Universiteti i Tiranës
Fakulteti i Ekonomisë
Dega Informatikë Ekonomike
Departamenti i Statistikës dhe Informatikës së Zbatuar

## TEMË DIPLOME

# Menaxhimi i rezervimit të vizitave mjekësore për pajisjet Android

Studenti:
Xhulio Doda

Udhëheqësi shkencor:
Prof.Dr. Areti Stringa

Tiranë 2018

## Abstrakt

Ky punim provon t'i japë zgjidhje problematikave administrative të klinikave apo spitaleve përsa i përket rezervimit të vizitave mjekësore. Eksperienca në klinikat apo spitalet nuk është aspak e kënaqshme. Pritja në rradhë të gjata dhe marrja e informacioneve të gabuara rreth orareve të doktorëve janë kthyer në një pritshmëri për pacientët.

Në seksionin e parë autori eksploron sistemin operativ Android duke u futur në detaje përsa i përket pjesëve integrale në cdo aplikacion. Më tej nga teoria u kalua në praktikë duke krijuar paraqitjen e jashtme në formatin përkatës XML (EXtended Markup Language) si dhe dizenjimi i kodit përgjegjës për funksionim e aplikacionit që nga rregjistrimi dhe log-imi dhe deri tek rezervimi dhe anullimi i një vizite. Në fund u krijua edhe aplikacioni i administratorit me funksionin e menaxhimit të shërbimeve qoftë ky shtimi apo fshirja e tij dhe marrja e informacionit rreth klientëve apo pacientëve.

Qëllimi i punimit është ti japë zgjidhje teknike problematikave me të cilat përballen si administratorët ashtu edhe pacienët. Me anë të zgjidhjeve teknologjike është bërë perpjekja për të zbutur barrierën në kohë të proceseve administrative dhe për të përmirësuar shërbimin në përgjithësi.

**Permbajtja e tabelës**

# 1. Hyrje

Gjatë dekadës së fundit, me ardhjen e smartphonave, jeta e njerëzve ka pësuar një metamorfozë totale. Disa vite më përparë njerëzit i përdornin pajisjet mobile, pra celularët, vetëm si një mjet apo kanal komunikimi. Ndërsa tani jemi të gjithë dëshmitar të këtij ndryshimi. Të gjithë e mbajnë me vete celularin kudo që shkojnë dhe qëndrojnë vazhdimisht të lidhur me botën virtuale.

Nuk ka qenë vetëm celulari ai që ka ndryshuar jetën e njerëzve, por aplikacionet e krijuar për smartphone-t dhe pajisjet e tjera mobile sic janë tabletët apo edhe orat inteligjente të dala në treg vitet e fundit. Këto aplikacione kanë bërë të mundur një komunikim më të thjeshtë dhe të shpejtë ndërmjet njerëzve. Mund të regjistrohesh në një rrjet social dhe nga pajisja jote mobile të shohësh postimet e miqve. Pra, një anë pozitive e aplikacioneve është thjeshtësia në gjetjen dhe aksesimin e informacionit.

Pra, një anë pozitive e aplikacioneve është thjeshtësia në gjetjen dhe aksesimin e informacionit. Duke qëndruar në shtëpi mund të porosisësh gatime nga restorante të ndryshme, të gjesh receta gatimi apo të reja nga bota politike, shoëbiz etj.

Një prej sistemeve operative që përdoret prej 2.53 miliard pajisjesh mobile është sistemi operativ Android [1]. Ky sistem operativ është open-source dhe linux-based. Shumë zhvillues krijojnë aplikacione të cilat do të ekzekutohen mbi këtë sistem kaq të shpërhapur dhe të suksesshëm në ditët e sotme. Dyqani ku mund t'i gjesh këto aplikacione dhe prej të cilit mund t'i shkarkosh për t'i përdorur më pas quhet PlayStore. PlayStore është dyqani zyrtar i Android, ku gjendet të gjitha aplikacionet, prej rrjeteve sociale, tek lojrat, programet për dërgimin e mesazheve dhe shumë të tjera.

Dhe vetë procesi për ta hedhur aplikacionin ne PlayStore eshte mëse i thjeshtë. Fillimisht duhet të përditësosh llogarinë tënde Gmail në një llogari Google Play Developer duke kryer nje pagesë të njëhershme prej 20 dollarësh. Me pas duke ndjekur hapat nga faqja zyrtare e vetë Google dhe aplikacioni juaj do jetë gati për tu shkarkuar brenda 2-3 orëve [2].

# 2. Hyrje në Android

## 2.1 Çfarë është Android?

Android është një system operativ open-source i bazuar në Linux, I cili përdoret për pajisjet mobile si smartphone dhe tablet. Android u zhvillua nga Open Handset Alliance, e drejtuar nga Google, ku merrnin pjesë dhe shumë kompani të tjera [3]. Ky sistem operativ për pajisjet mobile ofron një qasje të unifikuar për zhvillimin e aplikacioneve, kjo do të thotë se një

programues mund të programojë vetëm në Android dhe aplikacioni i tij do të ekzekutohet në shumë pajisje të tjera të bazuara në platformën Android.

Figura 2.1 Stema e Android

Ky sistem operativ ofron një framework të pasur i cili mundëson krijimin e aplikacioneve inovative dhe lojrave për pajisje mobile në një ambjent programimi në gjuhën Java. Aplikacionet në Android krijohen si një kombinim i disa pjesëve përbërëse të ndryshme dhe të vecanta të cilat mund të thërriten në mënyrë individuale. Për shembull një Activity ofron një ndërfaqe grafike për përdoruesin dhe një Service në mënyrë të pavarur zhvillon informacionet që i shfaqen përdoruesit.

Nga një pjesë përbërëse mund të inicializohet një pjesë tjetër përbërëse duke përdorur një Intent. Mund edhe të inicializohet një pjesë përbërëse tjetër në një aplikacion tjetër. Për shembull, nga What's App mund të kalojmë në Gmail duke shtypur një adresë e-mail ose mund të kalojmë në Gallery për të ngarkuar një foto.

Android ofron një framework përshtatës i cili lejon krijuesin të përdorë burime (resources) unike për konfigurime të ndryshme të pajisjeve. Mund të krijoni layout-e të ndryshme në XML për madhësi të ndryshme të ekranit dhe më pas sistemi vendos se cilin layout të aplikojë bazuar në madhësinë aktuale të ekranit.

Mund të deklaroni cilësi specifike të cilat i kërkon aplikacioni kështu që dyqanet si Google Play Store të mos e lejojë instalimin në pajisje të cilat nuk e kanë këtë specifikë.

Android është një sistem operativ open-source, i cili ka pushtuar një tregun e sistemeve operative mobile. Pra në ditët e sotme shumica e pajisje përdorin këtë platformë [4]. Ky fakt ka tërhequr një komunitet të madh zhvilluesish dhe programuesish dhe së bashku me to, një mori me aplikacione të cilësisë së lartë. Domethënë në gjendjen e sotme të Google Play Store vlerësohet më shumë një ide e mirë sesa një zhvillim i përsosur për një aplikacion. Kjo ka shtyrë kompanine Google të krijoje kurse ku mund të aftësoshesh në krijimin e një aplkacioni brenda

disa javëve duke supozuar pak njohuri paraprake ne informatike [5]. Duke sjellë zvogëlimin e kostove të prodhimit të një aplikacioni në këtë platformë.

## 2.2 Bazat e Aplikacioneve

Aplikacionet Android shkruhen në gjuhën e programimit Java. Më pas Android SDK (Software Development Kit) do të kompilojë kodin së bashku me çdo të dhënë dhe burimet në një APK, një paketë Android, e cila ruhet si një file me prapashtesën .apk . Një file APK përmban të gjitha pjesët përbërëse të një aplikacioni Android dhe është file të cilin përdorin pajisjet Android për të instaluar aplikacionin. Këto janë disa nga vecoritë e aplikacioneve në Android:

- Sistemi operativ Android është një sistem Linux me shumë përdorues ku çdo aplikacion është një përdorues i vecantë.
- Sistemi i cakton çdo aplikacioni një ID përdoruesi unike. Sistemi vendos të drejta aksesimi për çdo file në mënyrë që vetëm ai përdorues i caktuar të mund t'i aksesojë ato file.
- Çdo proces ka makinën e vet virtuale, pra kodi mund të ekzekutohet në mënyrë të pavarur nga proceset e tjera.
- Çdo aplikacion ka procesin e vet. Android e nis një process kur një prej komponentevë ose pjeseve përbërëse do të ekzekutohet, dhe të mbyllet kur nuk do të nevojitet më apo kur sistemi të ketë nevojë për memorje për aplikacionet e tjera.

Në këtë mënyrë sistemi Android i lejon aplikacionit të ketë akses vetëm në komponentët që i nevojiten dhe jo më tepër. Kjo krijon një ambjent të sigurt në të cilin aplikacionet nuk mund të aksesojnë pjesë të tjera të sistemit pervecëse atyre për të cilat iu është dhënë leje.

Megjithatë ka mënyra që një aplikacion të ndajë të dhëna me aplikacionet e tjera. Është e mundur që dy aplikacioneve t'iu caktohet I njëtji ID përdoruesi, në këtë mënyrë ata do të jenë në gjendje të aksesojnë filet e njëri-tjetrës. Për të kursyer burimet e sistemit, aplikacionet me të njejtin ID përdoruesi mund të ekzekutohen në të njëjtin process dhe të ndajnë të njëjtën makinë virtuale. Një aplikacion mund të kërkojë të drejta për të aksesuar të dhëna sic mund të jene SMS të përdoruesit, kamerën, kartën SD, Bluetooth dhe shumë të tjera. Në këto raste përdoruesi duhet të garantojë vetë këto të drejta.

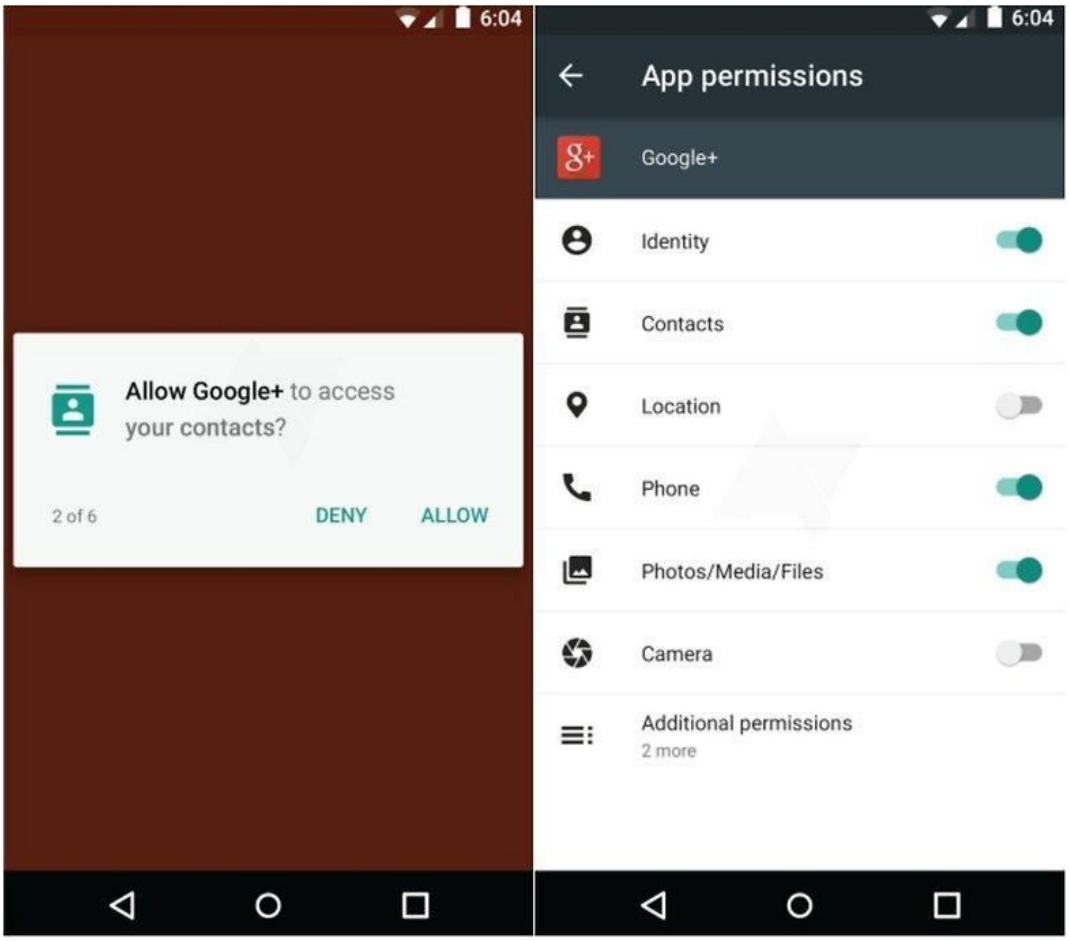


Figura 2.2 Kërkimi I të drejtës për aksesimin e të dhënave

## 2.3 Pjesët përbërëse të një Aplikacioni

Pjësët përbërëse të një aplikacioni apo ndryshe komponetët janë blloqe të domosdoshme për të ndërtuar një aplikacion Android. Shpesh këta komponentë bazohen në njëri-tjetrin, por secili ekziston si një entitet I pavarur dhe luajnë një rol specific në ndërtimin e aplikacionit dhe ndikojnë në sjelljen e përgjithshme të aplikacionit.

Ka katër lloje të ndryshme të komponentëve të aplikacionit. Secili shërben për një qëllim të caktuar dhe ka një cikël jete të ndryshme që përcakton sesi komponenti krijohet dhe shkatërrohet.

Këto janë katër llojet e komponentëve:

- Activities (aktivitetet)
- Services (shërbimet)
- Content providers
- Broadcast receivers

### 2.3.1 Activities

Një aktivitet përfaqëson një ekran të vetëm me një ndërfaqe të përdoruesit. Për shembull, një aplikacion email-i mund të ketë një aktivitet që shfaq listën e email-eve të reja, një aktivitet tjetër në të cilin mund të shkruani një mail të ri dhe një aktivitet tjetër për të lexuar mail-et. Edhe pse aktivitetet punojnë së bashku për të krijuar një tërësi të bashkuar, secili prej tyre është i pavarur nga të tjerët. Si i tillë, çdo aplikacion mund të fillojë një nga këto aktivitete, sic ndodh kur nga What's App hapim aplikacionin e kamerës për të dërguar një foto.

Pra, një aplikacion zakonisht konsiston në shumë aktivitete të lidhura ngushtë ndërmjet tyre. Shpesh një aktivitet i aplikacionit përcaktohet si "kryesori" (main activity), i cili i shfaqet përdoruesit kur hap aplikacionin për herë të parë. Më pas çdo aktivitet mund të nis një aktivitet tjetër në mënyrë që të kryejnë punë të ndryshme. Çdo herë që nis një aktivitet i ri, aktiviteti i mëparshëm ndalohet, por sistemi e ruan atë në stack. Kur nis një aktivitet i ri, ai vendoset (push) në majë të stack dhe merr vëmendjen e përdoruesit. Ky stack implementon algoritmin e tipit "last in, first out", kështu që, kur përdoruesi mbaron punë me aktivitetin dhe shtyp butonin Back, ky aktivitet do të dal (pop) nga stack-u dhe do të vazhdoj aktiviteti i mëparshëm.

Kur një aktivitet ndalohet sepse do të nis një i ri, do të njoftohet një ndryshim në gjendje nëpërmjet metodave "callback" të ciklit të jetës së aktivitetit. Ka disa metoda "callback" që një aktivitet mund të marri, si pasojë e ndryshimit së gjendjes së tij, nëse sistemi po e krijon, ndalon, rimarrë apo shkatërron atë. Secili callback ofron mundësinë për të performuar detyra specifike të cilat janë të përshtatëshme me ndryshimin e gjendjes. Nesë do të kemi ndalimin e aktivitetit, ai duhet të lëshojë çdo objekt të madh, si network apo lidhja me databazën. Kur aktiviteti rinis, mund të rimarri burimet e nevojshme dhe të rifillojë veprimet të cilat ishin ndërprerë. Këto tranzicione të gjendjes janë të gjitha pjesë e ciklit të jetës së një aktiviteti.

### 2.3.2 Services

Një shërbim është një pjesë përbërëse apo komponent i cili ekzekutohet në background për të kryer detyra të gjata. Një shërbim nuk ofron një ndërfaqe grafike për përdoruesin. Një shërbim mund të luajë muzikë në background ndërkohë që përdoruesi ndodhet në një aplikacion tjetër, apo të marrë të dhëna nga rrjeti pa bllokuar ndërveprimin e përdoruesit me një aktivitet.
Një component i një aplikacioni tjetër mund të nis një shërbim dhe ai do të vazhdoj të ekzekutohet në background edhe nëse përdoruesi do të hap një aplikacion tjetër. Për më tepër një component mund të lidhet me një shërbim dhe të ndërveproj me të dhe të performoj komunikim ndermjet proceseve.

Një shërbim ka dy forma:

- I nisur (started)

  Një sherbim quhet "i nisur" kur një component e nis atë duke thirrur metoden startService(). Pasi nis, shërbimi mund të punoj në background për një kohë të pacaktuar, edhe nëse komponenti që e nisi mund të shkatërrohet. Shpesh një shërbim kryen një detyrë të vetme dhe nuk kthen një rezultat tek thirrësi. Ai mund të shkarkojë apo ngarkojë një file në rrjet. Kur mbaron së kryeri detyrën shërbimi ndalon.

- I lidhur (bound)

  Një shërbim quhet "i lidhur" kur një component i një aplikacioni lidhet me të duke thirrur metoden bindService(). Një shërbim i lidhur ofron ndërfaqe klient-server që lejon komponentët të ndërveprojnë me shërbimin, të dërgojnë kërkesa, të marrin përgjigje etj. Një shërbim i lidhur punon aq sa kohë një component i aplikacionit është i lidhur me të. Me një shërbim mund të lidhen më shumë sesa një component njëherësh, pork ur të gjithë të prishin lidhjen shërbimi do të shkatërrohet.

Një shërbim ekzekutohet në thread-in kryesor të procesit host, shërbimi nuk krijon thread-in e vet dhe nuk punon në një proces të vecantë, vecse në rastet kur specifikohet e kundërta. Kjo nënkupton që, nëse shërbimi do të kryej punë intensive në CPU, duhet krijuar një thread i ri Brenda të cilit shërbimi të kryej punen e vet. Duke përdorur një thread të vecantë reduktohet risku i gabimeve ANR (Application Not Responding) dhe thread-i kryesor i aplikacionit të qëndrojë i dedikuar ndaj ndërveprimit me përdoruesin.

### 2.3.3 Content Providers

Një "content provider" apo një ofrues i përmbajtjes menaxhon një tërësi të dhënash të përbashkëta. Të dhënat mund të ruhen në një file sistemi, një databazë SQLite, në web, ose çdo memorie, magazinë, të qëndrueshme të cilën aplikacioni mund ta aksesojë. Nëpërmjet content provider aplikacionet mund të marrin apo edhe të modifikojnë të dhënat, nëse content provider-i e lejon një gjë të tillë. Për shëmbull, sistemi Android ofron një content provider i cili menaxhon informacionet rreth kontakteve të përdoruesit. Si i tillë, çdo aplikacion me të drejtat e nevojshme mund të marri një pjesë të këtyre të dhënave për të lexuar apo shkruar informacion rreth një personi në vecanti. Content providers janë gjithashtu të dobishme për të lexuar dhe shkruar të dhëna të cilat janë private për aplikacionin dhe jo të përbashkëta.

Pra, ofruesit e përmbajtjes ose ndryshe content providers menaxhojnë aksesin në një set të strukturuar të dhënash. Ata enkapsulojnë të dhënat, dhe mundësojnë mekanizma për përcaktimin e sigurisë të të dhënave. Content providers janë ndërfaqja standarte e cila bën lidhjen e të dhënave në një proces me kodin i cili ekzekutohet në një proces tjetër.

Kur nevojitet marrja e të dhënave në një content provider, përdoret objekti i tipit ContentResolver nga konteksti (Context) i aplikacionit për të komunikuar me ofruesin (provider) si një klient. Objekti ContentResolver komunikon me objektin ofrues. Objekti ofrues merr kërkesat për të dhëna nga klientët, kryen detyrën e kërkuar dhe më pas kthen rezultatet.

### 2.3.4 Broadcast Receiver

Një marrës broadcast-esh apo një broadcast receiver është një komponent i cili u përgjigjet njoftimeve broadcast. Shumë broadcast krijohen nga sistemi, për shembull, një njoftim broadcast që lajmëron se ekrani është fikur, bateria po mbaron, apo u bë një screenshot. Aplikacionet mund të lëshojnë njoftime broadcast gjithashtu, për shembull, të bëj të ditur tek aplikacionet e tjera se disa të dhëna u shkarkuan në pajisje dhe janë të disponueshme edhe për to. Broadcast receivers nuk shfaqin ndërfaqe për përdoruesin, por megjithatë ato mund të shfaqin lajmërime në status bar, për të lajmëruar përdoruesin kur një ngjarje broadcast është duke ndodhur.

Sic e përmendem dhe mësipër, një vecanti e sistemit Android është se çdo aplikacion mund të nis një komponent të një aplikacioni tjetër. Për shembull, nëse një përdorues do të bëj një fotografi me fotokamerën e pajisjes, ka mundësi që të ekzistojë një aplikacion tjetër që e bën këtë dhe që aplikacioni jonë mund ta përdorë. Pra në vend që programuesi të zhvilloj një aktivitet për të bërë fotografi, ai përdor aplikacionin Camera të Android. Programuesi nuk ka nevojë ta inkorporojë apo të bëj lidhjen me aplikacionin e kamerës. Ai thjesht nis aktivitetin e aplikacionit të kamerës që shërben për të fotografuar. Pasi mbarohet, fotografia i kthehet aplikacionit të mëparshëm. Ndërkohë përdoruesit do t'i duket se kamera është pjese e aplikacionit të mëparshëm.

Kur sistemi nis një komponent, ai nis nje proces për atë aplikacion (nëse nuk është ende në ekzekutim) dhe inicializon të gjitha klasat e nevojshmë për atë komponent. Ky komponent do të punojë në procesin e aplikacionit të vetë jo në procesin e aplikacionit që e nisi. Ndryshe nga aplikacionet në sistemet e tjera, aplikacionet Android nuk kanë një pikë të vetme hyrje (single entry point), nuk ka funksion main().

Çdo aplikacion ekzekutohet në procese të ndara me autorizime për file që nuk mund të aksesohen nga aplikacione të tjera. Për këtë arsye nuk mund të aktivizojë në mënyrë të drejtpërdrejtë një komponent të një aplikacioni tjetër. Megjithatë, këtë mund ta bëj sistemi operativ Android. Pra, për të aktivizuar një komponent të një aplikacioni tjetër, aplikacioni i mëparshëm duhet t'i nisi sistemit një mesazh i cili specifikon intent-in (synimin) që të nis një komponent të caktuar. Më pas sistemi akzivizon komponentin.

### 2.3.5 Aktivizimi i Komponentëve

Tre nga katër llojet e komponentëve activities, services, broadcast receivers aktivizohen nëpërmjet një mesazhi asinkron i quajtur intent. Intent-et lidhin komponentë individual me njëri-tjetrin, qoftë nëse komponenti i përket atij aplikacioni apo një tjetër. Një intent krijohet me anë të një objekti të tipit Intent, i cili specifikon një mesazh për të aktivizuar një komponent.

Për aktivitetet dhe shërbimet, një intent specifikon veprimin për të kryer, për shembull të shoh apo të dërgojë diçka. Në disa raste, mund të aktivizohet një aktivitet për të marrë rezultate prej tij. Në këtë rast aktiviteti duhet të kthej rezultatet në një Intent.

Për broadcast receivers, intenti specifikon vetëm lajmërimin i cili do të bëhet broadcast, për shembull, një broadcast i cili lajmëron se bateria po mbaron përfshin vetëm një string që tregon "bateria po mbaron".

Komponenti tjetër, content provider, nuk aktivizohet nga një intent. Ai aktivizohet vetëm kur është objektiv i një kërkese nga ContentResolver.

- Për të nis një activity:

Një aktivitet mund të niset si një rast i ri i një Activity duke kaluar një Intent në metodën startActivity(). Intent-i përshkruan aktiviteti i cili do të niset dhe mban të dhënat e nevojshme. Nëse dëshirohet të merren rezultate nga aktiviteti kur të mbarojë, thërritet metoda startAcivityForResult(). Aktiviteti fillestar e merr rezultatin si një objekt Intent të vecantë në metodën onActivityResult().

- Për të nis një service

Një service apo shërbim është një komponent që kryen detyra në background pa patur një ndërfaqe grafike. Një shërbim mund të niset për të kryer një detyre të vetme (one-time operation ) duke kaluar një Intent në metodën startService(). Intent-i përshkruan shërbimin i cili do të aktivizohet dhe mban të dhënat e nevojshme. Nëse shërbimi është i përcaktuar si një ndërfaqe klient-server, mund të kryhet lidhja me shërbimin nga një komponent tjetëe duke kaluar një intent në metodën bindService()

- Për të dërguar një broadcast

Një broadcast është një mesazh të cilin çdo aplikaxion mund ta marri. Sistemi shpërndan broadcast-e të ndryshme. Një broadcast mund të dërgohet tek aplikacionet e tjera duke kaluar një intent në metoden sendBroadcast().

## 2.4 Android Studio

Android Studio është një IDE (Integrated Development Environment) zyrtare për zhvillimin e aplikacioneve në platformën Android. Android Studio u publikua në 16 Maj të 2013 në konferencën I/O të Google. Bazuar në software-in e JetBrains IntelliJ IDEA, Android Studio është krijuar në mënyrë specifike për zhvillimin në platformën Android. Mbi editorin e fuqishëm

të IntelliJ, Android Studio ofron dhe më shumë vecori dhe cilësi të cilat shtyjnë produktivitetin kur ndërton një aplikacion Android si:

- Një sistem fleksibël i bazuar në Gradle
- Një emulator i pasur dhe i shpejtë
- Një ambjent i unifikuar ku mund të zhvillohen aplikacione për çdo pajisje Android
- Instant Run për të bërë ndryshime gjatë punës së aplikacionit pa ndërtuar një APK të re

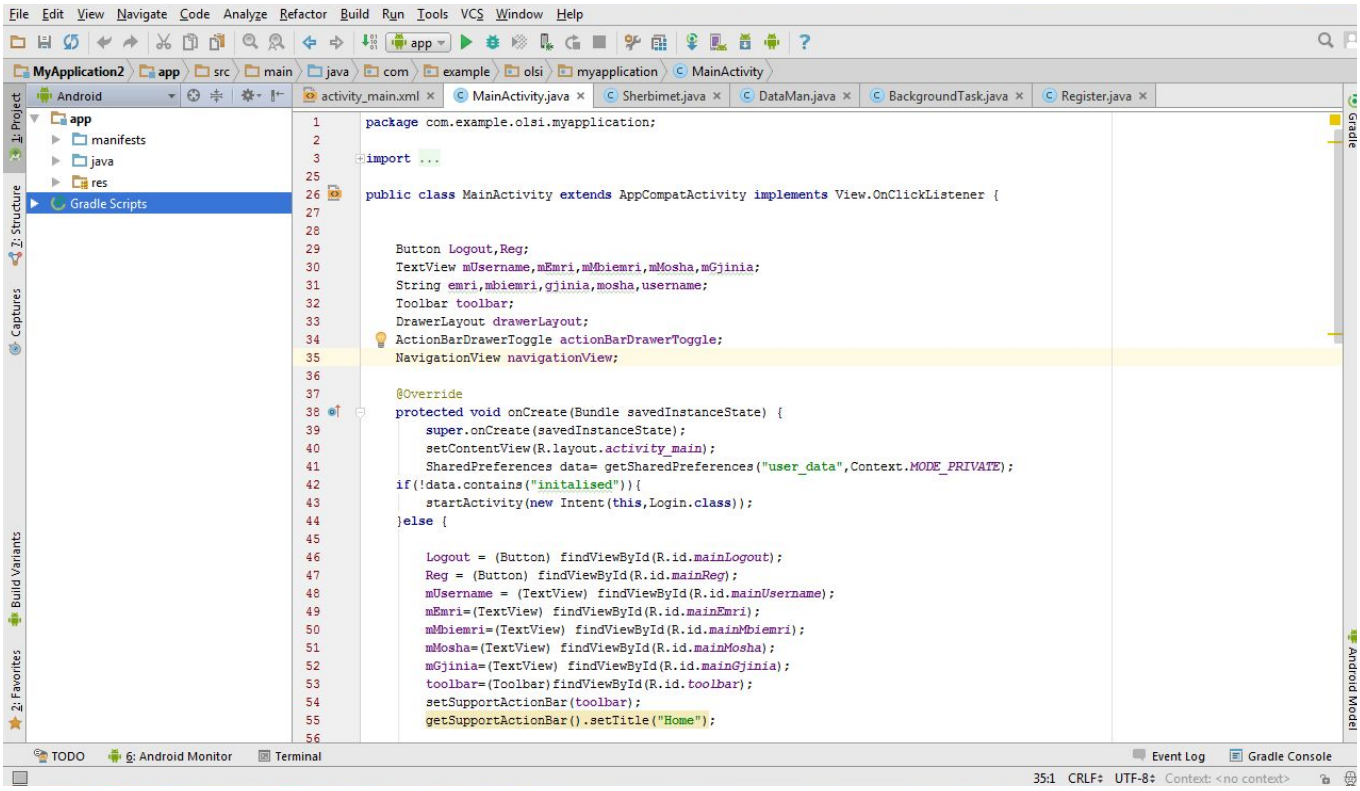


Figura 2.3 Ndërfaqja e Android Studio

### 2.4.1 Struktura e projektit

Çdo project në Android Studio përmban një ose më shumë module filet e source code dhe file burimi. Llojet e moduleve janë:

- Modulet e aplikacionit Android
- Modulet e librarisë
- Modulet e motorrit të aplikacionit Google

Android Studio i shfaq filet e projekteve në panelin anësor të majtë (Android project view). Ky panel është organizuar në module për të ofruar akses më të shpejt në filet apo burimet e projektit. Çdo modul aplikacioni përmban folderat e mëposhtëm:

- ***manifest**:* përmban file-in AndroidManifest.xml
- ***java**:* përmban të gjitha file-t me kod në Java
- ***res**:* përmban të gjitha file-t e tjera që nuk përmbajnë kod, si layout XML

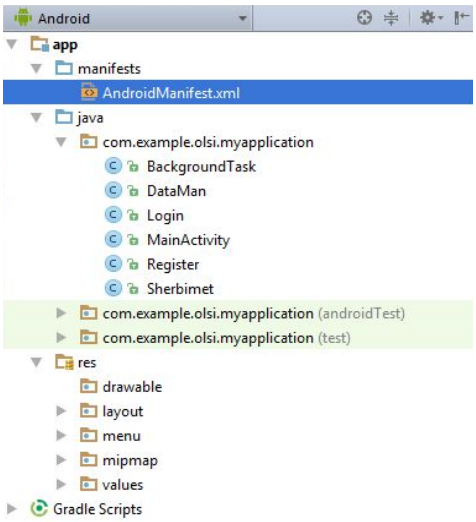


Figura 2.4 Paneli i moduleve

## 3. Krijimi i ndërfaqeve

### 3.1 Ndërfaqja e regjistrimit dhe e login-it

Ndërfaqet e aplikacionit i krijoj në XML (EXtensible Markup Language). XML është një gjuhë e krijuar për të ruajtur të dhëna, ndryshe nga HTML e cila shfaq të dhënat. XML është krijuar për të qenë më e thjeshtë për tu lexuar nga njerëzit dhe makina.

Për krijimin e ndërfaqes përdorim layout-et. Një layout përcakton strukturen pamore (vizuale) të ndërfaqes së përdoruesit. Android ofron një fjalor XML i cili i korrespondon klasës Vieë dhe nënklasave të saj. Avantazhi i deklarimit të ndërfaqes së përdoruesit në XML është se të mundëson një ndarje më të mirë të prezantimit të aplikacionit nga kodi, i cili kontrollon

sjelljen e ndërfaqes. Përshkrimet e ndërfaqes së përdoruesit janë të jashtme për kodin e aplikacionit, pra ndërfaqja mund të modifikohet më thjeshtë pa patur nevojën për të modifikuar apo ndryshuar kodin e aplikacionit.

Për të filluar punën e krijimit të ndërfaqes në panelin anësor ndjekim path-in res->layout->activity_register.xml, që është layout i activity të regjistrimit, aty ku dhe ndodhet file-i XML. Ndërfaqen e ndërtojmë si një *Linear Layout,* që është një grup view i cili i rradhit të gjithë elementet fëmijë në një drejtim të vetëm. Drejtimin mund ta spefikojmë me anë të atributit android:orientation. Në ndërfaqen e regjistrimit do të përdor orientimin vertikal, pra të gjithë fëmijët do të vendosen njëri poshtë tjetrit, një për çdo rresht, pavarësisht gjerësisë së tyre.

Fushat që do të jenë në ndërfaqen e përdoruesit janë respektivisht: emri, mbiemri, gjinia, mosha, username dhe password-i. Këto janë të dhëna të cilat do të merren nga përdoruesi nëpërmjet ndërfaqes dhe më pas do të dërgohen në databazë.

Për të shfaqur tekst përdoret elementi TextView ndërsa për të marr të dhënat nga përdoruesi përdoret elemente të tipit EditText. Më poshtë do të jepen dy shëmbuj të marrë nga file XML i ndërfaqes.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Username"
    android:padding="10dp"/>
```

Sic shihet elementi rrethohet nga tag-et dhe brenda tij vendosen atributet. Me anë të atributit *android: layout_width* dhe *android:layout_height* përcaktohen përmasat e elementit, që në kod do të jenë aq sat ë mbulojnë teksin e vendosur në atributin *android:text.*

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/regUsername"
    android:padding="10dp"/>
```

Elementi EditText shfaq në ekran një fushë ku mund të futen të dhënat. Në këtë element shtohet dhe atributi android:id, me anë të cilit elementit i caktohet një ID, që më pas të kapet dhe aksesohet me anë të kodit për tu marrë teksti.

Për elementin e password-it EditText i shtojmë një atributë shtëse, llojin e input, i cili do të për të fshehur karakteret për arsye sigurie. Atributi do të jetë android:inputType=”textPassword”.

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/regPassword"
```

```
    android:inputType="textPassword"
    android:padding="10dp"/>
```

Pasi ka plotësuar të gjitha fushat me të dhënat përkatëse, përdoruesi do të shtypi butonin "Regjistrohu", i cili do të nisi të dhënat. Butonin do ta krijojmë në XML me anë të elementit Button.

```
<Button
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Regjistrohu"
  android:id="@+id/regButton"/>
```

Duke patur disa fusha, ndërfaqja e përdoruesit është më e madhe sesa madhësia e ekranit të celularit. Në këtë rast do të përdorim ScrollView, i cili lejon layout-in që të jetë më i madh sesa ekrani fizik. Një ScrollView është një FrameLayout, si pasojë duhet vendosur një element fëmijë i cili do të ketë të gjithë përmbajtjen që do të behet scroll. Fëmija vetë mund të jetë një layout manager me një hierarki komplekse objektesh. Fëmija që përdor për aplikacionin në fjalë është një LinearLayout më atributin e orientimit vertikal. Pra, mundësohet një ashensor për të levizur lart ose poshtë në ndërfaqe për të akesuar të gjitha fushat nga përdoruesit.

```
<ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/ScrollView01">
```

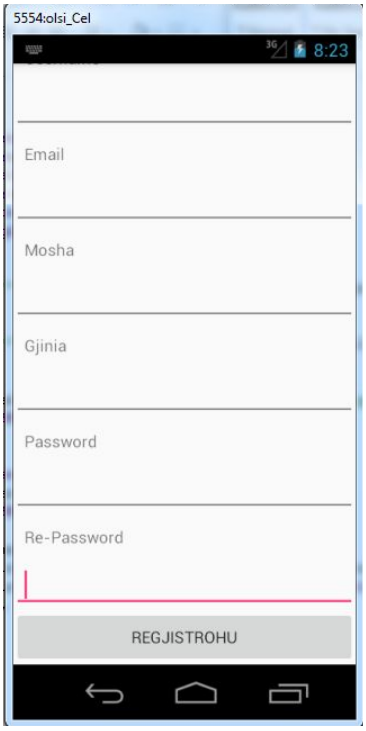


Figura 3.1 Ndërfaqja e regjistrimit

File i emërtuar activity_login.xml do të gjendet në panelin anësor në pathin app->res->layout->activity_login.xml. Përdorim përsëri një LinearLayout për krijimin e ndërfaqes në mënyrë që elementët të vendosen njëri pas tjetrit, duke përdorur atributin orientation:vertical, pra elementët do të vendosen njëri poshtë tjetrit për të krijuar rregull në grafikë.

Kur përdoruesi do të shkoj të logohet një këtë ndërfaqe, atij do t'i shfaqen dy fusha. Ato janë username dhe password-i të cilat janë të nevojshme për të bërë identifikimin e tij. Ashtu si në rastin e rregjistrimit përdor dy elementë: TextView për të shfaqur tekstin se çfarë duhet të vendosi përdouresi në hapësirat boshe dhe EditText të cilat krijijnë fushat kur përdoruesi do të japi të dhenat e tij për të bërë të mundur identifikimin e tij.

Elementëve EditText dhe Button u përcaktohet një ID, nëpërmejt atributit android:id në mënyrë që të behet e mundur kapja e këtyre elementëve në klasat korrespondente të java dhe aksesimit të tyre.

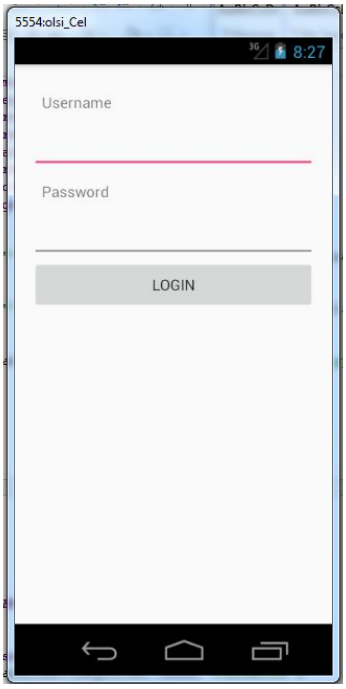

Figura 3.2 Nërfaqja e loginit

### 3.2 Ndërfaqet e activities të tjera

Pasi të logohet përdoruesi do të ketë nevojë për një menu me të cilën të navigojë apo të kalojë nga një activity në një tjetër. Pra një menu është jetësore për aplikacionin, si për atë të klientëve si edhe për atë të administratorit.
Për krijimin e kësaj menuje do të përdoret kontenieri i tipit DrawerLayout, në të cilin do të vendosim si fëmijë një NavigationView. Ky është një panel që shfaq opsionet kryesore të navigimit të aplikacionit në anën e majtë të ekranit. Eshtë i fshehur në pjesën më të madhe të kohës, por shfaqet kur përdoruesi rrëshqet gishtin nga e majta në të djathë të ekranit, ose shtyp butonin në anën e majtë të toolbarit. Ky element shfaqet me gjatësinë e ekranit.

```
<android.support.design.widget.NavigationView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:id="@+id/navigation_menu"
    app:menu="@menu/drawer_menu"
    >

</android.support.design.widget.NavigationView>
```

Në këtë NavigationView do të implementojë një menu në të cilin kam vendosur atributin id:drawer_menu, për ta kapur në një layout tjetër.
Në file-n XML drawer_menu.xml do të krijojë një menu duke vendosur item, të cilët do të shfaqen dhe përdoruesi do t'i klikojë për të naviguar Brenda aplikacionit të vizitave mjekësore.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/profili_id"
            android:title="Profili"
           ></item>
        <item
            android:id="@+id/rreth_nesh_id"
            android:title="Rreth nesh"
            ></item>
        <item
            android:title="Sherbimet"
            android:id="@+id/sherbimet_id"
            ></item>
        <item
            android:title="Rezervimet"
            android:id="@+id/rezervimet_id"
            ></item>
    </group>
    <item android:title="Rregullime">
        <menu>
            <item
                android:title="Modifikime"
                android:id="@+id/modifikime_id"
                ></item>
            <item
                android:title="Logout"
                android:id="@+id/logout_id"></item>
        </menu>
    </item>

</menu>
```

Pra, menuja do të hapet me anë të tag-eve <menu> dhe cdo element i menusë do të vendoset Brenda tag-eve <item>, emri do të përcaktohet me anë të atributit

android:tittle=”Modifikime”, dhe do të shtohet edhe atributi android:id që do të nevojitet në implementimin e listener të NavigationView.

Kjo menu do të ndodhet në të majtë të ekranit, pra duke bërë “swap” ose duke rrëshqikur gishtin nga e majta në të djathtë të aplikacionit do të shfaqet menuja. Për ta bërë më te dukshme do të krijojë një toolbar që do të ndodhet në cdo acivity të aplikacionit dhe do të mbajë dhe emrin e faqes apo activity ku ndodhet user-i. Për më tepër në krahune majtë të aplikacionit do të vendoset butoni i quajtur ndryshe “hamburger” që do të hap menunë. Pra dhe përdoruesit e rinjë do ta dinë sit ë hapin menunë.

Në një file të ri XML do të krijoj layout-in e ri të toolbar-it që do të ndodhet në cdo layout tjetër të aplikacionit, si ai i përdoruesit si dhe ai i administratorit.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/toolbar"
    android:background="?attr/colorPrimaryDark"
    android:minHeight="?attr/actionBarSize"
    android:fitsSystemWindows="true"
    app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    >
</android.support.v7.widget.Toolbar>
```

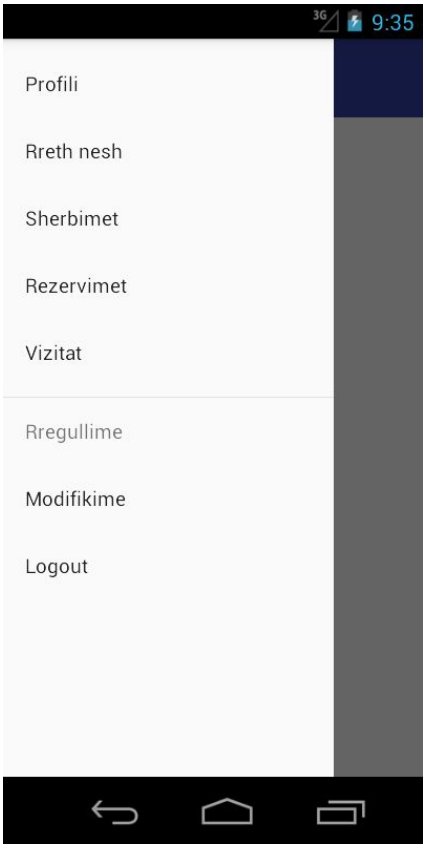


Figura 3.3 Paraqitja e NavigationView

Për ta vendos këtë toolbar të krijuar nga unë në cdo layout të aktiviteteve, përdor tagun <include>

```
<include
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    layout="@layout/toolbar_layout" />
```

Një element tjetër i dobishëm që do të përmend në këtë pjesë të relacionit është element i quajtur Spinner. Ai është një element që mundëson vendosjen e një menuje të tipit drop-down menu. Ky element do të përdoret në pjesën e rezervimit të vizitës, pra kur do të caktohet data, ora dhe muaji i vizitës mjekësore nëpërmjet aplikacionit. Më poshtë do të jepet një shembull, i krijimit të Spinner në file-in XML.

```
<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/spinner_muaji"
    android:layout_margin="10dp"
    android:layout_gravity="center_horizontal"/>
```

Në castin që përdoruesi do të shtyp në menu elementin “Sherbimet”, ai do të presi që në ekranin e celularit të tij të shfaqen të gjitha shërbimet e ofruara prej qendrës spitalore apo poliklinikë. Të dhënat e këtyre shërbimeve do të merren nëpërmjet një stringe JSON dhe do të ruhen në një ArrayList<> të përberë nga objekte të krijuara për t’i ruajtur këto të dhëna. Problemi qëndron në castin që këto të dhëna do të duhet t’i shfaqim në ndërfaqen e përdoruesit. Pra do të ketë një numër të madh objektesh që do të përmbajne të njëjtat lloje informacioni. Për t’i shfaq këto informacione do të krijohet një file XML dhe do të krijohet struktura e shfaqjes të të dhënave në fushat përkatëse. Kjo strukturë do të përsëritet aq herë sa do të jenë objektet, që të shfaqen të gjithë. Prandaj do të përdoret kontenieri i tipit ListView i cili bën të mundur paraqitjen e elementëve të shumtë në mënyrë të përsëritur, njëri mbi tjetrin në orientimin vertikal.

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/xhulio_list_view">
```

**4. Aplikacioni i përdoruesit**

Ky është aplikacioni që cdo përdorues dhe klient do të ketë në aparatin e tij celular Android. Me anë të këtij aplikacioni, sic është thënë dhe më lartë, përdoruesi ose në këtë rast klienti do të ketë mundësitë të shikoj shërbimet që ofron klinika, të rezervoj një takim për një vizitë mjekësore duke përcaktuar orarin, datën dhe muajin. Të shoh se cili është mjeku përgjegjës për secilën lloj vizite. Të shoh rezervimet që ka kryer dhe të anullojë një rezervim në rast nevoje.

Klasat java, duke përfshirë këtu dhe aktivitetet, që e bejnë të mundur krijimin dhe mirë-funksionimin e këtij aplikacioni janë:

1. Klasat e regjistrimit, loginit  profilit të përdoruesit/klientit:
    - Login
    - Register
    - MainActivity
2. Klasat që nevojiten për shfaqjen dhe përpunimin e shërbimeve:
    - Sherbim
    - Sherbimet
    - SherbimAdapter
    - Rezervim
3. Klasat për shfaqjen dhe përpunimin e rezervimeve
    - FutureCheck
    - FutureChecks
    - FutureCheckAdapter

4. Klasa e modifikimit të të dhënave të user-it
    - Modifikime
    - Pop
5. Klasa për prezantimin e klinikës
    - Rreth_Nesh
6. Klasa për komunikimin dhe përpunimin e të dhënave
    - BackgroundTask
    - DataMan

Regjistrimi dhe logini janë dy aktivitete (activity), si të tilla kanë një ndërfaqe grafike me të cilën përdoruesi mund të ndërveprojë. Pra, këto klasa kujdesen për krijimin e një dritareje në të cilën mund të vendoset ndërfaqja nëpërmjet metodes setContentView(View). Për krijimin e aktivitetit përdoret metoda onCreate(Bundle) e cila thirret në fillim të klasës. Metoda setContentView() merr si parametër një layout i cili do të përcaktoj ndërfaqen për këtë aktivitet të caktuar, pra bën lidhjen me ndërfaqen e përdoruesit.

### 4.1 Klasa Register dhe Login

Këto dy klasa mundësojnë krijimin e një user-i të ri dhe login-in e tij në aplikacion në mënryë që të përdori të gjitha funksionalitetet e aplikacionit.

#### 4.1.1 Klasa Register

Klasa Register kujdeset për rregjistrimin e përdoruesit. File-in për të aksesuar këtë klasë e gjejmë në panelin anësor  sipas path-it app->java->Register.

E nisim duke krijuar instancat e tipit EditText për të aksesuar dhe tërhequr të dhënat nga ndërfaqja ku i ka shkruar përdoruesi, dhe disa variabla të tipit string ku të mbajmë këto të dhëna të cilat më pas do t'i dërgohen databazës.

```
EditText regEmri,regMbiemri,regUsername,regPassword,regMosha,
regGjinia,regRePassword,regEmail;
Button Regjistrohu;
String emri,mbiemri,username,password,mosha,gjinia,repassword,email;
```

Më pas për të krijuar (inicializuar) aktivitetin thërritet metoda onCreate() dhe pas saj metoda setContentView() e cila përcakton se cila do të jetë ndërfaqja e aktivitetit të regjistrimit, ku si parametër i jepet file activity_register në XML.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
```

Për të aksesuar elementët e ndërfaqes së përdoruesit thërritet metoda findViewById() ku si parametër merr id e specifikuar në atributin e elementit në XML.

```
regEmri=(EditText)findViewById(R.id.regEmri);
regMbiemri=(EditText)findViewById(R.id.regMbiemri);
regUsername=(EditText)findViewById(R.id.regUsername);
regPassword=(EditText)findViewById(R.id.regPassword);
regRePassword=(EditText)findViewById(R.id.regRePassword);
regEmail=(EditText)findViewById(R.id.regEmail);
regMosha=(EditText)findViewById(R.id.regMosha);
regGjinia=(EditText)findViewById(R.id.regGjinia);
Regjistrohu=(Button)findViewById(R.id.regButton);
```

Për të vazhduar rregjistrimin përdoruesi do të shtyp butonin e titulluar “Rregjistrohu”. Pra, këtij butoni do t’i vendosim një listener i cili do të shkaktoj një event me shtypjen e butonit, praktikisht do të nis të dhënat në databazë. Kjo bëhet nëpërmjet mbishkrimit (override) të metodës onClick(), dhe klasa jonë do të implementojë ndërfaqen (interface) View.onClickListener.

```
@Override
public void onClick(View v) {
    emri = regEmri.getText().toString();
    mbiemri = regMbiemri.getText().toString();
    password = regPassword.getText().toString();
    repassword=regRePassword.getText().toString();
    email=regEmail.getText().toString();
    username = regUsername.getText().toString();
    mosha= regMosha.getText().toString();
    gjinia = regGjinia.getText().toString();
```

Metoda onClick() do të vendosë se çfarë do të ndodhë pasi përdoruesi të shtypi butonin. Në fillim të merren të dhënat nga fushat e ndërfaqes.

```
if(emri.equals("")||mbiemri.equals("")||password.equals("")||repasswor
d.equals("")||email.equals("")||username.equals("")||mosha.equals("")
|| gjinia.equals("")) {Toast.makeText(Register.this,"Plotesoni te
gjitha fushat",Toast.LENGTH_LONG).show();}
```

Më pas kontrolloj nëse përdoruesi nuk ka lënë fusha bosh. Për të parandaluar një regjistrim të gabuar, duke nisur stringje boshe në databazë. Nëse një nga fushat do të jetë bosh, do të shfaqet një mesazh për përdoruesin. Për ta shfaqur mesazhin përdorim Toast, i cili ofron një

feedback të thjeshtë në një dritare pop-up të vogël. Ai mbush vetëm hapësirën e nevojshme dhe aktiviteti ndërkohë qëndron i dukshëm dhe aktiv. Toast e nisim duke përdorur metodën makeText, e cila merr 3 parametra, kontekstin e aplikacionin, një mesazhin që do shfaqet, dhe kohëzgjatjen e toast.

```
    else if(!password.equals(repassword)){
                            Toast.makeText(Register.this,"Password    i
gabuar",Toast.LENGTH_LONG).show();}
```

Kontrollohet që përdoruesi ta ketë shkruar password-in saktë, duke vendosur një fushë të dytë re-password. Nëse përdoruesi ka gabuar gjatë shkrimit të password-it atëherë do të ngrihet një toast për ta lajmëruar atë.

```
else {  String method = "register";
        BackgroundTask backgroundTask = new BackgroundTask(this);
        backgroundTask.execute(method, emri, mbiemri, username,
password, mosha, gjinia,repassword,email);}}
```

Nëse përdoruesi nuk ka lënë fusha boshe ose nuk ka shkruar password-in gabim, të dhënat do t'i kalojnë një klase tjetër, BackgroundTask e cila do të mundësojë komunikimin me databazën. Për të kaluar të dhënat në klasën BackgroundTast përdoret metoda execute() e cila kalon stringat si parametra të klasës AsyncTask e cila shërben për menaxhimin e komunikimit me databazën. Për klasën BackgroundTask, e cila bën të mundur komunikimin me serverin, do të flas në pikat në vazhdim.

### 4.1.2 Klasa Login

Klasa Login do të kujdeset për logimin e përdoruesit. File-in për të aksesuar këtë klasë e gjejmë në panelin të Android Studio anësor sipas path-it app->java->Login.

E nisim duke krijuar instancat e tipit EditText për të aksesuar dhe tërhequr të dhënat nga ndërfaqja ku i ka shkruar përdoruesi, dhe disa variabla të tipit string ku të mbajmë këto të dhëna të cilat më pas do t'i dërgohen databazës. Në ndërfaqen e loginit ka vetëm dy fusha, username dhe password, të nevojshme për të bërë identifikimin e përdoruesit.

```
EditText loginUsername, loginPassword;
Button Login;
```

Aktiviteti krijohet nga metoda onCreate() dhe lidhet me ndërfaqen nëpërmjet metodës setContentView() e cila merr si parametër ndërfaqen e loginit. Elementët aksesohen nëpërmjet metodës findViewById() me anë të ID unike të tyre.

Butonit të loginit i vendosim një listener me anë të metodës setOnClickListener(). Kur të shtypet butoni, do të kontrollohet që fushat nuk janë lënë bosh dhe më pas do të kalohen të dhënat në klasën BackgroundTask me anë të metodës execute().

### 4.2 Komunikimi me serverin dhe databazën

Meqënëse komunikimi me serverin dhe databazën është jetesore për sistemet e menaxhimit, dhe në aplikacionin tim si rast specifik, po dal pak jashtë rradhe për të shpjeguar menyrën sesi do të kalohen të dhënat nga aplikacioni drejt serverit që me pas t'i ruajë ato në tabelat e posacme të databazës. Scriptet e serverit të cilët merren me përpunimin e të dhënave janë në gjuhën PHP.

#### 4.2.1 Koncepti i threading

Një aplikacion Android ka të paktën një thread kryesor. Ky thread krijohet në të njëjtën kohë që krijohet klasa Application për aplikacionin Android. Përgjegjësitë e thead-it kryesor janë të vizatojë ndërfaqen e përdoruesit, të menaxhojë ndërveprimin me përdoruesin, të vizatojë pixel-a në ekran, të nisi aktivitete. Çdo pjesë kodi që shtohet në një aktivitet përdorin burimet e mbetura nga thread-i kryesor, pra thread i UI (ndërfaqes grafike), në mënyrë që aplikacioni të jetë në gatishmëri (responsive) të përdoruesit.

Ka disa rreshta kod të cilat mund të ekzekutohen për një kohë shumë të gjatë, saqë nuk lejohen nga platforma Android të ekzekutohen në threadin kryesor, pra atë të ndërfaqes se përdoruesit. Një shembull, të cilin do ta përdor dhe për krijimin dhe funksionimin e aplikacionit tim, është thirrja në rrjet (network call). Mund të krijojmë një instancë të URLConnection dhe ta bëj thirrjen e rrjetit në aktivitetin që ekzekutohet në thread-in kryesor. Kodi do të kompilohet mirë, por gjatë kohës së ekzekutimit do të hidhet një përjashtim (exception) i tipit NetworkOnMainThreadException. Pra, ky është një përjashtim që nxirret kur një aplikacion kërkon të kryej detyra në rrjet në thread-in kryesor. Ky përjashtim nxirret për aplikacione të cilat synojnë të performojnë në sistemin operativ Android 3.0 Honeycomb ose API më të larta. Aplikacionet të cilat performojnë në versione më të vjetra janë të lejuara të kryejnë thirrje në rrjet në thread-in kryesor.

Rasti i një thirrje apo detyre në rrjet është një rast ekstrem. Mund të ketë plot detyra të tjera, të cilat nëse shtohen shumë, shtojnë punën, dhe kjo sjell si pasojë ndalimin apo ndarjen e kohës me thread-in kryesor. Në këtë rast aplikacioni ka tendencën të ndalojë ose të bllokohet. Përdoruesi do të mendojë se çfarë ka që nuk shkon me aplikacionin. Në rastin më të keq përdoruesit mund t'i shfaqet një mesazh i tipit error, i cili thotë "Application not responding.

Would you like to wait or kill the application". Pra rikthehemi në rastin e përmendur më lart ku aplikacioni nuk përgjigjet dhe përdoruesi do të detyrohet ta mbylli aplikacionin, që është diçka e pakëndëshme.

> "Çdo thread rezervon një vend në memorie i cili përdoret kryesisht për të ruajtur variablat lokale dhe parametrat gjatë ekzekutimit të një thread-i. Vendi në memorie rezervohet në castin kur krijohet thread-i dhe lirohet kur ai mbaron"
> - Anders Goransson, Efficient Android Threading

### 4.2.2 AsyncTask

AsyncTask është një klasë abstrakte e ofruar nga Android e cila ndihmon në përdorimin më të mirë të thread-it të ndërfaqes. Kjo klasë lejon të kryhen detyra në background dhe shfaq rezultatet në thread-in kryesor pa patur nevojë të manipulohen threads të ndryshëm apo handlers.

AsyncTask është projektuar për të qenë një klasë ndihmëse ndërmjet Thread dhe Handler. AsyncTask duhet të përdoret për detyra të shkurtra.

AsyncTask ka katër etapa:

1. onPreExecute(), thërritet në thread-in e ndërfaqes së përdoruesit përpara se të kryhet detyra. Ky hap përdoret zakonisht për të ngritur detyrën.
2. doInBackground(Params…), thirret në thread-in background menjëherë pasi onPreExecute() mbaron së ekzekutuari. Ky hap përdoret për të kryer përpunime dhe detyra në background të cilat mund të marrin pak kohë. Parametrat e asynchronous task kalohen në këtë hap. Rezultati i përpunimit do të kthehet nga ky hap dhe do t'i kalohen hapit të fundit. Gjatë këtij hapi mund të përdoret edhe metoda publishProgress(Progress…) për të publikuar një ose më shumë njësi të progresit. Këto vlera publikohen në thread-in kryesor, në hapin onProgressUpdate(Progress…)
3. onProgressUpdate(Progress…), thirret në thread-in kryesor, atë të ndërfaqes së përdoruesit, pas një thirrjë nga publishProgress(Progress…). Koha e ekzekutimit është e papërcaktuar. Kjo metodë përdoret për të shfaq çdo llojë progresi në ndërfaqen e përdoruesit ndërkohë që vijon përpunimi në background. Për shembull, mund të përdoret për të animuar një progress bar apo të shfaq ditarin në në fushe shkrimi tekst.
4. onPostExecute(Result), thirret në thread-in kryesor pasi ka mbaruar hapi doInBackground(), pra ka mbaruar përpunimi në background. Rezultati i doInBackground() i kalohet këtij hapi si një parametër.

AsyncTask përdor tre lloje parametrash: Params, Progress dhe Result, të cilat janë parametrat gjatë fazave të quajtura dhe ndryshe generic types. Ato shprehen në castin që trashëgohet klasa AsyncTask, AsyncTask<Params,Progress,Result>.

1. Params: lloji i parametrave i cili dërgohet përpara ekzekutimit të detyrës
2. Progress: lloji i njësive të progresit që publikohen gjatë përpunimit në background
3. Result: lloji i përfundimeve që vijnë nga përpunimi në background

### 4.2.3 Klasa BackgroundTask

Duke qenë se gjuha e përdorur për krijimin e një aplikacioni Android është Java, e cila është një gjuhe e orientuar drejt objektit dhe na lehtëson punë me përdorimin e objekteve ne kodin tonë, krijoj një klasë të re e quajtur BackgroundTask e cila do të bëj të mundur lidhjen me databazën. Kjo klasë do të trashëgojë klasën AsyncTask për të lehtesuar lidhjen dhe komunikimin, duke hequr ngarkesën nga threadi kryesor.

```
public class BackgroundTask extends AsyncTask<String,Void,String>
```

Sic shihet llojet e përgjithshme (generic types) që do të marrë AsyncTask janë string për Params dhe string për Result, ndërsa lloji Progress do të jetë void pasi nuk shfaqet asnjë njësi apo mesazh progresi gjatë punës në background. Pra, do të kalohen disa parametra të tipit string, të cilat do të jenë username dhe password në rastin e loginit për të bërë të mundur identifikimin e përdoruesit. Dhe do të merren disa parametra të tipit string, të cilat do të jenë të dhënat e tij, të marra nga databaza, të cilat do të shfaqen në profilin e përdoruesit. Në klasën Login dhe Register krijoj një objekt të tipit BackgroundTask. Në listenerin e butonave të login dhe regjistrim do të thirret metoda .execute() me anë të së cilës do të kalojnë parametrat e parë,Params, të cilat do të përdoren nga metoda doInBackground().

```
String method = "login";
BackgroundTask backgroundTask = new BackgroundTask(this);
backgroundTask.execute(method,username, password);
```

Më sipër kam marrë shembullin e thirrjes së metodës execute në klasën login. Sic shihet ndodhet dhe një string tjetër e emërtuar method, e cila shërben si kontroll për të kuptuar nëse do të komunikohet për të identifukuar një login apo për të rregjistruar një përdorues të ri.

Parametrat e marrë nga ndërfaqja e përdoruesit, do të ruhen në stringat përkatese dhe më pasë do të kalohen në metodën doInBackground(String…params), të cilën e mbishkruajmë për të bërë të mundur thirrjen në rrjet, pra komunikimin me databazën, duke kryer përpunimet e nevojshme në një thread background.

```
String method=params[0];
```

Të dhënat do të kalojnë si një vektor stringjesh params.Në fillim do të merret parametri i parë, pra stringu e parë, për të kontrolluar nëse do të jetë rasti i një regjistrimi apo i nevojës për tu identifikuar si një përdorues ekzistues. Me anë të kushtit if do të behet i mundur kontrolli, pra do të kemi dy rastet:

- `if(method.equals("register"))`
- `else if(method.equals("login"))`

Këto dy raste janë për të ilustruar shembujt e mësipërm, pasi klasa BackgroundTask menaxhon komunikimin me serverin e cdo klase të aplikacionit. Më pas do të vazhdohet me marrjen e parametrave të tjerë të nevojshëm.

Do të krijohet një instancë e klasës URL (Uniform Resource Locator) që do të nevojitet për të gjetur vendodhjen e një burimi në Internet. Adresa e serverit me të cilin do të duhet të komunikojë aplikacioni do t'i kalohet si parametër në deklarimin e instancës URL. Për të dërguar dhe marrë të dhëna do të krijohet një objekt i klasës HttpURLConnection duke thirrur URL.openConnection() e cila kthen një lidhje të re me burimin e specifikuar nga kjo URL,dhe më pas e hedh (casting) rezultatin tek HttpURLConnection. Vendosim një flamur i cili do të tregojë se kjo URLConnection do të lejojë output dhe input të të dhënave, gjë që nuk mund të bëhet pasi lidhja të vendoset. Kjo bëhet e mundur duke vendosur në true vlerën e tipin buleane (boolean) në metodën setDoOutput() dhe setDoInput(). Përcaktojmë dhe metodën, e cila do të jetë POST, me anë të metodës setRequestMethod().

```
String login_url="http://10.0.2.2/ServerSide/Login.php";
URL url=new URL(login_url);
HttpURLConnection httpURLConnection= (HttpURLConnection)
url.openConnection();
httpURLConnection.setRequestMethod("POST");
httpURLConnection.setDoInput(true);
httpURLConnection.setDoOutput(true);
```

Për dërgimin e të dhënave do të përdoret OutputStream dhe për marrjen e tyre do të përdoret InputStream, të cilat e bëjne më të thjeshtë.Nuk ka rëndësi nëse stream është një file apo string sepse mënyra e dërgimit dhe marrjes do të jetë e njëjta. Të dhënat të cilat do t'i nisen serverit do të buferohen më përparë nëpërmjet BufferedWriter, i cili mbështjell një Writer ekzistues dhe bufferon output-in.

```
OutputStream outputStream=httpURLConnection.getOutputStream();
BufferedWriter bufferedWriter=new
BufferedWriter(newOutputStreamWriter(outputStream,"UTF-8"));
```

Në një string të quajtur data do të ruhen të dhënat të cilat do t'i nisen serverit. Përpara se të nisen këto të dhëna do të enkodohen sipas formatit UTF-8. Për ta bërë këtë përdorim klasën

URLEncoder dhe metodën .encode(), duke përdorur ciftin celës-vlerë (key-value). Të gjitha karakteret përvec shkronjave (‘a-z’,’A-Z’), shifrave (0…9) dhe karaktereve '.', '-', '*', '_' do të kthehen në vlerën hekzadecimale përkatëse të paraprira nga %.

```
String data=URLEncoder.encode("username","UTF-8")  + "=" +
URLEncoder.encode(username,"UTF-8")+"&"+
URLEncoder.encode("password","UTF8")+"="+URLEncoder.encode(password,"U
TF-8");
```

Pastaj ky string që përmban të dhënat do të shkruhet në objektin bufferedWriter që krijuam dhe do të dërgohen duke thirrë .flush(). Pasi të jenë nisur, do të mbyllim outputstream-in dhe bufferedWriter.

```
bufferedWriter.write(data);
bufferedWriter.flush();
bufferedWriter.close();
outputStream.close();
```

Pasi nisen të dhënat drejt serverit të specifikuar si parametër gjatë krijimit të objektit URL, do të pritet një përgjigje prej serverit. Kjo përgjigje mund të përmbajë të dhëna, ose një komunikim nga server nëse ndodh një dështim apo ka probleme të tjera. Për të marrë përgjigjen nga serveri do të përdoret një objekt i InputStream dhe një BufferedReader, i cili është një klasë për të lexuar text nga një inputstream që përmban text duke i grumbulluar  (buffer)në mënyrë që të ofrojë një lexim efiçient të karaktereve, vektorëve dhe rreshtave.Madhësia e buffer-it nuk është e specifikuar.

```
InputStream inputStream=httpURLConnection.getInputStream();
  BufferedReader bufferedReader=new BufferedReader(new
InputStreamReader(inputStream,"iso-8859-1"));
  StringBuffer stringBuffer=new StringBuffer();
  String line="";
  while((line=bufferedReader.readLine())!=null){
      stringBuffer.append(line);
  }
bufferedReader.close();
  inputStream.close();
  httpURLConnection.disconnect();
  return stringBuffer.toString();
```

Pasi të lexohet mesazhi i ardhur nga server, ai do të kthehet dhe kjo string do të kalojë si parametër i llojit Result metodës onPostExecute().

Metoda e mbishkruar onPostExecute(String…result) do të menaxhojë mesazhin e komunikuar prej serverit. Nëpërmjet kësaj metode do të përcaktohet se çfarë do të ndodh nëse vjen një mesazh dështimi apo të dhënat e përdoruesit të sapo identifikuar në databazën e përdorur prej aplikacionit. Në rastin kur regjistrimi ka ndodh me sukses, apo ka një përdorues ekzistues të regjistruar me këto të dhëna, server do të nis një mesazh, i cili do të përpunohet në onPostExecute dhe ky mesazh do t'i shfaqet përdoruesit nëpërmjet AlertDialog.

Në rastin kur përdoruesi kërkon të logohet, pasi ndodh autentikimi në server, të dhënat e tij do t'i dërgohen aplikacionit në formën e një stringe të tipit JSON. Kjo stringë do të parsohet nëpërmjet metodës .user_data() të klasës së krijuar DataMan për menaxhimin e parsimit të të dhënave.

```
protected void onPostExecute(String result) {
    DataMan dataMan=new DataMan(result);
    if(result.equals("Not Found")){
        alertDialog.setMessage(result);
        alertDialog.show(); }
    else if(result.equals("Success")){
        Intent intent=new Intent(ctx, MainActivity.class);
        ctx.startActivity(intent);}}
```

### 4.2.4 JSON- format i marrjes së të dhënave

JSON (JavaScript Object Notation) është një format me peshë të lehtë për shkëmbimin e të dhënave. Ky format është i pavarur nga gjuha e programimit që perdoret dhe është i thjeshtë për tu kuptuar nga njerëzit, dhe për tu ndarë (parse) dhe gjeneruar nga makinat.

JSON është i ndërton në dy struktura:

- Një bashkim i cifteve celës-vlerë, që shihet si një objekt
- Një listë e renditur, që shihet një një vektor

Objektet në stringun JSON shprehen ndërmjet dy kllapave gjarpërueshe ( {} ) dhe përmbajnë ciftet celës- vlerë. Këto cifte ndahen me dypikësh ( : ), ciftet ndërmjet tyre ndahen me presje. Ndërsa vektorët në stringen JSON shprehen nëpërmjet kllapave katrore ( [] ).

Më poshtë do të jepet shembulli i stringës që server do t'i dërgojë aplikacionit kur përdoruesi me username: xhuliodo do të kryejë login-in.

{
"response":
{"id":"1","emri":"Xhulio","mbiemri":"Doda","mosha":"21","gjinia":"mashkull","username":"xhuliodo","password":"123"}

}

Pra, ka vetëm një objekt me emrin “response”, ndërsa në rastin kur përdoruesi do të kërkojë të shoh shërbimet, serveri do t’i dërgojë një string të gjatë, që përbehet nga një objekt, që përmban një vektor, elementët e të cilit janë objektet që përmbajnë të dhënat mbi shërbimet.

### 4.3 Klasa MainActivity dhe DataMan

Klasa MainActivity është klasa kryesore e aplikacionit, kjo është klasa (aktiviteti) që do të hapet e para kur përdoruesi të shtyp mbi ikonën e aplikacionit. Nëse përdoruesi është loguar më përpara do të shfaqet direkt kjo klasë, nëse përdoruesi nuk është i loguar në aplikacion, atëherë ai do të drejtohet në klasën login.

E nisim rradhën sipas një rendi llogjik, duke filluar nga hapi loginit. Përdoruesi do të vendos të dhënat e veta në fushat përkatëse të llojit EditText, këto do të merren dhe do të ruhen në stringat përkatëse në klasën Login.class dhe nëpërmjet një instance të krijuar nga klasa BackgroundTask do të merrën me metodën .execute(Params…) . Më pas këto të dhëna do t’i dërgohen serverit duke përdorur metodën POST. Në anën e serverit këto të dhëna do t’i marri file i quajtur Login.php që do të aksësohet nëpërmjet URL= “http://10.0.2.2/ServerSide/Login.php”. Duke e parë nga ana e serverit, brenda në file-in e loginit, të dhënat do të merren më anë të vektorit të stringut $_POST[] dhe do të ruhen në dy variabla.

Më pas server do të kërkojë në databazë nëse këto të dhëna janë të sakta, duke dërguar query në MYSQL. Nëse të dhënat gjenden atëherë, scripti në PHP do të krijojë një string JSON dhe do ta printojë, ndërkohë që aplikacioni rri në pritje.

```
echo json_encode(array('response'=>$user));
```

Nëse nuk gjendet, atëherë do të printohet një mesazh tjetër.

```
echo "Not Found";
```

Në castin që në metodën doInBackground() të klasës BackgroundTask do të vij një string, atëherë stringu do t’i kalohet metodës onPostExecute(), në të cilën do të kontrollojë nëse stringu përmban mesazhin “Not Found” apo stringun JSON, e cila përmban një vektor të titulluar response.

```
if(result.startsWith("response",2)){
    dataMan.user_data(ctx);}
```

Në rastin që logini u krye me sukses dhë në aplikacion vijnë të dhënat e përdoruesit, metoda onPostExecute() do të krijojë një instancë të klasës DataMan e cila parson të dhënat e përdoruesit dhe i ruan ato në një file të tipit SharedPreferences. Android lejon të ruhen të dhëna të një aplikacioni në një file, cili përbëhet nga cifti celës-vlerë. Edhe kur përdoruesi del prej aplikacionit pa bërë logout, ky file do të ekzistojë.

```
public void user_data(Context ctx){
        try {
            JSONObject parentobject=new JSONObject(data);
            JSONObject finalObject =
parentobject.getJSONObject("response");
            id=finalObject.getString("id");
            emri=finalObject.getString("emri");
            mbiemri=finalObject.getString("mbiemri");
            username=finalObject.getString("username");
            gjinia=finalObject.getString("gjinia");
            mosha=finalObject.getString("mosha");
        } catch (JSONException e) {
            e.printStackTrace();
        }

    SharedPreferences
pref=ctx.getSharedPreferences("user_data",Context.MODE_PRIVATE);
     SharedPreferences.Editor editor=pref.edit();
     editor.putString("emri",emri);
     editor.putString("mbiemri",mbiemri);
     editor.putString("username",username);
     editor.putString("gjinia",gjinia);
     editor.putString("mosha",mosha);
     editor.putString("id",id);
     editor.putBoolean("initalised",true);
     editor.putBoolean("logged_in",true);
     editor.commit();
     Intent intent=new Intent(ctx,MainActivity.class);
     ctx.startActivity(intent);}
```

Pra sic shihet dhe nga kodi, në metodë do të kalohet stringu i dërguar nga server, ku ai do të parsohet duke kërkuar vlerat sipas celësave përkatës, dhe më pas këto vlera do të ruhen në filen e tipit SharedPreferences. Dhe më pas do të niset aktiviteti MainActivity.

Në metodën onCreate() të klasës MainActivity do të krijohet një objekt i tipit file i cili do të përmbaje filen e sapokrijuar, duke marrë si parametër path-in e tij.

```
File file=new File("/data/data/"+getPackageName()+ "/shared_prefs/" +
"user_data.xml");
```

Me anë të kushtit "if" do të kontrollojë nëse ky file ekziston ( file.exists() ), dhe nëse file-i nuk ekziston atëherë do të startohet aktiviteti Login.class, pra përdoruesi do të shkojë në panelin e loginit. Në të kundërt të dhënat do të merren nga file dhë do të vendosen në fushat

përkatëse me metodën .setText(String string). Ndërfaqja e MainAcitivity, ose sic i paraqitet përdoruesit si Profili, do të këtë një container të tipit GridLayout, i cili mundëson 2 kolona për cdo rresht. Natyrisht në këtë aktivitet janë krijuar instancat e llojit Toolbar dhe NavigationView për t'i mundësuar përdoruesit navigimin në aplikacion.


Figura 4.1 Profili i përdoruesit

### 4.4 Grupi i klasave për menaxhimin e shërbimeve dhe vizitat e rezervuara

Për menaxhimin e sherbimeve dhe rezervimin specifikit të një shërbimi nga ana e përdoruesit kam krijuar dhe përdoren tre klasat: Shërbimi, Shërbimet dhe ShërbimAdapter. Ndërsa për të parë vizitat e rezervuara dhe anullimin e një rezervimi specifik kam krijuar dhe do të përdoren klasat: FututeCheck, FutureChecks dhe FutureCheckAdapter. Këto klasa vendosa t'i trajtoj së bashku pasi funksionimi i tyre llogjik është i ngjashëm.

#### 4.4.1 Klasa Shërbimi dhe FutureCheck

Pasi të dhënat mbi shërbimet e ofruara dhe rezervimet e kryera do të vijnë me anë të stringut sipas formatit JSON, është e nevojshme që këto të dhëna të ruhen në objekte. Për të mundësuar krijimin e këtyre objekteve krijova dy klasat: Shërbimi për të ruajtur të dhënat mbi shërbimet e ofruara, dhe FutureCheck për të ruajtur të dhënat mbi rezervimet e kryera.

Të dhënat që do të mbajë një objekt i tipit Shërbimi janë: id, shërbimi, lloji, përshkrimi dhe mjeku. Ndërsa të dhënat që do të mbajë një objekt i tipit FutureCheck janë: id, sherbimi, ora, mjeku , data, muaji , ora, viti dhe kontroll.

Id do të mbajë vlerën përkatëse te ID të shërbimit apo rezervimit përkatës, i cili është unik në database. Ndërsa kontroll do të mbajë një numër i cili do të shërbejë si kontroll, tre janë rastet:

- Rezervimi është aktiv kontroll=0
- Rezervimi është kryer kontroll=1
- Rezervimi është anulluar kontroll=-1

Secila nga këto klasa ka konstruktorin e vetë, në të cilin merrën vlerat dhe u caktohen secilës prej fushave. Këto fusha do të jenë privatë dhe do të aksesohen me anë të metodave String get(), që do te jenë për cdo fushe dhe do të kthejnë vlerën e fushës.

### 4.4.2 Klasa ShërbimAdapter dhe FutureCheckAdapter

Ndërfaqja e shërbimeve dhe e rezervimeve të kryera do t'i shfaqi përdoruesit një listë me disa element. Për ta bërë të mundur këtë kam përdoru elemenintin ListView në layout-in e seciles prej dy aktiviteteve. By default, ListView shfaq vetëm një rresht për cdo artikull (item), pra mund të shfaqi një vektor, ose më mirë një ArrayList me stringa e deklaruar si ArrayList<String>. Por në rastin e këtij aplikacioni, ajo cfarë dua të shfaq në këtë ListView nuk janë stringje, por objektet e krijuara nga klasat e sipërpërmendura: Sherbimi dhe FutureCheck.

Për këtë do të përdoren dy klasat: ShërbimAdapter dhe FutureCheckAdapter, të cilat do të veprojnë si përshtatës për të vendosur cdo vlerë të objektit në vendin e duhur, kontenierin e tipit TextView dhe butonat, dhe shfaqur të gjitha objektet përkatëse.

Të dyja këto klasa do të trashëgojnë klasën ArrayAdapter të përbëre nga elementet, që jane objekte të këtyre dy klasave.

```
public class SherbimAdapter extends ArrayAdapter<Sherbimi>
public class FutureCheckAdapter extends ArrayAdapter<FutureCheck>
```

Të dyja këto klasa do të përmbajnë metodën getView(), e cila do të bëj të mundur vendosjën e teksit(vlerës) në kontenierin e vet. Fillimisht do të krijojë një instancë të klasës LayoutInflater, që do të na duhet për të marrë file-in XML që do të shërbejë si strukturë për paraqitjen e objektit, dhe ia kalojmë këtë nje objekti të klasës View nëpërmjet metodës .inflate().

```
public View getView(final int position, View convertView, ViewGroup
parent) {

LayoutInflater xhulioInflater= LayoutInflater.from(getContext());
 View
v=xhulioInflater.inflate(R.layout.sherbimet_list_view,parent,false);

          item = getItem(position);
TextView
sherbimi_view=(TextView)v.findViewById(R.id.sherbimet_sherbimi_id);
TextView lloji_view
=(TextView)v.findViewById(R.id.sherbimet_lloji_id);
TextViewpershkrimi_view=(TextView)v.findViewById(R.id.sherbimet_persh
krimi_id);
 TextView
mjeku_view=(TextView)v.findViewById(R.id.sherbimet_mjeku_id);

         sherbimi_view.setText(item.getSherbimi());
         lloji_view.setText(item.getLloji());
         pershkrimi_view.setText(item.getPershkrimi());
         mjeku_view.setText(item.getMjeku());
         sherbim_id=item.getId();

         rezervo=(Button) v.findViewById(R.id.sherbimet_rezervo);
         rezervo.setTag(position);
         rezervo.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View v) {
                  Intent intent=new Intent(activity,Rezervim.class);
                  intent.putExtra("sherbim_id",sherbim_id);
                  activity.startActivity(intent); }
         });
         return v;}
```

Nëpërmjet këtij objekti të klasës Vieë mund të kap dhe të aksesojë të gjitha fushat e filet XML, të krijuar më përparë që do të shërbejë si strukturë për paraqitjen e këtyre të dhënave. Objekti item është një objekt i klasës Shërbimi, i cili do të marrë pozicionin se ku do të vendoset në ListView, me anë të numrit int position. Krijimi i objekteve dhe popullimi i ArrayList-es do të bëhët në klasat FutureChecks dhe Shërbimet.

Brenda në metodën getView() është mbishkruar dhe metoda .setOnClickListener() e cila do të përcaktojë se cfarë do të ndodhë në rastin se shtypet një buton. Problemi qëndron se duke pasur shumë artikuj në ListView do të nevojitet të dihet se në cilin prej artikujve është shtypur ky

buton dhe cfarë do të kryejë specifikisht. Sic shihet më lartë, në klasën ShërbimAdapter ky buton do të nis një aktivitet tjetër për caktimin e kohës së vizitës, pra datës, orës, muajit e vitit. Ndërsa në FutureCheckAdapter do të shërbejë për të anulluar një rezervim të kryer.

### 4.4.3 Klasa Sherbime dhe FutureChecks

Këto dy klasa janë klasat që do të shfaqin shërbimet dhe rezervimet e kryera apo anulluara nëpërmjet ndihës së dy klasave të tjera. Ato do të lidhen me filet XML përkatëse të cilat do të përmbajnë si cdo aktivitet tjetër toolbar-in dhe navigation view, për më tepër do të kenë dhe elementet ListView, të cilave do t'ju atrubuohet një ID përkatese.

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/xhulio_list_view">
</ListView>
```

Këto aktivitete do të inicializohen sipas mënyrës së mëtejshmë. Përdoruesi do të shtypi në navigation view Shërbimet apo Rezervimet, në listener do të krijohet një instancë e klasës BackgroundTask dhe do të kalohen parametrat nëpërmjet metodës .execute(). Në rastin e shërbimeve të ofruara do të kalohet vetëm një string i quajtur method që do të përmbajë se cfarë kërkon të aksesojë përdoruesi. Në rastin e Rezervimeve vec stringut method që do të ketë përmbajtjen "checkUps" dhe do të merret Id e përdoruesit e ruajtur në file-in SharedPreferences në menyrë që nga databaza të tërhiqen vetëm rezervimet e kryera nga ky përdorues, jo të gjitha.

Në castin që klasa BackgroundTask do të marrë stringun JSON ajo do t'ja kalojë klasave Sherbim dhe FutureCheckUps të cilat do të kujdesen për parsimin e stringës. Në ndryshim nga klasa MainActivity e cila ka vetëm një objekt, në keto dy rastë stringa përmban një objekt, me një vektor objektesh brenda.

```
String data=getIntent().getExtras().getString("data");
sherbimiArrayList=new ArrayList<Sherbimi>();
try {
    JSONObject primeObject=new JSONObject(data);
    JSONArray jsonArray=primeObject.getJSONArray("sherbimet");

    for(int i=0;i<jsonArray.length();i++){
        JSONObject jobj=jsonArray.getJSONObject(i);
        String id=jobj.getString("id_sherbim");
        String sherbimi=jobj.getString("sherbimi");
        String lloji=jobj.getString("lloji");
        String pershkrimi=jobj.getString("pershkrimi");
```

```
        String mjeku=jobj.getString("mjeku");
  sherbimiArrayList.add(i,new
Sherbimi(id,sherbimi,lloji,pershkrimi,mjeku));}
```

Të dhënat do të merren dhe parsohen nëpërmjet një cikli for dhe njëkohësisht do të bëhet popullimi i një ArrayList-e që përbehet nga objektet e këtyre dy klasave. Pas popullimit të ArrayList do të bëhet popullimi i ListView duke përdorur përshatësit që krijuam:

```
ListAdapter itemList= new
SherbimAdapter(this,R.layout.activity_sherbimet,sherbimiArrayList);
ListView sherbimiList=(ListView)findViewById(R.id.xhulio_list_view);
if (sherbimiList != null) {
    sherbimiList.setAdapter(itemList);
}
```

Pas këtij hapi lista është plotësuar dhe puna ka mbaruar.


Figura 4.2 Aktiviteti i Sherbimeve


Figura 4.3 Aktiviteti i Rezervimeve të kryera

### 4.4.4 Klasa Rezervim

Kjo klasë është përgjegjëse për caktimin e datës muajit, e orës për vizitën që përdoruesi dëshiron të kryejë. Ky aktivitet do të niset kur përdoruesi do të shtyp mbi butonin Rezervo në aktivitetin Shërbime. Në castin e shtypjës së butonit kësaj klase do t'i kalohet ID e shërbimit që përdoruesi ka dëshiruar të rezervojë.

Ky aktivitet do të shfaqet si një pop-up, për këtë do të vendosim kufijtë e aktivitetit më të vogla sesa madhësia e ekranit, duke marrë parametrat e gjerësisë dhe lartësise dhe duke i shumezuar ato respektivisht me 0.8 dhe 0.9 .

```
DisplayMetrics dm=new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(dm);
int width=dm.widthPixels;
int height=dm.heightPixels;
getWindow().setLayout((int)(width*.9),(int)(height*.8))
```

Për caktimin e kohës së dëshiruar për rezervimin kam përdorur Spinners, të cilët janë menu të tipit drop-doën. Në këtë mënuyrë përdoruesi do të shtyp mbi spinner dhe do t'i hapet një listë që do të përmbajë mundësitë që mund të zgjedh, dhe më pas do të shtyp butonin të titulluar "Kryej Rezervimin".

Në listener të këtij butoni do të merret ID e klientit nga file i ruajtur në castin e loginit dhe ID e shërbimit të dëshiruar që i kalohet nga aktiviteti i Sherbimeve. Më pas do të thirret instance e klasës BackgroundTask  dhe këto stringa sëbashku me stringën method që tregon së cfarë do të kryhet i kalohen klasës BackgroundTask.

### 4.5 Klasa për modifikimin e të dhënave të përdoruesit

Për modifikim e të dhënave kam krijuar dy aktivitete të quajtura Modifikime dhe Pop(e merr emrin sepse është një pop-up).

Në aktivitetin Modifikime përdoruesit do t'i shfaqet një ndërfaqe ku mund të zgjedhi se cilën prej të dhënave dëshiron të ndryshojë dhe duke shtypur mbi butonin përkatës atëhere do të niset aktiviteti Pop, i cili është një pop-up  dhe aty do të shkruajë të dhënën e re dhe do të dërgojë atë në server për ndryshimin në fushën përkatëse në databazë.

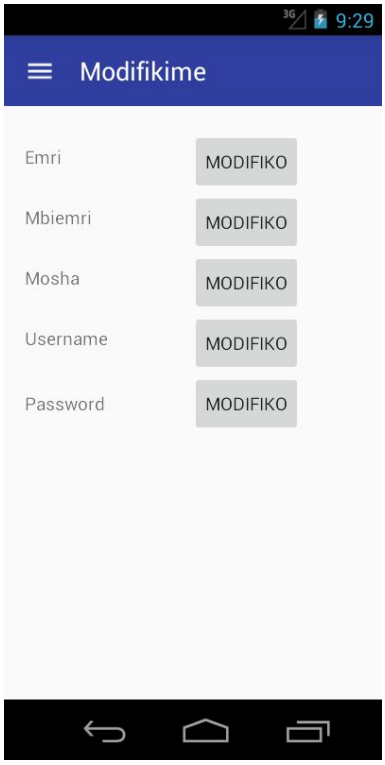


Figura 4.4 Aktiviteti Modifikime

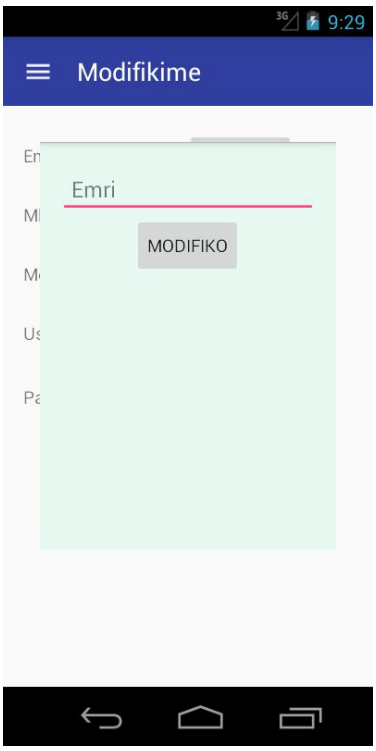


Figura 4.5 Aktiviteti Pop

### 4.6 Klasa Rreth_Nesh

Kjo klasë është pjesë e aktivitetit Rreth_Nesh, i cili shfaq të dhënat rreth klinikës. Këto të dhëna jepen nga administatori nëpërmjet aplikacionit të administatorit. Ky aktivitet niset duke shtypur mbi opsionin Rreth_Nesh në menunë anësore. Në listener do të krijohet një instancë e klasës BackgroundTask e cila menaxhon të gjitha komunikimet me serverin, dhe do të kalohet vëtëm një parametër, që është stringa method. Scripti PHP në pjesën e serverit do të dërgojë të dhënat të marra nga tabela rreth_nesh që ndodhet në databazë. Këto të dhëna do të merren dhe do të parsohen në klasën Java Rreth_Nesh.class . Pasi të parsohen dhe të ruhen në stringat përkatëse ato do të ruhen në fushat përkatëse.

Nëse përdoruesi ka një rast urgjent dhe ka nevojë të komunikojë më klinikën, ai mund të shtyp mbi numrin e celularit të klinikës dhe menjëherë do të hapet aktiviteti DIALER me numrin e klinikës të vendosur. Ose mund të shtypi mbi fushën email për informacione të mëtejshme.

Që të hapet aktiviteti i DIALER do të implementojë në listener të fushës TextView kodin e mëposhtëm:

```
switch(v.getId()) {
    case R.id.celular:
```

```
    Intent intent = new Intent(Intent.ACTION_DIAL);
    intent.setData(Uri.parse("tel:" + cel));
    startActivity(intent);
        break;
    case R.id.email:
  Intent emailIntent = new Intent(Intent.ACTION_SENDTO,
Uri.parse("mailto:" + email));
        startActivity(Intent.createChooser(emailIntent, "Chooser
Title"));
        break;}
```

## 5. Aplikacioni i administratorit

Ky është aplikacioni që do t'i ofrohet administatorit të klinikës, i cili me anë të këtij aplikacioni mund të shoh shërbimet që ofron klinika, të fshijë një shërbim që dëshiron, apo të shtojë një shërbim të ri. Nga ana tjetër ai mund të shoh dhe klientët e regjistruar në këtë apliakacion me qëllim rezervimin e vizitave mjekësore. Ajo që është më e dobishme është fakti që administatori të shoh se cilat janë rezervimet për vizita mjekësore dhe prej cilit klient është kryer. Në këtë mënyrë do të ketë më të thjeshtë për të organizuar punën e tij të përditshme.

Klasat java, duke përfshirë këtu dhe aktivitetet, që e bejnë të mundur krijimin dhe mirë-funksionimin e këtij aplikacioni janë:

1. Klasat për loginin dhe shfaqjen e profilit
    - Login
    - MainActivity
2. Klasat për shfaqjen dhe menaxhimin e shërbimeve
    - Sherbimet
    - Sherbimi
    - SherbimAdapter
    - ShtoSherbim
3. Klasat për shfaqjen dhe menaxhimin e rezervimive të kryera
    - FutureCheck
    - FutureCheckAdapter
    - FutureChecks
4. Klasat për shfaqjen e klientëve
    - Klientet
    - Klienti
    - KlientAdapter
5. Klasa për menaxhimin e komunikimit me serverin dhe databazën

- BackgroundTask

## 5.1 Klasat për identifikimin dhe shfaqjen e profilit

Klasa përgjegjëse për identifikim e përdoruesit, që është dhe një aktivitet, është Login.class. Ky është aktiviteti që do t'i shfaqet administatorit kur do të dëshirojë të përdori aplikacionin dhe do të nevojitet identifikimi i tij. Në ndërfaqe do t'i shfaqen fushat username dhe passëord dhe një buton për të kryer loginin. Të dhënat do të merren prej ketyre dy fushave dhe do t'i dërgohen serverit duke përdorur klasën ndihmëse për komunikimin me serverin BackgroundTask. Serveri do të kryjë identifikim e administratorit duke kërkuar në tabelën e quajtur admin në databazë. Nëse identifikimi do të jetë i suksesshëm, serveri do t'i nisë aplikacioni të dhënat që ndodhen në tabelën rreth_nesh, të cilat do të shfaqen në profiling e administatorit, të cilat dhe mund ti modifikojë.

Aktiviteti kryesor që do të krijohet në castin që administrator do të shtypë mbi ikonën e aplikacionit është MainActivity, ose sic i paraqitet personit si aktiviteti Rreth_Nesh. Në castin që krijohet ky aktivitet do të kontrollohet nëse administatori është identifikuar më përparë. Nëse kjo gjë ska ndodh atëherë do të nis aktiviteti Login.

## 5.2 Klasa BackgroundTask

Ashtu si edhe në aplikacionin e përdoruesit, edhe në aplikacionin e administatorit kam krijuar një klasë për të patur më të lehtë komunikimin me serverin. Kjo klasë është titulluar BackgroundTask, pasi pjesa më e rëndësishme e saj është të kryejë komunikimin, cka do të bëhet nga një thread në background. Ashtu sic e shpjegova dhe më përparë, thirrjet e rrjetit nuk lejohen të bëhen në thread-in kryesor, pasi mund të jenë të gjata dhe të bllokojnë thread-in. Në ndihmë më vjen klasa AsyncTask, të cilën klasa BackgroundTask do ta trashëgojë sëbashku me katër metodat e saj:

- onPreExecute()
- doInBackground()
- onProgressUpdate()
- onPostExecute()

Shumë e rendësishme është metoda doInBackground(), të cilën do ta mbishkruajnë për të menaxhuar komunikin për secilin rast. Të dhënat do të kalohen nëpërmjet metodës .execute() dhe parametrat hyrës do të jenë të tipit string, ashtu si edhe ato të daljes të onPostExecute().

Parametri i parë që do t'i kalojë kësaj klase do të jetë gjithmonë stringa method, e cila do të përmbajë qëllimin. Në klasën BackgroundTask do të merret ky parametër i tipit string dhe do të krahasohet me rastet përkatëse. Më pas klasa do të krijojë instancën e klasës URL për të caktuar URL ku do të dërgohen të dhënat, të cilat do të ruhen në një BufferedWriter dhe do të nisen me metodën flush(). Pas dërgimit do të pritet dhe përgjigjia e serverit e cila do të ruhet me anë të BufferedReader. Kjo stringë më pas do t'i kalohet metodës onPostExecute() e cila do të caktojë se kujt aktiviteti do t'i kalohen këto të dhëna.

Të dhënat do t'i dërgohen serverit në formatin e metodës POST, kurse server do t'i dërgojë sipas formatit JSON, i cili do të parsohet në aktivitetet përkatëse.

### 5.3 Klasat për menaxhimin e shërbimeve, rezervimeve dhe klientëve.

Për të bërë të mundur shfaqjen dhe menaxhimin e shërbimeve, rezervimeve dhe klientëve do të përdoren klasat e mëposhtme, të cilat janë të ngjashme me ato të përdorura në aplikacionin e përdoruesit të thjeshte:

Klasat për shërbimet:

- Sherbimet
- Sherbimi
- SherbimAdapter
- ShtoSherbi

Klasat për rezervimet:

- FutureCheck
- FutureChecks
- FutureCheckAdapter

Klasat për klientët

- Klienti
- Klientet
- KlientAdapter

#### 5.3.1 Klasa Sherbim, Klienti dhe FutureCheck

Të dhënat që do të vijnë mbi të tre llojet, do të kenë nevojë të ruhen në objekte në mënyrë që të shfaqen me vonë në mënyrë të rregullt. Prandaj kam krijuar këto tre klasa. Secila prej tyra ka konstruktorin e vet, i cili cakton vlerat fushave përkatëse, të cilat janë deklaruar si private. Për

të aksesuar të dhënat që ato ruajnë kam krijuar metodat e llojit get() që kthejnë stringat për secilën fushë.

Klasa Sherbimi do të përmbaje fushat: shërbimi, lloji, përshkrimi, id, mjeku dhe cmimi. Klasa Klienti do të përmbajë fushat: emri, mbiemri, mosha, gjinia, id. Klasa FutureCheck do të përmbajë fushat: id, shërbimi, mjeku, data, ora, muaji, viti, emri, mbiemri,kontroll, cel dhe email.

Fusha kontroll do të nevojitet për të kontrolluar nëse rezervimi është ende aktiv, është anulluar apo është kryer. Numri i celularit dhe adresa e email e klientëve do të shfaqet vetëm në aktivitetin e rezervimeve, pasi janë të dhëna delikate dhe nuk duhet që administrator t'i shoh edhe nëse klienti nuk ka kryer asnjë rezervim.

## 6. Konkluzione

Shpërhapja dhe evoluimi i përdorimit të smartphone-ave ka bërë të mundur një transformim në jetën e përditshme të njerëzve. Por ky ndryshim nuk ka patur efekt mbi mënyrën e rezervimit të vizitave mjekësore. Problematika si rradhët e gjata dhe keqinformime rreth orareve të doktorëve janë kthyer në normalitet. Kto probleme janë të zgjidhshme me anë të teknologjisë të platoformës Android në formën e një aplikacioni të aksesueshëm kurdo dhe kudo. Pikërisht në këtë mënyrë i kam zgjidhur problemet e parashtruara më lart.

Fillimisht kam hulumtuar pjesën teorike të komponentëve të një aplikacioni. Kam analizuar funksionin e secilit prej tyre dhe mënyrën e ndërveprimit me njëri-tjetrin si dhe mënyrën e ndërveprimit me vetë sistemin operativ Android. Gjithashtu kam shfaqur IDE (Integrated Development Environment) me të cilin kam punuar së bashku me strukturën e ndjekur të projektit.

Më pas, kam eksploruar mjetet e ofruara dhe mënyra e përdorimit të tyre duke shfrytëzuar gjuhën XML. Duke shfrytëzuar njohuritë e marra fillova të krijoj aplikacionin e përdoruesve, në këtë rast të pacienteve apo klientëve, më specifikisht me ideimin e ndërfaqjeve të aktiviteve me të cilat do ndeshet kushdo ta përdori.

Për të vazhduar më tej, kam krijuar llogjikën e rregjistrimit dhe log-imit nëpërmjet klasave Register dhe Login. Ato kujdesen që përdoruesi të mos lerë asnjë fushë bosh dhe gjithashtu për interaktivitetin në momentin që shtypet butoni përkatës.

Para se të trajtoja komunikimin me databazën kam trajtuar konceptin e threading se bashku me klasën AsyncTask. Koncepte të cila ndihmojnë në optimizimin e burimeve të aparatit ku aplikacioni po ekzekutohet. Me pas zhvillova klasën BackgroundTask që kryen komunikimin me databazën dhe kam trajtuar mënyrën sesi ky shkëbim ndodh së bashku me strukturën e të dhënave e përdorur.

Klasa kryesore MainActivity është ajo që i shfaqet përdoruesit kur aplikacioni hapet. Nëqoftëse dikush është log-uar atëherë do shfaqen informacionet personale të tij. Në të kundërt përdoruesi do të ridrejtohet në fazën e log-imit apo rregjstrimit. Nga ana tjetër klasa DataMan kujdeset për të percjellë informacionet nga databaza në aplikacion në formatin JSON.

Më tej, kam krijuar grupet e klasave për menaxhimin e shërbimeve dhe vizitave e rezervuara. Klasat Shërbimi dhe FutureCheck krijojnë objekte të cilat popullohen me të dhëna nga databaza dhe së bashku me ShërbimAdapter dhe FutureCheckAdapter i përshtasin ato në një ListView duke mundësuar shfaqjen e më shumë informacioneve nga sa mund të shfaqë madhësia e ekranit. Në fund klasat Sherbime dhe FutureChecks kujdesen për llogjikën e rezervimeve duke përdorur Id-në e përdoruesit dhe veprimeve paraprake rezervuese kurse klasa Rezervim është përgjegjëse për caktimin e muajit, datës dhe orarit të rezervimit.

Kam krijuar gjithashtu edhe klasën Modifikime e cila mundëson përditësimin e të gjithë dhënave. Ky funksion targeton më së shumti ndryshimin e fjalëkalimit duke rritur sigurinë e llogarisë. Ndërsa klasa Rreth_Nesh shfaq informacione rreth klinikës që janë të modifikueshme nga vetë ajo nëpërmjet aplikacionit të administratorit.

Si pjesë plotësuese e aplikacionit të përdoruesve kam krijuar edhe aplikacionin e administatorit. Pjesa e log-imit është shumë e ngjashme me aplikacionin e parë. Po ashtu komunikimi me databazën ndodh nëpërmjet klasës BackgroundTask duke mënjanuar shpërdorimin e burimeve të thread-it kryesor. Gjithashtu klasat për menaxhimin e shërbimeve, rezervimeve dhe klienteve edhe pse janë të ngjashme ato kryejnë funksione të ndryshme. Kjo arrihet me lidhjen e identifikuesve të pacintëve bashkë me rezervimet e kryera prej tyre. Me pas këto rezervime lidhen me doktorët që zhvillojnë shërbimin përkatës. Domethënë arrihet lidhja indirekte e pacientëve me doktorët.

Në përfundim mund të them se kam finalizuar mundin dhe përpjekjet e ideve të mia. Kam krijuar aplikacionet për të dyja palët si për pacienët ashtu edhe për klinikat. Kto dy aplikacione ofrojne lidhjen ndërmjet tyre duke shfrytëzuar mjetet dhe teknologjinë që ofron platforma Android.

## 7. Pasqyra e figurave

## 8. Referencat


[1] - https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (2016)
[2] - https://support.google.com/googleplay/android-developer/answer/113469?hl=en (2018)
[3] - http://www.openhandsetalliance.com/press_110507.html (2007)
[4] - https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/ (2018)
[5] - https://developers.google.com/training/ (2018)
[6] - Android Studio Development Essentials - Android 7 Edition (2017)
[7] - https://developer.android.com/index.html (2018)
[8] - "Thinking in Java (4th Edition)" - Bruce Eckel (2006)
[9] - "Efficient Android Threading" - Anders Goransson (2014)
[10] - "Android Programming: Pushing the Limits" - Erik Hellman (2013)
[11] - "The Busy Coder's Guide to Advanced Android Development" - Mark Murphy (2009)
[12] - "Java: A Beginner's Guide" - Herbert Schildt (2014)