MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**
**DEPARTAMENTUL CALCULATOARE**

# Sistem pentru gestionarea acțiunilor caritabile în ONG-uri

LUCRARE DE LICENŢĂ

Absolvent: **Raluca IOSIF**

Coordonator ştiinţific: **Prof. Asist. Ing. Cosmina IVAN**

**2019**

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**
**DEPARTAMENTUL CALCULATOARE**

DECAN,
**Prof. dr. ing. Liviu MICLEA**

DIRECTOR DEPARTAMENT,
**Prof. dr. ing. Rodica POTOLEA**

Absolvent: **Raluca IOSIF**

**Sistem pentru gestionarea acțiunilor caritabile în ONG-uri**

1. **Enunţul temei:** *Proiectul își propune realizarea unui sistem care să faciliteze acțiunile caritabile ale organizațiilor ONG, prin digitalizarea datelor și gestionarea acestora, oferind metode pentru organizare internă eficientă.*

2. **Conţinutul lucrării:** *Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii, Bibliografie, Anexa 1 – Lista Figurilor, Anexa2 – Lista Tabelelor, Anexa 3 – Glosar.*

3. **Locul documentării**: *Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare*

4. **Consultanţi**:

5. **Data emiterii temei:** 1 noiembrie 2018

6. **Data predării:** 8 iulie 2019

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**
**DEPARTAMENTUL CALCULATOARE**

# Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

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

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

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

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

Data

_____

Nume, Prenume

_____

Semnătura

# Cuprins

# Capitolul 1. Introducere

"I don't want to live in the kind of world where we don't look out for each other. Not just the people that are close to us, but anybody who needs a helping hand. I can't change the way anybody else thinks, or what they choose to do, but I can do my bit."

- Charles de Lint

Așa cum Charles de Lint spune în citatul de mai sus, poate nu putem schimba lumea, societatea, felul oamenilor de a gândi sau acțiunile lor, dar cu siguranță fiecare dintre noi ne putem îndeplini task-urile în cel mai bun mod posibil, contribuind astfel la o societate mai bună. Din această dorință s-a ales să se dezvolte acest sistem descris în lucrarea prezentă, pentru a-i ajuta prin intermediul tehnologiei pe cei care la rândul lor ajută oamenii în nevoie.

Organizațiile non-guvernamentale reprezintă instituții care aspiră la o societate mai bună, contribuind la dezvoltarea acesteia prin muncă voluntară, fără a urmări obținerea unui profit în folosul personal. Importanța lor în bunul mers și îmbunătățirea stării societatății și a comunității este semnificativă.

Acestea au devenit stakeholderi cheie într-un sistem care de multe ori nu poate face față numărului mare de atribuții. În ultimii ani, în timp ce multe organizații au crescut și volumul lor de lucru s-a multiplicat, capacitatea lor nu a putut tot timpul ține pasul cu tendințele digitalizării. De asemena, organizațiile non-guvernamentale se confruntă cu păstrarea încrederii față de binefăcătorii care donează resurse, fiind astfel nevoie și de un grad de transparență în ceea ce privește acțiunile desfășurate pe baza donațiilor primite.

Activitățile conduse de un simț al răspunderii și al empatiei față de societate reprezintă nucleul acestor organizații. S-a identificat că îndeplinirea atribuțiilor poate fi imbunătățită cu ajutorul tehnologiei informației. Organizarea de tip non-profit implică faptul că orice sumă obținută este investită în susținerea activităților și scopului urmărit.

Soluțiile ce țin de tehnologia informației pentru organizații au evoluat de-a lungul timpului. Ele sunt de obicei conectate între ele printr-o interfață manuală (cum ar fi Excel). Datele sunt generate extern și colectate într-o manieră decentralizată în sistemele variate ale fiecărei organizații. Acest lucru de multe ori are ca și efect existența unor date duplicate. La sistemele integrate, datele sunt colectate o singură dată și pot fi folosite în diferite scopuri în cadrul sistemului, ceea ce contribuie și la scăderea volumului de muncă. Acestea vin în plus cu un nou nivel de performanță și transparență, ceea ce este benefic atât pentru organizație cât și pentru donatori.

Prin proiectul propus s-a dezvoltat o aplicație mobilă care reprezintă un sistem de management al informațiilor. Astfel, se vine în ajutorul organizațiilor non-guvernamentale prin oferirea unui mod de stocare și gestionare compact, transparent și eficient a datelor cu care se lucrează în fiecare zi.

## 1.1. Context general

Organizațiile non-guvernamentale (denumite în continuare ONG-uri) sunt instutiții care lucrează independent de activitatea guvernului și întreprind activități sociale. Fiind denumite și organizații non-profit, se înțelege că nu au ca și scop principal obținerea de profit, ceea ce nu presupune neaparat lipsa totală a profitului, ci faptul că acesta este folosit

doar pentru atingerea scopului social propus. Finanțarea acestor organizații de obicei vine prin donații. Importanța ONG-urilor o reprezintă însuși scopul lor de a contribui în societate acolo unde este o nevoie.

Conform lucrării [1], ONG-urile sunt împărțite în mai multe tipuri, in funcție de orientarea activităților întreprinse și în funție de nivelul de operare.

Orientarea se referă la tipul de activități desfășurate, delimitând astfel următoarele tipuri de ONG-uri:

- **ONG-uri caritabile**: își desfășoară activitățile cu ajutorul voluntarilor, fiind susținute de beneficiari și incluzând organizații al căror obiectiv este ajutorarea persoanelor aflate în nevoie.
- **ONG-uri orientate pe dezvoltare**: au ca și scop final îmbunătățirea capacitatății unei comunități de a-și asigura propriile nevoi de bază. În această categorie sunt identificate două sub-categorii. Primele sunt ONG-urile orietate pe apartenență, în care comunitatea este formată chiar din membrii organizației și aceștia sunt beneficiarii. Cea de-a doua categorie este reprezentată de ONG-urile orientate pe servicii care au ca și scop oferirea diferitelor servicii altor organizații sau a unei populații.
- **ONG-uri orientate pe susținere**: vor să aibă o influență asupra unor probleme particulare legate de politică sau de luarea unor decizii, oferind totodată suport social organizațiilor care împărtășesc același scop, cât și societății în general.
- **ONG-uri orientate pe dezvoltarea educației**: se află în principiu în țările industrializate și au în vedere educarea cetățenilor din țările sub-dezvoltate.

În funcție de nivelul de operare s-a identificat următoarea clasificare:

- **ONG-uri bazate pe comunitate**: sunt responsabile pentru creșterea conștientizării situației oamenilor săraci din zonele urbane, ajutarea acestora prin educarea în ceea ce privește drepturile lor cât și oferirea acestor servicii.
- **ONG-uri regionale:** servesc unor regiuni întregi care fac parte din lumea a treia.
- **ONG-uri naționale**: acționează la nivel național, după cum și numele sugerează, și pot avea sub-organizații la nivel de oraș care asistă ONG-urile locale.
- **ONG-uri internaționale**: își desfășoară activitățile la nivel internațional și pot fi responsabile pentru fondarea unor ONG-uri locale, instituții și proiecte.

O clasificare completă a ONG-urilor a fost elaborată în cadrul Johns Hopkins University din SUA în lucrarea [2], scrisă de către Lester M. Salamon și Helmut K. Anheier. În vederea alcătuirii clasificării complete s-au identificat cinci caracteristici de bază, comune tuturor entităților care alcătuiesc sectorul non-profit. Acestea sunt:

1. Organizarea sau insituționalizarea într-o oarecare măsură: subliniază importanța prezenței unui factor instituțional în organizarea unor astfel de entități, ceea ce include o anumită structură organizațională, persistența scopurilor urmărite și existența unor limite organizaționale, cum ar fi diferențierea între membrii și non-membrii.

2. Separarea instituțională de guvern: semnifică deținerea unei identități instituționale separată de cea a statului, care nu exercită autoriate guvernamentală.
3. Autoguvernare, sau abilitatea de a-și controla propriile activități: implică deținerea unor proceduri guvernamentale interne și autonomie.
4. Distribuție non-profit: implică redistribuirea oricărui profit obținut pe parcursul unui an în susținerea misiunii asociației. Niciun profit nu ajunge la membrii asociatiei, indiferent de statutul acestora.
5. Voluntariat: sugerează existența unui anumit grad de participare sau implicare voluntară, excluzând astfel din sectorul non-profit organizațiile al căror statut de membru este impus sau stimulat într-un anume fel de lege.

Analizând clasificarea bazată pe lucrarea [1] și caracteristicile identificate în lucrarea [2], organizațiile non-guvernamentale pentru care s-a dezvoltat sistemul de management al informațiilor se pot încadra în categoria organizațiilor bazate pe comunitate, al căror obiectiv este de a avea un impact în societate prin ajutarea persoanelor în nevoie, distribuind pachete.

## 1.2. Contextul proiectului

Lucrarea prezentă are în vedere digitalizarea acestor ONG-uri și în acest scop s-au identificat câteva motive pentru care acest pas spre era digitală trebuie făcut:

- Digitalizarea datelor unei organizații susține luarea deciziilor bazate pe prelucrarea datelor
- Salvarea timpului prețios în situații de urgență
- Vizibilitate: susținerea luptei împotriva injustiției și promovarea egalității
- Productivitate
- Recuperarea datelor

Considerând faptul că ONG-urile lucrează spre a aduce o schimbare în societate și în viețile multor persoane care se află în nevoie, este vital ca acestea să își poată desfășura activitatea cât mai lin și fără impedimente. Astfel, o gestionare eficientă și digitalizată a datelor vine în ajutorul lor, eliminând stresul care vine odată cu multitudinea de informații păstrate în format fizic, pe foi. O bază de date diminuează efortul gestionării informațiilor, oferind de altfel și posibilitatea de recuperare a datelor. Mai mult decât atat, digitalizarea oferă organizațiilor o vedere de ansamblu necesară pentru luarea de decizii data-driven.

Una dintre problemele întâlnite în activitatea acestor organizații este identificarea tentativelor de fraudă. Deoarece scopul principal este de a ajuta persoanele care au nevoie și de a face o schimbare în societatea în care trăim, se dorește ca fiecare om să aibă șanse egale pentru obținerea unui ajutor social. Cu toate acestea, sunt persoane care vor să profite de această mișcare socială fie prin înregistrarea la mai multe asociații, fie prin reprezentarea necinstită a stării sale fizice sau financiare. Având informațiile fiecărei persoane stocate digital permite membrilor ONG-urilor să gestioneze mai bine aceste situații și să identifice un astfel de comportament.

În urma discuțiilor cu persoanele responsabile pentru astfel de acțiuni non-profit și pe baza propriei experiențe de voluntariat am observat prezența unei nevoi de stocare și manipulare a datelor în format digital în primul rând, pentru a se înlocui astfel documentele fizice. În plus, se dorește un nivel ridicat de performanță și organizare. Astfel s-a dezvoltat o aplicație mobilă care să fie mereu la îndemâna membrilor și să ofere exact aceste lucruri ce lipsesc organizațiilor non-profit, care încă nu au pășit în era digitală.

## 1.3. Conținutul lucrării

**Capitolul 1 – Introducere** – acest capitol oferă o prezentare generală a organizațiilor non-guvernamentale și contextul proiectului dezvoltat.

**Capitolul 2 – Obiectivele proiectului –** prezintă o descriere a obiectivelor atât particulare cât și generale care se vor a fi atinse în dezvoltarea acestuia.

**Capitolul 3 – Studiu bibliografic –** oferă o prezentare generală și un scurt istoric al sistemelor de management al informațiilor, urmată de o clasificare a acestora și ilustrarea avantajelor utilizării lor. Mai apoi, sunt introduse câteva sisteme similare și o analiză comparativă cu sistemul dezvoltat. La final, se prezintă funcționalitățile stabilite pentru proiectul prezent.

**Capitolul 4 – Analiză și fundamentare teoretică** – introduce cerințele funcționale și non-funcționale ale sistemului, descrie arhitectura conceptuală, câteva cazuri de utilizare și tehnologiile folosite în dezvoltare.

**Capitolul 5 – Proiectare de Detaliu și Implementare –** oferă o prezentare generală a aplicației, detaliază structura componentelor acesteia, reprezentată prin diagrame de pachet atât pentru aplicația client cât și pentru server. De asemenea, sunt detaliați și câtiva algoritmi semnificativi. La final, este ilustrată diagrama de deployment.

**Capitolul 6 -Testare și Validare –** oferă o viziune de ansamblu asupra testării și prezintă metodele abordate în testarea funcționalității componentelor aplicației, precum și rezultatele obținute.

**Capitolul 7 – Manual de Instalare și Utilizare –** menționează resursele software și hardware necesare instalării și uilizării aplicației, precum și instrucțiunile de folosire.

**Capitolul 8** – **Concluzii –** în cadrul acestui capitol se vor menționa contribuțiile personale la aplicație, se vor prezenta rezultatele obținute și viitoare posibilități de îmbunătățire și dezvoltare ulterioară.

# Capitolul 2. Obiectivele Proiectului

În cadrul acestui capitol se prezintă obiectivul principal al proiectului propus, precum și obiectivele generale urmărite în scopul atingerii obiectivului principal. Acestea au fost stabilite de la conceperea proiectului, pentru a defini scopul acestuia, care mai apoi au fost definite și structurate ca și cerințe funcționale pentru sistem.

## 2.1. Obiectivul principal

Obiectivul proiectului prezent este digitalizarea unor organizații non-profit care se ocupă cu distribuirea de ajutoare persoanelor dezavantajate, facilitând acțiunile civice desfășurate de acestea. Acesta este atins prin dezvoltarea unui sistem de management al informațiilor, pentru facilitarea desfășurării activităților membrilor organizației și gestionarea informațiilor cu care aceștia lucrează zi de zi.

Multe astfel de organizații din ziua de astăzi activează în afara mediului digital, toate informațiile despre produse, persoane și altele se află pe hârtii, principalul motiv fiind prioritizarea fondurilor. Astfel, în activitatea de zi cu zi a persoanelor implicate apar dificultăți în gestionarea tuturor informațiilor. Sistemul propus vine in ajutorul acestora prin digitalizarea datelor, putând astfel fi administrate mai simplu și mai eficient.

## 2.2. Obiective generale

Pentru îndeplinirea obiectivului principal trebuie îndeplinită o secvență de obiective secundare. Acestea sunt:

- Gestiunea eficientă a produselor de care dispune o organizație și a cazurilor sociale
- Alocarea eficientă a produselor de care dispune organizația la cazurile sociale (în funcție de proximitate sau de gradul de prioritate a nevoilor)
- Programarea transferurilor care au loc între o asociație și beneficiari sau între două asociații

Urmărind obiectivele secundare și îndeplinirea acestora, trebuie rezolvate următoarele obiective mai specifice:

- Înregistrarea noilor membrii: orice membru al organizației are posibilitatea de a se înregistra prin crearea unui cont de utilizator prin intermediul căruia să poată accesa și utiliza sistemul.
- Autentificare: utilizatorii înregistrați au posibilitatea de a se autentifica și de a accesa datele disponibile, în funcție de permisiunea oferită de către administrator pentru contul de utilizator asociat.
- Vizualizarea datelor specifice cazurilor sociale, cu persoanele și nevoile asociate, dezactivarea și adăugarea de cazuri sociale, sortate în funcție de prioritate.
- Vizualizarea stocurilor disponibile și actualizarea acestora.

- Programarea ședințelor interne și a transferurilor de pachete de la asociație către cazurile sociale.
- Asocierea stocurilor disponibile cu nevoile prezente în cazurile sociale active apartinând organizației.
- Vizualizarea rapoartelor generate pe baza activităților desfășurate, pentru evaluarea distribuirii stocurilor și statusul acestora.

Prin îndeplinirea obiectivelor specifice și cu ele totodată și obiectivele secundare, se obține digitalizarea dorită și enunțată în obiectivul principal, care reprezintă esența proiectului dezvoltat.

# Capitolul 3. Studiu Bibliografic

În acest capitol se oferă o prezentare generală asupra Sistemelor de Management a Informațiilor, tipurile acestora și avantajele lor, susținând alegerea sistemului implementat în lucrarea prezentă. Un aspect importat este prezentarea sistemelor similare celui dezvoltat, urmată de o analiză comparativă din punctul de vedere a funcționalităților identificate anterior și motivarea acelora alese și implementate în sistemul propus.

## 3.1. Sistemele de management ale informațiilor

Acest sub-capitol cuprinde a scurtă prezentare generală și un istoric al sistemelor de management al informațiilor, clasificarea acestora conform manualului [3], iar în final sunt prezentate avantajele utilizării unor astfel de sisteme.

### *3.1.2. Prezentare generală și scurt istoric*

Pentru susținerea funcționării ONG-urilor și digitalizarea acestora s-a dezvoltat o platformă pentru servirea și găzduirea acțiunilor sociale, având în vedere tipurile de acțiuni sociale și datele procesate. Acest lucru înseamnă, bine-nțeles, că platforma va fi una particulară organizațiilor alese în această lucrare, și anume cele care acționează în zona socială a oamenilor nevoiași prin a le oferi pachete cu bunuri materiale necesare.

Un sistem de management a informațiilor este un sistem de informații folosit într-o organizație pentru luarea de decizii, coordonare, gestionarea resurselor și analiza informațiilor specifice activităților întreprinse. Acesta poate fi un sistem computerizat alcătuit din componente hardware și software care au rol de motor principal pentru gestionarea informațiilor.

Istoria sistemelor de management a informației este divizată în cinci ere, prezentate de către Kenneth C. Laudon și Jane Laudon în manualul [3].

Prima eră este reprezentată de calculatoarele mainframe, care ofereau atât hardware cât și software, furnizate de IBM. Acestea ocupau săli întregi și necesitau o întreagă echipă pentru gestionarea lor. Odată cu avansarea tehnologiei însă, calculatoarele lor au ajuns la o versiune mai compactă, cu o capacitate mai mare și cost redus, potrivite pentru întreprinderi mai mari.

A doua eră, cea a calculatorelor personale, a început când microprocesoarele au devenit o concurență pentru calculatoarele mainframe și cele mini, accelerand procesul de decentralizare a puterii de calcul de la centre de date mari la birouri.

A treia eră, a rețelelor client-server, introduce modalitatea de partajare a informațiilor de pe un server către mai multe calculatoare aflate în aceeași rețea. Acest lucru permite unei mulțimi de oameni sa acceseze simultan date.

A patra eră, a tehnicii de calcul cloud, introduce un nou nivel de mobilitate alături de dispozitive mobile de mare viteză și retele Wi-Fi, astfel încât sistemul de management al informațiilor poate fi accesat de la distanță, de pe un laptop, de pe un telefon inteligent sau de pe o tabletă.

## 3.2. Clasificarea sistemelor de management a informațiilor

Tipurile de sisteme de management al informațiilor care au fost identificate sunt următoarele:

- Sisteme de Suport Decizional[1]: sunt aplicații de program folosite pentru compilarea informațiilor care provin din mai multe surse, pentru sprijin în rezolvarea de probleme și luarea de decizii.
- Sisteme de Informații Executive (cunoscute ca și sisteme de suport executive)[2]: sunt sisteme de suport pentru administrarea informațiilor care oferă acces rapid la rapoarte, provenind din departamente diferite ale unei organizații (de exemplu: Resurse Umane, Contabilitate ș.a.).
- Sisteme Informatice de Marketing[3]: sunt folosite pentru gestionarea aspectelor ce țin de marketing într-o organizație.
- Sisteme Informatice de Contabilitate[4]: se axează pe funcționalități ce țin de contabilitate.
- Sisteme de Gestionare a Resurselor Umane[5]: sunt folosite în aspecte legate de personalul unei organizații.
- Sisteme de Automatizare a Birourilor[6]: automatizează fluxul de muncă, susținând comunicarea și productivitatea.
- Sisteme de Management al Informațiilor în Școli[7]: sunt folosite pentru administrarea la nivel de școli.
- Sisteme de Planificare a Resurselor Întreprinderii[8]: folosite de organizații pentru colectarea, stocarea, gestionarea și interpretarea datelor ce provin din activitățiile desfășurate. Acestea urmăresc resursele organizației (bani, materie primă, capacitatea de producție).
- Baze de Date Locale[9]: sunt considerate ca reprezentând versiunea de bază a sistemelor de management a informațiilor

Sistemul dezvoltat este identificat ca un hibrid între un sistem de planificare a resurselor și un sistem de gestionarea resurselor umane, care are în compoziția sa o bază de date locală. Acesta oferă suport pentru gestionarea stocurilor și a persoanelor care reprezintă cazuri sociale, precum și pentru luarea deciziilor în ceea ce presupune distribuirea stocurilor.

Astfel, platforma non-guvernamentală dezvoltată a fost adaptată ca și o versiune particulară de sistem de management client-server pentru accesarea informațiilor de la distanță, prin intermediul unui dispozitiv mobil inteligent care să ruleze pe un sistem de operare Android.

[1] https://en.wikipedia.org/wiki/Decision_support_system
[2] https://en.wikipedia.org/wiki/Executive_information_system
[3] https://en.wikipedia.org/wiki/Marketing_information_system
[4] https://en.wikipedia.org/wiki/Accounting_information_system
[5] https://en.wikipedia.org/wiki/Human_resource_management_system
[6] https://en.wikipedia.org/wiki/Office_automation
[7] https://en.wikipedia.org/wiki/School_Information_Management_System
[8] https://en.wikipedia.org/wiki/Enterprise_resource_planning
[9] https://en.wikipedia.org/wiki/Database

### *3.2.1. Avantajele utilizării unui sistem de management a informațiilor*

Avantajele folosirii unui astfel de sistem de management a informațiilor de către o organizație sunt:

- Îmbunătățirea eficienței operaționale
- Posibilitatea identificării punctelor forte și a celor slabe, prin analizarea rapoartelor
- Posibilitatea de a se comporta ca și un instrument de planificare și comunicare
- Obținerea unui avantaj competitiv

În concluzie, în cadrul acestui capitol au fost prezentate sistemele de management al informațiilor, cuprinzând un scurt istoric și caracterizarea acestora, apoi s-a încadrat sistemul dezvoltat ca fiind un hibrid între un sistem de planificare a resurselor și un sistem de gestionarea resurselor umane. În încheiere s-au identificat avantajele utilizării sistemului propus.

## 3.3. Sisteme similare

Până în prezent au fost implementate diferite soluții pentru digitalizarea serviciilor publice cu ajutorul voluntarilor. Numeroase proiecte s-au dezvoltat sau sunt în curs de dezvoltare. Una dintre aceste inițiative aparține de civitech$^{RO}$, care este în sine o organizație non-guvernamentală a cărei activitate este aceea de a crea soluții publice digitale sau de a moderniza cele existente. Printre proiectele lor se numără: VoluntApp (aplicație de management a membrilor civitech$^{RO}$), fiipregatit.ro (platformă națională pentru Situații de Urgență), OpenEnergy (aplicație pentru încurajarea consumului responsabil al energiei electrice) și altele. O platformă care desfășoară activități civice similare este code4[10] care dezvoltă soluții IT în vederea rezolvării problemelor societății.

În continuare vor fi prezentate sisteme similare pentru managementul organizațiilor non-guvernamentale, pe baza funcționalităților și caracteristicilor identificate mai sus.

**VoluntApp[11]:** este o aplicație open source internă pentru managementnul voluntarilor, proiectelor și resursele organizației civitech$^{RO}$. Aceasta se află în curs de dezvolare, fiind completă în procent de 75% (conform datelor de pe platforma civitech). Aplicația poate fi folosită de către două tipuri de utilizatori: administrator și volutar. Pentru administrator, principalele funcționalități sunt: înregistrarea voluntarilor și completarea de profil, căutarea de voluntari în funcție de anumite criterii, înregistrarea proiectelor, alocarea voluntarilor pe proiecte, înregistrarea progresului pe proiecte și mass mailing (trimiterea în masă de mail-uri). Pentru voluntar, principalele funcționalități sunt: înregistrare, editare profil personal, acces la lista de proiecte și informațiile publice a acestora, acces la lista cu poziții disponibile și aplicare pentru poziții deschise.

Cu toate acestea, proiectul propus în această lucrare diferă de cele implementate de platformele menționate, prin tipul de date pentru care se dorește să se facă managementul

[10] code4.ro
[11] https://civictech.ro/proiecte/voluntapp-aplicatia-de-management-a-membrilor-civictech

și prin activitățiile desfășurate (și anume distribuirea de pachete de ajutor la persoanele nevoiașe).

**NGO Online[12]:** este un sistem de management special dezvoltat pentru ONG-uri. Este accesibil de oriunde, fiind bazat pe design Cloud și totodată personalizabil pentru nevoile și modalitățile de funcționare a fiecărui ONG. Este construit pe tehnologiile Microsoft Office 365, SQL Server și Azure. Printre funcționalităție de bază de numără următoarele: gestionarea donatorilor, a donațiilor și a bugetului în același loc, gestionarea documentelor, monitorizare și generare de rapoarte, integrare cu sisteme existente și altele.

**CIVICRM[13]:** este un sistem CRM (Customer-Relationship Management) open source dezvoltat de o comunitate de contribuitori și susținători. Este un software web destinat unei game vaste de organizații, în special celor non-profit. Sistemul oferă un set de funcționalități și se poate integra cu website-ul unei organizații. Printre funcționalități se numără următoarele: gestionarea membrilor (autentificare, plata pentru membership, reînoire automată și primire de notificări), gestionarea evenimentelor, marketing prin email, procesarea și urmărirea contribuitorilor, campanii, strângeri de fonduri, generare de rapoarte, gestionarea resurselor (cecuri, carduri de credit, tranzacții) ș.a.

### *3.3.1. Analiză comparativă a sistemelor studiate*

În tabelul 3.1 se prezintă o analiză comparativă între sistemele menționate mai sus și cel propus, pe baza funcționalităților identificate mai sus.

Tabel 3. 1 Analiză comparativă între sistemele similare

| Funcționalitate | ONG Management | VoluntApp | NGO Online | CIVICRM |
|---|---|---|---|---|
| Gestionarea informațiilor | D | D | D | D |
| Gestionarea resurselor umane | D | D | D | D |
| Confidențialitatea informațiilor personale | D | D | NS | NS |
| Calendar intern | D | N | D | N |
| Gestionare ședințe | D | D | D | D |
| Serviciu de mail | D | D | D | D |
| Gestionare donatori | N | N | NS | D |
| Generare de rapoarte | D | N | D | D |
| Personalizare în funcție de tipul de ONG | N | N | D | D |
| Interacțiune directă a beneficiarilor cu sistemul | D | N/A | N | N |

Notă: D = DA, N = NU, N/A = nu este aplicabil, NS = Nu se Știe

[12] https://www.preciofishbone.se/en/ngoonline/ngo-online/
[13] https://civicrm.org/

Funcționalitățile ce presupun gestionarea datelor specifice organizației sunt prezente în toate sistemele, fiind natural, considerând scopul pentru care acestea au fost create și anume tocmai pentru un management eficient al datelor interne organizațiilor.

Confidențialitatea informațiilor personale se referă la implementarea unei politici generale de protecție a datelor, în conformitate cu normele impuse prin regulația din Uniunea Europeană[14].

Generarea de rapoarte este folosită pentru analiza vizuală a activităților desfășurate și modul de folosire a resurselor, precum și disponibilitatea acestora în urma perioadei care a trecut.

Gestionarea donatorilor nu este prezentă în sistemul propus deoarece s-a plecat de la prezumția că organizațiile care folosesc acest sistem nu au ca și scop principal obținerea de noi donatori sau colaborarea cu cei actuali, ci mai degrabă se dorește un sistem care să-i susțină în activitățile de zi cu zi în gestionarea datelor.

Personalizarea sistemului în funcție de tipul de ONG a fost descoperită la sistemele NGO Online și CIVICRM și ea presupune particularizarea unui sistem generic în funcție de preferințele unei asociații, luând în considerare tipul de activități desfășurate, tipul de date cu care lucrează și funcționalitățile pe care și le doresc acestea. Motivul pentru care această funcționalitate a fost notată ca nefiind prezentă în sistemul dezvoltat este faptul că nu s-a urmărit implementarea unui sistem generic, care mai apoi să fie personalizat în funcție de fiecare tip de ONG, ci s-a plecat de la ideea inițială de organizație caritabilă care întreprinde acțiuni de distribuție non-profit.

Interacțiunea directă a beneficiarilor cu sistemul este o funcționalitate în plus, propusă pentru sistemul care se are în vedere în această lucrare. Cu aceasta se dorește atingerea unei eficiențe ridicate de gestionare a nevoilor unor persoane. Astfel, cei care au acces la internet și un dispozitiv mobil android compatibil, vor putea să adauge sau să elimine nevoi la cazul lor social.

La finalul acestui capitol ar trebui să fie cunoscute detalii generale despre sistemele de management al informațiilor și sistemele similare celui propus. Totodată, s-au făcut cunoscute funcționalitățile comune acestor sisteme, cu motivările de rigoare în ceea ce privește adoptarea sau nu a acestora în sistemul dezvoltat.

[14] https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

# Capitolul 4. Analiză și fundamentare teoretică

Acest capitol introduce cerințele funcționale și non-funcționale identificate pentru sistemul propus, actorii și modelele cazurilor de utilizare. Pentru fiecare actor este alcătuită o descriere detaliată a unui caz de utilizare semnificativ. Este prezentată, de asemenea, diagrama conceptuală a arhitecturii sistemului, precum și tehnologiile folosite, motivând alegerea făcută.

## 4.1. Cerințe

### *4.1.1. Cerințe funcționale*

Pe baza informațiilor prezentate în lucrarea [4] s-au extras atribuțiile și acțiunile specifice care se execută și sunt următoarele:

- Prezentarea raportului de activitate pentru perioada anterioară
- Gestionarea eficientă a resurselor
- Asigurarea unei planificari organizaționale eficiente
- Asigurarea integrității etice

Un sistem de management a unor astfel de organizații ar trebui să ofere soluții pentru:

- Managementul informațiilor
- Confidențialiatea informațiilor cu caracter personal
- Managementul resurselor umane
- Gestionare donatori
- Contabilitate și buget
- Organizare internă: programare ședințe și calendar personal
- Raportarea și controlul financiar
- Achiziții de materiale

Mapând aceste soluții pe sistemul propus și luând în considerare particularitățile funcționale propuse pentru soluția prezentă, se prezintă următoarea listă de cerințe funcționale:

1. Înregistrare: posibilitatea creării un cont pentru utilizarea aplicației
2. Autentificării unei persoane la organizația de care aparține. Utilizatorul rămâne autentificat, până când se deconectează.
3. Gestionarea informațiilor legate de cazurile sociale ale unei organizații (care au asociate persoanele beneficiare și nevoile acestora). Această funcționalitate include: vizualizarea datelor legate de persoane și nevoi, eliminarea unor persoane din cazul social și adăugarea de noi cazuri sociale (cu persoane și nevoi asociate)

4. Confidențialitatea informațiilor cu caracter personal, respectând normele GDPR[15] prin obținerea acordului fiecarei persoane pentru stocarea și procesarea datelor personale, în urma unei informări corecte
5. Gestionarea resurselor umane, care reprezintă membrii asociației și persoanele beneficiare, reprezentând un caz social
6. Gestionarea resurselor care servesc la întreprinderea acțiunilor sociale, și anume a produselor și stocurilor disponibile în cadrul unui ONG
7. Distrubuirea eficientă a stocurilor la persoanele care au nevoie de ele, prin maparea directă a unui stoc de nevoi (se oferă sugestie de stoc disponibil pentru o anumită nevoie)
8. Organizare internă, reprezentând posibilitatea de programare a ședințelor interne și a transferurilor de pachete către cazuri sociale, cu ajutorul unui calendar personal, precum și actualizarea status-ului în care se află o ședință sau un transfer
9. Adăugare nevoi: un beneficiar poate să adauge noi nevoi la cazul său social
10. Aprobare utilizatori în așteptare: administratorul trebuie să activeze noile conturi create și să le asigneze permisiuni pentru accesul la aplicație
11. Raportarea activității pentru o perioadă trecută, prin generarea automată de rapoarte vizuale
12. Opțiunea ”mi-am uitat parola”, pentru resetarea acesteia în cazul în care a fost uitată, prin trimiterea pe mail a unei noi parole generate
13. Schimbarea parolei: fie în urma resetării, fie din dorința utilizatorului
14. Vizualizare detalii generale despre profilul de utilizator
15. Deconectare

După cum se poate observa, funcționalitățile propuse diferă într-o anumită măsură de soluțiile identificate mai sus, deoarece sistemul prezent urmărește gestiunea unor resurse existente. Nu este relevantă pentru activitatea acestor tipuri de organizații gestionarea vreunui buget sau obținerea de noi resurse sau achiziții, ținând cont de faptul ca pachetele pentru ajutorul social sunt primite în mod regulat.

Cerințele funcționale în funcție de utilizator sunt prezentate în tabelul 4.1.

[15] GDPR (General Data Protection Regulation) este o regulație în legea UE ce privește protecția datelor și intimitatea tuturor persoanelor fizice din UE

Tabel 4. 1 Maparea funcționalităților la tipurile de utilizatori

| ID | Cerință funcțională | Utilizator |
|---|---|---|
| 1 | Înregistrare | M, A, B |
| 2 | Autentificare | M, A, B |
| 3 | Evidență cazuri sociale | M, A, B |
| 4 | Acord pentru prelucrarea datelor persoanale | M, A, B |
| 5 | Evidența persoanelor beneficiare | M, A |
| 6 | Gestionare stocuri | M, A |
| 7 | Sugestie de stoc disponibil pentru nevoi | M, A |
| 8 | Calendar- Programare/reprogramare întâlniri/ședințe | M, A |
| 9 | Adăugare nevoi | B |
| 10 | Aprobare utilizatori în așteptare | A |
| 11 | Generare rapoarte | M, A |
| 12 | Opțiune ”Mi-am uitat parola” | M, A, B |
| 13 | Schimbare parolă | M, A, B |
| 14 | Detalii profil utilizator | M, A |
| 15 | Deconectare | M, A, B |

Notă: A = Administrator, B = Beneficiar, M = Membru asociație

### *4.1.2. Cerințe non-funcționale*

Cerințele non-funcționale reprezintă proprietățile și contrângerile sistemului. Ele mai sunt numite și atribute de calitate. Pentru sistemul propus s-au identificat următoarele cerințe non-funcționale:

- **Durabilitate** – datele necesare vor fi disponibile și tranzacțiile efectuate vor supraviețui în cazul în care sistemul este nefuncțional, fiind stocate într-o memorie non-volatilă (tip de memorie care poate returna informații stocate chiar și în urma unei căderi de curent).
- **Eficiență** – sistemul ar trebui să aibă un timp scurt de răspuns (valoare dorită: 3/4 ms). Acestă valoare variază în funcție de: mașina pe care rulează aplicația server, conexiunea la internet și semnalul dispozitivului mobil.
- **Securitate** – securizarea datelor de autentificare a utilizatorilor și restricționarea accesului pe anumite pagini ale aplicației, în funcție de permisiunile utilizatorilor.
- **Testabilitate** - asigurată prin testarea unitară și integrată, cu succes, a funcționalităților.

## 4.2. Cazuri de utilizare

În acest subcapitol se vor prezenta actorii sistemului și atribuțiile acestora, prin intermediul diagramelor cazurilor de utilizare. Pentru fiecare actor se va prezenta un caz de utilizare semnificativ, conținând o diagramă asociată actorului principal, precondiții,

postcondiții, o secvență de evenimente reprezentând un "happy flow"[16] și o alta reprezentând un flux alternativ.

### *4.2.1. Actori*

Utilizatorii care pot folosi sistemul sunt:

- Membrii asociației (MA) – au acces la funcționalitățile aplicației și drept de citire sau scriere, în funcție de cum le este asociat de către administrator
- Administrator (A) – poate fi directorul executiv al asociației sau persoana cu cea mai mare autoritate. Acesta trebuie să aprobe cererile de înregistrare a restul utilizatorilor și să le dea drepturi
- Beneficiar (B) – acest tip de utilizator este reprezentativ pentru persoanele care beneficiază de pachete de la asociație. Funcționalitatea principală este aceea de a putea specifica nevoile pe care un astfel de utilizator le are.

### *4.2.2. Diagramele cazurilor de utilizare*

În figurile 4.1 și 4.2 sunt ilustrate diagramele cazurilor de utilizare (notate CU) pentru cei trei actori reprezentativi sistemului și anume: administrator, membru al organizației și beneficiar. Pentru utilizatorii memru asociație și administrator s-a construit o singură diagramă datorită faptului ca multe din cazurile de utilizare sunt comune pentru aceștia.

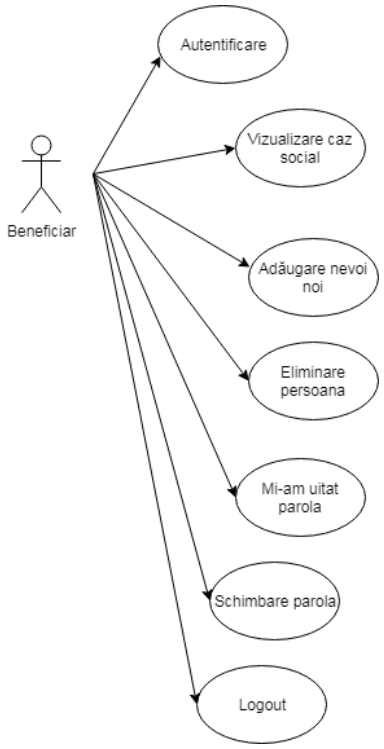



Figura 4. 1 Diagrama UML a CU pentru beneficiar

[16][16] Un happy flow reprezintă un scenriu cu acțiuni implicite care decurg fără erori sau condiții excepționale, producând un rezultat așteptat

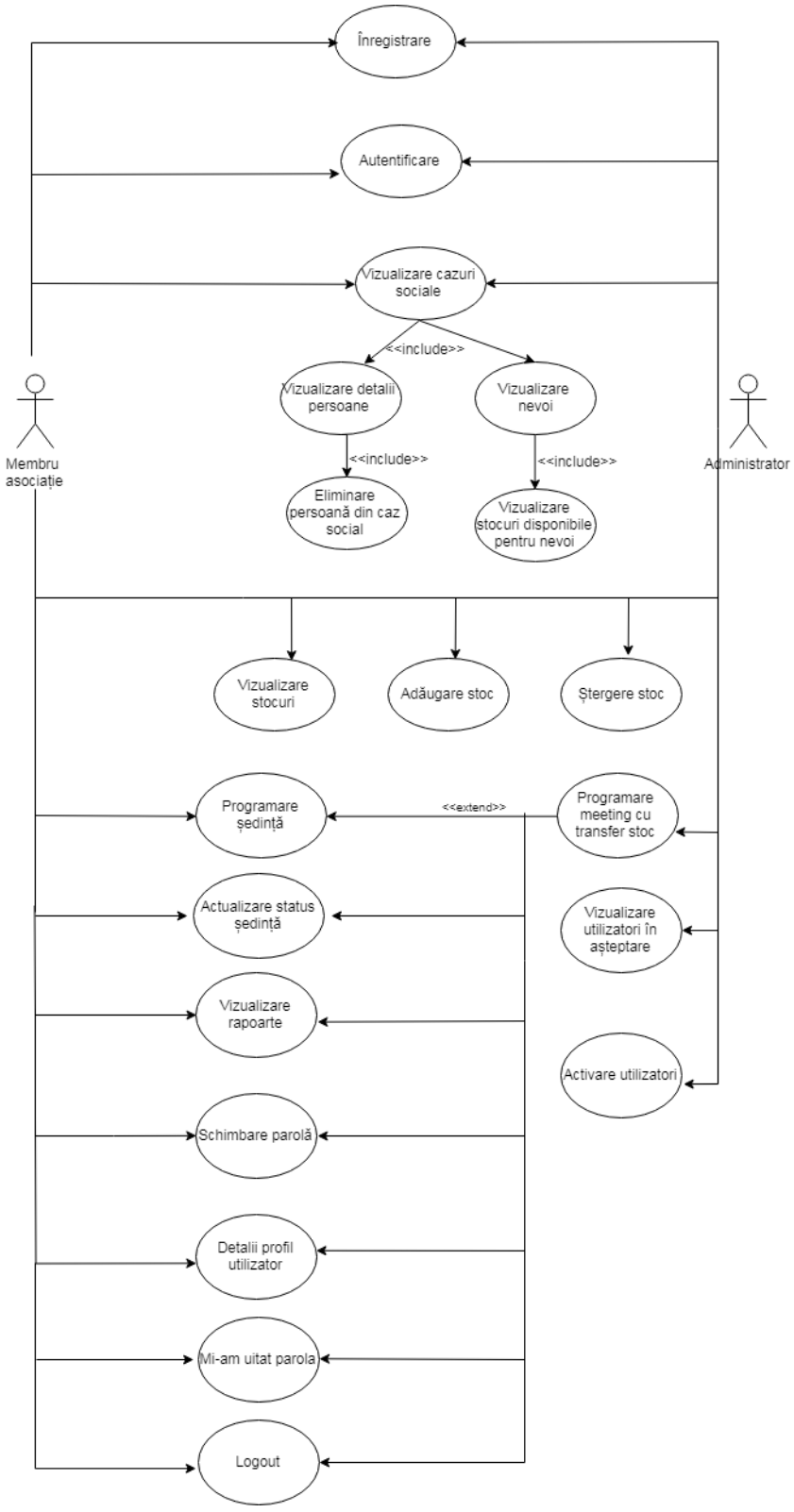



Figura 4. 2 Diagrama UML a CU pentru membru și administrator

### *4.2.3. Descrierea cazurilor de utilizare*

**Cazul 1 de utilizare (CU1)**

Denumire: Activarea unui cont

Actor principal: Administrator

Descriere: Administratorul sistemului poate activa contul unui utilizator care este în așteptare și poate asigna o permisiune de acces în aplicație.

Precondiții: Deținerea unui cont de administrator și existența unui cont de utilizator în așteptare.

Postcondiții: Activarea contului utilizatorului și asignarea permisiunii, având ca urmare posibilitatea utilizatorului de autentificare și folosire a sistemului conform drepturilor oferite.

Happy flow:

1. Actorul obține lista de utilizatori în așteptare generată de sistem
2. Actorul selectează un utilizator din listă
3. Actorul activează contul utilizatorului ales
4. Actorul asignează o permisiune dintre cele predefinite
5. Un mesaj de succes este afișat
6. Sistemul trimite în mod automat un mail de înștiințare către utilizator

Flux alternativ:

1. Lista de utilizatori este goală
    a. Actorul renunță la oprațiune sau revine la pasul 1 după a anumită perioadă de timp

**Cazul 2 de utilizare (CU2)**

Denumire: Autentificare

Actor principal: Membru asociație

Descriere: Pentru a accesa sistemul, utilizatorul trebuie să se autenfice, prin introducerea credențialelor (email și parolă).

Precondiții: Actorul deține un cont de utilizator activ.

Postcondiții: Utilizatorul este autentificat și are acces la datele și funcționalitățile oferite de sistem.

Happy flow:

1. Actorul alege optiunțea de autentificare.
2. Actorul introduce credențialele.
3. Actorul finalizează autentificarea prin apăsarea butonului ”Login”.
4. Sistemul verifică existența contului.
5. Sistemul validează credențialele.
6. Actorul este redirecționat către pagina de start.

Fluxuri alternative:

4. Cont inexistent sau inactiv
    a. Sistemul afișează un mesaj corespunzător.

b. Câmpurile completate se resetează.

c. Actorul repetă pașii începând cu 2.

5. Credențialele introduse sunt incorecte.
    a. Sistemul afișează un mesaj corespunzător.
    b. Câmpurile completate se resetează.
    c. Actorul repetă pașii începând cu 2.

**Cazul 3 de utilizare (CU3)**

Denumire: Vizualizare caz social

Actor principal: Membru asociație

Descriere: Un membru al unei asociației, autentificat în aplicație, poate sa vizualizeze detaliile unui caz social.

Precondiții: Actorul se afla pe pagina de cazuri sociale, iar aceasta pagini conține cel puțin un caz social.

Postcondiții: Detaliile disponibile pentru un caz social au fost puse la dispoziția actorului.

Happy flow:

1. Actorul selectează un caz social
2. Detaliile generale ale cazului social sunt afișate (persoane și nevoi)
3. Actorul selectează o persoană
4. Detaliile despre identitatea și statuturile persoanei respective sunt afișate prin intermediul unei ferestre de tip pop-up
5. Actorul revine pe pagina cazului social
6. Actorul selectează o nevoie
7. Detaliile nevoii selectate sunt afișate într-o fereastră de tip pop-up
8. Actorul revine pe pagina cazului social

Flux alternativ:

1. Actorul selectează un caz social
2. Detaliile generale ale cazului social sunt afișate
    a. Cazul social nu are asociat persoane sau nevoi
3. Actorul alege să:
    a. Revină la pagina cazurilor sociale
    b. Dezactiveze cazul social
        i. Un mesaj de confirmare este afișat
            1. Actorul renunța la acțiune de dezactivare și se revine pe pagina cazului social
            2. Actorul confirmă acțiune de dezactivare și cazul social este dezactivat

**Cazul 4 de utilizare (CU4)**

Denumire: Adăugare nevoie

Actor principal: Beneficiar

Descriere: Beneficiarul care are asociat un caz social poate să adauge o nouă nevoie.

Precondiții: Actorul este înregistrat și are asociat un caz social.

Postcondiții: O nouă nevoie este adăugată la cazul social al actorului.

Happy flow:

1. Actorul selectează opțiunea de adăugare a unei nevoi.

2. Actorul completează câmpurile necesare.
3. Actorul finalizează operațiunea.
4. Sistemul actualizează cazul social.
5. Sistemul afisează un mesaj de succes corespunzător.

Flux alternativ:
4. Sistemul detectează existența nevoii care se dorește să fie adăugată.
5. Sistemul afișează un mesaj de înștiințare corespunzător.

Pentru cazul de utilizare 3 s-a realizat o organigramă, ilustrată în figura 4.3, care să ilustreze vizual toate acțiunile și tot ce se implică acesta, în funcție de deciziile luate de utilizator. Pentru această organigramă s-au luat în considerare doar cazurile sociale care au asociate cel puțin o persoană și/sau o nevoie.

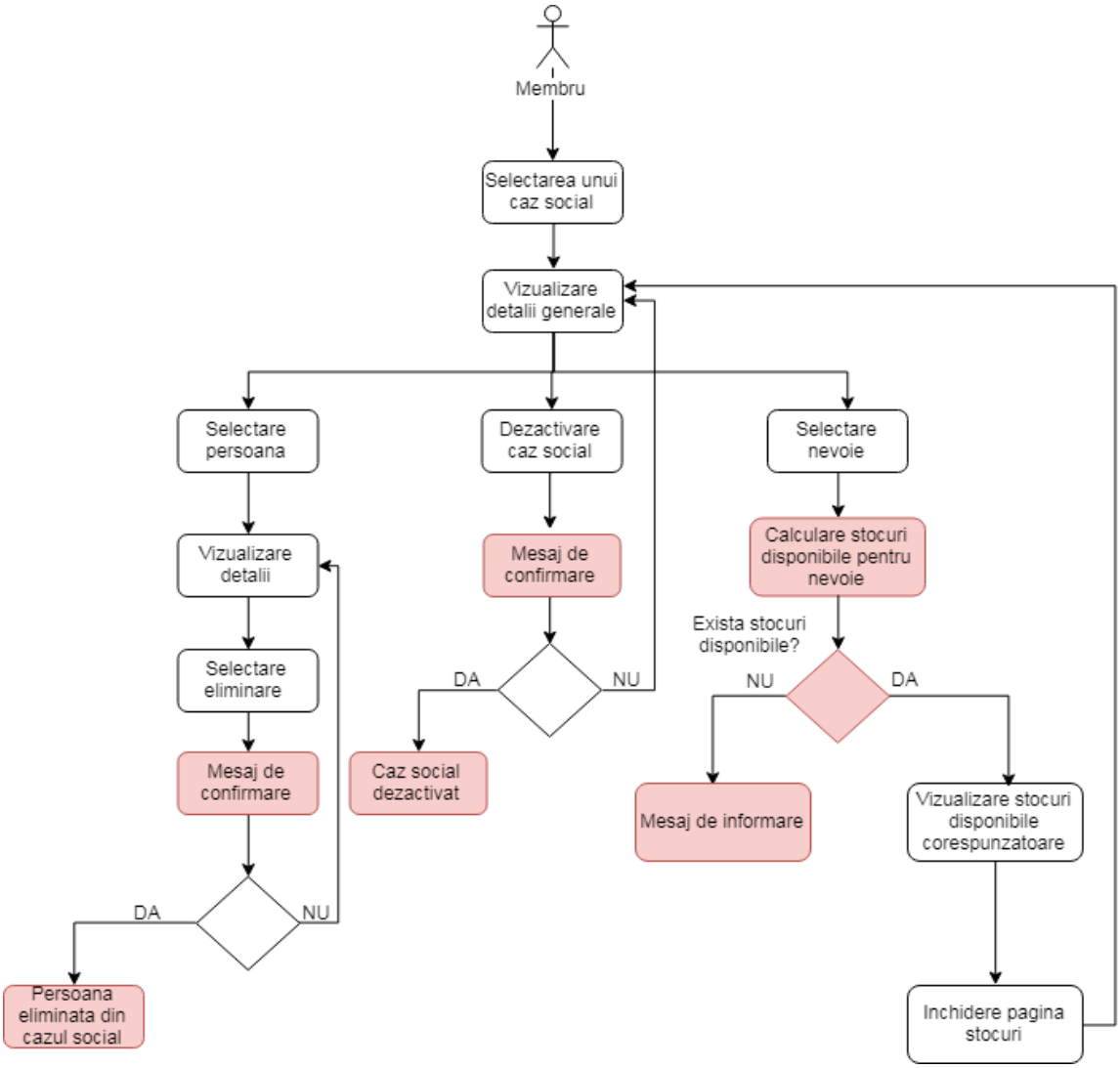



Figura 4. 3 Organigrama pentru CU3

Notă: Acțiunile corespunzătoare sistemului au fost marcate cu roșu în organigramă.

## 4.3. Arhitectura conceptuală a sistemului

În figura 4.4 este prezentată diagrama conceptuală a sistemului de tip client-server, alcătuită din cele două trei componente:

- Aplicație front-end (Client mobil)
- Server back-end (Server)
- Baza de date

Pentru server, care se ocupă de procesarea cererilor primite de la client, arhitectura a fost structurată pe nivele, fiecare dintre ele fiind dependente de nivelul de mai jos. Pornind de jos în sus, avem:

- Nivelul de mapări, de unde se fac apelurile către baza de date
- Nivelul de servicii (logică și implementare), unde se procesează datele preluate din baza de date și unde sunt implementați toți algoritmii necesari
- Nivelul de prezentare, unde sunt publicate funcționalitățile pe care clientul le poate apela.

Pentru client, care este aplicația cu care utilizatorul interacționează, s-a folosit o arhitectură bazată următoarele componente:

- Activități sau fragmente, care reprezintă interfața utilizator, unde se află componentele vizuale ale aplicației și unde sunt stabilite funcționalitățile acestora
- Modelul de vizualizare, care conține obiecte ce furnizează date pentru anumite componente de tip activitate sau fragment
- Repository, conține obiecte care implementează funcționalitatea logică de procesare a datelor
- Sursă de date la distanță și servicii web: datele care sunt folosite și procesate se află într-o bază de date aflată pe server, cu care aplicația client nu comunică prin intermediul unor servicii web, care apelează funcționalitățile publicate de către server.

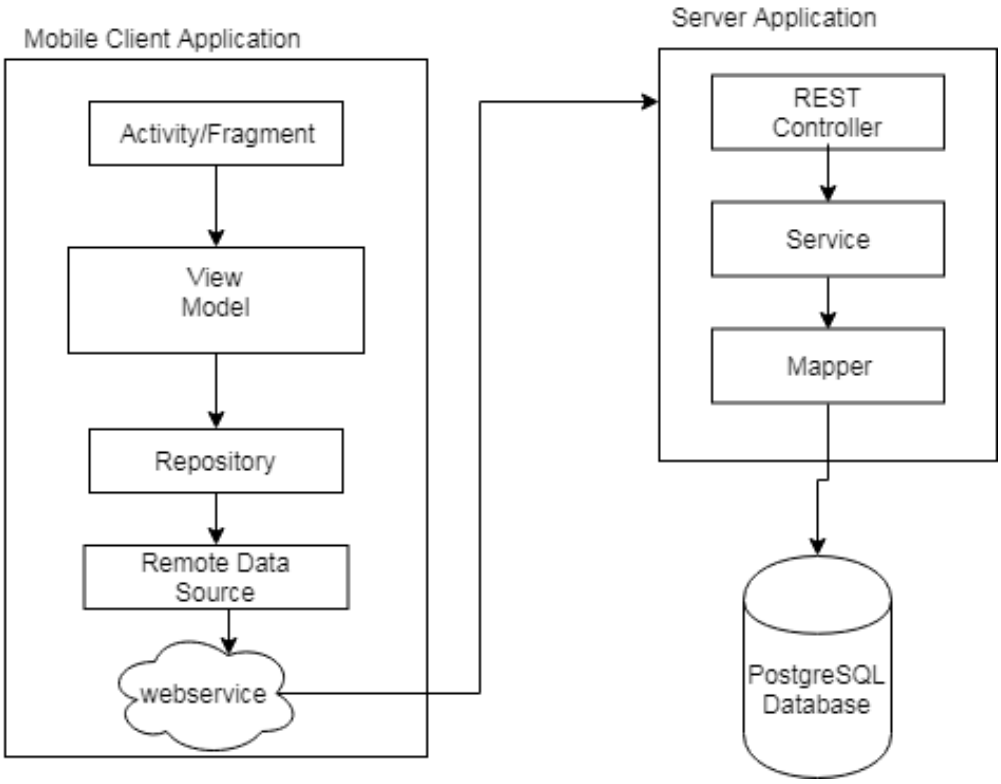



Figura 4. 4 Diagrama arhitecturii conceptuale

## 4.4. Aspecte tehnologice

În acest subcapitol se va oferi o descriere teoretică a soluției propuse, din punct de vedere tehnic, prezentându-se tehnologiile alese și motivația din spatele acestor alegeri, precum și evidențierea valorilor pe care ele le aduc sistemului implementat. În prima parte (alcătuită din secțiunile 4.4.1 – 4.4.5) se vor prezenta tehnologiile de bază folosite pentru implementarea serverului, iar în cea de-a doua parte se prezintă tehnologiile ce alcătuiesc aplicația client.

### *4.4.1. Framework-ul Spring Boot*

Sping Boot este un framework bazat pe limbajul de programare Java, care este folosit pentru crearea unui microserviciu. Un micro serviciu este o arhitectură care ii permite unui dezvoltator să dezvolte și să lanseze servicii în mod independent. Avantajele acestui framework sunt[17]:

- Dezvoltarea facilă a aplicației
- Ușor de înțeles
- Configurarea automată
- Oferă o aplicație Spring bazată pe adnotări

[17] https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

- Setare ușoară a proiectului

Spring Boot se bazează pe Spring, care este cel mai popular framework pentru dezvoltarea aplicațiilor Java la nivel de întreprindere, dar aduce însă în plus:

- Integrarea directă cu Tomcat, Jetty sau Undertow, care furnizează servere web pentru aplicații Java, unde acestea să poată rula
- Configurare automată (așa cum a fost menționat mai sus), care elimină nevoia de a configura manual fișiere XML
- Lansare ușoară a proiectului
- Reducerea numarului de linii de cod

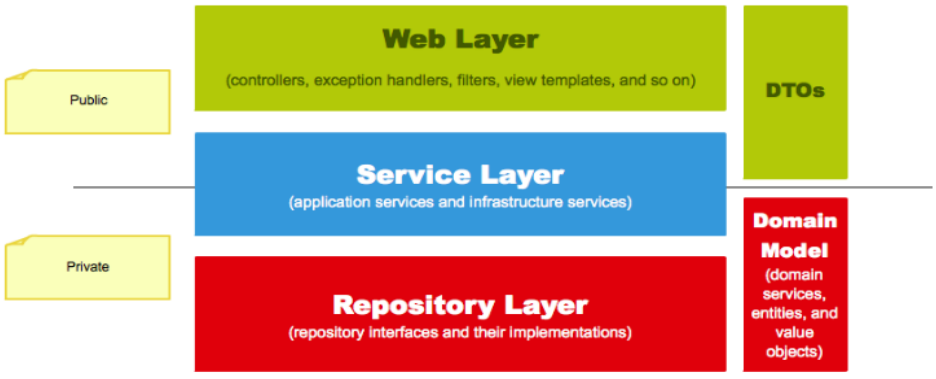
Pentru învățarea și utilizarea corectă a acestui framework s-a utilizat cartea [5], pe lângă resrusele disponibile online.

Cu ajutorului lui Spring Initializr[18], se pot genera proiecte Spring Boot de început, cu o structură stabilită și dependințele necesare, în funcție de nevoile fiecărei aplicații. Astfel se reduce timpul de setare și configurare a proiectului.

În figura 4.5 este ilustrată o arhitectură clasică pentru aplicațiile Spring Boot, ale cărei nivele sunt:

- Nivelul web, unde se află funcționalitățile făcute publice și puse la dispoziția aplicației client, pentru a putea fi apelate
- Nivelul de servicii, unde se găsește implementarea funcționalităților și a logicii
- Nivelul repository, unde se află interfețele și implementările acestora, pentru accesul la baza de date.

Modelul de domeniu specifică modelul intern al aplicației și este alcătuit din cele trei tipuri de obiecte (specificate în diagramă), care ar trebui folosite doar în implementare și nu făcute publice clientului. Obiectele DTO[19] sunt folosite pentru a transmite date între diferite procese și nivele ale aplicației, iar acestea sunt cele care ar trebui făcute publice pentru client.




[20]

Figura 4. 5 Arhitectura unei aplicații Spring Boot

[18] https://start.spring.io/
[19] Data Transfer Object
[20] https://www.petrikainulainen.net/software-development/design/understanding-spring-web-application-architecture-the-classic-way/

### *4.4.2. MyBatis*

MyBatis[21] este un framework folosit pentru persistența datelor, care oferă suport pentru SQL customizat, proceduri stocate și mapări avansate. Pentru setarea conexiunii la baza de date și mapările și operațiile asupra datelor se pot folosi adnotări sau fișiere XML, ceea se elimină nevoia de configurări manuale pentru aceste acțiuni. În sistemul dezvoltat s-au ales adnotările pentru mapările și operațiile asupra datelor.

Spre deosebire de alte framework-uri ORM, care mapează obiecte Java la tabelele din baza de date, MyBatis mapează metodele Java la instrucțiunii SQL, oferind control în întregime asupra execuției celor din urmă.

Având la dispoziție, pe lângă MyBatis, framework-urile JDBC și Hibernate, motivele pentru care cel dintâi a fost ales sunt:

- Controlul asupra frazelor SQL
- Maparea automată a setului de rezultate returnat la obiectele predefinite ca și parametrii metodei
- Simplificarea conexiunii la baza de date din punctul de vedere al codului
- Integrare ușoară cu aplicații Spring

### *4.4.3. PostgreSQL*

PostgreSQL[22] (cunoscut ca și Postgres) este un sistem open-source de management a bazelor de date relaționale. Este baza de date implicită pentru serverul macOS, fiind disponibilă și pentru Linux și Windows. Suportă trazacții ce îndeplinesc proprietățiile ACID (Atomicitate, Consistență, Izolare, Durabilitate), vederi automat actualizabile, triggere, chei străine și proceduri stocate. Sunt folosite tabele, alcătuite din rânduri, care la rândul lor conțin un set de coloane și sunt identificate unic printr-o cheie primară. Pentru asigurarea integrității atunci când sunt referite două tabele în relație una cu alta se folosesc chei străine. Pentru utilizarea PostgreSQL s-a folosit ca și referință cartea [6] și documentația oficială disponibilă online [7].

Printre companiile care folosesc PostgreSQL se numără: Apple, IMDB, Debian, Red Hat, Cisco, Skype ș.a.

Motivele pentru care s-a ales PostgreSQL ca și sistem de management a bazei de date pentru aplicație sunt:

- Soportul pentru obiectele definite de utilizator
- Flexibilitate și robustețe
- Posibilitatea crării facile a unui nou tip de date (utilizată pentru crearea enumărărilor folosite în multe dintre tabele)
- Posibilitate de stocare a unei cantități mari de date
- Integritatea datelor, conform proprietățiilor ACID.

În plus, se menționează și motivul personal, condus de dorința și curiozitatea de autilizare a unui nou sistem de management a bazelor de date.

[21] http://www.mybatis.org/mybatis-3/
[22] https://www.postgresql.org/

### *4.4.4. Gradle*

Gradle[23] este un sistem de automatizare a build-ului unei aplicații, bazat pe concepte specifice Apache Ant și Apache Maven. Acesta introduce o configurare bazată pe limbaj specific domeniului, în locul formularelor XML folosite de către Maven pentru configurarea proiectului.

Gradle oferă și suport pentru descărcarea și configurarea automată a dependințelor și a librăriilor.

Task-urile pe care Gradle le face disponibile (Figura 4.6) pot fi executate atât din interfață, cât și prin rularea comenzilor manual din terminal. De exemplu, pentru executarea unui build, acesta se poate selecta din meniul Gradle, după cum apare și în figura 4.6, sau se poate introduce în terminal-ul mediului de dezvoltare comanda: gradle build.

În Figura 4.7 se poate observa structura unui proiect Gradle. Sunt specifice fișierele build.gradle (unde sunt definite dependințele și plug-in-urile și unde pot fi definite task-uri specifie, pe lângă cele puse la dispoziție) și settings.gradle (unde se definesc sub-proiecte). Ambele fișiere sunt scrise în limbaj specific de domeniu (DSL) Groovy.

S-a ales Gradle, în detrimentul sistemelor Maven și Ant, deoarece acesta combină proprietățile celor două într-o soluție implementată în limbaj DSL bazat pe Groovy, caracterizată ciclul de viață și ușurința utilizării specifice Maven, alături de puterea și flexibilitatea specifice Ant.

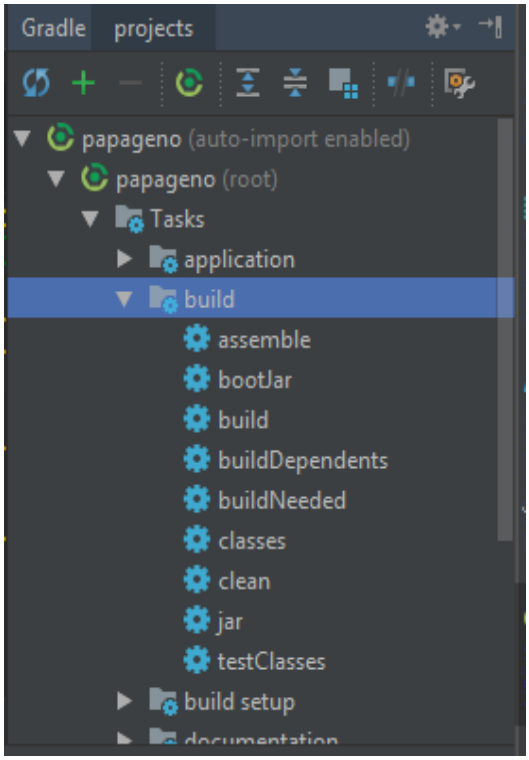



Figura 4. 6 Structura unui proiect Gradle

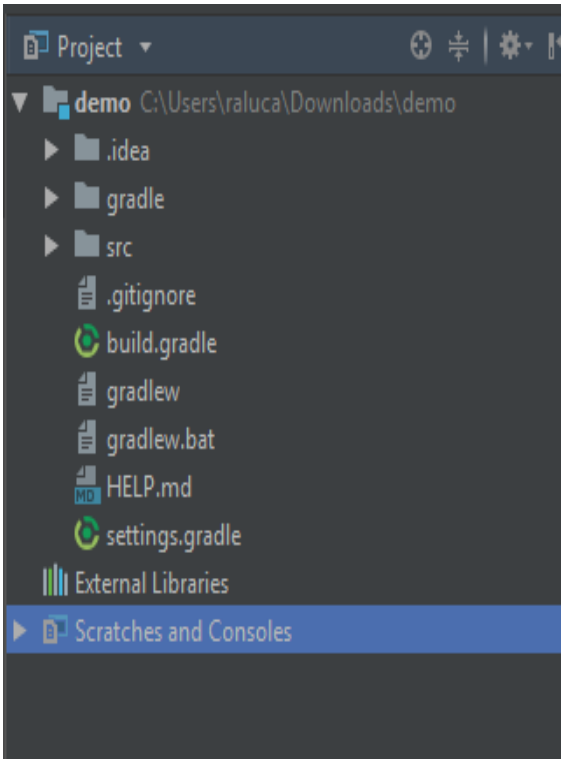



Figura 4. 7 Task-uri Gradle

[23] https://gradle.org/

### *4.4.5. Servicii Web RESTful*

Serviciile web RESTful (Representational State Transfer) sunt bazate pe arhitectura REST, care definește constrângerile pentru crearea serviciilor web, și sunt folosite pentru a crea și expune API-uri pentru aplicații web.

Scopul acestor servicii web este de a asigura comunicarea între client și server, făcând posibilă accesarea resurselor aplicației web . Protocolul de comunicare folosit este HTTP , care este cea mai utilizată metodă de acces a informațiilor în Internet. Resursele care pot fi accesate se pot reprezenta în formate diferite, cum ar fi XML sau JSON.

Pentru comunicarea între client și server sunt puse la dispoziție patru tipuri de metode, denumite cereri HTTP, iar acestea sunt:

HTTP GET: folosită pentru a trimite o cerere de obținere a unor resurse. Parametrii pot fi transmiși folosind adnotările: @PathVariable (extragerea unei valoari direct din URL) sau @QueryParam(pentru transmiterea unui parametru de interogare)

- HTTP POST: folosită pentru a transmite date la server, cu intenția de a fi stocate de către acesta. Datele sunt transmise în corpul metodei și parametrul care este primit este notat cu @RequestBody
- HTTP PUT: folosită pentru a transmite date la server, cu scopul de a fi subsitui valorile vechi (actualizarea datelor). La fel ca si la metoda POST, obiectul este transmis în corpul metodei, fiind notat cu @RequestBody
- HTTP DELETE: folosită pentru a cere serverului eliminarea unei anumite resurse, identificate printr-un identificator de obicei trimis ca si parametru de tip @PathVariable.

S-a ales implementarea serviciilor web REST datorită arhitecturii flexibile, a specificațiilor mai puțin restrictive, a disponibilității diferitelor tipuri de formate pentru mesaje (XML, HTML, JSON) și a performației mai ridicate, în comparație cu serviciile web SOAP, care sunt implementate conform protocolului SOAP, bazat pe limbaj XML.

În urătoarele secțiuni se vor prezenta tehnologiile folosite pe partea de client, care este o aplicație mobilă pentru sistemul de operare Android.

### *4.4.6. Android*

Android[24] este un sistem de operare pentru dispozitive mobile, dezvoltat de Google. Este bazat pe o versiune de Linux kernel și este destinat dispozivivelor cu touchscreen, cum ar fi tabletele sau telefoanele inteligente.

Cateva dintre caracteristicle Android sunt:

- Actualizări Google: aplicațiile se actualizează automat de catre Google
- Google Assistant: asistent personal digital oferit de Google, care este tot timpul gata să ajute (de la a efectua un apel, la a găsi informații, a deschide aplicații și multe altele)
- Securitate: Google Play Protect scanează în fiecare zi aplicațiile
- Milioane de aplicații disponibile

[24] https://www.android.com/

În figura 4.8 se poate observa o distribuție relativă a dispozitivelor în funcție de versiunea de Android pe care acestea rulează.

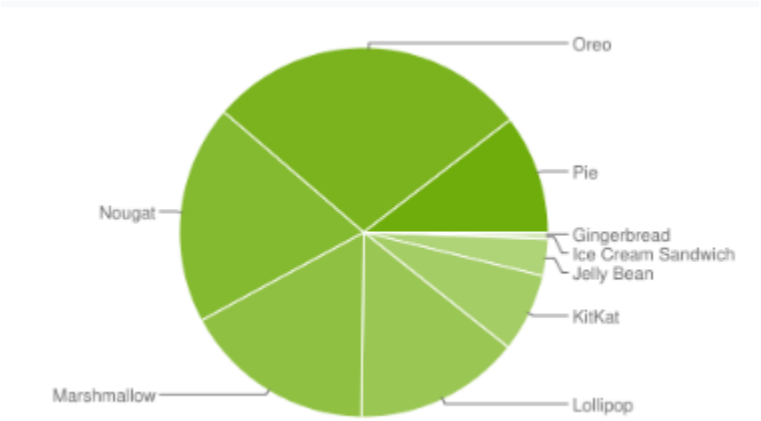



Figura 4. 8 Distribuția vizuală a dispozitivele în fucție de versiunea Android [25]

Motivele pentru care am ales Android ca și sitem de operare pe care a fost dezvoltată aplicația sunt: disponibilitatea dispozitivului mobil personal, posibilitatea de a putea dezvolta aplicația în limbajul de programare Java și preferința personală în favoarea acestui sistem de operare.

### *4.4.7. Retrofit*

Retrofit[26] este un client HTTP type-safe pentru Android și Java care facilitează primirea de obiecte JSON (sau alte date structurate) prin servicii web de tip REST. Acesta transformă un HTTP API într-o interfață Java și funcționează modelând peste un URL de bază declarat. Fiecare metodă din interfață este adnotată cu @GET, @POST, @PUT sau @DELETE (specificând tipul de API) și este specificată ca și parametru extensia URL-ului de bază. Aceste date specificate pentru metode sunt suficiente pentru Retrofit să genereze implementări funcționale. Cu ajutorul adnotărilor @Path și @Query se pot specifica și trimite parametrii pe care API-ul îi așteaptă sau de care are nevoie.

Retrofit nu are o problemă cu proprietăți lipsă, deoarece mapează doar ceea ce are nevoie, sau dacă trebuie să adauge proprietăți noi într-un JSON. De aceea este destul de flexibil.

Motivul pentru care am ales Retrofit ca și client HTTP pentru această aplicație este ușurința de utilizare, funcționalitățile sale high-level, care construiesc URL-ul în funcție de variabilele furnizate de dezvoltator, parsarea automată a obiectelor JSON și configurarea ușor de efectuat.

[25] https://developer.android.com/about/dashboards
[26] https://s-quare.github.io/retrofit/

### *4.4.8. Encripția parolelor*

Pentru asigurarea securității în ceea ce privește transferul de credențiale între client și server s-a implementat un codificator oferit de Base64[27], care este un grup de scheme pentru codarea datelor binare în text reprezentat în format radix-64[28]. Codifocatul folosește un cifru ce utilizează standardul avansat de criptare simetrică AES[29] și modul ECB(”Electronic CodeBlock”)[30] care reprezintă cea mai simplă modalitate de criptare în care mesajele sunt împărțite în blocuri și fiecare bloc este criptat separat. Pentru asigurarea dimensiunii de 8 biți a blocurilor se folosește PKCS5Padding[31] care adaugă biți de umplutură. Folosind această modalitate de encripție, la transferul parolelor între client și server, acestea se vor trimite în format criptat.

### *4.4.9. MPAndroidChart*

MPAndroidChart[32] este o librărie care poate fi folosită pentru generarea diferitor tipuri de diagrame în Android. Este suportată personalizarea lor și interacțiunea cu acestea. Datorită modalității simple de utilizare și multitudinii de funcții oferite este printe cele mai folosite librării pentru generare de diagrame. În acest proiect s-a folosit această librărie pentru generearea diagramelor radiale care facilitează monitorizarea stocurilor unei organizații.

### *4.4.10 SMTP*

SMTP (Simple Mail Tansfer Protocol)[33] este un protocol de comunicare care face parte din nivelul de aplicație a protocolului TCP/IP și este folosit pentru transmiterea de mail-uri electronice între servere. Majoritatea sistemelor care trimit E-mail-uri pe internet folosesc SMTP. Mesajele pot fi consumate folosind un client de E-mail, care poate utiliza unul dintre protocoalele POP (Post Office Protocol) sau IMAP (Internet Message Access Protocol). Componentele SMTP sunt:

- MUA (Mail User Agent) – agent care trimite și primește mesajele unui utilizator
- MTA (Mail Transfer Agent) – agent care mesaje între domenii
- MDA (Mail Delivery Agent) – agent care stochează mesajele în cutia poștală electronică a utilizatorului

În aplicația dezvoltată s-a folosit acest protocol pentru trimiterea de E-mail-uri la utilizatorii care vor să își reseteze parola.

[27] https://en.wikipedia.org/wiki/Base64
[28] https://en.wikipedia.org/wiki/Radix
[29] https://ro.wikipedia.org/wiki/AES
[30] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_Codebook_(ECB)
[31] http://www.herongyang.com/Cryptography/DES-JDK-What-Is-PKCS5Padding.html
[32] https://weeklycoding.com/mpandroidchart-documentation/
[33] https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

# Capitolul 5. Proiectare de Detaliu și Implementare

În acest capitol se vor prezenta pașii urmați și detaliile importante de proiectare și implementare a sistemului, care este alcătuit din trei componente: baza de date, server și aplicație mobilă. Ca și ghid pentru dezvoltare software s-a folosit cartea [8] pentru dezvoltare softare. Pentru dezvoltarea aplicației Android s-au folosit ca și resurse de învățare cartea [9] care conține elementele esențiale pentru dezvoltarea de aplicații Android bazate pe limbajul de programare Java și ghidul online de la Android pentru dezvoltatori [10].

În primele sub-capitole se va oferi o vedere de ansamblu asupra proiectului, o prezentare generală urmată de diagramele de clase pentru server și aplicație mobilă și diagrama modelului de date. În următoarele sub-capitole se vor detalia detaliile de implementare a unui algoritm relevant pentru aplicație și modul în care funcționează.

## 5.1. Prezentare generală

Aplicația dezvoltată este alcătuită din cele trei componente menționate mai sus, iar pentru dezvoltarea acestora s-au ales următoarele tehnologii:

- Pentru baza de date s-a ales sistemul de management pentru baze de date relaționale PostgreSQL, care rulează pe localhost, port 5432.
- Serverul s-a dezvoltat în Java, folosind ca și mediu de dezvoltare integrat IntelliJ IDEA
- Partea de client mobil a fost implementată în Android Studio, folosind tot Java ca și limbaj de programare.

## 5.2. Structura aplicației server

Serverul a fost construit pe o arhitectură de tip *Layered* (pe nivele), pentru a separa părțile de acces date, logică și control și pentru a oferi un nivel ridicat de abstractizare și de flexibilitate. În figura 5.1 este ilustrată diagrama pachetelor aplicației server, pentru a întețele în ansamblu structura aleasă. Din motive de claritate și pentru o înțelegere mai bună a structurii și a dependițelor s-a ales ilustrarea diagramei pachetelor și nu a claselor, a căror diagramă ar fi fost prea încărcată și neclară.

Codul sursă pentru aplicația server este compus din două module: main (modulul principal) și test. Modulul main conține toată implementarea propriu-zisă și configurările, iar modulul test conține teste pentru asigurarea corectitudinii codului și a comportamentului dorit.

### *5.2.1. Modulul principal*

În primul rând, se poate observa pachetul fără nicio dependință față de celelalte și anume ”Package Server”. Acesta conține clasa *ONGManagementServer,* care este clasa principală setată în configurarea de rulare. Ea încarcă și porneste aplicația din metoda main și este adnotată cu @SpringBootApplication(care poate fi folosită pentru autorizarea adnotărilor @EnableAutoConfiguration, @ComponentScan și @Configuration) și @ComponentScan(care specifică pachetul de unde să fie luate componentele aplicației).

Mai apoi, pornind de la pachetul de pe cel mai înalt nivel logic până la cel mai de jos, le avem următoarele, fiecare depinzând de cel de sub el.

**Pachetul controller**, unde avem toate clasele Controller, care conțin metodele expuse ca și REST API-uri. Clasele sunt adnotate cu: @RestController și @RequestMapping, care are ca și parametru un String constant, reprezentând calea (URL) către acel controller-ul respectiv.

**Pachetul service** conține interfețele și clasele de serviciu. Interfețele declară metodele care sunt disponibile public pentru un anumit serviciu, iar clasa de implementare conține toată logica de business pentru metodele din interfața pe care o implementează. Aceste metode declarate în interfețe sunt apelate din clasele controller. Clasele de implementare sunt adnotate cu @Service. Această abordare oferă un nivel mai înalt de abstractizare, făcând accesibile public doar metodele specificate în interfață.

**Pachetul mapper** conține interfețele pentru mapare, care fac apelurile către baza de date și conțin adnotări de mapare SQL pe metodele ce acces la baza de date. Pentru tipul operației SQL se poate specifica @Select, @Insert, @Delete sau @Update, având ca și parametru o constantă String reprezentând propoziția SQL de executat, iar pentru rezultate se poate specifica tipul prin @ResultType și modalitatea mapării, conținând o listă cu atributele din clasa Java specificată la tipul rezultatului, asociate cu coloanele din baza de date, notată cu @Results.

**Pachetul model** conține două sub-pachete: *base* și *dto*. Cel dintâi conține obiecte model, care sunt reprezentare unu-la-unu a tabelelor din baza de date, ca și obiecte Java. Al doilea sub-pachet conține obiectele de transfer, folosite pentru transferul datelor între nivelele de control și cel de serviciu, cât și pentru maparea rezultatelor returnate din baza de date pe un obiect dorit.

**Pachetul exception** conține clase care reprezintă excepții ce sunt aruncate în timpul rulării și extind *RuntimeException*. Ele sunt folosite pentru validari, notificarea rezultatului unei operații și raportarea unor erori apărute, fiind aruncate în general din interiorul claselor de implementare a serviciilor.

**Pachetul comparator** conține o clasă pentru compararea cazurilor sociale și implementează interfața Comparator. Este folosită pentru sortarea cazurilor sociale.

**Pachetul config** conține clasa în care este configurată conexiunea la baza de date PostgreSQL, având ca și atribute (variabile de instanță): driver-ul care implementează protocolul pentru conexiunea la baza de date, URL-ul către baza de date, numele de utilizator și parola. Acestea sunt notate cu @Value și valorile sunt preluate din fisierul application.yml.

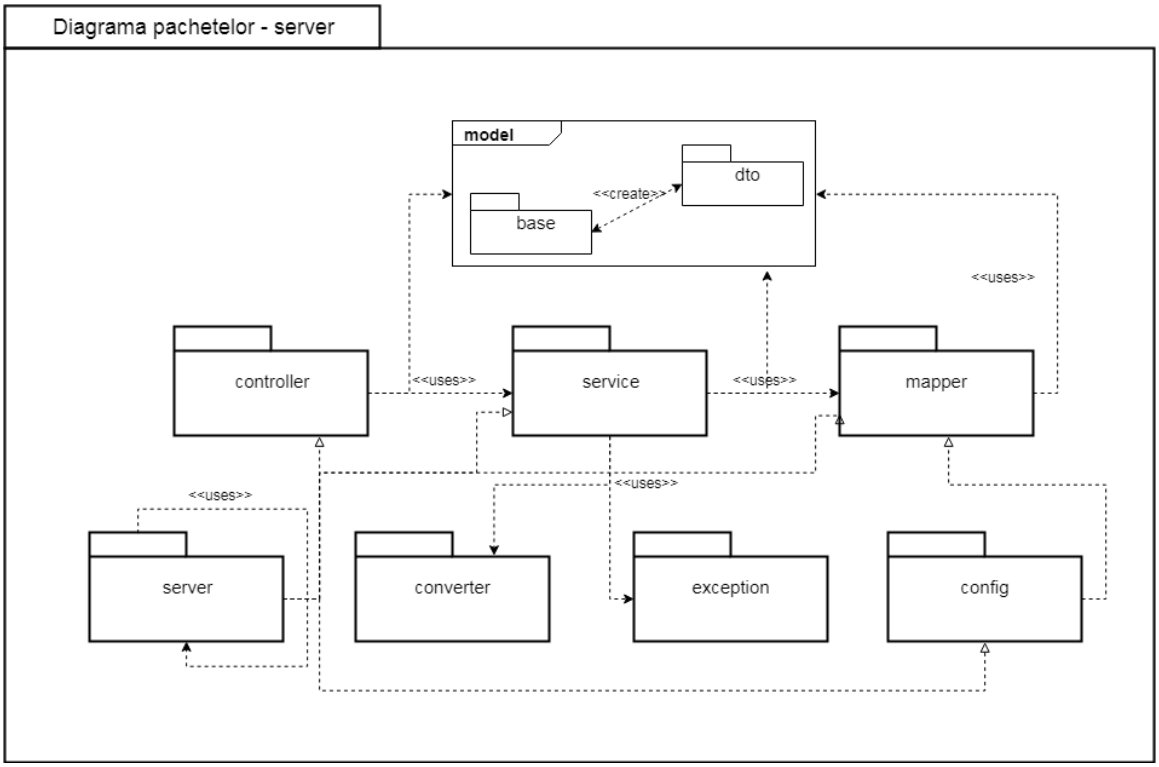



Figura 5. 1 Diagrama pachetelor de pe server

### *5.2.2. Modulul de test*

Acesta depinde de modulul principal, a cărui funcționalități le testează. Structura lui din mediul de dezvoltare integrat IntelliJ IDEA este ilustrată în figura 5.2.

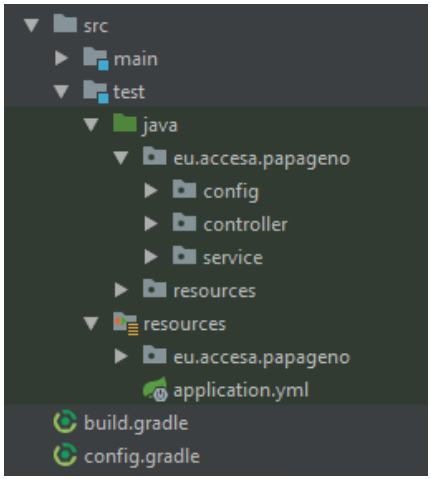



Figura 5. 2 Structura modulului de test

După cum se poate observa, acest modul este compus din trei pachete:

- Pachetul config – în care se testează conexiunea la baza de date
- Pachetul controller – în care se testează metodele la nivel de controllere
- Pachetul service – în care se testează funționalitățile implementate în clasele de servicii

## 5.3. Structura aplicației client

În figura 5.3 se poate observa diagrama pachetelor aplicației client, ale cărorcomponente (pachete) principale sunt: webservice, repository, viewmodel, model, fragment și activity. Ca și pachete auxiliare, care servesc la implementarea diferitor funcționalități sunt: validator, exception și adapter. În continuare se vor prezenta în ansamblu scopul fiecărui pachet.

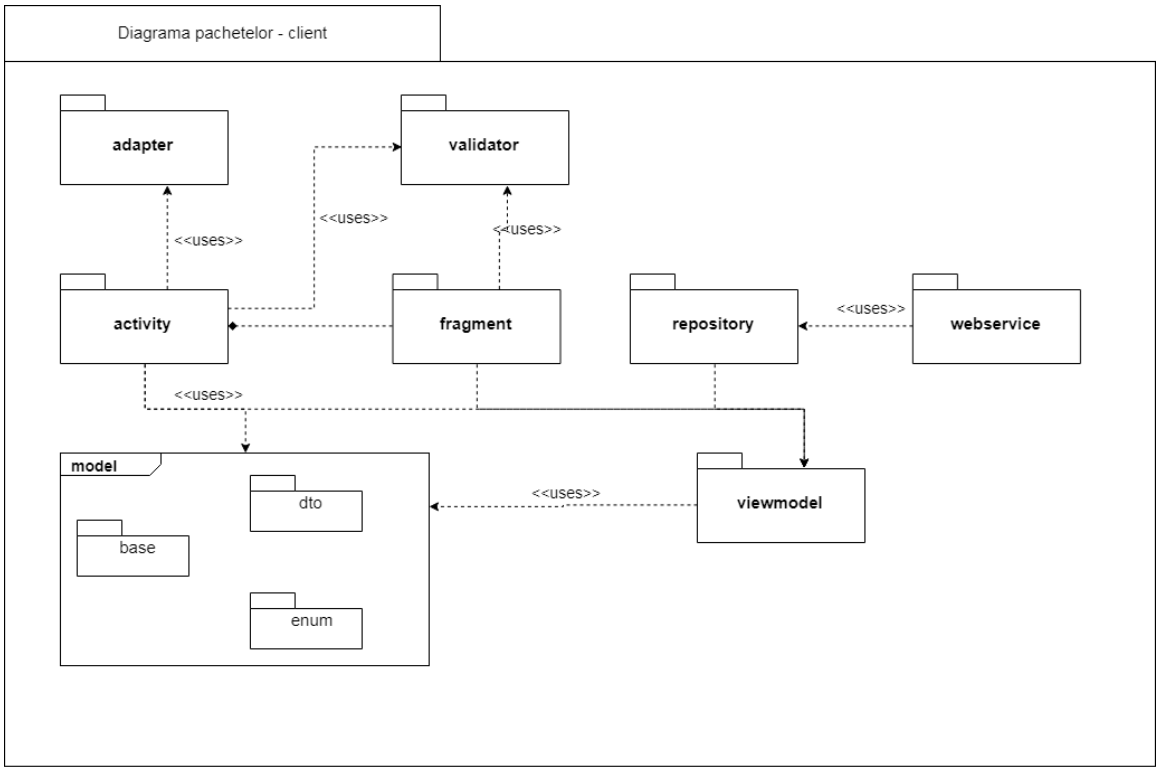



Figura 5. 3 Diagrama pachetelor aplicației client

**Pachetul webservice** conține interfețe care sunt folosite în comunicarea cu serverul, prin declararea și apelarea API-urilor REST pe care acesta din urmă le furnizează. Fiecare metodă are ca și tip returnat interfața Call care extinde Clonable care trimite cererea la serverul web și returnează răspunsul. De exemplu, *Call<SocialCase>* va face apel la server și va returna un obiect de tip SocialCase daca există, altfel va returna null.

**Pachetul repository** conține clase care se ocupă cu operațiile pe date, oferind o metodă de returnare a datelor ușoară pentru restul aplicației. Fiecare clasă repository conține o instanță a serviciului web asociat, instanțiat o singură dată și creat cu Retrofit, pe baza unui URL. Este implementată fiecare metodă declarată în interfața de serviciu, iar în această implementare este executat apelul (Call) menționat mai sus și este preluat și returnat rezultatul oferit de acest apel.

**Pachetul viewmodel** oferă încă un nivel de abstractizare. Acesta conține o instanță a clasei repository asociată și apelează separat fiecare metodă din această clasă, returnând rezultatele pentru activitatea sau fragmentul în care este folosită. Aceaste tipuri de clase nu știu modalitatea de obținere a datelor.

**Pachetul activity** conține clase activități (componente de bază a unei aplicații Android), care oferă o fereastră pentru construirea interfeței utilizator și în mod normal implementează o singură pagină din aplicație. Una dintre cele mai importante metode asociate cu o astfel de clasă este metoda *onCreate()* care rulează la crearea activității, inițializând componentele interfeței grafice. În interiorul acestor clase s-au făcut apelurile către metodele din clasele viewmodel, în mod asincron, folosind clase interioare private care extind *AsyncTask*. Astfel se evită rularea prea multor sarcini pe firul de execuție principal și odată cu aceasta, blocarea aplicației. Pentru ca o activitate să fie văzută de aplicație, aceasta trebuie declarată în fișierul *AndroidManifest.xml* (exemplu de conținut în figura 5.4). Aici se poate specifica și părintele unei activități și tema aplicată pe aceasta.

**Pachetul fragment** este alcătuit din clase de tip *Fragment*, care pot fi privite ca și sub-activități. Mai multe fragmente pot face parte dintr-o activitate, fiind strâns legat de aceasta și neputând fiind folosit independent de ea. Scopul pentru care s-au folosit fragmente în această aplicației a fost de a construi pagina de pornire astfel încât pe aceeași pagină, prin intermediul schimbării tab-ului să se poată executa atât autentificarea, cât și înregistrarea.

**Pachetul adaper** conține clase folosite pentru implementarea interfeței *Adapter*, utilizate în afișarea datelor într-o listă de tip *ListView*. Pentru fiecare listă utilizată în aplicație s-a construit câte un astfel de adaptor. Fiecare clasă folosește o resursă de tip *layout* pentru determinarea amplasării elementelor în listă.

**Pachetul validator** este alcătuit din clase menită să valideze anumite date introduse de către utilizator prin intermediul interfeței grafice (cum ar fi validarea formatului email-ului). Deși aceste validări se efectuează și pe partea de server, asigurarea lor și de pe partea clientului crește gradul de asigurare că datele primite sunt așa cum se așteaptă.

**Pachetul model** este compus din trei sub-pachete: **base** (conținele clasele de bază, la fel ca și în aplicația server), **dto** (conține obiectele de transfer folosite în același scop și fiind în principiu aceleași ca și în aplicația server) și **enums** (conține toate enumerările folosite ca și tipuri de date, fiind în relație 1-la-1 cu enumerările declarate în baza de date).

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.iosif.ongmanagement1">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ONGManagement1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:networkSecurityConfig="@xml/network_security_config">
        <activity
            android:name=".activity.LoginOrRegisterActivity"
            android:label="LoginOrRegisterActivity"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".activity.HomePageActivity"
            android:parentActivityName=".activity.LoginOrRegisterActivity"
            />
        <activity android:name=".activity.SocialCasesActivity"
            android:parentActivityName=".activity.HomePageActivity"
            android:theme="@style/Theme.AppCompat.Light.FullScreen"
        />
```

Figura 5. 4 Exemplu de conținut pentru AndroidManifest.xml

## 5.4. Modelul de date

În acest subcapitol se va prezenta diagrama ER (Entity Relationship) a modelului de date (figura 5.5). Acesta conține toate tabelele, cu atributele lor, prezente în baza de date folosită pentru aplicație, cât și relațiile dintre ele. Pentru identificarea unică a unei intrări în baza de date s-au folosit chei primare cu proprietate de auto-increment pentru generarea lor automată și în ordine, iar pentru relațiile între două tabele s-au folosit chei străine, care sunt reprezentate de o coloană în unul dintre tabele, aceasta făcând referință la cheia primară din cealaltă tabelă. Cheile străine au fost folosite pentru determinarea de ralații one-to-one și one-to-many. La relațiile one-to-one o intrare din tabelul care face referința corespunde unei intrări din tabelul referențiat, iar la relațiile one-to-many o intrare din tabelul referențiat poate corespunde mai multor intrări din tabelul care face referința.

Modelul bazei de date se află în forma normală Boyce-Codd. Pentru ca un tabel să se afle în această formă normală, acesta trebuie să îndeplinească următoarele criterii:

- Să se afle în forma normală 3 (FN3), ceea ce presupune:
  - Să se afle în forma normală 2 (FN2), pentru care trebuie:

- Să se respecte forma normală 1 (FN1), ceea ce presupune ca în fiecare coloană să existe valori atomice
- Să nu existe dependințe parțiale (un atribut nu poate să depindă doar de o parte a cheii primare, ci doar de întreaga cheie)

  - Să nu aibă dependințe tranzitorii (un atribit non-prim nu poate să depindă de un alt atribut non-prim, ci doar de unul prim/de cheie primară)

- Pentru orice dependință A->B (ceea ce înseamnă că A poate determina unic pe B și dacă B este un atribut prim, A nu poate fi un atribut non-prim), A trebuie să fie o super cheie (un set de una sau mai multe atribute care pot identifica unic un rând în tabel)

În continuare se vor prezenta detaliile fiecărui tabel prezent în baza de date.

**Tabelul organizations** conține organizațiile înregistrare și pentru care se poate folosi aplicația. Informațiile stocate în acest tabel sunt:

- id: cheia primară, tip de date: integer (dimensiune = 4 octeți)
- name: denumirea organizației, tip de date: character varying
- city: orașul din care face parte organizația, tip de date: character varying
- address: adresa la care se află sediul, tip de date: character varying

**Tabelul social_cases** conține cazurile sociale înregistrate la organizații. Informațiile pe care le conține sunt:

- id: cheie primară, tip de date: integer
- priority: prioritatea cazului social, tip de date: *priority_enum* (creată specific pentru această aplicație și conține valorile: “HIGH”, “MEDIUM” și “LOW”).
- organization_id: cheie străină, specifică organizația din care face parte cazul social, făcând referință la tabelul *organizations*, tip de date: integer
- title: titlul cazului social, tip de date: character varying
- dangerous: indică faptul că este periculos cazul social, tip de date: boolean
- disabled: indică dacă este dezactivat cazul social, tip de date: boolean

**Tabelul social_cases_needs** conține nevoile asociate cazurilor sociale. Informațiile pe care le conține sunt:

- id: cheie primară, tip de date: integer
- social_case_id: cheie străină care face legătura cu tabelul *social_cases*, specificând cazul social din care face parte o nevoie, tip de date: integer
- product_id: cheie străină care face referință la tabelul *products* și specifică produsul pentru care este o nevoie, tip de date: integer
- default_quantity: cantitatea inițială pentru o nevoie, tip de date: double
- current_quantity: cantitatea rămasă pentru ca nevoie să poată fi implinită, tip de date: double
- start_date: data începând cu care este nevoie de intrarea curentă din tabel, tip de date: date

- estimated_fulfilled_date: data la care este estimat ca nevoia să fie îndeplinită, tip de date: date
- time_period: perioada de timp pentru care nevoia este activă, tip de date: integer
- fulfilled: marchează dacă nevoia a fost sau nu împlinită, tip de date: boolean
- need_type: specifică tipul de nevoie, tip de date: *need_type_enum* (creată specific pentru această aplicație și conține valorile "ONE_TIME"- nevoia este active o singură dată "RECURSIVE"- nevoia este activă constant, pentru o perioadă de timp specificată în *time_period*)

**Tabelul products** conține detaliile produselor folosite doar pentru documentare, fără ne interesa existența sau nu a unui stoc. Informațiile pe care le conține sunt:

- id: cheie primară, tip de date: integer
- name: denumirea produsului, tip de date: character varying
- product_type: tipul produsului din stoc, tip de date: character varying

**Tabelul stocks** conține detaliile stocurilor pe care organizațiile le dețin. Coloanele pe care le conține sunt:

- id: cheie primară, tip de date: integer
- organization_id: cheie străină care face referință la tabelul *organizations* și specifică organizația de care aparține un stoc, tip de date: integer
- quantity: numărul de bucăți al produsului care alcătuiesc stocul, tip de date: double
- expiration_date: data la care produsele din stoc expiră, tip de date: date
- measurement_unit: unitatea de măsură a stocului (de exemplu: kg, bucată), tip de date: character varying

**Tabelul social_cases_persons** este un tabel de legătură, folosit pentru gestionarea relației între tabelele *social_cases* și *persons,* care s-ar afla în relație cu cardinalitate many-to-many,care a fost eliminată și înlocuită cu două relații one-to-many. Coloanele tabelului sunt:

- person_id: cheie străină care referențiază tabelul *persons,* tip de date: integer
- social_case_id: cheie străină care referențiază tabelul *social_cases,* tip de date: integer

Cele două coloane alcătuiesc o cheie primară compusă pentru tabel.

**Tabelul persons** conține datele persoanelor. Coloanele acestui tabel sunt:

- id: cheie primară, tip de date: integer
- first_name: prenumele persoanei, tip de date: character varying
- last_name: numele de familie, tip de date: character varying
- identity_card_id: cheie străină care referențiază tabelul *identity_cards,* tip de date: integer
- person_details_id: cheie străină care definește relația cu tabelul *person_details,* tip de date: integer
- org_id: cheie străină care referențiază tabelul *organizations* și specifică organizația de care aparține persoana, tip de date: integer

- dangerous: specifică dacă persoana este periculoasă sau nu (s-a întregistrat cel puțin o tentativă de fraudă), tip de date: boolean

**Tabelul person_details** conține detaliile asociate persoanelor. Coloanele sale sunt:

- id: cheie primară, tip de date: integer
- residence_id: cheie străină ce referențiază tabelul *residence_infos*, tip de date: intger
- handicap_id: cheie străină ce referențiază tabelul *handicaps,* tip de date: integer
- previous_earnings: reprezintă suma câștigată în ultima lună, tip de date: double
- occupational_status: ocupația profesională a persoanei, tip de date: *occupational_status_enum*(creată specific pentru această aplicație. Printre valorile conținute se numără: ” UNEMPLOYED”, ” EMPLOYED”, ”RETIRED” ș.a)
- marital_staus: situația maritală a persoanei, tip de date: *marital_status_enum* (creată specific pentru această aplicație. Printre valorile sale se numără: ”MARRIED”, ”DIVORCED”, ”WIDOW” ș.a)
- school_status: situația școlară sau academică a persoanei, tip de date: *academic_record_enum*(creată specific pentru această aplicație. Printre valorile sale se numără: ”WITHOUT EDUCATION”, ”PRESCHOOLER”, ”HIGHSCHOOL” ș.a)
- kinship: reprezintă relația unui minor cu adultul în grija căruia se află, tip de date: *kinship_enum* (creată specific pentru această aplicație. Printre valorile sale se numără: ”NATURAL”, ”ADOPTED”, ”IN TUTOR” ș.a)

**Tabelul identity_cards** conține detaliile documentelor de identitate a persoanelor, reprezentate prin coloanele:

- id: cheie primară, tip de date: integer
- cnp: codul numeric personal al persoanei, tip de date: character varying
- nationality: naționalitatea persoanei, tip de date: character varying
- serial_number: numărul seriei de pe buletin, tip de date: character varying
- released_by: specifică de cine a fost eliberat documentul, tip de date: character varying
- series: seria de buletin, tip de date: character varying
- identity_type: tipul documentului de identitate, tip de date: *identity_type_enum*(creată specific pentru această aplicație, având ca și valori: ”BI” – buletin și ”CI”- card de indentitate).
- released_date: data eliberării documentului, tip de date: date

**Tabelul residence_infos** conține informații referitoare la locurile de reședință ale persoanelor. Coloanele tabelului sunt:

- id: cheie primară, tip de date: integer
- landline: număr de telefon fix, tip de date: character varying
- mobile: număr de telefon mobil, tip de date: character varying
- email: adresa de email, tip de date: character varying
- address: adresa de domiciliu, tip de date: character varying

**Tabelul handicaps** conține detaliile despre dizabilități care au fost înregistrate la persoane. Coloanele care alcătuiesc tabelul sunt:

- id: cheie primară, tip de date: integer
- flag: specifică dacă o persoană are sau nu un handicap, tip de date: boolean
- handicap: denumirea sau descrierea scurtă a condiției, tip de date: character varying

**Tabelul users** conține utilizatorii aplicației. Coloanele care il alcătuiesc sunt:

- id: cheie primară, tip de date: integer
- permission: specifică tipul de permisiune/ nivelul de acces care îi este asociat unui utilizator, tip de date: *permission_enum*(enumerare creată pentru această aplicație, conține valorile: ”READ” – drepturi doar de citire și ”READ_WRITE”- drepturi de citire și de scriere)
- type: tipul de utilizator, tip de date: *user_type*(enumerare creată pentru aplicația dezvolatată, conținând valorile: ”ADMIN”, ”MEMBRU” și ”BENEFICIAR”)
- email: adresa de email a utilizatorului, tip de date: character varying
- password: parola contului de utilizator, tip de date: character varying
- first_name: prenumele utilizatorului, tip de date: character varying
- last_name: numele de familie, tip de date: character varying
- approved: specifică dacă utilizatorul a fost sau nu aprobat, tip de date: boolean

**Tabelul users_organizations** este un tabel de legătură pentru tabelele *users* și *organizations.* Colonele pe care le conține sunt:

- user_id: cheie străină ce referențiază tabelul *users*, tip de date: integer
- organization_id: cheie străină ce referențiază tabelul *organizations*, tip de date: integer

Cele două câmpuri alcătuiesc cheia primară compusă a tabelului.

**Tabelul meetings** conține ședințele programate de către utilizatori și este compus din următoarele coloane:

- id: cheie primară, tip de date: integer
- description: descrierea ședinței, tip de date: character varying
- conducted: specifică dacă a avut sau nu loc ședința, tip de date: boolean
- address: adresa la care a loc ședința/întâlnirea, tip de date: character varying
- user_id: cheie străină care face legătura cu tabelul *users* și specifică utiliatorul căruia îi aparține ședința, tip de date: integer
- meeting_date: data la care are loc ședința, tip de date: date
- time: ora la care a fost programată întâlnirea/ședința, tip de date: character varying

**Tabelul transfers** conține tranferurile de stocuri programate în cadrul unor ședințe. Coloanele sale sunt:

- id: cheie primară, tip de date: integer
- social_case_need_id: cheie străină, referențiază tabelul *social_cases_needs*, identificând nevoia care se vrea îndeplinită prin transfer, tip de date: integer
- quantity: cantitatea de stoc care trebuie transferat, tip de date: double

- status: statusul în care se află transferul, tip de date: *transfer_status_enum* (enumerare definită pentru această aplicație, care are ca și valori: ”IN PROGRESS”, ”DONE”, ”CANCELED”)
- meeting_id: cheie străină, referențiază tabelul *meetings*, tip de date: integer
- stock_id: cheie străină, referențiază tabelul *stocks* și identifică stocul din care se efectuează transferul, tip de date: integer
- organization_id: cheie străină, referențiază tabelul *organizations*, tip de date: integer
- creation_date: data la care transferul a fost programat, tip de date: date
- delivered_date: data la care a fost livrat transferul, tip de date: date

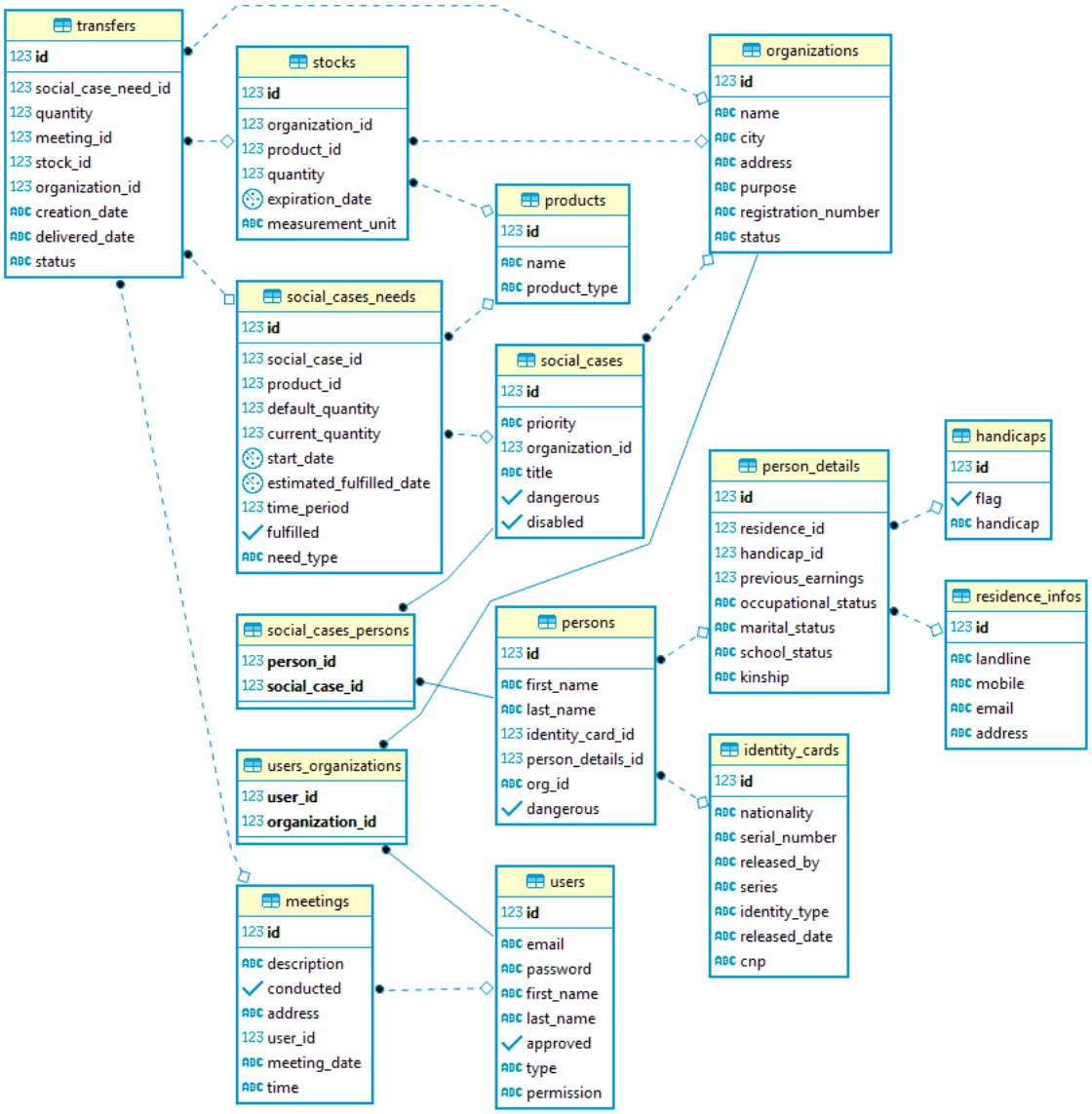



Figura 5. 5 Diagrama ER a bazei de date

## 5.5. Detalii de implementare funcționalități

În cadrul acestui sub-capitol se vor prezenta doi algoritmi relevanți în ceea ce privește funcționalitățile aplicației dezvoltate. Primul dintre ei este reprezentativ pentru funcționalitatea de adăugare a unui noi caz social, fiind ilustrat vizual printr-o organigramă (Figura 5.6), iar cel de-al doilea este o algoritm folosit pentru distribuirea eficientă și rapidă a stocurilor, pentru nevoi speciale.

### *5.5.1. Funcționalitatea de adăugare caz social*

Pentru adăugarea unui noi caz social s-au făcut numeroase validări, pentru a fi asigurată corectitudinea datelor și pentru împiedicarea duplicării aceluiași caz social. Ca și primă regulă, s-a stabilit că pentru a salva în baza de date a aplicației un nou caz social este necesar ca acesta să aibă asociat cel puțin o nevoie și/sau o persoană. Ca și a doua regulă stabilită este accea că nu pot exista două cazuri sociale cu același titlu și s-a hotărât ca și bună practică introducerea titlului astfel încât să conțină numele titularului cazului social sau numele familiei beneficiare. Pentru un caz social deja existent se pot adauga nevoi și persoane.

Formularul care trebuie completat pentru adăugarea unui nou caz social sau a unei persoane sau nevoi la un caz social poate fi observat în capitolul 7 (Figura 7.7).

În ceea ce privește datele persoanelor, se fac următoarele validări:

- Numele și prenumele să fie furnizate de către utilizator
- Datele cardului de identitate să fie introduse (seria, numărul, CNP-ul, cetățenia)
- CNP-ul să conțină 13 cifre

În ceea ce privește datele introduse pentru nevoi, se validează doar ca denumirea și cantitatea nevoii să fie introdusă, iar cea din urmă să fie diferită de 0.

Prin intermediul organigramei din figura 5.6 se ilustrează logica și fluxul deciziilor care se iau în funcție de datele introduse de către utilizator în formular. Semnificațiile culorilor folosite sunt:

- Galben – identifică acțiunile care vin din partea utilizatorului aplicației
- Albastru- identifică componentele de decizie
- Verde- identifică acțiunile care se iau pentru ramura de decizie corespunzătoare răspunsului ”DA” (când a fost trecut testul de dezicie)
- Roșu - identifică acțiunile care se iau pentru ramura de decizie corespunzătoare răspunsului ”DA” (când nu a fost trecut testul de decizie)

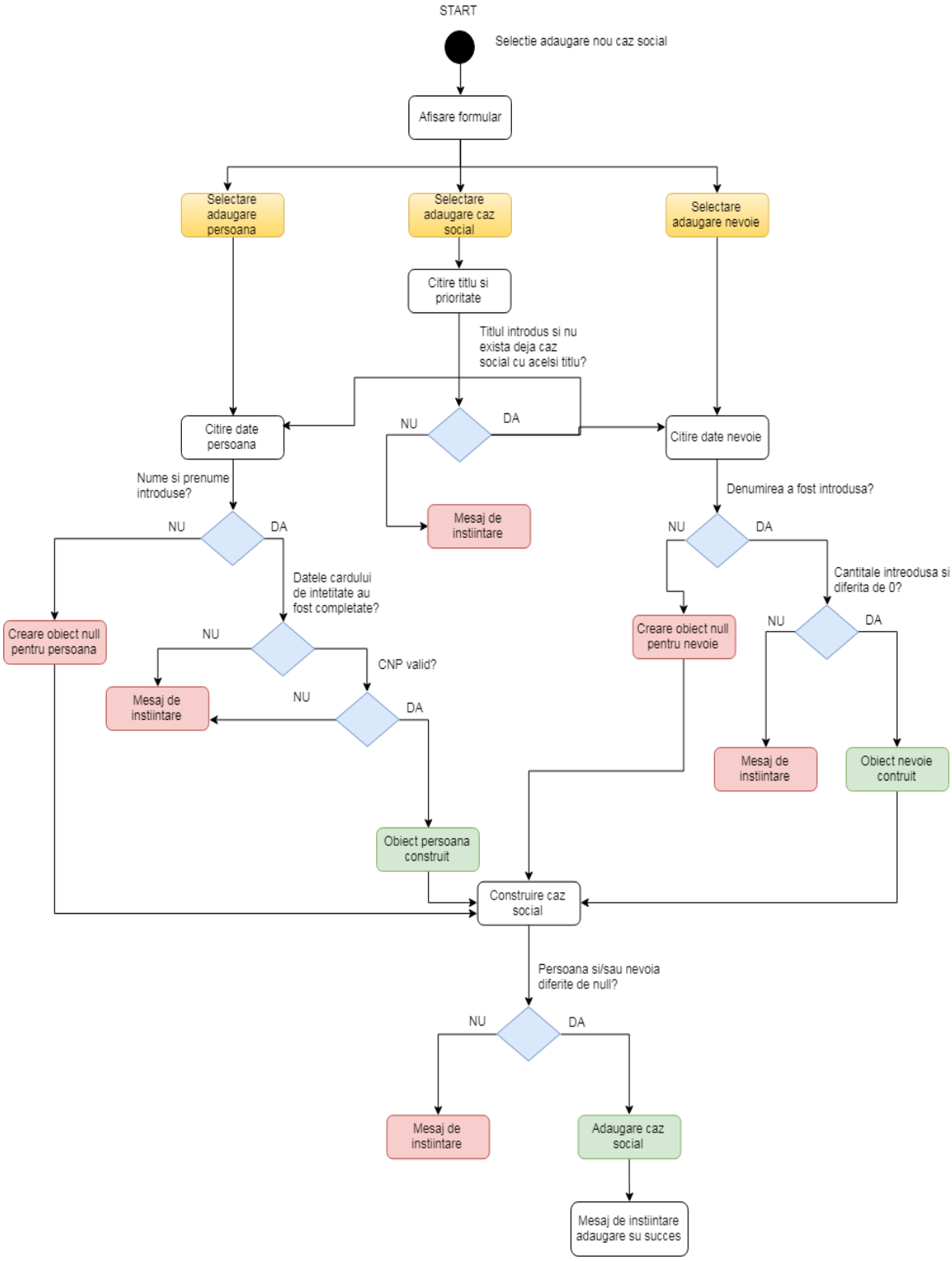



Figura 5. 6 Organigramă pentru funcționalitatea de adăugare caz social

### *5.5.2. Funcționalitatea de mapare a stocurilor pe nevoi*

Al doilea algoritm menționat caută stocuri disponibile pentru o anumită nevoie. În momentul în care aceasta este selectată dintr-un caz social, se afișează o sugestie de stoc disponibil, gata de a fi transferat pentru împlinirea nevoii selectate. Pașii urmați pentru acest algoritm sunt:

- Identificarea nevoii
- Căutarea stocurilor de produse care corespund produsului pentru care este nevoia
- Dacă mai mult de un stoc este găsit, se alege cel cu data de exprirare cea mai aproape de momentul prezent
- Se returnează datele stocului selectat
- Se afișează într-un pop-up aceste date
- Utilizatorul are posibilitatea de stabilire automată a unei întâliri de transfer de stoc
    - Datele stocului sunt transferate spre pagina calendarului utilizatorului
    - Aici va apărea un nou pop-up cu detaliile transferului (date despre beneficiar, stoc)
    - Utilizatorul poate alege să stabilească întâlnirea în ziua curentă și atunci tot ce va mai trebui să facă este să aleagă ora pentru întâlnire
    - Dacă se dorește stabilirea întâlnirii într-o altă zi, se revine pe calendar, de unde utilizatorul trebuie să selecteze o zi anume, apoi să alegă ora, ca și în cazul menționat anterior

Pentru ilustrarea fluxului urmat pentru acest algoritm, atât pe partea aplicației client cât și pe cea a aplicației server, s-au construit două diagrame de secvență (figura 5.7 și figura 5.8). Figura 5.9 este reprezentativă pentru aplicația client și ilustrează apelurile efectuate pentru calculare și afișarea unui stoc pentru o nevoie. Următorii pași pentru programarea unui transfer urmează în principiu același flux: din clasa *CalendarActivity* se propagă apelul pentru adăugarea unei programări prin aceleși tipuri de clase, dar specifice pentru ședințe (*MeetingViewModel*, *MeetingRepository*, *MeetingService*). Cele două diagrame sunt, de fapt, în legătură, deoarece din interiorul interfeței *StockService* se transmite apelul de metodă către *StockController* pe partea de server, de unde se continuă fluxul. La final, rezultatul este trimis la client.

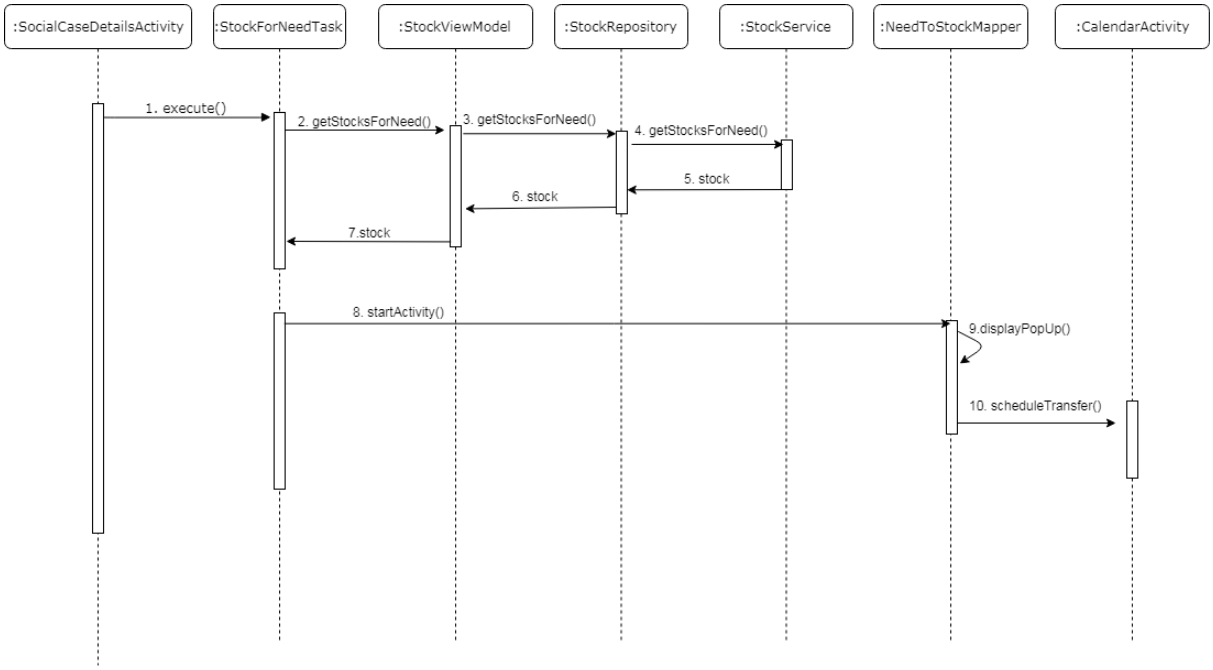



Figura 5. 7 Diagrama de secvență pentru calcul stoc - client

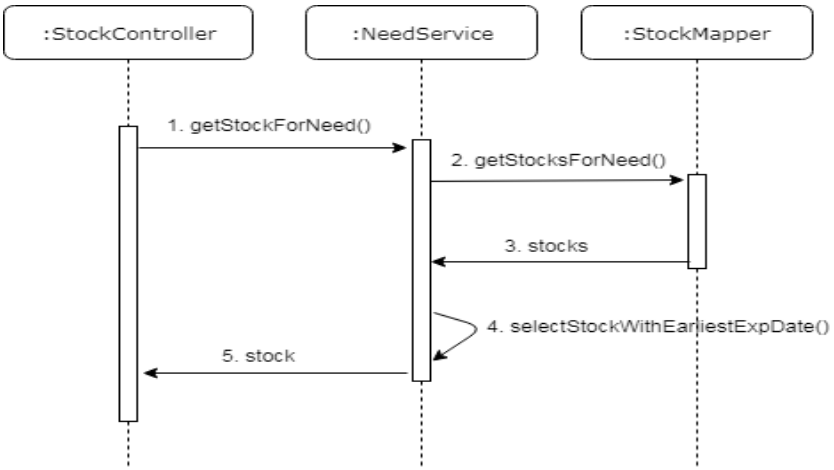



Figura 5. 8 Diagrama de secvență pentru calcul stoc - server

## 5.6. Diagrama de deployment

În figura 5.9 este prezentată diagrama de deployment a sistemului, cu cele trei componente care îl alcătuiesc: aplicația android ca și client, care este intalată pe un smartphone, aplicația server și baza de date. Aplicațiile server și client comunică între ele prin protocolul *HTTP*, prin transfer de mesaje. Cea dintâi este lansată ca și un arhivă Java (*server-0.0.x.SNAPSHOT.jar,* unde ”x” reprezintă numărul versiunii lansate), pentru a putea rula pe un server. Serverul comunică cu baza de date, conexiunea fiind realizată cu prin intermediul unui *driver JDBC*, comunicarea fiind suportată de către protocolul *TCP/IP*.

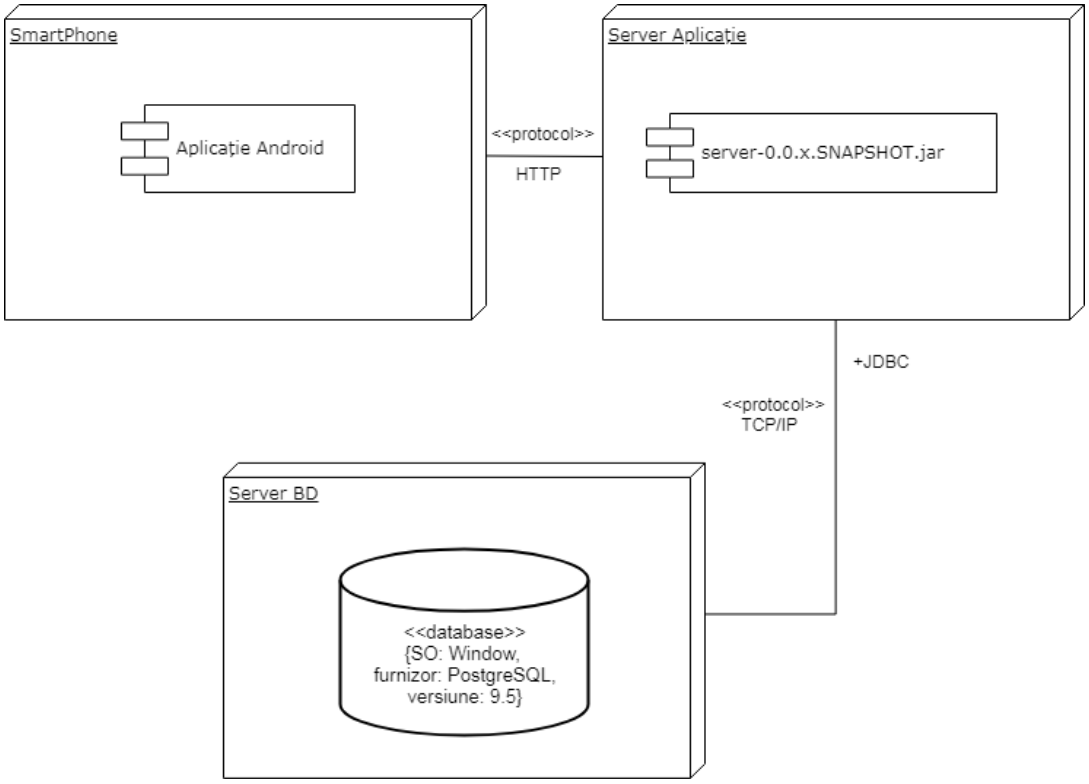



Figura 5. 9 Diagrama de deployment

# Capitolul 6. Testare şi Validare

În acest capitol sunt menționate tipurile de testare pentru un sistem, iar apoi se vor prezenta metodele de testare și de validare a sistemului, în primul rând al server-ului, iar apoi a aplicației client. Deoarece funcționalitățile ce țin de comunicarea cu baza de date au fost implementat exclusiv pe server, testarea acestora a fost concetrată pe această componentă, iar pentru componenta client s-au efectuat teste pentru testarea și validarea acțiunilor până la apelul către API-urile expuse de server.

## 6.1. Tipuri de testare

Așa este prezentat cum și în cartea [11] pentru dezvoltarea sistemelor informatice, testarea poate fi de două tipuri:

1. Testare statică: folosită pentru identificarea și corectarea erorilor umane, cum ar fi în producerea de documente, scrierea de cod sursă sau crearea listelor de date. Aceasta implică revizuirea specificației cerințelor sistemului, specificația funcțională, specificația de proiectare și cea de program (fiecare se efectuează în funcție de cea dinaintea ei).
2. Testare dinamică: folosită pentru verificarea calității software-ului livrat și este efectuată prin rularea unor teste folosind date te test. Aceasta se împarte în următoarele tipuri de testare:
    a. Testare unitară: se testează unități/componente individuale de software, unde o *unitate* este cea mai mică parte de software care poate fi testată.
    b. Testare de integrare: folosită pentru a asigura o comunicare prin trasnfer de date corectă între două module. Mai multe unități sunt grupate și testate împreună. Scopul acestui tip de testare este de a descoperi defectele în ceea ce privește comunicarea între unități integrate.
    c. Testare de sistem: se testează sistemul ca un întreg și are ca și scop evaluarea conformării sistemului cu specificația cerintelor.

## 6.2. Testarea aplicației server

Pentru testarea aplicației server s-au considerat relevate testarea unitară și cea de integrare. Astfel, pentru funcționalitățile din servicii s-au construit cazuri de test, împreună cu datele te test asociate, pentru a asigura un comportament corect.

Pentru testarea unitară s-a folosit framework-ul Mockito[34], care oferă suport pentru crearea de teste simple și clare, care produc erori de verificare curate și rulează automat. Într-o clasă de test avem declarată clasa pe care o testăm (care este de fapt implementarea unui serviciu) ca și variabilă instanță anotată cu @InjectMocks. Această anotare identifică pentru Mockito clasa care trebuie testată și în care trebuie să injecteze mock-uri. Obiectele *mock* sunt implementări dummy pentru o interfață sau o clasă în care se pot defini rezultatele pentru diferite apeluri de metode. Mai apoi, cu @Mock avem anotate clasele care trebuie injectate. Această anotare creează o implementare mock pentru clasele respective. Ca și exemplu de teste care au fost implementate, se poate consulta figura 6.1,

[34] https://site.mockito.org/

care reprezintă testele rulate pentru funcționalități ce țin de utilizator. După cum se poate observa, rularea a 12 teste a durat 1.818 secunde, ceea ce demostrează eficiența testării automate.

În plus, pentru a verifica endpoint-urile pentru API-urile expuse către client și status-urile așteptate s-au folosit teste unitare cu MockMvc, iar pentru a le putea testa funcționalitaea s-a folosit *Postman*[35], care este un mediu de dezvoltare folosit pentru testare de API-uri, prin intermediul căruia se pot introduce toți parametrii necesari apelarii și verifica răspunsurile, în diferite formate (de exemplu: JOSN, XML ș.a). Un exemplu de apel API prin intermediul uneltei *Postman* este oferit în figura 6.2.

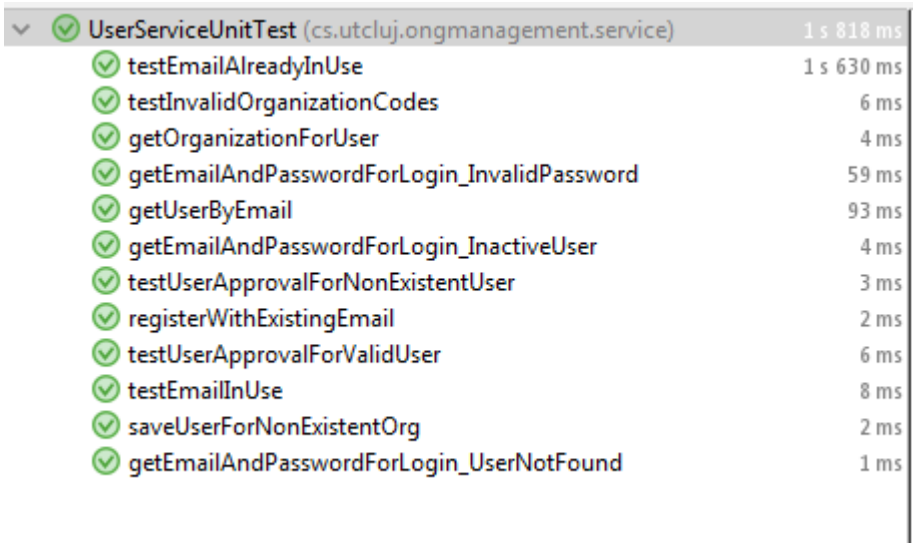



Figura 6. 1 Rezultatul rulării testelor automate

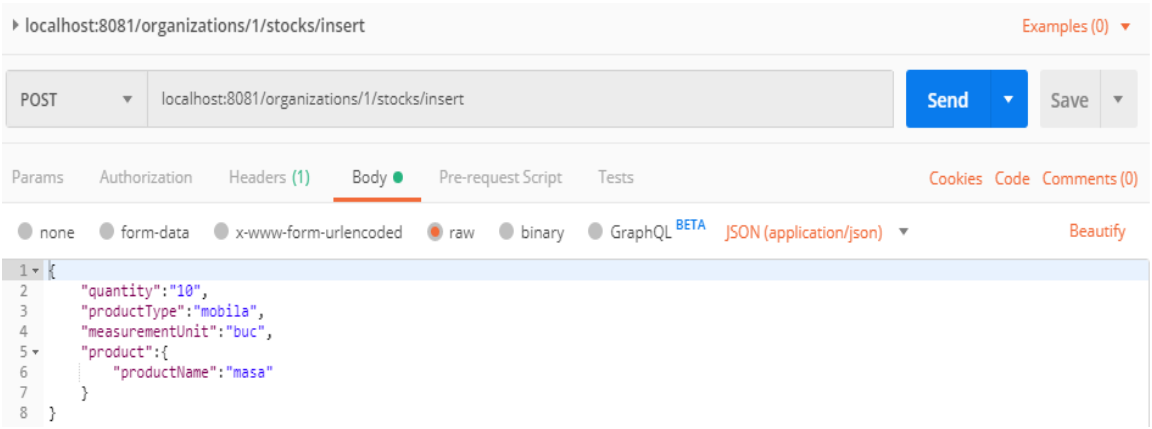



Figura 6. 2 Apel API din Postman

[35] https://www.getpostman.com/

Pentru testarea de integrare s-a folosit clasa MockMvc care face parte din *Spring MVC Test Framework* . Specific testelor de integrare este rularea automată a unor interogări SQL furnizate prin intermediul anotărilor @Sql, acestea putând să fie executate atât înaintea rulării testului (în cazurile în care vrem să adăugăm intrări în baza de date, de exemplu), cât și după (când vrem să eliminăm din baza de date intrările adăugate anterior).

## 6.3. Testarea aplicației client

Ținând cont de faptul că implementarea funcționalităților principale ce țin de comunicarea cu baza de date se află pe server, pentru aplicația client s-a optat testarea fluxului aplicației și a comportamentului. În lucrarea [12], care este un tutorial petru testarea aplicațiilor mobile, s-au identificat patru abordări de testare și acestea sunt:

- Testarea bazată pe emulator: implică folosirea diferitor dispozitive mobile virtuale
- Testare pe diferite dispozitive mobile fizice
- Testare Cloud
- Testare efectuată de un grup de utilizatori sau ingineri de testare

### *6.3.1. Testare manuală*

Testarea manuală de tip black-box a fost efectuată pe dispozitivul mobil personal și pe emulatoare, ceea ce înseamnă că nu s-a ținut cont de funcționalitatea internă a aplicației, ci s-au verificat doar rezultate obținute pentru datele de intrare furnizate. În acest sens, s-au creat mai multe scenarii de test, pentru funcționalitățile aplicației. În continuare se va prezenta scenariul de testare pentru cazul în care utilizatorul și-a uitat parola de la cont.

**Scenariu de test**:
**Denumire:** Utilizatorul și-a uitat parola și vrea să-i fie resetată
**Scenariul 1:**

1. Utilizatorul intră în aplicație și dacă este deconectat îi apare pagina de login
2. Utilizatorul apasă pe text-ul ”Mi-am uitat parola”
3. Pagina pentru resetarea parolei se deschide. Aceasta conține instrucțiunile necesare
4. Utilizatorul își introduce adresa de E-mail
5. Utilizatorul apasă butonul ”CONFIRM”
6. Adresa de E-mail nu este găsită în baza de date, iar utilizatorul este notificat de acest lucru

**Scenariul 2:**

1. Utilizatorul intră în aplicație și dacă este deconectat îi apare pagina de login
2. Utilizatorul apasă pe text-ul ”Mi-am uitat parola”
3. Pagina pentru resetarea parolei se deschide. Aceasta conține instrucțiunile necesare
4. Utilizatorul își introduce adresa de E-mail

5. Adresa de E-mail este găsită în baza de date. Se generează o parolă aleatorie, care mai apoi este trimisă pe mail.
6. Utilizatorul este notificat că a primit un mail nu noua parolă și cu intstrucțiunile necesare.

### *6.3.2. Testare automată*

Pentru automatizarea testelor manuale prezentate mai sus s-a folosit *Espresso Test Recorder*[36], care este o unealtă ce permite crearea de teste UI pentru aplicația Android, fără a fi nevoie de a scrie cod de test. Pentru proiectarea unui scenariu de test se pot înregistra interacțiunile cu un dispozitiv atât virtual cât și fizic, iar mai apoi *Espresso Test Recorder* va prelua înregistrarea și va genera automat un test UI care apoi poate fi rulat. În figura 6.3 se poate vedea o înregistrare în curs de desfășurare pentru activitatea de autentificare.

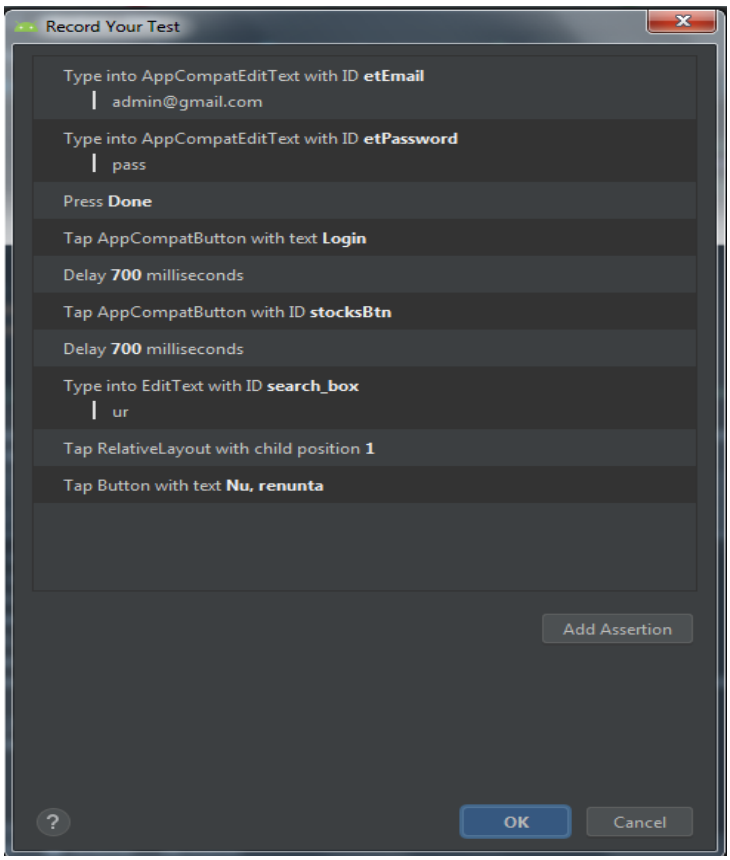



Figura 6. 3 Înregistrare cu Espresso Test Recorder

În acest capitol au fost prezentate modalitățile de testare a aplicației folosit, atât pe partea de server cât și pe partea de client, cu care s-au validat funcționalitățile sistemului.

[36] https://developer.android.com/studio/test/espresso-test-recorder

# Capitolul 7. Manual de Instalare și Utilizare

În cadrul acestui capitol se vor detalia resursele software și hardware necesare pentru instalarea și rularea aplicației, cât și o descriere clară, pas cu pas a modului de instalare și utilizare a aplicației.

## 7.1. Resurse necesare instalării

Pentru ca aplicația să poată fi instalată trebuie să se asigure resursele necesare, atât software cât și hardware. Se vor detalia resursele necesare pentru cele două componente, server și client, separat.

Pentru componenta server sunt necesare următoarele resurse hardware și software:

- Calculator, laptop sau mașină virtuală pe care să ruleze aplicația
- Memorie RAM de minim 4GB
- Frecvență procesor de cel putin 2.40GHz
- Sistem de operare: Windows / Linux
- Programele necesare instalate: PostgreSQL 9.5, pgJDBC, RabbitMQServer (pentru serviciul de mail)

Pe partea de client, pentru a putea instala și utiliza aplicația este nevoie de un dispozitiv mobil care să dispună de următoarele resurse software:

- Sistem de operare Android
- Versiune Android: 9.0 (Pie). Compatibilitatea cu versiuni mai noi este asigurată, dar cea cu versiuni mai vechi nu.
- Conexiune bună la o rețea de Internet (Wi-Fi – de preferat fibră optică, 3G, 4G)
- Cel putin 4.68 MB memorie liberă

Odată ce resursele menționate au fost asigurate, aplicația poate fi instalată, iar pentru a face acest lucru trebuie ca fișierul APK să fie descărcat pe dispozitivul mobil. Pentru rularea aplicației, aceasta se selectează din lista de aplicații de pe telefon (căutați dupa numele ONGManagement) și se deschide prin simpla atingere a iconiței.

## 7.2. Manual de utilizare

Aplicația dezvoltată este  destinată membrilor și beneficiarilor unor ONG-uri care se ocupă cu distribuirea de pachete persoanelor nevoiașe, pentru gestionarea activităților de zi cu zi.  Acest manual are ca scop prezentarea modului de utilizare a acestei aplcații.

### *7.2.1. Înregistrare și autentificare*

Prima pagina care se va deschide este cea în care utilizatorul se poate înregistra și autentifica în aplicație (Figura 7.1). Pentru înregistrare trebuie introduse numele și prenumele, o adresă de E-mail validă, o parolă, tipul de utilizator și organizația de care

acesta aparține. De asemenea, pentru a putea crea un cont, utilizatorul trebuie sa fie de acord cu procesarea datelor personale în scopul oferirii serviciilor aplicației (acordul acesta se oferă prin bifarea casuței, la fel cum se poate observa in figura 7.1- dreapta). Pentru autentificare este nevoie de credențiale valide, iar dacă acestea sunt furnizate se va deschide pagina Acasă (Figura 7.2). La următoare pornire a aplicației, dacă utilizatorul s-a autentificat în trecut, aceasta va fi prima pagină deschisă. Tot în cadrul paginii de autentificare se poate selecta și ”Mi-am uitat parola”, care va deschide o nouă fereastră ce conține instrucțiunile pentru resetarea parolei și un câmp pentru introducerea adresei de E-mail.

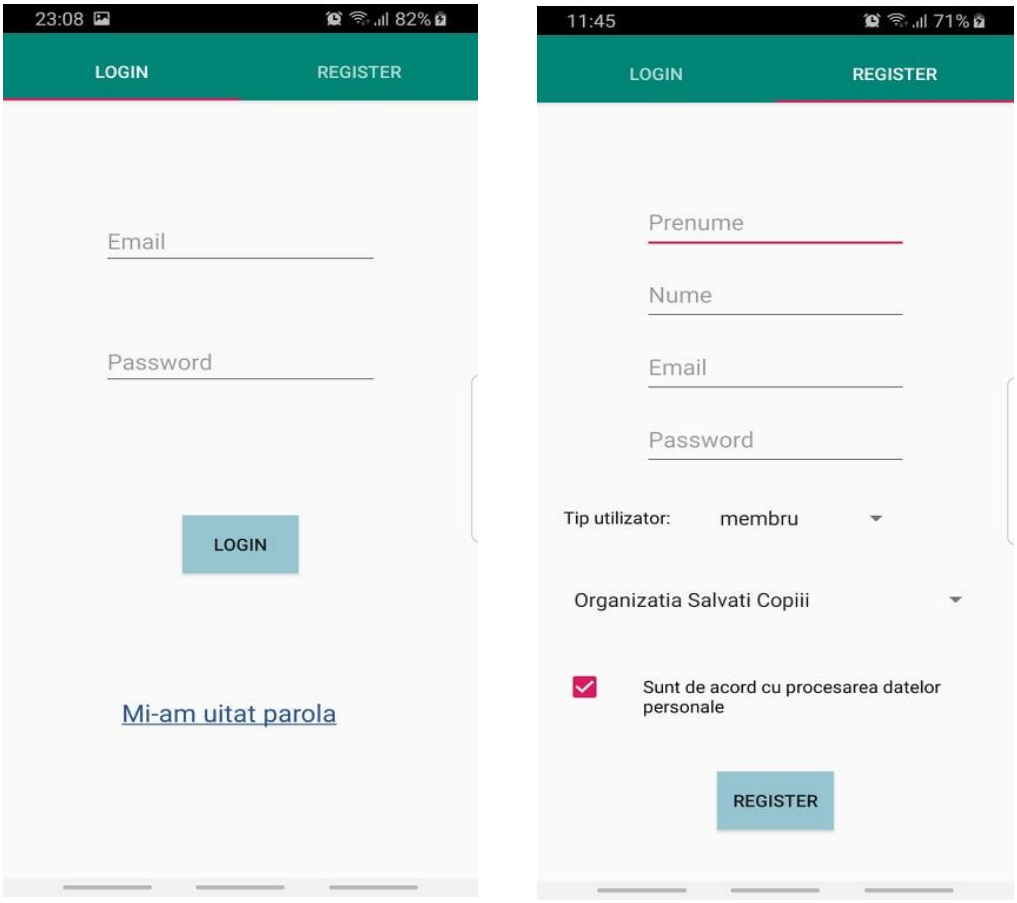



Figura 7. 1 Pagina pentru autentificare și înregistrare

## *7.2.2. Pagina Acasă*

Această pagină (Figura 7.2) este dispobiliă pentru utilizatorii care sunt membrii ai unei asociație și administratorilor. Conține cele patru elemente: stocuri, cazuri sociale, calendar și rapoarte, care reprezintă funcționalitățile de bază ale aplicației. Prin apăsarea oricărei dintre cele patru iconițe se vor deschide paginile corespunzătoare, care vor fi prezentate în subcapitolele următoare. În dreapta sus este dispobilil un meniu cu cele două

elemente reprezentate prin avatarul de utilizator și cele trei puncte. Avatarul are două funcționalități, în funcție de tipul de utilizator:

- Pentru utilizator membru, apăsarea iconiței va deschide un pop-up cu informații generale despre contul acestuia, cum ar fi: detalii despre organizația de care aparține, permisiunea de accesare a aplicației
- Pentru administrator, apăsarea iconiței va deschide o pagină cu toți utilizatorii în așteptare, pe care aceștia trebuie să-i aprobe și să le asigneze permisiuni (Figura 7.3).

Prin apăsarea celor trei puncte din partea dreapta-sus a paginii Acasă se deschide o listă cu cele două funcționalități disponibile: schimbarea parolei și deconectare. Selectarea schimbării parolei va deschide o pagină în care trebuie introdusă parola curentă, apoi cea de două ori, pentru confirmare. La final se apasă butonul ”CONFIRM” și o notificare apare, care sugerează o deconectare și o autentificare nouă.

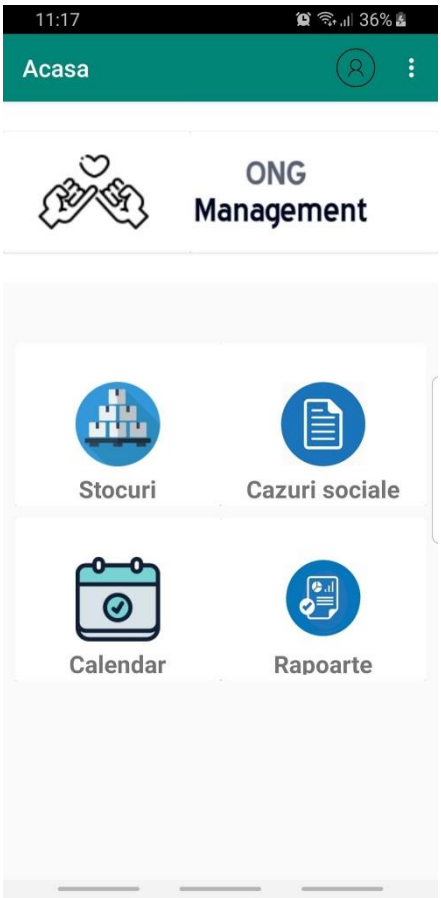



Figura 7. 2 Pagina Acasă


Figura 7. 3 Pagină aprobare utilizatori

### *7.2.3. Stocuri*

Pagina pentru stocuri afișează o listă cu toate stocurile pe care le are organizația de care utilizatorul aparține (Figura 7.4). Deoarece această listă poate fi destul de mare, există opțiunea de a folosi câmpul de căutare pentru a găsi un stoc în funcție de numele acestuia. Un stoc din listă se poate șterge (adică este setată cantitatea pe 0 și apoi e eliminat din listă), apăsând rândul corespunzător. Va apărea un mesaj de confirmare în urma căruia acțiunea poate fi finaliaztă sau se poate renunța la ea. Bineînțeles, se poate și adăuga un stoc nou prin apăsarea iconiței albastre în formă de plus. Se va deschide o pagină (Figura 7.5) în care se pot completa detele necesare, apoi se va apăsa butonul albastru pentru salvare din dreapta sus și un mesaj de informare va apărea pe ecran.

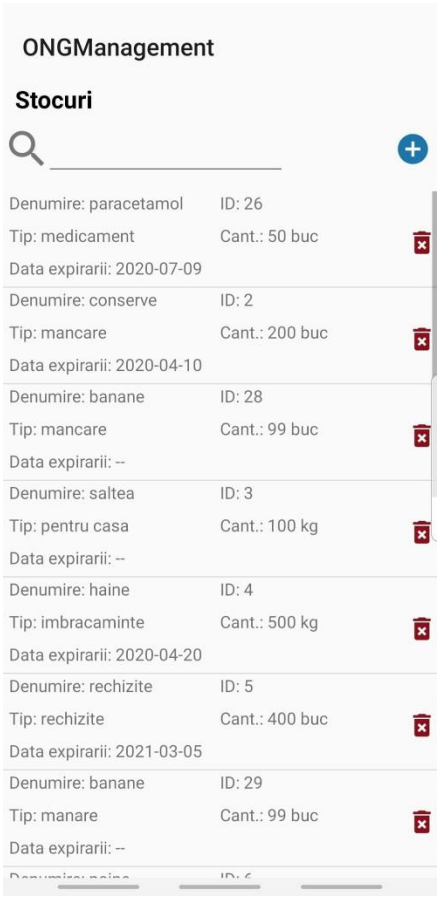



Figura 7. 4 Pagină adăugare stoc


Figura 7. 5 Pagină stocuri

### *7.2.4. Cazuri Sociale*

Din pagina de Acasă, prin apăsarea iconiței pentru cazuri sociale se va deschide o listă cu toate cazurile sociale care aparțin de organizație, sortate implicit în funcție de prioritate, în ordine descrescătoare (Figura 7.6). În listă sunt afișate doar câteva detalii generale despre cazurile sociale (titlu, prioritate). Căutarea folosind câmpul de căutare se

face în funcție de titlul cazului social, care ar trebui să conțină numele titularului. Pentru vizualizarea detaliilor complete ale unui caz social, trebuie apăsat pe elementul corespunzător din listă și astfel se va deschide o nouă pagină, care afișează toate persoanele și nevoile care apartin cazului social, cu detaliile corespunzătoare (Figura 7.7). Prin selectarea unei persoane (sugerată și de iconița dreptunghiulară în formă de act de indetitate) se va deschide o fereastră cu toate detaliile legate de indentitate și altele informații folositoare depsre persoana respectivă. Prin apăsarea iconiței din stânga-jos sub formă de cos de reciclare, această persoană poate fi eliminată din cazul social, în urma confirmării mesajului care va apărea pe ecran. Prin selectarea unei nevoi se poate obține stocul care este disponibil pentru transfer, dacă este cazul (doar dacă nevoia nu a fost deja împlinită – marcată cu bifă roz și dacă există un astfel de stoc – altfel se va afișa un mesaj de informare). Fereastra care se va deschide este ilustrată în figura 7.8. În continuare, prin apăsarea iconiței în formă de ceas, aplicația va deschide calendarul pentru programarea unui transfer cu stocul sugerat, către nevoia selectată. Procesul creării unei astfel de intrări în calendarul personal va fi prezentat în următorul sub-capitol.

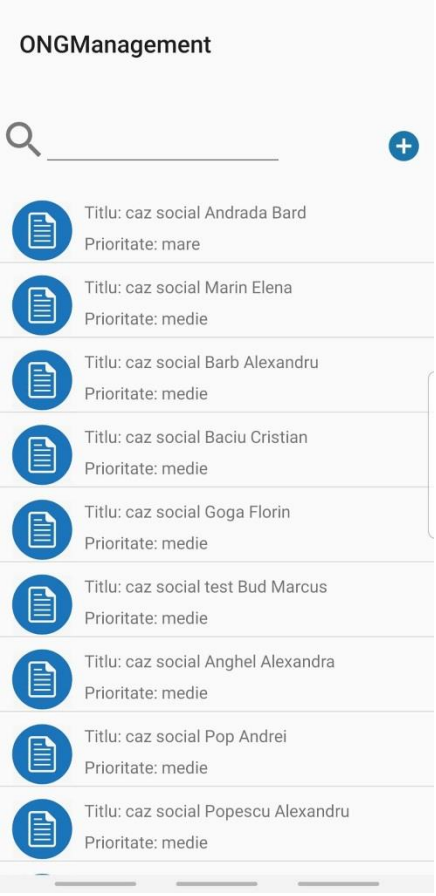



Figura 7. 6 Pagină cazuri sociale


Figura 7. 7 Pagină detalii caz social

Alte opțiuni pentru cazurile sociale sunt:

- Adăugare de nou caz social (prin apăsarea iconiței albastre în formă de plus din figura 7.6)- se va deschide pagina cu detaliile care trebuie completate (Figura 7.8). Această funcționalitate a fost explicată în secțiunea 5.5.1.
- Dezactivare caz social, prin apăsarea butonului roșu din dreapta sus (simbolul de interzis). Va apărea un mesaj de confirmare, prin care se poate completa sau aborta acțiunea. Dezactivarea cazului social este, de fapt, eliminarea acestuia din aplicație, deoarece sunt prezentate doar cazurile active, dar el va rămâne în baza de date.

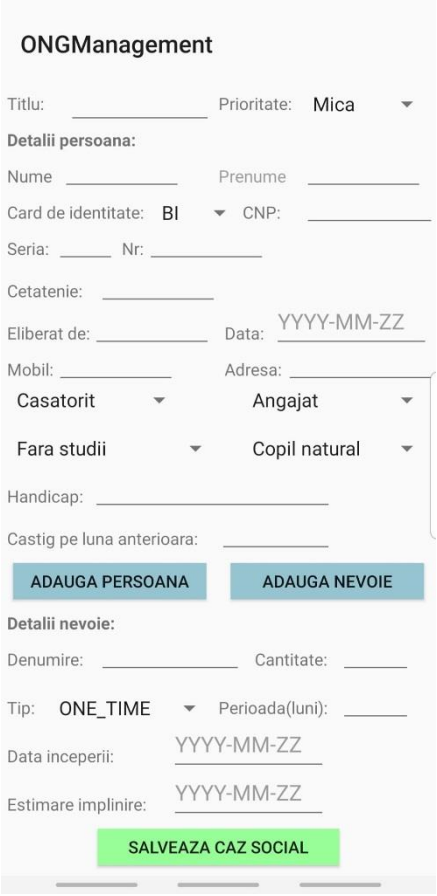



Figura 7. 8 Formular adăugare caz social

### *7.2.5. Calendar personal*

Această pagină (Figura 7.9) conține un calendar, în care se pot programa atât ședințe interne (care conțin doar o descriere, adresa și ora la care are loc ședința), cât și transferuri de stocuri către cazuri sociale (automat, din fereastra în care se sugerează un stoc pentru o nevoie, menționată în sub-capitolul anterior). Sunt două modalități pentru deschiderea paginii:

- Direct din pagina Acasă: ca rezultat, se afișează calendarul și o previzualizare a ședințelor din ziua curentă, sau o notificare ce informă utilizatorul că nu are nimic programat
- Din fereastra cu stoc pentru nevoie: în pagina pentru calendar se va deschide instant un mesaj pentru programarea transferului (Figura 7.10), care are opțiuni pentru programare pe ziua curentă sau alegerea unei alte zile. În primul caz, singurul pas rămas este selectarea orei prin intermediul ceasului care va apărea pe ecran (Figura 7.11) iar la final programarea adăugată va apărea la previzualizare. În cel de-al doilea caz, aplicația se întoarce la calendar și utilizatorul trebuie să selecteze ziua dorită, apoi se va afișa ceasul pentru alegerea orei dorite.


Figura 7. 9 Pagină Calendar

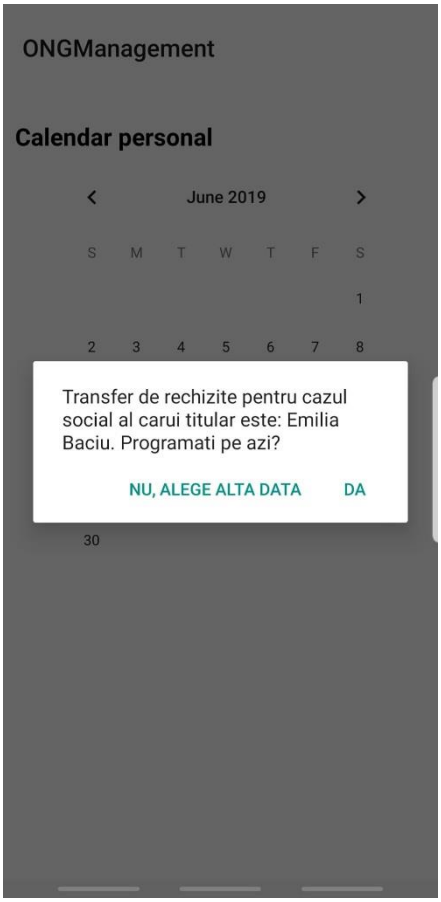



Figura 7. 10 Mesaj pentru salvare transfer

Prin selectarea oricărei zile din calendar, se va deschide o nouă pagină cu ședințele programate pentru ziua respectivă și un mini-formular de completat pentru introducerea unei noi ședințe (finalizată prin apăsarea butonului de plus). Această pagină este prezentată în figura 7.12 .

Pentru schimbarea statusului unei programări se selectează un element din lista drodown, fiecare acțiune generând un mesaj corespunzător de confirmare. În cazul transferurilor, în funcție de schimbarea de status se vor actualiza și stocurile. De menționat este faptul că pentru ședințele internte sunt disponibile doar două status-uri: *În așteptare* și *Efectuată*.

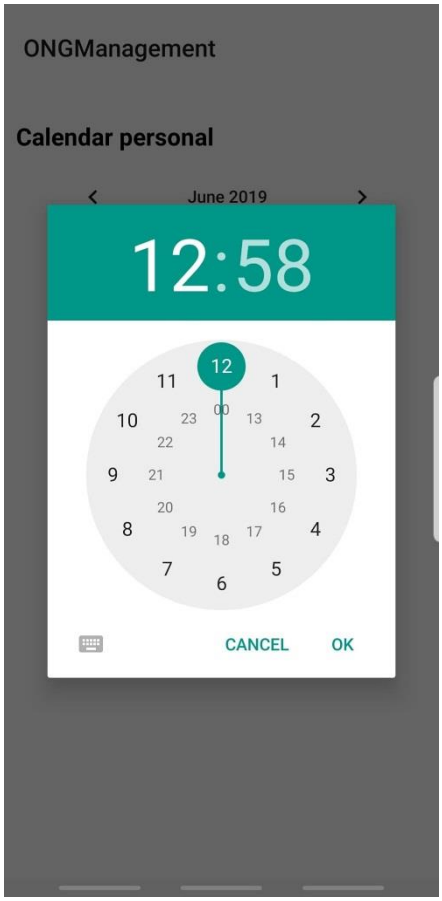



Figura 7. 11 Fereastră pentru selectarea orei

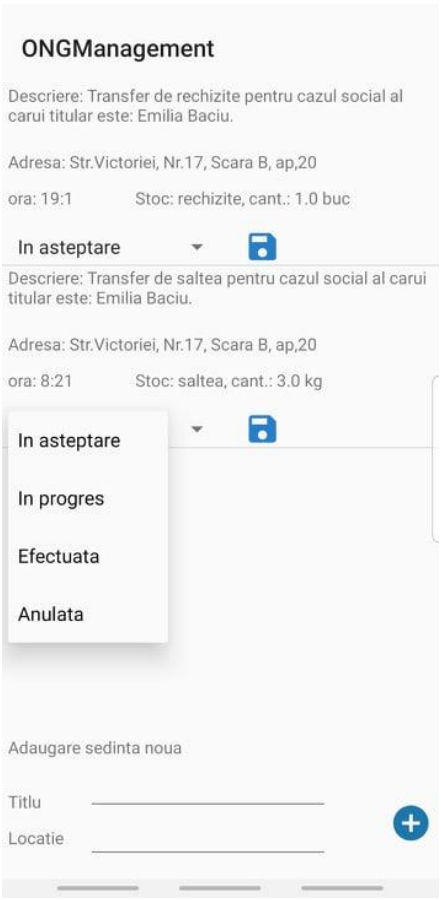



Figura 7. 12 Pagină ședințe

## *7.2.6. Pagină rapoarte*

Pagina de rapoarte (Figura 7.12) se deschide prin apăsarea iconiței corespunzătoare din pagina Acasă. Aceasta are ca scop obținerea rapidă a datelor ce țin de stocurile unei organizații și reprezentarea lor vizuală, pentru a putea fi citite ușor. Este disponibilă

furnizarea a trei seturi de date și reprezentarea lor sub forma unei diagrame radiale (PieChart) și acestea sunt:

- Stocurile care au fost transferate în ziua curentă către cazuri sociale
- Stocuri cu o cantitate mai mică decât o valoare introdusă de utilizator (în câmpul de lângă butonul ”GENEREAZĂ”)
- Stocuri care expiră în decurs de o lună

Dacă setul de date calculat conține cel puțin o valoare, se va deschide o nouă pagină care conține o diagramă, ca cea din figura 7.13). Acesta dinstribuie stocurile în funcție de cantități și le reprezintă cu diferite culori.


Figura 7. 13 Pagina de rapoarte

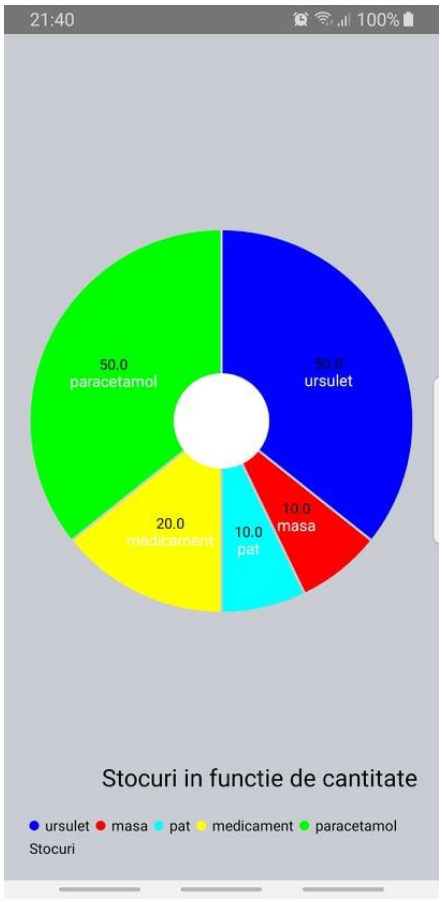



Figura 7. 14 Diagramă Radială

## *7.2.7. Pagina pentru beneficiar*

Al treilea tip de utilizator al aplicației dezvoltate este beneficiarul, care are un caz social la o anumită asociație. Motivul pentru care a fost introdus și acest tip este pentru posibilitatea de a vizualiza în format electronic datele despre cazul social care îi aparține, aceleași detalii pe care le deține și organizația, și de a putea să adauge noi nevoi. Această pagină (Figura 7.14) se deschide imediat după autentificare. Aici se pot completa detaliile

pentru adăugarea unei nevoi noi, acțiune finalizată cu apăsarea butonul ”Adaugă nevoie”. Pentru vizualizarea informațiilor conține în cazul social, se apasă butonul ”Vizualizare caz social” și se va deschide o pagină, asemenea celei deschise și pentru ceilalti utilizatori.

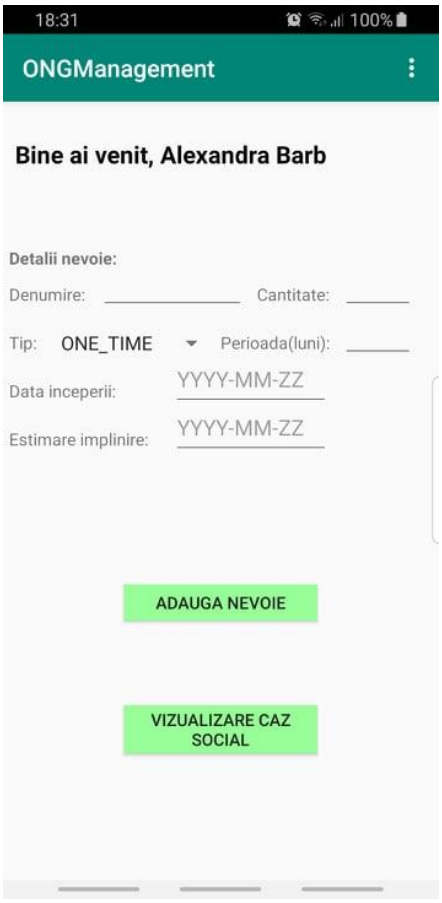



Figura 7. 15 Pagina pentru beneficiar

La finalul acestui capitol, utilizatorul ar trebui să poată instala cu succes aplicația și să o folosească fără probleme.

# Capitolul 8. Concluzii

În acest capitol se va concluziona lucrare prezentă, oferindu-se o viziune de ansamblu asupra a ceea ce s-a realizat pentru această aplicație. Privind dintr-o perspectivă critică aplicația dezvoltată, se vor propune posibile dezvoltări și îmbunătățiri ulterioare.

## 8.1. Contribuție personală și Realizări

Ideea inițială de o platformă digitală care să vină în ajutorul ONG-urilor din România în primul rând a apărut în cadrul proiectului de practică la care am participat, venind ca și cerere din partea unei astfel de organizații. Pornind de la această idee am dezvoltat o aplicație care să fie potrivită nu doar pentru o singură organizație, ci pentru mai multe care întreprind aceleași tipuri de acțiuni sociale. Astfel, s-a dezvoltat aplicația mobilă, considerându-se mai accesibilă decât un site web. Pe partea se server s-au îmbunătățit funcționalitățile și s-au adăugat altele noi, printre care se numără:

- Diferențierea tipurilor de utilizatori
- Oferirea de funcționalități specifice tipului de utilizator (Admin și beneficiar)
- Furnizare sugestie de stoc pentru nevoie și facilitatrea programării transferului ș.a
- Generarea rapoartelor
- Securitatea

Pe partea de client s-au dezvoltat următoarele funcționalități:

- Înregistrare
- Aprobare utilizatori de către administrator și asignare drepturi de acces la aplicație
- Autentificarea utilizatorilor
- Gestionarea cazurilor sociale (cu persoanele și nevoile asociate)
- Gestionarea stocurilor
- Calendar persoanal – care include programarea unor ședințe și editarea lor (schimbarea status-ului, reprogramare)
- Diferențierea tipurilor de acțiuni specifice pentru tipul de utilizator
- Generarea de rapoarte pentru monitorizarea stocurilor
- Funcționalități pentru uitarea parolei și schimbarea acesteia
- Pagină pentru beneficiar cu funcționalități disponibile de: vizualizarea datelor cazului social personal și adăugarea unor nevoi noi

Prin intermediul aplicației care a fost dezvolată s-a dorit venirea în ajutorul organizațiilor non-profit, cu o soluție de digitalizare a datelor. S-a dorit o aplicație ușor de folosit de către persoanele non-tehnice, disponibilă oriunde aceștia își desfășoară activitatea, pe dispozitivul mobil propriu. S-a pus accent mai mult pe funcționalitățile oferite, de procesare și gestionare a datelor. Un lucru important este ca doar membrii unor organizației sau beneficiarii organizației să poată folosi aplicația, deoarece datele

gestionate trebuie să fie private, motiv pentru care s-a ales nu doar accesul exclusiv prin autentificare cu un cont valid, ci și necesitatea ca un utilizator să fie aprobat de către administratorul aplicației, pentru activarea contului. În ceea ce privește cerințele funcționale, s-a urmărit oferirea unor funcționalități care să faciliteze prelucrarea rapidă și ușoară a informațiilor. Principalele date care devin greu de gestionat fără un sistem de managemet sunt cele care țin de stocuri și de cazuri sociale. De aceea, funcționalitățile dezvoltate vin în ajutorul membrilor asociației, prin posibilitatile de căutare și prelucrarea a acestor date. De asemenea, pentru stocuri vin în ajutor și diagramele radiale care se pot genera pentru urmărirea acestora.

S-a menționat dorința de a minimiza cazurile de fraudă, pentru ca fiecare persoană nevoiașă să aiba aceleași șanse, de aceea aplicația nu permite să se programeze transferuri de stocuri către anumite persoane, decât în mod automat, prin funcția de sugestie a celui mai potrivit produs, care vine și în ajutorul raționării stocurilor. Tot din aceeași dorință s-a restricționat adăugarea unei persoane la un caz social sau crearea unui caz social nou pentru o persoană care deja există în baza de date, cât și crearea de cont nou în aplicație cu o adresă de e-mail care este deja utilizată.

## 8.2. Dezvoltări și îmbunătățiri ulterioare

Cum desigur este loc de îmbunătățiri ale funcționalităților curente precum și de altele noi, se vor prezenta câteva idei care se pot lua în considerare pentru dezvoltare ulterioară și acestea sunt:

- Integrarea aplicației cu Google Maps pentru obținerea distanței între sediul organizației și reședința persoanelor beneficiare, pentru sortarea cazurilor în funție de distanță
- Introducerea notificărilor de reamintire în afara aplicației, pentru ședințele programate sau pentru stocurile pe care să expire sau să fie epuizate
- Posibilitate de gestionare donatori: se poate implementa o pagină separată cu informațiile despre donatorii unei organizații și modalități de interacționare cu aceștia (de exemplu: trimitere de mail-uri, efectuare apeluri)
- Funcționlitate care să ofere sugestii de cazuri sociale cu nevoi care se potrivesc cu un anumit stoc disponibil, în ideea de a facilita mai mult programările de transferuri și distribuirea eficientă de produse
- Posibilitatea înregistrării une organizații prin intermediul aplicației, pentru a o putea folosi. În momentul actual, organizația este adăugată de către administratorul aplicației
- Îmbunătățirea securității, fie prin altă abordare sau altă modalitate de encripție, fie prin generarea aleatorie a cheii secrete de fiecare dată când aceasta este folosită pentru criptare.

În concluzie, cu această aplicație se oferă o soluție validă și funcțională pentru digitalizarea datelor ONG-urilor caritabile, ea fiind însă doar începutul unei aplicații care poate deveni mai mult și mai bună, ușurând munca fiecărui voluntar.

# Bibliografie


[1] A. Vakil, „Confronting the Classification Problem: Toward a Taxonomy of NGOs”, *World Development,* vol. 25, nr. 12, pp. 2057-2070, 1997.

[2] L. M. Salamon și H. K. Anheier, „The International Classification of Nonprofit Organizations” 1996, Available: https://asauk.org.uk/wp-content/uploads/2018/02/CNP_WP19_1996.pdf

[3] K. C. Laudon și J. P. Laudon, „Management Information Systems: Managing the Digital Firm”, Prentice Hall, 2009, Available: http://dinus.ac.id/repository/docs/ajar/MIS_KC_Laudon.pdf

[4] „Ghid de bune practici pentru organizatii neguvernamentale” Opportunity Associates Romania , 2017, Available: http://www.coddeconduitaong.ro/resurse/Ghid_de_Bune_Practici_ONG.pdf

[5] C. Walls, „Spring Boot In Action”, Shelter Island, NY: Manning Publications Co. , 2016.

[6] S. Juba, A. Vannahme și A. Volkov, „Learning PostgreSQL„, Birmingham, UK: Packt Publishing, 2015.

[7] PostgreSQL 9.5.18 Documentation, The PostgreSQL Global Development Group, 1996-2019.

[8] I. Sommerville, „Software Engineering Tenth Edition”, Pearson Education Limited , 2016.

[9] N. Smyth, „Android Studio 3.4 Development Essentials - Java Edition”, Payload Media, 2019.

[10] „Android Developer Guides”, Available: http://developer.android.com/guide

[11] T. Ahmed, J. Cox și L.Girvan, „Developing Information Systems: Practical Guidance for It Professionals”, BCS Learning & Development Ltd, 2014

[12] J. Gao, X. Bai, W.-T. Tsai și T. Uehara, „Mobile Application Testing: A Tutorial”, IEEE Computer Society, 2014.

# Anexa 1: Lista de figuri

## Anexa 2: Lista de tabele

## Anexa 3. Glosar

| TERMEN | DEFINIȚIE |
|---|---|
| **DTO** | **D**ata **T**ransfer **O**bject, este un obiecte care poartă date între două procese, cel mai des întâlnite în aplicații structurate pe nivele. |
| **REST** | **RE**presentational **S**tate **T**ransfer este un stil arhitectural care suportă comunicarea standard între sisteme informatice pe web. |
| **API** | **A**pplication **P**rogramming **I**nterface reprezintă un set de funcții și proceduri care permit crearea de aplicații care să acceseaze caracteristici sau date ale unui sistem de operarre, a unei aplicații sau a unui serviciu. |
| **GDPR** | **G**eneral **D**ata **P**rotection **R**egulation este o reglementare generală din legea Uniunii Europeane, care impune protecția și confidențialitatea datelor personale fiecărui individ din UE. |
| **RAM** | **R**andom-**A**ccess **M**emory reprezintă cel mai comun tip de memorie, folosită pentru acces rapid. Este folosită pentru stocarea datelor |
| **APK** | **A**ndroid **P**ac**K**age este pachetul folosit în sistemul de operare Android pentru distribuția și intalarea aplicațiilor mobile. |
| **JSON** | **J**ava**S**cript **O**bject **N**otation este o sintaxă pentru stocare și schimb de date, sub forma unor perechi atribut-valoare. |
| **XML** | **E**xtensible **M**arkup **L**anguage este un limbaj de marcare care definește un set de reguli pentru codarea documentelor care poate fi citit atât de oameni cât și de mașini. |
| **HTTP** | **H**ypertext **T**ransfer **P**rotocol este cea mai des utilizată metodă pentru accesarea datelor în Internet, care sunt păstrate pe servere **W**orld **W**ide **W**eb (WWW) |
| **SOAP** | **S**imple **O**bject **A**ccess **P**rotocol este un protocol de mesagerie folosit pentru schimbarea de informații structurate în implementarea de servicii web. |
| **URL** | **U**nifrom **R**esource **L**ocator, cunoscută ca și adresă web, este o referință către o resursă web, căreia îi specifică locația și modalitatea de returnare. |
| **TCP/IP** | **T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol este o set de protocoale de comunicare folosite pentru conectare dispozitivelor dintr-o rețea pe internet. |
| **POP** | **P**ost **O**ffice **P**rotocol este un protocol standard de la nivelul aplicației folosit pentru a prelua E-mail-uri de la un server de mail. |

| | |
|---|---|
| **IMAP** | **I**nternet **M**essage **A**ccess **P**rotocol este un protocol standard folosit de clienții de E-mail pentru a prelua E-mail-urile de la un server, folosind o conexiune TCP/IP. |