MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

GuideMe – ghid virtual Android pentru muzeu bazat pe NFC

LUCRARE DE LICENŢĂ

Absolvent: **Cătălin-Ioan NARIȚA**

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

2018

Cuprins

Capitolul 1. Introducere – Contextul proiectului

În capitolul 1 se va face o mică descriere a aplicației GuideMe prin prezentarea contextului în care este plasată, motivul implementării ideii și conținutul acestei lucrări.

1.1. Contextul proiectului

Dispozitivele mobile au intrat demult în viață noastră cotidiană din simplul fapt că preferăm să avem la îndemână o sursă de informații rapidă și ușor de folosit. Astfel, interacțiunea cu dispozitivul mobil propriu a devenit o activitate ce ne ocupă o mare parte din fiecare zi.

Odată cu apariția dispozitivelor mobile de tip smartphone, aceste interacțiuni au devenit mult mai dese și mai interesante, în special din cauza faptului că dispozitivele de tip smartphone ridica gradul de inteligență artificială a dispozitivelor cu care interacționăm (ex: Google Assisstant pentru dispozitivele Android sau Siri pentru dispozitivele iOS). Această particularitate determină generarea unor multitudini de cazuri de utilizare care pot fi identificate cum ar fi comunicarea directă verbală cu un asistent personal virtual pentru a află anumite informații (temperatură de afară, prognoză meteo, informații despre anumite evenimente etc.).

În cadrul unui muzeu, un smartphone în combinație cu o aplicație potrivită ar putea oferi utilizatorului o experiență îmbogățită față de ghizii tradiționali. De obicei, la vizitarea unui muzeu se organizează vizite de grup ghidate de o persoană specializată în domeniu. Acest lucru poate duce la inconveniente cum ar fi: un vizitator dorește să viziteze doar anumite părți ale muzeului, nu tot ce este inclus în itinerariul vizitei de grup; se poate întâmpla ca unii vizitatori să nu fie atenți la explicațiile ghidului și astfel să piardă informații interesante despre anumite exponate; explicațiile ghizilor pot fi prea detaliate pentru unii vizitatori sau prea puține pentru cei ce doresc o vedere mult mai detaliată a ceea ce vizitează; experiența vizitatorilor poate fi de o calitate scăzută dacă spre exemplu se creează aglomerații în cadrul muzeului din cauza intersectării a mai multor grupuri de vizitatori.

Toate aspectele negative enumerate mai sus cât și altele care pot interveni pot fi evitate prin folosirea propriului smartphone pe post de ghid virtual al muzeului. Vizitatorii ar putea fi capabili să viziteze doar ceea ce își doresc și li se pare interesant, ar putea asculta explicațiile despre exponate oferite de mai multe ori dacă ceva li se pare foarte interesant sau poate neclar. Acest lucru poate fi realizat interactiv prin acțiuni cum ar fi scanarea de tag-uri NFC, comenzi vocale etc.

1.2. Motivația

Aplicația este dezvoltată astfel încât să ofere o experiență unică vizitatorilor. Metoda principală pentru a oferi această experiență este cea de descoperire treptată a exponatelor pe măsură ce utilizatorul scanează tag-urile NFC aferente. La început niciun utilizator nu va vedea în aplicație niciun exponat, însă acestea vor fi salvate pe măsură ce utilizatorul le vizitează.

Această experiență este îmbunătățită și prin funcționalitățile adiționale care ajută utilizatorul să se orienteze în cadrul muzeului (hartă), oferă sugestii cu privire la alte locații din apropiere ce ar putea fi de interes pentru utilizator, oferă flexibilitate pentru un număr

mai mare de utilizatori prin internaționalizarea aplicației și da posibilitatea acestora să scrie review-uri și să lase un rating, ceea ce ajută în principal instituția în cauza să se dezvolte pe baza feedback-ului primit.

1.3. Conținutul lucrării

În continuare va fi prezentat pe scurt conținutul acestei lucrări. Documentul este structurat după cum urmează:

- În capitolul 1 sunt prezentate contextul proiectului și motivația ce stă la baza dezvoltării sistemului.
- Capitolul 2 cuprinde obiectivele principale pe care le urmărește sistemul și problemele însoțite de soluțiile de rezolvare alese.
- Capitolul 3 realizează un studiu bibliografic ce se axează pe analiza utilizării tehnologiei NFC în cadrul muzeelor, analiza unor sisteme similare (în care sunt specificate argumente pro și contra pentru fiecare soluție) și analiză comparativă între aplicațiile studiate și aplicația dezvoltată.
- Capitolul 4 descrie arhitectura conceptuală a sistemului, cerințele funcționale și nonfunctionale, cazurile de utilizare identificate, urmând ca în ultima parte să fie descrise tool-urile și tehnologiile alese, însoțite de argumente.
- Capitolul 5 detaliază modul de implementare și de funcționare a sistemului. Fiecare componentă a sistemului este luată și analizată, funcționalitățile importante fiind illustrate prin diagrame de secvența.
- Capitolul 6 prezintă metodele de testare și validare a funcționalităților sistemului.
- Capitolul 7 prezintă informațiile necesare despre cerințele minime pe care un dispozitiv trebuie să le aibă pentru a fi capabil să ruleze aplicația și pașii necesari pentru a instalarea și rularea aplicației
- Capitolul 8 prezintă o concluzie asuprea a ceea ce a fost realizat și posibilele dezvoltări ulterioare ale sistemului.

.

Capitolul 2. Obiectivele Proiectului

În acest capitol vor fi prezentate obiectivele principale propuse pentru realizarea sistemului, problemele întâmpinate, soluțiile propuse și rezultatul aplicării acestor soluții.

2.1. Obiectivul principal

Obiectivul principal al sistemului este construirea unei aplicații Android care să servească drept ghid virtual pentru vizitatorii unui muzeu, capabil să ofere și alte beneficii pe lângă cele deja oferite de ghizii tradiționali. Sistemul va trebui să fie unul distribuit din cauza faptului că pe dispozitivul de tip smartphone al utilizatorului ar trebui să fie prezentă doar partea din aplicație cu care utilizatorul interacționează direct, partea de backend și de baze de date fiind prezente pe servere separate.Pe lângă obiectivul principal, sistemul ar trebui să ofere utilizatorului și servicii precum poziționarea acestuia în cadrul galeriilor, sugestii de puncte de interes bazate pe input-urile utilizatorilor, posbilitatea de a "evalua" ceea ce a vizitat și disponibilitatea aplicației în mai multe limbi.

2.2. Specificarea problemei

În acest subcapitol sunt prezentate problemele principale identificate în implementarea sistemului, pe cine/ce afectează, soluțiile posibilie și rezultatele obținute în urmă aplicării acestor soluții.

Problema principală identificată este dată de metodele de comunicare între modulele distribuite ale aplicației.

Utilizând o abordare bottom-up, intervine în primul rând problema comunicării între partea de backend și baza de date. Această problema afectează întreagul sistem și implicit utilizatorii aplicației. Soluția propusă este comunicarea prin socket-uri pusă la dispoziție de Google pentru bazele de date din Google Cloud. Astfel, o bază de date poate fi accesată remote prin folosirea acestui tip de comunicare, bineînțeles în prezența unor metode și informații adiționale de securitate pentru a restricționa accesul entităților neautorizate.

Odată ce problema comunicării părții de backend cu baza de date este rezolvată intervine problema comunicării părții de backend cu clienții (Android/web). Încă odată, problema comunicării afectează întregul sistem și implicit utilizatorii acestuia. Soluția propusă constă în untilizarea modelului arhitectural de comunicare prin servicii REST. Soluțiile de acest tip se bazează pe un set de constrângeri și proprietăți bazate pe protocolul HTTP. Serviciile REST permit sistemelor ce fac requesturi să acceseze și să manipuleze reprezentări textuale (JSON, XML etc.) a unor resurse web prin folosirea unor operații stateless uniforme și predefinite. Alte tipuri de servicii web cum ar fi spre exemplu SOAP expun propriile seturi de operații.

În cadrul sistemului dezvoltat, aplicațiile de client comunica cu serverul prin reprezentări JSON a resurselor accesate.

Capitolul 3. Studiu Bibliografic

În cazul acestui studiu bibliografic vom analiza tehnologia NFC și vom identifica sisteme similare cu cel propus, vom face o analiză a acestora, iar la sfârșit vom defini concluziile legate de abordarea aleasă pentru implementarea sistemului.

3.1. Utilizarea tehnologiei NFC în dezvoltarea aplicațiilor pentru muzeu[1]

Această tehnologie este mult mai convenabil și rapid de folosit decât scanarea unui cod sau conectarea prin Bluetooth. Tot ce un utilizator trebuie să facă este să își apropie telefonul de un chip NFC care poate fi încorporat într-un sticker/indicator sau în cazul nostru într-un exponat dintr-un muzeu. În momentul în care smartphone-ul și chip-ul NFC vin în contact, utilizatorul poate să primească informații despre exponat în diferite formate (text, video, audio, imagini etc.).

Muzeul din Londra este una dintre primele instituții publice care folosesc această tehnologie pentru a îmbogăți experiență vizitatorilor.

Chip-urile NFC ar putea fi integrate și în semnele/punctele de interes sau punctele de informație din afara muzeelor sau a altor instituții. Acest lucru ar putea oferi informații și explicații în mai multe limbi pentru a fi accesibile unui număr cât mai mare de vizitatori.

În afară de cazurile de utilizare evidente menționate mai sus, Muzeul din Londra a găsit multe alte modalități de a folosi tehnologia NFC: Au fost amplasate tag-uri NFC în jurul muzeului care le permit vizitatorilor să dea “follow”, “like” sau “check-în” instant pe site-urile de socializare cum ar fi spre exemplu Facebook, Twitter, Instagram etc. Este o metodă ieftină și de bună calitate de reclamă pentru Muzeu.

O altă metodă de atragere a utilizatorilor este de a le oferi acestora vouchere sau cupoane de reducere prin simpla apropiere a dispozitivului de tag-urile puse la dispoziție. Aceste vouchere și cupoane pot fi folosite în cadrul magazinelor cu suveniruri și cafenelelor din muzeu.

Muzeul are de asemenea tag-uri care în urma scanării lor utilizatorul poate să descarce aplicația “Soundtrack of London” care îi va oferi utiilzatorului un tur muzical al Londrei în timp ce acesta vizitează orașul.

Un alt mod în care muzeul utilizează tehnologia NFC este faptul că utilizatorul poate să doneze bani pentru muzeu și astfel să cumpere “Un an în cronologia istorică a Londrei”, ceea ce înseamnă că numele celui care a donat și un text va fi afișat timp de un an în cadrul muzeului.

[1] http://bigthink.com/disrupt-education/near-field-communication-adds-a-new-layer-to-museums

3.2. Aplicatii similare

Sistemele similare deja existente ale căror funcționalități au fost analizate cu scopul identificării celei mai potrivite soluții pentru sistemul propus sunt prezentate mai jos:

3.2.1. GNM-LuKo[2]

Este o aplicație dezvoltată de către Germanisches Nationalmuseum pentru o expoziție specială din cadrul muzeului Germanisches Nationalmuseum din Nürnberg, Germania. Aplicația se bazează pe tehnologia Bluetooth. Prin intermediul aplicației se pot selecta artefactele din apropiere după care se pot vizualiza diferite informații în legătură cu acestea. Informațiile sunt disponibile în format text și audio, fiecare dintre acestea fiind disponibile în două variante: o descriere din punctul de vedere a unei persoane specializate în domeniul respectiv, sau o simplă descriere potrivită pentru vizitatorii care nu sunt interesați de explicații specializate. Aceste informații sunt disponibile în limbile engleză și germană.

Aplicația va afișa exponatele din apropierea utilizatorului, nu doar cele de care acesta apropie dispozitivul mobil.

Este disponibilă o funcționalitate numită "Change-o-meter" care ajută utilizatorul să înțeleagă atitudinea să proprie spre schimbare. Această funcționează prin înregistrarea exponatelor pe lângă care a trecut utilizatorul. La fiecare al zecelea exponat utilizatorul va avea posibilitatea să răspundă la o întrebare în legătură cu exponatul, iar pe baza acestui răspuns aplicația îi va da utilizatorului un mic feedback despre atitudinea sa spre schimbare (dacă este orientat spre conservare sau spre schimbare). În momentul în care utilizatorul răspunde la întrebare, odată cu feedback-ul personal, acesta primește și o statistică cu media "vitezei" celorlalți utilizatori și dacă aceasta este spre conservare sau spre schimbare.

Aplicația mai oferă și o harta virtuală a galeriei în care sunt prezente locațiile tuturor artefactelor din galerie și locația utilizatorului curent (bazată pe locația artefactului curent).

Aplicația poate fi descărcată atât de pe Play Store cât și de pe AppStore.

3.2.2. Marioneta[3]

Este o aplicație dezvoltată de către CARBON by BOLD pentru muzeul " Museu da Marioneta" din Lisabona, Portugalia, muzeu dedicat interpretării și analizei istoriei păpușilor și a teatrelor de păpuși, axându-se în special pe cele de origine portugheză.

Aplicația necesită conexiune la internet pentru a avea acces la informații. Nu folosește tehnologii precum Bluetooth sau NFC.

Aplicația pune la dispoziție mai multe funcționalități cum ar fi: ghiduri propriu-zise în cadrul muzeului, informații generale despre muzeu, programul muzeului cu diferite activități și un modul pentru informații despre locațiile de interes din apropierea muzeului (restaurant, baruri, alte obiective de vizitat etc.).

În cadrul punctelor de interes, aplicația dispune și de o lista cu acestea, însă în versiunea curentă a aplicației selectarea unei locații și eventualele opțiuni în urmă accesării ei nu funcționează.

[2] https://play.google.com/store/apps/details?id=com.warptec.exhibits&hl=de
[3] http://www.museudamarioneta.pt/en

Ghidul de muzeu oferă informații cu privire la artefacte în format text sau audio în 4 limbi: Portugheză, Spaniolă, Engleză și Franceză.

Pentru a putea folosi ghidul audio, fiecare informație audio va trebui descărcată pe dispozitivul propriu (avantaj în cazul în care nu mai avem acces la internet și dorim să reascultam informațiile, dezavantaj din cauza faptului că este nevoie de un timp de așteptare de fiecare dată când dorim să aflăm o informație nouă sau în momentul în care memoria dispozitivului este limitată).

Aplicația oferă și o metodă de a atrage vizitatorii mai tineri prin implementarea unui mini-joc în cadrul galeriilor: căutarea de diferite artefacte, întrebări despre artefactele deja viziate etc.

Printre plusurile aplictiei se poate enumera faptul că pune la dispoziție informații din cadrul muzeului, chiar dacă utilizatorul nu se află fizic în cadrul acestuia. Acest lucru este benefic pentru utilizatori, însă poate fi negativ pentru instituția în sine deoarece utilizatorii care folosesc această aplicație fără să fie în muzeu s-ar putea să-și piardă interesul de a-l mai vizita odată ce au aflat informații despre exponatele din carul acestuia.

Unul dintre minusurile identificate în aplicație este layout-ul acesteia. Prin folosirea sa pe dispositive cu diferite dimensiuni și orientări ale ecranului putem observă că aplicația nu a fost dezvoltată pentru un uz general, ci doar pentru smartphone-uri cu o anumită dimensiune a ecranului. Mai exact elementele interactive din cadrul aplicației (butoane, liste) nu se încadrează uneori în dimensiunile disponibile.

Aplicația este disponibilă atât în Play Store cât și în App Store.

3.2.3. MN Coches[4]

Aplicația a fost dezvoltată de către RPGSI Business Solutions, S.A. pentru muzeul "Museu Nacional dos Coches" din Lisabona, Portugalia. Muzeul conține unele dintre cele mai bune expoziții de mijloace de transport istorice, fiind unul dintre cele mai vizitate muzee din oraș.

Aplicația oferă funcționalități cum ar fi: acces la toate artefactele din muzeu chiar dată utilizatorul nu se află în apropiere, obținerea de informații despre artefactul pe care îl observă utilizatorul doar dacă acesta este în apropiere, opțiune de a vizualiza toate artefactele în asnamblu pentru ca utilizatorul să își facă o idee despre poziția lor în cadrul muzeului și o opțiune de vizualizare a altor puncte de interes din jurul muzeului.

În cadrul activității de explorare (utilizatorul poate vedea toate exponatele) există și posibilitatea filtrarii acestora după nume pentru a ușura căutarea unui anumit exponat.

Fiecare artefact dispune de un meniu compus din 3 opțiuni: Trivia – informații detaliate despre artefact, Video – un video de prezentare a artefactului, 360 interior – o imagine 360 cu interiorul mijlocului de transport. Tot în cadrul vizualizării detaliilor unui artefact, utilizatorul are posibilitatea realizării unor acțiuni specifice rețelelor de socializare și anume de a da share pe anumite rețele la ceea ce dorește și de a da like unui anumit exponat.

Opțiunea de share preia textul descriptiv din cadrul exponatului respectiv și acesta poate fi trimis către alte persoane prin diferite alte aplicații disponibile pe dispozitivul respectiv.

[4] https://play.google.com/store/apps/details?id=com.rpgsi.coches

Unul dintre avantajele aplicației este că aduce niște plusuri conceptului de "user-friendliness" prin faptul că pune vizitatorilor la dispoziție diferite informații și animații amuzante despre exponate și totodată diferite întrebări și o parte care spune povestea unui anumit artefact.

3.2.4. Antipa Museo Tag[5]

MuseoTag este prima soluție din România bazată pe NFC dedicată muzeelor și expozițiilor. Este dedicată vizitatorilor Muzeului Național de Istorie Naturală "Grigore Antipa" din București.

Pe lângă tehnologia NFC, aplicația folosește și scanarea de coduri QR disponibile în apropierea exponatelor din muzeu.

Aplicație poate fi folosită și offline, dar utilizatorul nu va avea acces la informațiile suplimentare oferite în urmă scanării unui tag NFC sau a unui cod QR.

Aplicația dispune de un meniu cu 6 alegeri disponibile:

1. Exponate – conține 3 categorii mari de exponate, fiecare împărțite la rândul lor pe diferite criterii cum ar fi bioregiuni, specii, medii acvatice etc:
    a. Istorie naturală – conține informații despre originea vieții, evoluția speciei umane, minerale și roci, fosile și despre diferite săli interactive și ateliere din cadrul muzeului.
    b. Biodiversitatea Terrei – conține galerii împărțite pe diferite bioregiuni (netropicala, orientală, australiană etc.), fiecare conținând la rândul sau informații despre animalele ce trăiesc in acea bioregiune
    c. Biodiversitatea României – este o categorie asemănătoare cu cea prezentată mai sus, dar cu o granularitate mai ridicată datorită faptului că este prezentată o regiune mult mai restrânsă. Această categorie conține și o lista cu speciile dispărute din fauna României, dar și cu cele introduse în fauna României.
2. Informații – conține informații despre programele de vara și de iarnă, tarife, reduceri bilete și vizite ghidate.
3. Setări – conține o singură opțiune prin care utilizatorul poate alege dacă dorește să încarce pagini de pe internet în momentul în care acesta navighează în aplicație și este nevoie de acest lucru pentru a vizualiza anumite informații.
4. Harta – opțiune ce permite vizualizarea unei hărți virtuale a muzeului (3 nivele separate). Harta conține de asemenea și mici iconițe destinate să informeze utilizatorul ce categorie de exponate poate să vadă în acea zona a muzeului. Harta dispune și de funcționalitate de zoom în și zoom out, însă nu este interactivă și nu dispune de o funcționalitate care să arate automat poziția curentă a utilizatorului în interiorul muzeului, ci doar dacă acesta accesează informațiile despre un anumit exponat.
5. Despre – conține informații despre persoanele care au contribuit la dezvoltarea aplicației și serviciile folosite (ex: zxing pentru scanarea codurilor QR)

[5] http://www.antipa.ro/fii-informat
http://www.nfcexpert.ro/sample-page/

6. QR Code – ultimul modul al aplicației deschide aplicația de scanare a codului QR și îi permite utilizatorului să vizualizeze informații în plus despre anumite exponate care au în apropiere un cod QR disponibil.

Fiecare activitate din cadrul aplicației dispune și de un meniu cu 3 opțiuni: navigare spre meniu principal, setări (identic cu cel din meniul principal) și recenzie care redirecționează utilizatorul spre pagină de Google Play Store a aplicației pentru a lasă o recenzie în legătură cu experiență pe care a avut-o folosind aplicația.

Printre minusurile aplicației putem enumera:

- Unele galerii nu au nicio informație disponibilă
- Mesajele de atenționare pentru cazul în care utilizatorul nu dispune de conexiune la internet nu sunt construite corespunzător ceea ce duce la afișarea unor caractere nedorite în momentul în care sunt folosite diacritice
- Aplicația dispune de o funcționalitate care permite utilizatorului să vizualizeze poziția pe harta în cardrul muzeului a ultimului exponat vizualizat. Acest lucru poate fi realizat însă doar navigând înapoi de la activitatea de vizualizare a informațiilor la cea care afișează harta.
- Informațiile unor utilizatori spun că unele poze nu se încarcă pe anumite tipuri de telefoane.
- Fontul textului informativ este destul de mic, ceea ce poate face citirea informațiilor inconfortabilă.
- Informațiile sunt disponibile doar în limba română, cu toate că acestea sunt disponibile și în limba engleză în cadrul muzeului.

Aplicatia dispune si de unele plusuri cum ar fi:

- Design-ul interfeței utilizator este simplist, nu induce utilizatorul în eroare în momentul navigării.
- Nu necesită setări/informații suplimentare pentru utilizare (ex: cont utilizator) ceea ce o face foarte ușor de utilizat pentru scopul propus, însă acesta lipsa limitează unele funcționalități (ex: cumpărare bilete, preferințe utlizator etc.).
- Disponibilitate informații atât cu ajutorul tehnologiei NFC cât și QR code ceea ce duce la un număr mai mare de dispozitive pe care aplicația poate fi folosită cu succes.
- Disponibilitatea informațiilor cu privire la programul muzeului, la tarife și la ghidaje suplimentare în cadrul muzeului (ghid specializat, ghid audio în 3 limbi, ghid pentru grupuri).

3.3. Analiza comparativă

Tabelul de mai jos reprezintă o comparație între funtionalitatile aplicațiilor descrise mai sus și aplicația dezvoltată:

	GNM-LuKo	Marioneta	MN Coches	MuseoTag	GuideMe
Register/login funcitonality	No	No	Yes	No	Yes
Gallery map	Yes	Yes	No	Yes	Yes
Offline use	Yes	Yes	No	Yes	No
Wireless technology	Bluetooth	Uses internet connection	Bluetooth	NFC, QR code (not wireless technology)	NFC
Basic text info	Yes	Yes	Yes	Yes	Yes
Advanced text info	Yes	No	No	No	Yes
Basic audio info	Yes	Yes	No	No	Yes
Advanced audio info	Yes	No	No	No	Yes
Multi language info	Yes	Yes	Yes	No	Yes
Current location	Yes	Yes	No	Yes	Yes
Museum schedule	No	Yes	No	Yes	Yes
Surrounding interest places	No	Yes	Yes	No	Yes
Mini game	No	Yes	No	No	No
Video presentation	No	No	Yes	No	No
360 view	No	No	Yes	No	No

Funcționalitățile de înregistrare și logare sunt necesare pentru personalizarea conținutului fiecărui utilizator, de aceea sunt incluse în aplicația dezvoltată.

Harta galeriilor este o metodă de a ajuta utilizatorul să vizualizeze poziția sa în cadrul muzeului sau să îl ghideze spre galeriile de interes.

Din cauza faptului că datele sunt stocate într-o bază de date în cloud, funcționalitățile aplicației nu vor fi disponibile offline (utilizatorul va primi un mesaj de eroare dacă o conexiune la internet nu este disponibilă în momentul în care va încerca să se logheze în aplicație).

Aplicația va folosi strict tehnologia NFC din simplul fapt că în comparație cu celelalte disponibile este cea mai rapidă.

Aplicația are o serie de detalii despre fiecare exponat și galerie destinate atât unor persoane care vizitează de plăcere cât și unor persoane cu expertiză în domeniu care sunt capabile să înțeleagă o descriere mai amănunțită. Aceste detalii sunt disponibile în format text și audio.

Aplicația este disponibilă în două limbi (română și engleză), pentru a fi accesibilă unui număr mai mare de utilizatori. Harta galeriei oferă și informații cu privire la locația curentă a utilizatorului. Funcționalitatea ce include punctele de interes pune la dispoziția utilizatorului un API de recunoaștere a vocii de la Google. Acesta poate căuta anumite puncte de interes cum ar fi alte locuri de vizitat din apropiere, restaurante, baruri sau locuri de cazare. Prezentările video și 360 a artefactelor nu se potrivesc în contextul apicatiei, deci nu sunt incluse.

Capitolul 4. Analiză şi Fundamentare Teoretică

În cadrul acestui capitol sunt analizate cerințele funcționale și nonfunctionale, soluțiile software și hardware folosite pentru dezvoltarea aplicației. Capitoul este împărțit în 5 subcapitole pentru a analiza următoarele concepte: arhitectură conceptuală a sistemului, cerințe funcționale, cerințe nonfunctionale, cazuri de utilizare și tool-urile și tehnologiile folosite.

4.1. Arhitectura conceptuală

Figura 4.1 reprezintă diagramă conceptuală, care descrie la nivel înalt structura sistemului dezvoltat.

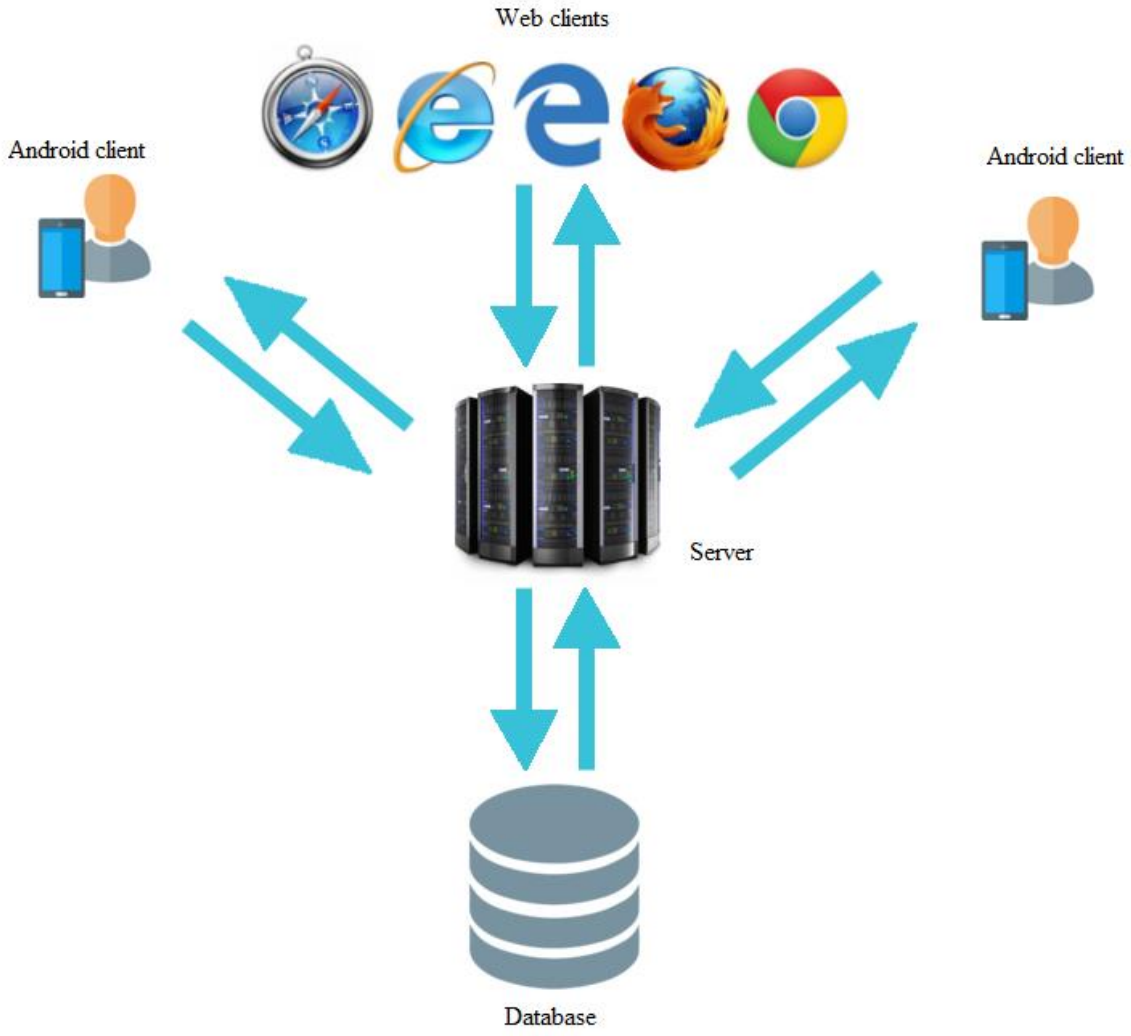


Figura 4.1 Arhitectura conceptuală a sistemului

Atât clienții web (browserele) cât și clienții Android comunica indirect cu baza de date, prin intermediul unui server. Acest lucu este necesar din cauza faptului că datele care sunt preluate din baza de date sau transmise către această trebuie mai întâi prelucrate, lucru

care ar fi un overload pentru clienți, ei fiind defapt entități care nu au alt rol decât a cere informații și de a afișa informațiile primite. Astfel, logica de bussiness(prelucrarea intermediară a datelor) este realizată pe partea de server.

4.2. Cerințe funcționale si nonfuncționale

În cadrul acestui capitol vor fi descrise cerințele funcționale și nonfunctionale ale aplicației. În cadrul unui proiect software, una dintre cele mai importante secțiuni este specificarea cerințelor. Aceste tipuri de cerințe se clasifică de obicei în două mari categorii:

1. Cerințe funcționale – funcționalitățile comportamentale ale sistemului care se traduc prin acțiunile ce pot fi realizate de către utilizator. Acestea sunt implementate prin limbajul de programare ales.
2. Cerințe nonfunctionale – cerințe ce asigura elementele de calitate ale sistemului dezvoltat (accesibilitate, performanță, integritate a datelor, interoperabilitate, robustețe, securitate, etc.).

4.2.1. Cerințe funcționale

Tabelul de mai jos reprezintă o vizualizare a cerințelor funcționale indentificate în aplicație:

	Utilizatori normali
CF1	Gestionare utiliaztori
CF1.1	Înregistrare utilizatori
CF1.2	Securizare prin criptarea parolelor
CF1.3	Logare utilizatori
CF2	Utilizare ghid virtual
CF2.1	Scanare tag NFC
CF2.2	Vizualizare listă galerii
CF2.3	Vizualizare detalii galerii (text și audio)
CF2.4	Vizualizare review-uri galerii
CF2.5	Adăugare review-uri galerii
CF2.6	Vizualizare listă artefacte descoperite în fiecare galerie
CF2.7	Vizualizare detalii artefacte (text și audio + explicații de bază sau avansate)
CF2.8	Vizualizare review-uri artefacte descoperite
CF2.9	Adăugare review-uri artefacte descoperite
CF3	Descoperire puncte de interes din apropiere (selectare manuală sau prin comanda vocala)
CF4	Schimbare limbă aplicație (română sau engleză)
CF5	Vizualizare hartă muzeu + pozitia curentă a utilizatorului în cadrul acestuia
CF6	Delogare utilizator
	Administrator
CF1	Logare administrator
CF2	Delogare admnistrator
CF3	Gestionare galerii (operatii CRUD)
CF4	Gestionare artefacte (operatii CRUD)

Cerințele funcționale reprezintă împreună soluția finală pentru implementarea unui sistem. Ele derivă în principal din analiza articolelor ce desciru sisteme similare și cazurile de utilizare identificate.

Gestionarea de utilizatori constă în principal în crearea si folosirea conturilor de utlizator. Criptarea parolelor se face pe partea de server cu ajutorul algoritmului SHA-256 (Secure Hash Algorithm pe 256 de biti, ireversibil). Procedura de comparare a parolelor se realizează în felul următor: Parolele utilizatorilor sunt stocate în baza de date sub forma unui hash generat de către algoritum SHA-256. În momentul în care un utilizator încearcă să se logheze în aplicatie, parola introdusă de el este criptată și ea cu ajutorul algoritmului SHA-256, după care este mai apoi comparată cu hash-ul din baza de date corespunzător parolei utilizatorului care încearca sa se logheze. Dacă cele 2 hash-uri sunt egale înseamnă că parola introdusă este corectă.

Prin scanarea tag-urilor NFC de pe diferite exponate se extrage din acestea un id unic ce este în prealabil asignat unui singur exponat din cadrul muzeului. În urma scanării, exponatul cu id-ul respectiv va fi adăugat în lista de artefacte descoperite de către utilizatorul respectiv.

Detaliile despre galerii și artefacte sunt disponibile atât în format text cât și în format audio, acesta din urmă folosindu-se de un API de text-to-speech de la Google.

Punctele de interes din apropiere sunt afișate într-o activitate ce include Google Maps în urma input-ului uui utilizator (alegere tipuri de locații predefinite sau prin comandă vocală).

4.2.2. Cerințe nonfuncționale

4.2.2.1. Accesibilitate

Sistemul dezvoltat reușește să respecte această cerință prin faptul că sistemul de operare Android este cel mai popular sistem de operare pentru dispozitive mobile, ceea ce face că aplicația să poate fi folosită de un număr foarte mare de utilizatori.

De asemenea, aplicația este disponibilă atâta în limba română cât și în limba engleză, ceea ce crește substanțial numărul de utilizatori ce pot folosi aplicația independent, fără a avea nevoie de ajutor în plus.

4.2.2.2. Performanță

Aplicația este dezvoltată pentru a fi folosită în cadrul unui muzeu, ceea ce înseamnă că nu va întâmpină probleme din cauza unui număr foarte mare de utilizatori care o folosesc. Totuși, dacă acest lucru se întâmplă, soluția optimă în acest caz este replicarea componentelor care acționează că un bottleneck în cazul unui număr mare de utilizatori. Aceste componente sunt baza de date și partea de backend. Prin replicarea lor, vom avea elemente redundante în sistemul nostru care pe lângă faptul că reduce riscul de a scădea performanță aplicației prin balansarea requesturilor, elimina și problema de „single point of failure”, ceea ce înseamnă că aplicația va fi în continuare funcțională dacă din motive nedorite unul dintre servere este nefuncțional. În imaginea de mai jos sunt prezentați o serie de timpi de răspuns pentru diferite request-uri făcute de pe clientul android.

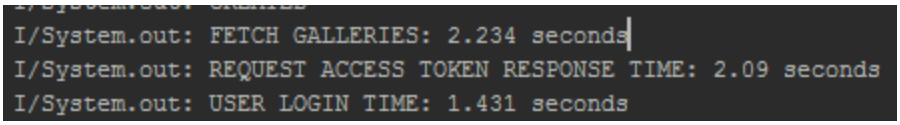


Figura 4.2 Timpi de raspuns

4.2.2.3. Integritate a datelor

Această cerință este satisfăcută de sistem prin folosirea de tranzacții și pessimistic locking. Aceste operații sunt realizate implicit prin folosirea ORM-ului Hibernate. Tranzacțiile reprezintă operații atomice. O operație atomică reprezintă o serie de acțiuni care sunt strâns legate una de alta. În cazul în care oricare dintre aceste acțiuni nu se încheie cu succes, operațiile ce ar urma după ea nu se mai execută, iar cele executate înainte sunt șterse (rollback). Acest lucru asigura că în urmă unei operații asupra bazei de date ea trece întotdeauna dintr-o stare consistentă în alta.

Pessimistic locking (tipul de locking default implementat de Hibernate) presupune că tranzacțiile concurente vor intra în conflict unele cu altele, așa că în momentul în care o resursă este citită ea va fi blocată până când aplicația nu mai folosește această resursă. Acest lucru se întâmplă și în cazul operațiilor de scriere.

4.2.2.4. Interoperabilitate

Acest tip de cerință face referire la faptul că părțile componente ale aplicației pot comunica între ele prin expunerea unor interfețe sau protocoale de comunicare, fără a fi nevoie ca fiecare componentă să știe cum este implementată componenta cu care dorește să comunice. În cazul aplicației dezvoltate componentele comunica între ele prin servicii REST (comunicarea aplicației web pentru administrator și a aplicației mobile cu partea de backend) și prin MySQL (comunicarea părții de server cu baza de date).

4.2.2.5. Robustețe

În cadrul sistemelor software, robustețea face referire a capacitatea sistemului de a trata erorile sau inputurile invalide în timpul funcționarii, fără că acesta să se oprească din funcționare când un astfel de eveniment are loc. Aplicația reușește să îndeplinească acestă cerință prin faptul că posibilele evenimente de acest tip sunt tratate (exception handling) și un feedback este transmis utilizatorului pentru ca acesta să știe ce s-a întâmplat.

4.2.2.6. Securitate

Securitatea este cea mai importantă cerință de satisfăcut în momentul în care sistemul în cauza lucrează cu dată cu caracter personal ale utilizatorilor sau dacă accesează resurse confidențiale.

În cadrul aplicației dezvoltate securitatea este asigurată prin mai multe implementări.

În primul rând, baza de date din Google Cloud poate fi accesată doar dacă entitatea ce încearcă să o acceseze știe numele instanței, numele și parola de utilizator și ip-ul prin care această poate fi accesată. Instanțele din Google Cloud pot fi configurate astfel încât să poate fi accesate doar de anumite ip-uri, ceea ce aduce un plus de securitate deasupra faptului că sunt necesare credentialele de acces.

Partea de backend este securizată folosind implementarea de Spring a protocolului Oauth2. Acest lucru securizează endpoint-urile REST prin două tipuri de credentiale: Credentiale de aplicație, lucru ce limitează aplicațiile care au acces la aceste resurse; credentiale de utilizator care limitează și mai mult accesul la aceste resurse.

În cadrul aplicațiilor web pentru administrator și Android pentru utilizatori normali, ele sunt securizate prin faptul că fiecare utilizator are un rol anume în baza de date (ROLE_ADMIN sau ROLE_USER), ceea ce înseamnă că un utilizator normal nu poate folosi aplicația web de management în scopuri nedorite.

4.3. Cazuri de utilizare

În această secțiune sunt prezentate cazurile de utilizare identificate în aplicație.

Primul set face referire la cazurile de utilizare ale user-ului normal, urmând ca mai apoi să fie tratate și cazurile de utilizare specifice pentru utilizatorul de tip admin.

4.3.1. Cazuri de utilizare user normal

În figura 4.2 sunt reprezentate fiecare dintre aceste cazuri de utilizare.

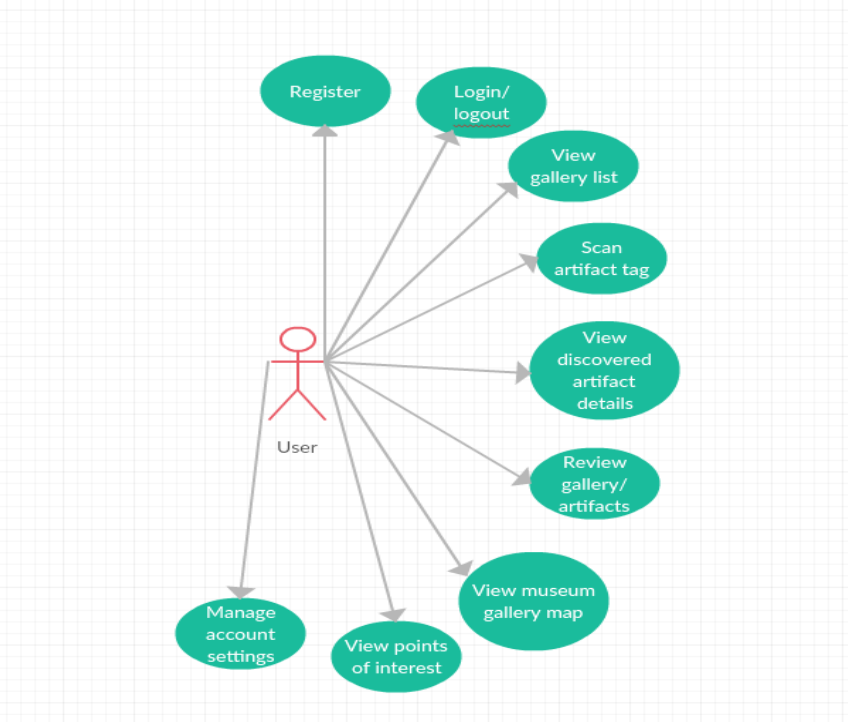


Figura 4.3 Cazurile de utilizare utilizator normal

Actorul principal în acest set de cazuri de utilizare este utilizatorul normal.

4.3.1.1. Cazul de utilizare: Register

Descriere: Înregistrarea unui nou utilizator al aplicației
Actor principal: Utilizatorul aplicației
Precondiții:

- Sistem funcțional
- Utilizatorul nu este înregistrat

Postcondiții: Utilizatorul este înregistrat

Scenariu principal de succes:

1. Utilizatorul pornește aplicația
2. Selectează opțiunea de creare de cont nou
3. Introduce datele de identificare necesare
4. Apasă butonul de înregistrare
5. Utilizatorul este înregistrat cu succes și redirecționat către pagina principală

Scenariu alternativ:

1. Utilizatorul pornește aplicația
2. Selectează opțiunea de creare de cont nou
3. Introduce datele de identificare necesare
4. Apasă butonul de ținregistrare
5. Username-ul sau email-ul sunt deja folosite și înregistrarea nu poate fi completată
6. Utilizatorul are posibilitatea de a modifica datele inițiale pentru a fi valide

4.3.1.2. Cazul de utilizare:Scan artifact tag

Descriere: Scanarea tag-ului NFC al unui artefact și redirecționarea către lista de artefacte descoperite

Actor principal: Utilizatorul aplicației

Precondiții:

- Sistem funcțional
- Utilizatorul este logat în aplicație
- Dispozitivul dispune de funcționalitate NFC

Postcondiții: Utilizatorul este redirectionat către lista de artefacte și le poate vizualiza pe cele descoperite

Scenariu principal de succes:

1. Utilizatorul se loghează in aplicație
2. Apropie telefonul de tag-ul unui artefact
3. Un nou artefact este adăugat în lista de artefacte descoperite
4. Utilizatorul este redirecționat către galeria în care se află artefactul pe care tocmai l-a descoperit

Scenariu alternativ:

1. Utilizatorul se loghează in aplicatie
2. Apropie telefonul de tag-ul unui artefact
3. Artefactul a fot deja descoperit de catre utilizator
4. Utilizatorul este doar redirecționat către lista de artefacte in care se afla artefactul respectiv

4.3.1.3. Cazul de utilizare: Căutare puncte de interes

Descriere: Lansarea în execuție a activitătii de puncte de interes și căutarea acestora pe baze diferitelor metode de input,

Actor principal: Utilizatorul aplicatiei

Precondiții:

- Sistem funcțional
- Utilizator logat în aplicație
- Permisiune pentru citirea locației acordată

Scenariu principal de succes:

1. Utilizatorul accesează activitatea de puncte de interes
2. Apasa butonul de înregistrare sunet
3. Rosteste fraza dorita
4. Se lanseaza automat aplicația Google Maps și foloseste fraza rostită de utilizator ca si search query
5. Sunt returnate detalii despre ceea ce a căutat utilizatorul

Scenariu alternativ:

1. Utilizatorul alege un punct de interes predefinit din dropdown-ul disponibil
2. Selectează raza de căutare prin modificarea slider-ului inclus în activitate
3. Harta inclusă in aplicatie afișează punctele de interes cu tipul dorit din raza de căutare selectată.

4.3.1.4. Cazul de utilizare: Adăugare review galerie

Descriere: Accesarea activității de gallery review, selectarea unui rating și adăugarea unui comentariu.

Actor principal: Utilizatorul aplicației

Preconditii:

- Sistem funcțional
- Utilizator logat în aplicatie
- Utilizatorul se afla pe activitatea de gallery details

Scenariu principal de succes:

1. Utilizatorul apasă butonul de adăugare review din dreapta sus din activitatea de gallery details
2. Introduce un comentariu și selectează un rating pentru galeria respectivă
3. Apasă butonul Submit pentru a adăuga noul review
4. Toți utilizatorii sunt capabili să vadă review-ul respectiv

Scenariu alternativ:

1. Utilizatorul apasă butonul de adăugare review
2. Introduce datele necesare
3. Conexiunea la internet este indisponibilă, astfel review-ul nou nu va fi adăugat.

4.3.2. Cazuri de utilizare administrator

În figura 4.5 sunt prezentate cazurile de utilizare specifice utilizatorului de tip administrator.

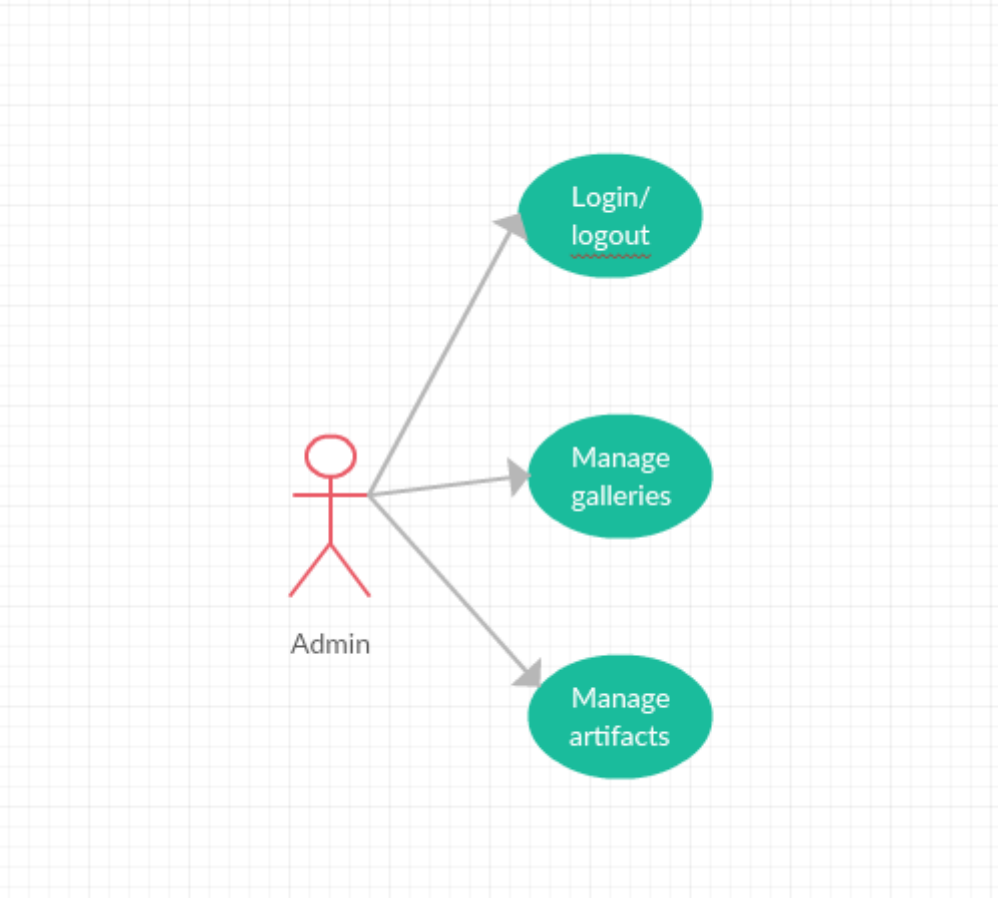


Figura 4.4 Cazurile de utilizare pentru utilizatorul de tip administrator

4.3.2.1. Cazul de utilizare: Manage galleries

Descriere: Administratorul are posibilitatea de a efectua operații CRUD asupra galeriilor

Actor principal: Administratorul

Preconditii:

- Sistem funcțional
- Administratorul este logat in aplicația web specifică task-urilor de administrare

Postconditii: Dacă administratorul a efectuat operații asupra galeriilor, starea curentă a galeriilor va corespunde cu schimbările aplicate

Scenariu principal de succes:

1. Administratorul se loghează in aplicație
2. Accesează lista cu galeriile
3. Adauga o nouă galerie

Scenariu alternativ:

1. Administratorul se loghează in aplicatie
2. Accesează lista cu galeriile
3. Serviciu indisponibil datorită problemelor de server, moment in care o pagină de eroare este afisată.

4.4. Tool-uri și tehnologii

4.4.1. Tool-uri

În acest subcapitol vor fi prezentate toate tool-urile folosite în dezvoltarea aplicației (scopul utilizării lor, facilități puse la dispoziție).

4.4.1.1. IntelliJ IDEA IDE[6]

IntelliJ IDEA este un mediu integrat de dezvoltare dedicat dezvoltării de aplicații software. Este dezvoltat de către JetBrains și este disponibil sub două versiuni:

1. Community edition – varianta open source care poate fi folosit în scopuri comerciale, dar pune la dispoziție o serie limitată de functionalitati
2. Ultimate edition – varianta ce necesită o licență plătită (sau obținută prin alte mijloace cum ar fi licență dedicată studenților, oferită de către angajator la locul de muncă etc.). Varianta această poate fi de asemenea folosită în scopuri comerciale și pune la dispoziția utilizatorului toate funcționalitățile disponibile.

De la versiunea 2017.1, IDE-ul include suport pentru Java 9, design Android, Play 2.0 și Scala.

Tool-ul pune la dispoziția utilizatorului un feature puternic de code assistance care cuprinde sugestii de completare a codului, navigare în cod ceea ce permite saltul ușor la clase sau declarații, refactorizarea codului și opțiuni pentru corectarea anumitor inconsistente prin oferirea de sugestii bazate pe analiză continuă a codului proiectului/aplicației dezvoltate.

Permite de asemenea integrarea unor build și dependency management tools cum ar fi Maven, Gradle, Grunt, Bower și SBT. Are de asemenea suport pentru sisteme de versionare a codului cum ar fi Gît, SVN, Mercurial și Perforce.

O altă funcționalitate pusă la dispoziție este posibilitatea de a accesa baze de date direct din IDE.

Una dintre funcționalitățile majore oferite de versiunea Ultimate este suportul pentru application servers cum ar fi Glassfish, Jboss, Jetty, Tomcat etc.

[6] https://www.jetbrains.com/idea/features

Am ales acest mediu de dezvoltare în detrimentul Eclipse și a altor tool-uri disponibile deoarece consider că este mult mai user friendly, iar per total este o aplicație mult mai robustă în comparație cu celelalte.

4.4.1.2. Android Studio[7]

Android Studio este IDE-ul oficial Google pentru dezvoltarea aplicațiilor mobile pentru sistemul de operare Android. Este dezvoltat pe baza IntelliJ IDEA. Este disponibil pentru platformele Windows, macOS și pentru sistemele bazate pe Linux. Vine că un înlocuitor pentru Eclipse Android Development Tools.

Tool-ul pune la dispozitie functionalitati precum:

1. Suport pentru build și managementul dependintelor bazat pe Gradle
2. Facilități de refactorizare și quickfixes specifice pentru Android
3. Lint tools pentru a analiză performanță, erori în cod, compatibilități între versiuni etc.
4. Template-uri pentru componente des folosite (activități de login, register, application drawers etc)
5. Editor pentru layout-ul aplicației ce permite utilizatorului să folosească componente drag-and-drop, opțiuni să vizualizeze layout-ul pe mai multe tipuri de disaply-uri etc.
6. Suport pentru dezvoltarea aplicațiilor Android dedicate dispozitivelor de tip wearable (smartwatches etc.)
7. Suport pentru integrarea cu serviciul de messaging de la Google (Firebase Cloud Messaging)
8. Emulator virtual Android pentru a nu fie nevoie neapărată de un dispozitiv fizic pentru testarea aplicației dezvoltate

Android Studio suportă toate limbajele de programare de la IntelliJ și PyCharm cum ar fi Python și Kotlin, iar Android 3.0 suportă facilitățile oferite de Java 7 și un subset din facilitățile oferite de Java 8.

Există multe alte tool-uri ce suportă dezvoltarea aplicațiilor Android (AIDE – Android IDE, Eclipse, Intellij IDEA, Xamarin etc,), însă am ales Android Studio din simplul fapt că este destinat strict pentru dezvoltarea aplicațiilor Android și astfel beneficiază de suport doar pentru acest tip de aplicații direct de la Google.

4.4.1.3. Webstorm[8]

Este un IDE puternic și inteliget dezvoltat de către JetBrains și oferă cele mai bune facilități de code assistance pentru JavaScript, HTML și CSS și o gama largă de tehnologii web moderne. Este capabil să dezvolte atât aplicații client cât și aplicații server folosind Node.js.

Tool-ul ofera code assistance pentru framework-uri bazate pe JavaScript cum ar fi AngularJS, Angular, React, Vue.js, Meteor. Ofera de asemenea suport si pentru dezvoltarea aplicatiilor mobile prin React Native, PhoneGap, Cordova si Ionic.

[7] https://en.wikipedia.org/wiki/Android_Studio
[8] https://www.jetbrains.com/webstorm/features

Webstorm pune la dispoziție un debugger pentru codul client-side și aplicațiile Node.js.

O altă facilitate este framework-uri de testare a aplicațiilor JavaScript cum ar fi Karma, Mocha, Jest și Protractor.

Pentru a ușura muncă utilizatorului pentru diferite configurări și dependințe, tool-ul oferă suport și pentru build tools cum ar fi Grunt, Gulp sau npm. Se pot crea scripturi pentru fiecare dintre aceste build tools pentru a încapsula anumite task-uri ce pot fi rulate la un simplu dublu-click pe numele scriptului.

IDE-ul oferă și tool-uri pentru code quality, dar poate rula și tool-uri externe cum ar fi ESLint, JSCS, TSLint, Stylelint, JSHint sau JSLint.

Oferă de asemnea template-uri de aplicații gata implementate, dar și o integrare cu Yeoman ce oferă o gama și mai largă de generatoare de proiecte.

O altă facilitate a tool-ului este cea de suport pentru sisteme de versionare (Gît, SVN, Mercurial și Perforce).

Datorită faptului că este un tool din suită JetBrains, prezintă aceleași beneficii că și Intellij IDEA, lucru care m-a determinat să îl folosesc pentru dezvoltarea părții de web a aplicației. Un alt motiv este faptul că tool-ul este destinat dezvoltării aplicațiilor web, punând astfel la dispoziție o mulțime de funcționalități cum ar fi suportul pentru limbajele de dezvoltare web (javaScript, TypeScript etc.).

4.4.1.4. Postman[9]

Postman este cel mai folosit tool pentru testarea endpoint-urilor REST din lume. Are o interfață user-friendly pentru transmitere de requesturi, salvare de răspunsuri, adăugare de teste și creare de workflow-uri.

Pune la dispoziție diferite facitlități precum:

1. Istoric requesturi
2. Variabile – transmise în cadrul unui request pentru a adaugă date suplimentare requestului
3. Environments – oferă capabilitatea de a crea mai multe medii de dezvoltare
4. Teste și scripturi pre-request
5. Colecții și descrieri de requesturi

Colecțiile de requesturi permit utilizatorilor să construiască suite de teste, documentații, servere de test (mock servers).

Suportă testare automată prin tool-ul încorporat „Collection Runner”. Acesta permite rularea tuturor requesturilor dintr-o colecție, iar fiecare request este salvat într-un log pentru debugging.

O altă facilitate este monitorizarea uni API. Această oferă informații despre uptime-ul API-ului, responsivitate și corectitudine. Rapoartele monitorizării sunt afișate sub formă grafică.

O alternativă la acest tool este Swagger, care este folosit în principal în dezvoltarea aplicațiilor enterprise, ceea ce este prea mult în cazul aplicației dezvoltate. Pe lângă acest lucru, Postman nu necesită setări specifice pentru a putea fi folosit. Trebuie doar instalat și utilizatorul își poate testa endpoint-urile de servicii. Totodată, dispune de o multitudine de

[9] https://www.getpostman.com/postman

alte funcționalități care facilitează testarea și mai mult cum ar fi suitele de teste descrise mai sus.

4.4.1.5. Google Cloud SQL[10][11]

Este un serviciu de baze de date oferit de Google care ușurează setarea, mentenanță, managementul și administrarea bazelor de date relaționale aflate în google Cloud.

Oferă suport pentru bazele de date ce folosesc MySQL sau PostgreSQL.

Serverele sunt disponibile în mai multe locații geografice cum ar fi US, EU sau Asia, deci timpul de acces la date poate fi minimizat indiferent de locația fizică a aplicației ce dorește acces la resursele respective.

Fiind un serviciu cloud, acesta oferă replicarea datelor pe mai multe zone pentru a evita single point of failure.

Oferă de asemenea suport și pentru diferiți conectori MySQL pentru a face posibilă conectarea remote din aplicații server.

Un utilizator poate folosi Cloud SQL pentru MySQL cu aplicații scrise în App Engine cu limbajele de programare Java, Python, PHP, Node.js, Go și Ruby sau cu aplicații externe care folosesc protocolul MySQL standard.

Spre deosebire de instanțele MySQL locale, CloudSQL încă nu suportă toate funcționalitățile (funcții definite de utilizator, privilegii SUPER, statement-uri SELECT care să salveze datele în fișiere output, instalare de plugin-uri etc.).

Am ales să folosesc o soluție cloud pentru baza de date deoarece sistemul dezvoltat este unul distribuit, iar acest lucru elimina problema majoră de Single Point of Failure.

4.4.1.6. Utilizarea Heroku în deployment-ul aplicațiilor[12]

Heroku reprezintă o soluție cloud de tip „Platform aș a Service” (PaaS) și este folosit pentru deployment-ul aplicațiilor web sau a aplicațiilor ce expun diferite servicii pentru acces la resurse.

La început a fost dezvoltat pentru a suportă doar limbajul de programare Ruby, dar în momentul actual suportă Java, Node.js, Scala, Closure, Python, PHP și Go.

În momentul actual Heroku suportă baze de date bazate pe PostgreSQL și Redis.

Aplicațiile ce rulează pe Heroku au de obicei un domentiu unit sub formă „applicationname.herokuapp.com” folosit să trimită requesturile HTTP la containerul aplicației (numit „dyno”). Aceste containere sunt prezente pe mai multe servere, ceea ce asigura redundanța și asfel uptime-ul aplicației este maximizat chiar dacă unele servere cedează.

Platforma dispune de servere de Gît proprii pentru a stoca codul sursă al aplicațiilor și pentru versionarea aplicațiilor.

Toate serviciile oferite de Heroku sunt bazate pe AWS (Amazon Web Services).

Adăugarea și deployment-ul unei aplicații pe platforma Heroku se poate face în câțiva pași simpli din-un terminal. În momentul în care noi funcționalități sunt adăugate în aplicație (este executată o comandă de push pe repository-ul de Gît oferit de heroku), un

[10] https://cloud.google.com/sql/docs/mysql/connect-external-app

[11] https://cloud.google.com/sql/docs/features

[12] https://www.heroku.com/platform

task de build al aplicației este automat pornit, iar dacă acesta se încheie cu succes, aplicația respectivă este restartata pentru că noile funcționalități să fie disponibile.

Am ales această platforma cloud în detrimentul altora cum ar fi AWS(Amazon Web Services), Microsoft Azure etc. Deoarece consider că procesul de deploy este mult mai ușor decât în cazul celorlalte, iar beneficiile pentru planurile ce nu necesită plata sunt destule pentru aplicația dezvoltată.

4.4.2. Tehnologii

În cadrul acestul subcapitol vor fi prezentate toate tehnologiile și framework-urile alese pentru dezvoltarea aplicației (motivul alegerii, avantaje, dezavantaje).

4.4.2.1. NFC[13]

NFC(Near Field Communication) reprezintă un set de protocoale de comunicare care permite comunicarea între două dispozitive electronice (unul dintre ele fiind de obicei mobil, ex: smartphone) în momentul în care acestea se află la o distanță de până la 10 cm unul de celălalt.

Acest tip de comunicare elimina necesitatea de a parcurge pași multipli pentru a realiza o conexiune.

NFC menține interoperabilitatea între diferite metode de comunicare wireless cum ar fi Bluetooth sau alte standarde.

Aceste circuite funcționează pe o anumită frecvența. Acest lucru permite un transfer de energie maxim prin aer. În momentul în care tag-ul este alimentat, el poate să transmită date pe frecvența de 13.56 MHz la viteze de 106, 212 sau 424 Kbps.

Tag-urile comunica folosind standardele wireless ISO 14443 A și B, standarde internaționale pentru smart-card-uri contactless, folosite în multe sisteme de trasport public. Din acest motiv dispozitivele NFC pot fi folosite cu tehnologiile contactless existente. Ele dispozitive pasive, ceea ce înseamnă că pot să funcționeze fără o sursă de energie proprie, dar se bazează pe un dispozitiv activ să fie în apropiere că să îl activeze.

Pentru a alimenta aceste dispozitive, se folosește inducția electromagnetica pentru a crea curent în dispozitivul pasiv. Dispozitivele active (ex: smartphne) sunt responsabile pentru generarea câmpului magnetic necesar. Acest lucru este realizat cu o simplă bobină, care produce câmpuri magnetice. Puterea câmpului magnetic poate fi ajustat prin varierea numărului de bucle din bobină sau crestearea intensității curentului care trece prin bobină. Totuși, mai mult curent necesită mai multă energie, ceea ce este un inconvenient pentru dispozitivele alimentate de o baterie. Din această cauza NFC operează la distanțe de câțiva centimetri.

Există mai multe tipuri de tag-uri, fiecare oferind diferite nivele de capacitate de stocare și viteze de transfer. Tipurile 1 și 2 vin cu capacități de stocare între 48 bytes și 2 kilobytes și pot transmite informația la viteze de 106 kbit/s. Deși ni se pare destul de puțîn, capacitatea este destulă pentru bucăți d einformatii mici cum ar fi un url. Aceste tag-uri

[13] https://www.Androidauthority.com/nfc-tags-explained-271872
http://nearfieldcommunication.org/about-nfc.html

sunt concepute pentru a fi eficiente din punct de vedere al costului și de asemenea pot fi refolosite dacă dorim să schimbăm datele stocate in ele.

Tipul 3 folosește un alt standard și poate transferă date la viteze de 212 kbit/s. Acestea sunt de obicei folosite pentru aplicații mai complicare, dar sunt readonly. Tipul 4 este de asemenea read-only, dar are o capacitate mai mare (pna ăla 32 kbytes) și viteze de comunicare între 106 și 424 kbit/s. Tipul 4 este compatibil cu amandoua standardele ISO14443 A și B.

Cel mai puternic argument în favoarea NFC este că sunt foarte ieftine de fabricat și întreținut și totodată să poată fi foloste pentru un număr mare de aplicații.

Cea mai importantă tema ce poate fi abordată în cadrul comunicației prin NFC este cea de securitate[14]. Posibilele atacuri de securitate sunt: Eavesdropping, Coruperea și manipularea datelot, Atacuri de interceptare, Furt.

Eavesdropping:

Acest tip de atac de securitate are loc în momentul în care o tranzacctie NFC este "ascultată". Individul/dispozitivul care face acest lucru nu trebuie să recepționeze fiecare semnal pentru a intră în posesia unor informații personale. Există două metode prin care acest tip de atac poate fi prevenit:

1. Prima se bazează pe distanță maximă la care se poate comunica între dispozitivele NFC.
2. A două metodă constă în canalele securizate. Când un astfel de canal este creat, informația este criptată și doar dispozitivele autorizate o pot decripta.

Coruperea și manipularea datelor:

Acest tip de atac are loc în momentul în care un criminal manipulează datele transmise către un receptor, acestea fiind imposibil de folosit ulterior în scopul în care au fost transmise. Acest atac se poate evita fie prin folosirea canalelor securizate, fie prin folosirea unor altor dispozitive NFC care "ascultă" la acest tipuri de atacuri și le previn înainte să afecteze datele transmise.

Atacuri de interceptare:

În cadrul acestui tip de atac o persona are rolul de itnermediar între două dispozitive NFC care interceptează și alterează informația transmisă. Acest tip de atât este dificil și mai puțin comun. Pentru prevenire, dispozitivele ce comunica ar trebui să folosească o comunicație de tip pasivă-activă (unul dintre ele transmite iar celălalt primește) și nu pere-to-peer.

Furt:

Nicun fel de criptare a datelor nu poate proteja un utilizator de furt. Dacă un smartphone/card contactless este furat, teoretic hoțul ar putea să îl folosească pentru a face tranzacții cu banii disponibili în contul utilizatorului respectiv. Din acest motiv, utilizatorii de smartphone ar trebui să fie conștienți și să își securizeze dispozitivele în vederea minimizării acestui risc.

O tehnologie similară care ar putea de asemenea să fie folosită pentru acest tip de aplicație este cea Bluetooth. Am ales totuși NFC[15] pentru acest proiect din următoarele motive:

[14] http://nearfieldcommunication.org/nfc-security.html
[15] http://nearfieldcommunication.org/bluetooth.html

- Tehnologia Bluetooth este mult mai costisitoare din punct de vedere al consumului de energie decât NFC (există însă BLE – Bluetooth Low energy care reușește să aibă un consum de energie mai mic decât la NFC)
- Distanță la care două dispositive NFC pot comunica este mult mai mică decât în cazul dispozitivelor Bluetooth, ceea ce este un avantaj destul de mare în momentul în care dorim să eliminăm interferențele de semnale din imediată apropiere (ex: dacă am fi folosit Bluetooth nu puteam garanta că informațiile primite sunt de la artefactul dorit)
- Cel mai mare avantaj pe care thenologie NFC il aduce este acela ca spre deosebire de Bluetooth, dispozitivele NFC se pot conecta intre ele aproape instant (este nevoie doar sa fie apropiate unul de celalalt), spre deosebire de Bluetooth care necesita o setare manuala a conexiunii dupa ce dispozitivele sunt la o distanta la care pot comunica intre ele.

4.4.2.2. Aplicații Android[16]

Android este un sistem de operare dezvoltat de Google, bazat pe o versiune modificată a kernel-ului de Linux și alte software-uri open source. Este destinat în principal pentru dispozitive mobile touchscreen cum ar fi telefoane sau tablete.

Aplicațiile Android sunt de obiceti dezvoltate în Java (însă se pot folosi și limbaje precum Kotlin și C++)folosind Android Software Development Kit. Odată dezvoltate, aplicațiile pot fi împachetate ușor și vândute fie printr-un store online cum ar fi Google Play, SlideME, F-droid sau Amazon Appstore.

Fiecare aplicație[17] Android „trăiește" în propriul sau sandbox securizat, protejat de facilități de securitate Android cum ar fi: sistemul de operare Android, în care fiecare aplicație individuală este tratată că un utilizator diferit; fiecare proces are propria mașină virtuală, deci codul unei applicatii rulează independent de celelalte aplicații; sistemul de operare asigneaza fiecărei aplicații un ID unic (folosit doar de sistem, aplicația nu știe acest lucru). Sistemul setează permisii fiecărui fișier din acea aplicație astfel încât doar o entitate cu ID-ul asignat aplicației le poate accesa.

Principala componentă a unei aplicații este activitatea. O activitate este un entry point pentru interacțiunea cu utilizatorul. Reprezintă un singur ecran cu o interfață utilizator (ex: o pagină de login). Aceste activități ajută la construirea unei structuri coezive, însă sunt independente unele de altele la modul în care orice aplicație este capabilă să pornească o activitate din altă aplicație dacă aplicația respectivă permite acest lucru.

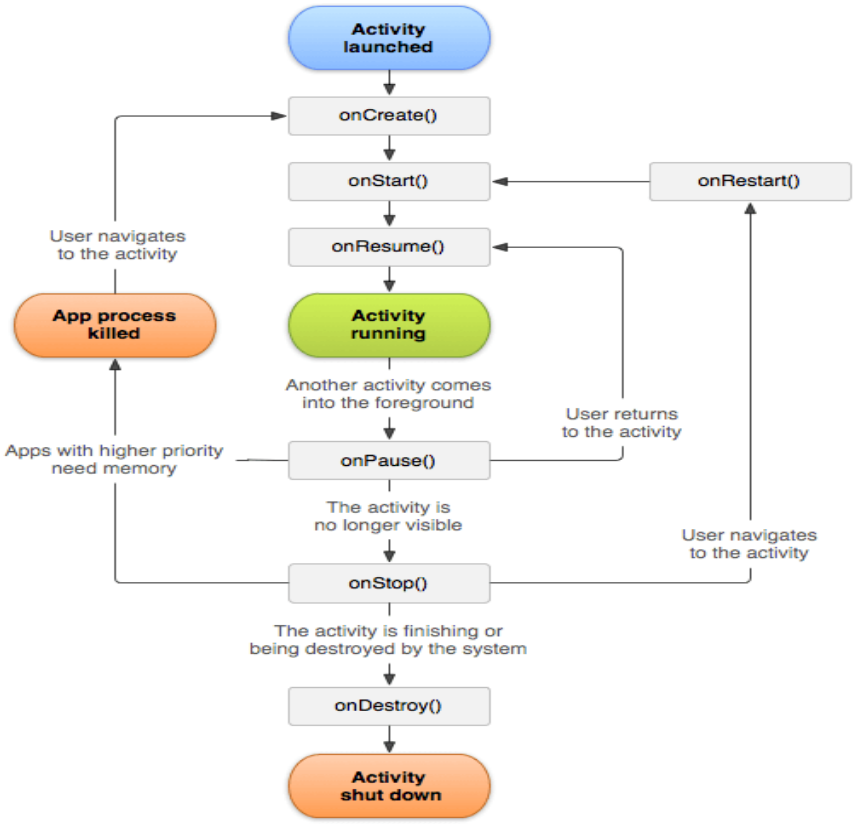
4.4.2.3. Ciclul de viață a unei activități Android

În timp ce utilizatorul navighează în cadrul unei aplicații, instanțele fiecărei activități trec prin diferite stări în ciclul lor de viață. Clasa Activiy pune la dispoziție o serie de callback-uri care ii permit activității respective să știe că o stare s-a schimbat. Sistemul este cel care creează, oprește, reia o activitate sau distruge procesul în care se află activitatea respective.

[16] https://en.wikipedia.org/wiki/Android_(operating_system)
[17] https://developer.Android.com/guide/components/fundamentals

În cadrul acestor metode de callback, utilizatorul poate să interacționeze cu fiecare activitate în parte, modelând astfel comportamentul acestora.

Figura 4.7 reprezintă o ilustrare a ciclului de viață unei activități Android.



[18]

Figura 4.5 Ciclul de viață a unei activități Android

Metodele de callback din fiecare activitate sunt prezentate mai jos.

OnCreate - Este metodă pe care este obligatoriu să o implementăm în fiecare activitate. Este folosită pentru operații de initializare. Primește un parametru care are că valoare starea anterioară salvată a activității.

OnStart – Invocarea acestei metode pregătește activitatea pentru a fi vizibilă de către utilizator. Este metodă în care aplicația initializeaza codul care menține interfață grafică. Metodă se termină foarte repede și la fel că și în cazul stării „Created", activitatea nu rămâne în această stare. În momentul în care această metodă se termină de executat activitatea intră în starea „Resumed".

OnResume – Când o activitate intră în starea „Resumed" sistemul invocă această metodă. Activitatea stă în această stare până când are loc un eveniment ce ia focusulde pe această activitate. Un astfel de eveniment poate fi de exemplu primirea unui apel, navigarea spre o altă activitate sau oprirea ecranului dispozitivului.

OnPause – Sistemul apelează această metodă în momentul primei indicății a faptului că utilizatorul dorește să părăsească activitatea respectivă. Nu indica faptul că activitatea va fi distrusă, ci doar că nu mai este în prim plan.

[18] https://developer.Android.com/guide/components/activities/activity-lifecycle

OnStop – Când activitatea nu mai este vizibilă pentru utilizator înseamnă că a intrat în starea „Stopperd", moment în care sistemul invocă metodă onStop(). Asta se întâmplă de exemplu când o nouă activitatea este în prim plan. În această metodă ar trebui să eliberăm orice resurse folosite dacă acest lucru nu se face automat. Un alt scenariu de folosire a metodei onStop() este pentru efectuarea de acțiuni care necesită o putere de calcul mare cum ar fi spre exemplu salvarea unor date într-o baza de date.

OnDestroy – Este invocată înainte că activitatea să fie distrusă. Sistemul invocă această metodă dacă activitatea se termină (dacă userul face asta sau dacă este invocată explicit metodă finish()), sau dacă sistemul distruge activitatea temporar datorită unei schimbări de configurație, cum ar fi spre exemplu rotația dispozitivului ce duce și la rotația ecranului sau activarea unui mod de vizualizare ferestre multiple.

4.4.2.4. Spring[19]

Spring este un framework Java foarte popular pentru dezvoltarea de aplicații web și enterprise. Este un framework foarte flexibil deoarece oferă posibilitatea configurării aplicației prin xml-uri, adnotări sau configurații Java.

Se bazează pe modelul IoC container (Inversion of Control) cunoscut și că DI (Dependency Injection). Este un proces în care controlul este dat aplicației în sine și nu utilizatorului. Obiectele în sine își definesc dependințele (celelalte obiecte cu care lucrează) doar prin argumente date în constructor, unor metode de tip factory sau prin proprietăți care sunt setate pe un obiect după ce este construit sau returnat dintr-o metodă de tip factory. După acest lucru, containerul în sine injectează acele dependințe când creează obiectul (bean-ul).

Este un framework foarte popular deoarece:

1. Abordarea DI duce la scrierea unui cod usor de testat
2. Capabilități de management de tranzacții cu date din baze de date foarte puternice și ușor de folosit
3. Integrare simplificată a altor framework-uri Java cum ar fi JPA/Hibernate, Strust, JSF etc.
4. Utilizarea pattern-ului MVC pentru dezvoltarea aplicațiilor web.

Am ales să folosesc Spring deoarece suită Spring oferă suport pentru multe funcționalități necesare cum ar fi securitatea aplicației (Spring Security), suport pentru accesul la baze de date (Spring Dată JPA), dezvoltare aplicații bazate pe microservicii (Spring Cloud Dată Flow) și oferă chiar și componente ce pot fi folosite în dezvoltarea aplicațiilor Android (Spring for Android).

4.4.2.5. Spring boot[20]

Datorită faptului că Spring-ul în sine oferă o mare flexibilitate a configurarii, complexitatea crește de asemenea, de aceea în multe cazuri configurarea unei aplicații Spring devine anevoioasă și expusă erorilor de configurare.

Cei ce au dezvoltat acest framework au oferit o soluție pentru a evita creșterea complexității în urmă configuratiilor prin dezvoltarea SpringBoot.

[19] https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html
[20] https://dzone.com/articles/why-springboot

Acest framework este capabil să facă toate configurațiile necesare automat, dar oferă și posibilitatea de a suprascrie aceste configurații dacă este necesar. Configurarile automate sunt realizate în urmă adăugării unor dependințe părinți care vor realiza această configurare automată prin adăugarea automată a dependintelor necesare.

Una dintre cele mai apreciate funcționalități oferite de SpringBoot este cea că oferă suport pentru Servlet Container. Acest lucru înseamnă că nu trebuie să facem deloy la aplicație pe un server extern instalat manual (Tomcat, Jetty etc.) deoarece dependințele adăugate se ocupă de acest lucru, acesta fiind motivul principal pentru care am ales să folosesc Spring Boot.

4.4.2.6. Spring Data JPA[21]

Implementarea unui layer de acces la baza de date a fost destul de greoaie o vreme. Era nevoie de foarte mult cod repetitiv pentru a execută simple query-uri pe o baza de date. Spring Dată JPA ține să ușureze implementarea unui astfel de layer prin eliminarea codului necesar să fie scris și generarea automată a acestui doar atunci când este nevoie. Acest lucru este realizat prin utilizarea de interfețe puse la dispoziție de Spring Dată JPA, a căror implementarea este realizată automat la runtime de către framework.

Oferă un suport sofisticat pentru a crea repository-uri de date bazat pe Spring și JPA. Prin crearea acestor repository-uri, vom avea deja acces la operațiile CRUD de baza, care primesc de obicei că parametru fie un id fie un obiect întreg.

Se pot bineinteles crea și operații mai sofisticate prin una din următoarele metode:

1. Definirea unei noi metode în interfață respectivă
2. Crearea unui query JPQL folosing adnotarea @Query
3. Folosirea suportului pentru Querydsl (Un moc de a crea query-uri type-safe)
4. Definirea de query-uri custom prin folosirea de NamedQuery din JPA

Am ales să folosesc Spring Dată JPA deoarece oferă posibilitatea de a comunica cu baze de date prin interfețe abstracte simple, implementarile fiind generate automat la runtime.

4.4.2.7. Hibernate[22]

Hibernate este un ORM tool (Object/Relațional mapping) care permite utiliaztorilor să creeze foarte ușor aplicații care accesează baze de date externe. Este folosit doar pentru bazede la date de tip realational.

În plus față de API-ul sau nativ, Hibernate reprezintă de asemenea și o implementare a JPA (Java Persistence API), de aceea poate fi folosit foarte ușor în orice aplicație ce suportă JPA.

Hibernate ne permite să dezvoltăm clase persistente urmând principiile OOP de baza (moștenire, polimorfism, asociere, compoziție), dar și Java collections framework. Nu are nevoie de nicio interfață pentru clasele persistente și permite translatarea într-o baza de date a oricrui structura de date.

[21] https://projects.spring.io/spring-data-jpa/
http://www.baeldung.com/the-persistence-layer-with-spring-data-jpa
[22] http://hibernate.org/orm

Hibernate suportă conceptul de lazy initialization, ceea ce înseamnă că datele din baza de date sunt aduse de către framework doar în momentul în care este nevoie de ele, pentru a elimina anumite operații costisitoare atunci când nu este nevoie de ele.

Un alt concept suportat de Hibernate este cel de optimistic locking[23]. O resursă nu este blocată în momentul în care este accesată de o tranzacție. În schimb,, starea resursei este salvată. Alte tranzacții au posibilitatea să acceseze și să modifice resursă concurent. În momentul în care ar trebui să aibă loc salvarea modificărilor, starea resursei este citită din baza de date și comparată cu starea ei inițială ce a fost salvată. Dacă aceste două stări diferă înseamnă că au avut loc conflicte în modificarea resursei și tranzacția este astfel oprită, nicio modificare fiind salvată.

Hibernate nu necesită că baza de date și tabelele să fie create în prealabil deoarece pune la dispoziție configurări care vor crea baza de date și tabelele automat la pornirea sistemului. Aceste configurări sunt bazate de clasele ce urmează a fi mapate la tabelele din baza de date și pe configurarie de acces la baza de date oferite de către utilizator.

Este un framework de acces la baze de date foarte fiabil în termeni de stabilitate și calitate, lucru dovedit de faptul că este folosit de un număr mare de Java developers, motiv pentru care a fost ales și de mine că și ORM.

4.4.2.8. Spring Security[24]

Spring Security este o altă extensie a framework-ului de Spring care se concentrează pe implementarea functionalitatilor de autentificare și autorizare în aplicații Java. Că și celelalte extensii Spring, marele avantaj al Spring Security constă în faptul că este foarte ușor de extins pentru a îndeplini anumite cereri custom.

4.4.2.9. Oauth2[25]

Oauth2 este un protocol de autentificare și autorizare. Acest protocol permite aplicațiilor de tip third-party să permită accesul la unele servicii HTTP, fie în numele unnui Resource Owner sau prin permiterea unei aplicații să obțînă acest acces in numele sau. Accesul este cerut de un client care poate fi de exemplu un website sau o aplicație mobilă.

Protocolul Oauth2 definește 4 roluri:

1. Resource Owner – în general o persoană care folosește aplicația
2. Resource Server - un server care deține date protejate (de exemplu un serviciu de la Google care deține informații personale)
3. Client – o aplicație care cere accesul la Resource Server (poate fi un website, o aplicație mobilă etc.)
4. Authorization server – server care emite un token de acces clientului. Acest token va fi folosit de către client pentru a face requesturi la Resource Server

Un token este un strîng generat random de către Authorization Server când clientul face o cerere de token.

Există două tipuri de token-uri:

---

[23] https://docs.jboss.org/jbossas/docs/Server_Configuration_Guide/4/html/TransactionJTA_Overview-Pessimistic_and_optimistic_locking.html

[24] https://projects.spring.io/spring-security

[25] http://www.bubblecode.net/en/2016/01/22/understanding-oauth2

1. Access token – este cel mai important deoarece permite aplicațiilor externe accesul la date. El este trimis de către client că parametru un header-ul requestului pe care îl face. Are o durata de viață limitată (definită de către authorization server), dar poate fi modificate în funcție de preferințe.
2. Refresh token – este emis în același timp cu acces token-ul, dar nu trebuie să fie trimis la fiecare request făcut de către client. Este folosit pentru a fi tirmis la authorization server pentru a reînnoi acces token-ul, eliminând astfel necesitatea de a introduce din nou datele de autentificare.

Tipuri de autorizare:

1. Authorization code grant – ar trebui să fie folosit în momentul în care clientul este un seb server.
2. Implicit grant – este folosit de obicei când clientul rulează într-un browser folosind un scripting language cum ar fi JavaScript. Acest tip de autorizare nu permite emiterea de refresh token.
3. Resource Owner Password Credentials grant – în acest tip de autorizare, credentialele sunt trimise clietului apoi către Authorization Server, deci este nevoie de încrede maximă între aceste două entități. Acest tip de autorizare este folosit în general când clientul a fost dezvoltate de aceeași autoritate că și Authorization Server-ul.
4. Client Credentials grant – este folosit când clientul însuși este Resource Owner. Nu există autorizație care să fie obținută de la un end-user.

4.4.2.10. REST[26]

REST (Representational State Transfer) este un stil arhitectural care oferă standarde de comunicare între sistemele computationale din cadrul Internetului, facilitând astfel comunicarea ușoară între ele. Sistemele RESTful sunt stateless și separă interesele clientului de cele ale serverului (Separation of Concerns).

Prin folosirea acestui stil arhitectural, implementarea clientului și a serverului poate fi făcută independent, fără că aceștia să știe unul de altul. Asta înseamnă că implementarea clientului poate fi modificată fără să afecteze serverul și invers.

Atâta timp cât fiecare parte știe formatul în care să transmită mesajele între ele, ele pot fi separate una de cealaltă. Separarea interfeței utilizator de date duce la flexibilitate mult mai mare și permite fiecărei părți să evolueze independent.

Prin folosirea acestui stil arhitectural mai mulți clienți pot să comunice cu același server dacă știu formatul mesajelor pe care le acceptă serverul.

Noțiunea de „stateless” face referire la faptul că serverul nu trebuie să știe starea în care se află clientul și viceversa. În acest mod, și clientul și serverul pot să înțeleagă orice mesaj primit fără să știe vreun lucru despre mesajale anterioare.

Într-o arhitectură REST, clienții trimit requesturi la server pentru a citi sau a modifică resurse, iar serverele trimit răspunsuri la aceste requesturi.

Un request este alcatuit de obicei din:

1. Un verb HTTP care denotă ce tip de operație să se execute
2. Un header care permite clietului să trimită informație adițională despre request

[26] https://www.codecademy.com/articles/what-is-rest

3. O cale până la resursă (sub formă unui URL)
4. Un message body opțional prin care clientul poate să trimită anumite date/anumiți parametri, dacă este necesar

Verbele HTTP sunt metode care denotă ce tip de operație trebuie să execute serverul. Există 4 verbe HTTP de baza:

1. GET – returnează o resursă specifică (de exemplu după un id), sau o colecție de resurse
2. POST – creează o nouă resursă
3. PUT – face un update la o resursă existența și este idempotent (un request de tip puț cu aceiași parametri va aduce modificări doar prima dată când este trimis; orice request consecutiv cu aceiași parametru nu va avea niciun efect)
4. DELETE – șterge o anumită resursă

În partea de headers clientul poate să trimită un parametru care denotă ce tip de răspuns este capabil să înțeleagă (ex: application/xml, applicatio/json, text/plain etc.).

Tratarea erorilor se face prin anumite coduri de răspuns [27]. În general există patru categorii de coduri de răpuns:

1. 1xx Informational responses - Conțin doar informații despre request
2. 2xx Success - Denotă că requestul a fost făcut cu succes
3. 3xx Redirection - Acest tip de cod de răspuns denotă faptul că entitatea care face requestul trebuie să completeze anumite alte acțiuni înainte că requestul să fie completat (redirect)
4. 4xx Client error – Indică erorile cauzate de către client
5. 5xx Server error – Indică erorile cauzate de către server

Formatul de mesaj cel mai utilizat în arhitecturile REST este cel de JSON (JavaScript Object Notation) deoarece mesajele trimise sub acest format sunt foarte ușor de interpretat și parsat, motiv pentru care am ales REST pentru comunicare între aplicațiile dezvoltate.

4.4.2.11. Angular[28]

Angular este o platforma open-source de dezvoltare a aplicațiilor de frontend web și mobile dezvoltată de Google și se bazează pe TypeScript că limbaj de programare.

Această platforma permite dezvoltarea de aplicații mobile native (folosind strategii luate de la Cordova, Ionic sau NativeScript), aplicații Desktop pe Max, Windows și Linux folosind aceleași metode Angular învățate pentru web, având în plus abilitatea de a aces API-uri native pentru diferite sisteme de operare.

Angular reușește să transforme template-urile create de user în cod care este foarte bine optimizat pentru mașinile virtuale de JavaScript, oferind utilizatorului toate beneficiile codului scris de mâna și productivitatea și viteză unui framework.

Aplicațiile scrise în Angular sunt universale deoarece sunt transformate doar în HTML, CSS și JavaScript la runtime.

[27] https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
[28] https://angular.io/features

Aplicațiile Angular se încarcă foarte rapid datorită faptului că doar modulele care sunt vizibile de către utilizator sunt încărcate la un moment dat, eliminând astfela problema începetinirii aplicației pe măsură ce dimensiunea acesteia crește.

Testarea se realizează foarte ușor folosind framework-uri precum Karma sau Protractor.

Platforma pune la dispoziție și animații ce nu scad viteză de răspuns a aplicației.

Am ales să folosesc Angular dincauza mecanismului de detectare a modificărilor care merge pe arborele componentelor HTML și verifică fiecare componentă pentru modificări și actualizează DOM-ul automat atunci când se schimbă proprietățile componentelor.

4.4.2.12. Typescript[29]

Typescript este un limbaj de programare open-source dezvoltat de Microsoft. Este un superset JavaScript prin faptul că oferă capabilități de OOP și de type-safety limbajulu de JavaScript.

Este destinat dezvoltării de aplicații mari. La runtime, tot codul TypeScript este transformat în JavaScript. Datorită faptului că este un superset JavaScript, orice program JavaScript este un program TypeScript valid.

Acest limbaj poate fi folosit atât pentru dezvoltarea aplicatiil de tip client cât și pentru dezvoltarea aplicațiilor de tip server (folosind Node.js).

Tipurile sunt foarte importante și permit programatorilor JavaScrip să folosească tooluri de development mult mai productive și practici cum ar fi refactorizarea codului și verificarea statică a tipurilor de date.

Pe lângă faptul că oferă posibilitatea de scriere de cod JavaScript typesafe, un alt motiv pentru care am ales TypeScript este faptul că Angular este construit pentru a fi folosit împreună cu Typescript. Orice aplicație Angular poate fi de asemenea scrisă doar în JavaScript.

4.4.2.13. Volley[30]

Volley este o librărie HTTP care ușurează lucrul în rețea pentru aplicațiile Android. Este dezvoltată de compania Google și este un proiect open-source, fiind disponibil pe GitHub.

Librăria oferă următoarele funcționalități:

1. Planificare pentru executarea automată a requesturilor
2. Executarea de requesturi asincrone. Acest lucru este deosebit de important doarece, în general operațiile asincrone au avantajul de a nu bloca o aplicația cât timp acestea sunt în execuție. Un request asincron nu împiedică utilizatorul din a execută un alt request înainte că primul să fi întors un răspuns.
3. Posibilitatea de a salva în cache răspunsurile de la anumite requesturi. Acest lucru facilitează viteză de funcționare a aplicației

[29] https://www.typescriptlang.org/

[30] https://developer.Android.com/training/volley

deoarece putem să salvăm unele date dacă știm că acestea nu se vor modifică foarte des dar avem nevoie de ele în majoritatea cazurilor. Astfel, prima dată datele vor fi aduse de la server prin request, iar cererile următoare pentru aceleași date vor folosi datele deja salvate. Acest cache poate fi customizat de către utilizator, de exemplu pentru setarea duratei datelor în cache.

4. Suport pentru prioritizarea unor requesturi prin așezarea acestora într-o coadă de priorități.

Volley excelează la operațiile de tip RPC (Remote Procedure Call) folosite pentru a popula interfaata utilizator, cum ar fi cererile pentru o pagină cu rezultatele unei căutări. Se integreza foarte ușor cu orice protocol și vine cu suport pentru JSON, imagini și șiruri de caractere normale.

Nu este indicată folsirea acestei librării pentru operații de download sau streaming deoarece ține toate răspunsurile în memorie în timpul parsarii, ceea ce pentru un smartphone poate fi uneori un bottleneck.

Am ales să folosesc această librărie deoarece este dezvoltată de Google, deci beneficiază de suport specializat. Un alt motiv este faptul că aplicația nu necesită operații de straming sau de download.

Capitolul 5. Proiectare de Detaliu și Implementare

În cadrul acestui capitol va fi prezentată implementarea sistemului printr-o abordare de tip bottom-up, deoarece aceasta a fost abordarea folosită si pentru implementare.

Am văzut în capitolul 3 o descriere și o diagramă arhitecturală de ansamblu a sistemului. Acest capitol se va axa pe descrierea amănunțită a fiecărei componente prezentate.

Pentru partea de baza de date se va descrie serverul pe care este plasată, modul de conectare la această, formă normală în care se află și descrierea fiecărui tabel ce intră în componentă să.

Pentru partea de backend se va descrie în amănunt modeul arhitectural folosit, rolul fiecărei componente din această arhitectură. De asemenea, vor fi prezente exemple care să clarifice abordările folosite pentru scenariul respectiv.

Partea de Android va conține detalii atât despre implementarea de logică a aplicației cât și despre modul de implementare a layout-urilor.

Partea de web pentru administrator va conține informații despre modul de trasnmitere a datelor pentru operațiile CRUD de baza implementate.

Unde este cazul, unele dintre funcționalități vor fi exemplificate și prin diagrame de secvență sugestive.

5.1. Baza de date

Baza de date folosită este de tip MySQL. Pentru stocarea bazei de date am folosit serviciul cloud de la Google numit Google Cloud SQL. Acesta permite crearea de proiecte/conexiuni care pot să aibă în componentă mai multe baze de date, după preferințele utilizatorilor. Pentru accesarea bazei de date este neovie să cunoaștem următoarele:

1. Numele bazei de date pe care dorim să o accesăm
2. Instance connection name: este un identificator unic al conexiunii/instanței în care este prezența baza de date și este alcătuită din 3 componente:
    a. Numele propriu-zis al instanței (generat random în cadrul planurilor basic, sau care poate fi ales în cadrul planurilor pentru soluțiile enterprise)
    b. Numele serverului pe care este hostata această instanța. Acesta este ales de către utilizator în funcție de distanță acestuia față de server.
    c. Numle proiectului/conxiunii din care face parte instanța
3. Url-ul propriuzis de conexiune la baza de date care specifică protocolul de comunicare și înglobează cele două date de identificare specificate mai sus.
4. Nu în ultimul rând, aceste informații trebuie bineînțeles să fie însoțite de un nume de utilizator și o parolă pentru a avea acces.

O ultima configurare de securitate se poate face chiar în consola de management a bazei de date din cadrul Google Cloud SQL care permite selectarea ip-urilor sau domentiilor care pot avea acces la baza de date. Dacă aplicația care dorește să acceseze baza de date cunoaște informațiile de conexiune, dare are un IP care nu este trecut în lista de IP-uri care au acces la baza de date, aplicație nu se poate conecta la această.

Totodată, pentru folosirea oricărui serviciu de Google Cloud este necesară generarea din cadrul consolei de administrare a unor API keys sau a unor fișiere JSON ce conțîn informații de indentificare pentru a spori și mai mult securitatea. În cazul aplicației implementate am ales să folosesc fișierul JSON generat automat. Pentru utilizare, acesta va fi setat că target pentru o variabilă de sistem creată de utilizator cu numele GOOGLE_APPLICATION_CREDENTIALS. Fișierul folosit în aplicație este este inclus in fișierele sursă a aplicației de server.

În cele ce urmează va fi prezentată diagramă bazei de date, iar mai apoi vor fi analizate tabelele din cadrul acesteia și va fi justificată soluția aleasă.

5.1.1. Diagrama bazei de date

Baza de date folosită în aplicație este de tip relațional (MySQL).

5.1.1.1. Forme normale

Prima formă normală (1NF) este o proprietate a tabelelor bazelor de date relaționale care spune că fiecare atribut din tabel conține doar valori atomice din domeniul de date declarat pentru atributul respectiv. Că exemplu putem lua atributul de email din tabela de User. Acest atribut va avea întotdeauna stocat un singur email pentru o înregistrare dată din tabela de User.

În momentul actual, țintă este că o baza de date să se afle fie în a treia formă normală, fie în formă normală Boyce-Codd[31]. Această formă normală încearcă să rezolve anumite anomalii ce apar în cazul formei normale 3. În cazuri rare, 3NF nu îndeplinește cererile BCNF, deci nu se află în BCNF. Un tabel se află în formă normală Boyce-Codd dacă toate redundanțele bazate pe dependențe funcționale au fost eliminate. În altă odine de idei, o baza de date se află în formă normală Boyce-Codd dacă și numai dacă pentru fiecare dintre dependențele sale X ->Y, cel puțin una dintre condițiile de mai jos este adevărată:

- X -> Y este o dependență funcțională trivială, ($Y \subseteq X$)
- X este o supercheie pentru baza de date

Un tabel in forma normala 3 care nu are multiple chei candidat care sa se suprapuna este garantat sa se afle si in BCNF. Daca insa tabelul are chei candidat care se suprapun el poate sa nu fie in BCNF, acest lucru depinzand de dependentele sale functionale.

În figura 5.1 este prezentată diagramă bazei de date.

[31] https://en.wikipedia.org/wiki/Boyce%E2%80%93Codd_normal_form

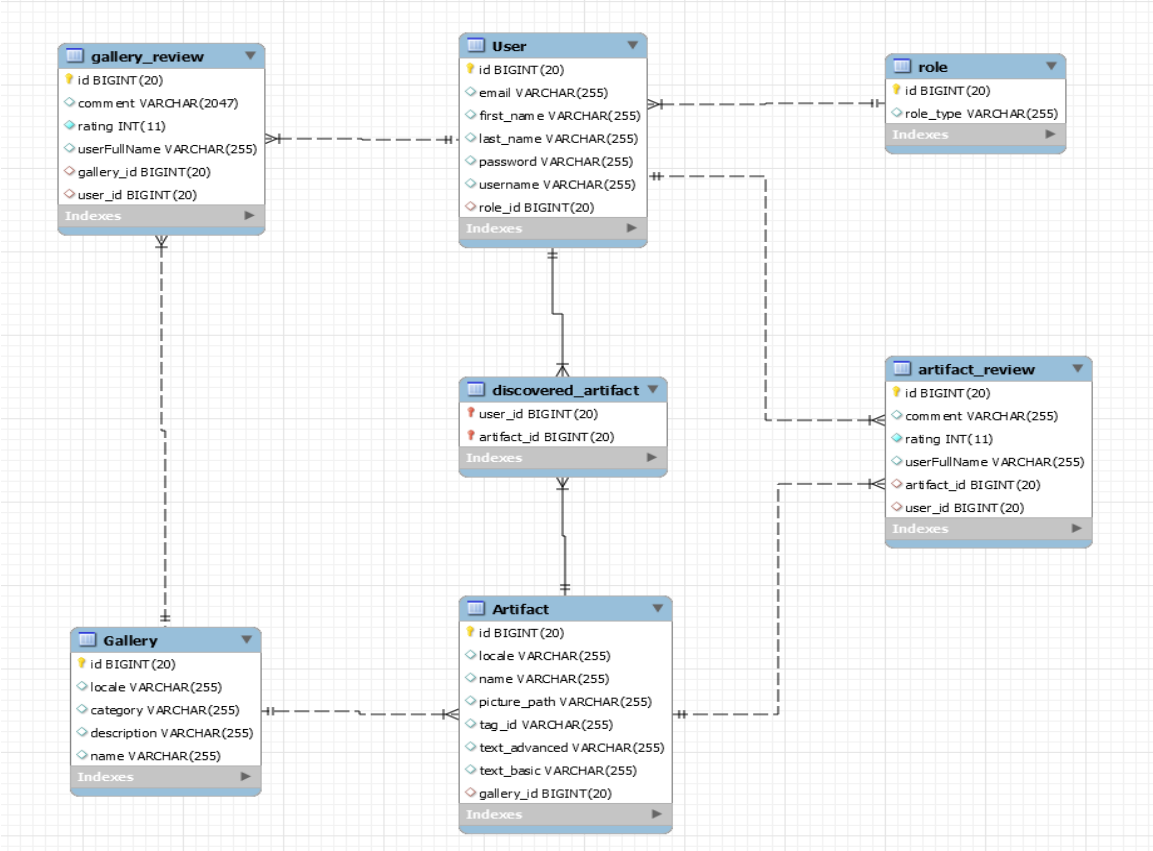


Figura 5.1 Diagrama bazei de date

5.1.2. Descrierea tabelelor prezente

După cum se poate observa, în componenta bazei de date folosite intră 7 tabele:

- Tabela User este cea care ține informațiile generale cu privire lautilizatorul aplicației (fie el utilizator normal sau administrator). Înregistrarea de „password” stochează parolă criptată a utilizatorului. Tabela conține o cheie străină care pointează spre tabela de rol care diferențiază tipurile de utiliaztori.
- Tabela Role conține pe lângă cheia primară o singură înregistrare (role_type) care poate să conțina doar valorile ROLE_USER sau ROLE_ADMIN. Aceste două valori ajută la diferențierea tipurilor de utilizatori.
- Tabela gallery conține informații cu privire la galeriile prezente în muzeu. Înregistrarea de locale este necesară pentru implementarea funcționalității de internaționalizare a aplicației. Textul din cadrul înregistrării “description” este folosit și pentru construirea unui fișier audio ce poate fi redat din cadrul aplicației.
- Tabela Artifact conține informații cu privire la artefactele din fiecare galerie. Are de asemenea o înregistrare „locale” pentru diferitele limbi suportate de aplicație. Pe lângă cheia primară, fiecare artefact este

identificat unic prin field-ul „tag-id” care este corespunzător unui id de tag NFC. Acesta tabela conține descrieri de tip basic și advanced pentru artefactul respectiv. Utilizatorul aplicației mobile nu poate să vadă artefactele în cadrul aplicației înainte că acestea să fie scanate de către el. După scanare, tabela discovered_artifact va fi populată cu id-iul artefactului scanat și astfel va putea fi vizibilă în aplicație.

- Tabela discovered_artifact este defapt o tabela intermediară ce elimina o relație de many-to-many între două entități, dar care, în cadrul aplicației servește și că o constrângere pentru ce artefacte este capabil utilizatorul să vizualizeze. Tabela este generată automat de către Hibernate, iar secvență de cod HQL (Hibernate Query Language) care face acest tip de diferențiere este următoarea: **SELECT a FROM Artifact a JOIN a.users u WHERE u.id = ? and a.gallery.id = ?**. În limbaj natural, secvență de mai sus se traduce în felul următor: Selectează toate înregistrările din tabela Artefact(a) care au id-ul utiliaztorului și id-ul galeriei egale cu cele transmise că parapetri. Semnele de întrebare din Right Hand Side sunt folosite că placeholders și vor fi înlocuite cu parametri transmiși când această operație este realizată.
- Tabelele artifact_review și gallery_review sunt folosite pentru stocarea de review-uri realizate de către utilizatori în urmă unei vizite. Am ales să le descriu împreună deoarece servesc în principal aceluiași scop, diferența fiind doar una dintre cheile străine (gallery_id sau artifact_id). Informațiile despre un review conțîn un rating, o descriere scurtă și numele utilizatorului care a scris review-ul respectiv. Numele utilizatorului este adăugat aici pentru a ușura operația de citire a review-urilor prin simplul fapt că nu trebuie să se facă interogări in plus pentru a află numele utilizatorului care a făcut respectiviul review după chiea străină user_id.

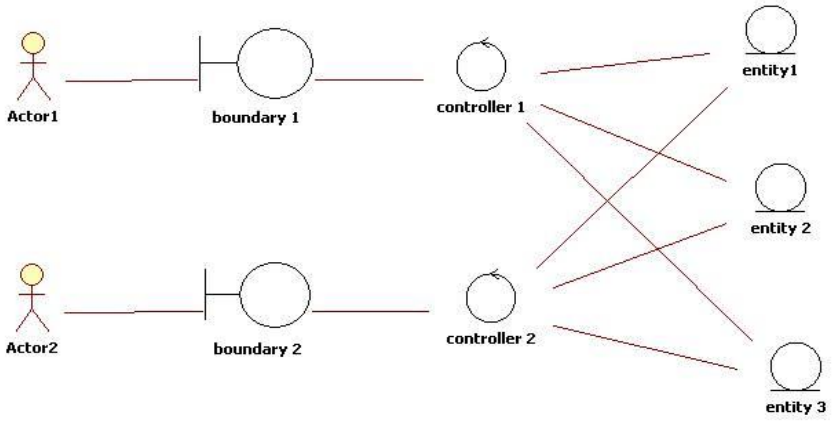
5.2. Aplicația de backend

Partea de backend a aplicației este scrisă în Java și folosește serverul de Tomcat pus la dispoziție de Spring Boot că și container de deployment pentru aplicație. Descrierea părții de backend va fi realizată urmând tot modelul bottom-up.

Că și stil arhitectural am ales să folosesc arhitectură ECB (Entity-Control-Boundary). Este o variație a pattern-ului Model-View-Controller.

5.2.1. Modelul architectural ECB

Modeul Entity-Control-boundary este o variatie a modelului Model-View-Controller. Diferanta fata de modelul MVC este faptul ca partea de View nu este neaparat parte de client sau de afisare a datelor, ci poate si fi o component ce expune anumite servicii spre a fi folosite de catre alte aplicatii. O alta diferenta este modul de comunicare intre componentele ECb(fiecare componenta comunica atat cu cea de deasupra ei cat si cu cea de dedesupt). Mai jos este prezentata o diagram conceptuala a unei astfel de arhitecturi.



[32]

Figura 5.2 Modelul arhitectural ECB

Entitățile sunt obiecte care reprezintă datele din sistem: Customer, Transaction, Cart, etc. Boundary-urile sunt obiecte care oferă o interfață de comunicare cu sistemul pentru actorii sistemului (ex: interfețe utilizator, gateway-uri, proxy-uri, etc.).

Controllerele sunt obiecte care mediază interacțiunea între entități și boundaries. Ele execută comenzile venite de la boundary.

In principal, o astfel de arhitectura functioneaza in felul următor:

1. Actorii interacționează cu obiectele de tip boundary
2. Obiectele boundary trimit comenzi obiectelor de tip controller
3. Obiectele de tip controller pot să trimită comenzi înapoi către boundary pentru a cere mai multe informații de la actori.
4. Controllerele execută operații de update pe entități
5. Obiectele de tip boundary se „reîmprospătează” (refresh) pentru a reflectă schimbările ce au avut loc în entități.

În cadrul aplicației, componentele stilului arhitectural sunt defapt pachetele care sunt folosite pentru structurarea aplicației. Pe lângă acestea trei mai există încă un pachet ce conține configurarile necesare pentru aplicație. Fiind locul unde se realizează partea de logică a plicatiei, pachetul de control este împărțit la rândul sau în subpachete pentru a separă și mai mult componentele independente. Logică de business are loc în clasele din pachetul service. Pachetele DAO și DTO sunt folosite pentru manipularea datelor venite din baza de date și din endpoin-urile REST.

În figura 5.2 este prezentată diagrama de pachete.

[32] http://www.cs.sjsu.edu/~pearce/modules/patterns/enterprise/ecb/ecb.htm

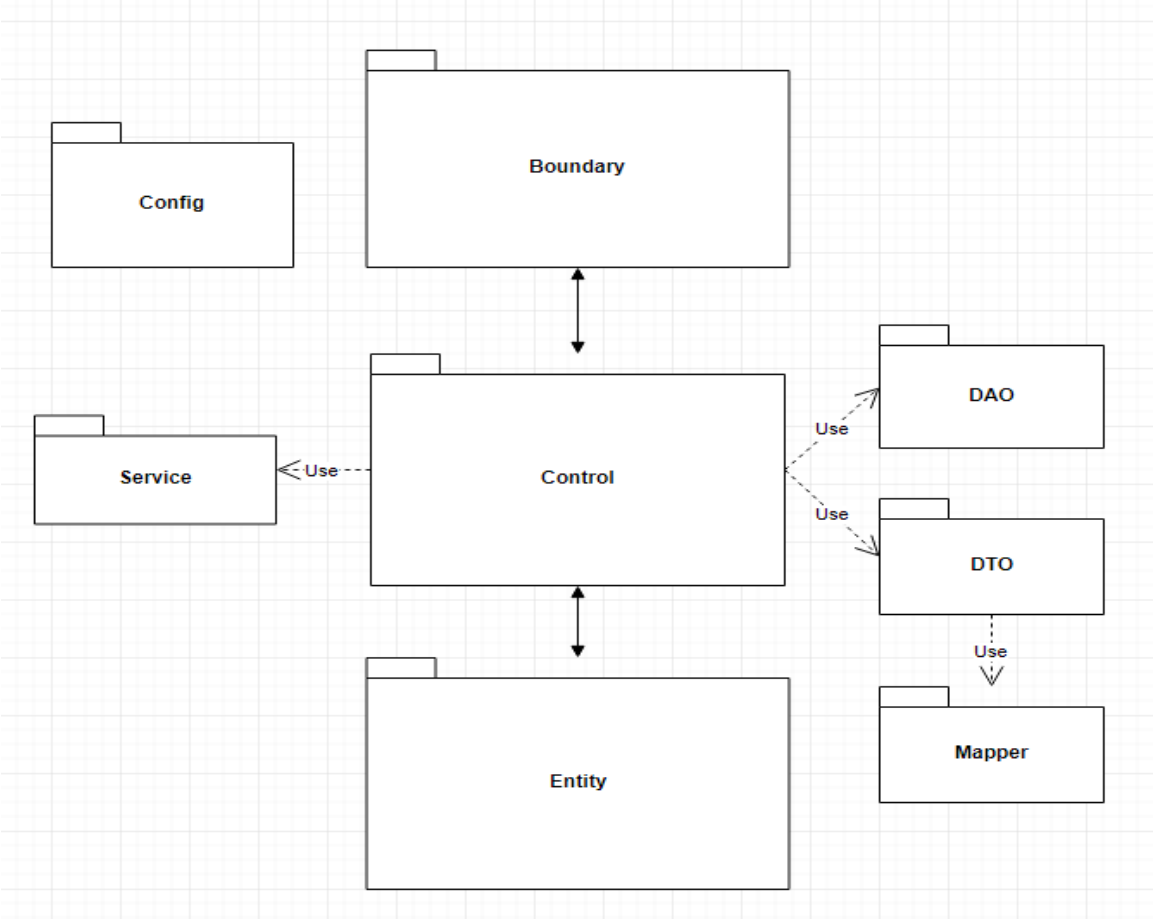


Figura 5.3 Diagrama de pachete

5.2.2. Entity

Această componentă conține clasele de model folosite care se mapeaza la tabelele din baza de date. Această mapare este realizată folosind Hibernate că și ORM. Această mapare este realizată prin folosirea de adnotări puse la dispoziție de către Hibernate. Principala problema care intervine când dorim să mapam clase de model la tabelele unei baze de date. Această problema apare din cauza faptului că entitățile de model din cadrul aplicației terbuie construite în stil orientat obiect, pe când baza de date MySQL nu cunoaște acest concept. Putem lua în considerareexemplul din figura 5.3 care ilustrează această problema și soluția oferită de către Hibernate:

Figura 5.4 Exemplu de relație one-to-many

Observăm în diagramă de mai sus o relație de one-to-many de la tabela role la user. Un user poate avea asignat un singur rol(utilizator normal sau admin), în timp ce un rol

poate fi asignat la mai mulți utilizatori. Această relație este reprezentată în cadrul bazei de date prin prezența unei chei străine în tabela de user ce pointeaza spre o anumită înregistrare din tabela de roluri și prin legătură însoțită de cardinalitati dintre cele două tabele. Ei bine, în limbaj orientat obiect avem neovie că această relație să se conformeze standardelor OOP, lucru realizat în următorul fel: Partea de relație „one” (în cazul acesta entitatea role) va conține o lista cu utilizatorii la care este asignat, în timp ce partea de „many” a acestei relații va conține o referință la cealaltă entitate (în cazul acesta o entitate de tip role). Un ORM acționează că un bridge între aceste două implementări pentru a putea fi posibilă comunicarea fără probleme între acestea.

În figura 5.4 este prezentată diagramă de clase pentru pachetul de entități.

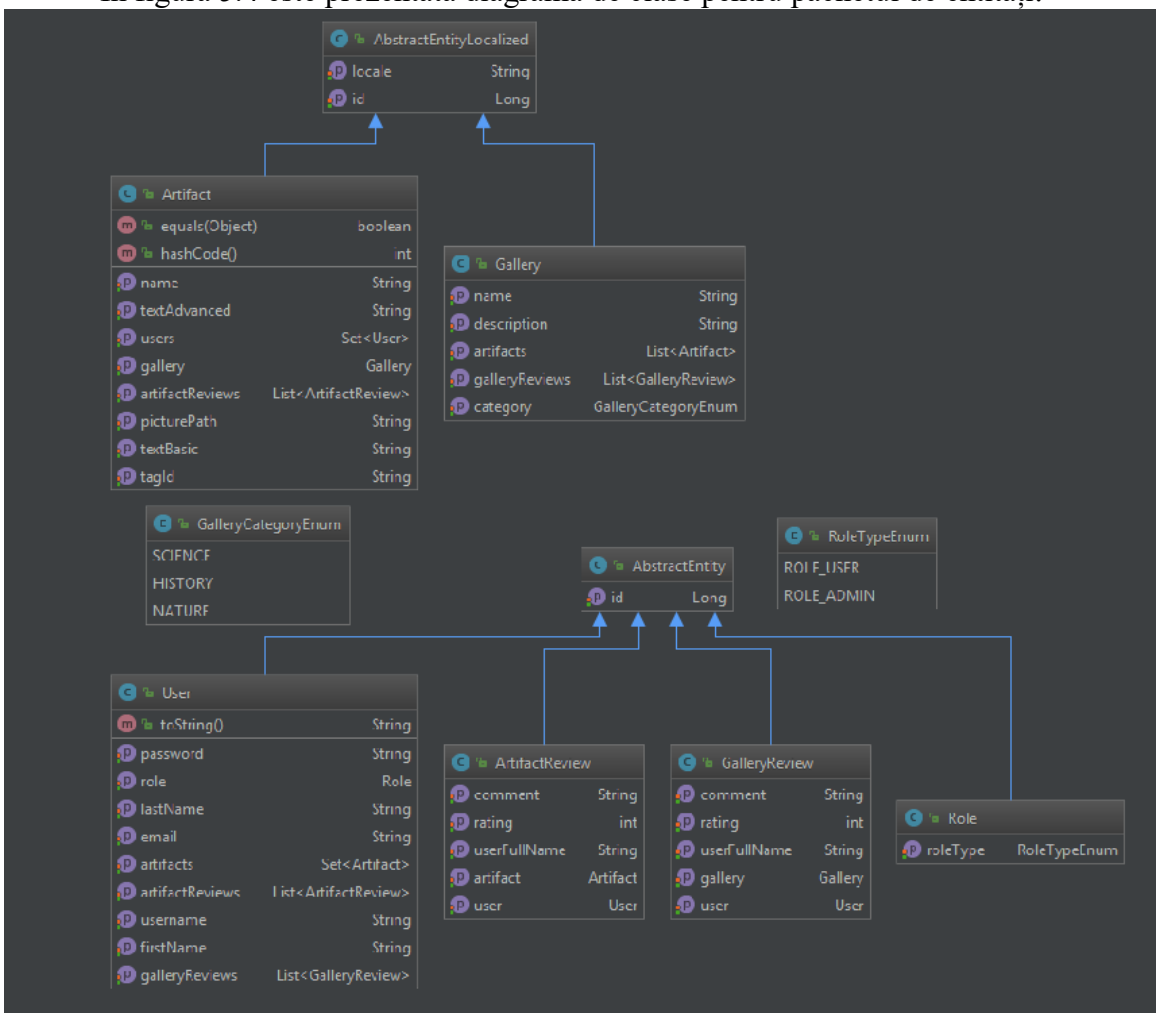


Figura 5.5 Diagrama de clase din cadrul pachetului de entități

5.2.3. Control

Partea de control a aplicației este cea mai complexă deoarece reprezintă un mediator între celelalte două componente. În cadrul aplicației, partea de control este și ea împărțită în trei componente principale. Diagramă de clase este împărțită în 3, pentru fiecare dinre subpachetele importante.

5.2.3.1. DAO-uri (Data Acess Objects)

DAO-urile reprezintă interfețe abstracte peste diferite modele de baza de date ce expun aplicației serviciile necesare pentru accesul la baza de date, fără a expune însă detaliile de implementare a bazei de date și modul în care operațiile asupra ei sunt realizate. Aceste tipuri de interfețe respectă și principiul de Single Responsibility deoarece separă datele de care are nevoie aplicația în termeni de tipuri de date și tipuri de obiecte specifice pentru un domeniu de modul în care aceste cerințe sun satisfăcute indiferite sisteme de baze de date.

Pentru implementarea acestor interfețe am folosit librăria de Spring Dată JPA. Modul de implementare este următorul:

1. Se declara o interfață fără niciun antet de metode, adnotată cu @Repository
2. Interfață creată este modificată astfel încât să extindă interfață JpaRepository<T, ID extends Serializable> pusă la dispoziție de către librărie
3. Parametri generici prezenți sunt: T – trebuie înlocuit cu numele entității asupra căreia se vor realiza operații; ID – trebuie înlocuit cu tipul de dată a cheii primare din entitatea respectivă
4. În urmă realizării pașilor de mai sus, la runtime spring va construi o implementare automată a acestora și va pune la dispoziție operații CRUD de baza asupra entității fără a fi neovie că acestea să fie declarate și implementate de către utilizator.

Bineînțeles, vor fi necesare și operații mai complexe asupra bazei de date. Acest lucru se realizează prin adăugarea în interfață declarată a unor antete de metode. Mai jos este prezentat un exemplu luat din aplicația dezvoltată, care realizează selecția tuturor artefactelor dintr-o galerie:

List<Artifact> getArtifactsByGalleryId(Long galleryId);

Prin simplă adăugare a acestui antet de metodă, se va realiza automat o implementare a unei metode ce selectează din tabela de artefacte toate artefactele care au galleryId egal cu id-ul galeriei dat că parametru. Numele filed-ului după care se face selecția este luat din numele metodei. Spring Dată JPA pune la dispoziție funcționalități de code completion în momentul în care programatorul dorește să implementeze aceste metode.

Totuși, există cazuri în care este nevoie de operații și mai complexe. Acest lucru este realizat prin implementarea de NamedQueries, care reprezintă obiecte ce au în principal în componentă un nume și un query. Query-ul reprezintă o abstractizare a limbajului MySQL și, în funcție de ORM-ul folosit este denumit fie HQL(Hibernate Query Language) fie JPQL(Java Persistence Query language). Diferența între HQL/JPQL și MySQL este faptul că HQL și JPQL sunt qury languages OOP. Pentru a înțelege mai bine acest lucru, mai jos sunt date două exemple(unul HQL iar celălalt MySQL) care realizează același lucru:

MySQL: SELECT * from Entity where Entity.id = 1;
HQL: SELECT e FROM Entity e WHERE e.id = 1;

În figura 5.5 este prezentată digrama de clase din pachetul DAO.

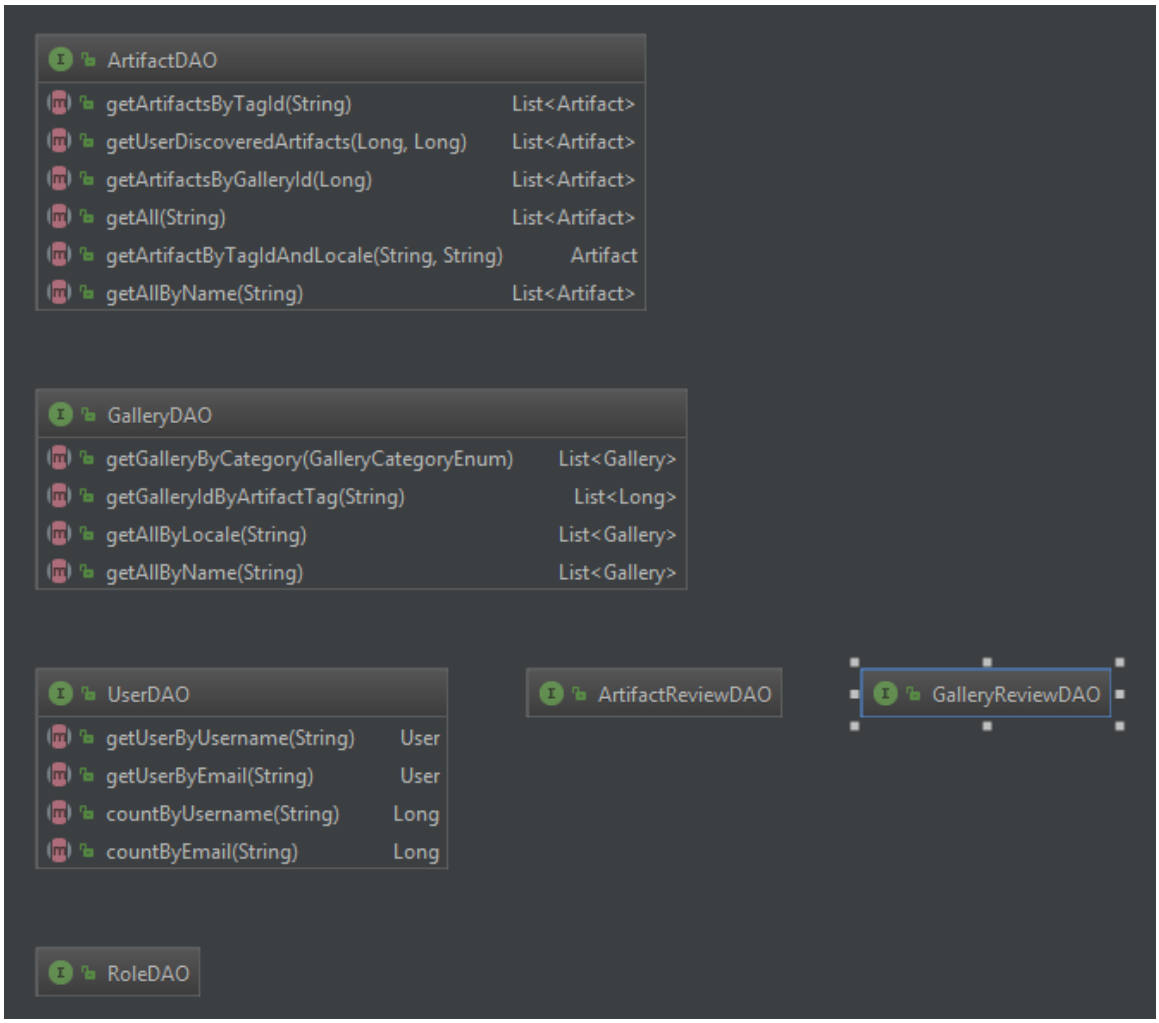


Figura 5.6 Diagrama de clase din pachetul DAO

5.2.3.2. DTO-uri(Data Transfer Obects)

Un Dată Transfer Object este un obiect ce înglobează doar date (fără logică de business), și uneori mecanisme de serializare/deserializare. Motivația folosirii acestor tipuri de obiecte este faptul că, în mod normal operațiile de transfer de date sunt de obicei costisitoare. Aceste obiecte înglobează date din mai multe operații, ceea ce reduce costul operațiilor.

În cadrul aplicației dezvoltate, DTO-urile sunt reprezentări în oglindă a entităților din clasele de model. Am folosit conceptul de DTO din cauza faptului că obiectele de tip entity din pachetele de model nu ar trebui să fie prezente în cadrul layer-ului de control. Maparea între aceste tipuri a fost realizată cu librăria MapStruct. Un alt motiv pentru care am folosit obiecte de tip DTO este faptul că a fost necesară restricționarea unor fileduri în cadrul proceselor de serializare/deserializare, iar realizarea acestui lucru în cadrul obiectelor de model ar fi dus la încălcarea principiului de Single Responsibility care rezultă implicit în inconsistența cerinței nonfunctionale de mentenabilitate.

Diagrama de clase pentru DTO-uri este prezentata in figura 5.6.

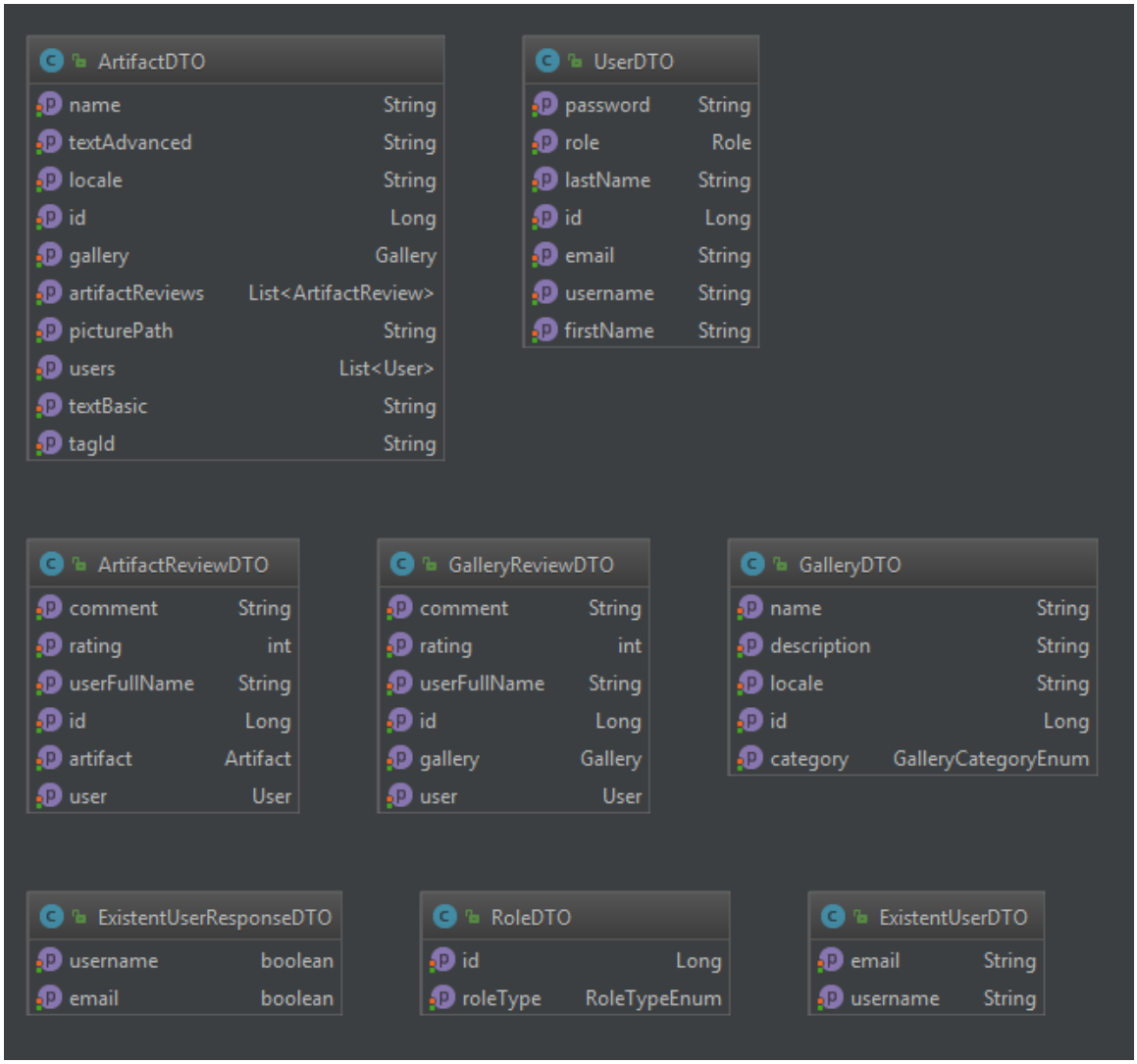


Figura 5.7 Diagrama de clase pentru pachetul DTO

Pentru utilizarea librăriei MapStruct este necesară declararea unor interfețe adnotate cu @Mapper (pusă la dispoziție de MapStruct) care să conțină o variabilă de instanța care va returna o instanța a uni mapper și două antete de metode care oferă posibilitatea conversiei din DTO in entity și invers. La runtime va fi realizată automat o implementare a acestor interfețe.

În figura 5.7 este prezentată diagramă de clase pentru pachetul de mappere.

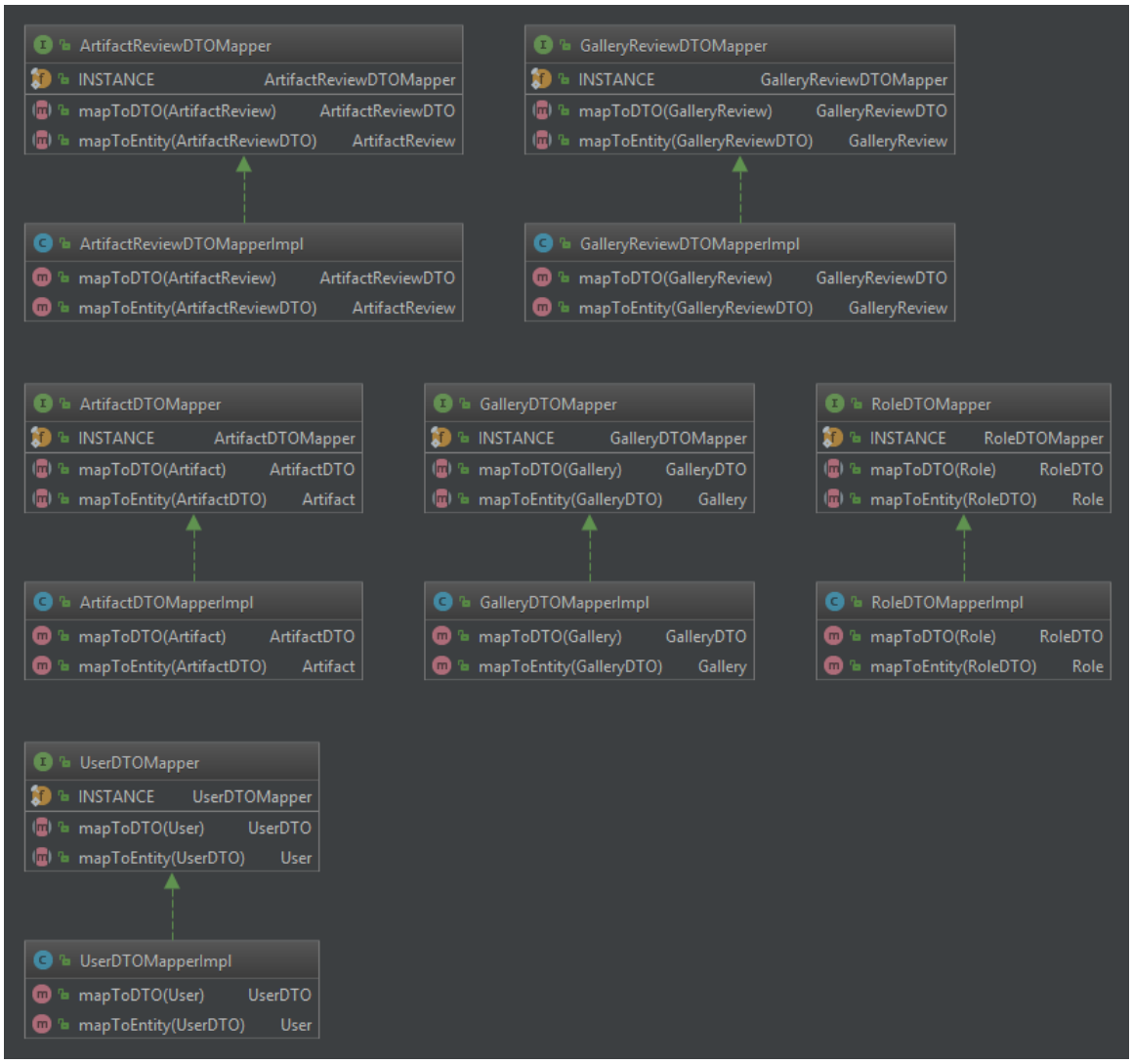


Figura 5.8 Diagrama de clase pentru pachetul mapper

5.2.3.3. Services

Pachetul de services nu face referire la servicii web, ci la logică de business necesară în cadrul aplicației. Fiecare clasa din cadrul acestui pachet este adnotată cu @Service. În cadrul acesor clase vor fi injectate obiecte de tip DAO cu adnotarea @Autowired. Clasele conțîn metode care primesc din partea de boundary obiecte de tip DTO, le convertesc în obiecte de tip entity cu mapperele descrise mai sus, după care sunt folоiste pentru operații asupra bazei de date prin obiectele de tip DAO. Totodată, există și operații care primesc id-uri sau alte date din partea de boundary și le folosesc pentru a aduce entități din baza de date, le convertesc în DTO-uri și le transmit mai departe către boundary. Metodele din servicii sau chiar clasele întregi pot fi adnotate cu @Transactional pentru că operațiile din cadrul metodei/clasei să fie atomice (dacă o parte a tranzacției eșuează toată tranzacția este anulată pentru a nu strică consistentă bazei de date).

In figura 5.8 este prezentata diagrama de clase a pachetului service.

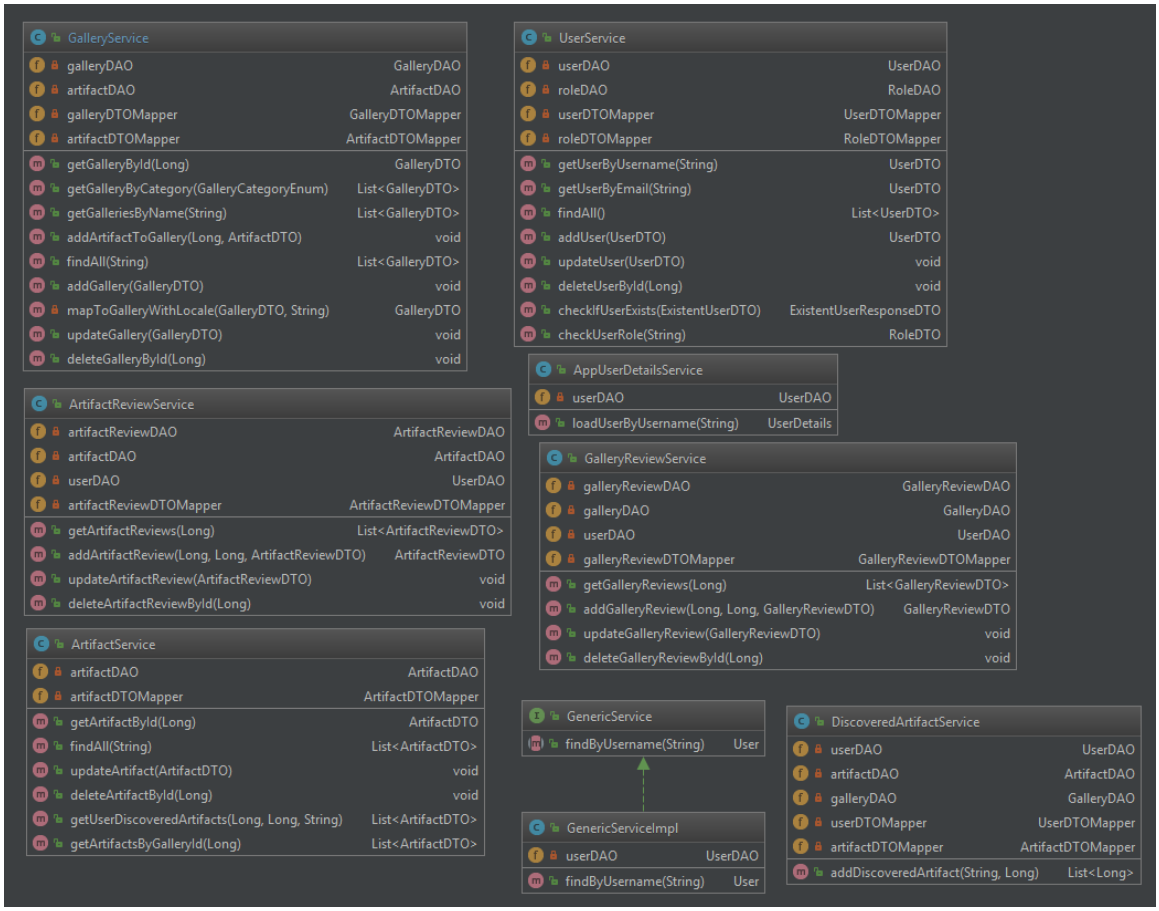


Figura 5.9 Diagrama de clase din pachetul service

5.2.4. Boundary

Partea de boundary este cea care realizează legătură între aplicația propriu-zisă și consumatorii de resurse externi. Este o componentă a cărei rol este de a prelua informații din afară și de a le transmite către servicii și de a prelua ceea ce transmit serviciile și a le transmite afară.

În cadrul aplicației, partea de boundary expune servicii REST pentru comunicare cu exteriorul. Conține instanțe injectate de către Spring a serviciilor de business necesare din componentă de controller.

Este locul în care se specifică ce metodă web este folosită pentru comunicare pentru fiecare operație (GET, POST, PUȚ), URL-ul prin care se poate realiza comunicarea și configurarea de securitate Ouath2 pentru fiecare REST endpoint pentru a restricționa accesul al diferite resurse în funcție de rolul utilizatorului care încearcă să folosească aceste resurse.

Figura 5.9 conține diagramă de clase pentru pachtul de boundary.

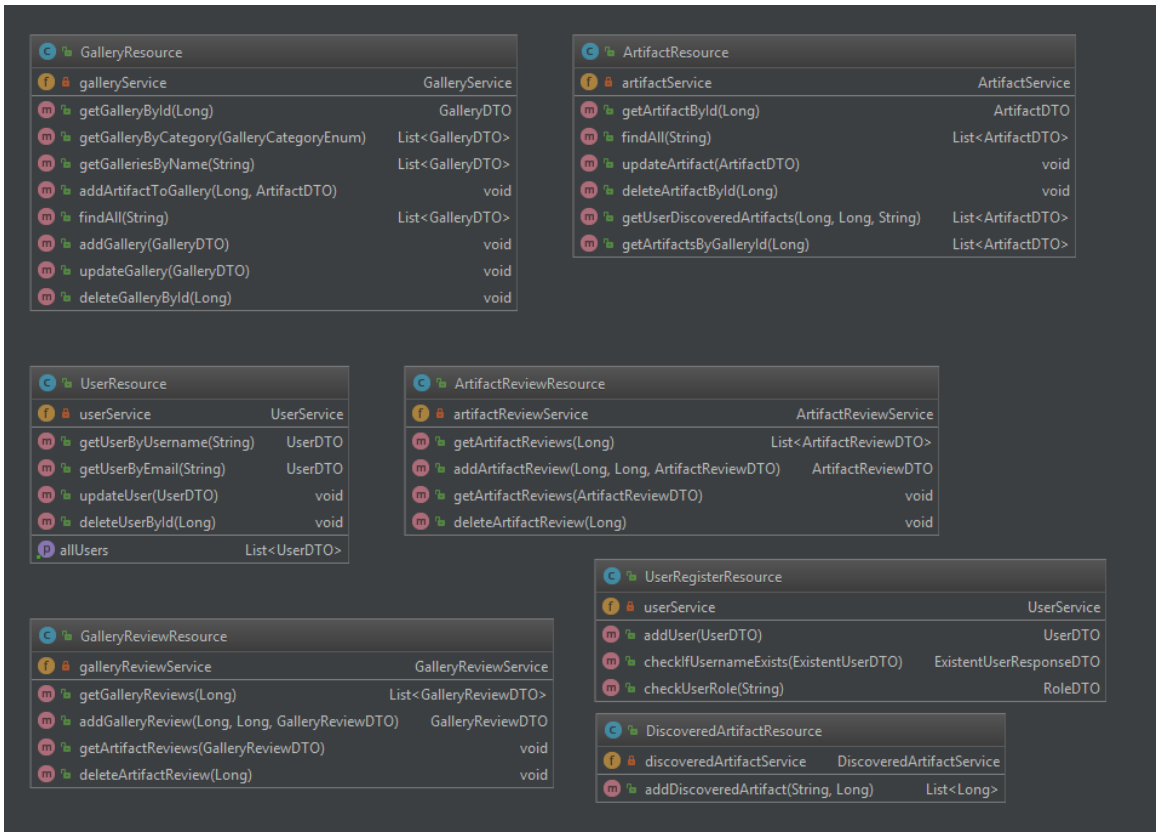


Figura 5.10 Diagrama de clase pentru pachetul boundary

5.2.5. Config

Pachetul contine clasele de configurari din cadrul aplicatiei de backend.

- **Hibernate Configuration** - Reprezintă o incapsulare a configurarilor pentru Hibernate cum ar fi limbajul de baze de date folosit, executare de query-uri inițiale, parametri de conexiune la baza de date etc.
- **Cross Origin Resource Sharing Filter** - Reprezintă un filtru ce adaugă fiecărui request anumite headere pentru a permite comunicarea aplicației cu aplicații externe.
- **Security Configuration** - Reprezintă configurări de securitate cum ar fi modul de criptare a parolelor și tipul providerilor tokenelor de acces la resurse pentru utilizatori.
- **Resource Server Configuration** - Specific protocolului Oauth2, realizează operații precum configurarea accesului la endpoint-urile aplicației și verificarea timpului de viață rămas a unui token de acces.
- **Authorization Server Configuration** - Este entitatea care realizeaza configurarea si eliberarea tokenelor de acces la resurse.

În figura 5.10 este prezentată diagramă de clase din pachetul de configurări.

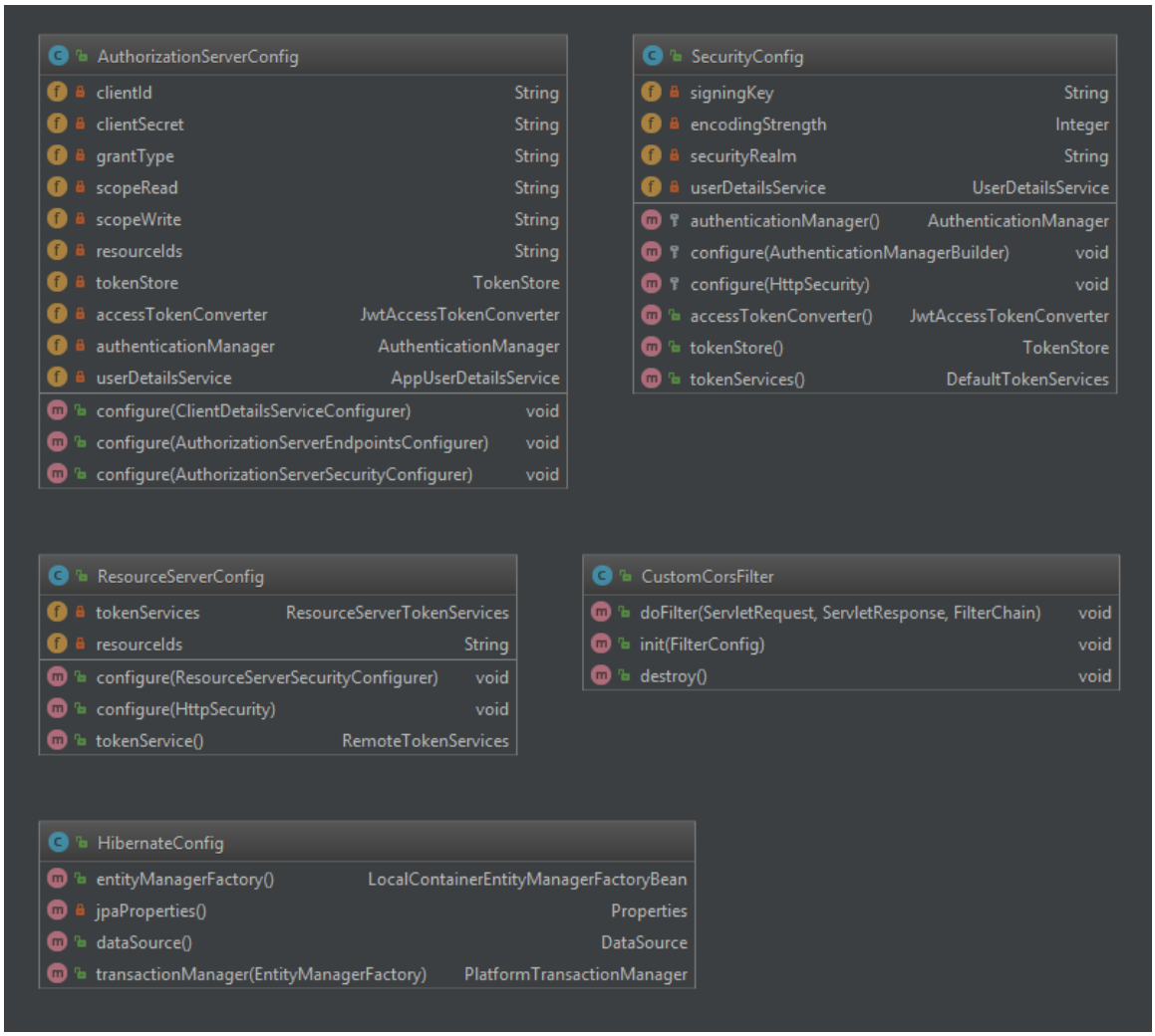


Figura 5.11 Diagrama de clase din pachetul de configurări

5.3. Aplicația Android

Descrierea aplicației Android va fi realizată în ordinea cronologică a implementării functionalitatilor.

5.3.1. Înregistrare utilizator

Layout-ul acestei activități este realizat folosind Relative Layout pentru a avea posibilitatea de a așeza elementele relativ la celelalte din același părinte.

După completarea input-urilor și apăsarea butonului de „Sign Up” se execută următoarele:

1. Mai întâi se verifică dacă filed-urile „Password” și „Repeat password” sunt la fel. Sunt necesare două astfel de field-uri pentru a se asigura a utilizatorul nu introduce greșit parolă pe care dorește să o folosească. Dacă aceste două parole nu se potrivesc, va apărea un mesaj de eroare.
2. Se testează dacă celelalte constrângeri din partea de aplicație sunt îndeplinite (filed-uri required, număr minim de caractere, pattern de email corect etc.).

3. După realizarea acestor validări se va testa dacă nu cumva email-ul sau username-ul introduse sunt deja folosite de către alt utilizator, caz în care se afișează un mesaj de eroare.
4. Dacă toate constrângerile de mai sus sunt îndeplinite, se construiește un request de înregistrare utilizator now cu datele introduse.
5. Dacă utilizatorul este înregistrat cu succes, se declanșează o acțiune automată de logare a utilizatorului în aplicație care salvează datele utilizatorului logat în containerul de SharedPreferences pus la dispoziție de sistemul de operare Android. Acest container are rolul de sesiune care ține salvate datele utilizatorului atâta timp cât acesta este logat în aplicație.

În figura 5.10 este prezentată diagramă de secvență pentru înregistrare utilizator.

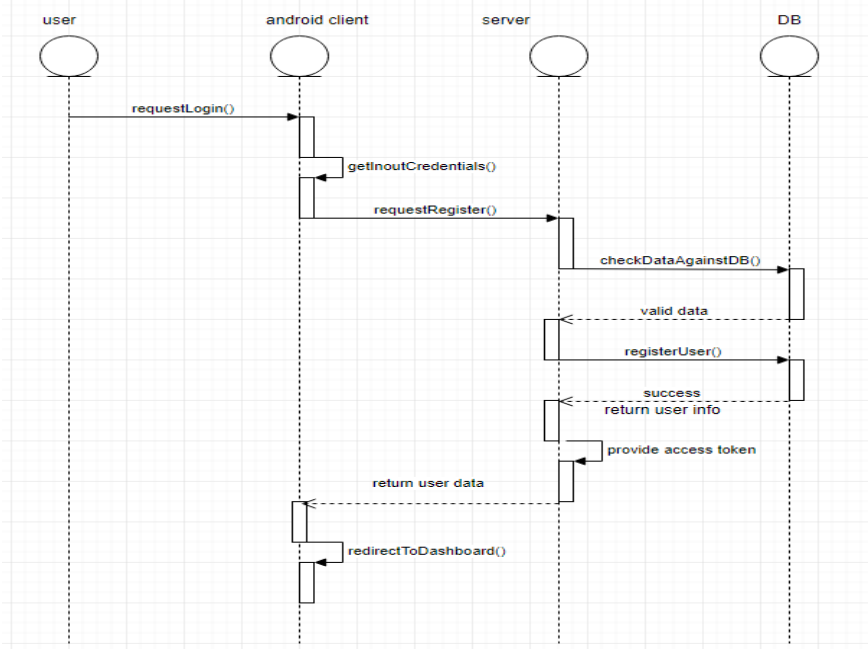


Figura 5.10 Diagrama de secvență pentru înregistrare utilizator

5.3.2. Logare utilizator

Activitatea de logare utilizator este construită în mod analog cu cea de înregistrare, dieferenta fiind faptul că sunt necesare doar fild-urile de username și password pentru acces în aplicație. În figura 5.11 este prezentată diagramă de secenva pentru cazul în care datele introduse sunt valide și utilizatorul se loghează cu succes.

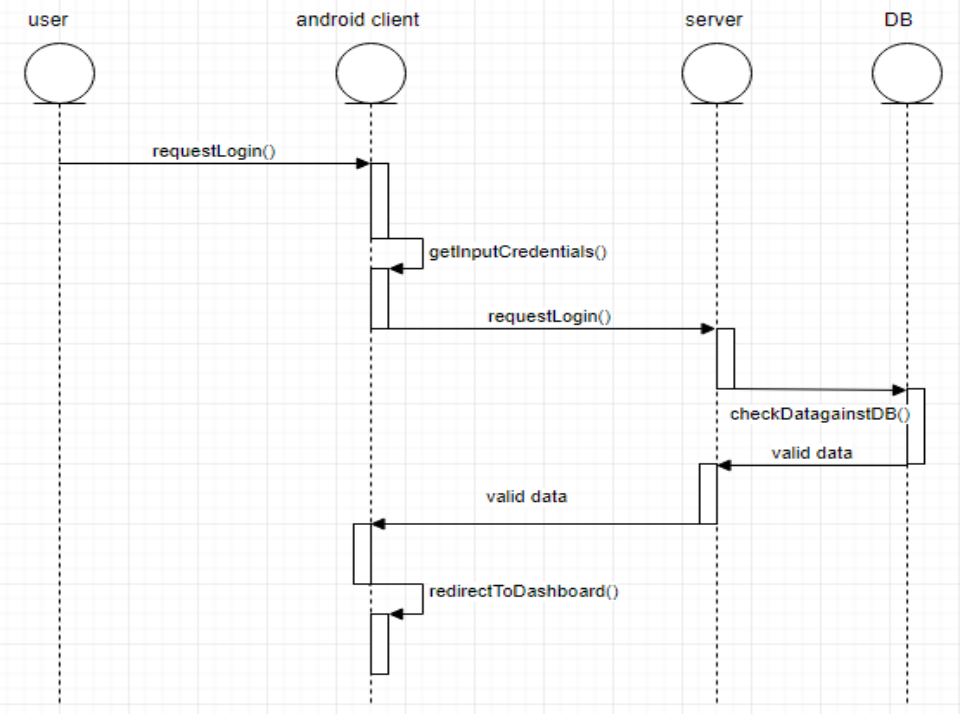


Figura 5.12 Diagrama de secvență pentru o operațiune de logare cu succes

În urmă completării filed-urilor și a apăsării butonului de login, se realizează un request de token de acces. Mesajele de eroare sunt construite în funcție de codul de răspuns sau tipul de eroare returnată în urmă tirmiterii acestui request (400 – bad request => username sau parolă greșită, 5xx – eroare de server etc.). Dacă răpunsul este „OK” (status code 200), se preia tokenul de acces de pe response body și este salvat în Shared Preferences pentru utilizatorul curent. Acest token este folosit pentru a autoriza accesul la anumite resurse puse la dispoziție de partea de backend. După ce tokenul de acces a fost preluat, utilizatorul este logat în aplicație, iar datele sale sunt și ele salvate în Shared Preferences, acesta fiind apoi redirecționat către activitatea principala a aplicației.

5.3.3. Activitatea de dashboard

Această este principala activitate a aplicației. Este locul din care utilizatorul poate să acceseze toate celelalte funcționalități prezente în aplicație: Vizualizare liste galerii, vizualizare harta muzeu, căutare puncte de interes, schimbare limba de afisre aplicație și delogare utilizator.

5.3.4. Delogare utilizator

În urmă apăsării acestui buton datele utilizatorului din Shared Preferences vor fi șterse iar acesta vai fi redirecționat către activitatea de login. Deschiderile succesive ale aplicației vor redirecționa utilizatorul către activitatea de login dacă acesta nu se logheaza. După logare, deschiderile succesive îl vor redirecționa direct pe activitatea de dashboard.

5.3.5. Puncte de interes

In cadrul acestei activitati, utilizatorul este capabil sa caute diferite alte puncte de interes din apropiere (hoteluri, restaurante, baruri, muzee etc.) pe o raza aleasa de el, fie prin comanda vocala sau prin selectarea unui anumit tip de locatie prin butoanele puse la dispozitie. In figura [TODO: add picture] este prezentata aceasta activitate.

Partea de comandă vocală este implementată cu ajutorul API-ului de Speech Recognition din Android. La prima utilizare a acestei funcționalități utilizatorului i se cere accesul la înregistrarea sunetului cu ajutorul microfonului dispozitivului. În urmă acceptării, utilizatorul poate să își folosească vocea să găsească punctele de interes dorite din apropiere. În urmă căutării, un set de markere sunt adăugate pe harta, iar atingerea lor face posibilă afișarea de informații cum ar fi numele locației și adresa la care se găsește.

5.3.6. Setări

Activitatea de setări a aplicației poate fi folosită de către utilizator pentru a schimbă limba aplicației. În momentul actual aplicația suportă limbile română și engleză. Din cauza faptului că nu este un good practice schimbarea limbii doar într-o aplicație, implementarea curentă redirecționează utilizatorul către setările telefonului de unde poate să schimbe limba, iar la reîntoarcerea în aplicație acesta va observă că și limba aplicației a fost schimbată. Selectarea unei limbi nesuportate de aplicație va avea că și rezultat faptul că aplicația își va păstra limba default (engleză). Pentru realizarea internaționalizării aplicației am stocat toate denumirile în fișierul strings.xml din cadrul resurselor proiectului deoarece pune la dispoziție duplicarea acestora în momentul în care dorim că aplicația să fie disponibilă în mia multe limbi. Aceste denumiri sunt scrise folosind xml în felul următor:

<string name=”identificator”>content</string>

<string name=”identificator”>continut</string>

Prin folosirea acestui stil de implementare, putem folosi identificatorii pentru stringurile pe care le dorim, iar la detectarea schimbării de limba, aplicația va folosi automat strîng-urile din fișierul corespunzător.

În figura 5.13 este prezentată o diagramă de secvență pentru vizualizarea etapelor ce au loc când utilizatorul dorește să schimbe limba telefonului.

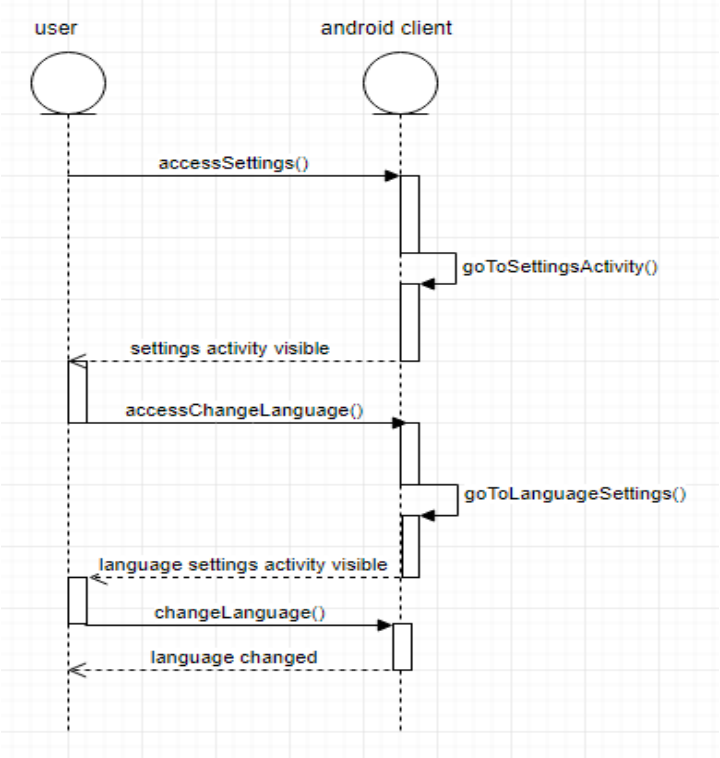
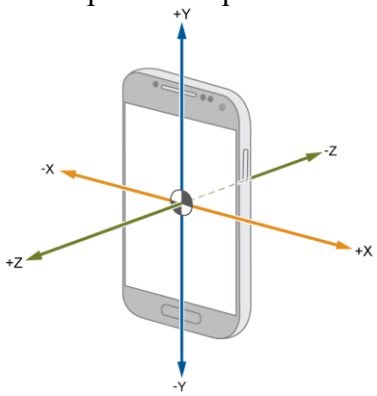


Figura 5.13 Diagramă de secvență pentru schimbarea limbii de afișare a aplicației

5.3.7. Harta muzeului

Harta muzeului oferă utilizatorului informații în timp real despre unde se află în interirul unei galerii, bazate pe ceea ce scanează acesta și pe senzorii de care dispozitivul mobil dispune.

Detecția mișcării utilizatorului s-a făcut cu ajutorul accelerometrului de care dispozitivul mobil dispune. Senzorul returnează un vector de dimensiune 3, fiecare valoare reprezentând accelaratia în m/s^2 pe una dintre cele 3 axe de coordonate. În figura 5.14 este prezentată poziționarea celor 3 axe în raport cu dispozitivul mobil.



[33]

Figura 5.14 Orientarea axelor în raport cu telefonul mobil

[33] https://www.mathworks.com/help/supportpkg/android/ref/accelerometer.html

Valorile returnate în acest vector au în componentă și accelerația gravitațională a Pământului, deci pentru o acuratețe bună a detectiei mișcării a fost nevoie eliminarea acestei componente. Acest lucru s-a realizat prin aplicarea unui filtru trece-jos asupra datelor returnate de către accelerometru. Prin aplicarea unui filtru trece-jos putem să izolăm componentă de gravitație din valorile returnate de senzor. Valoarea reală a accelratiei liniare se obține ulterior prin scăderea valorii gravitației din fiecare valoare returnată de senzor. Formulă pentru calcularea componenței gravitaționale este prezentată mai jos:

$Gravity^{XYZ}_{i} = ALPHA * Gravity^{XYZ}_{(i-1)} + (1 - ALPHA) * SensorValues^{XYZ}$

ALPHA – constanța derivată din constanța de timp ce reprezintă latentă introdusă de filtru la evenimentele care returnează valorile și la rată de returnare a valorilor de către senzor.

După ce avem aceste valori ale gravitației, valorile accelerației liniare pe fiecare axa sunt obținute prin scăderea din aceste valori a valorii gravitației pe axa respectivă. Pe baza accelerației putem determina dacă utilizatorul se mișcă sau nu.

Pentru determinarea orientării în spațiu a fost folosit senzorul de tip GAME_ROTATION_VECTOR care returnează orientarea dispozitivului că o combinație de un unghi și o axa în jurul căreia dispozitivul s-a rotit.

După ce aceste valori sunt disponibile, le putem combină astfel încât să simulăm mișcarea într-un plan bidimensional. În momentul în care se detectează o mișcare, estimăm dimensiunea medie a pasului unui utilizator, și calculăm poziția în planul XY astfel:

Pos_X += DimensiunePas * sin (unghiDeRotatie)
Pos_Y += DimnesiunePas * cos (unghiDeRotatie)

Cele două valori de mai sus au fost folosite pentru updatarea poziției utilizatorului în momentul în care acesta se mișcă.

5.3.8. Galerii

Accesarea activitatii de galerii redirectioneaza utilizatorul catre o lista cu galeriile disponibile in muzeu.

Prin apăsarea lungă pe una dintre galerii, utilizatorul va fi redirecționat către o activitate în care poate vizualiza detaliile despre galerie în format text și audio, va putea vizualiza review-urile lăsate de către ceilalți utilizatori pentru galeria respectivă, și va putea și el la rândul sau să lase un review pentru galeria respectivă.

Activitatea folosește un ListView pentru afișarea dinamică a unei liste și un adapter pentru a „adapta” tipul de date ce trebuie afișat în lista (în cazul acesta date de tip Gallery).

Funcționalitatea de format audio pentru descrierea galeriei folosete API-ul de Text-to-Speech de la Google. Pentru realizarea acestui lucru sunt necesari următorii pași:

1. Se face un request la URL-ul pus la dispoziție de API în care se transmite și un request body de formă:

```
{
   "input":{
         "text":" Textul care urmează a fi sintetizat în format audio."
    },
   "voice":{
```

```
        "languageCode":"en-en",
        "name":"en-US-Standard-A",
        "ssmlGender":"FEMALE"
    },
        "audioConfig":{
        "audioEncoding":"MP3"
    }
}
```

Observăm că putem să alegem limba dorită, genul vocii rezultat și formatul fișierului.

2. În urmă trimiterii acestui request vom primi un răspuns simplu de tip JSON care va conține o varianta encodata în baza 64 a fișierului audio, iar nouă nu ne ramanae decât să decodam acest text.
3. Decodarea acestui format a fost făcută folosind utilitarul Base64 din Android. În urmă realizării operației de decodare ne rezultă un vector de bytes care poate fi scris într-un fișier cu extensia dată de noi în request, în acest caz mp3.
4. Nu ne mai rămâne decât să folosim player-ul audio inclus în SDK-ul Android pentru a redă fișierul respectiv.
5. Pentru a evita umplerea memoriei telefonului, fișierul este șters automat după ieșirea utilizatorului din aplicație.

În momentul actual, API-ul de text-to-speech este în versiunea beta și nu suportă decât aproximativ 20 de limbi, în care română nu este inclusă, doar engleză.

Pe lângă simplul player pentru a redă sunetul, am folosit și un SeekBar pentru că utilizatorul să poate să se întoarcă la o anumită parte din explicația audio fără a fi nevoie să asculte de la început.

Dacă utilizatorul alege să atingă scurt o galeire din lista, acesta va fi redirecționat către activitatea cu artefactele descoperite de acesta din galeria respectivă.

În figura 5.15 este prezentată o diagramă de secvență pentru obitinerea și redarea de informații audio despre artefact.

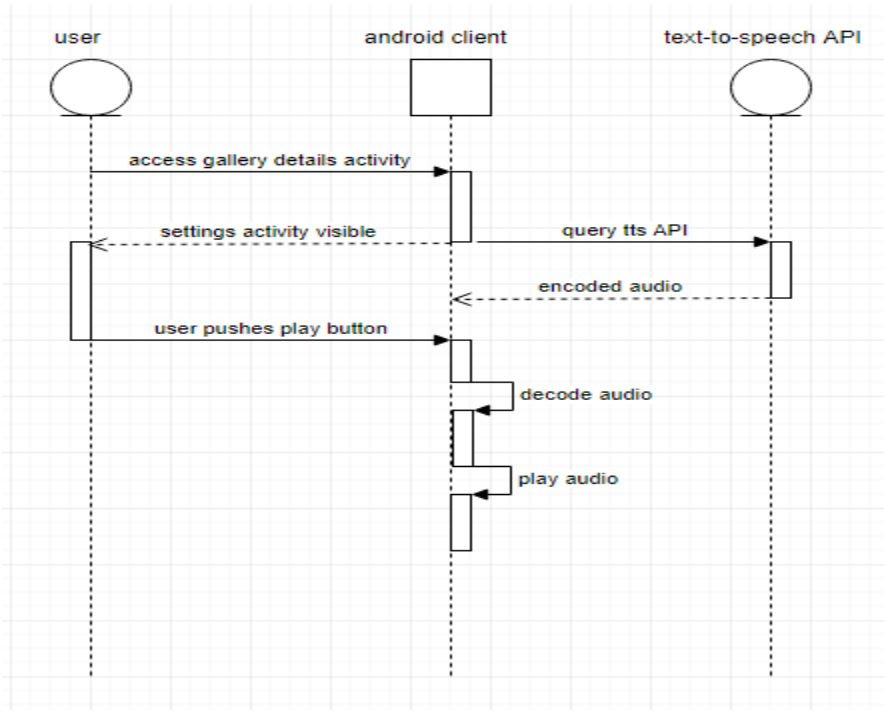


Figura 5.15 Diagramă de secvență pentru obținerea informațiilor despre galerie în format audio și redarea acestora pe dispozitivul mobil

5.3.9. Artefacte

În urmă scanării unui tag NFC, artefactul corespunzător tag-ului va fi adăugat în lista cu artefacte descoperite a utilizatorului curent, iar utilizatorul va fi redirecționat către galeria din care face parte artefactul nou descoperit.

Activitatea cu lista artefactelor este construită în aceeași manieră că și cea pentru lista galeriilor.

În cadrul activității de detalii a artefactelor, utilizatorul poate face aceleași acțiuni că și în cadrul activității de detalii pentru galerii. O funcitonalitate în plus este faptul că utilizatorul poate selecta care dintre explicații (de baza sau avansată) să fie afișate și redate audio, în funcție de preferințe. Acest lucru este realizat printr-un buton de tip toggle, care va schimbă între ele explicațiile la fiecare apăsare.

În figura 5.16 este prezentată diagramă de secvență pentru scanarea unui tag NFC.

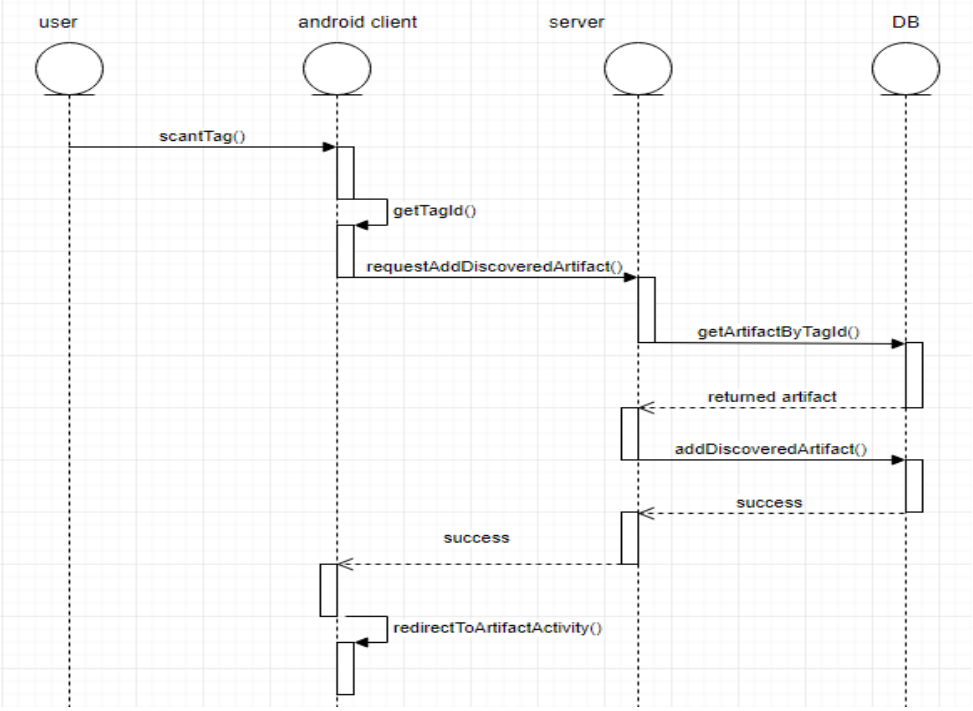


Figura 5.13 Diagrama de secventa pentru scanarea uni tag NFC

5.4. Aplicatia Web pentru administrator

Partea de web a sistemului este dedicată doar pentru utilizatorii de tip administrator. Scopul ei este de a facilita operațiile de management pe partea de baza de date care stochează informații cu privire la galeriile și exponatele din cadrul muzeului. În principal, din interiorul aplicației un administrator este capabil să realizeze operații CRUD pe galerii și pe artefacte.

În figura este prezentată diagramă de secvență pentru adăugarea unei noi galerii.

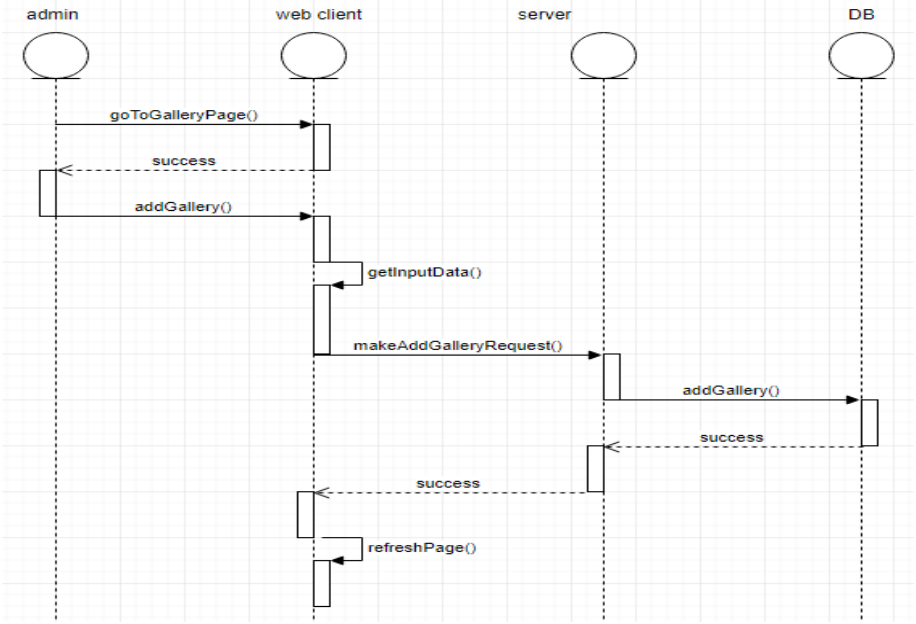


Figura 5.17 Diagramă de secvență pentru adăugarea unei noi galerii

Capitolul 6. Testare şi Validare

Obiectivul acestei secțiuni este de a prezența modalitățile de testare și de validare folosite pentru fiecare modul în decursul dezvoltării fiecăruia.

Procesul de testare a fost realizat împreună cu cel de implementare, evitând astfel volumul mare de probleme/bug-uri ce apăreau dacă testarea era făcută ulterior implementării.

6.1. Testare bază de date și backend

Testarea acestor două modul a fost realizată mâna în mâna deoarece endpoint-urile REST oferă o viziune foarte bună asupra modului de functioare a acestor două module. O prima metodă de testare ar putea fi utilizarea browserului pentru a introduce endpoint-urile REST. Acest lucru nu este practice însă deoarece în această manieră se pot testa doar metodele de tip GET, celelalte necesitând adăugarea de request body sau headere suplimentare pentru funcționare. Acest lucru a fost realizat prin folosirea tool-ului Postman, care oferă suport pentru testarea diferitelor tipuri de metode.

În figura 6.1 este prezentată utilizarea tool-ului Postman pentru testarea aducerii din baza de date a utilizatorului cu username-ul “cătălin”.

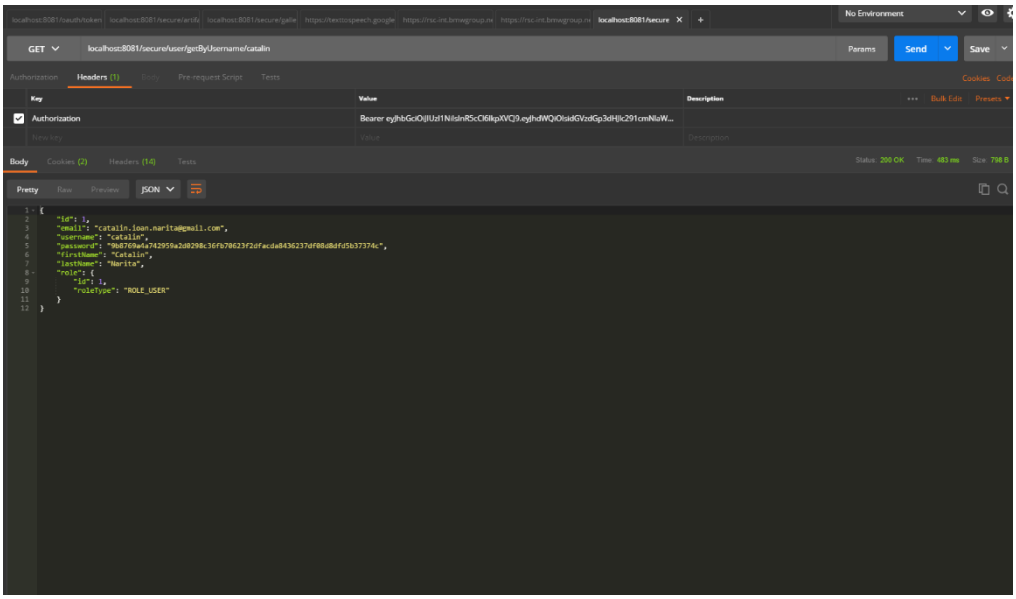


Figura 6.1 Testarea aducerii din baza de date a utiliatorului cu username-ul „cătălin”

După realizarea testării părții de backend și a bazei de date, putem testa atât părea de Android cât și partea de web ținând cont de faptul că serviciile REST funcționează correct, ușurând astfel procesul de debugging realizat în momentul in care o funcționalitate din partea de client da un alt rezultat decât cel așteptat.

6.2. Testarea aplicatiei web

Testarea aplicației web se poate face foarte ușor prin intermediul consolei pusă la dispoziție de browser, dar și prin simplul fapt că putem să ne dăm seama dacă

funcționalitatea da rezultate corecte datorită faptului că aceste rezultate sunt afișate în pagină web. În figura 6.2 este prezentată lista de galerii disponibile în baza de date înaintea adăugării uneia noi.

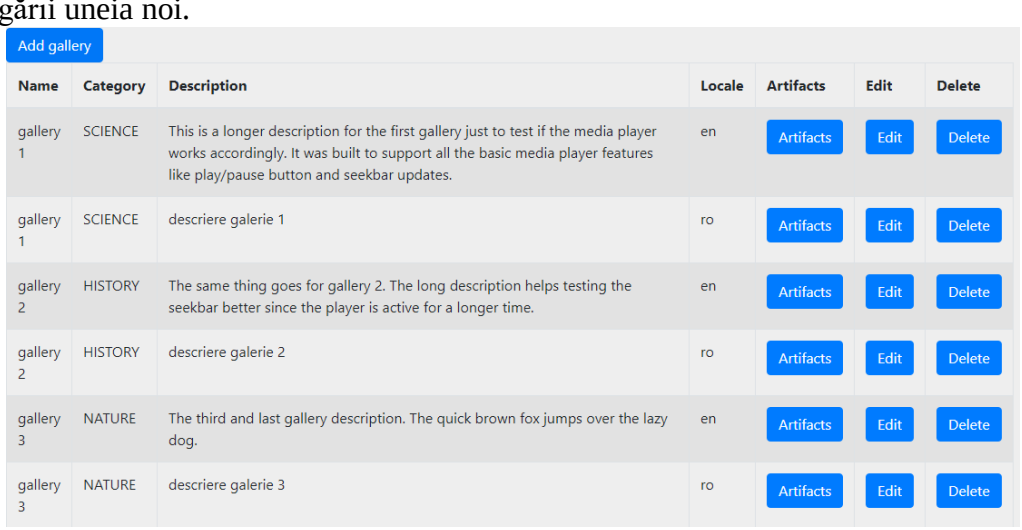


Figura 6.2 Lista de galerii înainte de adăugare

În figura 6.3 sunt prezentate cele două JSON-uri cu galleria nouă (unul pentru română, unul pentru engleză) ce urmează a fi trimise către backend.

Figura 6.3 JSON-urile corespunzătoare pentru galeria ce urmează a fi adăugată

In figura 6.4 este prezentata lista de galerii dupa adaugarea uneia noi.

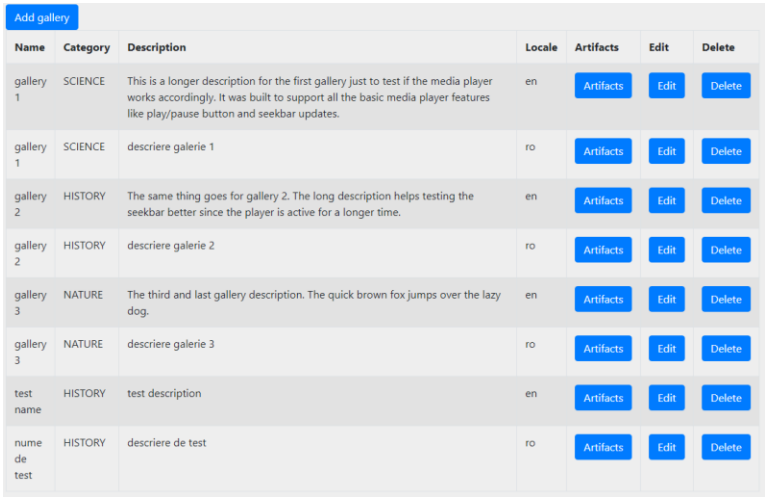


Figura 6.4 Lista de gallerii după adăugarea uneia noi

În cadrul dezvoltării aplicației web, s-a folosit și metodă de debugging direct în browserul web, pentru a analiză pas cu pas executarea unei anumite funcționalități.

În imaginea 6.5 este prezentat debugging-ul funcționalității de adăugare galerie nouă.

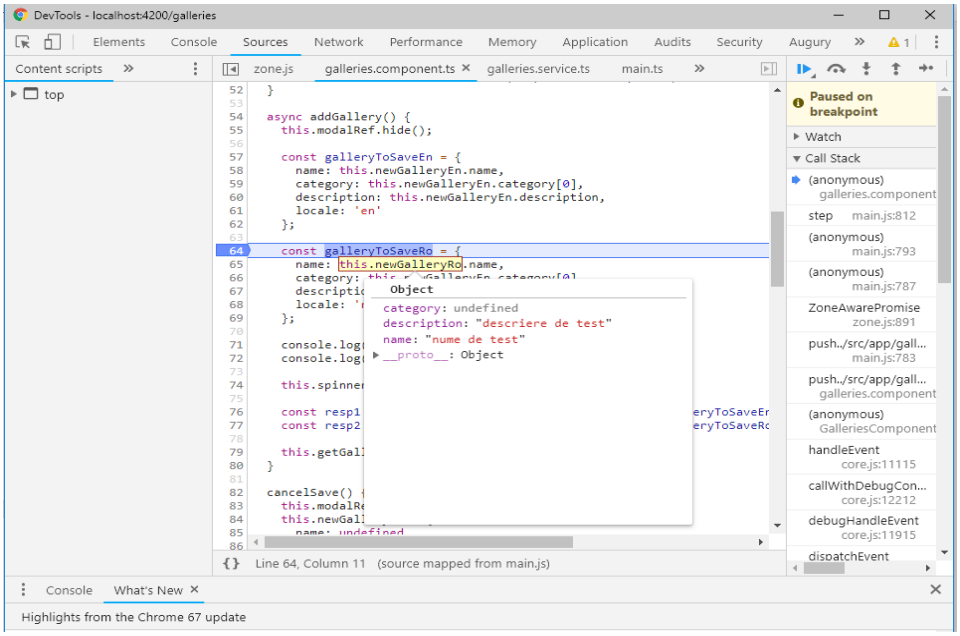


Figura 6.1 Debugging-ul functionalitatii de adaugare galerie noua

6.3. Testarea aplicației Android

Testarea aplicației Android s-a făcut în mod analog cu testarea părții de web, fiind însă utilizate log-urile disponibile în android și interfață de pe dispozitivul mobil. Acțiunile care nu au rezultatul droit vor fi urmate de dialoguri care spun ce s-a întâmplat și sugerează o soluție pentru evitarea aceste probleme.

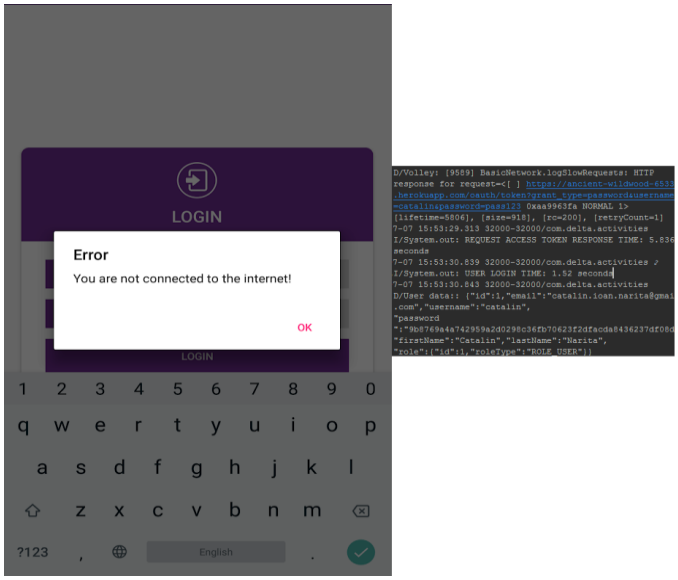


Figura 6.2 Mesaje de eroare și un log în urmă logarii utilizatorului

În imaginea 6.6, în partea stânga este prezentat un mesaj de eroare din cauza faptului că utilizatorul nu are conexiune la internet. În partea dreaptaputem vedea un log cu datele utilizatorului din baza de date în urmă acțiunii de logare a acestuia.

Capitolul 7. Manual de Instalare și Utilizare

Acest capitol urmărește precizarea resurselor necesare și definirea unui set de instrucțiuni ce sunt necesare pentru că un utilizator să fie capabil să instaleze și utilizeze aplicația dezvoltată.

7.1. Resurse necesare

Din cauza faptului că s-a optat pentru soluții cloud în cazul bazei de date și a părții de backend, acestea vor fi disponibile oricând, singurele resurse necesare fiind accesul la resursele virtuale necesare.

Pentru aplicația Android sunt necesare următoarele:

- Un telefon cu sistem de operare Android
- Conexiune la Internet (wifi sau date mobile)
- Un spațiu de stocare de aproximativ 11-12 Mb
- Un cablu prin care dispozitivul mobil poate fi conectat la un computer

Pentru partea de web a aplicației este necesară utilizarea unui calculator pe care rulează platforma Node.js și platforma Angular.

7.2. Instalare aplicație

7.2.1. Instalarea aplicației Android

Pentru instalarea aplicatiei Android sunt necesare următoarele:

- Descarcărea și instalarea IDE-ului Android Studio
- Importarea aplicației mobile in acesta
- Rularea acesteia fie pe dispozitivul mobil propriu fizic al utilizatorului, fie pe emulatorul virtual pus la dispoziție de Andriud Studio. Acesta din urmă are unele limitări cum ar fi indisponibilitatea folosirii senzorilor fizici din cadrul telefonului.

7.2.2. Instalarea aplicaţiei web

Pentru instalarea și utilizarea aplicației web este necesar să deținem o platforma pe care rulează Node.js și Angular. Se deschide un terminal cu root-ul în folderul de baza al aplicației și se rulează comandă „npm start”. După executarea acestora, utilizatorul poate să acceseze aplicația din interiorul browser-ului, introducând „localhost:4200” bară de căutare a acestuia.

7.3. Utilizare

În această secțiune va fi prezentată interacțiunea utilizatorului cu aplicația prin descrierea textuală însoțită de imagini a unor cazuri de utilizare.

7.3.1. Inregistrare utilizator

După ce utilizatorul a instalat și a pornit aplicația, în prim plan se va regăsi activitatea de login. Bineînțeles, la prima utilizare a aplicației utilizatorul nu are un cont

creat, dar are posibilitatea de a-și crea unul. În partea de jos a activității este prezent un mic dialog care îndrumă utilizatorul spre crearea unui cont nou. Prin atingerea zonei “Create one” utilizatorul va fi redirecționat către activitatea de înregistrare. În această activitate utilizatorul poate să își creeze un cont nou prin introducerea detaliilor necesare si atingerea butonului de „Sign Up”.

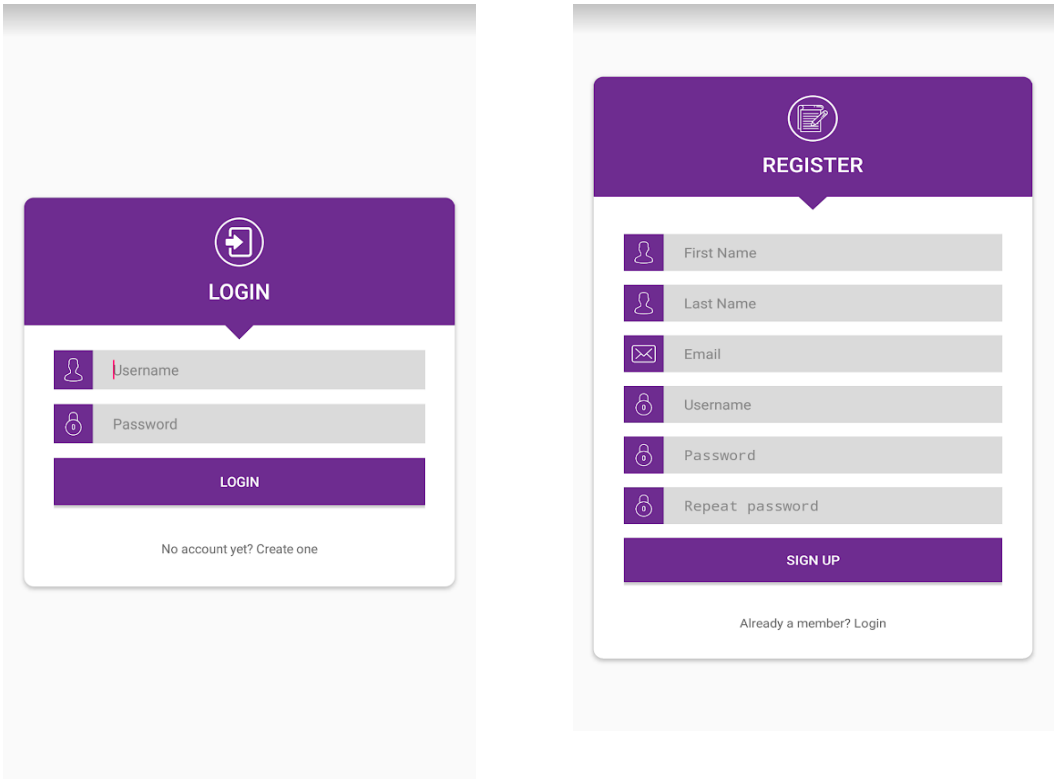


Figura 7.1 Activitatile de login și register

Dacă datele introduse sunt invalide sau dacă nu poate fi realizată conexiunea cu serverul vor fi afișate mesaje de eroare sugestive. În caz contrar, utilizatorul va fi redirecționat către activiatea de „Dashboard” (figura 7.3).

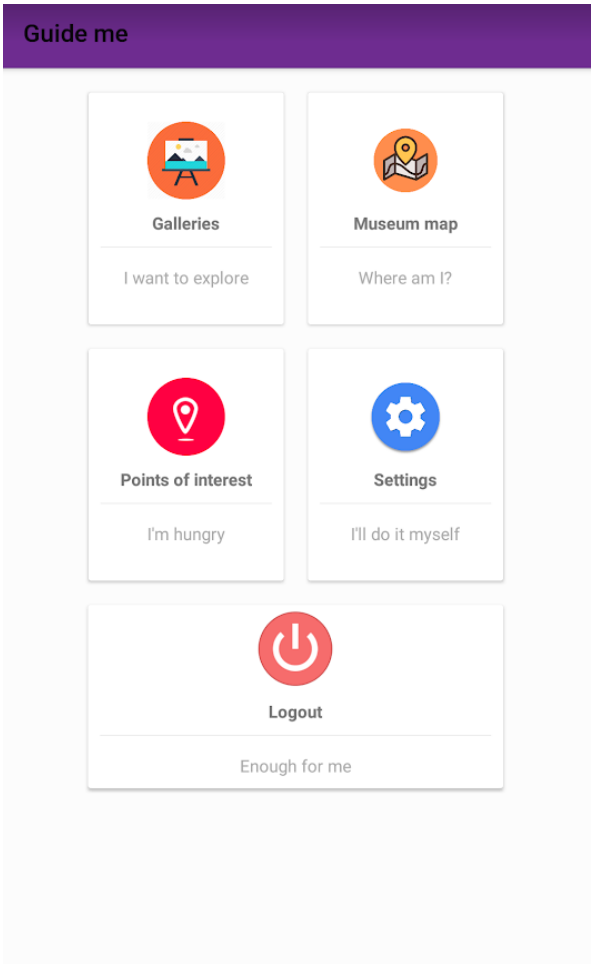


Figura 7.2 Activitatea de dashboard

Secțiunea de Galleries va redirecționa utilizatorul către o lista cu galeriile disponibile în cadrul muzeului. Museum map reprezintă o activitate ce afișează poziția curentă a utilizatorului în cadrul muzeului. Points of Interest va redirecționa utilizatorul către o activitate în care poate să caute puncte de interes din apropiere. Seciunea de Settings permite utilizatorului să schimbe limba de afișare a aplicației. Secțiunea de Logout va tirmite utilizatorul către activitatea de Login. Dacă acesta închide aplicația fără să se delogheze, la următoarea lansare a aplicației acesta va fi trimis direct pe activitatea de dashboard.

7.3.2. Scanare tag NFC

După logare, utilizatorul este capabil să scaneze tag-urile puse la dispoziție pentru fiecare exponat. În urmă scanării, utilizatorul va fi redirecționat către o activitate cu lista artefactelor descoperite de utilizator în cadrul muzeului, în funcție de galeria din care face parte exponatul respectiv. În figura 7.4 sunt prezentate două imgini. Prima este o notificare a faptului că utilizatorul a scanată un tag NFC, iar cea de-a două reprezintă activitatea în care sunt prezente exponatele descoperite de către utilizator.

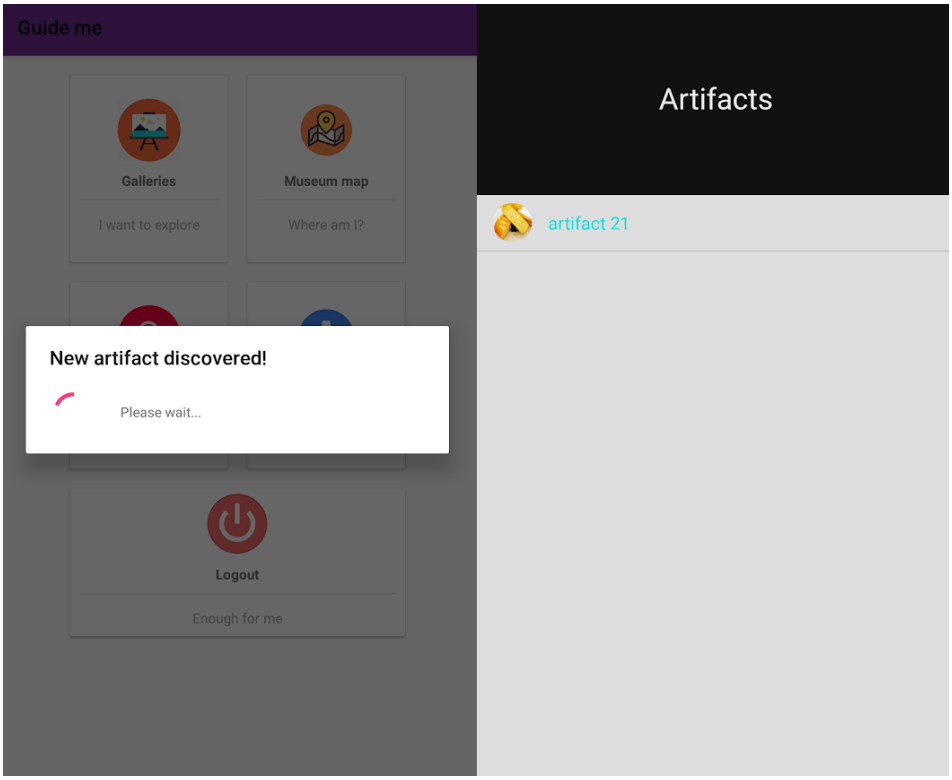


Figura 7.3 Scanare tag NFC

7.3.3. Schimbarea limbii de afișare a aplicației

După ce utilizatorul se înregistrează cu succes în aplicație, prin accesarea meniului „Settings” acesta va avea posibilitatea să schimbe limba aplicației. În cadrul activității vor fi prezente două butoane de tip radio pentru schimbarea limbii de afișare. După apăsarea unuia dintre aceste butoane utilizatorul va fi redirecționat către activitatea de Dashboard unde va observă că limba aplicației a fost schimbată. În momentul de față aplicația suportă limbile română și engleză. În figura 7.5 este prezentată activitatea de Settings și rezultatul schimbării limbii de afișare în română.

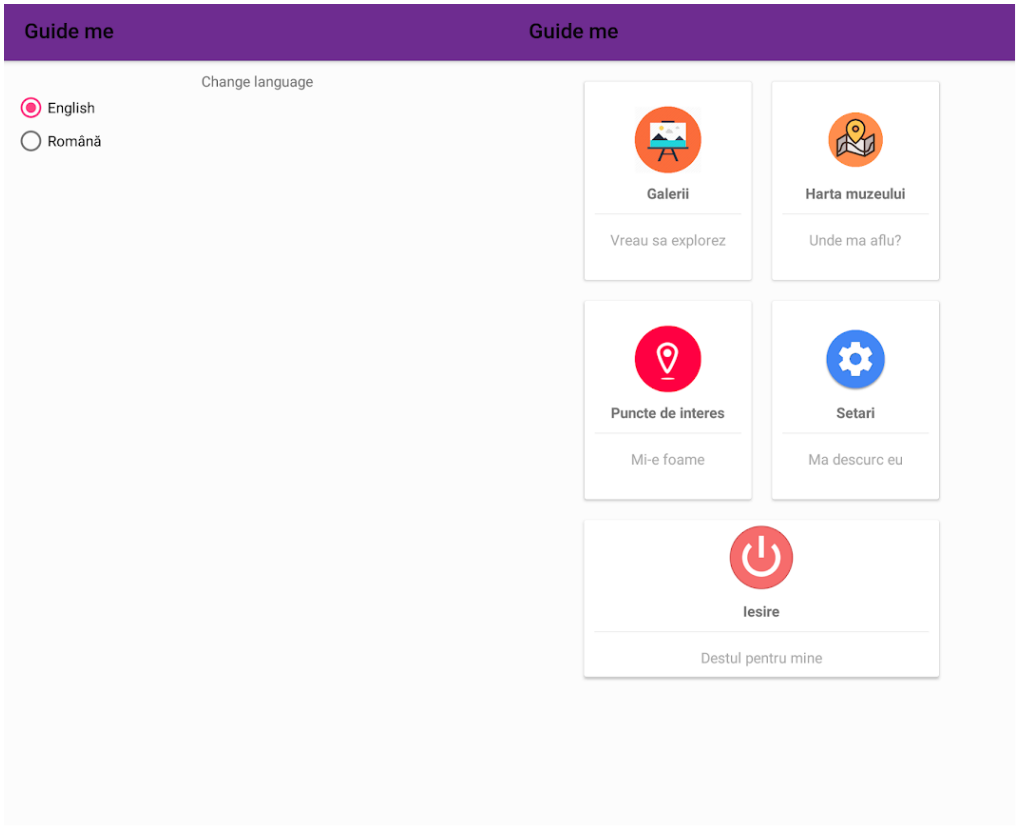


Figura 7.4 Activitatea de Settings și rezultatul schimbării limbii în română

7.3.4. Puncte de interes

În cadrul activității de puncte de interes utilizatorul poate să caute diferite din jurul sau. Utilizatorul are două posibilități:

- Poate să opteze pentru comandă vocală. Acesta apasă butonul înregistrare sunet, iar după sunetul ce indică începerea înregistrării acesta poate să înceapă să vorbească. După ce utilizatorul a încheiat fraza, textul extras de către API-ul de speech recognition va fi folosit că și text de căutare. Se va porni automat o aplicație Google Maps (dacă este disponibilă), iar textul extras va fi trimis că și search query pentru aplicația Google Maps. Această funcționalitate este folositoare în momentul în care utilizatorul dorește să facă o căutare mai ambiguă (spre exemplu: „find me something to eat”).
- Cea de-a două varianta este să folosească meniul de dropdown care conține o lista cu punctele de interes cele mai des căutate cum ar fi restaurante, hoteluri, baruri, spitale, cluburi etc. Totodată, prin utilizarea slider-ului din aplicație, utilizatorul poate să schimbe rază de căutare. Intervalul de căutare disponibil este cuprins între 0 și 50km. La fiecare modificare, fie ea schimbarea selecției din dropdown sau schimbarea razei de căutare, se va execută automat o căutare nouă.

În imaginea 7.6 sunt prezentate cele două moduri de căutare a punctelor de interes (stânga: căutare cu ajutorul Google Maps; dreapta: căutare bazată pe puncte de interes specifice și rază de căutare.).

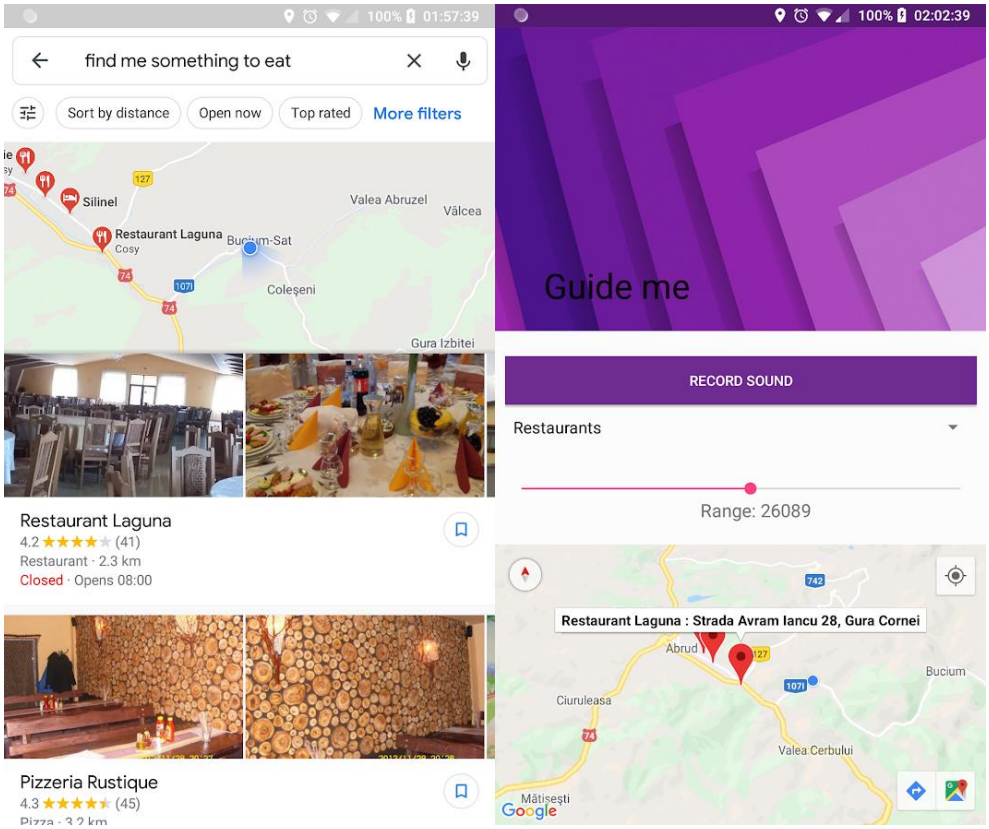


Figura 7.5 Rezultatele celor două metode de căutare ale punctelor de interes

7.3.5. Harta muzeului

Harta muzeului ajută utilizatorul să se orienteze în cadrul acestuia, oferindu-i informații cu privire la numărul de galerii, intrări/ieșiri ale acestora, numele și poziționarea artefactelor din acestea și nu în ultimul rând, poziția să în muzeu. Poziția utilizatorului va fi updatată atât în momentul în care activitatea de harta nu se află în foreground, cât și cât timp această este vizibilă. Poziția curentă a utilizatorului se bazează pe două aspecte:

- Poziția ultimului artefact scanat de către utilizator. În momentul în care utilizatorul scanează un artefact, automat cursorul ce indică poziția să va fi mutat la poziția artefactului pe care l-a scanat.
- Mișcarea utilizatorului detectată cu ajutorul senzorilor telefonului (Accelerometru și senzorul de tip GAME_ROTATION VECTOR).

Capitolul 8. Concluzii

În această secțiune vor fi prezentate concluziile asupra variantei finale a sistemului. Vor fi trecute în revista obiectivele care au fost atinse în proiect. Vor fi prezentate contribuțiile personale în cadrul aplicației, iar mai apoi o serie de posibilități de dezvoltare ulterioară a sistemului, ceea ce ar duce la diponibilitatea unui număr mai mare de servicii în cadrul aplicației, cu scopul de a oferi utilizatorilor o experiență mult mai bogată.

8.1. Contribuții personale

În cadrul aplicației s-a realizat implementarea robustă a unui sistem capabil să se comporte că un ghid inteligent pentru un muzeu, pe baza unor input-uri și a unor acțiuni executate de către utilizator.

Funcționalitatea principala (afișarea de informații în urmă scanării unui tag NFC) a fost gândită astfel încât fiecare artefact să aibă o valoare ce identifica unic atât artefactul cât și tag-ul asociat.

Funcționalitățile secundare cum ar fi căutarea punctelor de interes și poziționarea utilizatorului în cadrul muzeului ajută la mărirea volumului de servicii disponibile către utilizator, fără a plasa însă produsul într-un alt domeniu decât cel specificat, aceste servicii auxiliare fiind înrudite între ele.

Părțile componente ale sistemului au fost proiectate astfel încât să ofere posibilitatea de utilizare a acestuia într-o manieră mobile, cu condiția că utilizatorul să aibă conexiune la internet.

8.2. Dezvoltări ulterioare

Dezvoltările ulterioare ale sistemului contribuie atât la îmbunătățirea experienței utilizatorului, cât și la ușurarea procesului de mentenanță a sistemului.

Mai jos sunt prezentate o serie de dezvoltări ce ar putea contribui la realizarea celor precizate mai sus:

- Una dintre ideile de dezvoltare ulterioară ar putea veni pe partea de puncte de interes. Acest lucru implică implementarea unui sistem automat de recomandări, bazat pe o serie de preferințe pe care utilizatorul poate să aleagă să le selecteze în momentul creării unui cont sau în momentul în care dorește să acceseze funcționalitatea de puncte de interes.
- O altă funcționalitate constă în imeplementarea aceleiași aplicații pe alte sisteme de operare mobile, cum ar fi iOS, cee ace ar duce la o crește substanțial a numărului de utilizatori.
- O idee pentru promovarea muzeului respectiv ar fi amplasarea de tag-uri NFC, care în urma scanării să direcționeze utilizatorul automat către diferite site-uri de socializare, unde aceștia pot să execute acțiuni specific cum ar fi “like”, “follow” etc.
- Amplasarea unor astfel de tag-uri NFC care să le ofere utilizatorilor diferite beneficii (nu neapărat financiare) ar duce din nou la o creștere a numărului de utilizatori.

- Utilizarea de dispozitive beacon în cadrul muzeului pentru a îmbunătăți funcționalitatea hărții virtuale a muzeului
- Implementarea unor funcționalități ce să permită utilizatorilor să vadă programul de funcționare a muzeului și posibilitatea că aceștia să își poată achiziționa bilete direct din cadul aplicației.
- Adăugarea aplicației într-un magazin de aplicații Android (ex Play Store) pentru a elimina incovenienția faptului că instalarea aplicației necesită pași intermediari în plus.
- Deploymentul părții de aplicație web pe un serviciu cloud pentru a elimina necesitatea resurselor software menționate pe fiecare dispozitiv pe care această este folosită.
- Adăugarea descrierii audio și în limba română în momentul în care aceasta va fi disponinbilă.

Bibliografie

[1] Özgen Akman, „Near Field Communication Applications”, Aalto University, 2015, Disponibil la: https://aaltodoc.aalto.fi/bitstream/handle/123456789/18160/master_Akman_%C3%96zgen_2015.pdf?sequence=1&isAllowed=y

[2] Pedro Diogo Candeias Santiago Oliveira, „MyOcean, Improving an exhibition visit with wearable technology”, Tecnico Lisboa, 2016, Disponobil la: https://fenix.tecnico.ulisboa.pt/downloadFile/281870113703138/dissertacao.pdf

[3] Orkun Gençoğlu, “Shopping Assistant: Product Discovery for Senior Citizens within Local Groceries”, Aalborg University Copenhagen, 2013, Disponibil la: https://projekter.aau.dk/projekter/files/77307375/Shopping_Assitant_ICTE.pdf

[4] Timothy Boyle, NFC for Museums Case Study, 2013, Disponibil la: https://qfuse.com/blog/nfc-for-museums-case-study-museum-of-london/

[5] Hui-Huang Hsu, Wei-Jan Peng, Timothy K. Shih, „Smartphone Indoor Localization with Accelerometer and Gyroscope”, 17th International Conference on Netwok-Based Information Systems, Salerno, Italia, 2014

[6] SensorEvents, https://developer.Android.com/reference/Android/hardware/SensorEvent

[7] Mi Hu, Yu Weng, „Development of mobile travel guide application for museums”, Lapland University of Applied Sciences, 2016, teza de master, Disponibil la: https://www.theseus.fi/bitstream/handle/10024/107633/Hu%20Mi_Thesis.pdf?sequence=1

[8] Dirjan Georgiana Serena, „Utilizarea tehnologiei NFC si Beacon in implementarea unei aplicatii mobile destinata muzeelor”, lucrare de licenta, Universitatea Tehnica din Cluj-Napoca, 2017