UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA:Tehnologia informației

Aplicație pentru raportarea și gestionarea sesizărilor către autorități

Proiect de diplomă

Coordonator științific
Ș.l.dr.ing. Cristian Aflori

Absolvent
Dumitru-Daniel Davidescu

Iași, 2019

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII
LUCRĂRII DE LICENȚĂ

Subsemnatul(a) DAVIDESCU DUMITRU-DANIEL,
legitimat(ă) cu CI seria MZ nr. 389851, CNP 1970820226730
autorul lucrării APLICAȚIE PENTRU RAPORTAREA ȘI GESTIONAREA SESIZĂRILOR CĂTRE AUTORITĂȚI

elaborată în vederea susținerii examenului de finalizare a studiilor de licență organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi" din Iași, sesiunea IULIE a anului universitar 2018-2019, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi" din Iași (Manualul Procedurilor, UTI.POM.02 – Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data Semnătura

_____

Cuprins

Aplicație pentru raportarea și gestionarea sesizărilor către autorități

Dumitru-Daniel Davidescu

Rezumat

Proiectul reprezintă o aplicație Android pentru trimiterea de sesizări către autorități și o aplicație Web pentru administrator. Acestea comunică în permanență cu aplicația server, și va pune la dispoziție diferite funcționalități, pe partea de client. Aplicația este concepută pentru a veni în ajutorul cetățenilor unui oraș, pentru a trimite o sesizare legată de problemele cu care se întâmpină în viața de zi cu zi atunci când merg în oraș. Totodată aplicația este binevenită și pentru autorități, deoarece pot vedea mai ușor toate problemele cu care se confruntă cetățenii.

Principala componentă a proiectului este aplicația client Android, ce poate fi utilizată pe dispozitive ce rulează sistemul de operare Android, versiunea cel puțin 4.0. Aplicația necesită permanent o conexiune la Internet, pentru a putea profita din plin de funcționalitățile acesteia. Aplicația nu utilizează intensiv resurse grafice, astfel că, nu există nicio restricție hardware, în afară de prezența unei camere foto asupra dispozitivului.

După instalarea aplicației Android, utilizatorii vor putea să își creeze propriul cont, să se autentifice, după care să trimită o nouă sesizare, selectând categoria în care se încadrează aceasta, după care se obține adresa de unde este trimisă sesizarea, prin intermediul Google Maps. Pentru ca autoritățile să poată observa mai bine problema, aplicația pune la dispoziție și încărcarea unei imagini, și scrierea unei descrieri. De asemenea utilizatorii își pot vedea sesizările trimise, căutând în funcție de statusul acesteia, dar pot vizualiza și sesizările trimise de către toți utilizatorii aplicației în ultimele 24 de ore. Problemele ce pot fi sesizate fac parte din următoarele categorii: transport public, școli, spitale, câini fără stăpân, poliție, parcuri, iluminat public precum și mobilier urban. Unul din obiectivele aplicație a fost păstrarea unei interfețe cât mai simple și clare, astfel ca utilizatorul să poată trimite cât mai ușor și repede sesizarea.

O altă componentă importantă a proiectului o reprezintă aplicația Web pentru administrator. Principalele funcționalități ale aplicației Web sunt vizualizarea tuturor sesizărilor trimise de către utilizatorii aplicației Android, precum și vizualizarea anumitor statistici legate de numărul de sesizări trimise în funcție de categorie. Totodată administratorul poate schimba statusul sesizării, care, împreună cu o scurtă observație să trimită utilizatorului un email.

Introducere

Gropi pe străzi, clădiri dărăpănate, mașini care blochează trotuarele, deșeuri aruncate în locuri nepermise, copaci tăiați, câini fără stăpân, sunt doar câteva dintre problemele cu care cetățenii unui oraș se confruntă zilnic.

În zilele noastre, oamenii folosesc, din ce în ce mai des, telefoanele inteligente, și de aceea am ales realizarea unei aplicații Android, pentru a putea trimite sesizări către autoritățile orașului. De asemenea, în momentul de față, semnalarea unei nereguli într-un oraș, implică mult timp din partea cetățenilor, întrucât nu există o platformă online pentru acest lucru. Proiectul vine însoțit și de o aplicație web destinată administratorului, pentru a putea vedea mai ușor sesizările trimise de către cetățeni, și de a realiza o comunicare mai ușoară cu aceștia, permițând trimiterea de mesaje prin intermediul email-ului.

Scopul acestei teme este de a veni în ajutorul cetățenilor dar și administrației locale, întrucât doar prin implicare, vom putea avea un oraș mai curat și bine organizat.

Ideea acestui proiect a apărut din nevoia de a găsi o aplicație în care se poate semnala anumite probleme din orașul Iași. În urma cercetării aplicațiilor Android, am constatat că nu există o astfel de aplicație pentru municipiul Iași. Problemele punctuale din cartierele orașului, cum ar fi câini comunitari, resturi menajere sau moloz aruncate la întâmplare, trafic îngreunat din cauza vremii, avarii la apă rece și caldă, vor putea fi transmise direct de pe dispozitivul mobile, către administratorii aplicației, pentru a acționa cu promptitudine și a limita eventualele pagube.

Aplicația este destinată cetățenilor care dețin un dispozitiv mobile pe care rulează sistemul de operare Android, și care vor sa contribuie la modernizarea și buna organizare a orașului. Aceasta reprezintă cea mai simplă conexiune între cetățean și autoritățile administrative, prin punerea la dispoziția cetățeanului a unei forme moderne de sesizare și raportare a problemelor de ordin civic cu care se poate confrunta în spațiul public. De asemenea proiectul vine și cu o aplicație Web destinată administratorului, pentru a putea vedea sesizările trimise de către cetățeni.

Aplicația Android poate fi instalată pe dispozitive care rulează sistemul de operare Android, versiunea minimă 4.0. Am ales acest sistem de operare, datorită faptului că este cea mai răspândită tehnologie pe dispozitivele mobile actuale, fiind totodată o platformă open-source, și în prezent cel mai popular sistem de operare dintre cele dedicate dispozitivelor portabile.

Aplicația reprezintă o linie de urgență pentru problemele publice, care va contribui la dezvoltarea spiritului civic al cetățenilor și la creșterea încrederii în operatorii de servicii publice locale.

Pentru dezvoltarea proiectului, am folosit tehnologii precum Java, pentru aplicația Android, HTML, CSS, Bootstrap, Ajax, jQuery, Javascript pentru implementarea aplicației web pentru administrator, iar serverul a fost implementat în Node.js.

Lucrarea de față este împărțită în patru capitole. Primul capitol, Fundamentarea teoretică și documentarea bibliografică pentru tema propusă, prezintă principalele teme/subiecte similare cu această lucrare, descrierea tehnologiilor folosite pentru implementarea acestui proiect, precum și descrierea anumitor concepte care au fost necesare implementării aplicației.

Al doilea capitol, Proiectarea aplicației, prezintă principalele componente ale proiectului, descrierea pe scurt a principalelor funcționalități ale aplicației, precum și diferite diagrame UML, necesare pentru procesul de proiectare a aplicației.

În al treilea capitol, Implementarea aplicației, este prezentată implementarea aplicației client Android, aplicației Web destinată administratorului, precum și implementarea aplicației

server. De asemenea este descrisă mai detaliat interfața cu utilizatorul atât în cazul aplicației Android, cât și aplicației Web pentru administrator. Spre finalul capitolului, sunt prezentate principalele dificultăți întâmpinate pe parcursul dezvoltării proiectului.

Ultimul capitol, Testarea aplicației și rezultate experimentale, prezintă metodele de testare ale proiectului, precum testarea aplicației Android, validarea codului în cazul aplicației web pentru administrator, precum și testarea aplicației server, prin intermediul anumitor programe speciale. De asemenea, sunt prezentare rezultatele experimentale ale aplicației, precum consumul de memorie sau vizualizarea performanțelor aplicației server.

Capitolul 1. Fundamentarea teoretică și documentarea bibliografică pentru tema propusă

Tema proiectului este reprezentată de realizarea unei aplicații Android pentru trimiterea de sesizări către autorități. De asemenea, proiectul vine însoțit și de o aplicație web destinată administratorului, care are acces la toate sesizările trimise de către utilizatori. Principalul scop este de a oferi utilizatorilor o aplicație cât mai clară și ușor de folosit.

1.1. Referințe la teme/subiecte similare

Ideea acestui proiect a apărut din nevoia de a găsi o aplicație în care se poate semnala anumite probleme din orașul Iași. În urma cercetării aplicațiilor Android, am constatat că nu există o astfel de aplicație pentru municipiul Iași.

Printre subiectele similare cu cel al temei alese, se regăsesc următoarele aplicații: Portal de petiții online, RO-Alert, Let's do It Romania, Sesizări Constanța.

Portalul de petiții online este una dintre cele mai populare aplicații pentru petiții. Petițiile sunt menționate în social-media în fiecare zi, astfel încât crearea unei petiții este un mod extraordinar de a fi luat în seamă de către autorități. Dezavantajul acestui portal este că nu a fost încă implementată o versiune de aplicație și pentru dispozitivele mobile.

Sistemul RO-Alert permite difuzarea de mesaje pentru avertizarea și alarmarea populației în situații de urgență. Sistemul este folosit în situații majore în care viața și sănătatea cetățenilor sunt puse în pericol, cum ar fi fenomene meteo extreme, cutremure sau alte situații care amenință grav comunitățile. Aplicația RO-Alert atenționează cetățenii de anumite probleme, în schimb sistemul implementat de mine face posibilă atenționarea autorităților de către cetățeni.

„Let's do It Romania" este o aplicație disponibilă gratuit atât pe App Store[1] cât și pe Google Play[2], prin care utilizatorii pot raporta zone cu deșeuri din oraș sau din natură. Din 2015, până astăzi aplicația a fost descărcată de peste 10 000 de utilizatori, rezolvându-se peste 500 de cazuri.

1.2. Tehnologii folosite pentru implementare

1.2.1. Sistemul de operare Android

Android este o platformă software și un sistem de operare pentru dispozitive și telefoane bazată pe nucleul Linux[3], fiind un model de arhitectură MVC[4] (Model-View-Controller) dezvoltată inițial de compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance în care fac parte companii ca ARM Holding[5], Intel[6], HTC, LG, Motorola, Samsung Electronics[7] .

Dezvoltarea aplicațiilor Android se realizează în limbajul Java, dezvoltatorul folosindu-se

1 App Store este un serviciu Apple care constă într-un magazin online de aplicații pentru sistemul de operare iOS.
2 Google Play (în trecut Android Market) este un serviciu Google care constă într-un magazin online de aplicații, cărți , filme sau melodii pentru sistemul de operare Android. Acesta este disponibil pe internet, sau prin aplicația Android.
3 Linux este un sistem de operare.
4 MCV, sau Model-View-Controller este un șablon arhitectural folosit în industria software pentru izolarea cu succes a părții logice de interfața proiectului, rezultând în aplicații extrem de ușor de modificat.
5 ARM Holding este o compania britanică specializată în dezvoltarea procesoarelor.
6 Intel este o companie americană care fabrică semiconductoare, și cea care a inventat seria de procesoare x86.
7 HTC, LG, Motorola, Samsung sunt companii care fabrică smartphone-uri cu sistem de operare Android

de API-uri/biblioteci dezvoltate de Google. De astfel, se pot scrie și aplicații în limbajul C, fiind apoi compilate în cod mașină ARM și executate[1].

Pentru dezvoltarea unei aplicații Android sunt necesare [2]:

1. Kit-ul de dezvoltare pentru limbajul de programare Java (JDK);
2. SDK-ul de Android;
3. Un mediu integrat de dezvoltare (IDE): Eclipse cu plugin-ul ADT (Android Developer Tools), Android Studio;
4. Un dispozitiv pe care să ruleze aplicațiile: un emulator, sau un telefon mobile cu sistemul de operare Android;

1.2.1.1. Arhitectura sistemului de operare Android

Pentru a înțelege arhitectura unei aplicații Android este nevoie de un minim de cunoștințe cu privire la conceptele cheie ale aplicațiilor Android. Înțelegerea acestora va permite dezvoltatorului să controleze:

- componentele aplicației;
- ciclul de viață al aplicației;
- resursele;

Principalele funcționalități pe care sistemul de operare Android le oferă sunt:

- operațiile critice (acces la Internet, citire/scriere date de contact, monitorizare SMS, acces la modului GPS, acces la camera foto), pot fi restricționate sau se poate solicita permisiunea utilizatorului, utilizând fișierul manifest de configurare a aplicației, *AndroidManifest.xml [3].*

În continuare sunt prezentate componentele ce formează o aplicație Android, modul cum acestea interacționează, precum și felul cum sunt legate prin intermediul fișierului de configurare *AndroidManifest.xml.*

Următoarele elemente furnizează scheletul pe baza căreia este construită o aplicație Android [4]:

- **Activities:** Fiecare ecran din program este o extensie a clasei Activity. Activitățile utilizează elemente de vizualizare (Views) pentru a forma o interfață grafică care afișează informații și răspunde la acțiunile utilizatorului.
- **Services:** Componentele serviciilor rulează în fundal și trimit date activităților ce permit actualizarea informațiilor expuse și declanșarea de notificări atunci când un eveniment așteptat are loc.
- **Intents:** Intenția este mecanismul de comunicare între elementele unei aplicații Android.
- **Notifications:** Notificările sunt utilizate în scopul atenționării utilizatorilor asupra unui eveniment așteptat, fără ca activitățile curente să-și piardă focalizarea sau să fie întrerupte.

1.2.1.2. Android Studio

Mediul de dezvoltare recomandat de Google pentru platforma Android este Android Studio. Acesta a fost anunțat pe data de 16 Mai 2013 la conferința anuală Google. Acest mediu de dezvoltare este bazat pe software-ul „Intellij IDEA” aparținând companiei de software JesBrains și a fost creat special pentru dezvoltarea aplicațiilor pentru platforma Android.

Android Studio, odată cu apariția sa, a înlocuit ADT-ul (Android Development Tools) mediului de dezvoltare Eclipse, adăugând în același timp o interfață mult mai prietenoasă pentru dezvoltatori, precum și unelte mai ușor accesibile și mai ușor de folosit [2].

Pentru realizarea unei aplicații în Android Studio trebuie respectată o structură bine definită de directoare și fișiere, așa cum arată în Figura 1.1.

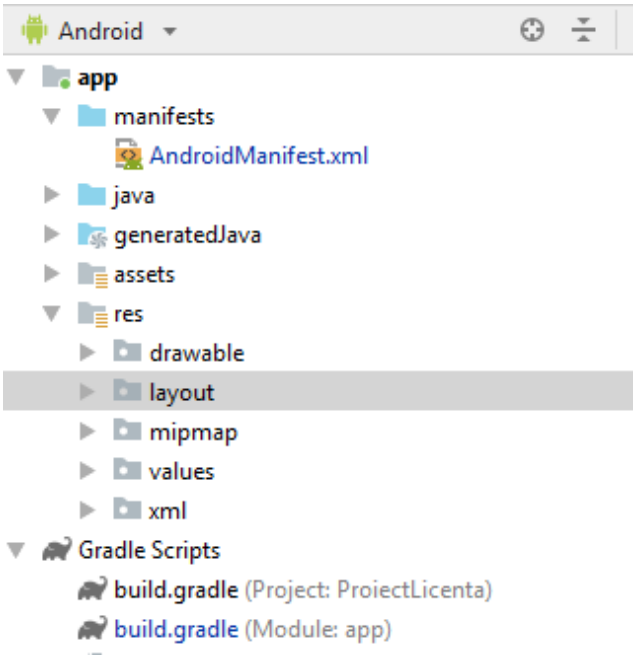

Figura 1.1: Structură de directoare și fișiere în Android Studio

În directorul *manifests/* avem *AndroidManifest.xml* care reprezintă fișierul de configurare a aplicației.

Directorul *java* conține toate fișierele sursă Java.

GeneratedJava conține toate fișierele generate de către Android Studio cum ar fi *R.java*.

Directorul *res* conține toate resursele care sunt folosite pentru crearea aplicației cum ar fi layout-uri, imagini.

Build.gradle (Module app) este fișierul care reprezintă configurațiile pentru build-ul modulului.

1.2.2. Node.js

Node.js este o platformă bazată pe JavaScript, construită peste motorul JavaScript V8 al Google Chrome, destinată dezvoltării de aplicații web rapide. Node.js folosește un model bazat pe evenimente în care operațiile de intrare/ieșire sunt gestionate asincron, astfel încât poate fi utilizat pentru aplicații de timp real care folosesc numeroase operații de intrare/ieșire și care rulează pe dispozitive distribuite, fără ca performanța acestora să fie afectată [5].

Aplicațiile Node.js rulează pe un singur fir de execuție deși platforma folosește mai multe fire pentru evenimentele legate de rețea și fișiere.

Node.js conține intern o biblioteca asincronă pentru operații I/O, socket și comunicații HTTP. Suportul pentru socket și HTTP permite platformei să joace rolul unui server web, fără a mai fi nevoie de software adițional pentru server web precum Apache[8].

Din perspectiva dezvoltării unui server web, Node.js prezintă următoarele beneficii [6]:

- Performanțe bune;
- Toate API-urile din bibliotecile Node.js sunt asincrone și bazate pe evenimente astfel încât nu trebuie să se aștepte ca o anumită operație să furnizeze un rezultat, după ce o anumită metodă a fost invocată, se trece mai departe, urmând ca un mecanism de notificare să indice momentul la care operația a fost terminată și poate furniza date;

8 Apache este o aplicație de tip server web ce a jucat un rol important în creșterea și dezvoltarea World Wide Web.

- Codul este scris în JavaScript;
- JavaScript este un limbaj de programare relativ nou care beneficiază de îmbunătățiri ale design-ului comparativ cu alte limbaje tradiționale web-server (Phython, PHP,etc.);
- Managerul de pachete node (NPM) oferă acces la sute de mii de pachete;
- Este portabil, având versiuni care rulează pe Microsoft Windows, Mac OS sau Linux;

O aplicație Node.js conține următoarele componente:

- încărcarea modulelor pe care la va utiliza este realizată prin intermediul directivei *require()* care primește ca parametru denumirea modulului respectiv;
- construirea unui server HTTP, folosind metoda *createServer()* care primește ca parametru o funcție de callback prin intermediul căreia vor fi gestionate cerere și răspunsul; indicarea portului și adresei prin intermediul căruia vor fi gestionate cererile și răspunsurile este realizată prin intermediul metodei *listen()*;

1.2.3. MongoDB

MongoDB este o bază de date NoSQL open-source orientată pe documente. Această bază de date beneficiază de suport din partea companiei 10gen. MongoDB a fost creat în 2007 de Eliot și Dwight (fondatori DoubleClick) ca un proiect open-source ca urmare a problemelor de scalabilitate ale bazelor de date relaționale. Inițial sistemul se numea 10gen însă în 2009, proiectul a fost redenumit ca și MongoDB. MongoDB face parte din familia sistemelor de baze de date NoSQL.

Diferența principală constă în faptul că stocarea datelor nu se face folosind tabele precum într-o bază de date relațională, MongoDB stochează datele sub formă de documente JSON[9] cu scheme dinamice (MongoDB numește formatul BSON) făcând integrarea datelor în anumite tipuri de aplicații mai ușor și mai rapid.

MongoDB suportă căutarea după câmpuri, interogări sau după expresii regulate. Interogările pot returna câmpuri specifice ale documentelor și pot de asemenea include funcții JavaScript definite de către utilizator. Orice câmp din interiorul unui document stocat în interiorul bazei de date poate fi indexat.

MongoDB poate rula pe o mulțime de servere, echilibrând încărcarea sau duplicând datele pentru a menține sistemul funcțional în cazul unei defecțiuni hardware.

Fiecare document Mongo are asociat un id, echivalent cheii primare. Această valoare poate fi furnizată din exterior atunci când se inserează documentul, sau poate fi generată de sistem [7].

Datorită faptului că MongoDB este ușor de folosit, driverul pentru Node.js poate fi o soluție optimă pentru o aplicație simplistă. Totuși, pentru aplicații mai complexe, se recomandă folosirea unui Object Document Mapper (ODM) pentru modelarea funcțiilor. Mongoose este ODM-ul oficial suportat pentru platforma Node.js.

Mongoose este o bibliotecă ce oferă maparea obiectelor MongoDB într-o interfață familiară în cadrul platformei Node.js. Mongoose transformă datele din baza de date în obiect JavaScript pentru a putea fi folosite în cadrul aplicației. Biblioteca lucrează cu modele și scheme, cele din urmă definind structura documentelor în cadrul unei colecții. Modelele sunt folosite pentru a crea instanțe a datelor ce vor fi stocate în cadrul documentelor.

9 JSON este un acronim în limba engleză pentru JavaScript Object Notation, și este un format de reprezentare și interschimb de date intre aplicații informatice. JSON este alternativa mai simplă, mai facilă decât XML.

1.2.4. Limbajul Java

Java este principalul limbaj de programare folosit în dezvoltarea aplicațiilor pentru sistemul de operare Android. Este o tehnologie inovatoare lansată de compania Sun Microsystems în 1995, care a avut un impact remarcabil asupra întregii comunități a dezvoltatorilor de software, impunându-se prin calități deosebite cum ar fi simplitate, robustețe și nu în ultimul rând portabilitate.

Caracteristicile principale, care l-au transformat într-un interval de timp atât de scurt într-una din cele mai populare opțiuni pentru dezvoltarea de aplicații sunt:

- Simplitate: elimină supraîncărcarea operatorilor, moștenirea multiplă;
- Ușurință în crearea de aplicații complexe ce folosesc programarea în rețea, fire de execuție, interfață grafică, baze de date;
- Robustețe: elimină sursele frecvente de erori ce apar în programare prin renunțarea la pointeri, administrarea automată a memoriei și eliminarea pierderilor de memorie printr-o procedură de colectare a obiectelor care nu mai sunt referite, ce rulează în fundal (garbage collector);
- Orientat pe obiecte: elimină complet stilul de programare procedural;
- Securitate: este un limbaj de programare foarte sigur, furnizând mecanisme stricte de securitate a programelor [8];

1.2.5. Limbajul HTML

HTML (Hyper Text Markup Language) este un limbaj de marcare (markup) care este utilizat pentru crearea de pagini ce pot fi încărcate și vizualizate într-un browser web. Scopul HTML este mai degrabă prezentarea informațiilor (paragrafe, fonturi, tabele) decât descrierea semanticii documentului [9].

Prima versiune a HTML-ului (HTML 1.0) a apărut în vara lui 1991 și avea suport din partea primului browser cu succes, Mosaic. Prima versiune oficială, HTML 2.0 a fost admisă ca standard în septembrie 1995. Următoarea versiune, HTML 3.0 nu a avut prea mult succes. Versiunea 4.0 introduce CSS (Cascade Style Sheets). În ianuarie 2008, a fost publicat un document de lucru pentru o nouă versiune, HTML5. Specificația sa a fost finalizată în octombrie 2014 [10].

Orice document HTML începe cu notația *<html>* și se termină cu notația *</html>*. Acestea se numesc în literatura de specialitate „TAG-uri”. Prin convenție, toate informațiile HTML încep cu o paranteză unghiulară deschisă „<” și se termină cu o paranteză unghiulară închisă „>”.

Cu tag-ul *<head>* se începe cea de-a doua secțiune, care conține informații ce nu se afișează în browser, cu excepția marcajului *<title>* în care se specifică titlul paginii web și apare ca titlul ferestrei browser-ului.

Marcajul *<head>* conține informații cu privire la cuvintele cheie, descriere, înserare de cod JavaScript în pagină.

În interiorul marcajului *<body>* se află tot conținutul paginii web care va apărea în browser (text, imagini, tabele, etc.) [11].

1.2.6. CSS

CSS sau Cascade Style Sheet este un standard pentru formatarea documentelor HTML. Acest lucru se face prin intermediul fișierelor de tip **.css** externe sau al codului CSS introdus prin elementul *<style>* și/sau atributului *style* al unui element. Un set de reguli CSS constă dintr-un selector și un bloc de declarații.

Selectorii CSS selectează elementele HTML în funcție de numele, id-ul, clasa unui element HTML, atribut [12].

1.2.7. Bootstrap

Bootstrap este un framework[10] pentru dezvoltarea front-end-ului aplicațiilor web. Inițial, a fost un proiect intern al companiei Twitter, pentru a asigura coerența internă a platformei.

Bootstrap include șabloane HTML și CSS pentru elemente tipografice, fonturi, butoane, tabele, imagini, precum și scripturi JavaScript [13].

Principalele caracteristici sunt:

- ușor de utilizat;
- flexibil („responsive”), se ajustează la telefoane, tablete sau desktop-uri;
- asigură compatibilitatea cu browserele principale.

1.2.8. Ajax

AJAX (Asynchronous JavaScript And XML) este o practică de programare pentru a construi pagini web mai complexe și mai dinamice, utilizând o tehnologie cunoscută sub numele de XMLHttpRequest.

Ajax permite actualizarea anumitor părți de DOM în pagini HTML fără a fi nevoie să se reîncarce toată pagina. De asemenea, AJAX permite comunicarea asincronă, codul continuă să ruleze în timp ce se încearcă reîncărcarea părții respective a paginii web [14].

Ajax utilizează un model de programare cu afișaj și evenimente. Aceste evenimente sunt în cea mai mare parte acțiuni ale utilizatorului, ele au drept consecință invocarea unor funcții asociate elementelor paginii.

Interactivitatea este obținută prin forme și butoane. Implementarea DOM permite asocierea elementelor paginii cu acțiuni și permite extragerea datelor din fișierele XML trimise de către server.

Pentru a obține date de la server, mașinăria Ajax utilizează obiectul XMLHttpRequest. Acest obiect oferă două metode pentru comunicarea cu serverul:

- **open**: pentru crearea unei conexiuni;
- **send**: pentru a trimite cereri către server.

1.2.9. jQuery

jQuery este o platformă de dezvoltare JavaScript, concepută pentru a ușura și îmbunătăți procese precum traversarea arborelui DOM[11] în HTML, managementul inter-browser al evenimentelor, animații și cereri tip AJAX. Biblioteca a fost lansată în 2006 de către John Resig [15].

jQuery se poate folosi pentru a rezolva următoarele probleme specifice programării web:

- înregistrarea și modificarea evenimentelor din browser;
- manipularea elementelor CSS;
- efecte și animații;
- cereri tip AJAX;
- procesare JSON;

10 Un framework este o aplicație ce conține o colecție de scripturi cu ajutorul căreia creatorii de pagini web pot realiza mult mai rapid un site complex.

11 DOM (Document Object Model) este o interfață de programare a aplicațiilor inter-platformă și independentă de limbaj care tratează un document HTML sau XML ca o structură de arbore în care fiecare nod este un obiect reprezentând o parte a documentului.

- extensii [16].

Întreaga bibliotecă jQuery este un singur fișier de tip JavaScript care conține implementarea tuturor metodelor sale. Poate fi inclusă într-o pagină web prin includerea unei copii locale, sau poate fi accesată printr-o rețea de livrare a conținutului, CDN (Content Delivery Network). Printre CDN-urile care oferă jQuery sunt MaxCDN, Google sau Microsoft.

Includerea prin intermediul unui CDN este simplă, de exemplu:

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
```

jQuery are două moduri de utilizare:

- prin intermediul funcției **$**, care este o metodă „factory" pentru obiectul jQuery. Aceste funcții pot fi utilizate în cascadă, deoarece returnează obiecte jQuery;
- prin intermediul funcțiilor cu prefixul **$**. Acestea sunt funcții utilitare care nu acționează direct asupra obiectului jQuery.

1.3. Concepte

Implementarea practică a aplicației necesită cunoștințe generale despre protocolul HTTP și conceptul de serviciu web.

1.3.1. Serviciu web

Spațiul World Wide Web reprezintă un sistem de distribuție locală sau globală a informațiilor hipermedia, văzut ca univers informațional compus din elemente de interes, denumite resurse, desemnate de identificatori globali, URI (Uniform Resource Identifiers) [17].

Multitudinea de protocoale și standarde disponibile începând de la sfârșitul secolului trecut în sfera Internetului au dat posibilitatea comunicării între aplicații pe sisteme aflate la distanțe mari, cu acces la Internet. Serviciile de prelucrare de informații pornesc de la cele mai banale, cum ar fi execuția de operații aritmetice, și până la servicii complexe cum ar fi serviciile de autentificare.

Un serviciu Web (Web Service) este o aplicație Web de tip client-server, în care un server furnizor de servicii (numit și „Service Endpoint") este accesibil unor aplicații client (care nu sunt de tip browser) pe baza adresei URL a serviciului. Serviciul Web și clienții săi pot rula pe platforme diferite și pot fi scrise în limbaje diferite, deoarece comunică prin protocoale standard HTTP, XML, JSON.

Serviciile web au două tipuri de utilizare. În primul rând componentele unui serviciu web sunt reutilizabile. Sunt lucruri de care aplicațiile au nevoie foarte des. De ce să construim mereu aceste lucruri când putem să le reutilizăm? De aceea au fost create serviciile web. În al doilea rând, serviciile web conectează aplicații existente deja. Ele ajută să rezolvi problema interoperabilității permițând schimbul datelor între diferite aplicații și platforme. Acestea reprezintă doua avantaje majore ale serviciilor web.

Furnizorul de servicii expune un API[12] pe Internet, adică o serie de metode ce pot fi apelate de clienți. Aplicația client trebuie să cunoască adresa URL a furnizorului de servicii și metodele prin care are acces la serviciul oferit (nume, parametri, rezultat). Interfața API este limitată la câteva operații în cazul serviciilor de tip REST, și nelimitată ca număr și ca diversitate a operațiilor în cazul serviciilor de tip SOAP.

12 API (Application Programming Interface) reprezintă un set de definiții de sub-programe, protocoale și unelte pentru programarea de aplicații software. Un API poate fi pentru un sistem web, sistem de operare, sistem de baze de date, hardware sau biblioteci software

Diferența dintre o aplicație Web clasică și un serviciu Web constă în principal în formatul documentelor primite de client și a modului cum sunt ele folosite. Într-o aplicație Web clientul primește documente HTML transformate de un browser în pagini afișate, iar clientul unui unui serviciu Web primește un documente XML (sau JSON) folosit de aplicația client, dar care nu se afișează direct pe ecran.

Din punct de vedere al tehnologiilor folosite există două tipuri de servicii Web:

- Servicii de tip REST (RESTful Web Services), în care cererile de la client se exprimă prin comenzi HTTP (GET, PUT, POST, DELETE), iar răspunsurile sunt primite ca documente XML sau JSON;
- Servicii de tip SOAP (Simple Object Access Protocol), în care cererile și răspunsurile au forma unor mesaje SOAP (documente XML cu un anumit format) transmise tot peste HTTP [18].

Lucrarea curentă va aborda prima categorie de servicii web, întrucât acestea au căpătat o popularizare deosebită în ultimii ani și joacă un rol important în cadrul unor soft-uri folosite zilnic de sute de milioane de oameni, precum rețelele de socializare sau aplicațiile mobile.

1.3.2. Protocolul HTTP

Hypertext Transfer Protocol (HTTP) este un protocol de nivel aplicație pentru transmiterea distribuită și colaborativă a datelor în cadrul sistemelor informatice bazate pe hypermedia. Acesta constituie fundamentul pentru totalitatea transferurilor de date ce au loc în cadrul World Wide Web. Specificațiile și standardele ce au condus la dezvoltarea protocolului HTTP au fost coordonate de consorțiul World Wide Web și de Internet Engineering Task Force (IETF) [19].

HTTP este un protocol de comunicație responsabil de transferul de hipertext[13] dintre un client și un server web. Acesta este un protocol fără stare, pentru persistența informațiilor intre accesări fiind necesar să se utilizeze soluții adiacente (cookie, sesiune, câmpuri ascunse).

Principalele concepte cu care lucrează acest protocol sunt *cererea* și *răspunsul* după cum se poate vedea și în Figura 1.2.

- **Cererea** este transmisă de client către serverul web și reprezintă o solicitare pentru obținerea unor resurse (identificare printr-un URL). Aceasta conține denumirea metodei care va fi utilizată pentru transferul de informații, locația de unde se găsește resursa și versiunea de protocol. Un client deschide o conexiune, de regulă pe portul 80.

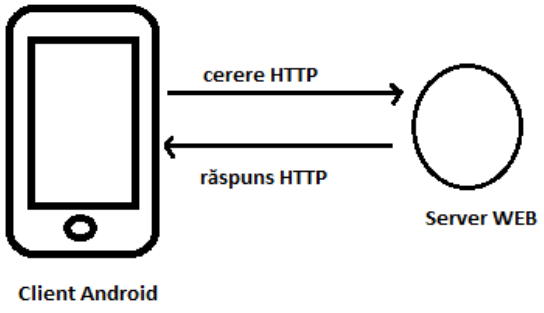

Figura 1.2: Protocolul HTTP

- **Răspunsul** este transmis de serverul web către client. Serverul procesează cererea și transmite înapoi un răspuns, conexiunea fiind apoi închisă [20].

Metodele uzuale ale protocolului HTTP 1.1 sunt următoarele:

13 Hipertext: text structurat ce conține legături.

- **GET**: descărcarea resursei specificate de pe serverul web pe client. Tipul resursei returnate poate fi oricare, precum o pagina web sau un fișier video.
- **POST**: o cerere de transmitere spre acceptare a unei resurse de la client la server.
- **HEAD**: acțiune identică cu GET cu excepția faptului că resursa nu este returnată. Se folosește în general pentru a reduce traficul de rețea în momentul în care este necesară doar verificarea existenței unei resurse.
- **PUT**: acțiune de actualizare a unei resurse existente.
- **DELETE**: ștergerea resursei specificate de pe serverul web, rezultatul operației depinzând de permisiunile pe care le deține utilizatorul ale cărui date de autentificare au fost transmise în antete.
- **OPTIONS**: o interogare a server-ului despre capabilitățile sale. Răspunsul conține o listă de metode HTTP suportate de o resursă, precum și alte informații utile.
- **TRACE**: solicitare de retransmitere a cererii primite de serverul web de la client, pentru a se testa corectitudinea acesteia.

Un răspuns HTTP este format din linia de stare, antetele de răspuns și posibile informații suplimentare, conținând o parte sau toată resursa care a fost solicitată de client de pe serverul web.

Codurile de status ale răspunsurilor sunt formate din trei cifre și reflectă rezultatul procesării unei cereri. Acestea sunt clasate în cinci categorii:

- *Informaționale (cu forma 1xx):* răspuns provizoriu, indicând faptul că cererea a fost primită.
- *De succes (2xx)*: răspuns ce indică faptul că cererea a fost primită, înțeleasă, acceptată și procesată cu succes.
- *De redirectare (3xx)*: răspuns transmis de serverul web ce indică faptul că trebuie realizate acțiuni suplimentare din partea clientului. În cazul în care redirectarea se repetă de mai multe ori, se poate suspecta o buclă infinită.
- *Eroare la client (4xx)*: răspuns transmis de serverul web ce indică faptul ca cererea nu a putut fi îndeplinită, datorită unei erori la nivelul clientului. Mesajul include și o entitate ce conține o descriere a situației, inclusiv tipul acesteia.
- *Eroare la server (5xx)*: cod de răspuns ce indică clientului faptul că cererea nu a putut fi îndeplinită datorită unei erori la nivelul serverului web.

1.3.3. Stilul arhitectural REST

Representational State Transfer (REST) este un stil arhitectural de dezvoltare a aplicațiilor Web axat asupra reprezentării datelor [21]. Acesta a fost inventat de Roy Fielding în teza sa de doctorat pentru a descrie un stil de arhitectură a sistemelor în rețea.

Serviciile web RESTful reprezintă un mijloc modern pentru integrarea și asigurarea interoperabilității sistemelor software. Adoptarea serviciilor web RESTful, denumite și API-uri, a cunoscut în ultimii ani o popularizare foarte rapidă mai ales în cadrul dezvoltării aplicațiilor web și mobile.

REST nu este un standard, ci un stil arhitectural, însă face standarde de utilizare precum: HTTP, UTL-ul, XML/HTML/GIF [22].

În teorie, REST nu este „legat” de web ci poate fi folosit independent de acesta. Totuși, majoritatea implementărilor sunt, în practică, servicii sau aplicații web, acest lucru fiind și o consecință a faptului că protocolul HTTP a inspirat proiectarea sa.

De aceea, REST poate fi aplicat oriunde se folosește protocolul HTTP, iar prin generalizare, putem afirma faptul că web-ul este cel mai larg sistem care aderă la stilul

arhitectural REST.

Printre principalele avantaje ale unui serviciu RESTful se numără [23]:

- independența față de platformă, astfel, sistemul de operare al server-ului poate diferi de cel al clientului;
- independența față de limbajul de programare, astfel, o aplicație scrisă în limbaj Java poate comunica fără probleme cu o aplicație de tip server scrisă în limbaj C# sau JavaScript;
- este integrat foarte ușor în alte servicii web deja existente;
- este simplu de implementat.

Serviciile REST prezintă și un dezavantaj, și anume faptul că folosește cookie-uri. Un cookie este un mic fișier text stocat pe calculatorul utilizatorului ce-i oferă posibilitatea site-ului web căreia îi este asociat de a obține anumite informații asupra clientului, ca de exemplu: nume utilizator, parolă, număr de telefon. Cookie-urile nu sunt sigure din moment ce sunt păstrate în text clar, de aceea ele pot fi accesate de către oricine. Există însă o metodă de a rezolva acest lucru, aceea de a cripta și decripta manual, însă acest lucru implică scrierea de cod în plus ce afectează performanța aplicației deoarece necesită un timp suplimentar pentru criptare/decriptare [24].

Capitolul 2. Proiectarea aplicației

Aplicația este formată din trei componente: o aplicație client ce rulează pe dispozitive Android, o aplicație Web pentru contul de administrator, și o aplicație server.

Aplicația client rulează pe dispozitive Android cu versiunea cel puțin 4.0. Astfel, aplicația poate fi folosită pe aproximativ toate dispozitivele mobile pe care rulează sistemul de operare Android. Atât aplicația Android cât și aplicația Web va comunica cu aplicația server a sistemului.

2.1. Aplicația client Android

Aplicația Android a fost dezvoltată în Android Studio. Întrucât principala componentă a proiectului este aplicația mobile, s-a pus mare accent pe design-ul acesteia, precum și pe funcționalitățile pe care aceasta le oferă.

În continuare sunt prezentate principalele funcționalități ale aplicației Android. Acestea vor fi prezentate mai detaliat în capitolul următor.

Principalele funcționalități ale aplicației sunt:

- Înregistrarea/autentificarea utilizatorului
  - necesară pentru a putea utiliza aplicația în continuare
  - permite utilizatorului să se înregistreze sau să se autentifice în cadrul aplicației
- Resetarea parolei
  - în cazul în care utilizatorul și-a uitat parola, acesta va primi pe email un cod cu care își va putea schimba parola
- Vizualizarea profilului
  - utilizatorul își va putea vedea datele, precum și numărul de sesizări trimise și/sau rezolvate
  - tot în această secțiune utilizatorul are opțiunea de a-și schimba parola
- Trimiterea unei noi sesizări
  - aceasta este etapa cea mai importantă din cadrul aplicației
  - implică mai multe etape: alegerea categoriei în care se încadrează sesizarea, localizarea și obținerea adresei exacte, încărcarea unei imagini, descrierea mai detaliat a problemei, și trimiterea sesizării către server prin intermediul unui serviciu REST.
- Vizualizarea sesizărilor trimise
  - utilizatorul are opțiunea de a vedea toate sesizările trimise de când și-a creat contul
  - sesizările sunt filtrate în funcție de status
- Vizualizarea sesizărilor trimise în ultimele 24 de ore de către toți utilizatorii
  - utilizatorul are posibilitatea să vadă sesizările trimise de către toți utilizatorii aplicației din ultimele 24 de ore
- Vizualizarea diferitelor statistici după numărul de sesizări trimise în funcție de categorie, sau de utilizator.

2.1.1. Arhitectura aplicației client Android

Utilizatorul va interacționa cu aplicația Android, respectând paradigma client-server, aceasta urmând să trimită cereri serverului, și să aștepte un răspuns de la acesta. Odată ce a primit răspunsul, aplicația îi va furniza utilizatorului informațiile cerute (vezi Figura 2.1).

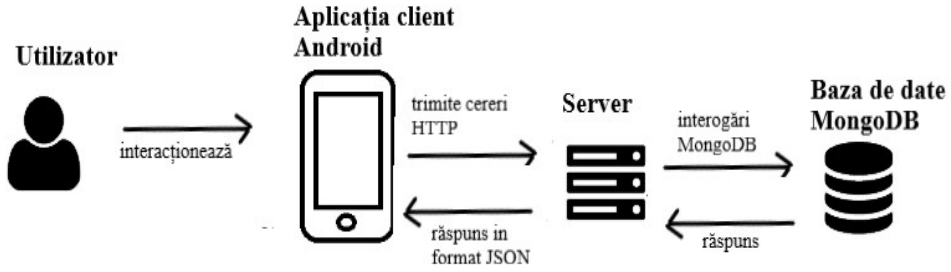

Figura 2.1: Arhitectura aplicației client Android

Comunicarea dintre cele două părți se bazează pe schimbul de obiecte JSON, realizat prin intermediul protocolul HTTP.

2.1.2. Retrofit

Pentru comunicarea cu API-ul voi folosi biblioteca Retrofit, o bibliotecă open-source. Retrofit este una dintre cele mai populare librării folosită de Android, construit de către firma Square. Aceasta este o librărie cu ajutorul căreia se poate implementa sistemul Android ca un client REST, și care furnizează un framework puternic pentru interacțiunea cu API-uri și trimiterea de cereri în rețea utilizând biblioteca *OkHttp [25]*.

OkHttp este un client HTTP pentru Android și Java, care îmbunătățește modul prin care aplicațiile comunică prin rețea.

Librăria face descărcarea datelor în format XML sau JSON de la un API, convertind datele în obiecte ale domeniului, în cazul nostru în obiecte Java. Retrofit realizează conversia obiectelor JSON în obiecte Java utilizând biblioteca GSON[14].

Pentru obținerea bibliotecilor GSON și Retrofit am inclus în fișierul *app/GradleScripts/build.gradle* următoarele dependențe:

```
implementation 'com.squareup.retrofit2:converter-scalars:2.3.0'
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
```

După ce am inclus dependențele, am creat o interfață în care am specificat metodele API-ului. În această interfață am specificat tipul cererii HTTP (GET,POST,PUT), adresa la care se află metoda și tipul conținutului.

În secvența de mai jos se poate observa o parte din metodele API-ului, folosite de aplicația client.

Pentru că API-ul implementat în această lucrare este securizat, am adăugat pentru fiecare metodă, parametrul *@Header(„Authorization")* pentru cererile HTTP care să adauge în antetul acestora jetonul de autentificare.

14 GSON este o bibliotecă Java realizată de către cei de la Google ce are ca scop conversia obiectelor Java în obiecte JSON și invers.

```
public interface MyService {
    @POST("register")
    @FormUrlEncoded
    Call<String>registerUser(@Field("nume") String nume,
                             @Field("prenume") String prenume,
                             @Field("email") String email,
                             @Field("password") String password);

    @POST("login")
    @FormUrlEncoded
    Call<String>loginUser(@Field("email") String email,
                          @Field("password") String password);

    @GET("users/{email}")
    Call<String>getName(@Path("email")String email,
                        @Header("Authorization")String token);
}
```

Următorul pas este să obținem o instanță a unui obiect care să implementeze interfața definită mai sus. Obiectul este creat de librăria Retrofit, dar trebuie să mai adăugăm câteva configurări precum adresa URL a API-ului.

```
public class RetrofitClient {
    private static Retrofit instance;
    public static Retrofit getInstance(){
        if(instance==null)
        {
            instance=new Retrofit.Builder()
                .baseUrl("http://192.168.137.1:3000/")
                .addConverterFactory(ScalarsConverterFactory.create())
                .build();
        }
        return instance;
    }
}
```

2.2. Aplicația Web pentru administrator

Aplicația Web a fost creată cu scopul de a ajuta administratorii să vizualizeze sesizările trimise de către utilizatorii aplicației Android. Pentru crearea acesteia au fost utilizate tehnologii precum HTML, CSS, Bootstrap, Ajax, jQuery.

Principalele funcționalități ale aplicației sunt:

- Vizualizarea sesizărilor primite de la utilizatori, sub forma unui tabel;
- Filtrarea datelor după categorie și status;
- Schimbarea statusului și trimiterea unui email utilizatorului care va conține un scurt comentariu și statusul nou;
- Diferite statistici pe baza numărului de sesizări trimise în funcție de categorie, sau un top al utilizatorilor
- Vizualizarea pe hartă a adresei de unde s-a trimis sesizarea;

Implementarea acestor funcționalități va fi detaliată în capitolul următor.

2.3. Diagrame UML

UML (Unified Modeling Language) este un limbaj de modelare utilizat pentru specificare, construirea și documentarea sistemelor de aplicații orientate obiect și nu numai. O diagramă este o prezentare grafică ale unui set de elemente, cel mai adesea exprimate ca un graf de noduri (elementele) și arce (relațiile) [26].

Diagramele UML au fost realizate în Altova UModel. Altova Umodel este un instrument vizual pentru crearea de diagrame UML. Pot genera cod Java, C# și Visual Basic .NET pe baza diagramelor și poate să realizeze diagrame UML ale programelor existente [27].

2.3.1. Diagrame use-case

Diagrama cazurilor de utilizare (use-case diagram) este un tip de diagramă din care reiese modul de utilizare a sistemului informatic, modul în care utilizatorii interacționează cu acesta (în corespondență directă cu task-urile acestor utilizatori).

În continuare sunt prezentate cele trei diagrame de tip Use-Case, prima pentru procesul de înregistrare/autentificare, a doua pentru momentul după ce utilizatorul s-a logat cu succes în aplicația Android, iar a treia pentru aplicația Web destinată administratorului.

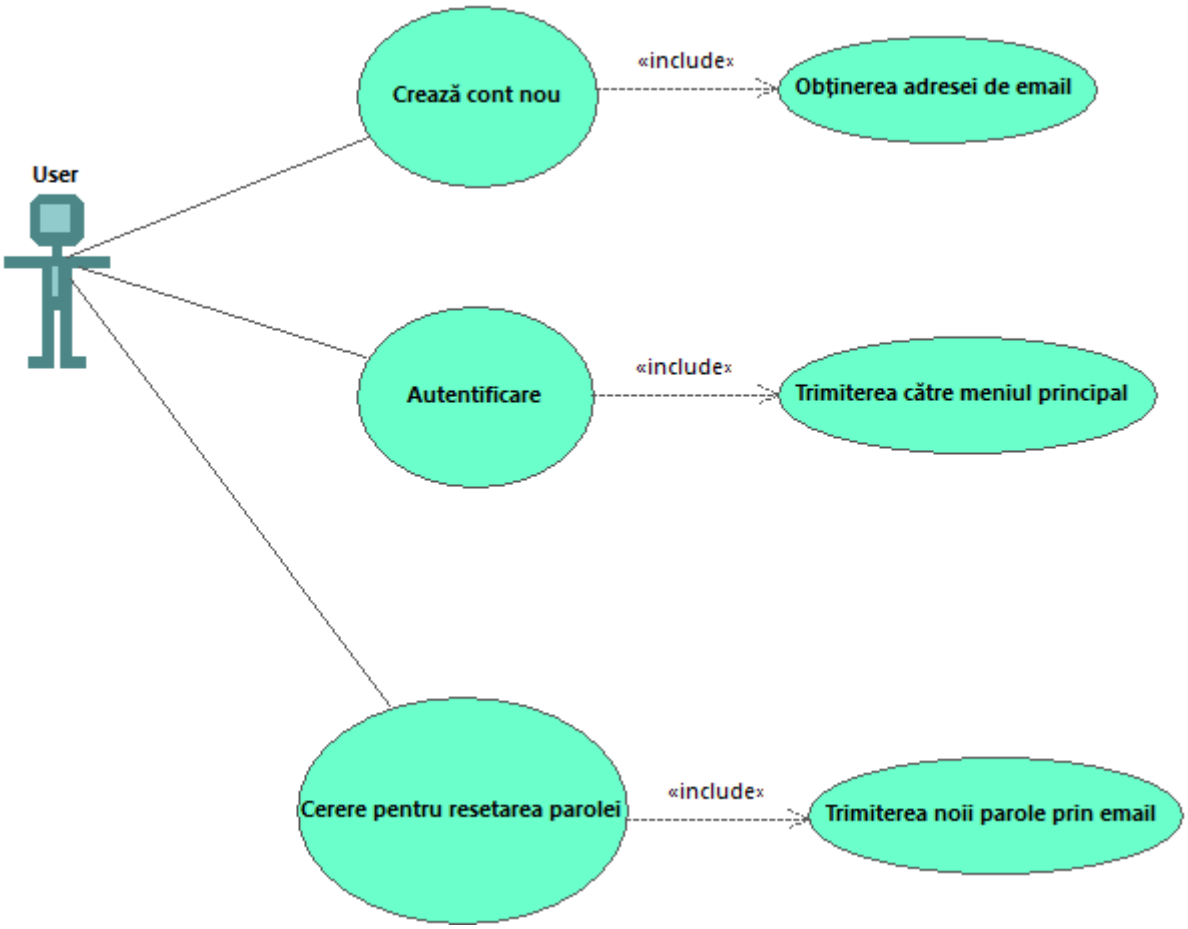

Figura 2.2: Diagrama Use-Case aferentă procesului de înregistrare/autentificare

Figura 2.3: Diagrama use-case aferentă procesului de după autentificare

În Figura 2.3 sunt prezentate principalele funcționalități de care dispune utilizatorul după ce s-a logat cu succes în aplicația Android.

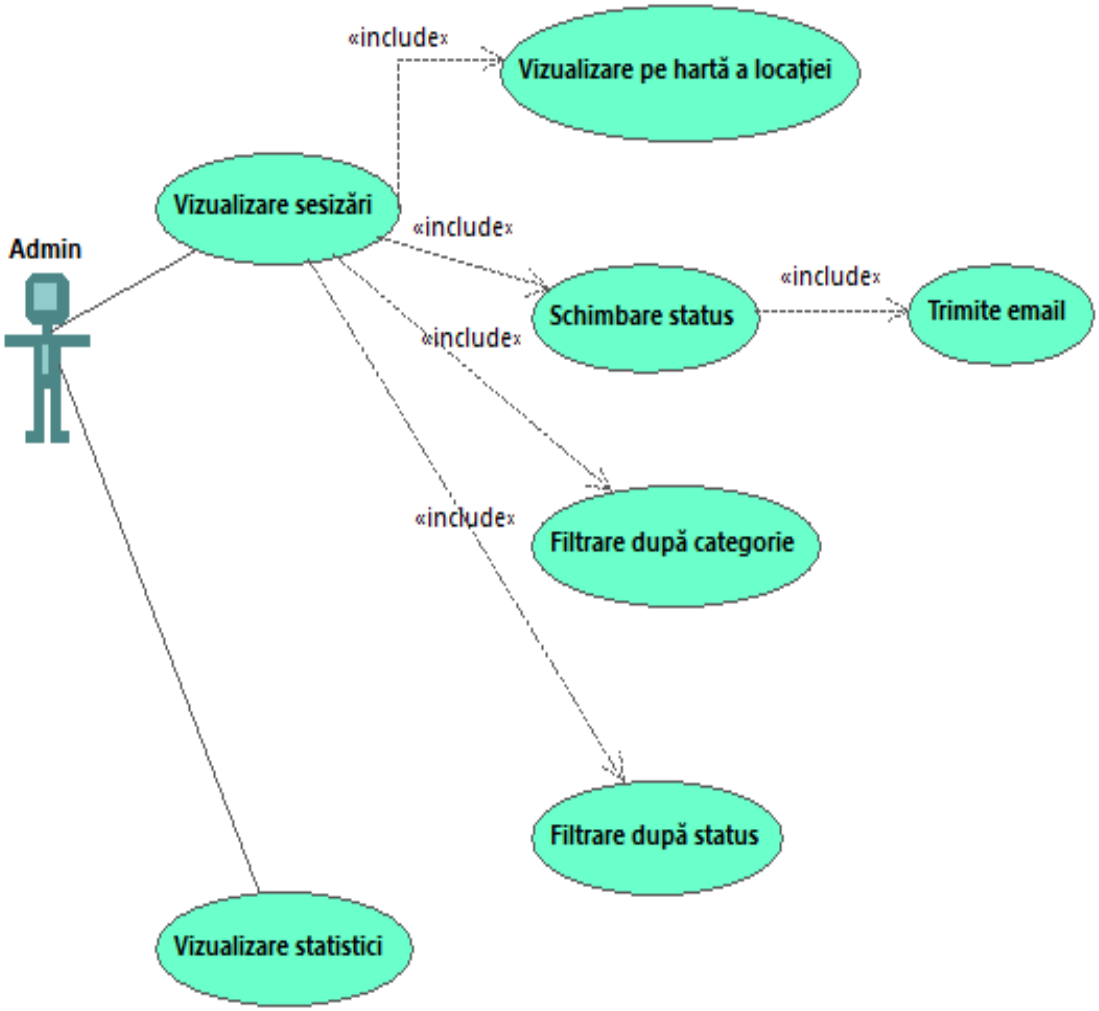

Figura 2.5: Diagrama Use-Case pentru aplicația dedicată administratorului

În figura Figura 2.5 sunt prezentate principalele funcționalități ale aplicației dedicate administratorului.

2.3.2. Diagrama de activități

Diagrama de activități scot în evidență controlul execuției de la o activitate la alta, și sunt folosite pentru modelarea proceselor sau a algoritmilor din spatele unui anumit caz de utilizare.

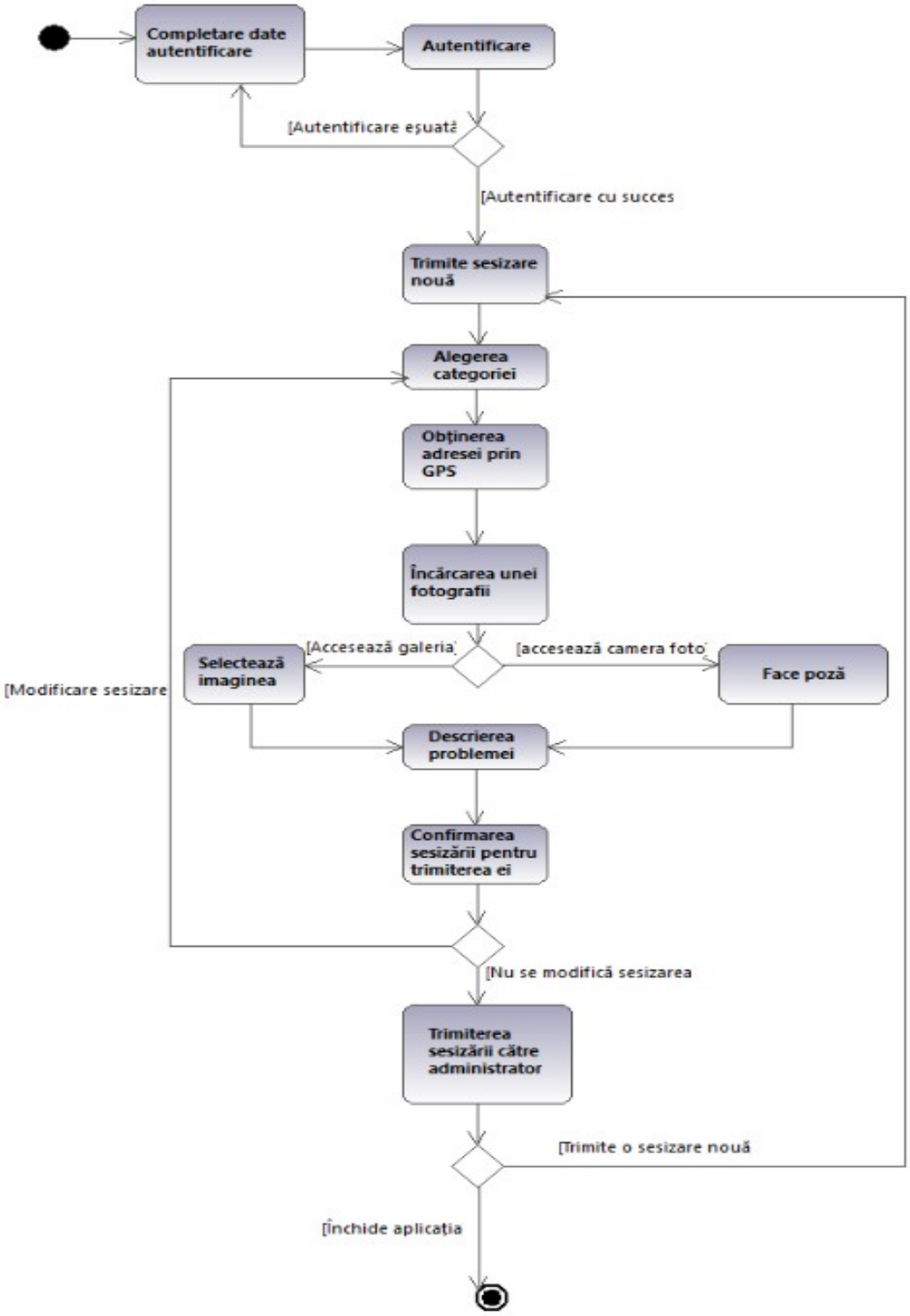

Figura 2.6: Diagrama de activități cu decizii pentru procesul de trimitere a unei sesizări

Diagrama din Figura 2.3 prezintă procesul ce pornește de la autentificarea utilizatorului și dorește să trimită o sesizare, până la confirmarea că sesizarea a fost trimisă cu succes.

Capitolul 3. Implementarea aplicației

3.1. Implementarea aplicației client Android

Din momentul lansării și până în prezent au apărut numeroase versiuni de Android, care au adus modificări importante platformei mobile. Ultima versiune apărută este Android 9, lansat în anul 2018.

Nu toți utilizatorii sistemului de operare Android sunt la zi cu ultimele versiuni, și astfel, dezvoltatorii sunt nevoiți să își proiecteze aplicațiile astfel încât să își păstreze utilizatorii existenți, dar să fie și în pas cu ultimele apariții în materie de tehnologii mobile.

De fiecare dată când Google introduce o nouă versiune de Android, este introdus și un nou SDK. Android SDK este un set de unelte de dezvoltare utilizate în crearea aplicațiilor Android. Setul include componente precum: emulator, exemple de cod sursă, biblioteci necesare, documentație, debugger. Un emulator simulează un dispozitiv și îl afișează pe calculator. Acesta oferă posibilitatea de a testa aplicațiile Android fără a avea nevoie de un dispozitiv hardware (telefon sau tabletă).

Dezvoltatorii trebuie să descarce și să instaleze fiecare versiune de SDK actuală, care este compatibilă pentru un anumit tip de telefon. Cea mai utilizată metodă de a dezvolta o aplicație Android este prin intermediul unui IDE[15], deoarece majoritatea conțin o interfață grafică ce are ca scop mărirea vitezei de lucru.

Pentru a acoperi un număr cât mai mare de dispozitive, dar să am acces și la cele mai noi versiuni ale funcționalităților pe care le-am implementat în aplicație, am ales nivelul minim al API-ului Android pe care aplicația să îl suporte să fie nivelul 16, corespunzător versiunii de Android 4.1 Jelly Bean.

Înainte de a putea utiliza aplicația Android, am adăugat câteva permisiuni esențiale pentru funcționalitățile aplicației, în fișierul *AndroidManifest.xml*. Pentru ca aplicația să poată comunica cu serverul web, trebuie să avem acces la Internet. Pentru localizarea adresei trebuie să avem acces la locație, iar pentru încărcarea unei fotografii trebuie să permitem accesul camerei foto precum și galeriei dispozitivului mobile.

```
<uses-permission android:name=”android.permission.INTERNET”/>
<uses-permission android:name=”android.permission.ACCES_NETWORK_STATE”/>
<uses-permission android:name=”android.permission.ACCES_FINE_LOCATION”/>
<uses-permission android:name=”android.permission.ACCES_COARSE_LOCATION”/>
<uses-permission android:name=”android.permission.CAMERA”/>
<uses-permission android:name=”android.permission.WRITE_EXTERNAL_STORAGE”/>
```

3.1.1. Interfața cu utilizatorul

În cadrul dezvoltării aplicației Android s-a pus mare accent pe interfața cu utilizatorul, dar s-a mers pe ideea unui design simplu, care să fie ușor de înțeles de către utilizator.

Interfața grafică în cadrul aplicațiilor Android este definită prin utilizarea unor fișiere în format *.xml*. Pentru aceasta am folosit componenta „Activity”.

Activitatea („Activity”) reprezintă o componentă a aplicației Android ce oferă o interfață grafică cu care utilizatorul să interacționeze. O activitate poate invoca o altă activitate pentru a

15 IDE (Integrated Development Environment) reprezintă un mediu de dezvoltare interactiv care oferă facilități complete pentru procesul de dezvoltare software.

realiza diferite sarcini, prin intermediul unei intenții. În acest moment, activitatea veche este oprită și salvată pe stivă, după care este pornită activitatea nouă. Restaurarea și reînceperea activității vechi este realizată în momentul în care activitatea nouă este terminată.

O aplicație poate avea în componența ei una sau mai multe componente de tip Activity, însă doar una dintre acestea poate fi marcată ca și „Activitatea principală" a aplicației, acesta fiind primul obiect de tip Activity care va fi lansat în execuție după pornirea aplicației, în cazul nostru MainActivity, unde utilizatorul se poate înregistra sau autentifica în aplicație.

O activitate poate fi utilizată numai dacă este definită în fișierul *AndroidManifest.xml*, în cadrul elementului de tip <*application*>.

În Anexa 1. putem observa fișierul XML corespunzător paginii de Autentificare/Înregistrare.

3.1.1.1. Înregistrarea/Autentificarea utilizatorului

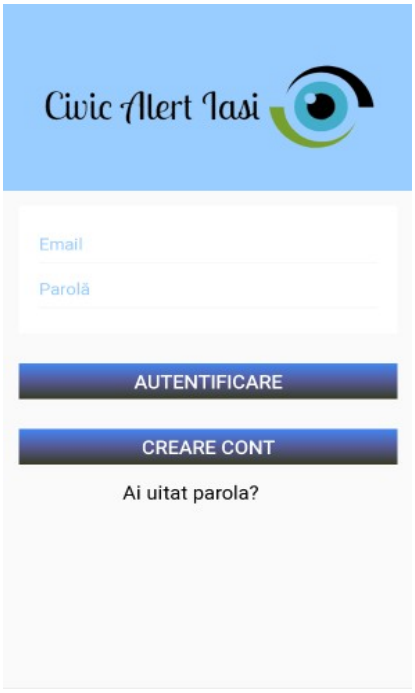

Figura 3.1: Autentificarea în cadrul aplicației

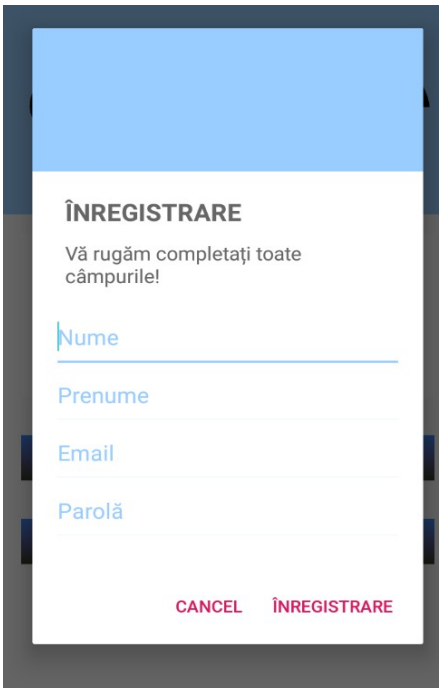

Figura 3.2: Înregistrarea unui nou utilizator

Pentru a putea utiliza aplicația, utilizatorul are nevoie de un cont personal. Pentru realizarea acestui lucru, trebuie să introducă numele, prenumele, adresa de email precum și o parolă conform cu Figura 3.2. După introducerea acestor date, aplicația client va face o cerere către aplicația server a sistemului pentru crearea unui cont. Dacă adresa de email a mai fost utilizată de către un alt utilizator, se va afișa un mesaj de atenționare, altfel contul se va crea cu succes.

După ce contul a fost creat, utilizatorul se va putea autentifica, pentru a putea folosi în continuare aplicația (vezi Figura 3.1). Pentru realizarea acestui lucru, trebuie să își introducă adresa de email și parola. După introducerea datelor, aplicația client va face o cerere către aplicația server a sistemului, pentru a verifica dacă adresa de email există, iar parola este corectă. Dacă datele au fost introduse corect, utilizatorul este logat cu succes, iar odată cu aceasta este generat și un token care va fi folosit pentru securitatea aplicației. Pentru fiecare cerere ce va fi trimisă de către aplicația client, la serverul web, întâi se verifică acel token dacă este corect, și

după se execută metoda. Token-ul este activ până când utilizatorul apasă pe butonul de Logout, sau după ce au trecut 10 ore de la ultima autentificare.

Pentru realizarea design-ului activității din Figura 3.1 am utilizat librăria MaterialEditText pentru câmpurile unde se vor introduce email-ul și parola, un buton pentru autentificare, precum și două TextView pentru resetare parolei sau crearea unui nou cont. Atunci când utilizatorul apasă pe „Creare Cont" o fereastră de tip dialog va apărea (vezi Figura 3.2), ca urmare a apelării funcției *clickCreateAccount()* din clasa *MainActivity.java*. În cadrul funcției am creat o instanță a *MaterialStyledDialog.Builder* unde am setat titlul, descrierea, apelând metodele *setTitle()*, *setDescription(),* precum și apelarea funcției care realizează cererea către server pentru înregistrare.

Prin intermediul Shared Preferences din cadrul aplicației, am acordat funcționalității de autentificare o caracteristică de persistență a datelor, astfel dacă un utilizator închide aplicația cât timp este conectat, nu va fi nevoit să se conecteze din nou atunci când o va redeschide. Shared Preferences reprezintă o modalitate de stocare a unor date, în cazul nostru a adresei de email, într-un format XML care se vor a fi accesibile pentru fiecare activitate a aplicației Android. Astfel, atunci când un utilizator se va conecta, o preferință este creată. Aceasta va conține adresa de email, precum și token-ul de autentificare generat. Atunci când utilizatorul dorește să se deconecteze, prin intermediul butonului Logout, se va șterge această preferință, urmând a fi recreată la următoarea conectare/înregistrare.

```
SharedPreferences sharedPreferences;
SharedPreferences.Editor editor=sharedPreferences.edit();
editor.putString(Constants.EmailKey,txt_email.getText().toString());
editor.apply();
editor.commit();
```

3.1.1.2. Resetarea parolei

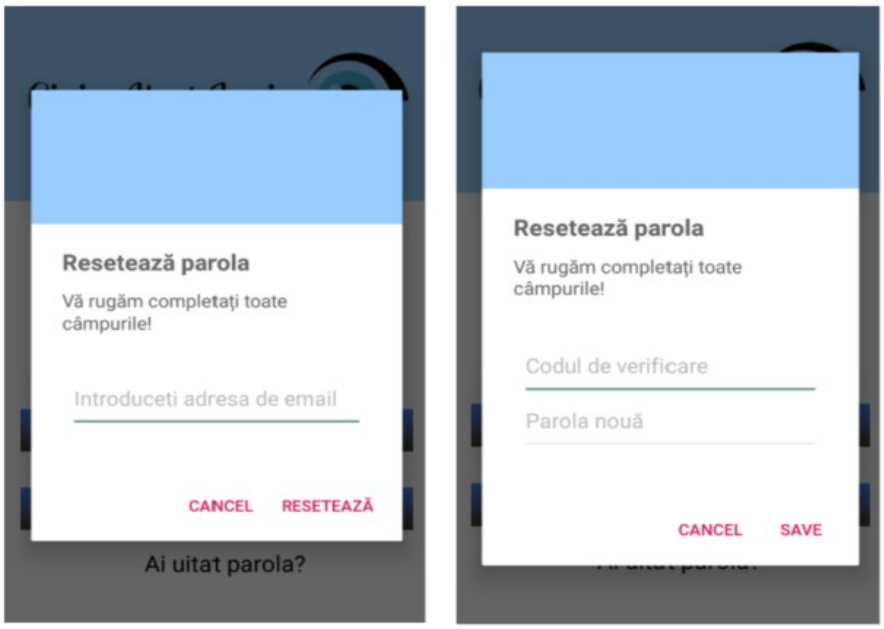

Figura 3.3: Resetarea parolei

Dacă utilizatorul și-a uitat parola, acesta are opțiunea să o reseteze, apăsând pe butonul „Ai uitat parola?" unde va fi apelată funcția *clickResetPassword()* din clasa *MainActivity.java*.

Acesta va trebui să își introducă adresa de email. După ce a apăsat pe butonul „Resetează”, se apelează funcția *resetToken(email)*, unde aplicația va face o cerere către server, care va trimite un email pe adresa furnizată ca parametru. Dacă cererea se realizează cu succes este apelată funcția *verificareToken()* care va deschide o fereastră de tip dialog, unde utilizatorul va introduce codul de verificare primit pe email și noua parolă. Emailul va conține un mesaj cu un cod de verificare, care este valabil doar cinci minute. După ce utilizatorul introduce codul de verificare, și noua parolă, aplicația va face iar o cerere către server pentru a verifica codul, iar dacă acesta este introdus corect, parola va fi schimbată cu succes.

3.1.1.3. Vizualizarea sesizărilor trimise

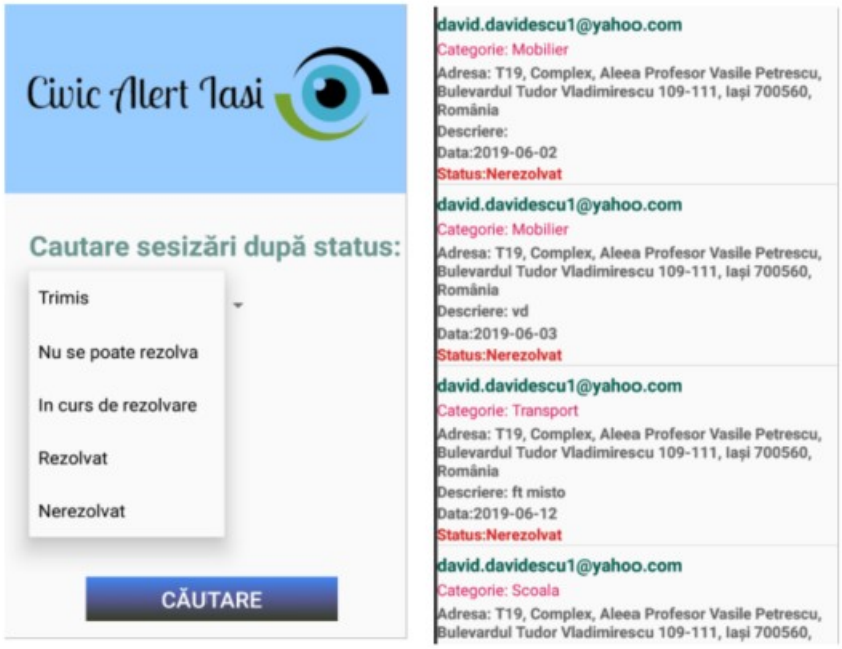

Figura 3.4: Vizualizarea sesizărilor trimise în funcție de status

Această funcționalitate oferă utilizatorului posibilitatea să vizualizeze sesizările trimise în funcție de statusul acesteia. Acesta are la dispoziție o listă cu opțiuni, iar când alege una dintre ele, aplicația face o cerere către server, pentru a prelua datele care vor fi afișate într-o altă activitate. Pentru afișarea colecțiilor de date se folosesc elemente grafice de tip container, referite sub denumirea de liste. Acestea controlează dimensiunile și modul de dispunere al componentelor. Pentru a realiza căutarea sesizărilor în funcție de status, am stocat opțiunile în cadrul unei liste (obiect de tipul Spinner), astfel încât la selecția unui element din cadrul acesteia, se va apela funcția *getMyComplaint()* din clasa *Sesizarile_mele.java* care trimite o cerere către server având ca parametri emailul utilizatorului, opțiunea selectată din cadrul listei, și token-ul pentru securitate.

Serverul va trimite răspunsul sub forma unui JSONArray, unde se realizează parsarea acestuia după email, categorie, adresa, descriere, dată și status. Datele sunt afișate într-un element de tip ListView, date furnizate de un ListAdapter, care trebuie precizat prin intermediul metodei *setAdapter(ListAdapter)*.

3.1.1.4. Trimiterea unei noi sesizări

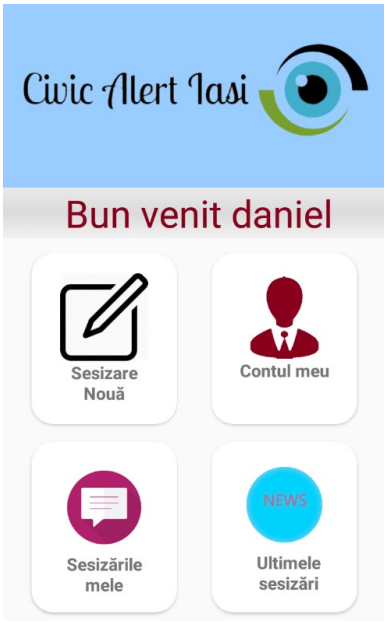

Figura 3.5: Meniul aplicației

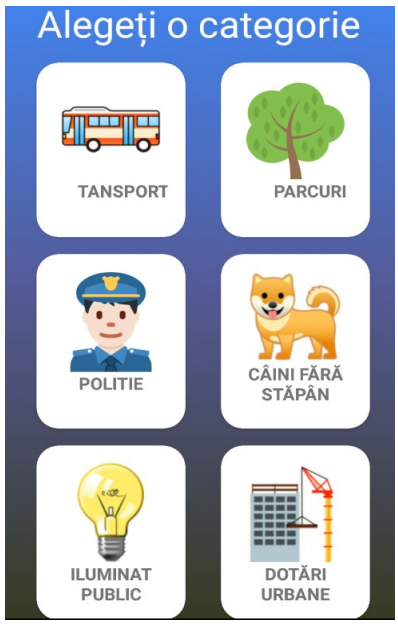

Figura 3.6: Alegerea categoriei

Figura 3.7: Localizare Google Maps

Pentru a putea trimite o sesizare nouă, utilizatorul va apăsa butonul „Sesizare Nouă” (vezi Figura 3.5). Acest buton va direcționa aplicația către o nouă activitate care va pune utilizatorul să aleagă categoria în care se încadrează sesizarea (vezi Figura 3.6). După ce a fost selectată categoria, aplicația va localiza automat locația de unde se trimite sesizarea, cu ajutorul API-ului de la Google Maps (vezi Figura 3.7). API-ul se activează în cadrul console Google API, generându-se totodată și o cheie Android prin care aplicația care rulează pe dispozitivul mobile va putea să acceseze o astfel de funcționalitate.

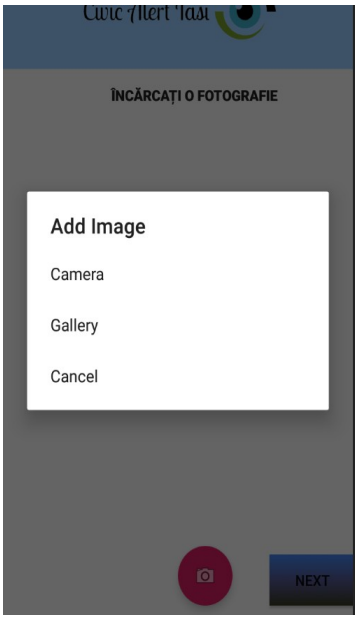

Figura 3.8: Adăugarea unei fotografii

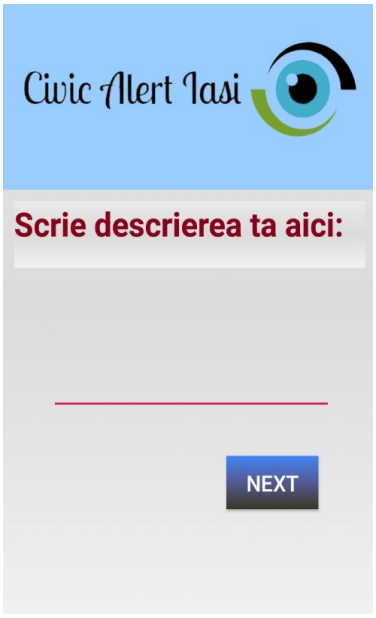

Figura 3.9: Descrierea problemei

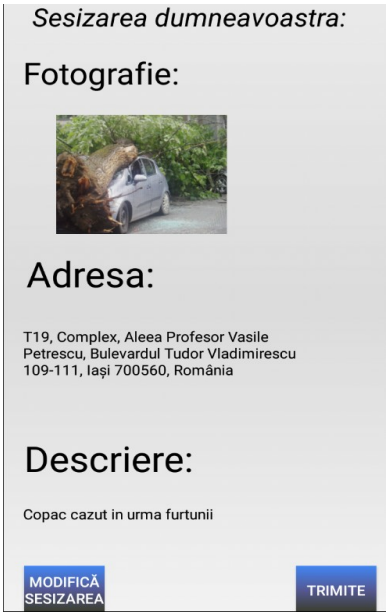

Figura 3.10: Vizualizarea sesizării

Următorul pas este încărcarea unei fotografii care să descrie cât mai bine situația. Utilizatorul are opțiunea să aleagă dacă încarcă o fotografie din galeria telefonului, sau accesează camera pentru a face o poză direct din aplicație (vezi Figura 3.8). După ce fotografia a fost încărcată, următorul pas este ca utilizatorul să descrie problema care urmează a fi raportată către autorități (vezi Figura 3.9).

În final, utilizatorul are șansa de a vizualiza sesizarea înainte să o trimită către autorități (vezi Figura 3.10). Dacă consideră că sesizarea trebuie modificată acesta are opțiunea de a se întoarce la activitatea unde își poate alege categoria. Dacă utilizatorul nu are nimic de modificat asupra sesizării, va apăsa pe butonul „Trimite". Aplicația va face o cerere către serverul web, iar dacă nu există nici o eroare, acesta va insera datele trimise de către utilizator în baza de date.

3.2. Implementarea aplicației Web destinată administratorului

Pentru a putea vizualiza sesizările trimise de către utilizatorii aplicației Android, a fost dezvoltată o aplicație Web destinată administratorului. Pentru crearea acesteia au fost utilizate tehnologii precum HTML, CSS, Bootstrap, Ajax, jQuery. Aceasta va comunica în permanență cu aplicația server.

3.2.1. Interfața cu utilizatorul

Pentru realizarea aplicației Web destinată administratorului, s-a ales o interfață simplă, și cât mai ușor de înțeles. Odată logat cu un cont special de administrator, acesta are acces toate funcționalitățile aplicației web.

3.2.1.1. Dashboard

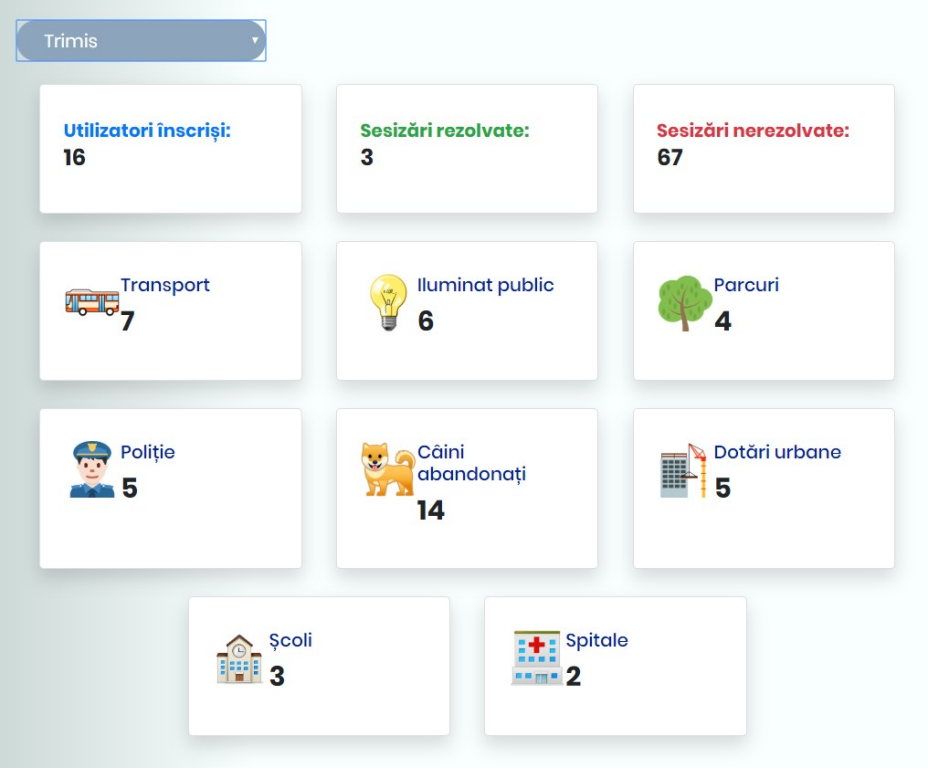

Figura 3.11: Numărul de sesizări pentru fiecare categorie în funcție de status

După ce administratorul se autentifică cu succes, acesta este redirecționat către pagina Home, unde poate vedea numărul de sesizări trimise pentru fiecare categorie în parte, în funcție

de status, precum și numărul de utilizatori înscriși în aplicație, numărul de sesizări rezolvate sau nerezolvate.

3.2.1.2. Vizualizarea sesizărilor

Administratorul are acces la toate sesizările trimise de către utilizatori, care sunt afișate sub forma unui tabel (vezi Figura 3.12). Pentru a ușura munca acestuia, administratorul poate filtra rezultatele după categorie și status.

Figura 3.12: Vizualizarea sub forma tabelară a sesizărilor trimise de utilizatori

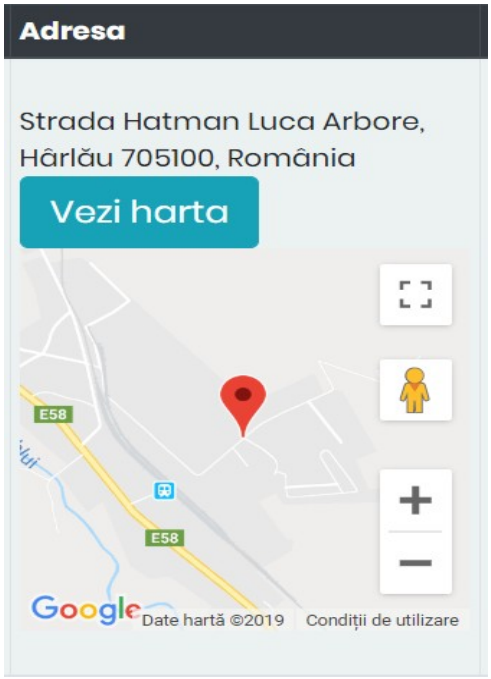

Figura 3.13: Vizualizarea adresei pe hartă

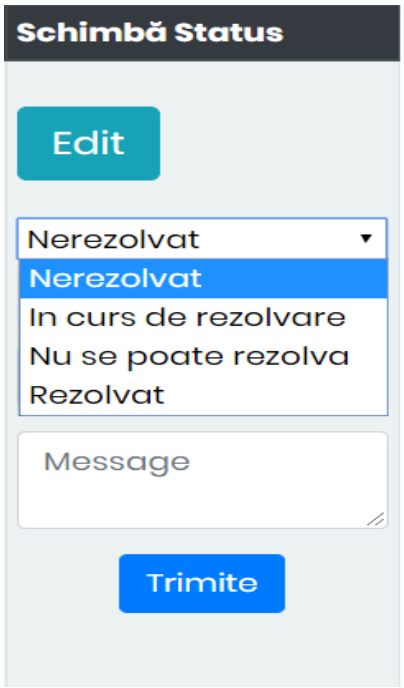

Figura 3.14: Schimbarea statusului si trimiterea unui email utilizatorului

O altă funcționalitate este aceea că administratorul are posibilitatea să schimbe statusul sesizării, în funcție de stadiul rezolvării acesteia, și să informeze persoana care a trimis sesizarea, printr-un email (vezi Figura 3.14). De asemenea administratorul poate vedea pe hartă adresa exactă unde se află problema care trebuie rezolvată. Acest lucru este posibil datorită API-ului de la Google Maps, Maps JavaScript API. Pentru localizare, am utilizat funcția Geocoder, care primește ca parametru adresa.

3.2.1.3. Statistici

Pentru a avea o evidență mai clară a numărului de sesizări trimise în funcție de fiecare categorie în parte, am creat o diagrama de tip PieChart ca în figura de mai jos.

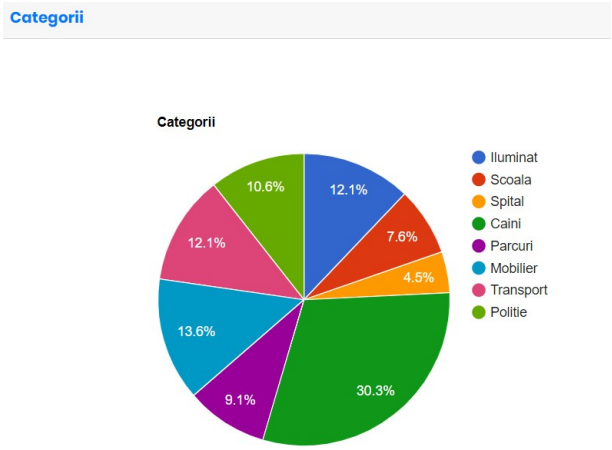

Figura 3.15: Diagrama PieChart pentru numărul de sesizări

De asemenea, am realizat și un top al utilizatorilor în funcție de numărul de sesizări trimise de fiecare.

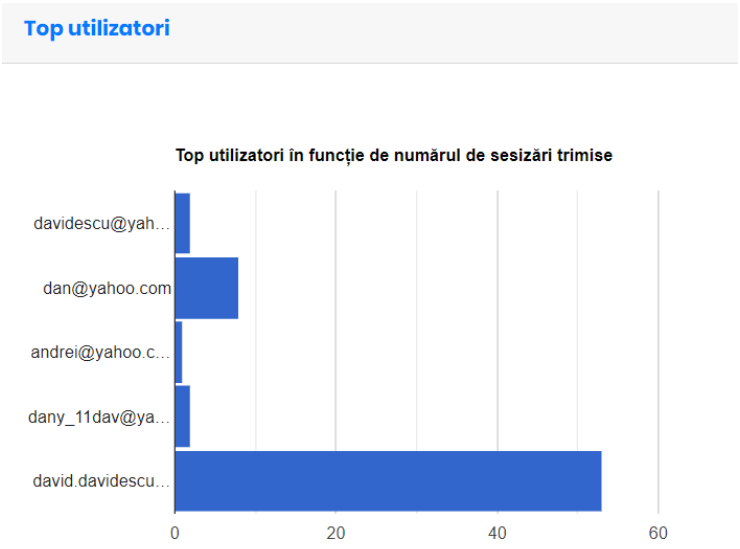

Figura 3.16: Diagramă pentru utilizatorii cei mai activi

Pentru realizarea acestor diagrame am utilizat Google Charts.

3.3. Implementarea aplicației server

Pentru implementarea aplicației server, am ales un server Node.js, implementat în framework-ul Express.js. Aplicația server se ocupă de toate funcționalitățile aplicației client Android precum și aplicației web pentru administrator.

Express reprezintă o platformă minimală și flexibilă care oferă un set de funcționalități pentru dezvoltarea facilă și rapidă de aplicații web. Acesta este unul dintre cele mai populare web framework-uri pentru Node.js.

Platforma Express poate fi instalată pentru a fi folosită împreună cu Node.js în vederea

dezvoltării de aplicații web.

```
>:$ npm install express --save
```

Instrucțiunea de mai sus reprezintă instalarea framework-ului Express. De asemenea împreună cu Express este recomandat să fie instalate și următoarele module:

- *body-parser*: folosit pentru gestiunea datelor reprezentate în format JSON, binar, text sau URL;
- *multer*: pentru procesarea resurselor de tip *multipart/form-data*.

Folosind Express, putem defini foarte simplu modul în care aplicația server răspunde cererilor trimise de aplicația client Android, precum și de pagina Web, prin folosirea rutelor. O astfel de abordare permite modularizarea aplicației.

În dezvoltarea aplicației server am folosit editorul de cod sursă Visual Studio Code. Acesta este un editor de cod care poate fi folosit cu o varietate de limbaje de programare printre care și JavaScript. Acesta permite utilizatorilor să deschidă unul sau mai multe directoare, care apoi pot fi salvate în spații de lucru pentru reutilizarea ulterioară.

Aplicația server suportă o varietate de interogări iar principalele sunt:

- */register* – această interogare are rolul de a crea un nou utilizator în cadrul aplicației. După ce utilizatorul completează numele, prenumele, adresa de email și parola, se verifică dacă adresa de email există deja, iar dacă nu utilizatorul va fi creat în cadrul bazei de date cu adresa de email furnizată și cu parola care va fi criptată cu ajutorul unei funcții de hash securizată, și anume bcrypt. Dacă mai mulți utilizatori au aceeași parolă, parola criptată va fi diferită.
- */login* – această interogare are rolul de a autentifica utilizatorul în cadrul aplicației client Android. Odată introdusă adresa de email și parola, se va verifica dacă aceasta există în baza de date, iar dacă corespund, autentificarea se realizează cu succes. Verificarea parolei criptate se realizează cu funcția *compareSync* din cadrul librăriei bcrypt. Dacă autentificarea se realizează cu succes, aplicația client Android primește un token de la server. Atunci când clientul face o cerere către server, acesta atașează în header o cheie numită *authorization* ce va conține token-ul primit la autentificare, care este întâi validat de către server.
- */users/resetPassword/* – este folosit pentru a genera o nouă parolă utilizatorului completând adresa de email. Dacă email-ul se află în baza de date, se inserează în o parolă provizorie precum data și ora resetării acesteia. Intre timp se trimite email utilizatorului un cod generat random. Acesta este valabil cinci minute de la momentul trimiterii. Pentru a calcula timpul scurs de la trimiterea parolei provizorii până la completarea acesteia de către utilizator, s-a calculat diferența dintre momentul când utilizatorul a completat codul primit și data inserată în baza de date. După ce utilizatorul a completat codul primit în email și noua parolă, se verifică codul cu parola provizorie din baza de date, care a fost criptată, iar dacă acestea coincid, se inserează noua parolă, care la rândul ei va fi criptată cu ajutorul unei funcții de hash securizată, bcrypt.
- */profile/:email* – odată ce token-ul din cadrul header-ului a fost validat, utilizatorul poate să-și vadă profilul. Aceasta este posibil printr-o funcție de agregare în baza de date, prezentată mai jos.

```
user.aggregate([{$project:{_id:0,nume:1,prenume:1,email:1,created_at:{$substr:
[“$created_at”,0,10]}}},{$match:{email:email}}]);
```

- */NrOfCategoryByStatus/:status* – această interogare este folosită pentru a putea vedea numărul de sesizări trimise pentru fiecare categorie, în funcție de statusul acesteia. Este folosită în cadrul aplicației web pentru administrator. Aceasta este posibilă printr-o funcție de agregare în baza de date, și anume în colecția „uploads.files", prezentată mai jos, unde *req.params.status* reprezintă valoarea parametrului „status" din interogare.

```
gfs.files.aggregate([{$match:{"metadata.status":req.params.status}},{$group:{_id:
{categorie:'$metadata.categorie',status:'$metadata.status'},count:{$sum:1}}}])
```

- */SchimbăStatus* – această interogare are rolul de a schimba statusul sesizării. Această funcționalitate este posibilă doar pentru aplicația web dedicată administratorului. Modificarea statusul se realizează printr-o comanda MongoDB de update, asupra colecției *uploads.files*, prezentată mai jos. Odată cu schimbarea statusului, se trimite și un email către persoana care a inițiat sesizarea cu schimbările făcute precum și cu un mesaj cu eventuale observații.

```
gfs.files.update({'metadata.IdSesizare':_id},{$set:{'metadata.status':status}})
```

- */preiaUltimeleSesizari* – această interogare este folosită pentru ca administratorul să poată vedea sesizările trimise de către toți utilizatorii în ultimele 24 de ore. Acest lucru este posibil printr-o funcție de căutare în MongoDB utilizând operatorii de căutare $lt (valoarea mai mică decât un anumit prag) și $gte (valoarea mai mare sau egală cu o valoare specificată), prezentată mai jos. Funcția *new Date()* returnează data și ora curentă.

```
gfs.files.find({"uploadDate":{'$lt':new Date(),
                              '$gte':new Date(new Date().setDate()-1))}
```

- */upload* – odată ce token-ul din cadrul header-ului a fost validat, utilizatorul are posibilitatea de a trimite o sesizare către baza de date. Pentru aceasta am folosit librăria *multer-grisfs-storage*, precum și *multer* din NodeJS. Mai întâi am creat un obiect de stocare cu următoarele configurații: adresa URL a bazei de date MongoDB unde sunt stocate colecțiile cu fișiere, precum și o funcție pentru a încărca metadatele în baza de date. *Multer* este o librărie NodeJS pentru manipularea datelor, care este folosit în principal pentru încărcarea fișierelor.
- */file/:IdSesizare* – este folosită pentru a extrage imaginile din baza de date, în funcție de id. Primul pas este să verifice dacă există id-ul dat ca parametru, apoi să citească date din baza de date GridFS. Datele sunt citite din baza de date folosind librăria *gridfs-stream*, cu ajutprul funcției *createReadStream*. Funcția *pipe()* citește datele pe măsură ce devin disponibile, și le scrie în răspunsul cererii.

3.3.1. Trimiterea unui email

Pentru a putea trimite email-uri utilizatorilor am folosită librăria *nodemailer*, ce permite trimiterea de email-uri cu conținut text, dar și cu conținut de tip HTML. Această librărie utilizează protocolul STMPS, care reprezintă o metodă de securizare a protocolului STMP. Protocolul SMTP (Simple Mail Transfer Protocol) permite aplicațiilor să transmită mesaje în

format electronic pe Internet [28].

Pentru instalarea modului Nodemailer cu NPM, se execută următoarea instrucțiune:

```
npm install --save nodemailer
```

Pentru inițializarea librăriei *nodemailer* avem nevoie de o adresă de email, în cazul meu adresa de gmail, ce va fi folosită pe post de server pentru trimiterea de email-uri. Utilizatorii primesc email-uri atunci când doresc resetarea parolei, sau în momentul în care statusul sesizării lor a fost schimbat.

3.3.2. Autorizarea pe bază de token

Autorizarea pe bază de token, este utilizată pentru a limita utilizatorii care sunt autentificați în aplicație să folosească anumite servicii web. Logica acestui proces poate fi descrisă astfel:

- Atunci când utilizatorul se autentifică cu succes, primește un token de la server ca și răspuns;
- În momentul în care utilizatorul face o cerere către server, acesta atașează în header o cheie numită *authorization* ce va conține token-ul primit la autentificare;
- După ce token-ul a fost atașat în header, acesta este validat de către server, iar dacă validarea se realizează cu succes, cererea va merge mai departe. În caz contrat, serverul returnează un răspuns în care anunță clientul că token-ul trimis de el este invalid, sau a expirat.

Pentru a genera un token de la server, am folosit librăria *jsonwebtoken.* Principalele funcții ale aceste librării sunt: *sign()* și *verify()*.

Funcția *sign()* este utilizată atunci când utilizatorul se autentifică cu succes, și folosită pentru generarea unui token. Primul parametru reprezintă datele ce vor fi criptate, în cazul meu, adresa de email cu care se autentifică utilizatorul, iar al doilea parametru reprezintă o cheie unică cu ajutorul căreia se criptează datele din primul parametru. Ultimul parametru reprezintă timpul cât este valabil token-ul în cadrul aplicației, în cazul nostru zece ore. Cheia unică am declarat-o într-un fișier separat *config.json*.

```
const token=jwt.sign(result,config.secret,{expiresIn:"10h"});
```

Funcția *verify()* este utilizată pentru a verifica dacă un token trimis de către aplicația client este valid. Aceasta primește ca parametru token-ul trimis, și cheia unică.

```
const decoded=jwt.verify(token,config.secret);
```

3.3.3. Conectarea la baza de date

Aplicația server folosește și o bază de date NoSQL, și anume MongoDB. Pentru aceasta am folosit ODM-ul[16] oficial suportat pentru platforma Node.js, Mongoose.

Mongoose este o bibliotecă ce oferă maparea obiectelor MongoDB într-o interfață familiară în cadrul platformei Node.js. Aceasta transformă datele din baza de date în obiecte JavaScript pentru a putea fi folosite în cadrul aplicației.

Pentru stocarea imaginilor în baza de date am folosit GridFS. GridFS este o specificație pentru stocarea și recuperarea fișierelor care depășesc dimensiunea de 16 MB. În loc să stocheze

16 ODM (Object Data Modeling) este un model de date bazat pe programarea orientată pe obiecte

un fișier într-un singur document, GridFS împarte fișierul în părți sau bucăți și stochează fiecare bucată separat. În mod implicit, GridFS folosește o dimensiune standard a bucăților de 255kB, cu excepția ultimei bucăți. Dacă fișierele care nu sunt mai mari decât dimensiunea bucății, acestea au doar o bucată finală. GridFS este utilă nu numai pentru stocarea fișierelor care depășesc 16MB, ci și pentru stocarea tuturor fișierelor.

GridFS utilizează două colecții pentru a stoca fișiere. O colecție stochează fișierele cu bucăți, iar cealaltă stochează fișierele cu metadate [29].

GridFS plasează cele două colecții în aceeași bază de date:

- fs.files;
- fs.chunks;

Structura colecției care stochează fișierele cu bucăți (fs.chunks) este următoarea:

```
{
     “_id”:<ObjectId>,
     “files_id”:<ObjectId>,
     “n”:<num>,
     “data”:<binary>
}
```

Colecția care stochează fișierele cu metadate (fs.files) are următoarea structură:

```
{
     “_id”:<ObjectId>,
     “length”:<num>,
     “chunkSize”:<num>,
     “uploadDate”:<timestamp>,
     “md5”:<hash>,
     “filename”:<string>,
     “contentType”:<string>,
     “metadata”: {
            “IdSesizare”:<num>,
            “email”:<string>,
            “categorie”:<string>,
            “street”:<string>,
            “descriere”:<string>,
            “status”:<string>,
     }
}
```

Mai sus se poate observa că în „metadata” au fost adăugate informații despre sesizările trimise de către utilizatori, precum: IdSesizare, email, categoria, adresa, descrierea și statusul.

Pe lângă colecția care stochează imaginile și datele despre sesizare există și o colecție (users) pentru a stoca datele utilizatorilor, necesar pentru procesul de înregistrare/autentificare după cum se poate observa mai jos. Se vor stoca numele, prenumele, emailul (care va fi unic), parola, data și ora când a fost creat contul, iar în momentul când utilizatorul dorește resetarea parolei, pentru scurt timp se va insera o parolă temporală formată dintr-un cod, precum și data când a fost creată parola temporală.

```
{
     “_id”:<ObjectId>,
     “nume”:<string>,
     “prenume”:<string>,
     “email”: {type:String,unique:true},
     “password”:<string>,
     “created_at”:<timestamp>,
     “temp_pass”:<string>,
     “temp_password_time”:<string>,
}
```

3.4. Dificultăți întâmpinate

În realizarea proiectului de diplomă am întâmpinat dificultăți începând de la proiectarea aplicației, până la implementarea și testarea acesteia.

Una din cele mai mari provocări întâmpinate în realizarea aplicației Android a fost învățarea mediului de lucru Android Studio, deoarece este unul complex, și necesită mult timp pentru înțelegerea acestuia. O altă problemă întâmpinată au fost actualizările destul de dese al mediului de lucru Android Studio.

Alegerea unui limbaj pentru dezvoltarea serverului, a fost o provocare, întrucât am dorit ca aplicația să răspundă într-un timp scurt la cereri, și să nu îl facă pe utilizator să aștepte prea mult pentru realizarea diferitelor operații.

Capitolul 4. Testarea aplicației și rezultate experimentale

Testarea unui program evidențiază prezența erorilor și nu absența lor. Este singura tehnică de validare non-funcționale, deoarece programul trebuie lansat în execuție pentru a i se analiza comportamentul [30].

Testarea reprezintă o parte crucială din procesul de dezvoltare al unei aplicații. În procesul de realizare a unor aplicații mobile sau web, unul dintre pașii cei mai importanți îl reprezintă testarea. Doar așa se poate asigura funcționarea perfectă a aplicației. Înainte de a lansa o aplicație web sau mobile, trebuie să ne asigurăm ca furnizăm o aplicație de calitate pentru utilizatorii existenți și cei viitori.

4.1. Testarea aplicației Android

Pentru folosirea aplicației client Android, aceasta trebuie instalată pe un dispozitiv ce rulează o versiune mai mare sau egală cu Android 4.0.

Opțiunile de testare pentru dispozitivele mobile sunt: testarea pe dispozitive reale sau testarea folosind emulatoare.

SDK-ul de Android pune la dispoziția dezvoltatorilor o serie de elemente cu ajutorul cărora dezvoltatorii își pot testa aplicațiile fără a avea nevoie neapărat de un dispozitiv mobile.

Printre aceste elemente se numără Android Virtual Device Manager, o unealtă cu ajutorul căreia se pot crea Android Virtual Device-uri, ce emulează dispozitive având anumite configurații hardware, și care pot fi folosite ca și instrumente de testare și rulare a aplicațiilor.

Cea mai bună metodă de testare a unei aplicații mobile este încărcarea sa pe un dispozitiv mobile. Asta presupune postarea pe un server web, sau conectarea localhost la dispozitivul pe care rulează serverul web.

4.2. Testarea aplicației web destinată administratorului

Un prim pas în testarea aplicației web, o reprezintă validarea codului HTML, pentru a ne asigura ca nu există nicio eroare majoră. Chiar dacă nu validează funcționarea perfectă a respectivei pagini web, ne ajută să eliminăm orice eroare din timp.

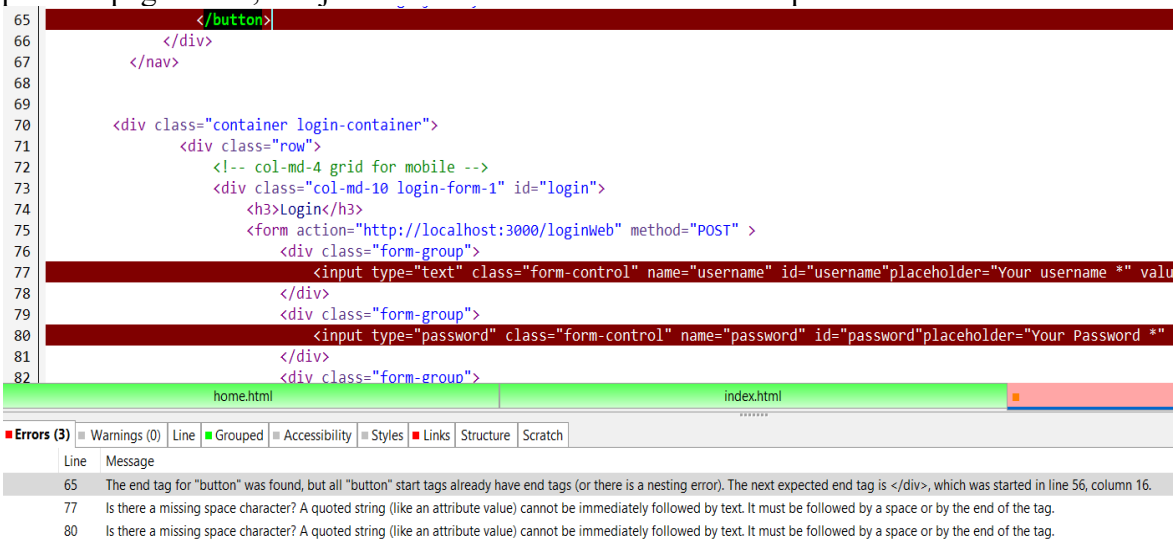

Figura 4.1: Testarea codului HTML utilizând aplicația CSS HTML Validator Pro 2019

Pentru testarea paginilor web am folosit aplicația CSS HTML Validator Pro 2019. În cadrul aplicației poți verifica codul HTML sau CSS, pentru validare.

4.3. Testarea aplicației server

Aplicația server a fost testată în paralel cu aplicația Android, utilizând extensia browser-ului Chrome, Advanced REST Client. Aceasta a fost de mare ajutor în dezvoltarea API-ului pentru că astfel am economisit timp prețios pentru procesul de scriere a codului și rezolvării erorilor.

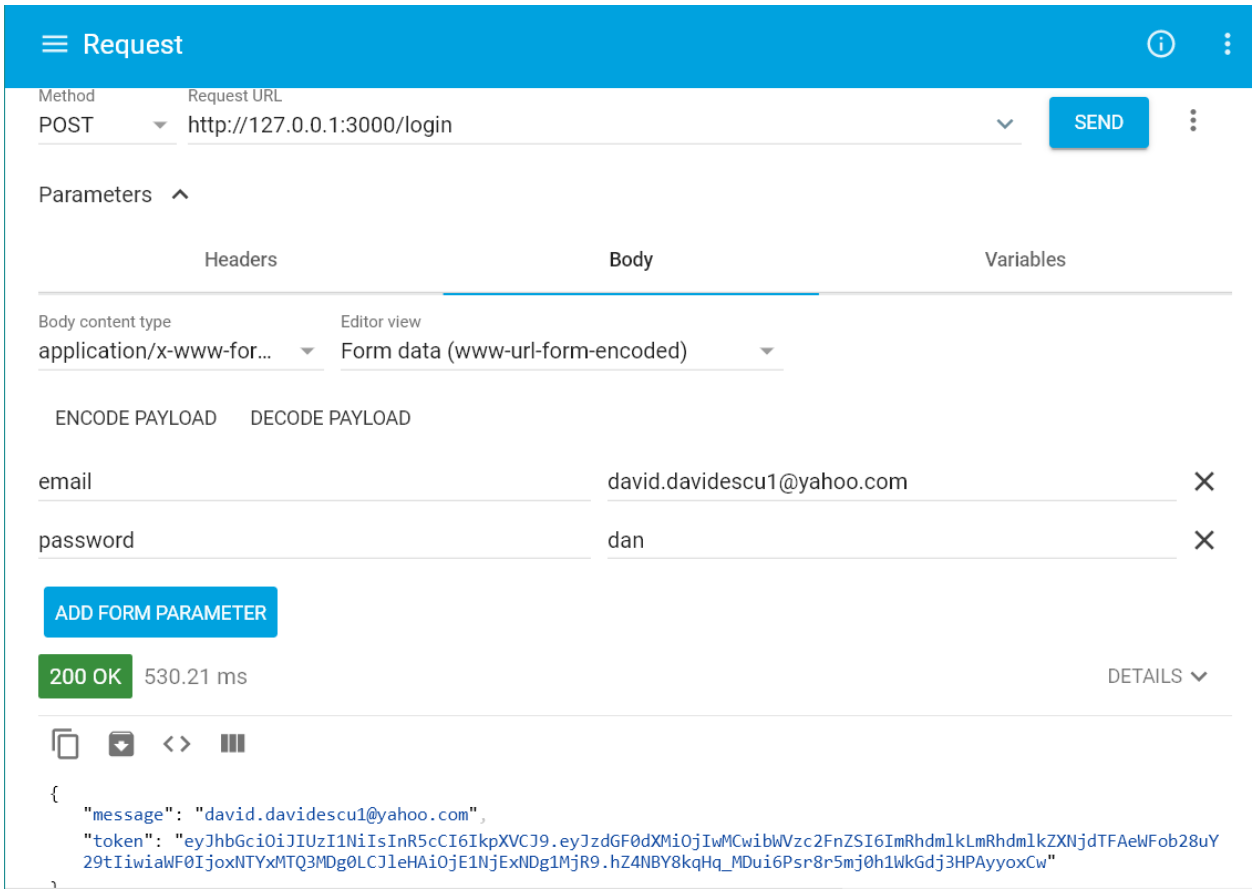

Figura 4.2: Testarea aplicației server cu ajutorul extensiei Advanced REST Client

Extensia permite utilizatorilor să lanseze cereri HTTP foarte rapid și simplu. De asemenea, extensia salvează toate cererile care au fost făcute, pentru a veni în ajutorul programatorilor, evitând situația ca aceștia să introduce de fiecare dată aceleași date de intrare.

În Figura 4.2 se poate observa modul de folosire a extensiei și răspunsul primit în urma lansării cererii. Operația care a fost testată este cea de autentificare. Se poate observa că în secțiunea Body a extensiei, am introdus numele parametrilor precum și valoarea acestora. Răspunsul este reprezentat în formatul JSON care conține un mesaj cu adresa de email folosită precum cu token-ul generat care va fi utilizat pentru continuarea aplicației.

În Figura 4.3 am testat cererea, care afișează sesizările trimise în funcție de email și statusul acesteia. Această operație a fost posibilă datorită token-ului de securitate introdus în header, cu numele Authorization și cu valoarea generată atunci când utilizatorul s-a autentificat

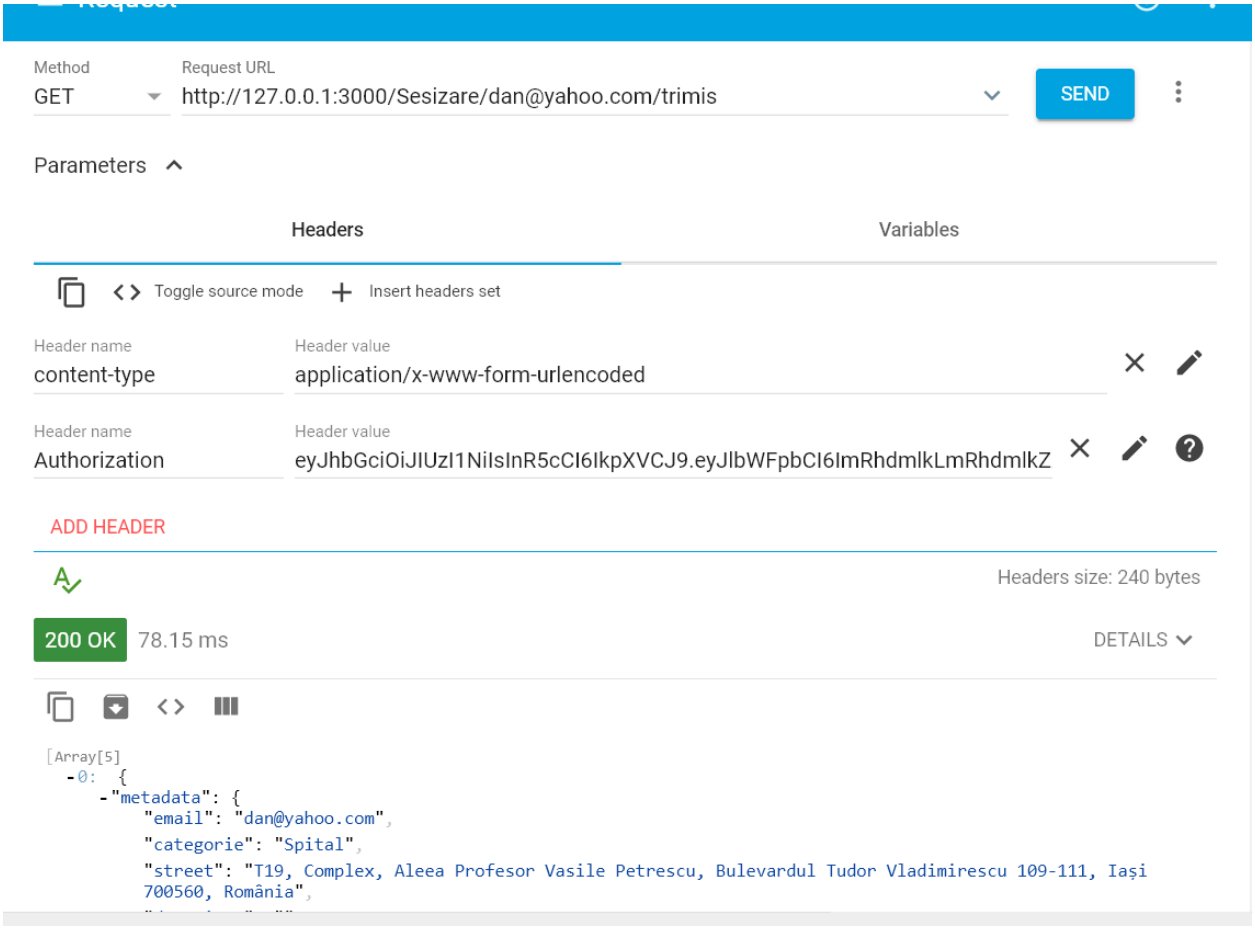

Figura 4.3: Testarea unei cereri incluzând header-ul Authorization

4.4. Rezultate experimentale

Întrucât aplicația client Android, nu folosește în mod intensiv elemente de interfață cu utilizatorul, amprenta memoriei este relativ mică după cum se poate observa în Figura 4.4.

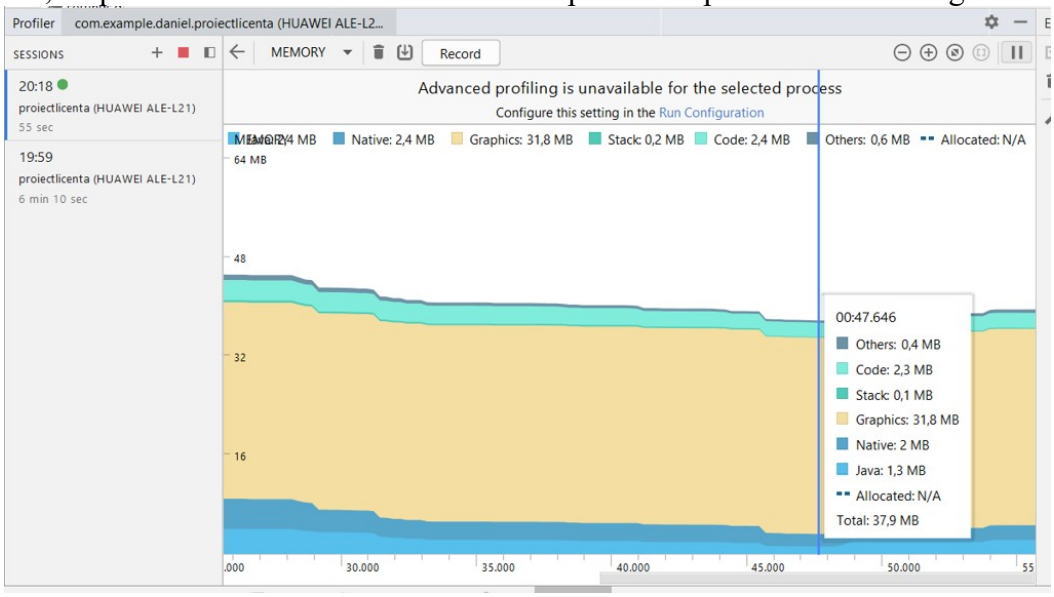

Figura 4.4: Amprenta de memoria a aplicației client Android

După cum se poate observa, aplicația mobile consumă în medie aproximativ 38MB, iar maximul amprentei de memorie la care se poate ajunge este de aproximativ 45MB, atunci când utilizatorul dorește să aleagă categoria.

Extensia browserului Google Chrome, REST Advanced Client, oferă pe lângă răspunsul serverului și timpul de răspuns pentru fiecare cerere. Astfel, am putut vizualiza performantele aplicației server. În figura de mai jos se poate observa timpul de răspuns a aplicației server pentru cererea de login este de aproximativ 330 ms.

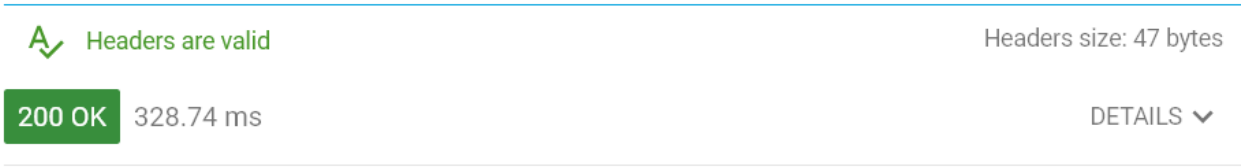

Figura 4.5: Timpul de răspuns a serverului pentru cererea de login

Concluzii

Ideea de bază de la care s-a plecat în realizarea acestui proiect este de a veni în ajutorul cetățenilor unui oraș, pentru a ușura procesul de trimitere a unei sesizări către autorități, prin realizarea unei aplicații mobile pe care rulează sistemul de operare Android. De asemenea, acest proiect vine și în sprijinul autorităților, prin realizarea unei aplicații web dedicată administratorului, care poate vizualiza sesizările trimise de către cetățeni, și să țină la curent utilizatorii cu statusul rezolvării problemei trimise.

O altă idee de la care am plecat în procesul de dezvoltare a fost realizarea unui sistem cât mai ușor de utilizat, atât pe partea de aplicație Android, cât și pe partea de aplicație web pentru administrator. Am ales o interfață clară, care să poată fi utilizată de orice categorie socială, iar pe partea de administrator funcționalitățile sunt clare și ușor de înțeles. Aplicația Android realizează trecerile de la o activitate la alta foarte rapid, fără ca utilizatorul să trebuiască să aștepte foarte mult până când primește informațiile necesare.

Consider că proiectul pe care l-am dezvoltat ar avea un impact pozitiv asupra cetățenilor, deoarece ar fi ceva nou pentru aceștia și cu adevărat folositor. De asemenea, aplicațiile de acest gen din celelalte orașe au un succes major, și sunt utilizate din ce în ce mai des de către cetățeni. Sper, ca pe viitor, aplicația să reprezinte o linie de urgență pentru problemele publice din oraș, care va contribui la dezvoltarea spiritului civic al cetățenilor și la creșterea încrederii în serviciile publice locale.

Pe viitor, intenționez dezvoltarea aplicației și pe dispozitive mobile care rulează pe sistemul de operare iOS sau Windows. De asemenea, în funcție de popularitatea aplicației, doresc implementarea și unei aplicații web, care să poată trimite sesizări. În momentul de față doar aplicația de administrator este disponibilă pe web.

În dezvoltarea ulterioară a aplicației, doresc crearea unui modul care să trimită notificări utilizatorului cu privire la problemele care sunt în apropierea zonei unde locuiește acesta. Acest lucru implică faptul ca atunci când își creează un cont, să completeze și adresa unde locuiește.

De asemenea, pe partea de administrator doresc distribuirea automată a sesizărilor către fiecare departament, în funcție de categorie.

Bibliografie

[1] Cristian Stoica, Android Smart Presentation [Online], Disponibil la adresa: https://aimas.cs.pub.ro/amicity/doc/GeorgeCristianStoica-AndroidSmartPresentationFeedback-document.pdf, Accesat: 2012.

[2] Dragos Niculescu, Introducere în programarea Android [Online], Disponibil la adresa: https://ocw.cs.pub.ro/courses/eim/laboratoare/laborator01, Accesat: 2019.

[3] *, Android Tutorial-Concepte [Online], Disponibil la adresa: http://www.itcsolutions.eu/2011/09/08/android-tutorial-concepte-activitati-si-resurse-ale-unei-aplicatii-android/, Accesat: 2017.

[4] Dragos Niculescu, Structura unei aplicații Android [Online], Disponibil la adresa: https://ocw.cs.pub.ro/courses/eim/laboratoare/laborator04, Accesat: 2019.

[5] Andrei Cojocaru, Utilizarea platformei Node.js pentru dezvoltarea aplicațiilor web [Online], Disponibil la adresa: http://aipi2015.andreirosucojocaru.ro/laboratoare/laborator06, Accesat: 2015.

[6] Mozilla Developer, Introducere in Express/Node [Online], Disponibil la adresa: https://developer.mozilla.org/ro/docs/Learn/Server-side/Express_Nodejs/Introduction, Accesat: 2019.

[7] Cristi Aflori, „Extragerea cunoștințelor din baze de date”, , 2019.

[8] Cristian Frasinaru, „Curs practic de Java”, , 2016.

[9] „Noțiuni de bază HTML”, Mozilla, Mountain View, USA.

[10] Valentin Clocotici, Scurta istorie a HTML [Online], Disponibil la adresa: https://profs.info.uaic.ro/~val/istoric.html, Accesat: 2019.

[11] Paul Pașcu, Structura documentului HTML [Online], Disponibil la adresa: http://www.seap.usv.ro/~paulp/structura.html, Accesat: 2017.

[12] Mihai Gabroveanu, Formatarea paginilor Web prin intermediul foilor de stil CSS [Online], Disponibil la adresa: http://inf.ucv.ro/~mihaiug/courses/web/slides/Curs%204%20-%20CSS.pdf, Accesat: 2018.

[13] Dan Neamțu, Ce este Bootstrap şi cum te poate ajuta în dezvoltarea de site-uri web moderne [Online], Disponibil la adresa: https://ctrl-d.ro/tutoriale/ce-este-bootstrap-si-cum-te-poate-ajuta-in-dezvoltarea-de-site-uri-web-moderne/, Accesat: 2016.

[14] „AJAX”, Mozilla, USA.

[15] „License jQuery Project”, jQueryFoundation, .

[16] John Resig, „Advancing JavaScript with Libraries”, , , pp. , 2007.

[17] Lenuța Alboaie, Sabin Buraga, „Servicii Web”, Polirom, 2006.

[18] Olteanu Ana-Cristina, „Arhitectura serviciilor web”, , , pp. 5-6, 2016.

[19] C. Airinei, Protocolul HTTP [Online], Disponibil la adresa: http://ctptc-airinei.ro/catinfo/iacSimo/html/http.html, Accesat: 2016.

[20] Dragoș Niculescu, Invocarea de Servicii Web prin Protocolul HTTP [Online], Disponibil la adresa: https://ocw.cs.pub.ro/courses/eim/laboratoare/laborator07, Accesat: 2019.

[21] Dr. Sabin Buraga, Dezvoltarea aplicațiilor WEB- servicii Web în stil REST [Online], Disponibil la adresa: https://profs.info.uaic.ro/~busaco/teach/courses/wade/presentations/web02ServiciiWeb-REST-API.pdf, Accesat: 2018.

[22] Roger Costello, Building Web Services the REST Way [Online], Disponibil la adresa: http://www.xfront.com/REST-Web-Services.html, Accesat: 2017.

[23] M. Elkstein, What is REST? [Online], Disponibil la adresa: http://rest.elkstein.org/, Accesat: 2019.

[24] Codrut Neagu, Ce sunt cookie-urile și ce fac ele? [Online], Disponibil la adresa: https://www.digitalcitizen.ro/intrebari-simple-sunt-cookie-urile-si-care-este-scopul-lor, Accesat: 2018.

[25] Square, Retrofit A type-safe HTTP client for Android and Java [Online], Disponibil la adresa: https://square.github.io/retrofit/, Accesat: 2019.

[26] Mihai Gabroveanu, Reprezentarea UML a claselor [Online], Disponibil la adresa: http://inf.ucv.ro/~mihaiug/courses/poo/slides/Curs%2005%20-%20UML.pdf, Accesat: 2005.

[27] Florin Leon, Capitol 5 Diagrame UML [Online], Disponibil la adresa: http://florinleon.byethost24.com/lab_ip.htm, Accesat: 2019.

[28] Airinei C., Protocolul SMTP [Online], Disponibil la adresa: http://ctptc-airinei.ro/catinfo/iacSimo/html/smtp.html, Accesat: 2017.

[29] MongoDB, GridFS [Online], Disponibil la adresa: https://docs.mongodb.com/manual/core/gridfs/, Accesat: 2019.

[30] Florin Leon, Faza de testare (I) [Online], Disponibil la adresa: http://florinleon.byethost24.com/Curs_IP/IP12_Testarea1.pdf, Accesat: 2019.

Anexe.

Anexa 1. Fișierul XML pentru interfața grafică aferentă procesului de autentificare/înregistrare

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:weightSum="4"
tools:context=".MainActivity">

<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:background="@color/myTextColor"
    app:srcCompat="@drawable/logo"
    tools:ignore="VectorDrawableCompat" />


<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:background="@android:color/white"
    android:orientation="vertical"
    android:padding="20dp">

    <com.rengwuxian.materialedittext.MaterialEditText
        android:id="@+id/edit_email"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:hint="Email"
        android:inputType="textEmailAddress"
        app:met_baseColor="@color/myTextColor"
        app:met_iconPadding="0dp"
        app:met_primaryColor="@color/myTextColor"
        app:met_textColorHint="@color/myTextColor">
</com.rengwuxian.materialedittext.MaterialEditText>

    <com.rengwuxian.materialedittext.MaterialEditText
        android:id="@+id/edit_password"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:hint="Parolă"
        android:inputType="textPassword"
        app:met_baseColor="@color/myTextColor"
        app:met_iconPadding="0dp"
        app:met_primaryColor="@color/myTextColor"
        app:met_textColorHint="@color/myTextColor">
</com.rengwuxian.materialedittext.MaterialEditText>

</LinearLayout>

<Button
    android:id="@+id/btn_login"
    style="@style/Base.Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="match_parent"
    android:layout_height="38dp"
    android:layout_margin="16dp"
    android:background="@drawable/gradient"
    android:text="AUTENTIFICARE"
    android:textColor="@android:color/white"
    android:textSize="20dp" />

<TextView
    android:id="@+id/txt_create_account"
    style="@style/Base.Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="match_parent"
    android:layout_height="38dp"
    android:layout_margin="16dp"
    android:background="@drawable/gradient"
    android:text="CREARE CONT"
    android:textColor="@android:color/white"
    android:textSize="20dp" />

<TextView
    android:layout_width="210dp"
    android:layout_height="38dp"
    android:layout_marginLeft="120dp"
    android:id="@+id/reseteaza_parola"
    android:text="Ai uitat parola?"
    android:textColor="@android:color/black"
    android:textSize="20dp" />


</LinearLayout>
```