UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENŢĂ

**ReaderLiSh**

propusă de

***Buzemurgă Mihaela***

**Sesiunea:** februarie*, 2016*

Coordonator ştiinţific

**Asist. Dr. Vasile Alaiba**

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

**FACULTATEA DE INFORMATICĂ**

**ReaderLiSh**

***Buzemurgă Mihaela***

**Sesiunea:** *februarie, 2017*

Coordonator ştiinţific

**Asist. Dr. Vasile Alaiba**

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA

DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*ReaderLiSh*” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

* toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
* reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
* codul sursă, imaginile etc. preluate din proiecte *open*-*source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
* rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,

Absolvent *Buzemurgă Mihaela*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că nu sunt de acord ca Lucrarea de licență cu titlul „*ReaderLiShr*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de testetc. ) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, nu sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iași să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent *Buzemurgă Mihaela*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ReaderLiSh**

*„Cartea reflectă ca o oglindă lungul şir de secole al vieţii omenirii, istoria luptei sale pentru existenţă, pentru un viitor mai luminos, suferinţele, bucuriile, înfrângerile şi biruinţele sale toate. Iubiţi cartea, îngrijiţi-o și citiţi cât mai mult. Cartea ne este prieten credincios, de nădejde.”*

*(G. F. Morozov)*

# Motivație

Sunt o amatoare a cărților, în special a celor de dezvoltare personală. Consider că niciodata nu ar trebui să ne oprim din citit, acesta reprezentând principalul stimulent al creierului. Știind că nu sunt singura care gândește așa și că mai sunt și alte persoane care împărtășesc această idee, m-am gândit la o aplicație care să vină în ajutorul celor care doresc să citească și în momentele în care nu iși pot lua o carte cu ei. După cum știm, tehnologia a avansat într-un ritm destul de alert, iar telefonul a devenit indispensabil în zilele noastre. Așadar, aplicația va fi destinată celor care dispun de un dispozitiv mobil cu Android.

ReaderLiSh este un concept de aplicație care dispune de mai multe functionalitați. În primul rând aplicația permite citirea cărților pe care tu singur le incarci în baza de date, sau a celor deja existente, adaugate de administrator. În al doilea rand, aplicația permite cu ajutorul GPS-ului găsirea de biblioteci și librarii în zona în care te afli, sau o alta selectată de utilizator. Ceea ce aduce nou aplicația, față de cele deja existente, este una dintre optiunile în care iti poti cauta o carte după o anumită stare de spirit. Dacă utilizatorul dorește într-un moment de supărare să citească o carte care să îi ridice moralul și să îi îmbunătățească starea de spirit, va putea foarte usor să scrie sentimentul “fericire” sau “bucurie” sau orice alt astfel de sentiment, iar aplicația îi va returna o serie de cărți care fac parte din categoria aleasa. Pe langă aceste functionalitati, ReaderLiSh va functiona și ca un fel de rețea de socializare. Citind o carte, un utilizator îi poate acorda un calificativ (Like) și o poate recomanda și celorlalti utilizatori care folosesc această aplicație.

Toate aceste facilități le-am îmbinat într-un singur concept, ReaderLiSh, nume compus din principalele funcționalități ale aplicației: Read, Like & Share.

# Context

Un prim argument în ceea ce privește rolul lecturii în dezvoltarea personală este efectul pe care-l exercită o carte asupra unui individ: prin lecturarea unei cărți, omul își îmbogățește lumea interioară prin intermediul imaginației. Desigur, există și multe filme bazate pe anumite cărți, însă avantajul citirii unei opere este că fiecare își poate imagina într-un anumit fel un personaj, un loc sau poate să judece în felul său un eveniment.

Un al doilea argument în favoarea lecturi este faptul că aceasta oferă subiecte de meditație și reprezintă o sursă imensă de idei, idei care pot fi împărtășite altor oameni sau chiar aplicate în viața de zi cu zi. Astfel, un anumit paragraf dîntr-o carte de psihologie poate să reprezinte un răspuns la o întrebare care demult sălășuiește în mintea individului. Ideea principală a unui roman poate fi privită din mai multe puncte de vedere, la fel și acțiunile protagonistului sau chiar decorul, iar aceste aspecte ale unei cărți pot fi discutate la infinit cu un om ce a citit la rândul său opera dată.

În concluzie, consider că lectura este un aspect important în viața individului, indiferent de forma pe care o are: carte, revistă, ziar, carte audio. În dezvoltarea personală aceasta joacă rolul de sursă de informație, ajută la antrenarea imaginației și reprezintă o formă de socializare și descoperire a oamenilor ce împart aceeași pasiune.[[1]](#endnote-1)

# Obiectiv

Îmi propun realizarea unei aplicații care să vină în ajutorul personelor pasionate de citit, persoane care doresc să-și imbunățatească cultura generală prin lecturarea unor cărti de pe un dispozitiv mobil. Aplicația este destinată celor care nu dispun de foarte mult timp liber, dar chiar și așa, acest timp vor să îl investească într-un mod cât mai inteligent.

# Cerințe funcționale

În momentul în care un utilizator își va descărca și instala aplicația, își va crea un cont în baza de date. Contul va fi completat și personalizat în funcție de preferințele fiecărui utilizator.

Principalul scop al aplicației este citirea cărților. Așadar, utilizatorii vor putea citi cărțile puse la dispoziție de către aplicație in momentul instalării, sau le vor putea citi pe cele încarcate de ei.

Odată deschisa o carte, utilizatorii vor putea face și mici adnotări asupra textului. Ei vor putea sublinia un paragraf, vor putea pune un semn la o anumită pagină și desigur să selecteze și anumite fragmente din carte care li se par mai interesante, pentru ca mai apoi să le poată găsi cu ușurință.

Prin activarea GPS-ul, aplicația va avea activă optiunea de a căuta bibliotecile și librăriile din zona în care se afla, sau o altă zonă setată de utilizator. Aplicația pune la dispoziție si această facilitate, deoarece încurajeaza si cititul cărților sub formă tipărită.

După citirea fiecărei cărti, utilizatorul poate aprecia sau nu cartea, printr-un calificativ (Like).

Așa cum mentionam la inceput, aplicația funcționează și ca o rețea de socializare. Utlizatorii care citesc anumite cărți vor putea împărtăși impresiile cu ceilalti utilizatori ai aplicației. De asemenea ei pot face și anumite recomandari pe baza cărților citite.

.

# Capitolul I

# Platformă de dezvoltare:

* Platforma de dezvoltare Android

Android este una din cele mai renumite platforme de dezvoltare și un sistem de operare pentru dispozitive și telefoane mobile. Prin intermediul acestuia se pot realiza aplicații pentru milioane de dispozitive din întreaga lume.

Aplicațiile mobile joacă un rol foarte important in piață in această perioadă.Drept urmare, atât producătorii de telefoane, furnizorii de platforme, cât și operatorii de rețea se află în competiție pentru a realiza cea mai mare comunitate de dezvoltare.

* SDK-ul Android

SDK-ul Android conține un set de instrumente de dezvoltare, precum biblioteci, program de depanare, un emulator de dispozitiv, documentație, mostre de cod și tutoriale menite să simplifice dezvoltarea de aplicații mobile. Platformele de dezvoltare sprijinite în prezent înglobează calculatoare pe X86 care rulează Linux, Mac OS sau mai recent Windows 7 sau 8.1.4

Îmbunătățirile aduse la SDK-ul Android facilitează dezvoltarea de aplicații destinate primelor versiunii ale platformei Android, astfel se pot realiza aplicații și pentru dispozitivele mai vechi. Instrumentele pentru dezvoltare pot fi descărcate în funcției de platforma și versiunea dorită pentru a se asigura compatibilitatea acestora cu dispozitivele pentru care sunt realizate.

# Tehnologii folosite:

Pentru dezvoltarea aplicției “ReaderLiSh ” au fost folosite următoarele tehnologii:

* Sistemul de operare Android
* Google Maps API
* Server Java
* Fire de execuție
* SQLite

# Sistemul de operare Android

*Android* este un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux, dezvoltată inițial de compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance. Android permite dezvoltatorilor să scrie cod gestionat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor Java dezvoltate de Google. Aplicații scrise în C și în alte limbaje pot fi compilate în cod mașină ARM și executate, dar acest model de dezvoltare nu este sprijinit oficial de către Google. Lansarea platformei Android, la 5 noiembrie 2007, a fost anunțată prin fondarea Open Handset Alliance, un consorțiu de companii de hardware, software și de telecomunicații, consacrat dezvoltării de standarde deschise pentru dispozitive mobile. Google a lansat cea mai mare parte a codului Android sub licența Apache, o licență de tip free-software și open-source.

Începând cu 21 octombrie 2008, Android a fost disponibil ca *Open Source*. Google a deschis întregul cod sursă, care anterior era disponibil sub licența Apache. Aceasta permite producătorilor să adauge extensii proprietare, fără a le face disponibile comunității open source. În timp ce contribuțiile Google la această platformă se așteaptă să rămână open source, numărul versiunilor derivate ar putea exploda, folosind o varietate de licențe.

# Arhitectura sistemului de operare: Android

În continuare aș dori să prezint arhitectura *sistemului de operare Android* care cuprinde cinci secțiuni pe patru nivele, acestea fiind: Kernelul Linux, biblioteci, motorul Android și cadrul pentru aplicații.10

*Kernelul Linux* utilizează o serie de patch-uri pentru versiune oficială, și conține toate driverele pentru componentele hardware, precum camera, tastatura, WiFi și dispozitive audio. De asemenea, la Kernelul Linux au fost adăugate unele funcționalități specifice Android, acestea fiind: wakelocks, low-memory killer, binder, logger.

*Wakelocks* este o soluție pentru problema de power management folosită de Android, care are ca scop reducerea consumului prin intrarea în starea de ”sleep” pentru momentele în care nu este utilizat.

În ceea ce privește *low memory killer*, scopul său este de a opri componentele care nu au fost folosite o perioadă lungă de timp.

Următoarea funcționalitate este reprezentată de *binder*, un mecanism de RPC/IPC, care are ca scop capabilitate de invocare remote a obiectelor asemănătoare cu obiectele COM din Windows.11

Și în cele din urmă *logger,* care are ca scop substituirea *sistemului clasic de logging* al kernelului, cu scopul diminuării numărului de task switch-uri și scrieri în fișier, prîntr-un buffer circular.

O altă componentă este *motorul Android*, alcătuit dîntr-o serie de biblioteci de bază, care permit utilizatorilor să dezvolte aplicații mobile, folosind ca limbaj de programare Java; aceste biblioteci permit accesul la funcțiile unui dispozitiv și anume: telefonie, mesaje, gestiunea pachetelor. La baza arhitecturii stau regiștri, fiind echipată cu un compilator JIT, care permite modificarea executabilului obținut pe dispozitivul mobil.

Următoarea componentă este cadrul pentru aplicații: *Android framework*, care furnizează diverse funcționalități ale sistemului de operare pentru ca programatorii să le poată transpune în aplicațiile lor. Această componentă oferă dezvoltatorilor posibilitatea de a realiza aplicații complexe și inovative, întrucât aceștia sunt liberi să utilizeze hardware-ul echipamentelor, de informațiile despre locație, rularea de servicii în background, setarea de alarme, precum și adăugarea de notificări pe bara de stare. Programatorilor li se oferă acces la aceleași API-uri ca și aplicațiile distribuite cu Android, prin urmare arhitectura aplicațiilor este proiectată astfel încât să fie simplificată reutilizarea componentelor: o aplicație poate publica anumite funcționalități, și o altă aplicație să le poată utiliza.

Ultima componentă este reprezentată de *nivelul aplicații*, care oferă atât produsele încorporate în dispozitivele mobile, precum: Camera, Music player, Contacts, și Video player, cât și produsele disponibile pe Play Store.

# Funcționalitățile sistemului de operare Android

Principalele funcționalități pe care sistemul de operare Android le oferă sunt: *stocare*, care folosește SQLite, bază de date relațională ce permite utilizarea eficientă a resurselor; *conectivitatea* prin diverse modalități, precum: 3G, WiFi, Bluetooth, WiMAX, GPRS, EDGE; *WiFi direct,* care permite interconectarea între diverse dispozitive având o lățime de bandă mare; *Android Beam*, prin care utilizatorii partajează conținut instant prin apropierea dispozitivelor respective. O altă funcționalitate este *navigarea pe Internet* bazată pe motorul open source pentru navigare WebKit împreună cu motorul JavaScript de la Chrome V8 suportând HTML5 și CSS3.

*Multimedia* admite mai multe formate precum: H.263, M-peg-4, AMR-WEB, AAC, JPEG; *multi-touch,* care suportă posibilitatea de contact în mai multe puncte concomitent; *multi-tasking*; *GCM*(Google Cloud Messaging) permițând dezvoltatorilor expedierea de date de dimensiuni reduse, în lipsa unei soluții de sincronizare proprietară.

# Google Maps API

Google a lansat în anul 2005 Google Maps API, care permite programatorilor integrarea de hărților de la Google în propriile aplicații; acest serviciu fiind gratuit. API-ul permite accesul la serverele Google Maps, descărcarea de date, afișarea unei hărți, și trimiterea unui răspuns la interacțiunea cu harta.

O altă facilitate pe care API-ul Google Maps o furnizează este inserarea de informații suplimentare despre obiectele de pe o porțiune a hărții, astfel încât să permită interacțiunea utilizatorului cu harta. Totodată, acesta permite adăugarea de obiecte grafice pe hartă, precum: ancore pentru pozițiile specifice pe hartă, cunoscute sub denumirea de markeri, segmente de linii, segmente închise, imagini și elemente grafice de tip bitmap atașate la poziții specifice pe hartă.

Hărțile afișate de Google Maps API prezintă următoarele caracteristici: titlul acestora nu includ conținut personalizat, iar referitor la pictograme, nu toate permit acțiunea de click. În plus față de funcționalitea de cartografie, API sprijină o gamă completă de interacțiuni cu harta, în concordanță cu modelul Android UI, un exemplu ilustrativ este posibilitatea configurării interacțiunilor cu harta, prin definirea de *listeneri*, concept care are ca scop oferirea de răspunsuri la gesturile utilizatorilor. Principala clasă care se utilizează în cazul utilizării hărților este *GoogleMap*. Aceasta modelează harta în cadrul aplicației, iar în cadrul interfeței utilizator, harta va reprezentată prîntr-un MapFragment sau MapView. Clasa *GoogleMap* permite realizarea următoarelor acțiuni: conectarea la serviciul Google Maps, descărcarea componentelor hărții, afișarea de diverse controale, precum zoom și răspunderea la acțiunea de zoom.

*MapFragment* este o subclasă Android, care permite plasarea unei hărți într-un fragment Android și oferă acces la obiecte de tip GoogleMap. Spre deosebire de o extensie de vizualizare (View), un fragment reprezintă un comportament sau o porțiune din interfața utilizator dîntr-o activitate, astfel o activitate poate include mai multe fragmente. De asemenea, un fragment poate fi utilizat în mai multe activități, dar și utilizarea mai multor fragmente pentru realizarea unui interfețe.

În ceea ce privește *MapView* este o subclasă a clasei *Android View*, care permite plasarea hărții într-o extensie de vizualizare. Un View reprezintă o regiune a ecranului, element esențial în dezvoltarea de aplicații Android și widget-uri. Mai mult decât atât, prin intermediul unui MapFragment, MapView-ul acționează asemănător unui container pentru hartă.

# Fire de execuție

Un fir de execuție, numit și *thread,* este o secvență de instrucțiuni care poate fi executată concurent cu o altă secvență, într-un mediu *multithreading,* în timp ce partajează aceeași zonă de memorie. Acest fire de execuție rulează în paralel pentru a crește eficiența programelor. În sistemele cu procesoare multiple sau cu nuclee multiple, firele de execuție rulează în același timp pe procesoare sau nuclee diferite. Pentru procesoarele cu un singur nucleu, sistemul împarte timpul de execuție între *thread*-uri, aspect prezentat în figura de mai jos.



La nivel de implementare se poate utiliza standardul *Pthreads (POSIX Threads)*, ce definește un API pentru crearea și manipularea firelor de execuție. Pentru implementarea unui server TCP care deservește clienți multiplii se utilizează acest standard deoarece aplicațiile ce folosesc fire de execuție sunt în general mai rapide decât dacă utilizează procese.

# SQLite

SQLite este o bibliotecă care implementează o bază de date SQL tranzacțională și care nu are nevoie de o configurație și nici de un server. Codul pentru această librărie este public și poate fi folosit pentru orice scop.

Spre deosebire de alte baze de date SQL, SQLite nu are un server separat pentru procesare. Acesta citește și scrie direct în memoria sistemului. Oferă o bază de date complexă cu posibilitatea de a crea tabele multiple, indici, *trigger*-e și *view*-uri iar tranzacțiile sunt *ACID (Consistency, Isolation, Durability*) chiar dacă sunt întrerupte de erori. De asemenea, formatul fișierelor scrise de aceasta bibliotecă este *cross-platform* oferind posibilitatea de a fi utilizat atât pe sisteme de 32 de biți cât și pe cele de 64 sau pe arhitecturi de tip *big-endian* sau *little-endian*.

Datorită acestor facilitați, SQLite este opțiunea ideală în implementarea unei aplicații performante.

Capitolul II

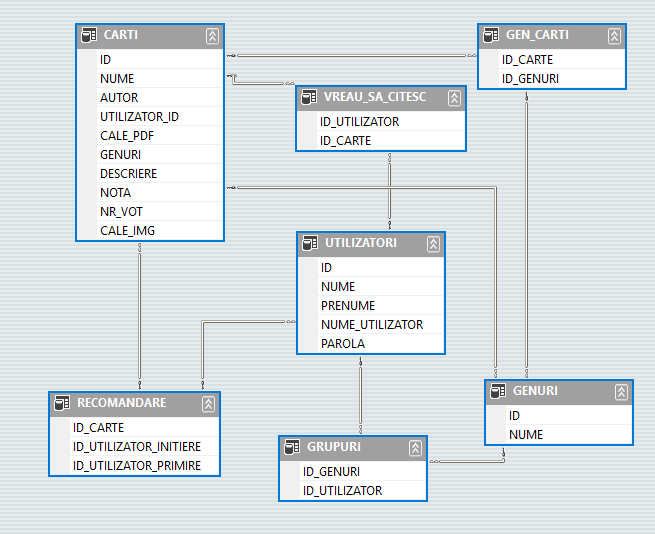
În acest capitol voi evidenția arhitectura aplicatiei ReaderLiSh, principalele module ale aplicațtiei, precum și modul de comunicare a acestora.

În ceea ce privește arhitectura aplicației, aceasta este de tip client-server, după modelul pe două nivele în care primul strat este reprezentat de aplicația client și anume aplicația Android iar cel de-al doilea strat este serverul Java. Pentru stocarea datelor am utilizat baza de date SQLite, ce oferă eficientă.

# Baza de date

Baza de date este formată dintr-un numar de șapte tabele.

* Tabela Utilizatori reține datele utilizatorilor care sunt inregistrați;
* Tabela Carti este tabela în care se rețin informații despre carțile de de server;
* Tabela Genuri stochează genurile cartilor disponibile;
* Tabela Gen\_Carti este o tabelă intermediară care face legătura dintre tabelele Cărți și Genuri; am folosit această tehnică pentru a evita relația “many to many”;
* Tabela Grupuri este cea care reține grupurile din care face parte un utilizator;
* Tabela Recomandare păstreaza toate informațiile legate de recomandările dintre useri;
* Tabela Vreau\_Sa\_Citesc va fi folosita la a reține cărțile pe care un utilizator va dori să le citească în viitor;



# Server

Pentru implementarea server-ului Java am urmărit cât mai multe principii de programare orientată pe obiecte pentru a realiza o aplicație ușor de înțeles, modificat, întreținut și testat.

Astfel, clasele încapsulează datele, ascund reprezentarea și sunt ușor de refolosit. De asemenea, responsabilitățile sunt alocate astfel încât coeziunea în sistem rămâne ridicată. O coeziune ridicată înseamnă că responsabilitățile pentru un element din sistem sunt înrudite și concentrate în jurul aceluiași concept. Totodată, sarcinile sunt împărțite în așa fel încât cuplarea rămâne slabă. O cuplare slabă presupune dependențe puține între clase, impact scăzut în sistem la schimbarea unei clase și potențial ridicat de refolosire.

In Fig. 1 este reprezentată structura serverului care este formată din 14 pachete și anume:

* Pachetul Server conține următoarele clase:
* Clasa ServerMain este clasa principală care instanțiază obiectul ThreadPooledSever: acesta pornește toate firele de execuție destinate clienților. În momentul în care un client se conectează, acestuia i se atribuie un fir de execuție prin care vor fi procesate cererile acestuia.
* Clasa ThreadPooledServer
* Pachetul WorkerClientThread conține clasa cu același nume.
* WorkerClientThread: această clasă se ocupă exclusiv de cititrea mesajelor din partea clienților. În funcție de tipul mesajului, acesta va fi trimis unei clase specifice. În urma procesării, va rezulta un obiect „mesaj” care va fi trimis înapoi clientului;
* Pachetul User:
  + Clasa User conține datele despre un client;
* Pachetul TransformerBytes:
  + Clasa TransformerBytes va fi folosită pentru a serializa obiectul „mesaj” ;
* Pachetul SessionManager:
  + Clasa SessionManager se ocupă de sesiunea utilizatorului, dar și de înregistrarea, logare și delogarea acestuia. Aceasta mai conține două clase cu următoarele funcționalități:
    - SessionIdentifier, folosită pentru a returna un identificator unic pentru sesiunea utilizatorului;
    - Session, care stocheaza datele sesiunii.
* Pachetul Reusable conține patru enumerații pentru facilita lizibilitatea codului dar și pentru optimizarea acestuia ( modificarea unei valori dintr-o enumerație va duce la actualizarea automata a acesteia in codul serverului):
  + - ServerInfo
    - RespondeEnum
    - DatabaseInfo
    - RequestEnum
* Pachetul Nota:
  + Clasa Nota conține informațiile despre o notă.
* Pachetul Mesaj
  + Clasa Mesaj este folosită pentru a stoca mesajele dintre client si server. Acest proces se va realiza prin serializare.
* Pachetul ManagerGrup
  + Clasa ManagerGrup se ocupa de organizarea grupurilor.
* Pachetul Grup
  + Clasa Grup stochează informațiile unui grup.
* Pachetul FileManager
  + Clasa FileManager se ocupă de prelucrarea cărtilor și anume salvarea și clasificarea lor pe server;
* Pachetul Database conține doua clase:
  + Clasa ManagerDb se ocupă de pornirea bazei de date dar si de toate cererile celorlalte clase;
  + Clasa Database se ocupă strict de conexiunea cu SQLite.
* Pachetul UserInteraction:
  + Clasa UserInteraction se ocupă de notificările utilizatorului, dar și de recomandările dintre utilizator;
* Pachetul ManagerNote:
  + Clasa ManagerNote se ocupă de interacțiunea utilizatorului cu notele;
* Pachet Carte
  + Clasa Carte se ocupa de stocare informațiilor despre o carte.

# Scenarii de utilizare

În acest subcapitol vor fi prezentate toate posibilele șcenarii de utilizare ale aplicației cu imaginile aferente, pentru o mai buna înțelegere .

1. **Autentificarea**

Pentru a se realiza conexiunea la server, înainte de a intra în aplicație, utilizatorul va trebui sa stabilească o conexiune la internet. În momentul în care se va intra în aplicație, acesta va fi întampinat de un ecran de autentificare. În partea de jos a ecranului va fi un link cu trimitere la pagina de înregistrare, în cazul în care nu aveți un cont deja făcut.

Pentru a putea intra în cont, utilizatorul va avea nevoie de numele de utilizator si de parola cu, care și-a creat contul. Atât în cazul în care autentificarea s-a realizat cu success, cât și în caz contrar, utilizatorul va primi un mesaj specific.



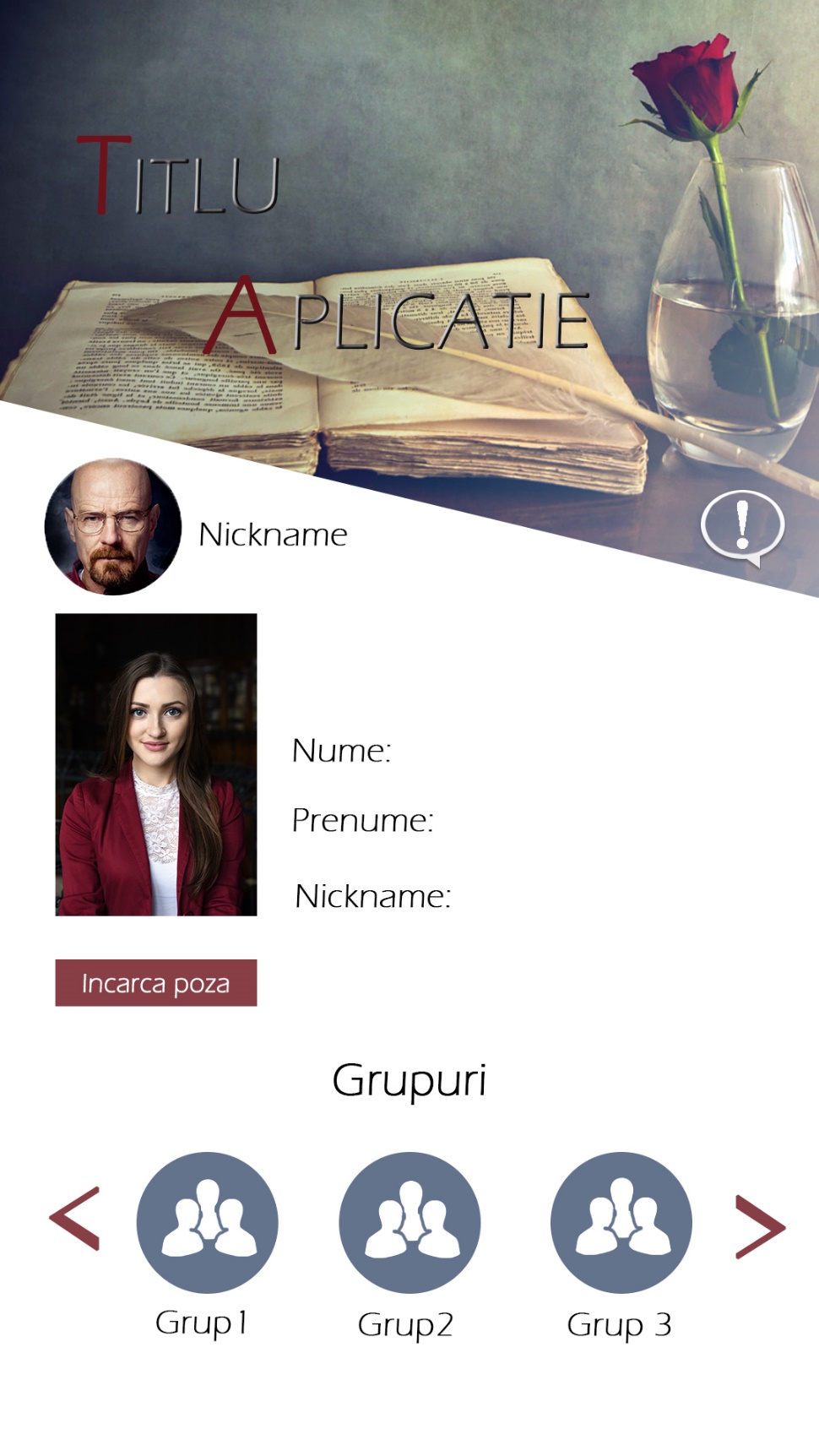
1. **Inregistrarea**

În cazul în care abia luați contact cu această aplicație, veți avea la dispoziție ecranul pentru înregistrare. Va trebui să se completeze câmpurile Nume, Prenume, NumeUtilizator si Parola. În cazul in care unul, sau mai multe câmpuri nu sunt completate, utilizatorul nu va putea finaliza acțiunea de înregistrare, fiind atenționat asupra acestui aspect.



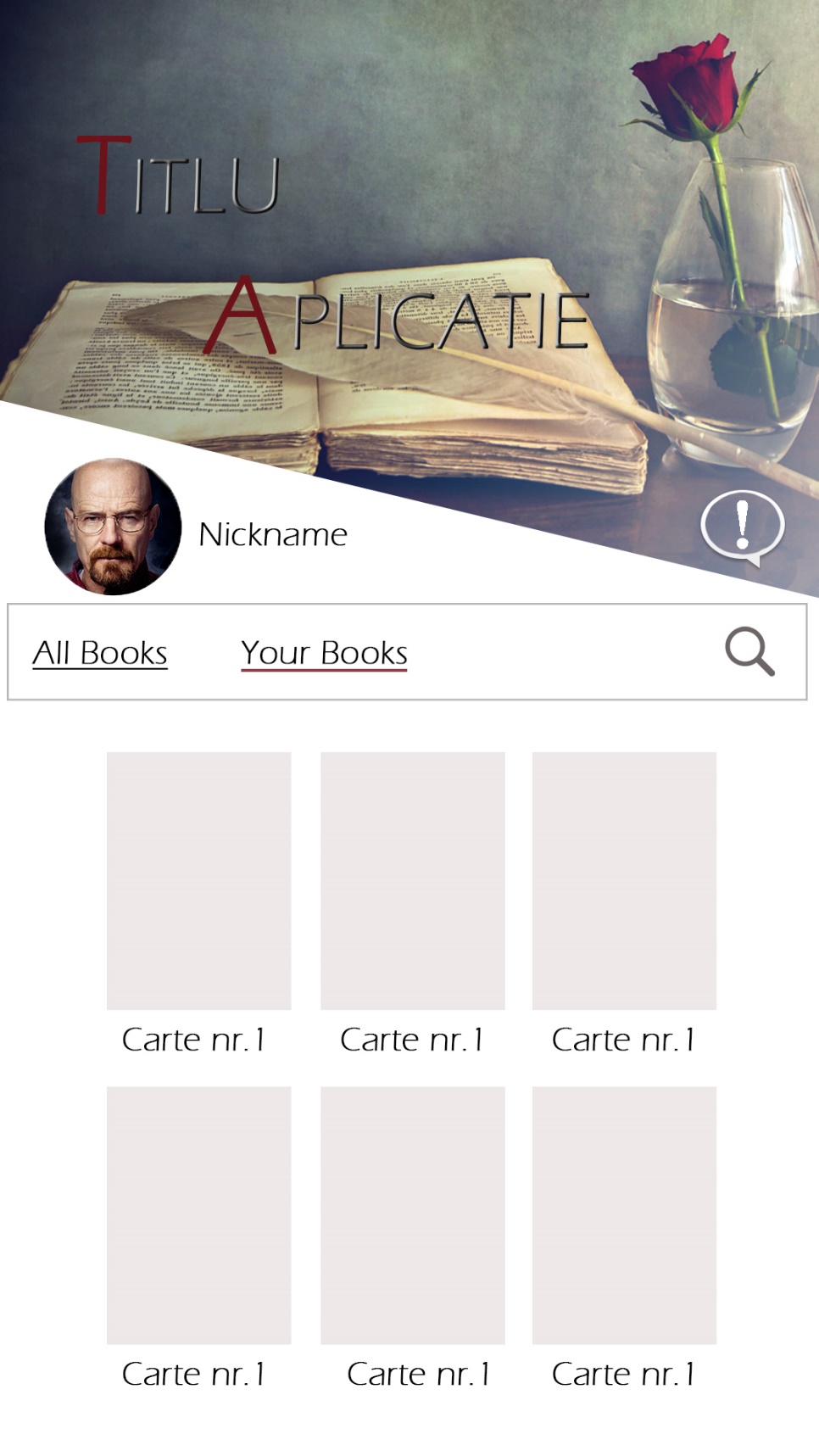
1. **Modificare profil**

În momentul în care utilizatorul a reușit să treacă peste etapele anterioare, va avea posibilitatea de a-și complete profilul, alegând să iși seteze o poză de profil. Tot în cadrul acestui ecran, utilizatorul va putea vedea grupurile din care face parte, precum și semnul care il atenționează atunci cand primește o notificare.



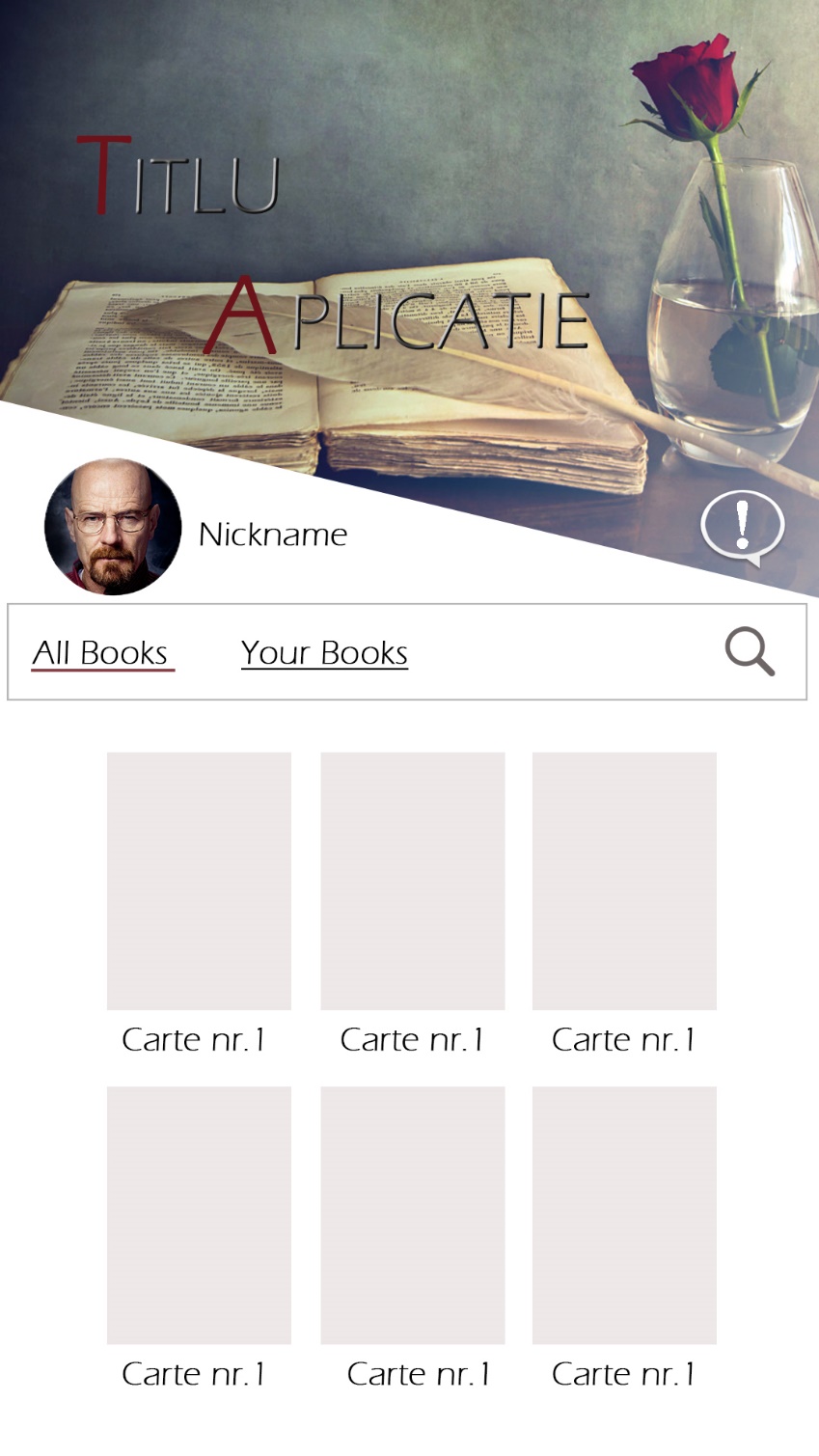
1. **Afișarea carților mele**

În această fereastra, utilizatorul va putea vedea carțile pe care le-a încărcat. De asemenea va avea și posibilitatea de a căuta o carte prin intermediul opțiunii “Căutare”.



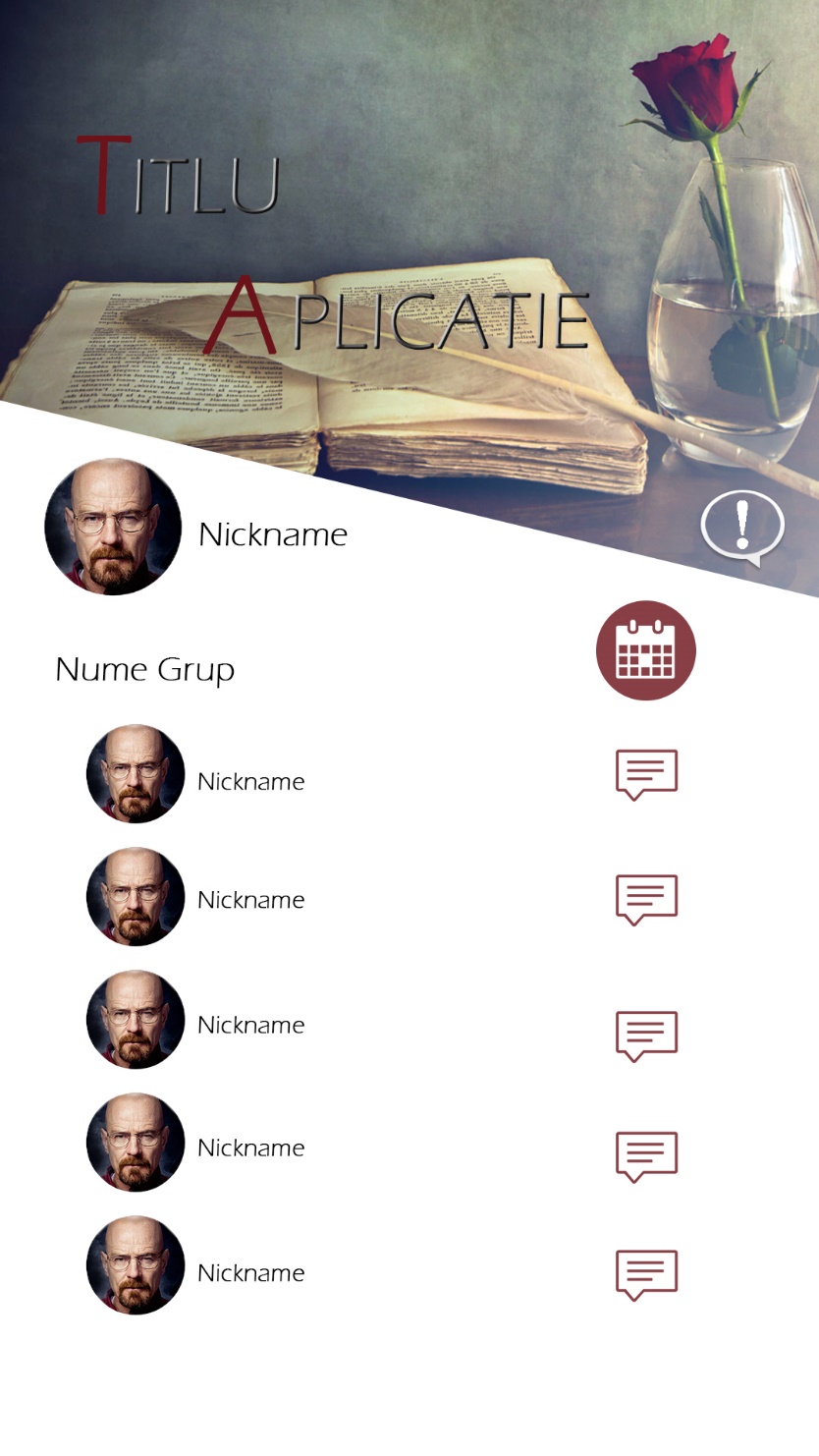
1. **Afișarea tuturor carților**

În această fereastra, utilizatorul va putea vizualiza toate cărțile disponibile atât din intermediul aplicației, cât și cele încarcate de utilizator. De asemenea va avea și posibilitatea de a căuta o carte prin intermediul opțiunii “Căutare”.



1. **Vizualizarea unui grup**

În figura “…” utilizatorul va putea vedea toți membrii grupului. De asemenea prin apăsarea iconiței din dreapta unui membru al grupului, vei putea face o recomandare de carte specifică grupului.



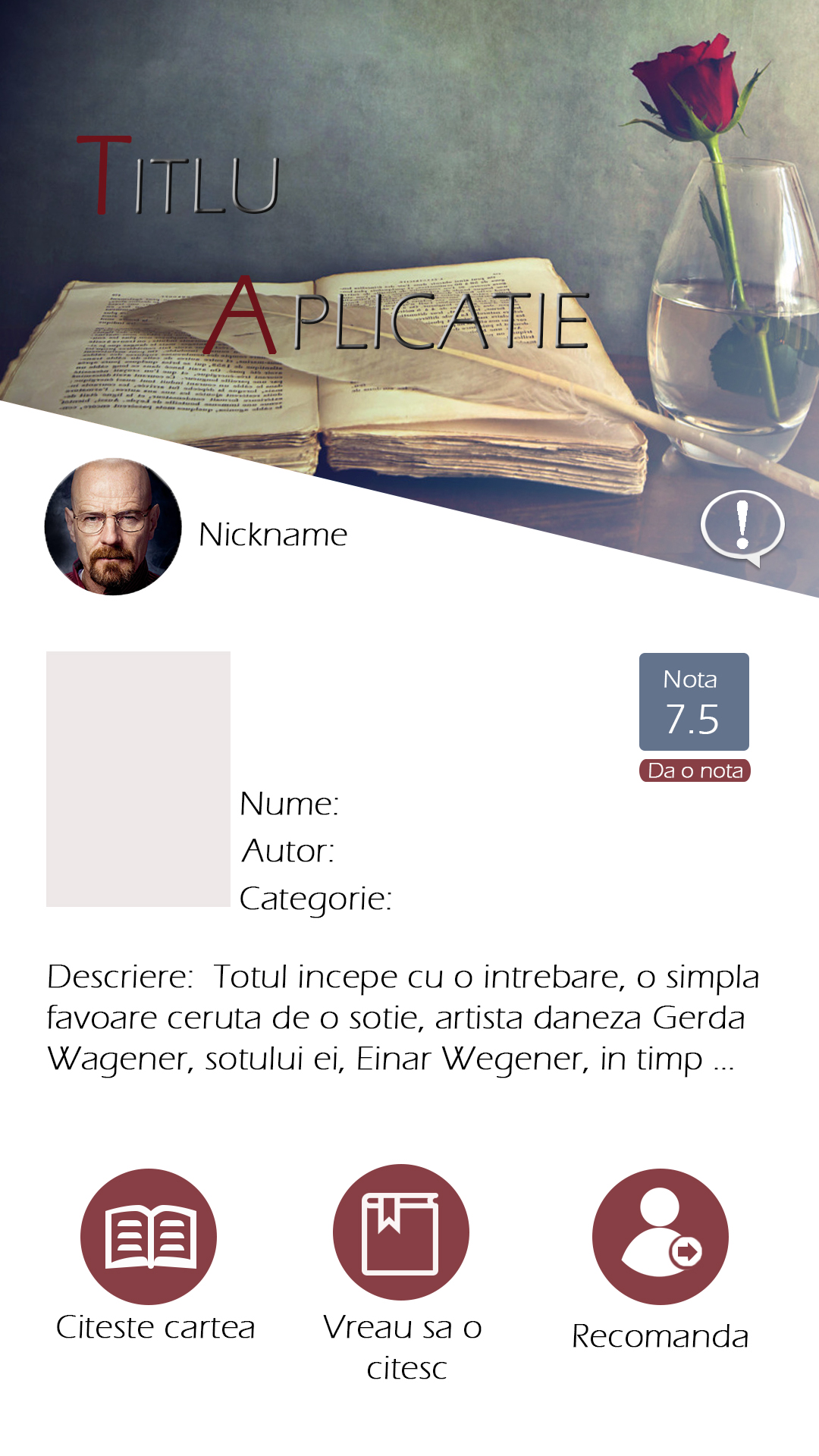
1. **Vizualizare carte**

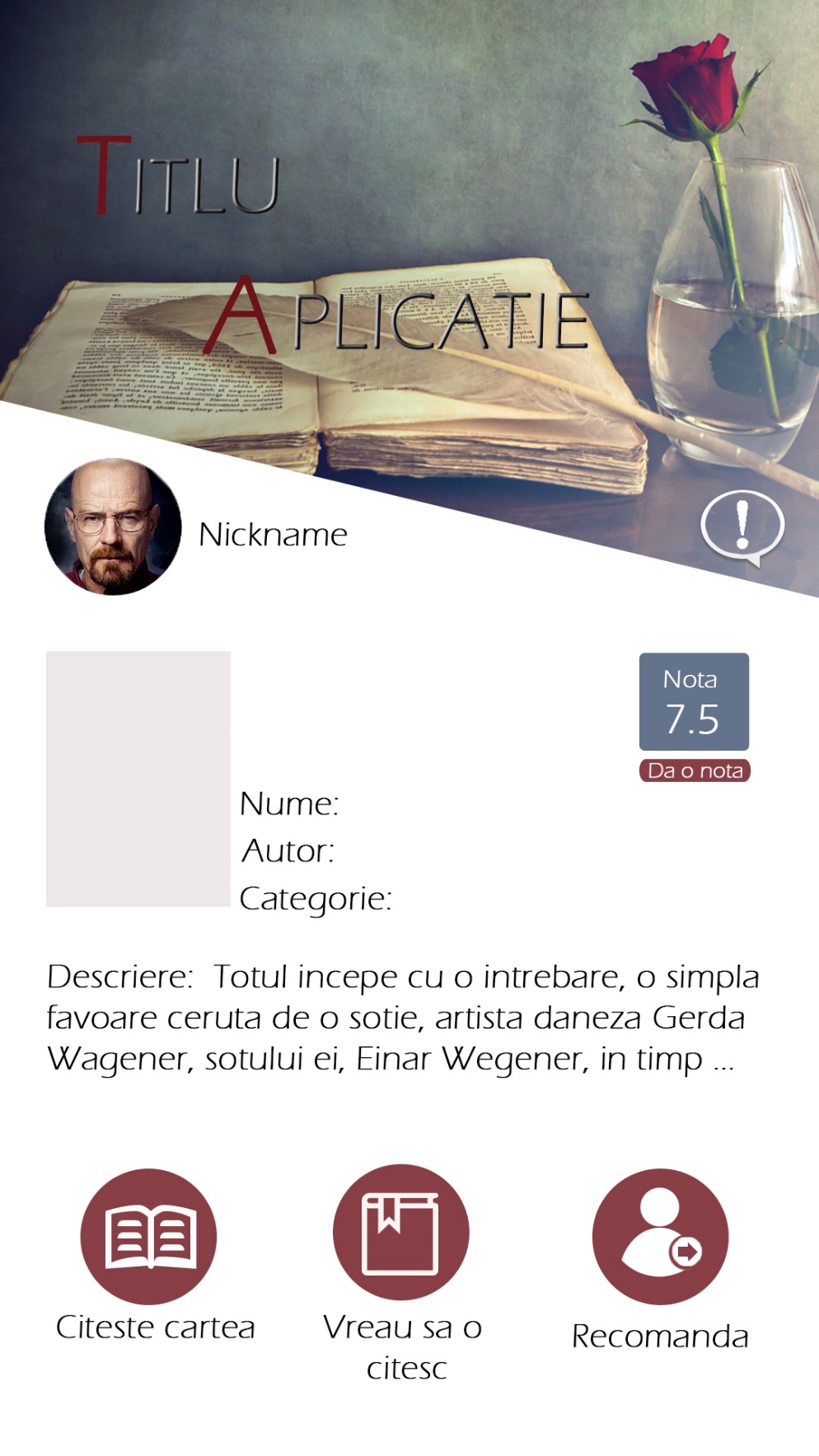
În “Vizualizare Carte” utilizatorul va putea vizualiza informațiile despre o carte:

* Nume
* Prenume
* Categorie
* Nota
* Descriere

În partea de jos a ecranului, utilizatorul va dispune de trei butoane cu urmatoarele functionalități:

* “Citește cartea” - odată aleasă această funcționalitate, se va deschide modul vizualizare;
* “ Vreau sa o citesc” – această funcționalitate îți permite să îți creezi o lista cu acele cărți pe care dorști să le citești pe viitor;
* “Recomandă” – această opțiune iți oferă posibilitatea de a recomanda cartea respectivă unui utilizator;





1. http://www.eseuargumentativromana.com/2016/07/rolul-lecturii-in-dezvoltarea-personala.html [↑](#endnote-ref-1)