echipa 1310A-1310B-1311A

Hriscu Cornelia-Ștefana

Huşman Carla Gabriela

Pîslariu Andreea

Facultatea de Automatică şi Calculatoare
Universitatea Tehnică „Gheorghe Asachi” din Iaşi

Cuprins

Cuprins 2
1. Scopul documentului 3
2. Conţinutul documentului 3
3. Modelul datelor 3
   3.1. Structuri de date globale 3
   3.2. Structuri de date de legătură 3
   3.3. Structuri de date temporare 4
   3.4. Formatul fişierelor utilizate 4
   3.5. Descrierea bazei de date 4
      3.5.1. Diagrama schemei bazei de date 4
      3.5.2. Descrierea tabelelor 5
4. Modelul arhitectural şi modelul componentelor 7
   4.1. Arhitectura sistemului 7
      4.1.1. Şabloane arhitecturale folosite 7
      4.1.2. Diagrama de arhitectură 7
   4.2. Descrierea componentelor 8
   4.3. Restricţiile de implementare 9
   4.4. Interacţiunea dintre componente 9
5. Modelul interfeţei cu utilizatorul 11
   5.1. Succesiunea interfeţelor 11
   5.2. Ferestrele aplicaţiei 12
      5.2.1. Fereastra Intro 12
      5.2.2. Fereastra de autentificare 13
      5.2.3. Fereastra de înregistrare 14
      5.2.4. Fereastra de setări 15
      5.2.5. Fereastra de parcurgere a testului
      5.2.6. Fereastra de vizualizare a rezultatelor 16
6. Elemente de testare 18
   6.1. Componente critice 18
   6.2. Alternative 18

1. Scopul documentului

Acest document descrie proiectarea soluției pentru o aplicație de simulare a testelor de admitere la Facultatea de Automatică și Calculatoare. Această aplicație va fi utilizată de elevii care doresc să se pregătească pentru testele de admitere și va permite utilizatorilor să practice probleme de matematică și informatică.

2. Conţinutul documentului

Documentul este format din patru secţiuni esenţiale:

- *Modelul datelor* – prezintă principalele structuri de date folosite, precum şi schema bazei de date
- *Modelul arhitectural şi modelul componentelor* – prezintă şabloanele arhitecturale folosite, arhitectura sistemului şi descrie componentele arhitecturii
- *Modelul interfeţei cu utilizatorul* – prezintă interfaţa cu utilizatorul şi succesiunea ferestrelor acesteia
- *Elemente de testare* – prezintă componentele critice şi alternative de proiectare a acestora.

3. Modelul datelor

3.1. Structuri de date globale

Aplicația are o instanță globală a clasei de conexiune la baza de date, implementată folosind sablonul de proiectare Singleton. Această instanță permite tuturor modulelor din aplicație să aibă acces la conexiunea la baza de date și să efectueze operațiuni CRUD pe tabelele respective. De asemenea, aplicația utilizează o structură de date globală pentru gestionarea stărilor quiz-ului, implementată folosind șablonul de proiectare Stare.

3.2. Structuri de date de legătură

Aplicația se conectează la o bază de date SQLite pentru a stoca și recupera date despre utilizatori și imagini pentru întrebările din test. Modulul GUI al aplicației comunica cu modulul de baze de date prin intermediul argumentelor ce conțin cheile primare din schema bazei de date (de exemplu, identificatorii de utilizatori sau de întrebări de test). Aplicația utilizează structuri de date de legătură pentru a gestiona aceste comunicări între module.

3.3. Structuri de date temporare

Aplicația nu utilizează structuri de date temporare care să presupună un consum semnificativ de resurse de memorie.

3.4. Formatul fişierelor utilizate

În ceea ce privește formatul fișierelor utilizate, aplicația de teste de simulare a admiterii utilizează imagini pentru a afișa întrebările de test, care sunt stocate direct în baza de date SQLite utilizând tipul de date BLOB. Astfel, imaginile sunt convertite și salvate în baza de date ca obiecte BLOB, ceea ce permite gestionarea lor eficientă prin intermediul sistemului de gestiune a bazei de date.

3.5. Descrierea bazei de date

3.5.1. Diagrama schemei bazei de date

Modelul bazei de date este format din următoarele tabele prezentate în figura de mai jos:

3.5.2. Descrierea tabelelor

Schema bazei de date cuprinde următoarele tabele:

- **Utilizatori –** conține informații despre utilizatori, precum numele, prenumele, numele de utilizator, și parola. Tipul de date al coloanelor Nume, Prenume, NumeUtilizator, Parola este de tip varchar și poate stoca până la 255 de caractere.

- **IntrebariMatematica & IntrebariInformatica** – conțin informații despre întrebările pentru testul de informatică și matematică, respectiv. Acestea conțin două coloane: **Intrebare** și **Raspuns**. Coloana **Intrebare** este de tip BLOB, care poate stoca date binare, cum ar fi imagini, iar coloana **Raspuns** este de tip integer și conține răspunsul corect la întrebare.

4. Modelul arhitectural şi modelul componentelor

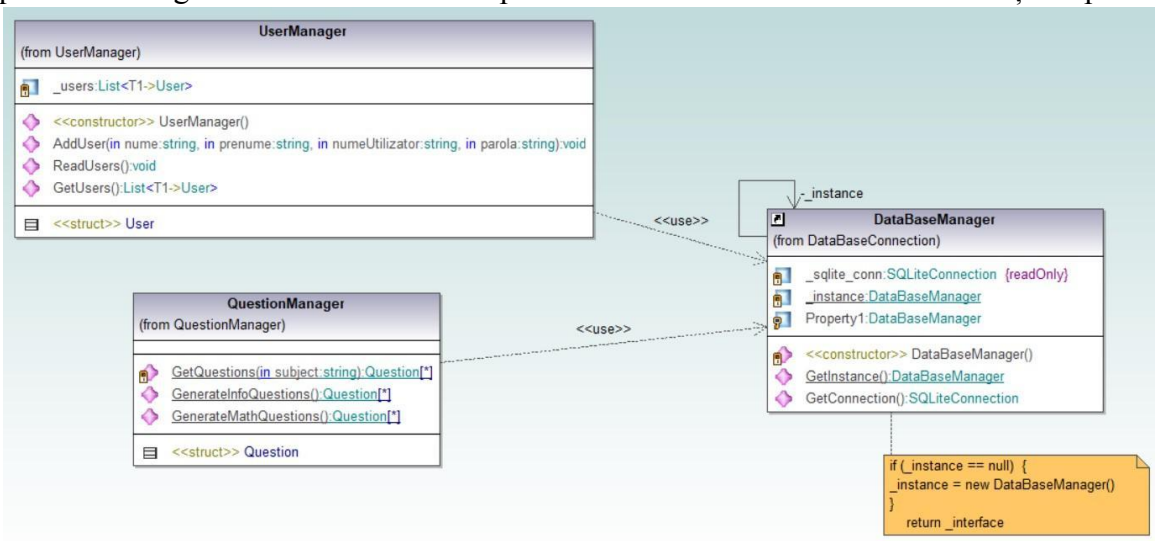
4.1. Arhitectura sistemului

4.1.1. Şabloane arhitecturale folosite

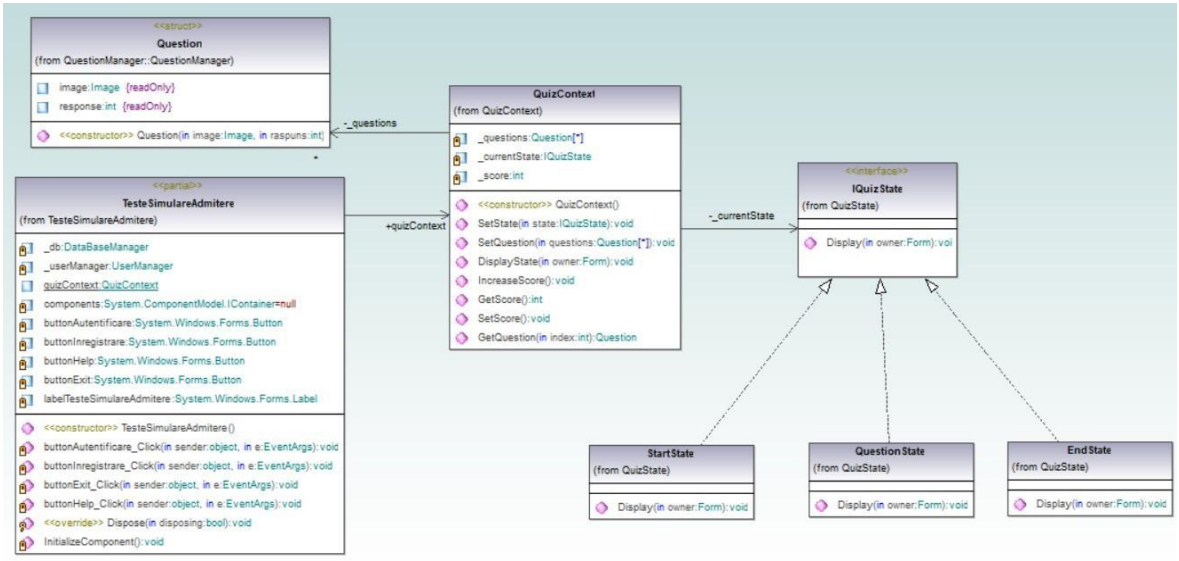
Aplicația este proiectată folosind modelul arhitectural Model-View-Controller (MVC) Această arhitectură este utilizată pentru a separa modelul de date, logica de afaceri și interfața cu utilizatorul în componente distincte, ușor de gestionat și de modificat. Modelul este responsabil pentru stocarea datelor, logica de afaceri și validarea datelor. View-ul este responsabil pentru afișarea datelor utilizatorului, precum și pentru colectarea și transmiterea datelor către controller. Controller-ul este responsabil pentru procesarea datelor, interacțiunea cu modelul și cu view-ul, precum și pentru gestionarea fluxului de date în aplicație.

În implementarea aplicației, s-au utilizat următoarele șabloane de proiectare:

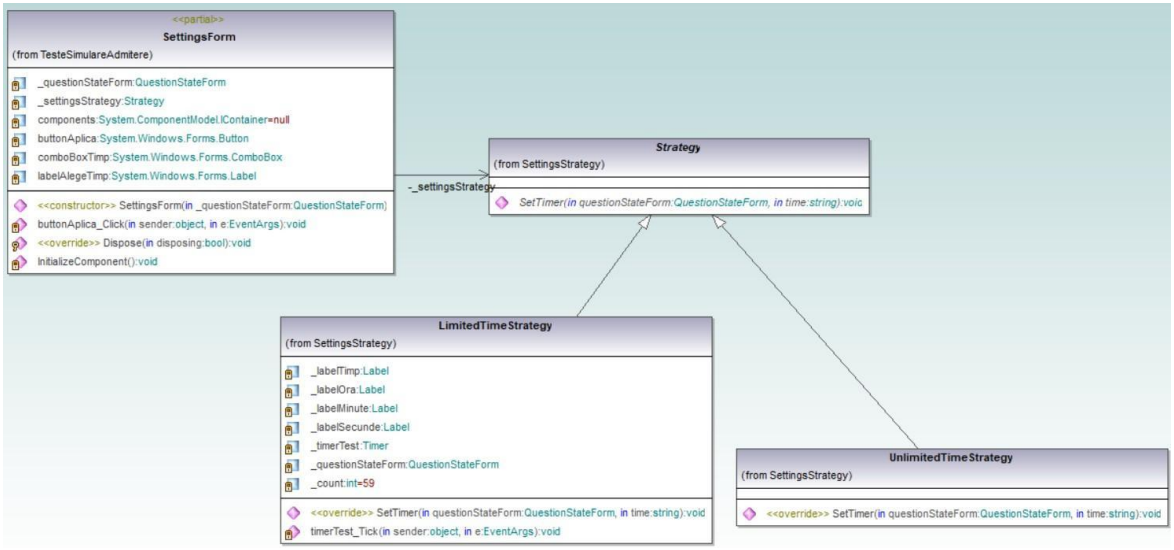
- ***Singleton***: Acest șablon arhitectural este utilizat pentru a asigura că există o singură instanță a unei clase în aplicație. În cazul acestei aplicații, Singleton este utilizat pentru a garanta că există o singură conexiune la baza de date, indiferent de numărul de utilizatori care accesează aplicația. Astfel, conexiunea la baza de date va fi mai eficientă și se evită problemele legate de deschiderea multiplelor conexiuni la baza de date în același timp.



- ***Şablonul de stare***:Acest șablon arhitectural este utilizat pentru a gestiona stările diferite ale quiz-ului. În cazul acestei aplicații, stările pot fi începutul quiz-ului, momentul în care se răspund la întrebări și încheierea quiz-ului. Fiecare stare poate avea comportamente specifice asociate cu ea, cum ar fi încărcarea întrebărilor în cazul stării de început, evaluarea răspunsurilor în cazul stării de quiz și afișarea rezultatelor în cazul stării de încheiere a quiz-ului.



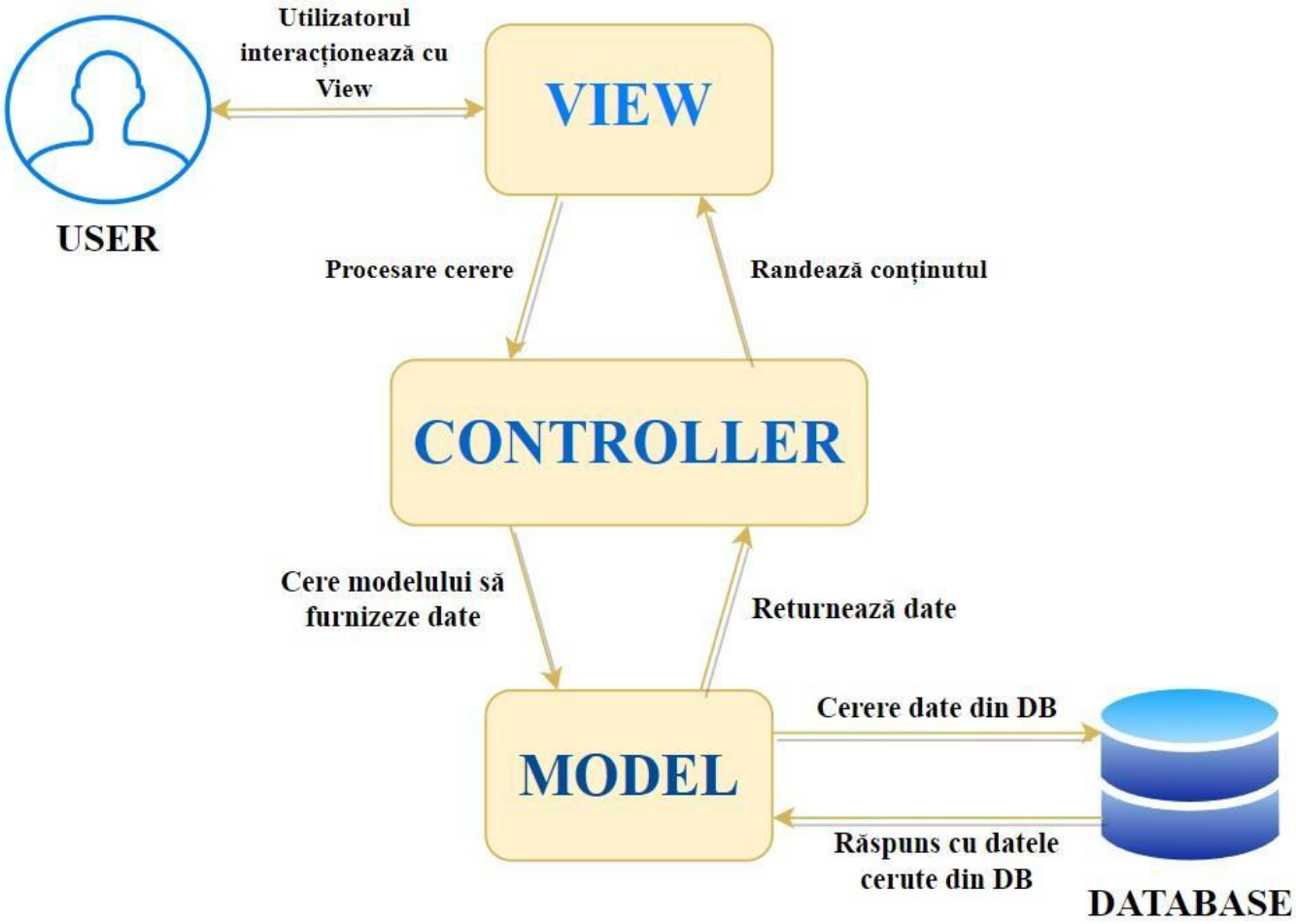
- ***Şablonul de strategie***: Acest şablon arhitectural este utilizat pentru a permite utilizatorului să aleagă modul în care se va desfăşura testul. În cazul acestei aplicaţii, utilizatorii pot alege între un test cu timp nelimitat sau un test cu timp limitat şi pot selecta durata testului. Această implementare ar putea implica crearea a două strategii diferite pentru desfăşurarea testului, una cu timp limitat şi una fără limită de timp, şi o metodă pentru a permite utilizatorului să selecteze una dintre acestea şi să introducă durata testului în cazul testului cu timp limitat.



Aceste şabloane arhitecturale sunt utilizate pentru a îmbunătăţi modularitatea, scalabilitatea şi extensibilitatea aplicaţiei, făcând-o mai uşor de dezvoltat şi de întreţinut pe termen lung.

4.1.2. **Diagrama de arhitectură**

Diagrama de arhitectură de mai jos descrie componentele arhitecturii aplicaţiei şi relaţiile de interacţiune dintre acestea.



4.2. Descrierea componentelor

Aplicaţia constă din următoarele module interconectate. Aceste module lucrează împreună pentru a implementa arhitectura MVC și pentru a asigura o separare clară între logica de business, interfața utilizatorului și manipularea datelor.

- **Modulul GUI** (Graphical User Interface)
  Acesta reprezintă componenta View din arhitectura MVC. Modulul GUI este responsabil de interfața grafică cu care utilizatorul interacționează. Acesta afișează informațiile către utilizator și primește acțiunile utilizatorului prin intermediul elementelor de interfață, cum ar fi butoane, câmpuri de text, etc. Modulul GUI nu conține logica de afaceri, ci doar se ocupă de prezentarea și interacțiunea cu utilizatorul.

- **Modulul de Logică**

  Acesta reprezintă componenta Controller din arhitectura MVC. Modulul de Logică conține logica de afaceri a aplicației. Acesta se ocupă de procesarea datelor și implementarea regulilor de business specifice. Modulul de Logică primește acțiunile utilizatorului de la modulul GUI și gestionează interacțiunea cu modulul de Baze de date și cu modulul GUI pentru a realiza operațiile necesare. În aplicația de simulare a testelor de admitere, modulul de Logică include logica pentru generarea întrebărilor de test, evaluarea răspunsurilor și calcularea rezultatelor.

- **Modulul de Baze de date**

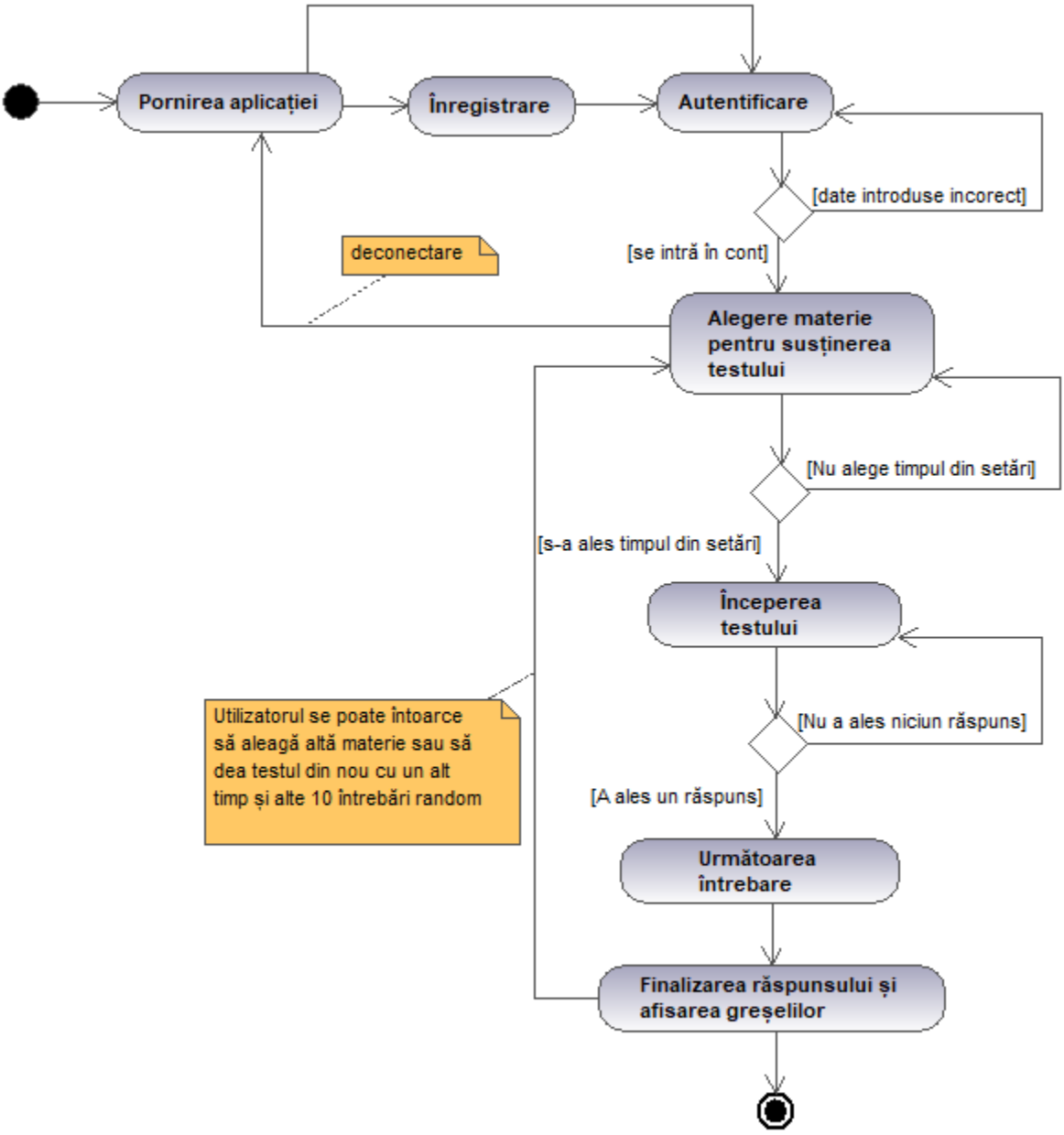
  Acesta reprezintă componenta Model din arhitectura MVC. Modulul de Baze de date gestionează stocarea și accesul la date. Acesta se ocupă de crearea, actualizarea și interogarea bazei de date utilizate în aplicație. Modulul de Baze de date poate include clase și metode care facilitează manipularea datelor și comunicarea cu sistemul de gestiune a bazei de date. În aplicația de simulare a testelor de admitere, modulul de Baze de date ar gestiona stocarea și recuperarea întrebărilor și răspunsurilor din baza de date SQLite.

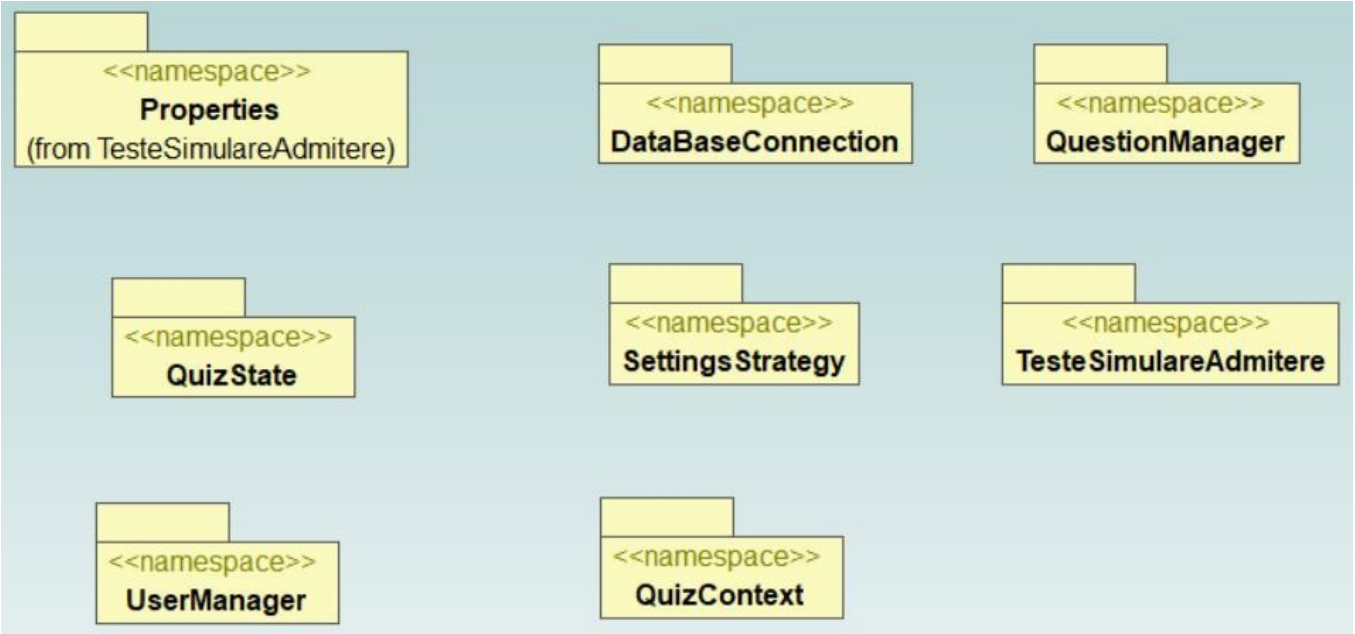
4.3. Restricţiile de implementare

- Va fi dezvoltat un modul de Baze de date care să faciliteze comunicarea cu baza de date SQLite utilizând DB Browser SQLite. Acest modul va include funcționalități pentru crearea și administrarea bazei de date, precum și pentru stocarea și recuperarea datelor relevante pentru simularea testelor de admitere.
- Modulul GUI, modulul de Logică și modulul de Baze de date vor fi implementate în limbajul de programare C#.
- Modulele vor fi dezvoltate ca proiecte de tip class library, generând DLL-uri separate.
- Se vor respecta principiile și standardele de programare specifice limbajului C# în implementarea modulelor.
- Se vor utiliza bibliotecile și framework-urile disponibile în C# pentru a obține funcționalitățile dorite în fiecare modul.

4.4. Interacţiunea dintre componente

Interacțiunea dintre componentele aplicației se bazează pe separarea responsabilităților și utilizarea șabloanelor de proiectare pentru a crea o arhitectură modulară, ușor de întreținut și extensibilă. DLL-urile separat implementate ca proiecte de tip class library oferă posibilitatea de a organiza și împărți codul în funcție de funcționalități și responsabilități specifice.



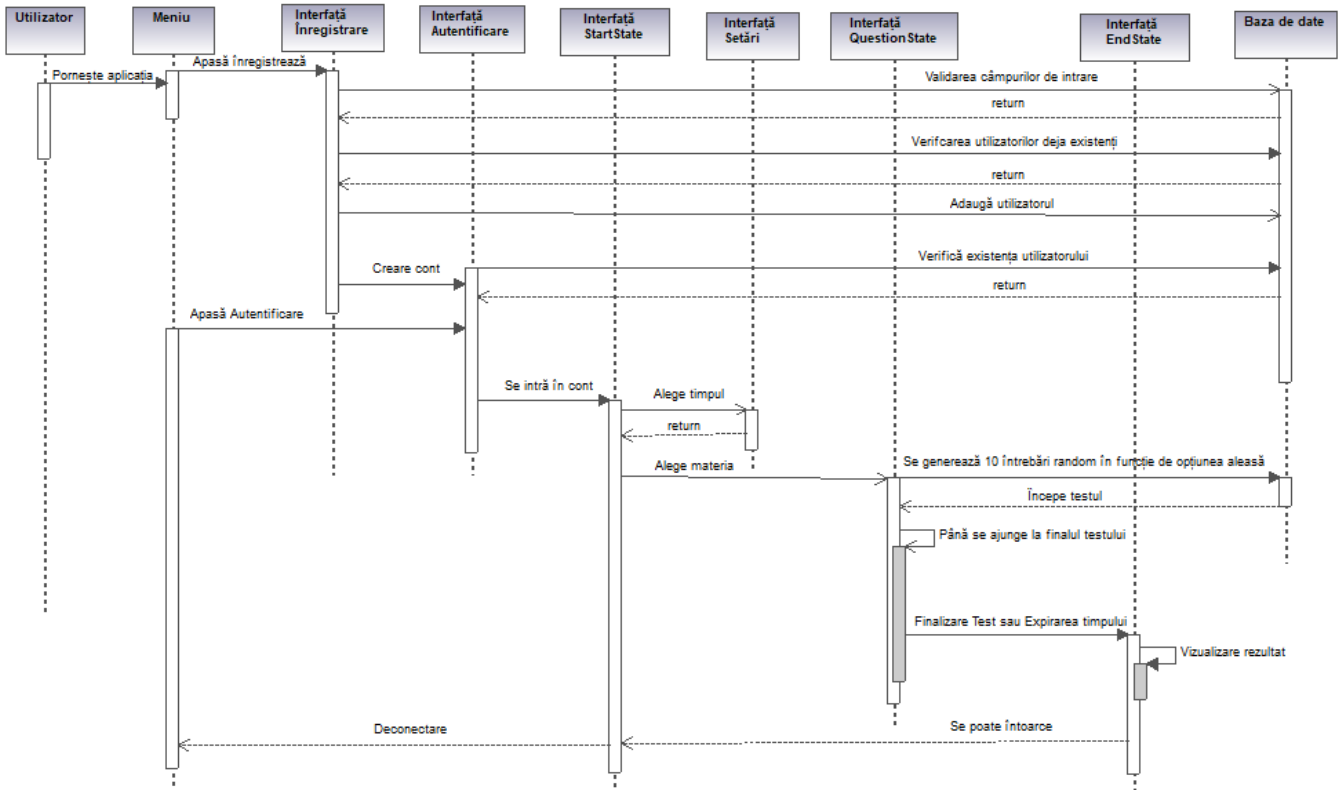


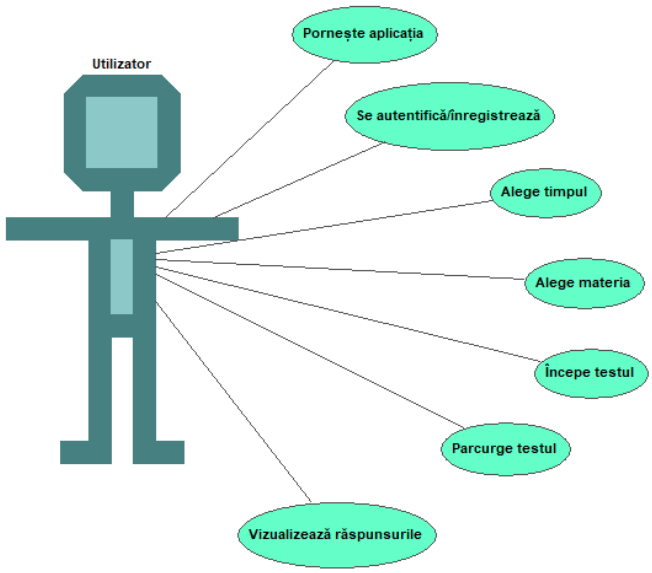
- Interacțiunea între client și modulul GUI:
    - Utilizatorul interacționează cu aplicația prin intermediul modulului GUI. Modulul GUI, implementat într-un proiect de tip Windows Form furnizează interfața grafică și elementele vizuale pentru interacțiunea cu utilizatorul.
- Interacțiunea dintre modulul GUI și modulul de Logică:
    - Modulul GUI, comunică cu modulul de Logică pentru a transmite acțiunile utilizatorului și a primi rezultatele procesării.
- Interacțiunea dintre modulul de Logică și modulul de Baze de date:
    - Modulul de Logică utilizează metode și funcționalități din modulul de Baze de date pentru a efectua operațiile de stocare, interogare și manipulare a datelor necesare pentru simularea testelor de admitere.
- Interacțiunea dintre modulele aplicației și DLL-urile separate:
    - DLL-urile separate, implementate ca proiecte de tip class library, conțin implementări ale unor sabloane de proiectare specifice, cum ar fi Singleton, State și Strategy.
    - Modulele aplicației utilizează DLL-urile separate prin includerea acestora ca referințe în proiectele respective.
    - Utilizarea DLL-urilor separate permite împărțirea funcționalităților și a logicii comune între module, facilitând reutilizarea și modularizarea codului.

5. Modelul interfeţei cu utilizatorul

5.1. Succesiunea interfeţelor

În cadrul modulului GUI, ferestrele aplicaţiei sunt afişate respectând un flux stabilit în conformitate cu standardele academice. Acest flux este descris grafic în figura de mai jos:





Generated by UModel www.altova.com

5.2. Ferestrele aplicaţiei

5.2.1. Fereastra Intro

Fereastra principală a aplicaţiei corespunde clasei TesteSimulareAdmitere.cs şi arată astfel:

//imagine

Această fereastră este panoul de comandă al utilizatorului în sensul că, de aici, utilizatorul alege ce acțiune dorește, determinând în felul acesta executarea unui flux specific.

- Butonul ***AUTENTIFICARE***
  - Permite autentificarea în aplicație pentru a accesa funcționalitățile disponibile.
- Butonul ***ÎNREGISTRARE***
  - Oferă posibilitatea de creare a unui cont nou pentru a beneficia de toate funcționalitățile disponibile.
- Butonul ***HELP***
  - Furnizează acces la documentația aplicației, care oferă instrucțiuni detaliate despre modul de utilizare a aplicației.
- Butonul ***EXIT***
  - Permite închiderea aplicației

5.2.2. Fereastra de autentificare

Prin apăsarea butonului ***AUTENTIFICARE*** se va închide fereastra principală și se va deschide o fereastră nouă care va permite autentificarea utilizatorului. După ce câmpurile nume utilizator și parolă au fost completate se apasă pe butonul de ***INTRĂ ÎN CONT***. Dacă contul există utilizatorul este redirecționat către o altă fereastră, însă dacă acesta este inexistent se va afișa mesajul "Nume utilizator sau parolă incorecte!" fiind recomandat ca utilizatorul să apese pe butonul ***ÎNAPOI*** pentru a se putea întoarce la pagina inițială și să își creeze un cont.

//imagine

5.2.3. Fereastra de înregistrare

Prin apăsarea butonului ***ÎNREGISTRARE*** se va închide fereastra principală și se va deschide o fereastră nouă care va permite utilizatorului să își creeze un cont prin completarea câmpurilor Nume, Prenume, Nume Utilizator, Parolă și apăsând ulterior pe butonul de ***CREAZĂ CONT***. Dacă numele de utilizator sau parola nu respectă condițiile impuse se va afișa mesajul "Nume de utilizator sau parole prea scurte. Minim 5 caractere.", iar dacă condițiile sunt respectate utilizatorul va fi redirecționat către pagina de autentificare. Dacă acesta se răzgândește pe parcursul creării contului poate apăsa pe butonul ***ÎNAPOI*** pentru a se putea întoarce la pagina inițială.

// imagine

5.2.4. Fereastra de setări

Prin apăsarea butonului ***SETĂRI*** se va deschide o fereastră care va permite utilizatorului să își seteze intervalul de timp dorit pentru efectuarea simulării. După alegerea opțiunii se va apăsa butonului ***APLICĂ*** ceea ce va avea ca efect închiderea ferestrei actuale.

///imagine

5.2.5. Fereastra de parcurgere a testului

După autentificarea cu succes se va deschide o nouă fereastră care va conține mesajul "Înainte de a începe, alege timpul pe care îl dorești pentru parcurgerea simulării.", utilizatorul fiind nevoit să apese pe butonul ***SETĂRI***. Odată setat timpul, se vor activa butoanele ***MATEMATICĂ*** și ***INFORMATICĂ*** pentru a putea alege materia la care se dorește a da simulare. Pe interfață se mai regăsește și butonul Deconectare care permite utilizatorului ieșirea din cont și revenirea la pagina principală.

Prin apăsarea butonului ***MATEMATICĂ*** sau ***INFORMATICĂ*** se va închide fereastra cu opțiuni și se va deschide una nouă care conține întrebarea și cele 4 variante de răspuns. În această fereastră există butonul care permite trecerea la următoare întrebare de abia după ce una din variante este selectată. În colț dreapta-sus utilizatorul poate verifica cât timp mai are al dispoziție pentru completarea testului.

///imagine

5.2.6. Fereastra de vizualizare a rezultatelor

La ultima întrebare butonul ***URMĂTOAREA ÎNTREBARE*** devine ***FINALIZEAZĂ TESTUL***!. Prin apăsarea butonului se va închide fereastra cu Întrebări și se va deschide o nouă fereastră în care va fi afișat rezultatul obținut, mesajul "Admis" sau "Respins", posibilitatea utilizatorului de a se verifică prin parcurgerea întrebărilor cu ajutorul butoanelor ***ÎNAINTE***/***ÎNAPOI*** și observarea concordanței dintre varianta de răspuns corectă și cea aleasă de acesta. Totodată mai există și un buton de întoarcere la fereastra cu materii pentru a susține un nou test.

///imagine

6. Elemente de testare

6.1. Componente critice

Componentele critice pentru aplicația de teste simulare admitere ar putea fi:

- Modulul de generare a întrebărilor de test și evaluarea răspunsurilor: Acest modul este esențial pentru a asigura corectitudinea și relevanța întrebărilor de test, precum și evaluarea corectă a răspunsurilor furnizate de utilizatori.
- Modulul de interacțiune cu baza de date: Acest modul gestionează stocarea și recuperarea întrebărilor, răspunsurilor și altor date relevante utilizate în aplicație. O funcționare corectă și eficientă a acestui modul este crucială pentru a asigura integritatea datelor și accesul rapid la informațiile necesare.

6.2. Alternative

Alternativele pentru a optimiza aceste componente critice în cadrul aplicației de teste simulare admitere ar putea fi:

- Utilizarea unui algoritm eficient de generare a întrebărilor de test: Integrarea unui algoritm specializat pentru generarea întrebărilor de test, care să asigure diversitate și relevanță în selecția întrebărilor. Acest lucru ar contribui la îmbunătățirea calității testelor și la creșterea gradului de acuratețe în evaluarea cunoștințelor utilizatorilor.
- Optimizarea interacțiunii cu baza de date: Utilizarea unor tehnici și strategii eficiente pentru a accelera interogările și manipularea datelor în baza de date. Aceasta poate include indexarea corectă a tabelelor, utilizarea cache-ului de date sau optimizarea interogărilor SQL. În plus, ar putea fi luate în considerare metode de denormalizare a datelor pentru a reduce complexitatea interogărilor și timpul de răspuns.