

---

# **Specificația cerințelor**

## **software**

**pentru**

## **Gestionarea unei**

## **biblioteci**

**Versiunea 1.0 aprobată**

Realizat de Aeloaiei Denisa-Valentina, Agavriloaei Marina,  
Bîrleanu Andreea, Stroici Andrei

**Universitatea Tehnică „Gheorghe Asachi” Iași**

**26.04.2025**

## Cuprins

<b>1. Introducere .....</b>	<b>1</b>
1.1 Scop .....	1
1.2 Convenții .....	1
1.3 Publicul țintă și sugestii de lectură .....	1
1.4 Scopul produsului .....	2
1.5 Referințe .....	2
<b>2. Descriere generală .....</b>	<b>2</b>
2.1 Perspectiva produsului .....	2
2.2 Funcțiile produsului .....	3
2.3 Clasele de utilizatori și caracteristicile acestora .....	4
2.4 Mediul de operare .....	4
2.5 Constrângeri de design și implementare .....	5
2.6 Documentația utilizatorului .....	5
2.7 Presupuneri și dependențe .....	5
<b>3. Cerințe ale interfețelor externe .....</b>	<b>6</b>
3.1 Interfața cu utilizatorul .....	6
3.2 Interfața hardware .....	9
3.3 Interfața software .....	9
3.4 Interfața de comunicații .....	10
<b>4. Caracteristicile sistemului .....</b>	<b>11</b>
4.1 Autentificare pe roluri .....	11
4.2 Înregistrare împrumut .....	11
4.3 Înregistrare retur .....	12
4.4 Autentificare client .....	12
4.5 Adăugare client nou .....	13
4.6 Ștergerea unei cărți .....	13
4.7 Adăugarea unei cărți .....	14
4.8 Afișare listă abonați problematici .....	14
4.9 Modificare status abonați .....	14
4.10 Căutare angajat după număr de telefon .....	15
4.11 Adăugare angajat .....	15
4.12 Ștergerea unui angajat .....	16
<b>5. Alte cerințe nefuncționale .....</b>	<b>16</b>
5.1 Cerințe de performanță .....	16
5.2 Cerințe de siguranță .....	17
5.3 Cerințe de securitate .....	18
5.4 Atribute ale calității software-ului .....	18
5.5 Alte cerințe nefuncționale .....	19
5.6 Reguli de business .....	19
<b>6. Alte cerințe .....</b>	<b>20</b>
6.1 Notificare clienți retur .....	20
6.2 Cerințe privind baza de date .....	20

## Istoric al reviziilor

Nume	Data	Motivul schimbărilor	Versiune

# 1. Introducere

## 1.1 Scop

Aplicația pe care am realizat-o se numește SmartLibrary și se află la prima versiune. Aceasta are rolul de a gestiona o bibliotecă, oferind o soluție digitală pentru înregistrarea împrumuturilor și retururilor de cărți, precum și pentru administrarea clienților.

Aplicația realizată utilizează o bază de date SQL pentru stocarea informațiilor referitoare la cărți, clienți și împrumuturi. Aceasta se adresează angajaților bibliotecii, care pot avea două roluri distincte: bibliotecar și administrator.

Bibliotecarul este responsabil de interacțiunea directă cu clienții, realizând operațiunile de împrumut și retur, verificarea disponibilității unei cărți, precum și înregistrarea de noi clienți. Administratorul gestionează catalogul de cărți și are autoritatea de a aplica sau ridica sancțiuni, în funcție de istoricul returnărilor. De asemenea, administratorul poate să adauge angajați sau să îi șteargă.

## 1.2 Convenții

Documentul urmărește următoarele convenții:

- Cerințele majore sunt grupate în funcție de funcționalitățile majore (de exemplu autentificare pe roluri, împrumut, retur) și sunt numerotate ierarhic
- Pentru fiecare funcționalitate sunt prezentate: descrierea, secvențele stimul/răspuns și cerințele funcționale
- Atributele evaluative (Prioritate, Beneficiu, Penalizare, Cost, Risc) sunt exprimate numeric pe o scală de la 1 la 10 și sunt scrise cu majuscule pentru evidențiere.
- Toate mesajele transmise între componente (client, server, interfață) urmează convenții proprii descrise în secțiunile de interfață.
- Codul sursă și exemplele de mesaje JSON sunt formate clar, cu indentare.
- Termenii tehnici folosiți frecvent (ex: „împrumut”, „abonat problematic”, „rol”) sunt explicați în Glosarul din Appendix A.
- Fontul folosit în document este Times, iar dimensiunea standard este 12. Titlurile sunt evidențiate prin îngroșare și dimensiune crescută.
- Elementele de cod sau de format special de fișiere (de exemplu JSON) sunt scrise folosind fontul „Courier New” cu dimensiunea standard de 12.

## 1.3 Publicul țintă și sugestii de lectură

Acest document se adresează atât utilizatorilor aplicației, cât și dezvoltatorilor, care vor folosi specificațiile pentru implementarea, testarea și întreținerea produsului software.

Utilizatorii finali sunt în principal angajații bibliotecii, mai exact bibliotecarii și administratorii bibliotecii, care vor interacționa zilnic cu aplicația pentru gestionarea activităților zilnice.

Dezvoltatorii aplicației, inclusiv programatorii, testerii, și membrii echipei de mentenanță, vor utiliza acest document pentru a înțelege cerințele funcționale și nefuncționale ale sistemului, precum și detaliile de interfațare și constrângerile de implementare.

Pentru o înțelegere eficientă, se recomandă parcurgerea în ordine a documentului, începând cu secțiunea **2 – Descriere generală**, urmată de secțiunea **4 – Caracteristicile sistemului**, pentru detaliile tehnice.

De asemenea, pentru cei care vor să înțeleagă mai bine modul cum este realizată aplicația sau pentru dezvoltatori recomandăm citirea standardului UML și „Design Patterns: Elements of Reusable Object-Oriented Software” de Gang of four.

## 1.4 Scopul produsului

În prezent se realizează un proces continuu de digitizare a domeniilor. Astfel, noi am venit cu o soluție pentru gestionarea activităților unei biblioteci.

Aplicația permite înregistrarea și actualizarea informațiilor despre cărți (dacă se adaugă sau se elimină o carte, sau se poate seta dacă o carte este un exemplar de sală sau nu), clienți (adăugarea de noi clienți, restricționarea accesului la cărți) și împrumuturi automatizând procesele de împrumut, retur și sancționare.

Prin utilizarea aplicației propuse se va automatiza și optimiza fluxurile de lucru realizate de angajații bibliotecii, dar se pot reduce și erorile manuale și se poate îmbunătăți accesul la informații. Produsul se adresează exclusiv personalului unei biblioteci, în interiorul căreia există două roluri principale cu atribute distincte: bibliotecar și administrator.

## 1.5 Referințe

[1] Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma et al., Addison-Wesley, 1994.

[2] OMG UML Specification, Version 2.5.1, December 2017,  
<https://www.omg.org/spec/UML/>.

[3] Oracle SQL Developer Data Modeler, Oracle Corporation, Version 24.3.1.351.0831,  
<https://www.oracle.com/database/sqldeveloper/technologies/sql-data-modeler/>.

[4] Microsoft Visual Studio Community Edition, Microsoft Corporation, Version 2022,  
<https://visualstudio.microsoft.com/vs/community/>.

# 2. Descriere generală

## 2.1 Perspectiva produsului

Aplicația realizată de noi este constituită pe o arhitectură care folosește paradigma client-server. Partea de client este compusă dintr-o interfață grafică care este disponibilă angajaților bibliotecii (bibliotecarii și administratorii), iar partea de server este cea care va notifica clienții când se apropie termenul de retur al cărții împrumutate prin trimiterea unui email clientului, permite accesul la baza de date de tip SQL (pentru adăugarea de cărți sau clienți, pentru eliminarea de cărți sau modificarea statusului unui client).

Produsul este construit ca un sistem modular, în care clientul comunică cu serverul prin intermediul protocoalelor de rețea (protocolul de nivel transport TCP), astfel clientul se poate conecta la server la distanță. Pentru o creștere a siguranței produsului, serverul va fi disponibil în rețeaua locală, pentru a evita accesul din exterior la acesta prin intermediul mecanismului de NAT care este disponibil pe orice router.

Soluția prezentată este independentă de alte sisteme externe, dar datorită folosirii arhitecturii client-server aceasta permite în viitor realizarea unei aplicații mobile disponibilă angajaților bibliotecii.

## 2.2 Funcțiile produsului

1. Autentificare utilizator
  - 1.1. Descriere: Utilizatorii se vor putea autentifica prin introducerea numelui de utilizator, a parolei și selectarea rolului. După introducerea acestor date, dacă datele sunt valide (nume utilizator, parolă și rol) acesta va fi redirecționat către meniul specific rolului pe care îl are.
  - 1.2. Cerințe funcționale:
    - 1.2.1. Sistemul va cripta mesajele trimise între client și server.
    - 1.2.2. Sistemul va verifica datele utilizatorului dacă sunt valide.
2. Gestionarea conturilor de utilizatori
  - 2.1. Descriere: Administratorii se vor ocupa cu modificarea statusului contului unui client (blocarea contului, impunerea unor restricții, sau ridicarea lor), iar bibliotecarul se va ocupa de adăugarea de noi clienți (deoarece el interacționează cu clienții în mod direct).
  - 2.2. Cerințe funcționale:
    - 2.2.1. Administratorul va putea vizualiza lista tuturor clienților problematici și va vedea motivul pentru care sunt pe acea listă.
    - 2.2.2. Administratorul va putea impune restricții sau le poate ridica asupra conturilor clienților.
    - 2.2.3. Bibliotecarul va putea adăuga noi clienți.
3. Gestionarea conturilor angajaților
  - 3.1. Descriere: Administratorul poate adăuga sau șterge conturile angajaților pentru a simula dinamica personalului bibliotecii.
  - 3.2. Cerințe funcționale:
    - 3.2.1. Adăugarea unui nou cont de angajat.
    - 3.2.2. Ștergerea contului unui angajat
4. Gestionarea cărților din bibliotecă:
  - 4.1. Descriere: Administratorul are posibilitatea de a adăuga cărți noi în bibliotecă, dar și să retragă anumite exemplare.
  - 4.2. Cerințe funcționale:
    - 4.2.1. Adăugarea unei cărți (administratorul poate să adauge noi cărți în bibliotecă, fie că este un nou exemplar pentru o carte existentă, fie un exemplar dintr-o carte care nu mai există în bibliotecă).
    - 4.2.2. Ștergerea unei cărți (administratorul poate lua decizia de a scoate din circulație, carte să nu mai fie disponibilă pentru împrumut, un anumit exemplar).
5. Realizarea unui împrumut
  - 5.1. Descriere: operație realizată de bibliotecar pentru că acesta este cel care interacționează direct cu clientul.
  - 5.2. Cerințe funcționale:
    - 5.2.1. Validarea datelor clientului pentru a se garanta că acesta are un cont existent la biblioteca noastră.
    - 5.2.2. Verificarea statusului contului clientului pentru a anunța clientul dacă are contul restricționat sau blocat.
    - 5.2.3. Dacă nu are contul restricționat atunci se va selecta cartea pe care vrea să o împrumute. Dacă clientul are contul restricționat va fi informat că acesta poate împrumuta cartea doar în sala de lectură. Dacă clientul o poate împrumuta acasă, se va seta o dată de retur.
6. Realizarea returului:

- 6.1. Descriere: operațiunea este realizată de bibliotecar deoarece acesta va interacționa direct cu clientul.
- 6.2. Cerințe funcționale:
  - 6.2.1. Validare date client, pentru a garant că acesta are un cont existent la biblioteca gestionată.
  - 6.2.2. Marcare carte ca fiind disponibilă, iar dacă clientul a întârziat cu returul atunci clientul va fi anunțat.
- 7. Verificarea unei cărți
  - 7.1. Descriere: Operație realizată de bibliotecar prin intermediul căreia determină disponibilitatea unei cărți.
  - 7.2. Cerințe funcționale:
    - 7.2.1. Introducerea informațiilor despre carte, chiar dacă sunt parțiale
    - 7.2.2. Realizarea unei căutări după câmpurile introduse în baza de date și returnarea rezultatelor

## 2.3 Clasele de utilizatori și caracteristicile acestora

Aplicația realizată se adresează la două clase de utilizatori: administrator și bibliotecar. Fiecare are atribute distincte (bibliotecarul se ocupă de realizarea împrumuturilor, retururilor, verificarea disponibilității unei cărți și respectiv, adăugarea de noi clienți, iar administratorul se ocupă de gestionarea conturilor clienților, pentru impunerea unor restricții și ridicarea lor, gestionează conturile angajaților și stocul cărților din bibliotecă).

În cele ce urmează se vor prezenta caracteristicile fiecărei clase de utilizatori.

Administratorul se ocupă de gestionarea cărților din bibliotecă. Astfel, acesta poate pune în circulație noi cărți, dar poate lua și decizia de le scoate din circulație. De asemenea, el poate să se ocupe și de gestionare conturilor angajaților: prin adăugarea și ștergerea conturilor acestora, realizând fluxul de personal din interiorul bibliotecii. Deși acesta nu interacționează direct cu clienții librăriei, acesta poate vizualiza conturile clienților problematici și poate lua decizia de a bloca contul, de a impune restricții sau de a ridica a le ridica.

Bibliotecarul este cel care interacționează direct cu clienții bibliotecii. De aceea caracteristicile sale sunt strâns legate de acest aspect. Astfel, acesta poate să adauge un client nou în baza de date a bibliotecii, poate să verifice dacă datele clientului sunt valide, dar și poate întreprinde unele operații asupra cărților. Mai exact, el este cel care realizează împrumuturile și retururile cărților dar poate oferi și sugestii de cărți.

## 2.4 Mediul de operare

Aplicația va rula pe o mașină care folosește sistemul de operare Windows 10 sau Windows 11. Mașina folosește o platformă de tip x86\_64 (64-bit).

Aplicația este dezvoltată folosind limbajul de programare C# și se folosește de o versiune de .Net Framework 4.8.

Pentru a rula aplicația pe o stație de lucru este necesar să fie instalat Microsoft .Net Runtime corespunzător versiunii folosite.

Aplicația poate interacționa cu alte aplicații Windows native, baze de date SQL Server 2019 sau versiuni ulterioare și trebuie să coexiste cu software antivirus și firewall standard din Windows, fără conflicte.

Pentru dezvoltarea și depanarea aplicației prezentate se utilizează mediul de dezvoltare Visual Studio 2019/2022.

Aplicația trebuie să fie compatibilă cu setările de securitate standard ale Windows și cu politicile organizației privind permisiunile utilizatorilor.

## 2.5 Constrângeri de design și implementare

1. Aplicația prezentată folosește o arhitectură de tip client server. Comunicarea dintre cele două componente se realizează folosind protocolul de comunicație TCP. Acest lucru impune o sincronizare atentă și o gestionare corectă a erorilor de rețea.
2. În cadrul clientului, aplicația care va rula pe calculatorul acestuia va fi compusă din două elemente principale: interfața cu care acesta va interacționa și backend-ul. Cele 2 vor comunica folosind protocolul TCP. Conexiunea la server a clientului se realizează prin intermediul backend-ului.
3. Parolele angajaților sunt securizate prin intermediul unei metode de criptare unidirecțională, utilizându-se funcția hash „SHA1CryptoServiceProvider”. Scopul principal al acestei metode este păstrarea confidențialității și integrității datelor din baza de date, prevenind recuperarea parolelor originale chiar și în cazul în care baza de date ar fi compromisă. Astfel, parolele nu sunt stocate în clar, ci sub formă de hash-uri, ceea ce face extrem de dificilă reconstrucția parolelor de către un atacator.
4. Aplicația cu care interacționează clientul, cât și serverul sunt realizate complet folosind limbajul de programare C#. Interfața grafică pentru client este realizată folosind Windows Forms.
5. Pentru a păstra datele persistente în baza de date folosim o bază de date relațională realizată în SQLite. Pentru că acest tip de bază de date nu suportă o accesare simultană, accesul concurent la baza de date este realizat de manual. De asemenea, numai serverul poate accesa baza de date.
6. Aplicația trebuie să respecte convențiile de programare și structurare impuse de cadrul educațional, iar codul trebuie să fie lizibil și documentat corespunzător.

## 2.6 Documentația utilizatorului

Pentru utilizatorii care aleg să folosească produsul nostru vor primi materiale necesare pentru a-l putea utiliza. Suportul oferit este constituit de un manual care include toate explicațiile necesare pentru a putea realiza toate funcționalitățile aplicației, dar îl ajută să înțeleagă mai bine aplicația pe care o folosește. Formatul în care va fi disponibil acest material este doc(Word) și respectiv PDF.

## 2.7 Presupuneri și dependențe

1. Se va considera că biblioteca dispune de hardware-ul necesar pentru a putea să ruleze serverul necesar pentru aplicația care rulează pe mașina bibliotecarului sau a administratorului. Dacă această presupunere se dovedește a fi incorectă va fi necesar să se găsească o soluție alternativă de găzduire.
2. Se presupune că biblioteca dispune de o rețea locală, în care rulează serverul și la care se conectează stațiile cu care interacționează bibliotecarul sau administratorul.
3. Se presupune că toți utilizatorii vor avea acces stabil la rețeaua locală din interiorul bibliotecii pentru a putea realiza comunicarea în conformitate cu constrângerile de performanță impuse.
4. Aplicația va depinde de o bază de date SQLite, care poate fi accesată doar de server, și care va fi disponibilă numai clienților care se conectează. Această bază de date va oferi toate informațiile necesare: cele despre cărți, împrumuturi, returnări și respectiv despre clienți.
5. Proiectul presupune că sistemul de operare care rulează este Windows 10 sau o versiune mai nouă.
6. Se presupune că aplicația nu va necesita accesul din afara rețelei locale din interiorul bibliotecii.
7. Se presupune că biblioteca va furniza datele necesare pentru popularea bazei de date (datele despre cărți, angajați și clienți).
8. Se presupune că personalul bibliotecii va primi o instruire minimă privind utilizarea aplicației și a procedurile de bază.
9. Se presupune că biblioteca va menține o versiune stabilă a sistemului de operare și a dependențelor necesare pentru rularea aplicației fără erori.

### 3. Cerințe ale interfețelor externe

#### 3.1 Interfața cu utilizatorul

Aplicația va fi realizată folosind limbajul de programare C#, utilizând Windows Forms/WPF, oferind o interfață clasică, de tip desktop, specifică sistemelor de tip Windows. Butoanele, controalele și meniurile vor respecta stilul nativ Windows pentru o experiență familiară utilizatorului.

Prin intermediul interfeței grafice se va oferi funcționalități de bază de navigare, introducere și căutare date, printr-un meniu principal sau butoane de pe ecranul principal.

Caracteristici generale:

- Interfața va respecta un stil unitar, bazat pe o schemă de culori neutră și fonturi lizibile.
- Componentele meniului principal sunt grupate în funcție de tipul acțiunii pe care o realizează angajatul bibliotecii.

Constrângeri privind layout-ul:

- Se vor utiliza componente standard (input text, dropdown, butoane, checkbox) în conformitate cu framework-ul Windows Forms.

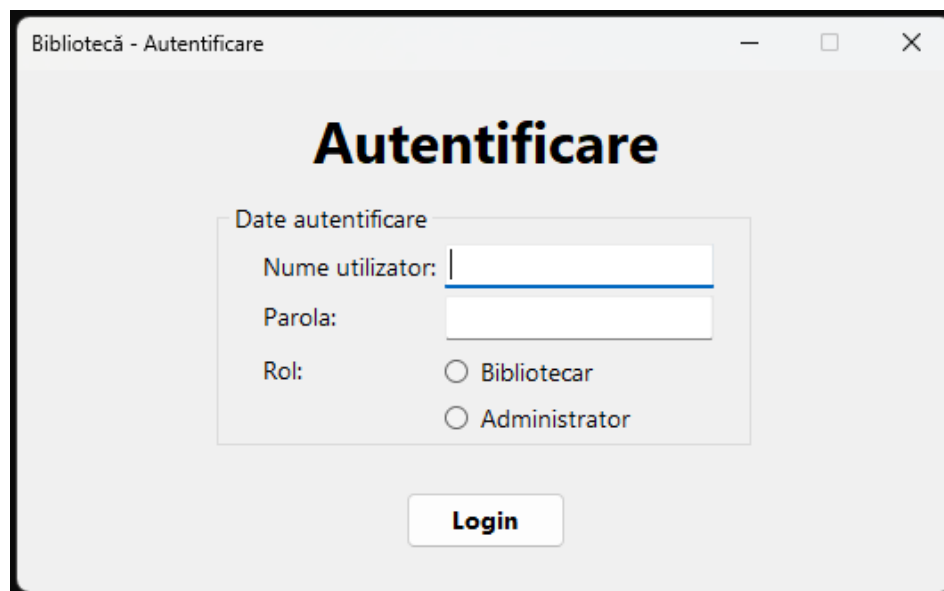
Funcționalități generale:

- Căsuță de căutare
- Confirmări la acțiuni critice
- Mesaje de avertizare

Componente care necesită interfață grafică:

- Modulul de autentificare
- Panoul principal
- Formularele pentru gestionarea clienților și a împrumuturilor.

Când aplicația de lansează în execuție se afișează un meniu de autentificare. Pentru a se autentifica utilizatorul trebuie să introducă următoarele date: numele de utilizator, parola și să selecteze rolul pe care îl are.



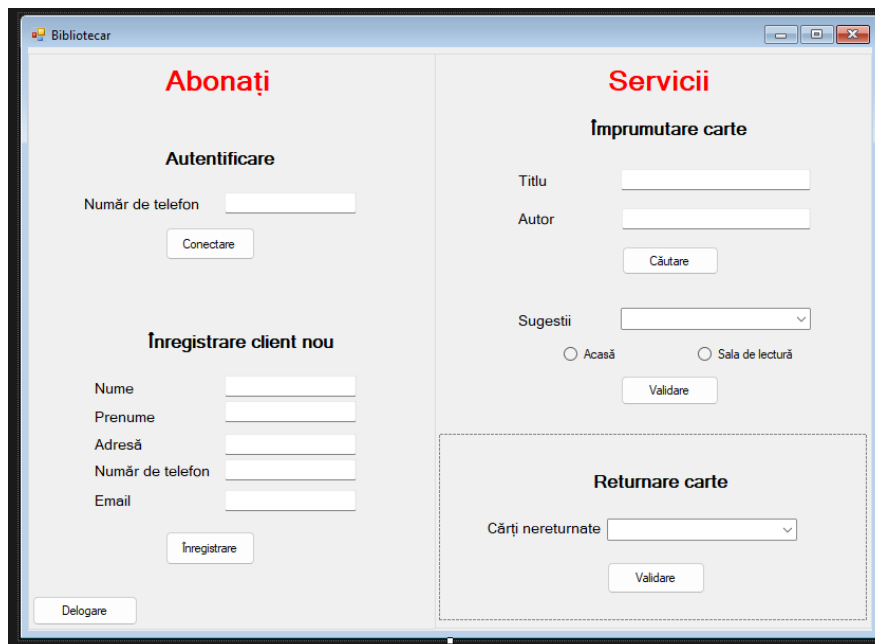
The image shows a Windows Forms application window titled "Biblioteca - Autentificare". The window has a light gray background and a dark gray title bar with standard Windows window controls (minimize, maximize, close). The main content area features a large, bold, black heading "Autentificare" centered at the top. Below the heading is a white rectangular box with a thin gray border containing the login form. The form is titled "Date autentificare" in a small, gray font. It includes three labels: "Nume utilizator:" followed by a text input field, "Parola:" followed by a password input field, and "Rol:" followed by two radio button options: "Bibliotecar" and "Administrator". Below the form box is a white button with a black border and the text "Login" in bold black font.

Figură 1 Interfață autentificare



În funcție de rolul selectat, bibliotecar sau administrator, dacă datele sunt valide atunci se va deschide un nou meniu, personalizat pentru fiecare rol.

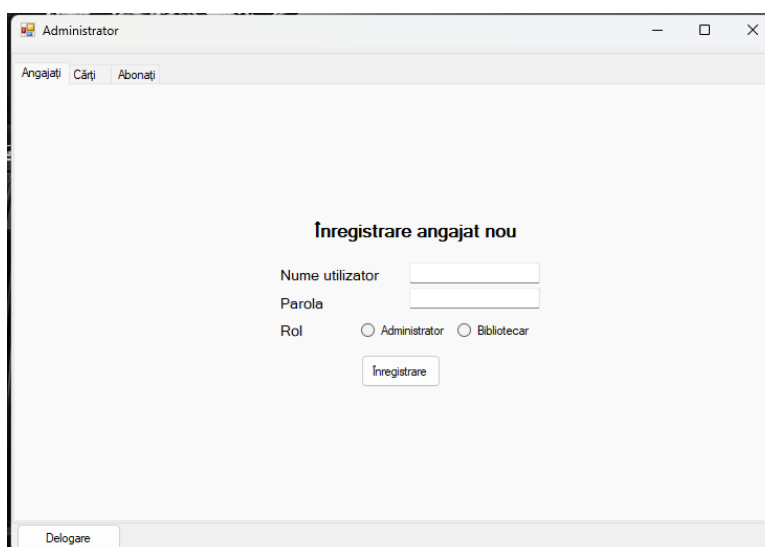
Bibliotecarul este cel care interacționează direct cu clienții. Pentru acesta panoul meniului principal este împărțit în 2. În partea din stânga sunt acțiunile pe care le face asupra conturilor clienților: autentificarea lor sau adăugarea de noi clienți. În partea din stânga sunt meniurile legate de cărți: împrumuturile, returnurile și oferirea de sugestii.



Figură 2 Interfața bibliotecarului

Administratorul dispune de o interfață mai complexă, organizată pe file, fiecare cu un rol bine definit. Filele disponibile sunt afișate în partea de sus a meniului.

Întâi avem fila cu angajați, de unde poate înregistra noi angajați, pentru a putea menține fluxul de angajați din bibliotecă. Acesta este prima filă pe care o vede el după ce se autentifică.



Figură 3 Prima filă a interfeței administratorului

În a doua filă se vor gestiona cărțile disponibile ale bibliotecii.

Administrator

Angajați Cărți Abonați

**Adăugare carte**

ISBN

Titlu

Autor

Editură

Gen

Adaugă

**Ștergere carte**

ISBN

Căutare

Șterge

Delogare

**Figură 4 Fila a doua din interfața administratorului**

Ultima filă este cea care îi permite administratorului să modifice statusul clienților bibliotecii, mai exact să impună și să scoată restricții asupra clienților în funcție de modul în care au realizat returnările.

Administrator

Angajați Cărți Abonați

☐ Selectare abonat

☐ Afișare abonați

**Gestiune abonați**

Număr de telefon

Căutare

Date despre abonat

**Afișare abonați cu probleme**

Căutare

Abonați

☐ Aplicare restricție ☐ Eliminare restricție ☐ Blocare abonat

Validare

Delogare

**Figură 5 Fila a treia din interfața administratorului**

## 3.2 Interfața hardware

Aplicația este compusă din trei componente importante: interfața, backend-ul clientului și respectiv serverul. Comunicația se realizează între interfață și backend, respectiv între backend și server. Fiecare are un format specific de mesaje.

Interfața și backend-ul folosesc ca și comunicație șiruri de caractere. Elementele care compun mesajul sunt despărțite de caracterul „|”. Structura este următoarea: primul element din mesaj este constituit de tipul operației care urmează să se execute, urmat de datele necesare acelei operații.

Comunicarea dintre backend și server se realizează folosind un mesaj serializat sub formă de JSON. Acesta conține următoarele chei: operația și respectiv datele operației care sunt organizate sub forma unui vector. În cadrul vectorului se găsesc perechi cheie valoare, care vor fi transformate la momentul primirii mesajului într-un dicționar.

## 3.3 Interfața software

Această secțiune descrie conexiunile dintre produsul software dezvoltat și alte componente software externe, inclusiv baze de date, sisteme de operare, biblioteci și alte instrumente integrate. De asemenea, sunt specificate protocolul de comunicare și datele schimbate între aceste componente.

1. Conexiuni cu componente software externe
  - 1.1. Baza de date: Aplicația utilizează o bază de date relațională SQLite pentru stocarea informațiilor despre cărți, clienți, împrumuturi și angajați. Accesul la baza de date este realizat exclusiv de server, folosind interogări SQL pentru operațiile CRUD (Create, Read, Update, Delete).
    - 1.1.1. Versiune: SQLite 3.x
    - 1.1.2. Protocol de acces: Conexiune directă prin intermediul bibliotecii System.Data.SQLite.
  - 1.2. Sistemul de operare: Aplicația este dezvoltată pentru a rula pe sistemele de operare Windows 10 și Windows 11, utilizând .NET Framework 4.8. Dependențe: Microsoft .NET Runtime corespunzător versiunii utilizate.
  - 1.3. Mediul de dezvoltare: Visual Studio: Versiunile 2019 sau 2022 sunt utilizate pentru dezvoltare și depanare.
2. Mesaje și fluxuri de date
  - 2.1. Comunicare client-server: Format mesaje: JSON, cu structura:

```
{
  "operation": "nume_operatie",
  "data": [
    {
      "key1": "value1"
    },
    {
      "key2": "value2"
    }
  ]
}
```
  - 2.2. Protocol: TCP, cu criptare a mesajelor pentru securitate.
  - 2.3. Comunicare interfață-backend: Format mesaje: Șiruri de caractere separate prin caracterul |, cu structura: tip\_operatie|date\_1|date\_2|...
3. Servicii necesare
  - 3.1. Autentificare utilizator: Serverul validează credențialele primite de la client folosind funcția hash SHA-1 pentru parole.

- 3.2. Gestionarea bază de date: Serverul procesează interogările SQL și returnează rezultatele în format JSON.
- 3.3. Notificări: Serverul trimite email-uri de notificare clienților cu privire la termenele de retur.
4. Date partajate
  - 4.1. Baza de date: Formată din tabelele:
    - Utilizatori (nume, rol, parolă);
    - Cărți (titlu, autor, cod ISBN etc.);
    - Abonați (date personale, status, limite);
    - Împrumuturi (dată împrumut/returnare, id client, id carte).
  - 4.2. Fișiere de configurare: Setările de rețea (adresa IP a serverului, port) sunt stocate într-un fișier JSON local.
5. Constrângeri de implementare:
  - Accesul concurent la baza de date este gestionat manual pentru a evita conflictele.
  - Toate mesajele între client și server sunt criptate

### 3.4 Interfața de comunicații

Această secțiune definește cerințele și specificațiile legate de comunicarea între componentele sistemului și cu alte sisteme externe. Se detaliază protocoalele, formatele mesajelor, securitatea și alte aspecte relevante pentru schimbul de date.

1. Protocoale de comunicare:
  - 1.1. TCP – Utilizat pentru comunicarea dintre client și server, asigurând o conexiune stabilă și orientată pe conexiune.
    - 1.1.1. Porturi:
      - Serverul ascultă pe un port dedicat (ex. 12345).
      - Conexiunile sunt inițiate de către clienți (bibliotecari/administratori).
    - 1.1.2. Handshake: Clientul trimite o cerere de conectare, iar serverul răspunde cu o confirmare.
2. Formatul mesajelor
  - 2.1. Client-Server:
    - Mesajele sunt serializate în JSON pentru flexibilitate.
    - Răspunsurile includ un cod de stare (success, error) și detalii suplimentare.
3. Server-Client (notificări):
  - 3.1. Email-urile au un șablon prestabilit:

Subiect: Reminder - Termen de returnare carte

Mesaj: Stimate client, vă reamintim că data limită pentru returnarea cărții "[Titlu]" este [Data].

4. Securitate și criptare
  - Criptarea mesajelor: Toate datele transmise între client și server sunt criptate folosind AES-256.
5. Performanță și toleranță la erori
  - Latență: Răspunsurile serverului trebuie să ajungă în maxim 2 secunde pentru operațiuni critice (ex: împrumut/retur).
6. Constrângeri
  - 6.1. Acces restricționat: Serverul acceptă conexiuni doar din rețeaua locală a bibliotecii (adrese IP din intervalul configurat).
  - 6.2. Firewall: Aplicația trebuie să funcționeze fără modificări ale setărilor de firewall ale sistemului.

Această structură asigură o comunicare fiabilă și securizată între componentele sistemului, respectând celelalte cerințe din SRS.

## 4. Caracteristicile sistemului

### 4.1 Autentificare pe roluri

#### 4.1.1 Descriere și prioritate

Această funcționalitate permite utilizatorilor să se autentifice folosind un nume, o parolă, dar și selectând rolul pe care îl are utilizatorul care dorește să se conecteze. În funcție de credențialele introduse, sistemul le va valida și va oferi acces la funcționalitățile specifice rolului pe care îl are utilizatorul.

- **Prioritate:** ridicată
- **Beneficiu:** 9
- **Penalizare:** 8
- **Cost:** 4
- **Risc:** 3

#### 4.1.2 Secvența stimul/răspuns

1. Stimul: utilizatorul introduce datele de autentificare: numele de utilizator, parola și rolul.

Răspuns: Sistemul verifică dacă datele sunt valide

2. Stimul: utilizatorul apasă butonul de autentificare

Răspuns:

- Dacă datele sunt valide, utilizatorul este redirecționat către interfața care corespunde rolului său.
- Dacă datele nu sunt valide se va afișa un mesaj de eroare.

#### 4.1.3 Cerințe funcționale

- Sistemul trebuie să permită introducerea unui nume utilizator, unei parole și selectarea unui rol.
- Sistemul trebuie să valideze credențialele introduse folosind o bază de date.
- În cazul în care datele sunt valide, sistemul trebuie să redirecționeze utilizatorul la interfața specifică rolului său.
- În cazul unor date de autentificare invalide, sistemul trebuie să afișeze un mesaj de eroare.

### 4.2 Înregistrare împrumut

#### 4.2.1 Descriere și prioritate

Această funcționalitate este disponibilă bibliotecarului și este folosită de momentul în care un client care nu este blocat dorește să realizeze un împrumut. Automat se va seta o perioadă de retur de 14 zile dacă clientul împrumută cartea acasă sau dacă acesta va împrumuta cartea în sala de lectură se va seta ca dată de retur identică cu data curentă.

- **Prioritate:** ridicată
- **Beneficiu:** 8
- **Penalizare:** 7
- **Cost:** 5
- **Risc:** 4

#### 4.2.2 Secvența stimul/răspuns

1. Stimul: bibliotecarul caută cartea care o dorește clientul

Răspuns: sistemul caută în baza de date cartea după titlu și autor, acceptând căutări după cuvinte cheie și trimite lista de cărți.

2. Stimul: Se selectează cartea pe care o dorește clientul să o împrumute.

Răspuns: se introduce în baza de date un împrumut și se setează o dată de retur pentru carte.

#### 4.2.3 Cerințe funcționale:

- Sistemul trebuie să fie capabil să caute o carte după autor sau titlu, și căutarea să permită și o căutare după titlu și autor parțiale.
- Sistemul trebuie să returneze o listă de cărți care corespund datelor de căutare introduse.
- Sistemul trebuie să fie capabil să seteze o dată de retur pentru o carte în funcție de tipul împrumutului (acasă sau în sala de lectură).
- Sistemul trebuie să înregistreze împrumutul făcut de client și să reducă numărul de cărți pe care le poate împrumuta.
- Sistemul va modifica statusul cărții din disponibilă în indisponibilă.

### 4.3 Înregistrare retur

#### 4.3.1 Descriere și prioritate

Această funcționalitate este disponibilă numai bibliotecarului, deoarece acesta interacționează cu clientul. Acesta va verifica lista de cărți pe care le-a împrumutat și va seta că s-a realizat împrumutul.

- **Prioritate:** ridicată
- **Beneficiu:** 8
- **Penalizare:** 8
- **Cost:** 5
- **Risc:** 4

#### 4.3.2 Secvența stimul/răspuns

1. Stimul: vizualizare cărți împrumutate  
Răspuns: o listă de cărți pe care le-a împrumutat clientul
2. Stimul: selectare carte din lista de cărți împrumutate  
Răspuns: se inserează în baza de date data returului

#### 4.3.3 Cerințe funcționale:

- Sistemul va seta data de retur a cărții.
- Sistemul va mări limita de cărți pe care clientul o poate împrumuta.
- Sistemul va modifica statusul cărții din indisponibilă în disponibilă.

### 4.4 Autentificare client

#### 4.4.1 Descriere și prioritate

Prin intermediul acestei funcționalități se permite autentificarea clientului bibliotecii folosind numărul de telefon. Pe baza acestuia se va modifica interfața bibliotecarului prin afișarea opțiunilor pe care le poate face: dacă clientul este blocat el poate face doar returnuri, dacă clientul este restricționat el va putea împrumuta cărți doar în sala de lectură și dacă el nu are restricții va putea să împrumute cartea și acasă.

- **Prioritate:** ridicată
- **Beneficiu:** 7
- **Penalizare:** 8
- **Cost:** 5
- **Risc:** 4

#### 4.4.2 Secvența stimul/răspuns

1. Stimul: introducerea numărului de telefon.  
Răspuns: validarea numărului de telefon.
2. Stimul: apăsarea butonului de autentificare client.

Răspuns: validarea numărului de telefon prin verificarea în baza de date și activarea elementelor din interfață în funcție de statusul clientului: blocat, cu restricții sau fără restricții.

#### 4.4.3 Cerințe funcționale

- Sistemul va verifica dacă șirul de caractere introdus este un număr de telefon valid.
- Sistemul va verifica dacă există un client cu acel număr de telefon în baza de date.
- În cazul în care numărul este valid se vor activa elementele din interfață pe baza statusului clientului
- În cazul în care un număr de telefon este invalid se va afișa un mesaj de eroare.

## 4.5 Adăugare client nou

### 4.5.1 Descriere și prioritate

Această funcționalitate este disponibilă numai bibliotecarului și are rolul de a permite constant bibliotecii să adauge noi clienți. Pentru aceasta bibliotecarul va completa un formular cu datele necesare și prin apăsarea unui buton acesta va trimite la server o cerere de inserare în baza de date a datelor introduse.

- Prioritate: ridicată
- Beneficiu: 7
- Penalizare: 5
- Cost: 5
- Risc: 4

### 4.5.2 Secvența stimul/răspuns

1. Stimul: introducerea datelor client

Răspuns: validarea datelor necesare pentru clientul nou

2. Stimul: apăsarea butonului de înregistrare a noului client.

Răspuns: serverul va introduce în baza de date datele clientului

### 4.5.3 Cerințe funcționale

- Sistemul va valida datele clientului prin verificarea numărului de telefon să respecte un format și prin asigurarea că toate datele clientului sunt oferite.
- În caz de succes sistemul va insera în baza de date, datele despre noul client.
- În caz de eșec se va afișa un mesaj de eroare

## 4.6 Ștergerea unei cărți

### 4.6.1 Descriere și prioritate

Această funcționalitate este disponibilă numai administratorului și care îi permite să modifice stocul cărților. Astfel, acesta va introduce ISBN-ul cărții și va primi o listă de cărți și prin selectarea unui element din listă și la apăsarea butonului ștergere se va șterge cartea din baza de date.

- **Prioritate:** medie
- **Beneficiu:** 7
- **Penalizare:** 5
- **Cost:** 5
- **Risc:** 4

### 4.6.2 Secvența stimul/răspuns

1. Stimul: introducerea ISBN-ului cărții.

Răspuns: o listă de cărți care au ISBN-ul introdus.

2. Stimul: Selectarea unei cărți.

- Răspuns: validarea inputului unui stimul.
3. Stimul: apăsarea butonului ștergere.  
Răspuns: Ștergerea cărții din baza de date.
- 4.6.3 Cerințe funcționale
1. Sistemul trebuie să fie capabil să șteargă o carte din baza de date, folosind numai identificatorul unic al cărții.

## 4.7 Adăugarea unei cărți

### 4.6.1 Descriere și prioritate

Aceasta este o funcție disponibilă administratorului și care îi permite să modifice stocul cărților prin introducerea unui nou exemplar al unei cărți. Pentru aceasta el va trebui să introducă un set de informații (ISBN, titlu, autor, editură și gen) și apoi să apese butonul adaugă.

- **Prioritate:** medie
- **Beneficiu:** 7
- **Penalizare:** 5
- **Cost:** 5
- **Risc:** 4

### 4.6.2 Secvența stimul/răspuns

1. Stimul: introducerea datelor unei cărți  
Răspuns: validarea datelor despre carte introduse
2. Stimul: apăsarea butonului adaugă  
Răspuns: inserarea în baza de date a informațiilor despre noua carte .

### 4.6.3 Cerințe funcționale

- Sistemul va verifica validitatea datelor despre carte.
- Sistemul va introduce în tabelele corespunzătoare datele despre cărți.

## 4.8 Afișare listă abonați problematici

### 4.8.1 Descriere și prioritate

Această funcționalitate este disponibilă numai administratorului bibliotecii. Prin apăsarea butonului căutare el va primi o listă cu abonați problematici. În acea listă se vor afla termeni necesari pentru administrator pentru a modifica statusului clientului

- **Prioritate:** medie
- **Beneficiu:** 7
- **Penalizare:** 6
- **Cost:** 7
- **Risc:** 7

### 4.8.2 Secvența stimul/răspuns

1. Stimul: apăsarea butonului de căutare abonați problematici.  
Răspuns: afișarea unei liste cu abonați care sunt considerați problematici.

### 4.8.3 Cerințe funcționale

- Sistemul trebuie să fie capabil să identifice clienții problematici ai bibliotecii: cei care întârzie constant cu returnarea cărților, sau cei care au întârziat prea mult timp pentru returnarea unei cărți.

## 4.9 Modificare status abonați

### 4.9.1 Descriere și prioritate



Această funcționalitate este disponibilă numai administratorului și are scopul de a impune sau să ridice restricții. Acesta va selecta un abonat, fie din lista de abonați problematici, fie abonatul care a fost căutat cu ajutorul numărului de telefon.

- **Prioritate:** ridicată
- **Beneficiu:** 8
- **Penalizare:** 6
- **Cost:** 4
- **Risc:** 3

#### 4.9.2 Secvența stimul/răspuns

1. Stimul: selectarea unui abonat  
Răspuns: verificarea selectării unui abonat
2. Stimul: selectarea noului status al clientului.  
Răspuns: verificarea selectării unui status pentru client.
3. Stimul: apăsarea butonului de modificare a statusului.  
Răspuns: modificarea statusului clientului

#### 4.9.3 Cerințe funcționale

- Sistemul trebuie să fie capabil să afișeze o listă de abonați problematici
- Sistemul trebuie să fie capabil să caute un abonat după numărul de telefon.
- Sistemul trebuie să fie capabil să modifice statusul unui abonat.
- Sistemul trebuie să conțină un meniu care să permită selectarea noului status.

### 4.10 Căutare angajat după număr de telefon

#### 4.10.1 Descriere și prioritate

Această funcționalitate este disponibilă numai administratorului. Acesta poate căuta un abonat pe baza numărului de telefon și poate vizualiza datele despre acesta.

- **Prioritate:** medie
- **Beneficiu:** 7
- **Penalizare:** 6
- **Cost:** 7
- **Risc:** 7

#### 4.10.2 Secvența stimul/răspuns.

1. Stimul: administratorul introduce numărul de telefon  
Răspuns: validare număr de telefon
2. Stimul: apăsarea butonului de căutare  
Răspuns: afișarea informațiilor despre client.

#### 4.10.3 Cerințe funcționale

- Sistemul trebuie să fie capabil să caute un abonat după numărul de telefon
- În cazul în care numărul introdus există în baza de date, sistemul trebuie să fie capabil să afișeze datele clientului.
- În cazul în care numărul introdus nu există, sistemul trebuie să afișeze un mesaj de eroare.

### 4.11 Adăugare angajat

#### 4.11.1 Descriere și prioritate

Această funcționalitate este disponibilă administratorului și prin intermediul ei acesta poate să modifice baza de date a angajaților prin inserarea de noi angajați cu cele 2 roluri disponibile: administrator și bibliotecar.

- **Prioritate:** ridicată
- **Beneficiu:** 8
- **Penalizare:** 6

- **Cost:** 4
  - **Risc:** 3
- 4.11.2 Secvența stimul/răspuns
1. Stimul: introducere date angajat  
Răspuns: validare date angajat
  2. Stimul: apăsarea butonului înregistrare  
Răspuns: crearea unei noi linii în tabela Utilizator.
- 4.11.3 Cerințe funcționale
- Sistemul trebuie să fie capabil să introducă date în tabela de utilizatori.

## 4.12 Ștergerea unui angajat

### 4.12.1 Descriere și prioritate

Această funcționalitate este disponibilă numai administratorului și are ca scop urmărirea fluxului de personal. Acesta este limitat la ștergerea numai a bibliotecarilor, neputând să șteargă un administrator. Pentru a șterge un administrator va fi necesară intervenția echipei care gestionează aplicația.

- **Prioritate:** ridicată
- **Beneficiu:** 8
- **Penalizare:** 6
- **Cost:** 4
- **Risc:** 3

### 4.12.2 Secvența sistem/răspuns

1. Stimul: introducerea numelui de utilizator a angajatului  
Răspuns: verificare nume utilizator
2. Stimul: apăsarea butonului de ștergere.  
Răspuns: se va șterge din baza de date un utilizator.

### 4.12.3 Cerințe funcționale

- Sistemul trebuie să verifice dacă numele de utilizator nu este al unui administrator, în acest caz oprindu-se operația de ștergere.
- Sistemul trebuie să fie capabil să șteargă un bibliotecar din baza de date.
- În cazul în care ștergerea unui angajat al bibliotecii eșuează, sistemul trebuie să fie capabil să afișeze un mesaj de eroare.

## 5. Alte cerințe nefuncționale

### 5.1 Cerințe de performanță

Aplicația de bibliotecă trebuie să asigure un timp de răspuns adecvat și o funcționare stabilă, astfel încât utilizatorii (administratori și bibliotecari) să poată interacționa cu sistemul în mod fluent, fără întârzieri semnificative.

Cerințele de performanță includ următoarele aspecte:

#### 1. Timp de răspuns

Pornirea aplicației poate necesita mai mult timp, întrucât este necesară conectarea prin TCP a clientului la server. Cu toate acestea, sistemul trebuie să răspundă la solicitările de autentificare imediat după introducerea datelor de conectare.

- Afișarea listelor de cărți, abonați sau angajați trebuie să se realizeze în maximum 3 secunde, pentru baze de date cu până la 10.000 de înregistrări.

- Operațiile de adăugare, modificare sau ștergere a datelor (ex. angajat, carte, abonat) trebuie să fie procesate în maximum 2 secunde de la inițierea acțiunii.
2. Concurență
    - Aplicația trebuie să suporte minim 5 utilizatori activi simultan, fără scăderi notabile de performanță.
    - Sub încărcare normală, timpul de răspuns nu trebuie să crească cu mai mult de 20% față de valorile specificate.
  3. Interactivitate în timp real
    - Operațiile frecvente efectuate de bibliotecari (ex. împrumut/returnare carte) trebuie să fie confirmate de sistem în sub 1 secundă, pentru a susține activitatea de lucru cu publicul.
  4. Gestionarea erorilor
    - În cazul unor erori de comunicare cu baza de date, sistemul trebuie să afișeze un mesaj de eroare clar în mai puțin de 1 secundă, fără a bloca interfața.
    - Aplicația trebuie să prevină blocarea completă în urma erorilor interne.
  5. Inițializare sistem
    - Timpul de pornire al aplicației (inclusiv inițializarea conexiunii la baza de date) nu trebuie să depășească 5 secunde pe un sistem cu cerințe minime (Windows 10, 4GB RAM, procesor dual-core).

## 5.2 Cerințe de siguranță

Sistemul trebuie să protejeze datele sensibile ale utilizatorilor și să prevină accesul neautorizat, asigurând un nivel adecvat de securitate informatică pentru o aplicație de gestiune a unei biblioteci.

1. Autentificare și control acces
  - Fiecare utilizator trebuie să se autentifice folosind un nume de utilizator și o parolă.
  - Accesul la funcționalități este restricționat pe baza rolului utilizatorului: administrator sau bibliotecar.
  - Administratorii pot gestiona angajați, cărți și abonați.
  - Bibliotecarii pot doar vizualiza și gestiona cărți și abonați.
2. Stocarea parolelor
  - Parolele nu trebuie stocate în format text clar în baza de date.
  - Se va utiliza un algoritm de hashing securizat pentru stocarea parolelor.
3. Validarea datelor introduse
  - Toate datele introduse de utilizatori trebuie validate atât la nivel de interfață grafică, cât și la nivel de back-end, pentru a preveni:
    - inserții SQL (SQL injection)
    - cod malițios (ex. scripturi)
4. Confidențialitate
  - Informațiile despre clienți și împrumuturi trebuie accesate doar de către utilizatorii autorizați, conform rolului definit.

- Este interzisă expunerea publică sau accidentală a datelor personale (ex. parole, nume de utilizator).
5. Mecanisme de protecție
    - Aplicația trebuie să aibă un mecanism de deautentificare clar și funcțional, pentru a evita accesul neautorizat în cazul părăsirii stației de lucru.

### 5.3 Cerințe de securitate

1. Autentificare și controlul accesului
  - Toți utilizatorii trebuie să se autentifice cu credențiale unice (nume utilizator și parolă)
  - Parolele trebuie să fie stocate în format hash utilizând algoritmi moderni.
2. Protecția datelor
  - Toate datele sensibile trebuie criptate.
  - Implementarea unui sistem de clasificare a datelor:
    - Date personale - nivel maxim de protecție
    - Date operaționale - nivel mediu
    - Date publice - nivel minim
3. Securitatea aplicației
  - Protecție împotriva vulnerabilităților comune:
  - SQL Injection (folosire parametrizată a interogărilor)
  - XSS (Cross-Site Scripting)
4. Securitatea rețelei
  - Restricționarea accesului la resurse critice
  - Segmentarea rețelei pentru izolarea componentelor sensibile
5. Conștientizarea utilizatorilor
  - Training anual de securitate pentru personal
  - Ghiduri de bune practici pentru utilizatori

Această abordare cuprinzătoare asigură o postură robustă de securitate, acoperind toate aspectele critice ale sistemului, de la protecția datelor la conștientizarea utilizatorilor și răspunsul la incidente.

### 5.4 Atribute ale calității software-ului

1. Fiabilitate
  - Disponibilitate: Sistemul trebuie să ofere o disponibilitate de minim 99% pe parcursul orelor de funcționare a bibliotecii
2. Performanță
  - Timp de răspuns: Sub 2 secunde pentru 95% din operațiunile uzuale
  - Scalabilitate: Suport pentru creșterea numărului de utilizatori cu 20% anual fără modificări majore
  - Utilizare resurse: Consum memorie stabil sub 500MB în condiții normale de funcționare
3. Uzabilitate
  - Interfață intuitivă: Noii utilizatori pot efectua operațiuni de bază după maxim 30 de minute de familiarizare

- Documentație: Manuale online și tooltips integrate pentru toate funcțiile complexe
- 4. Securitate
  - Protecție date: Criptare pentru toate datele sensibile
- 5. Mentenanță
  - Modularitate: Arhitectură pe module independente cu interfețe bine definite
  - Documentație cod: 80% acoperire comentarii pentru componentele critice
  - Testabilitate: Suport pentru teste automate pe 90% din funcționalități
- 6. Portabilitate
  - Compatibilitate: Funcționare pe Windows 10/11 (prin .NET Framework)
  - Dependențe: Minimă, cu toate bibliotecile third-party incluse în pachet
  - Configurare: Adaptare ușoară la diferite medii de operare
- 7. Robustete
  - Validare intrare: Sistem avansat de validare a datelor de intrare
  - Gestionare erori: Mesaje clare și sugestii de rezolvare pentru erori comune
- 8. Testare
  - Suite teste: Acoperire minimă de 75% cu teste automate
  - Testare performanță: Instrumente integrate pentru măsurarea timpilor de răspuns
  - Mediu testare: Separare clară între mediile de test și producție
- 9. Standarde de codificare
  - Convenții: Respectarea standardelor de codificare C# Microsoft
- 10. Metrici de calitate: Zero vulnerabilități cunoscute de severitate ridicată

Aceste atribute asigură dezvoltarea unui produs software de înaltă calitate care îndeplinește nevoile bibliotecii și ale utilizatorilor săi, oferind o experiență stabilă, sigură și eficientă pe termen lung.

## 5.5 Alte cerințe nefuncționale

1. Conformitate: Aplicația respectă standardele ISO pentru biblioteci și GDPR.
2. Documentație: Este oferit un manual DOCX/PDF.
3. Implementare: Mediu de dezvoltare este standardizat, iar procesul de build și testare este automatizat.
4. Instruirea este formată dintr-un program de training de 2 ore.

Aceste cerințe completează specificațiile sistemului, asigurând că soluția va răspunde tuturor nevoilor organizației atât pe termen scurt, cât și pe termen lung, în conformitate cu cele mai bune practici din domeniu.

## 5.6 Reguli de business

1. Politici de împrumut
  - 1.1. Perioadă standard de împrumut:
    - Acasă: 21 zile
    - Sala de lectură: 1 zi (se returnează înainte de a se părăsi biblioteca)
  - 1.2. Număr maxim de cărți împrumutate simultan = 5 (indiferent de locație)
2. Penalizări pentru întârzieri
  - 2.1. Suspendare drepturi: fiecare utilizator are un set de drepturi de împrumut al cărților în funcție de retururi: dacă a întârziat prea mult cu o carte sau dacă întârzie constant la returul cărților. Suspendarea drepturilor este modificată de către administrator, prin modificarea statusului utilizatorului: blocat, cu restricții sau fără restricții. Abonatul blocat este un abonat care poate doar să returneze o carte, cererile de împrumut fiind

refuzate de bibliotecar (în interfață meniul pentru împrumuturi este dezactivat). Abonatul cu restricții este cel care poate împrumuta cărți numai în sala de lectură (operație indicată din interfață prin dezactivarea opțiunii de a împrumuta cărți acasă). Abonatul fără restricții este cel care poate împrumuta cărți și acasă și în sala de lectură, în limita numărului de cărți împrumutate pe care le poate realiza.

3. Acces și restricții
  - 3.1. Restricții pentru utilizatorii cu sancțiuni:
    - Cont restricționat: împrumut doar în sală
    - Cont blocat: acces suspendat
4. Cerințe pentru împrumut: Validare identitate cu buletin/carte de identitate
5. Gestionarea cărților
  - 5.1. Adăugare în catalog:
    - Doar de către administrator
    - Verificare ISBN și metadate obligatorie
6. Retrageră din circulație: Decizie finală luată de administrator
7. Politici pentru personal
  - 7.1. Atribuții clare:
    - Bibliotecari: doar operațiuni de împrumut/retur
    - Administratori: gestiune conturi și modificări catalog
8. Scenarii speciale
  - Cărți pierdute/ deteriorate: Ban permanent.

## 6. Alte cerințe

### 6.1 Notificare clienți retur

Această funcționalitate este realizată automat, fără intervenția bibliotecarului sau a administratorului. Acesta presupune parcurgerea o dată la 24 de ore a bazei de date, tabeli cu abonați, cu scopul de a verifica dacă există abonați restricționați sau care au întârziat cu returul unei cărți. Dacă există un abonat care respectăm aceste condiții, se va trimite un mail personalizat, după cum s-a putut vedea structura în [Interfața de comunicații](#).

- Tip notificare: reamintire retur carte
- Mijloace de comunicare: email trimis automat de sistem
- Periodicitate: se va verifica zilnic baza de date.
- Cerințe tehnice:
  - Integrare serviciu SMTP.

### 6.2 Cerințe privind baza de date

Sistemul va folosi o bază de date relațională pentru stocarea informațiilor despre utilizatori și abonați. Structura bazei de date trebuie să susțină eficient operațiile efectuate de utilizatori și să asigure integritatea și securitatea datelor.

- Structură minimă obligatorie:
  - Tabela „utilizator”: este o tabelă care reține datele despre utilizator( nume utilizator, parolă, rol, id).
  - Tabela „împrumuturi”: aceasta va păstra datele despre împrumut (id împrumut, id carte, id abonat, dată împrumut, dată retur, data efectuării returului).

- Tabela „abonat”: este tabela care conține datele despre clienții bibliotecii (id abonat, nume, prenume, adresa, telefon, email, limita, status).
- Cerințe de integritate:
  - Oprirea ștergerilor altor administratori de către un administrator.
  - Oprirea ștergerilor unui cont de angajat de către un bibliotecar.
  - Oprirea ștergerii sau adăugării unei cărți de către bibliotecar.
- Cerințe de performanță:
  - Posibilitatea de a căuta o carte după un titlu sau autor parțial oferite.
  - Căutarea unei cărți să dureze mai puțin de o secundă.
- Cerințe de securitate:
  - Parolele vor fi păstrate în baza de date folosind o funcție hash.
  - Accesul la baza de date se va realiza numai prin intermediul interfeței grafice, nu direct de către utilizatori.
- Cerințe de persistență:
  - Folosirea de tranzacții pentru a nu introduce în baza de date datele, numai dacă toate acestea sunt valide.

## Appendix A: Glosar

- Bibliotecar: Utilizator al aplicației care se ocupă de împrumutul și returnarea cărților și de gestionarea clienților.
- Administrator: Utilizator cu drepturi extinse, responsabil pentru gestionarea cărților, conturilor angajaților și abonaților problematici.
- Client: Persoană care împrumută cărți de la bibliotecă.
- Împrumut: Operațiunea de a oferi temporar o carte unui client, înregistrată în sistem.
- Retur: Operațiunea de returnare a unei cărți împrumutate de către client.
- Restricție: Limitare impusă unui client care întârzie sau nu respectă regulile bibliotecii (ex. poate împrumuta doar în sală).
- Cont blocat: Contul clientului este suspendat și nu mai poate împrumuta cărți.
- SHA1: Algoritm de criptare unidirecțională folosit pentru stocarea parolelor.
- SQLite: Sistem de gestionare a bazelor de date relaționale, utilizat pentru salvarea datelor în aplicație.
- TCP: Protocol de transport utilizat pentru comunicarea între client și server.
- JSON: Format de schimb de date folosit între client și server pentru transmiterea operațiilor și rezultatelor.

## Appendix B: Modele de analiză

În această parte se va prezenta partea de analiză și proiectare pentru realizarea aplicației. În această etapă s-au realizat o serie de diagrame care să descrie complet funcționalitățile și modul de procesare, care vor fi detaliate în cele ce urmează.

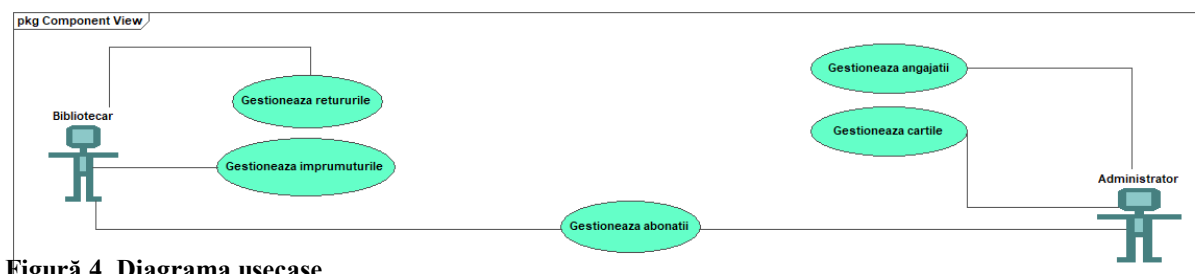
Prima dată s-a realizat diagrama de cazuri de utilizare, sau „use case” după cum mai este cunoscută. Aceasta este ce acare descrie funcționalitățile principale ale aplicației și tipul de utilizatori care vor utiliza această aplicație. Astfel s-a realizat o listă de cazuri de utilizare compusă din:

- Gestionare abonați
- Gestionare împrumuturi
- Gestionare retururi
- Gestionare cărți
- Gestionare angajați

Pentru aceste activități s-a constatat că sunt necesari doi actori distincți:

- Administrator
- Bibliotecar

Diagrama de cazuri de utilizare este reprezentată în imaginea de mai jos:



Figură 4 Diagrama usecase

Generated by UModel

www.altova.com

Figură 5 Diagrama de cazuri de utilizare

În cele ce urmează se vor prezenta și descrie fiecărui actor.

#### 1. Administratorul

Acesta este cel care are o poziție administrativă în bibliotecă fiind responsabil pentru schema de personal al bibliotecii și de menținerea stocului de cărți disponibile din bibliotecă. De asemenea, deși el nu interacționează direct cu abonații bibliotecii, acesta poate realiza unele operații asupra tabelii cu angajați din baza de date, operație detaliată mai jos.

#### 2. Bibliotecatul

Acesta este un utilizator al aplicației propuse care se ocupă de interacțiunea cu abonații bibliotecii. Acesta este responsabil de realizarea împrumuturilor și returnurilor, dar poate realiza și unele operații asupra bazei de date cu abonați.

Cazurile de utilizare sunt prezentate mai jos:

##### 1. Gestionarea abonaților

Gestionarea abonaților este diferită în funcție de ce actor interacționează cu ea. Funcțiile pe care le realizează fiecare actor asupra abonaților sunt distincte.

Administratorul este cel care se ocupă de monitorizarea statusului clienților bibliotecii. Pe baza acestui status se poate limita accesul al unele facilități ale bibliotecii (pentru utilizatorii restricționați se elimină dreptul de a împrumuta o carte acasă, în timp ce clienții blocați nu mai pot realiza niciun împrumut). Dar indiferent de statusul pe care îl au abonații nu se limitează opțiunea de a returna cărți.

În cazul bibliotecarului gestionarea abonaților are o altă semnificație: ce de a verifica dacă datele utilizatorului sunt valide sau dacă nu. De asemenea, considerând faptul că acesta interacționează direct cu abonații bibliotecii, acesta are posibilitatea de a adăuga abonați noi în baza de date.

##### 2. Gestionare împrumuturi

Gestionarea împrumuturilor este disponibilă numai bibliotecarului. Aceasta este compusă din mai multe operații care trebuie să se ruleze în ordinea specificată. Întâi se va prelua numărul de telefon pentru a ne asigura că există în baza de date o linie care să corespundă cu datele introduse spre a fi verificate. Dacă datele sunt valide, se vor activa meniurile în funcție de statusul pe care îl are abonatul. După ce autentificarea s-a realizat cu succes se va căuta cartea după titlu sau autor. În cele din urmă dacă abonatul decide să continue împrumutul atunci se va introduce în tabela „împrumuturi” cu o dată de retur distinctă în funcție de tipul împrumutului: dacă împrumutul se realizează doar în sala de lectură atunci data de retur coincide cu data curentă. În caz contrar data de retur va fi după 14 zile de la realizarea împrumutului.

##### 3. Gestionare returnuri

Acest caz de utilizare descrie operația de retur a unei cărți. Această operație poate fi realizată numai de bibliotecar și care poate fi realizată de orice abonat, indiferent de statusul pe care îl are. Această operație este compusă din mai multe operații mai mici. Întâi se va căuta clientul în baza de date după numărul de telefon, care este unic și nu se va repeta în baza de date. După ce datele au fost valide se va afișa și o listă de cărți împrumutate. Bibliotecarul va selecta o care din cele



returnate și fiind marcate ca fiind împrumutate, iar după ce va selecta o carte pentru retur atunci se va apăsa butonul de retur se va realiza efectiv returul cărții, marcat de inserarea în baza de date a informațiilor necesare retururilor.

#### 4. Gestionare cărți

Această funcționalitate este disponibilă numai administratorului care este și responsabil de menținerea stocului de cărți. Astfel acest caz de utilizare are două funcționalități mari principale: ștergerea unei cărți și adăugarea unei cărți.

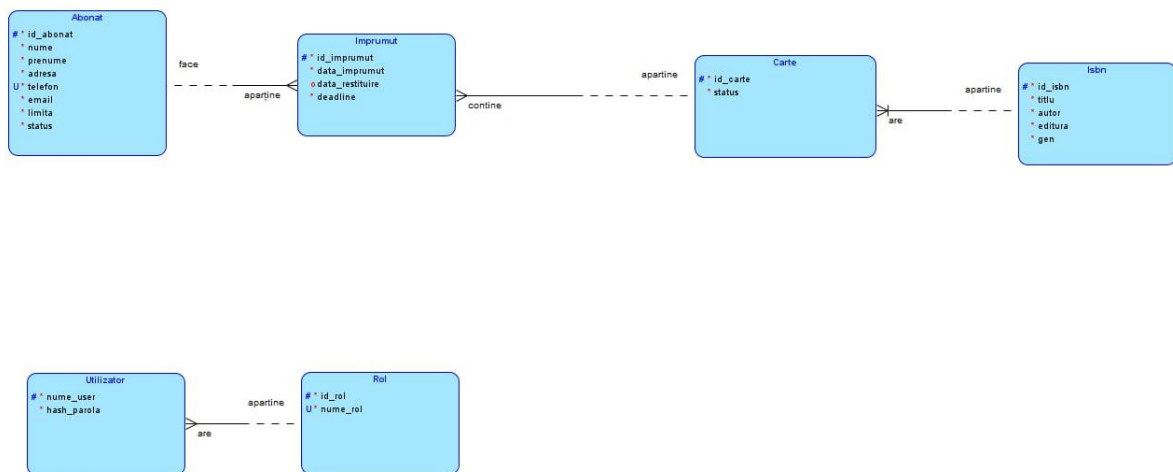
Pentru ștergerea unei cărți se va introduce ISBN-ul unei cărți, se va întoarce o listă de cărți. Din această listă, administratorul poate să selecteze o carte și prin apăsarea unui buton de interfață se realizează procesul de ștergere.

Pentru a adăuga o carte este necesar completarea unui formular care conține datele necesare adăugării unei noi cărți. La apăsarea butonului de adăugare de carte se va introduce în baza de date noua carte, dar nu după ce s-a realizat și o verificare asupra topurilor și datelor trimise.

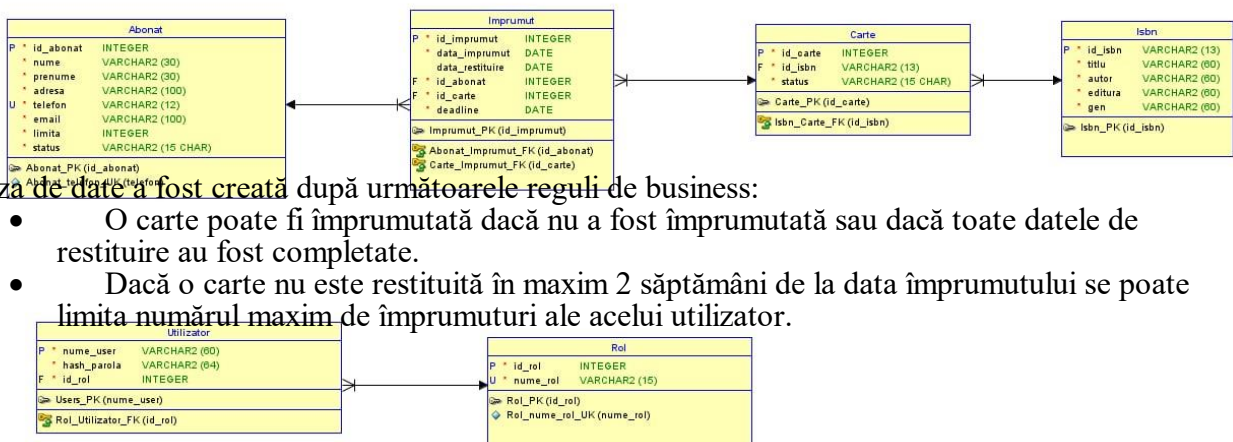
#### 5. Gestionare angajați

Această funcționalitate este disponibilă numai administratorului, și are și ea anumite limitări. Astfel, prin intermediul acestui caz de utilizare se permite o păstrare dinamică a fluxului de angajați din bibliotecă. Însă administratorul poate doar să șteargă un bibliotecar, sau poate adăuga și un bibliotecar, dar și un administrator. Dacă se dorește ștergerea unui administrator este necesară intervenția noastră la aceste proiecte.

După realizarea diagramei de cazuri de utilizare s-a realizat diagrama entitate-relație care descrie baza de date de tip SQLite folosită. Această diagramă este reprezentată în figura de mai jos.



Figură 6 Diagrama entitate relație



Baza de date a fost creată după următoarele reguli de business:

- O carte poate fi împrumutată dacă nu a fost împrumutată sau dacă toate datele de restituire au fost completate.
- Dacă o carte nu este restituită în maxim 2 săptămâni de la data împrumutului se poate limita numărul maxim de împrumuturi ale aceluși utilizator.

Figură 7 Diagramă entitate relație

- Numărul de telefon al unui abonat trebuie să fie unic.
- Abonatul are minim 14 ani pentru a i se face un cont.(trebuie sa aibă buletin pentru adresa de reședință)
- Data de restituire a fiecărui împrumut va fi ulterioară sau identică cu data de împrumut.
- Pentru cărțile împrumutate în sala de lectură trebuie să fie aceeași dată de împrumut și de retur.
- Data de retur nu poate fi o dată ulterioară datei calendaristice curente.
- Data de împrumut este data curentă.
- Numărul de telefon trebuie sa aibă 9 cifre și prefixul aferent acestuia.
- Adresa de email trebuie să aibă un format valid de ex. nume\_utilizator@domeniu.top-level\_domain
- Fiecare carte va fi identificată printr-un cod de inventar.(id\_carte)
- Numele, prenumele , adresa de domiciliu, numărul de telefon, id abonatului, emailul abonatului, numărul maxim de cărți care pot fi împrumutate de un abonat nu pot fi nule.
- Un abonat poate împrumuta maxim numărul de cărți care îl are trecut pe cont.
- Numărul maxim de cărți pe care le poate împrumuta simultan un abonat este 5.
- ISBN este din 10 sau 13 cifre.
- Numele și prenumele sunt formate doar din litere.
- Administratorii și Bibliotecarii sunt identificați în funcție de numele de utilizator.
- Este păstrat doar hash-ul parolei.(SHA-256).

După s-a realizat diagrama de activități care are rolul de a descrie toate activitățile din interiorul sistemului. Astfel, aceasta încearcă să acopere toate activitățile pe care le are fiecare componentă a sistemului: clientului și serverul.

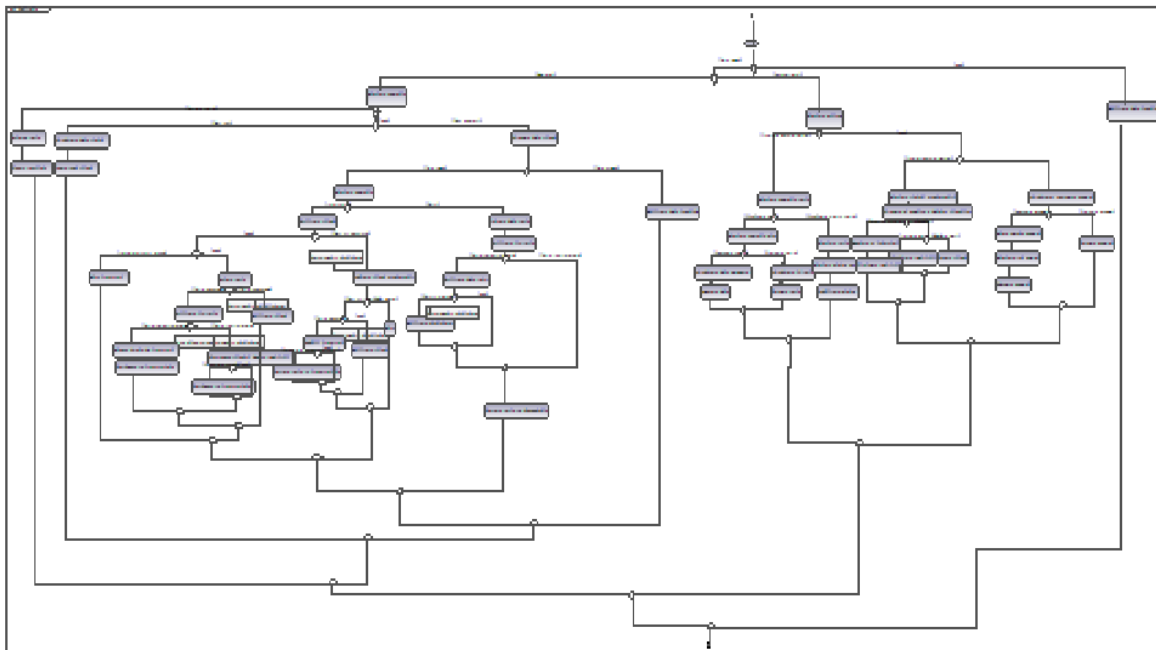


Diagrama de activitate detaliază pașii parcurși de un utilizator de tip client în cadrul aplicației. Activitatea începe cu lansarea aplicației și continuă cu autentificarea în sistem. După autentificare clientul poate accesa diverse funcționalități precum consultarea listei de cărți disponibile, filtrarea cărților, returnarea unei cărți sau vizualizarea istoricului propriu.

Diagrama include multiple condiții decizionale care ghidează ramificarea fluxului în funcție de acțiunile utilizatorului sau de starea contului (ex: dacă are sau nu cărți întârziate, dacă este restricționat, dacă o carte este disponibilă etc.).

Sunt reprezentate și acțiunile administrative automatizate, precum afișarea mesajelor de eroare, restricționarea împrumuturilor sau actualizarea bazei de date cu noile stări ale împrumuturilor.

Activitatea se încheie fie prin deconectarea utilizatorului, fie prin închiderea aplicației. Diagrama reflectă complexitatea și interactivitatea funcționalităților disponibile pentru client, precum și condițiile de validare aplicate de sistem pentru menținerea coerenței datelor și a regulilor de funcționare.

Pentru că aplicația noastră se bazează pe o paradigmă client server, atunci a fost realizată și o diagramă de activități care descrie serverul, inserată mai jos.

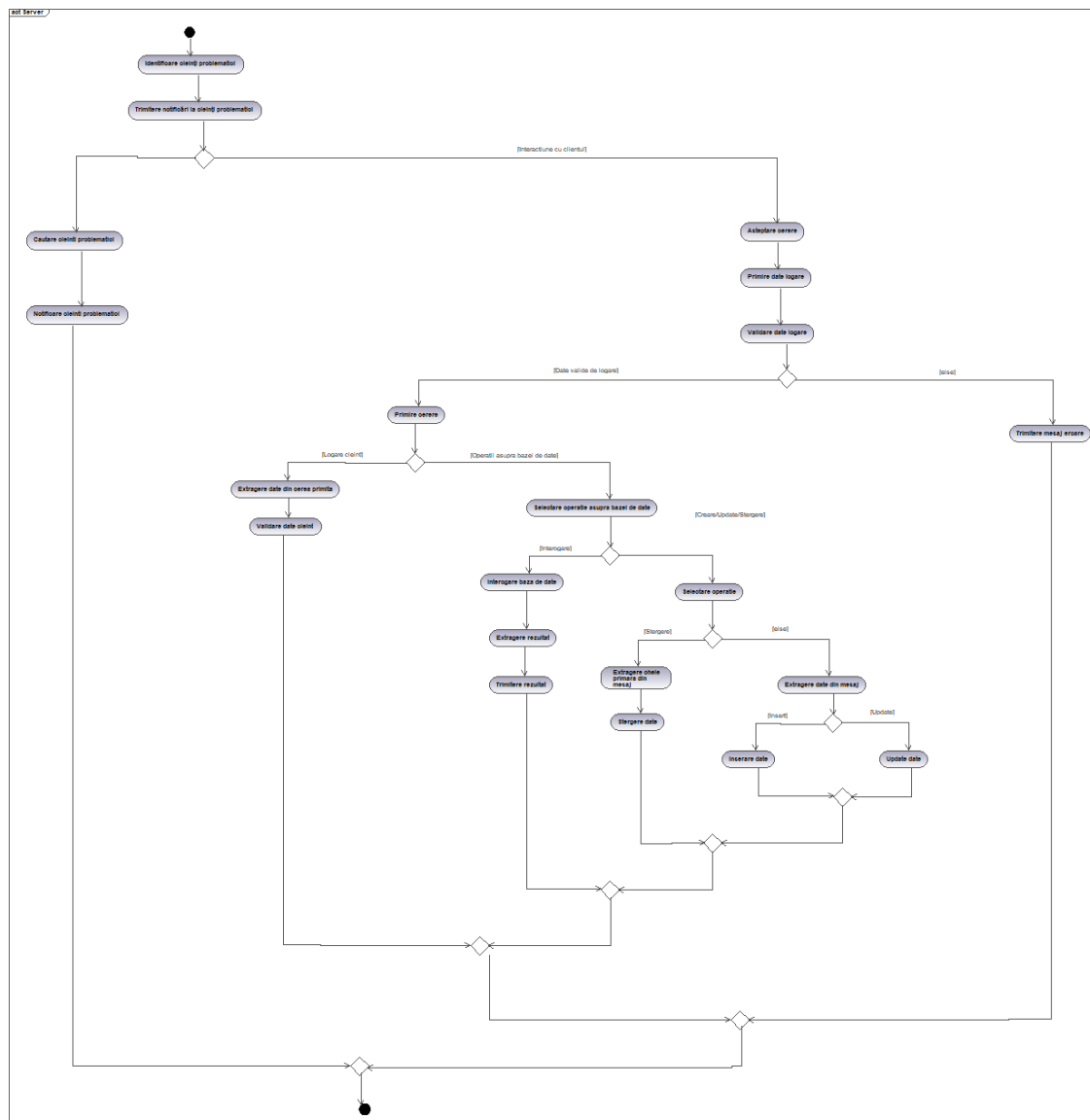
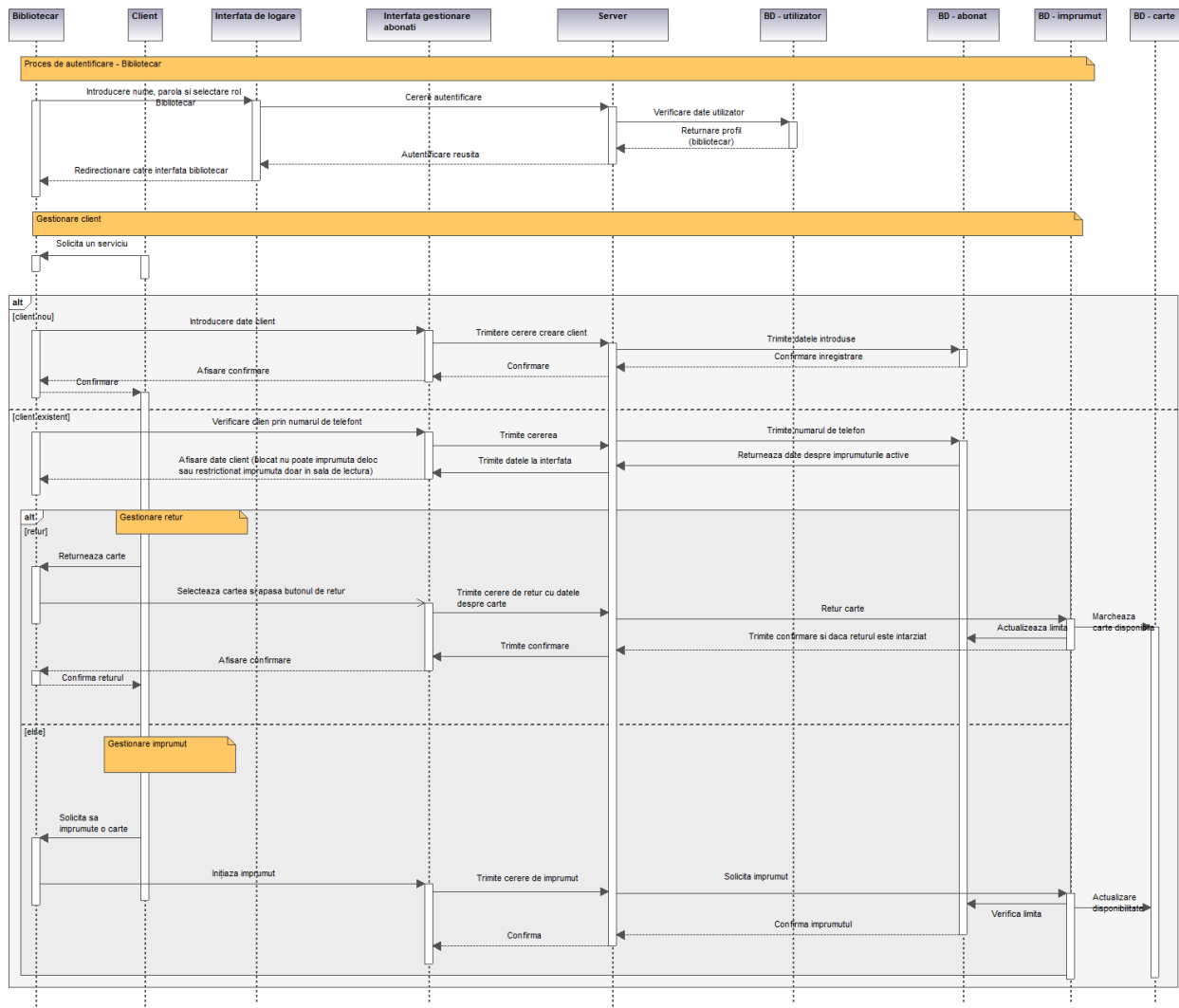


Diagrama descrie pașii parcurși de server pentru procesarea unei cereri. Operațiile sunt grupate pe baza operațiilor CRUD realizate asupra bazei de date. Acestea sunt reprezentate într-o manieră generală care deoarece toate operațiile care se realizează sunt: inserare, ștergere, update și ștergere. Operație de notificare este realizată de un șablon de proiectare de tip Observer, care este notificat când se găsește un client problematic. Parcurgerea bazei de date se realizează o dată 24 de ore.

Pentru a facilita înțelegerea modului în care funcționează aplicația s-au realizat 3 diagrame: una pentru bibliotecar, una pentru administrator și una pentru server.

Diagrama pentru bibliotecar este reprezentată mai jos.



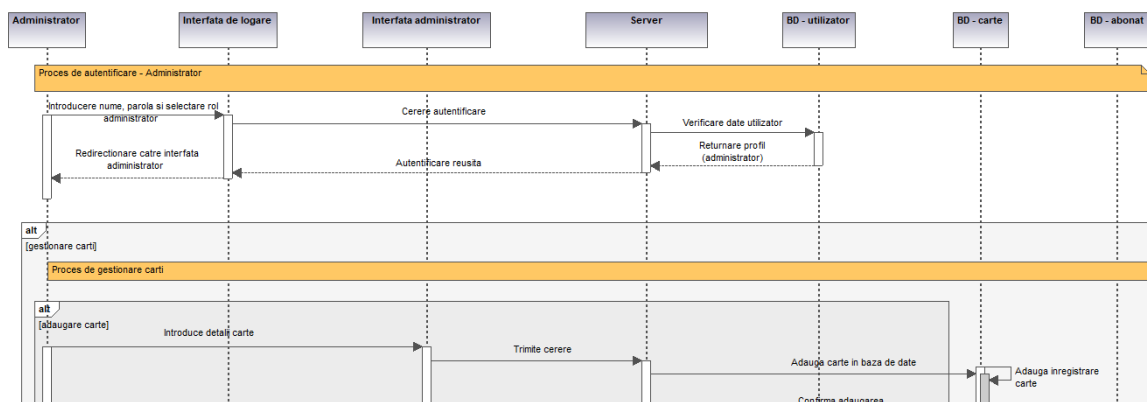
Generated by UModel

www.altova.com

Figură 10 Diagramă de secvențe bibliotecar

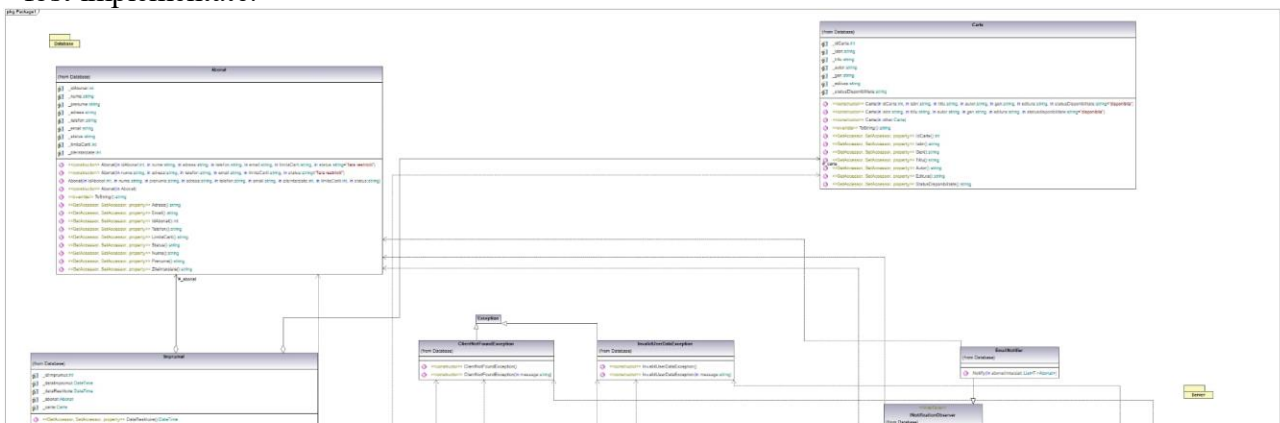
În această diagramă se ilustrează toate entitățile active care interacționează cu bibliotecarul, de la cele software cum ar fi gestionarea bazei de date până la clientul care interacționează cu bibliotecarul.

După ce a fost realizată diagrama de secvențe pentru bibliotecar a fost realizată diagrama de secvențe pentru administrator care este prezentată mai jos:



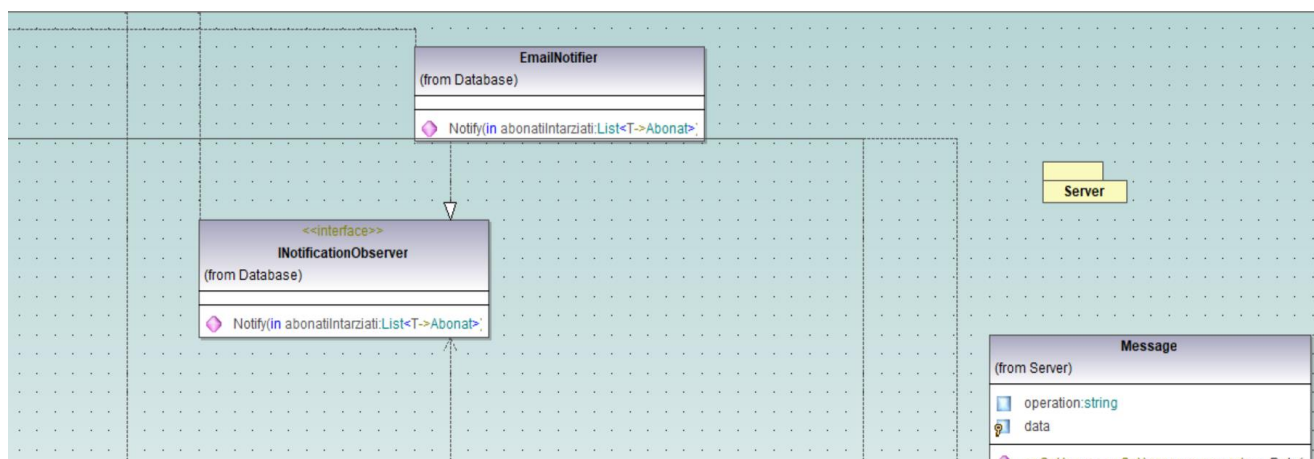
**Figură 11 Diagramă de secvențe administrator**

După ce s-au realizat aceste diagrame, care oferă o imagine amplă și complexă asupra sistemului s-a realizat diagrama de clase, care descrie care sunt entitățile încapsulate sub formă de obiecte care au fost implementate.



**Figură 12 Diagrama de clase**

În cadrul acestei diagrame sunt acoperite componentele aplicației: serverul și clientul care la rândul său este compus din 2 componente: interfața și un backend care realizează comunicarea cu serverul și realizează unele operații asupra răspunsului serverului.



**Figură 13 Diagrama UML pentru șablonul Observer**

### **Interfața Observer**

- Definim INotificationObserver cu o metodă Notify(List<Abonat> abonatiIntarziati).
- Orice clasă care vrea să „asculte” modificările implementează această interfață.

### **Clasa Subject (NotificationSubject)**

- Păstrează o listă privată de observatori (\_observers).
- Are un câmp intern cu starea de transmis (\_abonatiIntarziati).
- Metoda AddObserver(...) permite oricărei instanțe de INotificationObserver să se înregistreze.
- Când datele se schimbă, codul apelează SetAbonatiIntarziati(...) pentru a actualiza lista de abonați întârziati, apoi NotifyObservers().
- NotifyObservers() parcurge fiecare observator și îi apelează Notify(...), transmițând lista actualizată.

### **Observer concret (EmailNotifier)**

- De exemplu, EmailNotifier implementează INotificationObserver.
- În Notify(...), iterează prin lista de abonați întârziati și trimite (sau afișează în consolă) mesaje de notificare.

- Subiectul nu știe cum trimite observatorul notificarea; doar îi cheamă metoda Notify(...).

### **Fluxul general**

La pornirea aplicației, creăm un NotificationSubject și îi adăugăm observator:

```
var subject = new NotificationSubject();  
subject.AddObserver(new EmailNotifier());
```

Când detectăm abonați întârziati (de ex. o dată pe zi), construim List<Abonat> și facem:

```
subject.SetAbonatiIntarziati(lateList);  
subject.NotifyObservers();
```

Acest apel declanșează automat Notify(...) în EmailNotifier, fără ca NotificationSubject să cunoască detalii despre cum anume e trimisă notificarea.

## **Appendix C: Elemente de cod relevante**

După cum s-a menționat mai sus clientul este compus din 2 componente: interfața și backend-ul. Pentru a crea o experiență plăcută s-a ales să se pornească backend-ul automat în momentul în



care se lansează în execuție interfața în mod automat, fără intervenția utilizatorului. Codul care realizează această operație este ilustrat mai jos:

```
string relativePath =
@"..\..\..\ClientBackend\bin\Debug\ClientBackend.exe";
string workerPath =
Path.GetFullPath(Path.Combine(Directory.GetCurrentDirectory(),
relativePath));
bool showConsole = true;

var startInfo = new ProcessStartInfo
{
    FileName = workerPath,
    UseShellExecute = false,
    CreateNoWindow = !showConsole,
    WindowStyle = showConsole ? ProcessWindowStyle.Normal :
ProcessWindowStyle.Hidden
};
Process proc = Process.Start(startInfo);
```

Conectarea la server se realizează prin intermediul unei conexiuni TCP. Astfel, clientul trimite o cerere de conectare la server, iar serverul o acceptă.

Serverul acceptă o conexiune în mod blocant, iar pentru fiecare client se va genera un fir de execuție separat conform codului de mai jos:

```
while (_isRunning)
{
    TcpClient client = _listener.AcceptTcpClient();
    Console.WriteLine("Client connected.");
    Thread thread = new Thread(() => HandleClient(client));
    thread.Start();
}
```

Clientul se conectează la server prin intermediul deschiderii unei conexiuni TCP. Deoarece adresa IP a serverului nu e cunoscută de către client, acesta parcurge o listă de adrese IP în căutarea serverului (operație realizată în constructor). Verificarea se realizează prin încercări, așteptând la fiecare încercare un timp de 50 de milisecunde. Din acest motiv interfața pornește mai târziu de la momentul în care se lansează în execuție programul. Acest proces este ilustrat în codul de mai jos:

```
public ClientBackend()
{
    _port = new TcpListener(System.Net.IPAddress.Any, 8081);
    string subnet = "192.168.137.";
    int port = 12345;

    for (int i = 1; i <= 254; i++)
    {
        string ip = subnet + i;
        try
        {
            Console.WriteLine($"Trying to connect to {ip} on port {port}...");
            _server = new TcpClient();
            var task = _server.ConnectAsync(ip, port);
            if (task.Wait(100))
            {
                Console.WriteLine($"Server found at IP: {ip}");
                break;
            }
        }
        catch { }
    }
}
```

```

        }
    }
    catch
    {
        // I cannot connect to this IP, continue searching
    }
}
}
/// <summary>
/// Acceptă conexiunea de la client și începe să asculte pentru mesaje.
/// </summary>
public void AcceptConnection()
{
    _port.Start();
    while (true)
    {
        TcpClient client = _port.AcceptTcpClient();
        Console.WriteLine("Client connected.");
        Thread thread = new Thread(() => HandleClient(client));
        thread.Start();
    }
}
}

```

## Appendix D: Cazuri de testare

Nr. caz test	Metoda testată	Lista de parametri	Rezultat așteptat	Rezultat obținut	Starea testului
1	InsertUser	testUser = Utilizator("TestUser5", "parola", "bibliotecar")	true și mesaj în consolă "Utilizator adăugat cu succes"	true și mesaj în consolă "Utilizator adăugat cu succes"	Passed
2	InsertUser	testUser = Utilizator("TestUser8", "parola", "administrator")	true și mesaj în consolă "Utilizator adăugat cu succes"	true și mesaj în consolă "Utilizator adăugat cu succes"	Passed
3	InsertUser	testUser = Utilizator("TestUser", "parola", "Inexisting role")	false și mesaj în consolă "Rolul specificat nu există în baza de date"	false și mesaj în consolă "Rolul specificat nu există în baza de date"	Passed
4	InsertUser	testUser = Utilizator("TestUser", "parola", "bibliotecar") (apel dublu)	false și mesaj în consolă "Eroare la inserarea utilizatorului în baza de date: ..."	false și mesaj în consolă "Eroare la inserarea utilizatorului în baza de date: ..."	Passed
5	InsertIsbn	testCarte = Carte(6, "0000000000009", "Carte de test", "Autor de Test", "Gen de test", "Teste pentru toți")	true și mesaj în consolă "Noul isbn a fost adăugat cu succes."	true și mesaj în consolă "Noul isbn a fost adăugat cu succes."	Passed
6	InsertIsbn	testCarte = Carte(0, "00000000001", "Carte de test", "Autor de Test", "Gen de test", "Teste pentru toți"); apel dublu InsertIsbn	true și mesaj în consolă "Isbn-ul există deja"	true și mesaj în consolă "Isbn-ul există deja"	Passed
7	InsertIsbn	testCarte = Carte(0, "0000000000X", null, "Autor de Test", "Gen de test", "Teste pentru toți")	false și mesaj în consolă "Eroare la inserarea în tabela Isbn: ..."	false și mesaj în consolă "Eroare la inserarea în tabela Isbn: ..."	Passed
8	InsertBook	testCarte = Carte(0, "0000000000020", "Carte Test Insert", "Autor Insert", "Gen", "Editura", "disponibil")	return idCarte > 0	return idCarte > 0	Passed

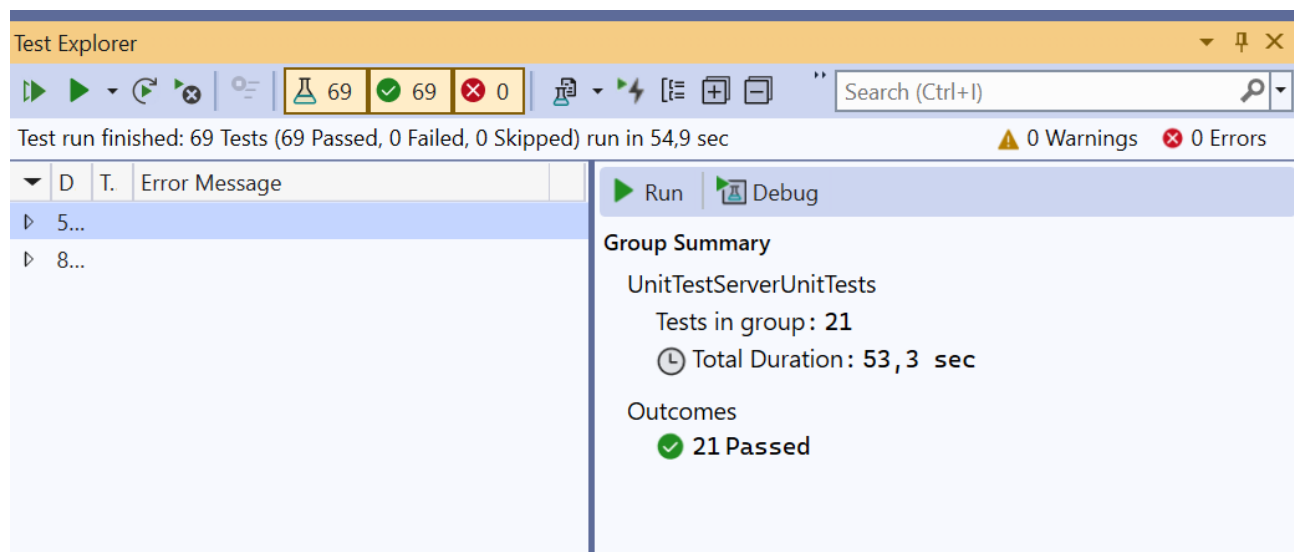
9	InsertBook	testCarte = Carte(0, "0000000000X", null, "Autor Test", "Gen", "Editura", "disponibil")	return -1	return -1	Passed
10	GetCartiByIsbn	inserare carte cu ISBN "00000000000021", apoi GetCartiByIsbn("00000000000021")	listă non-goală, există carte cu titlul "Carte Valid ISBN"	listă non-goală, există carte cu titlul "Carte Valid ISBN"	Passed
11	GetCartiByIsbn	GetCartiByIsbn("ISBN INEXISTENT")	listă goală	listă goală	Passed
12	IsCarteDisponibila	inserare carte cu disponibilitate "disponibil", idCarte; invocare IsCartedisponibil(idCarte)	true	true	Passed
13	IsCarteDisponibila	inserare carte, realizare împrumut, invocare IsCartedisponibil(idCarte)	false	false	Passed
14	IsCarteDisponibila	IsCartedisponibil(-12345)	false	false	Passed
15	InsertLoan	inserare carte și abonat, InsertLoan(abonatId, idCarte, "acasă")	true	true	Passed
16	InsertLoan	inserare carte cu stare "indisponibil", abonat; apel dublu InsertLoan	false la al doilea apel	false la al doilea apel	Passed
17	InsertLoan	InsertLoan(-12345, idCarte, "acasă")	false	false	Passed
18	InsertLoan	InsertLoan(-1, -1, "acasă")	false	false	Passed
19	DeleteUser	inserare utilizator TestUser2; DeleteUser("TestUser2")	true	true	Passed
20	DeleteUser	DeleteUser("User Not Found")	false și mesaj în consolă "Utilizatorul nu a fost găsit sau a fost deja șters"	false și mesaj în consolă "Utilizatorul nu a fost găsit sau a fost deja șters"	Passed

21	DeleteUser	inserare utilizator TestAdminDelete; DeleteUser("TestAdminDelete")	false și mesaj în consolă "Utilizatorul nu a fost găsit sau a fost deja șters"	false și mesaj în consolă "Utilizatorul nu a fost găsit sau a fost deja șters"	Passed
22	DeteteBook	inserare carte cu id valid; DeleteBook(idCarte)	true	true	Passed
23	DeleteBook	DeleteBook(-77777)	false și mesaj în consolă "Nu a fost găsită nicio carte cu acest id."	false și mesaj în consolă "Nu a fost găsită nicio carte cu acest id."	Passed
24	GetAbonatByPhone	inserare abonat; GetAbonatByPhone("0700765432")	obiect Abonat nenul, Nume = "NumeValid"	obiect Abonat nenul, Nume = "NumeValid"	Passed
25	GetAbonatByPhone	GetAbonatByPhone("TELEFONINEXISTENT")	aruncă Database.ClientNotFoundException	aruncă Database.ClientNotFoundException	Passed
26	UpdateStatusAbonat	inserare abonat; UpdateStatusAbonat(abonatId, "cu restricții")	true	true	Passed
27	UpdateStatusAbonat	inserare abonat; UpdateStatusAbonat(abonatId, "statusInexistent")	false	false	Passed
28	UpdateStatusAbonat	UpdateStatusAbonat(-88888, "cu restricții")	false	false	Passed
29	UnrestrictAbonat	inserare abonat cu restricții; UnRestrictAbonat(abonatId)	true	true	Passed
30	UnrestrictAbonat	UnRestrictAbonat(-99999)	false	false	Passed
31	BlocareAbonat	inserare abonat; BlocareAbonat(abonatId)	true	true	Passed
32	BlocareAbonat	BlocareAbonat(-77777)	false	false	Passed
33	RestrictAbonat	inserare abonat; RestrictAbonat(abonatId)	true	true	Passed
34	RestrictAbonat	RestrictAbonat(-33333)	false	false	Passed
35	Login	inserare utilizator TestLogin; Login(testUser)	true	true	Passed

36	Login	Login(Utilizator("Test FailLogin", "gresit", "bibliotecar"))	aruncă Database.InvalidUserDataException	aruncă Database.InvalidUserDataException	Passed
37	InsertClient	abonat = Abonat(..., "0722001234", ...); InsertClient(abonat)	true și obiect nenul la GetAbonatByPhone "0722001234"	true și obiect nenul la GetAbonatByPhone "0722001234"	Passed
38	InsertClient	abonat = Abonat(0, null, "Prenume", ..., "0722002345", ...)	false și mesaj în consolă "Eroare la adăugarea unui abonat în baza de date"	false și mesaj în consolă "Eroare la adăugarea unui abonat în baza de date"	Passed
39	GetLoaned Books	inserare carte și abonat; InsertLoan(...) apoi GetLoanedBooks(abonatlId)	listă conține cartea împrumutată, apoi ReturnBook și ștergere	listă conține cartea împrumutată, apoi ReturnBook și ștergere	Passed
40	CautareCartiPartial	inserare 3 cărți; CautareCartiPartial("Alpha", "Author")	listă cu exact o carte (id1)	listă cu exact o carte (id1)	Passed
41	CautareCartiPartial	CautareCartiPartial("Nonexistent", "Nobody")	listă goală	listă goală	Passed
42	GetStatusAbonat	inserare abonat cu status "cu restricții"; GetStatusAbonat(abonatlId)	"cu restricții"	"cu restricții"	Passed
43	GetStatusAbonat	GetStatusAbonat(-9999)	null	null	Passed
44	ReturnBook	ReturnBook(-1, -1)	false	false	Passed
45	ReturnBook	inserare carte și abonat; InsertLoan(...) apoi ReturnBook(...)	true	true	Passed
46	CautareIntarziati	inserare abonat cu restricții; CautareIntarziati()	lista conține abonatul restricționat	lista conține abonatul restricționat	Passed
47	CautareIntarziati	inserare abonat normal; CautareIntarziati()	lista nu conține abonatul normal	lista nu conține abonatul normal	Passed
48	CautareDoarIntarziati	inserare abonat cu restricții; CautareDoarIntarziati()	listă goală (fără întârziere)	listă goală (fără întârziere)	Passed
49	Login	username=existingUser, password=correctHas	Login successful.	Login successful.	Passed

		h, role=bibliotecar			
50	Login	username=nonexistent, password=wrong, role=bibliotecar	Login failed. Invalid data.	Login failed. Invalid data.	Passed
51	Login	Two consecutive logins with username=existingUser, password=correctHash, role=bibliotecar	First login: Login successful., Second login: Login failed.	First login: Login successful., Second login: Login failed.	Passed
52	RegisterSubscriber	nume=TestSub, prenume=User, adresa=Some Address, telefon=<unique>, email=testsub@example.com	Subscriber Register successful.	Subscriber Register successful.	Passed
53	RegisterSubscriber	nume=Already, prenume=Exists, adresa=Any, telefon=0700111222, email=dup@example.com	Subscriber registration failed.	Subscriber registration failed.	Passed
54	LoginSubscriber	username=0700765432	Subscriber Login successful   <id>   <status>   ...	Subscriber Login successful   <id>   <status>   ...	Passed
55	LoginSubscriber	username=0000000000	Subscriber Login failed	Subscriber Login failed	Passed
56	SearchBooks	title=ZZZNonexistent Title, author=Nobody	No books found.	No books found.	Passed
57	SearchBooks	title=Alpha, author=AuthorMatch	List of matching books in format id~title~author   ...	List of matching books in format id~title~author   ...	Passed
58	InsertLoan	subscriberId=-1, bookId=-1, selectedLocation=acasa	Inserted Loan failed.	Inserted Loan failed.	Passed
59	InsertLoan	subscriberId=<valid>, bookId=<valid>, selectedLocation=acasa	Inserted Loan successful.	Inserted Loan successful.	Passed
60	GetLoans	subscriberId=-1	No loans found.	No loans found.	Passed
61	ReturnBook	subscriberId=-1, bookId=-1	Book return failed.	Book return failed.	Passed
62	GetStatusClient	subscriberId=-1	Status not found.	Status not found.	Passed
63	RegisterEm	username=tempEmp,	Employee registration	Employee	Passed

	ployee	password=pw, role=nonexistentRole	failed.	registration failed.	
64	DeleteLibrarian	username=noSuchUser	Librarian deletion failed.	Librarian deletion failed.	Passed
65	AddBook	isbn=INVALIDISBN, title=null, author=A, genre=G, publisher=P	No response (null)	No response (null)	Passed
66	SearchBook	isbn=0000000000	No books found with the given ISBN.	No books found with the given ISBN.	Passed
67	DeleteBook	idBook=-9999	Book deletion failed.	Book deletion failed.	Passed
68	SearchSubscribers	no parameters	No subscribers found with restrictions or blocked.	No subscribers found with restrictions or blocked.	Passed
69	UpdateStatus	subscriberId=-1, status=cu restrictii	Status update failed.	Status update failed.	Passed



Figură 13 Teste



## Appendix E: Modul de utilizare a programului bazat pe fișierul help

### Descriere generală

Această aplicație este concepută pentru a simplifica și eficientiza activitatea din cadrul unei biblioteci. Prin intermediul ei, angajații bibliotecii pot gestiona cu ușurință catalogul de cărți, pot înregistra împrumuturi și returnări, precum și administra utilizatorii bibliotecii. Aplicația este destinată în special bibliotecarilor și administratorilor, fiecare având roluri și responsabilități bine definite pentru a asigura buna funcționare a bibliotecii:

- **Bibliotecarul** se ocupă de împrumuturi, returnări, verificarea disponibilității cărților și înregistrarea noilor clienți.
- **Administratorul** gestionează catalogul de cărți, aplică sau ridică sancțiuni și administrează conturile angajaților.

### Ce probleme rezolvă aplicația?

- Automatizează procesele de împrumut și retur, eliminând necesitatea operațiunilor manuale.
- Reduce erorile în gestionarea cărților și a clienților.
- Asigură acces rapid la informațiile despre cărți și utilizatori.

Permite o administrare clară și organizată a rolurilor în cadrul bibliotecii.

### Autentificare

Pentru a accesa aplicația, utilizatorul trebuie să se autentifice corect.

#### Pași pentru autentificare:

1. Introduceți numele de utilizator în câmpul dedicat.
2. Introduceți parola în câmpul special pentru parolă.
3. Selectați rolul corespunzător: **Bibliotecar** sau **Administrator**.
4. Apăsați butonul „**Login**” pentru a intra în aplicație.

The screenshot shows a window titled "Biblioteca - Autentificare". The main heading is "Autentificare". Below it, there is a form titled "Date autentificare". The form contains three input fields: "Nume utilizator:", "Parola:", and "Rol:". The "Rol:" field has two radio button options: "Bibliotecar" and "Administrator". Below the form is a "Login" button. Red arrows and numbers 1 through 4 point to the input fields and the button respectively, corresponding to the steps in the authentication process.

# Cerințe pentru bibliotecar

## Autentificare

După autentificare, se afișează o nouă interfață cu panoul din dreapta activat.

## Gestionarea abonaților

1. Bibliotecarul poate înregistra un abonat nou, completând următoarele informații: nume, prenume, adresă, număr de telefon și email, apoi apasă butonul „**Înregistrare**”.
2. Pentru găsirea unui abonat existent, bibliotecarul poate căuta după numărul de telefon.
3. Dacă abonatul este blocat, acesta nu poate efectua împrumuturi noi, ci doar poate returna cărți.

The image displays two screenshots of a web application titled "Bibliotecar".

**Left Screenshot:** The "Abonați" panel is active. It contains an "Autentificare" section with a "Număr de telefon" input field and a "Conectare" button. Below it is the "Înregistrare client nou" section with fields for "Nume" (Popescu), "Prenume" (Ana), "Adresă" (Hapau 11), "Număr de telefon" (0747116566), and "Email" (ana@gmail.com). A red arrow points to the "Înregistrare" button. The "Servicii" panel is inactive.

**Right Screenshot:** The "Servicii" panel is active. The "Autentificare" section now shows the "Număr de telefon" field filled with "0747116566" and the "Conectare" button. A red arrow points to the "Conectare" button. The "Înregistrare client nou" section is still visible but inactive.

## Servicii de împrumut și returnare

1. După selectarea unui abonat valid, în partea dreaptă se activează panoul Servicii cu opțiunile disponibile.
2. Pentru împrumutare carte, bibliotecarul poate căuta o carte după titlu sau autor (căutare parțială). După apăsarea butonului „**Căutare**”, apare o listă cu sugestii de cărți.
3. Se selectează locul în care abonatul dorește să împrumute cartea: **Acasă** sau **Sala de lectură**, apoi se apasă butonul „**Validare**” pentru confirmare.
4. Pentru returnarea cărților, după autentificarea abonatului prin numărul de telefon, bibliotecarul apasă butonul „**Căutare**” pentru a vedea lista cărților nereturnate.
5. Se selectează cartea care se returnează și se apasă butonul „**Validare**” pentru înregistrarea returului.

## Delogare

Bibliotecarul poate încheia sesiunea de lucru prin apăsarea butonului „**Delogare**”, situat în partea stângă jos a interfeței.

## Cerințe pentru administrator

### Autentificare

După autentificare, administratorul are acces la o interfață structurată în **trei tab-uri principale**.

#### 1. Gestionarea angajaților

##### o Înregistrare angajat nou

Administratorul poate adăuga un angajat completând următoarele câmpuri: **Nume utilizator**, **Parolă** și selectează rolul de **Bibliotecar** sau **Administrator**.

După completare, se apasă butonul „**Înregistrare**” pentru salvarea datelor.

##### o Eliminare angajat

În partea dreaptă a interfeței, administratorul poate elimina **doar bibliotecari**. Se introduce numele angajatului în câmpul **Nume** și se apasă butonul „**Eliminare**”.

## 2. Gestionarea cărților

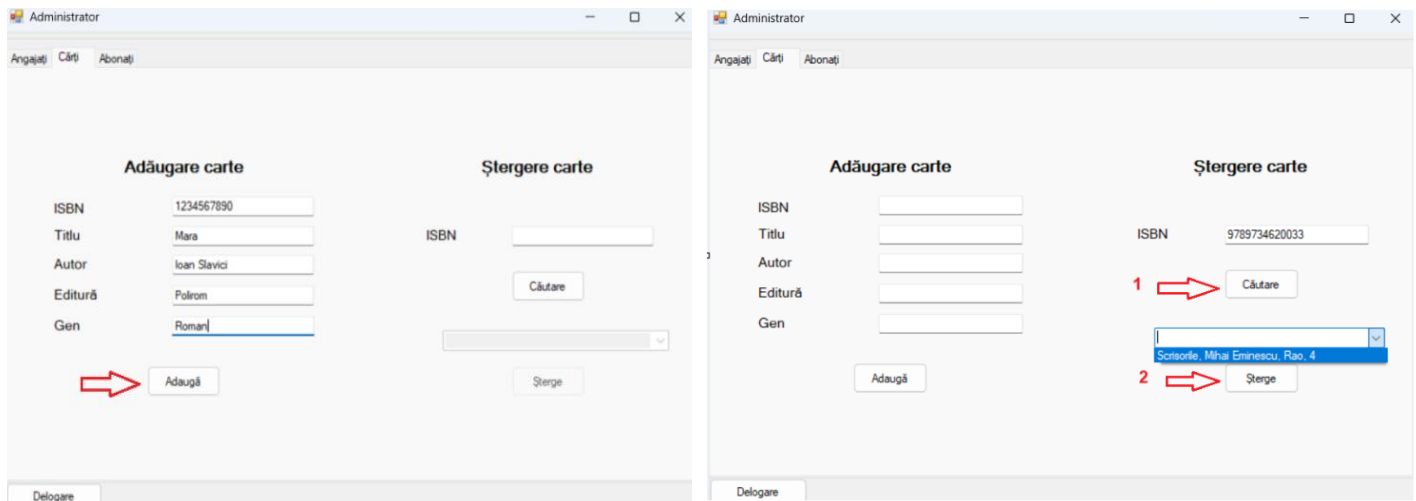
### ○ Adăugare carte nouă

Se completează următoarele câmpuri: **ISBN, Titlu, Autor, Editură și Gen**. După completare, se apasă butonul „**Adaugă**” pentru a înregistra cartea în sistem.

### ○ Ștergere carte

În partea dreaptă, administratorul poate elimina cărți introducând **ISBN-ul** în câmpul corespunzător și apăsând butonul „**Căutare**”.

Va apărea o listă cu rezultate, din care se selectează cartea dorită și se apasă butonul „**Șterge**” pentru confirmare.



## 3. Gestionarea abonaților

### ○ Gestionare abonat individual

Administratorul introduce **numărul de telefon** al abonatului și apasă butonul „**Căutare**”.

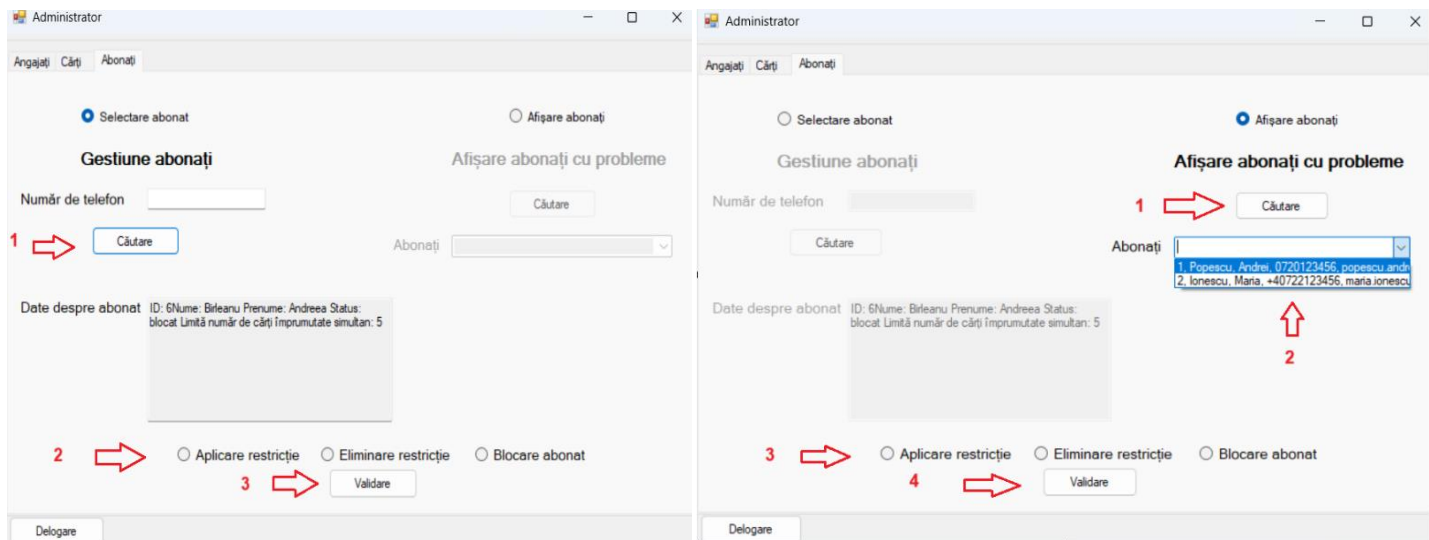
În partea stângă se vor afișa detalii despre abonat, iar administratorul poate: aplica restricții „**Aplicare restricție**”, să elimine restricții „**Eliminare restricție**” sau să blocheze un abonat „**Blocare abonat**”.

### ○ Gestionare abonați cu probleme

În partea dreaptă, administratorul apasă butonul „**Căutare**” pentru a obține lista cu abonații care:

- Sunt restricționați
- Au întârzieri la returnarea cărților

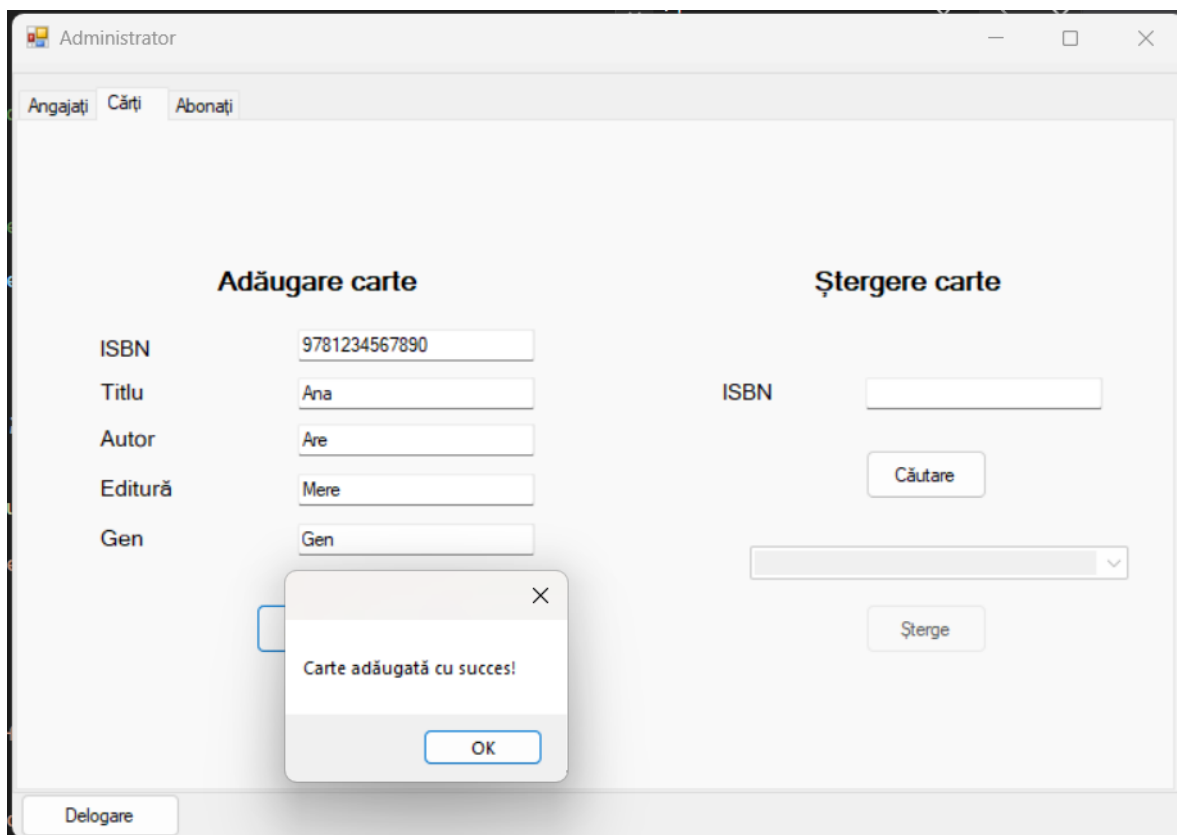
Se poate selecta un abonat din listă și aplica una dintre următoarele acțiuni: „**Aplicare restricție**”, „**Eliminare restricție**” sau „**Blocare abonat**”.



### Delogare

Administratorul poate încheia sesiunea de lucru prin apăsarea butonului „Delogare”, situat în partea stângă jos a interfeței.

## Appendix F: Capturi ecran – cum arată programul în execuție



Administrator

Angajați

Cărți

Abonați

Selectare abonat

Afișare abonați

Gestiune abonați

Afișare abonați cu probleme

Număr de telefon

Căutare

Abonați

1, Popescu, Andrei, 0720123456, popescu.andrei@gmail.com  
2, Ionescu, Maria, +40722123456, maria.ionescu@yahoo.com  
3, Ionescu, Vasile, 0730005530, ionescu@gmail.com  
5, Ionescu, Laura, 0751386931, laura@yahoo.com

Date despre abonat

ID: 1   Nume: Popescu   Prenume: Andrei  
Status: cu restricții   Limită număr de cărți  
Împrumutate simultan: 2

Aplicare restricție

Eliminare restricție

Blocare abonat

Validare

Delogare

Bibliotecar

Abonați

Servicii

Autentificare

Număr de telefon

Conectare

Împrumutare carte

Titlu

Autor

Căutare

Înregistrare client nou

Nume

Prenume

Adresă

Număr de telefon

Email

Înregistrare

Returmare carte

Căutare

Cărți nereturnate

Validare

Abonatul are restricții! Nu poate împrumuta cărți acasă.

OK

Delogare

Administrator

Angajați Cărți Abonați

### Înregistrare angajat nou

Nume utilizator

Parola

Rol ☐ Administrator ☐ Bibliotecar

### Eliminare angajat (bibliotecar) existent

Nume utilizator

×

Autentificare reușită ca Administrator!

Administrator

Angajați Cărți Abonați

☒ Selectare abonat ☐ Afișare abonați

### Gestiune abonați

Număr de telefon

Abonați

☐ Aplicare restricție ☐ Eliminare restricție ☐ Blocare abonat

×

Autentificare eșuată! Nu există niciun abonat cu acest număr de telefon.

Administrator

Angajați

Cărți

Abonați

☒ Selectare abonat

☐ Afișare abonați

Gestiune abonați

Afișare abonați cu probleme

Număr de telefon

Căutare

Abonați

Date despre abonat

ID: 1

Nume: Popescu

Prenume: Andrei

Status: cu restricții

Limită număr de cărți împrumutate simultan: 2

☐ Aplicare restricție

☐ Eliminare restricție

☐ Blocare abonat

Validare

Delogare

Bibliotecar

Abonați

Servicii

Autentificare

Număr de telefon

Conectare

Înregistrare client nou

Nume

Prenume

Adresă

Număr de telefon

Email

Înregistrare

Delogare

Împrumutare carte

Titlu

a

Autor

Căutare

Sugestii

☐ Acasă

Amintiri din copilărie, Ion Creanga, 1

Baltagul, Mihail Sadoveanu, 5

Luceafărul, Mihai Eminescu, 7

Mara, Ioan Slavici, 8

Validare

Returnare carte

Căutare

Cărți nereturnate

Validare



Administrator

Angajați Cărți Abonați

### Adăugare carte

ISBN

Titlu

Autor

Editură

Gen

Adaugă

### Ștergere carte

ISBN

Căutare

Șterge

Delogare

Administrator

Angajați Cărți Abonați

### Adăugare carte

ISBN

Titlu

Autor

Editură

Gen

Adaugă

### Ștergere carte

ISBN

Căutare

Șterge

Delogare

×

Carte ștersă cu succes!

OK

Administrator

Angajați

Cărți

Abonați

☐ Selectare abonat

☒ Afișare abonați

Gestiune abonați

Afișare abonați cu probleme

Număr de telefon

Căutare

Abonați

2, Ionescu, Maria, +40722123456, maria.ion

Date despre abonat

ID: 1

Nume: Popescu

Prenume: Andrei

Status: cu restricții

Limită număr de cărți împrumutate simultan: 2

☐ Aplicare restricție

☒ Eliminare restricție

☐ Blocare abonat

Validare

Statusul abonatului a fost actualizat cu succes!

OK

Delogare

## Lista de cerințe

### 1. Aeloaiei Denisa-Valentina:

- Interfața
- Diagrama de use-case
- Clasele corespondente interfeței: AdminView, BibliotecarView, MainView

### 2. Agavriloaei Marina:

- Diagrama entitate relație
- Diagrama de clase
- Testele (69)
- Șablonul de proiectare (Observer)

### 3. Bîrleanu Andreea:

- Diagramele de secvențe
- Help-ul cu chm
- Help-ul cu Doxygen
- Clase de excepții: ClientNotFoundException, InvalidUserDataException
- Clase implementate: Database, Utilizator, Abonat, Imprumut, Carte

### 4. Stroici Andrei:

- Diagrama de activități
- Documentație
- Clase implementate: Server, ClientBackend

Comentariile și anteturile au fost adăugate de către autorul codului.

## 2. Principiile și condițiile de organizare a testelor

### 1. Testare modulară și atomică

- Fiecare metodă de test (`[TestMethod]`) verifică un singur scenariu de funcționare sau de eroare.
- Se urmărește ca, în cazul unui eșec pe parcursul unui test, diagnosticul să fie clar (de ex. mesajul de eroare returnat de metodele din DLL, excepția aruncată etc.).

### 2. Izolarea resurselor și resetarea stării

- Pentru clasa `Database.Database`, la începutul fiecărui test se deschide o conexiune spre o bază de date „curată” (`ClearDatabase/biblioteca.db`) aflată în folderul de testare.
- După fiecare test, conexiunea se închide în `[TestCleanup]` și, acolo unde a fost inserat ceva temporar (de ex. utilizator, carte, abonat), elementele sunt șterse explicit pentru a reveni la starea inițială.
- Pentru clasa `Server.Server`, înainte de fiecare test se resetează instanța singleton `Database._database`, iar baza de date e adusă la starea de referință („`ClearDatabase`”). De asemenea, la final, serverul este oprit și utilizatorii inserați temporar (ex. `existingUser`) sunt șterși.

### 3. Principiul „Arrange–Act–Assert”

- *Arrange*: se pregătește datele de intrare (de ex. se construiesc obiecte `Utilizator`, `Carte` sau `Abonat` cu atribute specifice).
- *Act*: se apelează metoda din clasa under test (fie din `Database.Database`, fie printr-un mesaj transmis serverului).
- *Assert*: se validează condițiile așteptate:
  - Dacă metoda trebuie să returneze `true/false`, se face `Assert.IsTrue(...)` sau `Assert.IsFalse(...)`.
  - Dacă metoda aruncă o excepție, se utilizează atributul `[ExpectedException]`.
  - Dacă metoda scrie ceva în consolă (ex. notificări de succes/eroare), capturăm output-ul în `StringWriter` și verificăm conținutul cu `StringAssert.Contains(...)`.

### 4. Acoperirea scenariilor pozitive și negative

- Pentru fiecare funcționalitate de bază (inserare, ștergere, căutare, împrumut etc.) există cel puțin un test care validează comportamentul așteptat („happy path”) și cel puțin un test care verifică comportamentul în situații eronate:
  - Date invalide (ex. `null`, stringuri nevalide, ID-uri negative).
  - Date care violează constrângeri de integritate (ex. dublarea unei chei unice).
  - Entități inexistente (ex. ștergerea unui utilizator inexistent, împrumuturi pe ID-uri invalide).

### 5. Testare black-box

- Pentru clasa `Database`, testele se fac apelând exclusiv metodele publice (în cazul unor metode private cum este `IsCartedisponibil`, se folosește reflexie pentru a le invoca și a valida rezultatul).
- Pentru clasa `Server`, testele trimit mesaje JSON către server (prin `TcpClient`) și verifică răspunsurile primite pe socket (astfel simulând cererile efective ale unui client).
-

#### 6. Filtrarea pe baze de roluri și restricții

- În cazul gestionării rolurilor de utilizator, există teste care verifică inserarea unui `Utilizator` cu rol valid („bibliotecar”, „administrator”) și cu rol inexistent (trebuie să eșueze).
- În cazul abonaților, se verifică inserarea și excepțiile în funcție de telefon unic sau `null`.

### 3. Obiectivele testării și ce se urmărește a fi verificat

#### • Conectarea și gestiunea tranzacțiilor

Verificarea faptului că se poate deschide conexiunea către baza SQLite, că tranzacțiile pentru inserarea unei cărți sunt atomic executate, iar commit/rollback-urile funcționează în cazul erorilor.

#### • Inserarea de entități cu date valide

- **Utilizator:** roluri valide („bibliotecar”, „administrator”) → metoda `InsertUser(...)` returnează `true` și afișează „Utilizator adăugat cu succes”.
- **Isbn** (metoda `InsertIsbn`) → verificare format, inserare într-o singură tranzacție, afișare mesaj de succes.
- **Carte** (metoda `InsertBook`) → inserare a înregistrării în tabelele `Isbn` și `Carte` concomitent, obținerea `id_carte`.
- **Abonat** (metoda `InsertClient`) → inserarea unui abonat nou, valabilitate telefon unic.

#### • Gestionarea excepțiilor și a condițiilor de eroare

- Inserare utilizator cu rol inexistent → metoda `InsertUser` returnează `false`, afișează „Rolul specificat nu exista în baza de date”.
- Inserare dublă de utilizator cu același `nume_user` → aruncă o eroare SQL (constrângere unică), se prinde și se afișează un mesaj corespunzător („Eroare la inserarea utilizatorului în baza de date: ...”).
- Inserare ISBN cu date invalide (`null` la câmp obligatoriu) → metoda `InsertIsbn` aruncă o excepție SQLite, se prinde și se returnează `false`.
- Inserare Carte cu date invalide (ISBN incorect sau date lipsă) → metoda `InsertBook` returnează `-1`.
- Inserare Client cu `nume null` → metoda `InsertClient` returnează `false` și afișează „Eroare la adăugarea unui abonat în baza de date: ...”.
- Ștergere utilizator inexistent sau administrator → `DeleteUser` returnează `false`, afișează „Utilizatorul nu a fost găsit...”.
- Ștergere carte cu `id_carte` invalid sau statut diferit de „disponibil” → `DeleteBook` returnează `false` și afișează „Nu a fost găsită nicio carte cu acest id.”

#### • Funcționalități de căutare

- **GetCartiByIsbn:**
  - ISBN valid și carte disponibilă → se reîntoarce cel puțin un obiect `Carte` cu acele atribute.
  - ISBN invalid (sau tabelă goală) → metoda returnează listă goală.
- **GetLoanedBooks:**
  - După efectuarea unui împrumut, `GetLoanedBooks` returnează lista de cărți nereturnate.
  - Pentru un abonat inexistent sau fără împrumuturi active → listă goală (sau `null`, tratat ca listă nepopulată).

- **Funcționalități de împrumut și returnare**
  - **InsertLoan:**
    - Dacă cartea este disponibilă și abonatul există (ID valid) → returnează `true`, setează `status = 'indisponibil'` și decrementarea limitei abonatului.
    - Dacă cartea e deja „indisponibil” sau abonatul nu există → returnează `false` și se afișează un mesaj corespunzător.
  - **ReturnBook:**
    - Dacă există un împrumut activ (`Data_restituire = null`), se actualizează `data_restituire`, se setează `status='disponibil'` și se incrementează limita abonatului. Returnează `true`.
    - Dacă nu există un împrumut activ pentru acea combinație (ID\_invalid), se face rollback și returnează `false`, cu afișarea mesajului „Nu există un împrumut activ pentru această carte.”
- **Verificarea disponibilității cărților (metodă privată)**
  - `IsCartedisponibil(int idCarte)` (invocată prin reflexie):
    - ID valid și `status='disponibil'` în tabelă → `true`.
    - ID invalid sau `status!='disponibil'` sau eroare SQL → `false`.
- **Gestionarea statusului abonat**
  - **GetStatusAbonat(int idAbonat):**
    - ID valid → returnează stringul „blocat”, „cu restricții” sau „fara restricții” (sau `null` dacă ID invalid).
  - **UpdateStatusAbonat(int idAbonat, string mesaj):**
    - ID valid și mesaj fiabil („blocat”, „cu restricții”, „fara restricții”) → `true` și mesaj de succes.
    - ID invalid sau mesaj inexistent (în lipsa înregistrării) → `false`.
  - **UnRestrictAbonat, BlocareAbonat, RestrictAbonat:** reiau `UpdateStatusAbonat` cu mesajul corespunzător.
- **Detectarea abonaților întârziati sau cu restricții**
  - **CautareIntarziati():** lista care cuprinde abonații cu:
    - fie `status = „cu restricții”`,
    - fie cel puțin un împrumut cu `deadline < azi` și `data_restituire IS NULL`.
  - **CautareDoarIntarziati():** listează doar abonații cu împrumuturi active ce au `deadline < azi` (fără să includă cei cu „cu restricții” care nu au împrumut întârziat).
  - Testele verifică:
    - Prezența unui abonat cu restricții în `CautareIntarziati` (trebuie inclus).
    - Excluderea unui abonat fără nicio întârziere din `CautareIntarziati` (trebuie absent).
    - Dacă nu există întârziere, `CautareDoarIntarziati` → listă goală.
- **Funcționalități de server (componenta `Server.Server`)**
  - Se testează prin trimiterea mesajelor JSON structurate în obiecte `Message` și verificarea răspunsului textual primit:
    1. **login** (autentificarea angajatului):
      - Credențiale valide → „Login successful.”.
      - Credențiale invalide → „Login failed. Invalid data.”.
      - Al doilea login consecutive pe același `username` → al doilea apel eșuează („Login failed.”).
    2. **registerSubscriber:**

- Număr de telefon unic → „Subscriber Register successful.”.
  - Număr de telefon duplicat → „Subscriber registration failed.”.
3. **loginSubscriber:**
- Telefon existent → răspuns de forma „Subscriber Login successful|<IdAbonat>|<Status>|...”.
  - Telefon inexistent → „Subscriber Login failed”.
4. **searchBooks:**
- Fără potriviri → „No books found.”.
  - Cu potriviri (există cel puțin o carte parțial corespondentă titlu/autor) → returnează un șir concatenat cu separator „~” și „|” (ex. id~titlu~autor|...).
5. **insertLoan:**
- ID abonat și ID carte valide și carte disponibilă → „Inserted Loan successful.”.
  - ID-uri invalide sau carte indisponibilă → „Inserted Loan failed.”.
6. **getLoans:**
- Pentru abonat cu împrumuturi active → șir concatenat similar celui de la `searchBooks` (ex. idCarte~Titlu~Autor|...).
  - Pentru abonat fără împrumuturi → „No loans found.”.
7. **returnBook:**
- Pentru abonat/cartă cu împrumut activ → „Book returned successful.”.
  - În caz contrar („No active loan”) → „Book return failed.”.
8. **getStatusClient:**
- ID abonat valid → „Status found: <status>”.
  - ID invalid → „Status not found.”.
9. **registerEmployee (InsertUser):**
- Rol valid → „Employee registered successful.”.
  - Rol invalid → „Employee registration failed.”.
10. **deleteLibrarian (DeleteUser):**
- Utilizator non-administrator existent → „Librarian deleted successful.”.
  - Utilizator inexistent sau administrator → „Librarian deletion failed.”.
11. **addBook (InsertBook + InsertIsbn):**
- Date valide → „Book added successful.”.
  - Date invalide (ex. titlu null) → mesaj null (metoda nu mai trimite nimic – clientul primește null response).
12. **searchBook (GetCartiByIsbn):**
- ISBN valid → răspuns formatat idCarte~Titlu~Autor~Editura|....
  - ISBN inexistent → „No books found with the given ISBN.”.
13. **deleteBook (DeleteBook):**
- ID carte valid și carte disponibilă → „Book deleted successful.”.
  - ID invalid → „Book deletion failed.”.

#### 14. `searchSubscribers` (CautareIntarziati):

- Există abonați cu restricții/întârziere → răspuns formatat  
`id~nume~prenume~...|...`
- Nu există astfel de abonați → „No subscribers found with restrictions or blocked.”.

#### 15. `updateStatus` (UpdateStatusAbonat):

- ID abonat valid → „Status updated successful.”.
- ID invalid → „Status update failed.”.

## 4. Grupurile de teste pentru clasa `Database.Database`

Pentru `Database.Database`, testele au fost împărțite în următoarele categorii (grupuri), conform funcționalităților implementate:

### 4.1. Teste pentru inserarea și autentificarea utilizatorilor

- **`InsertUserSuccessfullyBibliotecar`**
  - **Scop:** Verifică că un utilizator cu rol „bibliotecar” (rol valid) poate fi inserat cu succes.
  - **Ce se verifică:**
    - Metoda `InsertUser` returnează `true`.
    - Pe `Console.Out` apare șirul „Utilizator adaugat cu succes”.
    - La final, utilizatorul test (`TestUser5`) este șters.
- **`InsertUserSuccessfullyAdministrator`**
  - **Scop:** Verifică inserarea cu succes a unui utilizator cu rol „administrator” (rol valid).
  - **Ce se verifică:**
    - `InsertUser` returnează `true`.
    - Mesajul de consolă conține „Utilizator adaugat cu succes”.
- **`InsertUserRoleNotFound`**
  - **Scop:** Verificarea scenariului în care rolul specificat nu există în tabelă.
  - **Ce se verifică:**
    - `InsertUser` returnează `false`.
    - Mesajul de consolă conține „Rolul specificat nu exista in baza de date”.
- **`InsertUserSqlError`**
  - **Scop:** Verifică gestionarea situației în care se încearcă inserarea a doi utilizatori cu același `nume_user` (constrângere unică SQL).
  - **Ce se verifică:**
    - Primul `InsertUser(testUser)` returnează `true`.
    - Al doilea apel `InsertUser(testUser)` returnează `false`.
    - Mesajul de consolă indică „Eroare la inserarea utilizatorului in baza de date: ...”.
- **`DeleteUserSuccessfully`**
  - **Scop:** Ștergerea unui utilizator non-administrator existent.
  - **Ce se verifică:**



- `InsertUser` pe `TestUser2` → `true`.
  - `DeleteUser("TestUser2")` → `true` (metoda `DeleteUser` returnează `true`).
- **DeleteUserUserNotFound**
  - **Scop:** Încercarea de ștergere a unui utilizator inexistent.
  - **Ce se verifică:**
    - `DeleteUser("User Not Found")` returnează `false`.
    - Mesajul de consolă conține „Utilizatorul nu a fost gasit sau a fost deja sters”.
- **DeleteUserSqlError**
  - **Scop:** Verifică scenariul în care se încearcă ștergerea unui administrator (constrângere: nu se poate șterge administrator).
  - **Ce se verifică:**
    - Se inserează `TestAdminDelete` cu rol „administrator”.
    - Apel `DeleteUser("TestAdminDelete")` → `false`.
    - Mesajul de consolă conține „Utilizatorul nu a fost gasit sau a fost deja sters sau este administrator.”.
- **LoginSuccessfully**
  - **Scop:** Autentificarea cu date valide pentru utilizator (rol „bibliotecar”).
  - **Ce se verifică:**
    - `InsertUser("TestLogin", ..., "bibliotecar")` → `true`.
    - Apel `Login(...)` → `true`.
    - La final, utilizatorul `TestLogin` este șters.
- **LoginUnsuccessfully**
  - **Scop:** Verificarea aruncării excepției `InvalidUserDataException` când datele de login sunt incorecte (parola greșită).
  - **Ce se verifică:**
    - Apel `Login("TestFailLogin", "gresit", "bibliotecar")` → aruncă `InvalidUserDataException`.

## 4.2. Teste pentru gestionarea ISBN-urilor (tabela `ISBN`)

- **InsertIsbnSuccessfully**
  - **Scop:** Inserarea cu succes a unui nou ISBN.
  - **Ce se verifică:**
    - Apel `InsertIsbn(testCarte)` cu `Isbn="00000000000009"` → `true`.
    - Mesajul de consolă conține „Noul isbn a fost adaugat cu succes.”
- **InsertIsbnAlreadyExisting**
  - **Scop:** Inserarea unui ISBN deja prezent în bază.
  - **Ce se verifică:**
    - Primul apel `InsertIsbn(testCarte)` cu `Isbn="000000000001"` → `true`.
    - Al doilea apel `InsertIsbn(testCarte)` → `true` (metoda recunoaște existența și afișează „Isbn-ul exista deja”).
    - Mesajul de consolă conține „Isbn-ul exista deja”.
- **InsertIsbnSqlError**
  - **Scop:** Inserarea unui ISBN cu date invalide (titlu `null`).
  - **Ce se verifică:**
    - Apel `InsertIsbn(testCarteCuTitluNull)` → `false`.

- Mesajul de consolă conține „Eroare la inserarea in tabela Isbn: ...”.

### 4.3. Teste pentru gestionarea cărților (tabela `carte`)

- **InsertBookSuccessfully**
  - **Scop:** Inserarea cu succes a unei cărți noi (se presupune că ISBN-ul nu există în prealabil).
  - **Ce se verifică:**
    - Apel `InsertBook(testCarte)` → returnează `idCarte > 0`.
- **InsertBookSqlError**
  - **Scop:** Inserarea unei cărți cu date invalide (titlu `null`).
  - **Ce se verifică:**
    - Apel `InsertBook(testCarteCuTitluNull)` → returnează `-1`.
- **GetCartiByIsbnIsbnValid**
  - **Scop:** Obținerea cărților după un ISBN valid care tocmai a fost inserat.
  - **Ce se verifică:**
    - Se inserează `Carte testCarte` cu ISBN "00000000000021" → obținem `idCarte`.
    - Apel `GetCartiByIsbn("00000000000021")` → lista returnată are `Count > 0` și conține cel puțin un obiect cu `Titlu == "Carte Valid ISBN"`.
- **GetCartiByIsbnIsbnInvalid**
  - **Scop:** Apel `GetCartiByIsbn("ISBNINEXISTENT")` pentru un ISBN care nu există.
  - **Ce se verifică:**
    - Metoda returnează o listă cu `Count == 0`.

### 4.4. Teste pentru disponibilitatea cărților (metodă privată `IsCartedisponibil`)

- **IsCarteDisponibilaTrueCarte**
  - **Scop:** Pentru o carte proaspăt inserată cu `status="disponibil"`, `IsCartedisponibil(idCarte)` ar trebui să returneze `true`.
  - **Ce se verifică:**
    - Se apelează prin reflexie metoda privată `IsCartedisponibil(...)` și se obține `true`.
- **IsCarteDisponibilaFalse**
  - **Scop:** Dacă cartea a fost împrumutată (prin `InsertLoan`), `IsCartedisponibil(idCarte)` returnează `false`.
  - **Ce se verifică:**
    - Se inserează o carte cu `status="disponibil"`, apoi un abonat nou și se face `InsertLoan(...)` → cartea devine indisponibil.
    - Apel `IsCartedisponibil(idCarte)` → `false`.
- **IsCarteDisponibilaIdInvalid**
  - **Scop:** Apelul cu un `idCarte` invalid (negativ) trebuie să returneze `false`.
  - **Ce se verifică:**
    - Prin reflexie se apelează `IsCartedisponibil(-12345)` → `false`.

#### 4.5. Teste pentru împrumuturi și returnări (tabelele `Imprumut` și `Carte`)

- **InsertLoanSuccessfully**
  - **Scop:** Inserarea cu succes a unui împrumut când cartea este disponibilă și abonatul există.
  - **Ce se verifică:**
    - Se inserează o carte disponibilă și un abonat nou (apel `InsertClient`).
    - `InsertLoan(idAbonat, idCarte, "acasa") → true`.
- **InsertLoanCarteIndisponibila**
  - **Scop:** Încercare de împrumut când cartea este deja marcată `status="indisponibil"`.
  - **Ce se verifică:**
    - Se inserează o carte cu `status="indisponibil"` manual (la crearea obiectului `Carte`).
    - Se creează un abonat și se face un `InsertLoan` → cartea devine „indisponibil” (procedeul intern nu permite `InsertLoan` pentru cartea deja indisponibilă).
    - Se face al doilea apel `InsertLoan(...)` → `false`.
- **InsertLoanAbonatInexistent**
  - **Scop:** Încercarea de împrumut cu un `idAbonat` invalid (nu există în tabelă).
  - **Ce se verifică:**
    - `InsertLoan(-12345, idCarteValabil, "acasa") → false`.
- **InsertLoanSqlError**
  - **Scop:** Încercarea de împrumut cu ambele ID-uri invalide (-1, -1).
  - **Ce se verifică:**
    - `InsertLoan(-1, -1, "acasa") → false`.
- **ReturnBookNoActiveLoanReturnsFalse**
  - **Scop:** Apelarea `ReturnBook` când nu există un împrumut activ pentru combinația dată.
  - **Ce se verifică:**
    - `ReturnBook(-1, -1) → false`.
- **ReturnBookAfterLoanReturnsTrue**
  - **Scop:** După ce s-a efectuat un împrumut, `ReturnBook` trebuie să returneze `true`.
  - **Ce se verifică:**
    - Se inserează carte și abonat, se face `InsertLoan(...)`, apoi `ReturnBook(...)` → `true`.

#### 4.6. Teste pentru gestionarea clienților (tabela `Abonat`)

- **GetAbonatByPhoneValidPhone**
  - **Scop:** Obținerea unui abonat după telefon când acesta există.
  - **Ce se verifică:**
    - Se inserează un abonat cu telefon valid („0700765432”).
    - `GetAbonatByPhone("0700765432") → obiect Abonat` nenul și `Nume == "NumeValid"`.
- **GetAbonatByPhoneInvalidPhone**
  - **Scop:** Așteptare excepție `ClientNotFoundException` când telefonul nu există.

- **Ce se verifică:**
  - `GetAbonatByPhone("TELEFONINEXISTENT")` → aruncă `ClientNotFoundException`.
- **InsertClientSuccessfully**
  - **Scop:** Inserarea cu succes a unui abonat nou cu date valide.
  - **Ce se verifică:**
    - `InsertClient(abonatNouCuTelefonUnic)` → `true`.
    - Apoi se apelează `GetAbonatByPhone(...)` și se verifică că atributul `Nume` este corect („ClientTest”).
- **InsertClientSqlErrorNullName**
  - **Scop:** Inserarea unui abonat cu **nume = null** (câmp obligatoriu).
  - **Ce se verifică:**
    - `InsertClient(abonatCuNumeNull)` → `false`.
    - Mesajul de consolă conține „Eroare la adaugarea unui abonat in baza de date: ...”.

#### 4.7. Teste pentru căutări și filtre (tabelele `Isbn`, `Carte`, `Imprumut`, `Abonat`)

- **CautareCartiPartialReturnsMatchingBooks**
  - **Scop:** Verifică căutarea parțială după titlu și autor.
  - **Ce se verifică:**
    - Se inserează trei cărți:
      1. "AlphaTitle" + "AuthorMatch"
      2. "BetaTitle" + "AuthorMatch"
      3. "GammaTitle" + "OtherAuthor"
    - Apel `CautareCartiPartial("Alpha", "Author")` → lista conține exact o carte (ID-ul lui `c1`).
    - Titlul și ID-ul se potrivesc cu `c1`.
- **CautareCartiPartialNoMatchesReturnsEmptyList**
  - **Scop:** Când nu există rezultate pentru cuvintele parțiale.
  - **Ce se verifică:**
    - `CautareCartiPartial("Nonexistent", "Nobody")` → listă cu `Count == 0`.
- **GetLoanedBooksAfterLoanReturnsBook**
  - **Scop:** După efectuarea unui împrumut, `GetLoanedBooks` returnează cartea împrumutată.
  - **Ce se verifică:**
    - Se inserează `Carte` și `Abonat`, apoi se face `InsertLoan`.
    - `GetLoanedBooks(idAbonat)` → listă nenulă și conține cartea cu `IdCarte` și `Titlu` potrivite.

#### 4.8. Teste pentru statusul abonatului și abonații întârziati/with restrictions

- **UpdateStatusAbonatAbonatValidMesajValid**
  - **Scop:** Actualizarea statusului (cu `restrictii`) pentru un abonat existent.
  - **Ce se verifică:**
    - Se inserează un abonat nou.
    - `UpdateStatusAbonat(idAbonat, "cu restrictii")` → `true`.
- **UpdateStatusAbonatAbonatValidMesajInvalid**

- **Scop:** Dacă mesajul nu există (nu corespunde niciunei valori permise), metoda returnează false.
  - **Ce se verifică:**
    - Se inserează un abonat, apoi se apelează `UpdateStatusAbonat(idAbonat, "statusInexistent")` → false.
- **UpdateStatusAbonatAbonatInvalid**
  - **Scop:** ID abonat invalid → `UpdateStatusAbonat(-88888, "cu restrictii")` → false.
- **UnrestrictAbonatAbonatValid**
  - **Scop:** Eliminarea restricțiilor pentru un abonat existent (`UnRestrictAbonat`).
  - **Ce se verifică:**
    - Se inserează un abonat cu `status="cu restrictii"`.
    - `UnRestrictAbonat(idAbonat)` → true.
- **UnrestrictAbonatAbonatInvalid**
  - **Scop:** ID invalid → `UnRestrictAbonat(-99999)` → false.
- **BlocareAbonatAbonatValid**
  - **Scop:** Blocarea unui abonat existent (`BlocareAbonat`).
  - **Ce se verifică:**
    - Se inserează un abonat cu `status="fara restrictii"`.
    - `BlocareAbonat(idAbonat)` → true.
- **BlocareAbonatAbonatInvalid**
  - **Scop:** ID invalid → `BlocareAbonat(-77777)` → false.
- **RestrictAbonatAbonatValid**
  - **Scop:** Aplicarea restricțiilor unui abonat existent (`RestrictAbonat`).
  - **Ce se verifică:**
    - Se inserează un abonat cu `status="fara restrictii"`.
    - `RestrictAbonat(idAbonat)` → true.
- **RestrictAbonatAbonatInvalid**
  - **Scop:** ID invalid → `RestrictAbonat(-33333)` → false.
- **GetStatusAbonatValidReturnsCorrectStatus**
  - **Scop:** Obținerea statusului unui abonat când ID-ul e valid.
  - **Ce se verifică:**
    - Se inserează un abonat cu `status="cu restrictii"`.
    - `GetStatusAbonat(idAbonat)` → returnează "cu restrictii".
- **GetStatusAbonatInvalidIdReturnsNull**
  - **Scop:** ID invalid → `GetStatusAbonat(-9999)` → null.

## 4.9. Teste pentru căutarea abonaților întârziati/with restrictions

- **CautareIntarziatiRestrictedClientIncluded**
  - **Scop:** Verifică că un abonat cu `status="cu restrictii"` este returnat de `CautareIntarziati()`.
  - **Ce se verifică:**
    - Se inserează un abonat cu `status="cu restrictii"`.
    - `CautareIntarziati()` → lista conține cel puțin abonatul inserat.
- **CautareIntarziatiNoIssuesReturnsEmptyList**
  - **Scop:** Un abonat fără întârziere (status „fara restrictii”) nu trebuie returnat în `CautareIntarziati()`.
  - **Ce se verifică:**

- Se inserează un abonat cu `status="fara restrictii"` și fără împrumuturi întârziate.
  - `CautareIntarziati()` → listă dar nu include abonatul inserat (se verifică `Exists(a => a.Telefon == ...) == false`).
- **CautareDoarIntarziatiNoOverdueReturnsEmptyList**
  - **Scop:** Pentru un abonat cu restricții dar fără întârziere efectivă, `CautareDoarIntarziati()` trebuie să returneze listă goală.
  - **Ce se verifică:**
    - Se inserează un abonat cu `status="cu restrictii"` care nu are împrumuturi.
    - `CautareDoarIntarziati()` → `Count == 0`.

## 5. Grupurile de teste pentru clasa `Server.Server`

Pentru `Server.Server`, testele au fost organizate în funcție de fiecare operațiune pe care serverul o primește și o procesează, prin mesaje JSON de tip `Message`. Implementarea serverului grupează cererile pe câmpul `operation`. Următoarele categorii de teste se regăsesc în `UnitTestServerUnitTests`:

### 5.1. Teste pentru autentificarea și gestionarea sesiunii de angajați

- **LoginValidCredentialsReturnsSuccess**
  - **Operă:** „login”
  - **Date de intrare:**

```
{
  "operation": "login",
  "data": [ { "username": "existingUser", "password": "correctHash", "role": "bibliotecar" } ]
}
```

  - - **Scop:** Autentificare angajat cu credențiale valide, rol „bibliotecar” pre-inserat (`existingUser`).
    - **Ce se verifică:**
      - Răspunsul serverului conține „Login successful.”.
- **LoginInvalidCredentialsReturnsFailInvalidData**
  - **Operă:** „login”
  - **Date de intrare:** „nonexistent” / „wrong” / „bibliotecar”.
  - **Scop:** Autentificare cu credențiale invalide.
  - **Ce se verifică:**
    - Răspunsul conține „Login failed. Invalid data.”.
- **LoginDuplicateAttemptReturnsFailOnSecond**
  - **Operă:** „login” (trimis de două ori consecutiv, cu aceleași date valabile).
  - **Scop:** Verifică că primul mesaj obține „Login successful.”, iar al doilea (pe aceeași conexiune) obține „Login failed.”.
  - **Ce se verifică:**
    - Răspunsul primului apel conține „Login successful.”.
    - Răspunsul celui de-al doilea apel conține „Login failed.”.

## 5.2. Teste pentru înregistrarea și autentificarea abonaților

- **RegisterSubscriberNewPhoneReturnsSuccess**
  - **Operă:** „registerSubscriber”
  - **Date de intrare:** abonat cu număr de telefon unic (gen „099900123456”).
  - **Scop:** Înregistrare abonat nou.
  - **Ce se verifică:**
    - Răspunsul conține „Subscriber Register successful.”.
- **RegisterSubscriberDuplicatePhoneReturnsFail**
  - **Operă:** „registerSubscriber”
  - **Date de intrare:** abonat cu telefon deja existent în bază (0700111222).
  - **Scop:** Înregistrare abonat cu telefon duplicat.
  - **Ce se verifică:**
    - Răspunsul conține „Subscriber registration failed.”.
- **LoginSubscriberExistingPhoneReturnsSuccessData**
  - **Operă:** „loginSubscriber”
  - **Date de intrare:** telefonul 0700765432, deja inserat în [TestInitialize].
  - **Scop:** Autentificarea abonatului după telefon.
  - **Ce se verifică:**
    - Răspunsul conține șirul „Subscriber Login successful|<IdAbonat>|<Status>|...”.
- **LoginSubscriberInvalidPhoneReturnsFail**
  - **Operă:** „loginSubscriber”
  - **Date de intrare:** telefon inexistent (0000000000).
  - **Scop:** Autentificare eșuată pentru abonat.
  - **Ce se verifică:**
    - Răspunsul conține „Subscriber Login failed”.

## 5.3. Teste pentru căutarea și afișarea cărților

- **SearchBooksNoMatchReturnsNoBooks**
  - **Operă:** „searchBooks”
  - **Date de intrare:**  
{ "title": "ZZZNonexistentTitle", "author": "Nobody" }.
  - **Scop:** Căutare fără potriviri.
  - **Ce se verifică:**
    - Răspunsul conține „No books found.”.
- **SearchBooksMatchExistsReturnsList**
  - **Operă:** „searchBooks”
  - **Date de intrare:** { "title": "Alpha", "author": "AuthorMatch" } (având cel puțin o carte în baza de date cu aceste parțialități).
  - **Scop:** Căutare cu potriviri.
  - **Ce se verifică:**
    - Răspunsul conține caracterul „~” (adică a fost returnat un șir formatat id~Titlu~Autor|...).

## 5.4. Teste pentru împrumuturi (operațiunea „insertLoan”)

- **InsertLoanInvalidIdsReturnsFail**
  - **Operă:** „insertLoan”
  - **Date de intrare:** {"subscriberId": "-1", "bookId": "-1", "selectedLocation": "acasa"}.
  - **Scop:** Încercarea de inserare a unui împrumut cu ID-uri invalide.
  - **Ce se verifică:**
    - Răspunsul conține „Inserted Loan failed.”.
- **InsertLoanValidReturnsSuccess**
  - **Etape:**
    1. **addBook:** se trimite mesaj „addBook” pentru a crea o carte temporară cu ISBN="TEMPISBN123".
      - Răspuns: „Book added successful.”
    2. **registerSubscriber:** se înregistrează un abonat temporar (telefon unic, ex. „099900...”).
      - Răspuns: „Subscriber Register successful.”
    3. **searchBook:** se caută cartea după isbn="TEMPISBN123".
      - Răspuns: "<bookId>~TempTitle~TempAuthor~TempPub|" → extragem bookId.
    4. **loginSubscriber:** se autentifică abonatul folosind telefonul temporar.
      - Răspuns: "Subscriber Login successful|<subscriberId>|..." → extragem subscriberId.
    5. **insertLoan:** se trimite {"subscriberId": "<subscriberId>", "bookId": "<bookId>", "selectedLocation": "acasa"}.
      - **Scop:** Verifică împrumutul în scenariul normal (abonat și carte valide).
      - **Ce se verifică:** Răspunsul conține „Inserted Loan successful.”.

## 5.5. Teste pentru obținerea împrumuturilor (operațiunea „getLoans”)

- **GetLoansNoActiveReturnsNoLoans**
  - **Operă:** „getLoans”
  - **Date de intrare:** {"subscriberId": "-1"} (abonat inexistent).
  - **Scop:** Când nu există împrumuturi active pentru abonatul specificat.
  - **Ce se verifică:** Răspunsul conține „No loans found.”.

## 5.6. Teste pentru returnarea unei cărți (operațiunea „returnBook”)

- **ReturnBookNoActiveReturnsFail**
  - **Operă:** „returnBook”
  - **Date de intrare:** {"subscriberId": "-1", "bookId": "-1"} (împrumut inexistent).
  - **Scop:** Încercarea de returnare când nu există împrumut.
  - **Ce se verifică:** Răspunsul conține „Book return failed.”.

## 5.7. Teste pentru interogarea statusului clientului (operațiunea „getStatusClient”)

- **GetStatusClientInvalidIdReturnsNotFound**



- **Operă:** „getStatusClient”
- **Date de intrare:** { "subscriberId": "-1" }.
- **Scop:** Solicitare status pentru abonat inexistent.
- **Ce se verifică:** Răspunsul conține „Status not found.”.

## 5.8. Teste pentru gestionarea angajaților prin server (operațiunile „registerEmployee” și „deleteLibrarian”)

- **RegisterEmployeeInvalidRoleReturnsFail**
  - **Operă:** „registerEmployee”
  - **Date de intrare:** { "username": "tempEmp", "password": "pw", "role": "nonexistentRole" }
  - **Scop:** Înregistrare angajat cu rol inexistent.
  - **Ce se verifică:** Răspunsul conține „Employee registration failed.”.
- **DeleteLibrarianInvalidUserReturnsFail**
  - **Operă:** „deleteLibrarian”
  - **Date de intrare:** { "username": "noSuchUser" }.
  - **Scop:** Ștergerea unui bibliotecar care nu există sau este administrator (deci nu poate fi șters).
  - **Ce se verifică:** Răspunsul conține „Librarian deletion failed.”.

## 5.9. Teste pentru adăugarea și ștergerea cărților prin server (operațiunile „addBook”, „searchBook”, „deleteBook”)

- **AddBookInvalidDataReturnsNullResponse**
  - **Operă:** „addBook”
  - **Date de intrare:** { "isbn": "INVALIDISBN", "title": null, "author": "A", "genre": "G", "publisher": "P" }.
  - **Scop:** Inserare carte cu date invalide (title=null).
  - **Ce se verifică:**
    - Serverul scrie în consolă „Book addition failed.”, dar **nu trimite nicio linie** către client → din perspectiva testului, `SendSingle(...)` returnează `null`.
- **SearchBookNoMatchReturnsNoBooks**
  - **Operă:** „searchBook”
  - **Date de intrare:** { "isbn": "000000000000" } (ISBN inexistent).
  - **Scop:** Căutare carte după ISBN inexistent.
  - **Ce se verifică:** Răspunsul conține „No books found with the given ISBN.”.
- **DeleteBookInvalidIdReturnsFail**
  - **Operă:** „deleteBook”
  - **Date de intrare:** { "idBook": "-9999" }.
  - **Scop:** Ștergerea unei cărți cu `id_carte` invalid.
  - **Ce se verifică:** Răspunsul conține „Book deletion failed.”.

## 5.10. Teste pentru căutarea abonaților cu restricții/întârziere (operațiunea „searchSubscribers”)

- **SearchSubscribersNoneMatchReturnsNoSubscribers**
  - **Operă:** „searchSubscribers”
  - **Date de intrare:** listă goală (nu se transmit date suplimentare).
  - **Scop:** Nu există abonați cu restricții sau întârziere.
  - **Ce se verifică:** Răspunsul conține „No subscribers found with restrictions or blocked.”.

### 5.11. Teste pentru actualizarea statusului abonatului (operațiunea „updateStatus”)

- **UpdateStatusInvalidSubscriberReturnsFail**
  - **Operă:** „updateStatus”
  - **Date de intrare:** {"subscriberId": "-1", "status": "cu restrictii"}.
  - **Scop:** Încercarea de actualizare a statusului pentru un abonat inexistent.
  - **Ce se verifică:** Răspunsul conține „Status update failed.”.

## 6. Concluzii

- Toate funcționalitățile principale (CRUD, autentificare, împrumuturi, căutări) sunt acoperite de cel puțin un test pozitiv („happy path”) și un test negativ (eroare SQL, date invalide sau entități inexistente).
- Testele unitare asigură robustețea codului, validând că în situații de eroare nu există scurgeri de date, tranzacții neterminate sau state inconsistente.
- Prin abordarea black-box (pentru subclass-le private) și simularea traficului client-server (pentru `Server.Server`), s-a asigurat că API-ul expus de librării funcționează conform așteptărilor, iar clienții pot interpreta corect răspunsurile.
- Structura testelor, organizată pe grupuri de funcționalități (Utilizatori, ISBN, Cărți, Abonați, Împrumuturi, Căutări, Server), permite întreținerea ușoară: la adăugarea unei noi metode în DLL, e simplu de identificat în ce categorie va trebui adăugat testul.