MAK mobilna aplikacija

Peršić, Matej

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

MAK MOBILNA APLIKACIJA

Završni rad

Matej Peršić

Osijek, 2021.

SADRŽAJ

1. UVOD

U današnjem vremenu pametnih uređaja, postoji potreba za aplikacije koje su prvenstveno za njih i kreirane. Premda platforme rade s istim podatcima, za svaku od njih treba izraditi posebno programsko rješenje. U ovom radu bit će opisani procedura i postupak izrade programskog rješenja za mobilnu aplikaciju sustava MAK na FERIT-u.

Prvo poglavlje sadrži opis zadatka ovog završnog rada.

U drugom poglavlju dan je pregled područja. U njemu je opisan razvoj kompleksnosti mobilnih aplikacija pojavom pametnih telefona, kratak opis sustava Android i navedene su tehnologije korištene za rješavanje problematike u radu.

U trećem poglavlju dan je pregled sustava MAK. Opisana je njegova svrha i dana je kratka podjela na pojedine uloge unutar sustava koje su korištene u radu.

U četvrtom poglavlju opisan je programski jezik Kotlin. Navedeni su osnovni tipovi i operatori koji se u njemu koriste. Prikazani su primjeri funkcija s povratnim tipom i bez njega. Opisane su osnovne kolekcije.

U petom poglavlju proučena je problematika prijave i odjave korisnika u sustav, kao i određivanje koje aktivnosti korisnik vidi s obzirom na njegovu ulogu. Također, prikazano je kako pratiti aktivnog korisnika unutar sustava preko generiranog tokena i njegovog korisničkog imena. Pojašnjeno je programsko rješenje i pokazani su primjeri ulaza i izlaza za pojedinu ulogu u sustavu.

Za uspješno programsko rješenje potrebno je povezati znanje Kotlina i HTTP zahtjeva. Uz pomoć njih i ostalih navedenih tehnologija, treba doći do valjanog programskog rješenja..

1.1. Zadatak završnog rada

Zadatak završnog rada je napraviti mobilnu verziju postojeće Web aplikacije za mentoriranje diplomskih i završnih radova MAK. Studentima treba omogućiti dodavanje i nadzor svojih radova, a mentorima vođenje svojih studenata.

2. PREGLED PODRUČJA

Pojava prvih mobilnih uređaja vuče korijene iz 1908. godine, kada je u Kentuckyju napravljen patent za prvi bežični telefon. Prvi mobilni uređaji zapravo ni nisu bili mobilni uređaji, nego dvosmjerni radio koji je omogućio komunikaciju ljudima poput vozača taksija i hitnih službi. [1]

Kako je tehnologija brzo napredovala, tako su i mobilni uređaji postali obvezan alat koji većina ljudi već godinama svakodnevno nosi u svom džepu. Uz manje mobilne uređaje popraćene sve sofisticiranijom tehnologijom, počele su se pojavljivati i nove funkcije na njima, od zapisa bilješki, pa i novih igara, kao što je legendarna Zmija na Nokija telefonima. [2]

Pojavom prvog iPhone-a, došlo je do značajne revolucije u tržištu mobilnih uređaja. Do tog trenutka, mobiteli su bili korišteni uglavnom za jednostavnije zadatke poput slanja poruka i poziva, no danas su jedina granica korisnosti mobilnog uređaja aplikacije koje korisnik na njemu ima instalirane. S novom generacijom pametnih mobitela, počele su se pojavljivati i moćnije, složenije aplikacije koje olakšavaju korisnicima prethodno zahtjevnije zadatke. Nikad zahtjevnije, a istovremeno nikada zasićenije tržište mobilnih aplikacija, iz dana u dan sve više napreduje. Uz dva tehnološka diva koji dominiraju tržištem, Appleom i Googleom i njihovim brojnim korisnicima, velika je potreba za kvalitetnim aplikacijama koje se daju što lakše i brže izraditi. [3]

Alat koji se koristi za Appleov OS zove se Xcode, a programski jezici i danas popularne tehnologije za razvoj u njima UI-Kit i noviji, Swift-UI. Usprkos velikim cijenama njihovih uređaja i relativno malom postotku iPhone-a na tržištu, oni ostvaruju gotovo trećinu dobiti iz cijele mobilne industrije. [4]

Za Google-ovu platformu Android za koju je napravljena aplikacija u ovom radu, alat koji se koristi je Android Studio, a programski jezici u kojima se radi razvoj aplikacije su Java i Kotlin, dok se softver Flutter koji koristi za hibridni razvoj aplikacija. [5]

Za programska rješenja slična ovom mogu se promotriti mobilna aplikacija za sustav Merlin i mobilna aplikacija Studentski Raspored FERIT-a. Obje su povezane sa sustavom AAI identiteta, te kao takve prilikom uspješne autentifikacije moraju pružiti korisniku odgovarajuće sučelje, te pristup i rad s podatcima koji su njemu jedinstveni. Također, gotovo svaka popularnija aplikacija koja ima i web i mobilnu platformu funkcionira na sličan način, gdje se radi s istim podatcima i istom bazom podataka, koji se korisniku trebaju prikazati na način koji mu je poznat iz web rješenja, i obrnuto.

2.1. Android

Android je operacijski sustav za mobilne uređaje poput pametnih telefona, tablet računala, pa čak i za pametne satove i automobile. Prva verzija Androida lansirana je 2008. godine, a njegova najnovija verzija je Android 12. Danas je najrašireniji operacijski sustav zbog toga što je besplatan i pristupačan, pa je prisutan u uređajima svih cjenovnih razreda. [6]

2.2. Korištene tehnologije

Za razvoj programskog rješenja ovog završnog rada korišten je programski jezik Kotlin, koji je popularan zbog lake čitljivosti i lakšeg rada u odnosu na Javu. Uz njega je korišten FileAndFolderPicker za lakši rad s podatcima s korisnikovog uređaja i Retrofit za rad s HTTP zahtjevima. [7]

3. SUSTAV MAK

Na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek, za olakšani rad sa završnim i diplomskim radovima koristi se Informacijski sustav za mentorske aktivnosti – MAK. Unutar njega studenti, mentori, ocjenjivači i ostale uloge unutar sustava mogu odraditi svoje dužnosti vezane uz radove.

3.1. Osnovne informacije

Informacijski sustav za mentorske aktivnosti – MAK, sustav je korišten za lakše provođenje postupaka vezanih uz pisanje i ocjenjivanje završnih i diplomskih radova na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. On pokriva dio procedure od trenutka kada Odbor odobri dodjelu teme ili mentora studentu do trenutka kada predsjednik Povjerenstva za obranu, odnosno tajnik Odbora za završne i diplomske ispite potvrdi prolaznu ocjenu.

3.2. Uloge unutar sustava

Unutar sustava MAK postoje brojne uloge. Prema ulogama, korisnik vidi određeni dio aplikacije. Uloge koje su korištene u ovom završnom radu su student, mentor i ocjenjivač.

Student je osoba koja unutar sustava dodaje svoj završni rad i može pratiti stanje procedure u kojem se njegov završni rad nalazi.

Mentor je osoba unutar sustava koja može nadgledati svoje studente i potvrditi verzije studentovih radova. Ako je riječ o studentu s diplomskog ili stručnog studija, mentor unutar sustava može odrediti povjerenstvo za obranu.

Ocjenjivač je osoba unutar sustava koja dobiva rad studenta na čitanje i daje povratnu informaciju o radu.

4. PROGRAMSKI JEZIK KOTLIN

Kotlin je programski jezik koji je Google u 2019. godini predstavio kao preferirani programski jezik za Android aplikacije. On je moderan programski jezik koji je popularan zbog lakšeg čitanja i pisanja u odnosu na Javu.

4.1. Tipovi i operatori

U Kotlinu se mogu koristiti sljedeće tipove za:

1. Cijele brojeve
    - Byte
    - Short
    - Int
    - Long
2. Realne brojeve
    - Double
    - Float
3. Booleove vrijednosti
    - Boolean
4. Znakove i niz znakova
    - Char
    - String
5. Polja
    - Array

Uz njih, mogu se koristiti i mnoge ostale, složenije kolekcije podataka. [8]

Kotlin podržava sljedeće operatore:

- + operator zbrajanja
- - operator oduzimanja

- * operator množenja
- / operator dijeljenja
- % operator ostatka cjelobrojnog dijeljenja
- == operator jednakosti
- != operator različitosti
- = operator pridjeljivanja

I ostale relacijske i operacijske operatore.

4.2. Funkcije

Kao i u ostalim programskim jezicima, u Kotlinu funkcija može i ne mora vraćati neku vrijednost.

Na slici 4.1 nalazi se primjer funkcije koja ispisuje Hello World.

```
fun printHello(){
    println("Hello world!")
}
```

Sl. 4.1. Funkcija koja ispisuje Hello World

Na slici 4.2. se može vidjeti primjer funkcije koji vraća String Hello.

```
fun uppercaseHello() : String {
    return "Hello".uppercase()
}
```

Sl. 4.2. Funkcija koja vraća Hello u velikim slovima

4.3. Kolekcije

U Kotlinu postoje brojne kolekcije, a njihova najopćenitija podjela je na liste, setove i mape.

Lista pohranjuje elemente u specifičnom redu, gdje je prvi element indeksa 0, a zadnji element ima indeks jednak broju koji predstavlja veličinu liste umanjenom za 1.

Set služi za pohranu različitih, jedinstvenih elemenata, čiji je red generalno nedefiniran.

Mapa služi za pohranjivanje parova ključ-vrijednosti, gdje su ključevi jedinstveni. [9]

4.4. Korištene biblioteke

Za rješavanje problema rada s HTTP zahtjevima korištena je Retrofit biblioteka. Za rješavanje problema učitavanja datoteka s korisnikovog uređaja je korišten FileAndFolderPicker. [10]

5. PROGRAMSKO RJEŠENJE

Pomoću postojeće forme za sustav MAK i uspješne prijave, treba se odrediti je li riječ o studentu ili ostalim korisnicima, kako bi se korisniku prikazalo odgovarajuće sučelje u kojem može odraditi svoje odgovornosti unutar sustava.

5.1. Zamisao rješenja

Program treba riješiti na način da se s postojeće forme za prijavu korištenjem AAI identiteta, nakon uspješne prijave odredi je li riječ o studentu ili ostalim korisnicima. Ako je riječ o studentu, prikazati mu dio aplikacije unutar kojih može raditi i nadgledati proceduru svog rada, a ako nije, korisniku prikazati dio vezan za mentorske i ocjenjivačke aktivnosti. Kroz kretanje između aktivnosti, treba pratiti aktualnog korisnika preko njegovog generiranog tokena i korisničkog imena. Kada je korisnik gotov s radom, omogućiti mu odjavu iz sustava.

5.2. Postupak rješavanja

Prvi problem koji treba riješiti je učitavanje tokena i korisničkog imena u *MainActivityju*. Za to je korišten *Javascript interface* koji se može vidjeti na slici 5.1. Pomoću njega se čita i sprema tekst s web stranice nakon uspješne autentifikacije korisnika i nakon toga se poziva metoda *startNewActivity*.

```
class MyJavaScriptInterface {
    val browser = webView
    var output: ArrayList<String> = arrayListOf<String>()
    var outputString = ""

    @JavascriptInterface
    fun processHTML(html: String) {
        // process the html as needed by the
        for (i in html.indices)
            output.add(html[i].toString())
        for (s in output)
            outputString += s
        outputString = Html.fromHtml(outputString).toString()
        startNewActivity(outputString)
    }
}
```

Sl. 5.1. Javascript Interface

Kako bi se napisano sučelje povezalo s WebViewom unutar *MainActivityja*, koristi se kod za povezivanje *Javascript Interfacea* s WebViewom na slici 5.2.

Nakon uspješnog preusmjeravanja, poziva se metoda *startNewActivity*. Kako bi se raspoznao student od ostalih korisnika u sustavu, API od kojeg se dobije String na web stranici oblika token/username sadrži nastavak /student. Ako String učitan s web stranice sadrži /student, pokreće

```
webView.settings.javaScriptEnabled = true
webView.addJavascriptInterface(MyJavaScriptInterface(), "HTMLOUT")

webView.webViewClient = object : WebViewClient() {
    override fun onPageFinished(view: WebView?, url: String?) {
        //successfull redirect
        if (webView.url == "https://www.ferit.unios.hr/mak/api/v1/api.php") {

webView.loadUrl("javascript:HTMLOUT.processHTML(document.documentElement.outerHTML);
")
        }
    }
}
webView.loadUrl("https://www.ferit.unios.hr/new-
login?r=https://www.ferit.unios.hr/mak/api/v1/api.php")
```

Sl. 5.2. Kôd za povezivanje Javascript Interfacea s WebViewom

```
fun startNewActivity(tokenUserName: String) {
    if (tokenUserName.contains("/student")){
        val parameter=tokenUserName.substringBefore("/student")
        val intent = Intent(this, StudentsMainActivity::class.java)
        intent.putExtra("tokenUserName", parameter)
        startActivity(intent)
    }
    else {
        val intent = Intent(this, OtherUserMainActivity::class.java)
        intent.putExtra("tokenUserName", tokenUserName)
        startActivity(intent)
    }
}
```

Sl. 5.3. Metoda startNewActivity u MainActivityju

se studentska aktivnost, inače se pokreće aktivnost za ostale korisnike. Kod za metodu *startNewActivity* se nalazi na slici 5.3.

Kako bi se u ostalim aktivnostima korisnicima omogućilo odjavljivanje vraćanjem na *MainActivity*, na njegov početak dodan je parametar s kojim se provjerava je li on postavljen iz prethodne aktivnosti za brisanje kolačića iz WebViewa, čijim se uklanjanjem omogućuje ponovna prijava korisnika, točnije vraćanje aplikacije u početno stanje. Kod za odjavu se nalazi na slici 5.4.

```
val remove = intent.getStringExtra("logOut")
if(remove=="true"){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        cookieManager.removeAllCookies { aBoolean ->
            // a callback which is executed when the cookies have been removed
            Log.d("TAG", "Cookie removed: $aBoolean")
        }
    } else cookieManager.removeAllCookie()
}
```

Sl. 5.4. Kôd za odjavu korisnika

5.3. Detaljan pregled aktivnosti

5.3.1. Studentski dio

U studentskoj glavnoj aktivnosti *StudentsMainActivity* prije svega se dohvaća String poslan iz metode s*tartNewActivity* iz *Main Activityja*. On se dijeli na dio student koji predstavlja korisničko ime studenta i token, koji se skupa koriste za okidanje različitih HTTP zahtjeva. To se ostvaruje kodom na slici 5.5. i isto se radi i u aktivnostima za ostale korisnike u sustavu.

```
var userName:String=intent.getStringExtra("tokenUserName").toString()
token= userName.substringBefore("/")
student = userName.substringAfter("/")
```

Sl. 5.5. Dohvaćanje i obrada korisničkog imena studenta i njegovog tokena u StudentsMainActivity

Nakon toga poziva se metoda *initializeViews* u kojoj se inicijaliziraju *viewPager* i *tabLayout* koji se koriste za prikazivanje fragmenata unutar glavne studentske aktivnosti. Navedeno se može vidjeti na slici 5.6. Nakon metode *initializeViews* u kodu slijedi Retrofit metoda s kojom se dohvaćaju informacije o studentu. Dohvaćanje podataka o studentu se može vidjeti na slici 5.7. Ako je GET zahtjev uspješan, poziva se metoda *setUpPager*, inače student dobiva odgovarajuću Tost (engl Toast) poruku o grešci. Izlistanje metode *setUpPager* nalazi se na slici 55.8.

```
private fun initializeViews() {
    tablayout=findViewById(R.id.studentTabLayout)
    viewPager=findViewById(R.id.studentViewPager)
}
```

Sl. 5.6. Metoda initializeViews studentove glavne aktivnosti

```
val makApi = ServiceGenerator.createGetStudentInfoApi()
val call = makApi.getStudentInfo(token,student)
 call.enqueue(object: Callback<GetStudentInfo>{
     override fun onResponse(call: Call<GetStudentInfo>, response:
Response<GetStudentInfo>) {
         for(post in response.body()!!.data){
            setUpPager(post)
         }
     }

     override fun onFailure(call: Call<GetStudentInfo>, t: Throwable) {
         Toast.makeText(this@StudentsMainActivity,"ERROR: ${t.message.toString()}",
Toast.LENGTH_LONG).show()
     }

 })
```

Sl. 5.7. Dohvaćanje podataka o studentu

Kao adapter za *viewPager* koristi se *StudentPagerAdapter* kojem se kroz konstruktor daje broj kartica, token, korisničko ime i objekt klase *StudentInfo*, koji se može vidjeti na slici 5.9. On omogućuje dobivanje različitih fragmenata za različite indekse.

```
private fun setUpPager(data: StudentInfo) {
    tablayout.addTab(tablayout.newTab().setText(getString(R.string.AboutPaper)))
    tablayout.addTab(tablayout.newTab().setText(getString(R.string.FinalVersion)))
    tablayout.addTab(tablayout.newTab().setText(getString(R.string.LogOut)))
    val pagerAdapter=StudentPagerAdapter(supportFragmentManager,data,3, token,
student)
    viewPager.adapter=pagerAdapter

viewPager.addOnPageChangeListener(TabLayout.TabLayoutOnPageChangeListener(tablayout)
)
    tablayout.addOnTabSelectedListener(object: TabLayout.OnTabSelectedListener{
        override fun onTabSelected(tab: TabLayout.Tab?) {
            viewPager.currentItem=tab!!.position
        }

        override fun onTabUnselected(tab: TabLayout.Tab?) {
        }

        override fun onTabReselected(tab: TabLayout.Tab?) {
        }

    })
}
```

Sl. 5.8. Metoda setUpPager

```
class StudentPagerAdapter(fm: FragmentManager, data: StudentInfo,
                          var totalTabs: Int,
                          var token: String, var userName: String):
FragmentPagerAdapter(fm) {
    private var studentInfo = data
    override fun getCount(): Int {
        return totalTabs
    }

    override fun getItem(position: Int): Fragment {
        return when(position){
            0 -> {
                return StudentFinalPaperInfo(studentInfo,token,userName)
            }
            1 -> {
                return
StudentConfirmsFinalPaper(studentInfo.jmbag,studentInfo.status)
            }
            else -> return StudentLogOutFragment()
        }
    }
}
```

Sl. 5.9. StudentPagerAdapter

U fragmentu *StudentFinalPaperInfo* student može vidjeti podatke o svom radu kao što su status, id, ime rada i slično. Također može staviti prvu i konačnu verziju svoga rada, što su gotovo identične procedure. Na slici 5.10. može se vidjeti metoda *onViewCreated* u kojoj se vrši inicijalizacija TextViewova i tipki (engl. buttona), postavljanje njihovih vrijednosti i *onClickListenera* na tipku s kojom student može učitati početnu i konačnu verziju svog rada.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    initializeViews(view)
    setUpValues()
}
```

Sl. 5.10. onViewCreated fragmenta StudentFinalPaperInfo

Kako bi student mogao prijeći u novu aktivnost u kojoj okida POST zahtjev na kojem šalje podatke o svom radu, prvo se provjeravaju prava za pristup čitanju i pisanju na uređaj. Dok aplikacija nema prava za pristup pisanju i čitanju, klikom na tipku će ih tražiti od korisnika. Ako je status procedure rada valjan i prava su odobrena, pokrenut će se nova aktivnost koja će studentu omogućiti učitavanje početne ili konačne verzije u sustav. Budući da je procedura i logika za njih ista, obje neće biti prikazane u radu. Kod vezan uz navedeno se nalazi na slici 5.11.

```
private fun setUpValues() {
        id.text = getString(R.string.ID) + ": " + data.id
        studij.text = getString(R.string.Study) + ": " + data.naziv_studija
        ime.text = getString(R.string.Name) + ": " + data.ime
        prezime.text = getString(R.string.Surname) + ": " + data.prezime
        jmbag.text = getString(R.string.JMBAG) + ": " + data.jmbag
        naslovTeme.text = getString(R.string.Theme) + ": " + data.naziv_rada
        datumDodjele.text = getString(R.string.Assigned) + ": " + data.datum_dodjele
        mentor.text = getString(R.string.Mentor) + ": " + data.mentorime
        sumentor.text = getString(R.string.Comentor) + ": " + data.sumentor
        rokPredaje.text = getString(R.string.Deadline) + ": " + data.rok_za_predaju
        statusProcedure.text = getString(R.string.ProcedureStatus) + ": " +
data.status

    submitPaper.setOnClickListener {
        val formattedProcedure =
statusProcedure.text.toString().substringAfter(getString(R.string.ProcedureStatus) +
": ")
        if (permissionGranted()) {
            if (formattedProcedure == "1/9" || formattedProcedure == "0/9") {
                startFirstVersionActivity()
            } else if (formattedProcedure == "2/9" || formattedProcedure == "3/9") {
                startFinalVersionActivity()
            } else {
                Toast.makeText(context, "Invalid procedure status",
Toast.LENGTH_LONG).show()
            }
        }
        else {
            requestPermission()
        }
    }
}
```

Sl. 5.11. Metoda setUpValues fragmenta i postavljanje onClickListenera na tipku u StudentFinalPaperInfo

Kada su prava i status procedure valjani, u novoj aktivnosti student će moći odabrati početnu verziju svoga rada u sučelju kao na slici 5.12. Kada ju odabere i pritisne kvačicu u desnom kutu uređaja, okinut će se POST zahtjev i student će dobiti Tost poruku o uspješnosti zahtjeva. Nakon toga studentu je prikazana tipka s kojom se može vratiti na prethodnu aktivnost ili to može

napraviti pomoću tipke za natrag. Kod vezan uz odabiranje datoteke i POST zahtjev se može vidjeti na slici 5.13.

Sl. 5.12. Odabiranje datoteke s uređaja pomoću FileAndFolderPickera

```
val singleFilePickerDialog = SingleFilePickerDialog(this, {
    Toast.makeText(this, "Picking canceled!", Toast.LENGTH_SHORT).show()
}) { files: Array<File> ->
    myFileURI = files[0].path
    val myFile = File(myFileURI)
    val requestFile = RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
myFile)
    val body = MultipartBody.Part.createFormData("fileToUpload", myFile.name,
requestFile)
    val fileName = RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
myFile.name)
    val call = makApi.pushInitialPaper(fileName, body, token, userName, idRada)
    call.enqueue(object : Callback<Any> {
        override fun onResponse(call: Call<Any>, response: Response<Any>) {
            Toast.makeText(applicationContext, response.body().toString(),
Toast.LENGTH_LONG).show()
            Toast.makeText(applicationContext, "Paper submitted successfully!",
Toast.LENGTH_LONG).show()
        }

        override fun onFailure(call: Call<Any>, t: Throwable) {
            Toast.makeText(applicationContext, "Something went wrong! Error:
${t.message.toString()}", Toast.LENGTH_LONG).show()
        }
    })
}
singleFilePickerDialog.show()
```

Sl. 5.13. Spremanje početne verzije studentovog rada

Drugi fragment koji student može koristiti u glavnoj aktivnosti je *StudentConfirmsFinalPaper*. U njemu se nalazi jedna tipka s kojom student može potvrditi konačnu verziju svoga rada, ako je njegov status 4/9. Na njen klik dobiva povratnu informaciju o uspješnosti zahtjeva ili je upozoren da mu status procedure nije valjan. Na slici 5.14. može se vidjeti kod koji se nalazi unutar metode *onViewCreated* navedenog fragmenta. U njemu se postavlja *onClickListener* na tipku, a može se i vidjeti koji se zahtjev okida.

```
button.setOnClickListener {
    if(status == "4/9"){
        val makApi = ServiceGenerator.createStudentConfirmsFinal()
        val call = makApi.studentConfirmsPaper(jmbag)
        call.enqueue(object: Callback<Any> {
            override fun onResponse(call: Call<Any>, response: Response<Any>) {
                Toast.makeText(context,"Procedure success!",
Toast.LENGTH_LONG).show()
            }
            override fun onFailure(call: Call<Any>, t: Throwable) {
                Toast.makeText(context,"Procedure fail: ${t.message.toString()}",
Toast.LENGTH_LONG).show()
            }

        })
    }
    else {
        Toast.makeText(context,"Invalid procedure status", Toast.LENGTH_LONG).show()
    }
}
```

Sl. 5.14. Postavljanje onClickListenera na tipku unutar fragmenta StudentConfirmsFinalPaper

Posljednji fragment koji student može koristiti zove se *StudentLogOutFragment*. Unutar njega se nalazi tipka na čiji se klik korisnik može odjaviti, točnije poslati String u *MainActivity* i tako vratiti aplikaciju u početno stanje. Navedeno je odrađeno kodom na slici 5.15.

```
logOutButton.setOnClickListener {
    val intent = Intent(context, MainActivity::class.java)
    //clearing cookies, logging out
    intent.putExtra("logOut","true")
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
    startActivity(intent)
}
```

Sl. 5.15. OnClickListener na tipku s kojom se student može odjaviti u StudentLogOutFragmentu

5.3.2. Mentorski dio

Ostali korisnici nakon uspješnog logiranja mogu vidjeti tri tipke u *OtherUserMainActivityju*, gdje svaka od njih vodi korisnika u drugom smjeru. U *onCreate* metodi prvo se dohvaća string userName koji je poslan u *MainActivityju*. Klikom na tipku mentor, okida se GET zahtjev u kojem se dohvaćaju mentorovi studenti, a nakon dohvaćanja tih podataka mentor može vidjeti sve svoje studente i raditi s njihovim radom. Klikom na tipku ocjenjivač, okida se GET zahtjev u kojem se dohvaćaju ocjenjivačevi studenti, nakon čijeg dohvaćanja ocjenjivač može raditi s radovima

studenata koje on ocjenjuje. Implementacija *onClickListenera* za navedene tipke može se vidjeti na slici 5.16. Uz njih se nalazi tipka za odjavu, čije je implementacija identična onoj na slici 5.15.

```
mentorActivityButton.setOnClickListener {
    val intent = Intent(this, MentorsRecyclerActivity::class.java)
    intent.putExtra("tokenUserName", userName)
    startActivity(intent)
}
ocjenjivacActivityButton.setOnClickListener {
    val intent = Intent(this, OcjenjivacActivity::class.java)
    intent.putExtra("tokenUserName", userName)
    startActivity(intent)
}
```

Sl. 5.16. onClickListeneri za tipke u prijelaz Mentorove i Ocjenjivačeve aktivnosti

Prva aktivnost koju mentor može vidjeti je *MentorsRecyclerActivity*. Unutar nje se dohvaćaju i prikazuju mentorovi studenti, koji se šalju u *MentorsStudentRecyclerAdapter* uz njegov token i korisničko ime, što može se vidjeti na izlistanju koda na slici 5.17.

```
val makApi = ServiceGenerator.createGetStudentsApi()
val call = makApi.getStudents(token,user)
call.enqueue(object: Callback<StudentsForMentorActivityRoot> {
    override fun onResponse(call: Call<StudentsForMentorActivityRoot>, response:
Response<StudentsForMentorActivityRoot>) {
        students= response.body()!!.data as ArrayList<StudentForMentorActivity>
        Log.d("body", response.body().toString())
        recyclerView.adapter= MentorsStudentsRecyclerAdapter(students,token,user)
        if (students.isEmpty()){
            Toast.makeText(applicationContext,"No students found",
Toast.LENGTH_LONG).show()
        }
    }

    override fun onFailure(call: Call<StudentsForMentorActivityRoot>, t: Throwable) {
        Toast.makeText(applicationContext,"Error: ${t.message.toString()}",
Toast.LENGTH_LONG).show()
    }
})
```

Sl. 5.17. Postavljanje RecyclerViewa unutar MentorsRecyclerActivityja

Na slikama 5.18. i 5.19. može se vidjeti kod ključan za *MentorsStudentRecyclerAdapter* unutar kojeg se postavlja *onClickListener* na pojedini element u RecyclerViewu, gdje se poziva metoda *choosingActivity* s kojom se pokreće aktivnost *MentorChoosesActivity* koja omogućuje mentoru izbor između potvrđivanja početne i konačne verzije rada studenta, kao i odabir povjerenstva za obranu na diplomskom i stručnom studiju.

```
 inner class ViewHolder(itemView: View):RecyclerView.ViewHolder(itemView){
    var studentName:TextView=itemView.findViewById(R.id.student_cell_name)
    var
studentNazivRada:TextView=itemView.findViewById(R.id.student_cell_zavrsniRad)
    init {
        itemView.setOnClickListener {
            var position = adapterPosition
            var aai = students[position].aai
            var id = students[position].id
            var status = students[position].status
            var prva_verzija = students[position].prva_verzija
            var sumentor = students[position].sumentor
            var mentor = students[position].mentor
            var rad_na_stupu:String = students[position].rad_na_stupu
            var tip_studija = students[position].tip_studija
            choosingActivity(itemView,aai,id,prva_verzija,status!!,sumentor,
rad_na_stupu,  tip_studija, mentor)
        }
    }
```

Sl. 5.18. Postavljanje onClickListenera na pojedinog studenta u RecyclerViewu

```
private fun choosingActivity(itemView: View, aai: String, id: String, prva_verzija:
String,
                             status: String, sumentor:String, rad_na_stupu:String,
                             tip_studija: String, mentor: String) {
    itemView.context.startActivity(Intent(itemView.context,
MentorChoosesActivity::class.java)
        .putExtra("aai",aai)
        .putExtra("id",id)
        .putExtra("prva_verzija",prva_verzija)
        .putExtra("token",token)
        .putExtra("user",user)
        .putExtra("status",status)
        .putExtra("sumentor",sumentor)
        .putExtra("rad_na_stupu", rad_na_stupu)
        .putExtra("tip_studija",tip_studija)
        .putExtra("mentor",mentor)
    )
}
```

Sl. 5.19. Metoda choosingActivity

Unutar aktivnosti *MentorChoosesActivity*, mentor može vidjeti tri tipke koje može koristiti za aktivnosti vezane sa studentovim radom. Inicijalizaciju *onClickListenera* za tipke u kojima mentor potvrđuje početnu i konačnu verziju rada može se vidjeti na slici 5.20.

Na slici 5.21. može se vidjeti inicijalizacija tipke na čiji klik pokreće aktivnost u kojoj mentor može odabrati povjerenstvo za obranu na diplomskom i stručnom studiju. Ako je riječ o studentu na preddiplomskom studiju ili student nema valjano stanje procedure za prijelaz u sljedeću aktivnost, mentor dobiva odgovarajuću Tost poruku.

```
confirmInitialVersion.setOnClickListener {
    if (status == "1/9"){
        startConfirmInitialVersionActivity(id)
    }
    else {
        Toast.makeText(this,"Invalid status, should be 1/9", Toast.LENGTH_LONG).show()
    }
}
confirmFinalVersion.setOnClickListener {
    if (permissionGranted()){
        if (status == "3/9"){
            startConfirmFinalStudentsPaperActivity(aai)
        }
        else {
            Toast.makeText(this,"Invalid status, should be 3/9",
Toast.LENGTH_LONG).show()
        }
    }
    else {
        requestPermission()
    }
}
```

Sl. 5.20. Postavljanje onClickListenera na tipke u aktivnosti MentorChoosesActivity

```
strucniDiplomskiUnosZaObranuBtn.setOnClickListener {
    if (tip_studija == "Preddiplomski"){
        Toast.makeText(this,"Invalid study status", Toast.LENGTH_LONG).show()
    }
    else if (!validStrucniDiplomskiStatus(status)) {
        Toast.makeText(this,"Invalid procedure status", Toast.LENGTH_LONG).show()
    }
    else{
        startDiplomskiStrucniMentorPovjerenstvoActivity(aai, mentor, sumentor, id)
    }
}
```

Sl. 5.21. Postavljanje onClickListenera za postavljanje povjerenstva obrane u MentorChoosesActivityju

```
confirmInitialVersionButton.setOnClickListener {
    val grana = chooseBranchSpinner.selectedItem.toString()
    val requestBodyGrana = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), grana)
    val makApi = ServiceGenerator.createConfirmInitialVersion()
    val call = makApi.confirmInitialVersion(id, user, token,requestBodyGrana)
    call.enqueue(object: Callback<Any> {
        override fun onResponse(call: Call<Any>, response: Response<Any>) {
            Toast.makeText(applicationContext,"Initial version confirmed!",
Toast.LENGTH_LONG).show()
        }

        override fun onFailure(call: Call<Any>, t: Throwable) {
            Toast.makeText(applicationContext, "Error ${t.message.toString()}",
Toast.LENGTH_LONG).show()
        }
    })
}
```

Sl. 5.22. Potvrđivanje početne verzije rada u aktivnosti ConfirmInitialStudentsPaper

Za potvrđivanje početne verzije u aktivnosti *ConfirmInitialStudentsPaper*, mentor treba izabrati granu u kojoj se nalazi rad u padajućem izborniku koji je realiziran pomoću Spinnera i nakon toga pritisnuti tipku submit kojom se okida POST zahtjev za potvrđivanje početne verzije, na koji

mentor dobiva odgovarajuću Tost poruku vezanu uz uspješnost zahtjeva. Kod za *onClickListener* tipke podnesi (engl. Submit) može se vidjeti na slici 5.22.

Za potvrđivanje konačne verzije u aktivnosti *ConfirmFinalStudentsPaper* mentor treba označiti checkBox koji govori je li riječ o potvrdi konačne verzije ili ne, unijeti generiranu Ephorus vrijednost rada, u padajućim izbornicima odabrati vrijednosti za znanje, rezultate, jasnoću i samostalnost studentova rada. Nakon toga treba dati objašnjenje i odabrati studentovu granu. Na kraju mentor pritiskom na tipku Choose a file vidi sučelje kao na slici 5.12. gdje odabire konačnu verziju rada i njenim odabirom, okida POST zahtjev nakon kojeg dobiva odgovarajuću poruku o uspješnosti navedenog zahtjeva. Kod vezan uz tipku za odabir slike, kao i okidanje POST zahtjeva se nalazi na slici 5.23. i . 5.24.

```
chooseFileButton.setOnClickListener {
    id = student.id
    izracunajSatnicu(student)
    val singleFilePickerDialog = SingleFilePickerDialog(this, {
                Toast.makeText(this, "File picking cancelled!",
Toast.LENGTH_SHORT).show()
    }) { files: Array<File> ->
        myFileURI = files[0].path
        var konacna_verzija_potvrde = "false"
        if (konacnaVerzijaCheckBox.isChecked) {
            konacna_verzija_potvrde = "true"
        }
        val grana = chooseBranchFinalSpinner.selectedItem.toString()
        val ephorus = numberPicker.text.toString().toInt()
        znanja = znanjaSpinner.selectedItem.toString()
        rezultati = rezultatiSpinner.selectedItem.toString()
        jasnoca = jasnocaSpinner.selectedItem.toString()
        samostalnost = samostalnostSpinner.selectedItem.toString()
        val obrazlozenje = obrazlozenjeET.text.toString()
        val pismeni_ocjena = getMentorOcjena()
        val myFile = File(myFileURI)
        val requestFile = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), myFile)
        val body = MultipartBody.Part.createFormData("ephorus_izvjestaj",
myFile.name, requestFile)
        val fullName = RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
myFile.name)
        val postMakApi = ServiceGenerator.createConfirmFinalVersion()
        val konacnaVerzijaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), konacna_verzija_potvrde)
        val granaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), grana)
        val ephorusBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), ephorus.toString())
        val znanjeBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), znanja)
```

Sl. 5.23. OnClickListener za chooseFileButton u ConfirmFinalStudentsPaper aktivnosti

```
        val rezultatiBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), rezultati)
        val jasnocaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), jasnoca)
        val samostalnostBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), samostalnost)
        val obrazlozenjeBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), obrazlozenje)
        val pismeniBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), pismeni_ocjena)
        val mentorBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), "$mentorSatnica sati, $sumentorSatnica sati")
        val call = postMakApi.confirmFinalVersion(id, user, token, granaBody,
konacnaVerzijaBody, ephorusBody, fullName, body, znanjeBody, rezultatiBody,
jasnocaBody, samostalnostBody, pismeniBody, obrazlozenjeBody, mentorBody)
        call.enqueue(object : Callback<Any> {
            override fun onResponse(call: Call<Any>, response: Response<Any>) {
                Toast.makeText(applicationContext, "Final version confirmed!",
Toast.LENGTH_LONG).show()
            }
override fun onFailure(call: Call<Any>, t: Throwable) {
            Toast.makeText(applicationContext, "Error: ${t.message.toString()}",
Toast.LENGTH_LONG).show()
        }

    })
}
singleFilePickerDialog.show()
```

Sl. 5.24. Nastavak onClickListenera za chooseFileButton u ConfirmFinalStudentsPaper aktivnosti

U zadnjoj mentorskoj aktivnosti *DiplomskiStrucniMentorPovjerenstvoActivity* mentor može odabrati povjerenstvo za obranu pomoću tri Spinnera. Kada ga odabere, može pritisnuti na button koji okida POST zahtjev i dobiti odgovarajuću Tost poruku o njegovoj uspješnosti. *OnClickListener* te tipke može se vidjeti na slikama 5.25. i 5.26.

```
spremiIzmjeneDiplomskiStrucni.setOnClickListener {
    val makApi = ServiceGenerator.createMentorPovjerenstvoDiplomskiStrucni()
    setUpValuesForBody()
    val oznakaPredsjednikaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), oznakaPredsjednika)
    val imePrezimePredsjednikaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), imePrezimePredsjednika)
    val oznakaMentorIliSumentorBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), oznakaMentorIliSumentor)
    val imeMentorIliSumentorBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), imeMentorIliSumentor)
    val oznakaClanUzPredsjednikaMentoraSumentoraBody =
RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
oznakaClanUzPredsjednikaMentoraSumentora)
    val imeClanUzPredsjednikaMentoraSumentoraBody =
RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
imeClanUzPredsjednikaMentoraSumentora)
```

Sl. 5.25. OnClickListener na tipku u aktivnosti DiplomskiStrucniMentorPovjerenstvoActivity

```
   val call = makApi.mentorPovjerenstvoDiplomskiStrucni(token,id, user,
oznakaPredsjednikaBody,imePrezimePredsjednikaBody,

oznakaMentorIliSumentorBody,imeMentorIliSumentorBody,oznakaClanUzPredsjednikaMentora
SumentoraBody,imeClanUzPredsjednikaMentoraSumentoraBody)
call.enqueue(object: Callback<Any>{
        override fun onResponse(call: Call<Any>, response: Response<Any>) {
            Toast.makeText(applicationContext,"Povjerenstvo confirmed!",
Toast.LENGTH_LONG).show()
        }
override fun onFailure(call: Call<Any>, t: Throwable) {
        Toast.makeText(applicationContext,"Povjerenstvo confirm error:
${t.message.toString()}", Toast.LENGTH_LONG).show()
    }

})
```

Sl. 5.26. Nastavak OnClickListenera tipke u aktivnosti DiplomskiStrucniMentorPovjerenstvoActivityja

5.3.3. Ocjenjivački dio

Kao i u glavnoj metorskoj aktivnosti, u ocjenjivačkoj aktivnosti *OcjenjivacActivity* studente pokazujemo unutar RecyclerViewa. Oni se dohvaćaju okidanjem GET zahtjeva u kodu koji se može vidjeti na slici 5.27.

```
val makApi = ServiceGenerator.createGetRadoviOcjenjvac()
val call = makApi.getRadoviOcjenjivaca(token,user)
call.enqueue(object: Callback<StudentsForMentorActivityRoot> {
    override fun onResponse(call: Call<StudentsForMentorActivityRoot>, response:
Response<StudentsForMentorActivityRoot>) {
        students= response.body()!!.data as ArrayList<StudentForMentorActivity>
        recyclerView.adapter= OcjenjivacStudentsRecyclerAdapter(students,token,user)
        if (students.isEmpty()){
            Toast.makeText(applicationContext,"No students found",
Toast.LENGTH_LONG).show()
        }
    }
    override fun onFailure(call: Call<StudentsForMentorActivityRoot>, t: Throwable)
{
        Toast.makeText(applicationContext, "Getting radovi procedure error
${t.message.toString()}", Toast.LENGTH_LONG).show()
    }
})
```

Sl. 5.27. Dohvaćanje studenata u OcjenjivacActivityju

Popunjavanje RecyclerViewa studentima radi se pomoću *OcjenjivacStudentsRecyclerAdaptera* koji je sličan *MentorsStudentRecyclerAdapteru*, a razlika je to što se u *onClickListeneru* na element u RecyclerViewu poziva drugačija metoda *PaperGradingActivity*, koja se može vidjeti na slici 5.28. Ako je riječ o studentu na preddiplomskom studiju, ocjenjivaču se pokreće aktivnost *PreddiplomskiOcjenjivacActivity*. Unutar nje on iz padajućeg izbornika odabire vrijednosti za znanja, rezultate i jasnoću te unosi mišljenje o radu i ako trebaju dodatne promijene rada. Klikom

na tipku Submit se okida POST zahtjev, a njen *onClickListener* se može vidjeti na slikama 5.29. i 5.

```
private fun paperGradingActivity(itemView: View, id: String,
                                 status: String, sumentor:String,
                                 tip_studija: String, mentor: String,
pismeni_ocjena: String) {
    if(tip_studija=="Preddiplomski"){
        if (statusIsValid(status)) {
            itemView.context.startActivity(Intent(itemView.context,
PreddiplomskiOcjenjivacActivity::class.java)
                    .putExtra("id", id)
                    .putExtra("token", token)
                    .putExtra("user", user)
                    .putExtra("pismeni_ocjena", pismeni_ocjena)
            )
        } else {
            Toast.makeText(itemView.context, "Invalid paper status",
Toast.LENGTH_LONG).show()
        }
    }
    else {
        if (statusIsValid(status)) {
            itemView.context.startActivity(Intent(itemView.context,
ObranaRadaActivity::class.java)
                    .putExtra("id", id)
                    .putExtra("token", token)
                    .putExtra("user", user)
                    .putExtra("mentor", mentor)
                    .putExtra("sumentor", sumentor)
            )
        } else {
            Toast.makeText(itemView.context, "Invalid paper status",
Toast.LENGTH_LONG).show()
        }
    }
}
```

Sl. 5.28. Metoda paperGradingActivity OcjenjivacStudentsRecyclerAdaptera

```
    val izmjene_odbor = RequestBody.create("multipart/create-
form".toMediaTypeOrNull(),izmjene)
    val call =
djelatnikApi.ocijeniRad(token,user,idRada,znanja_odbor,jasnoca_odbor,rezultati_odbor
,
            ukupna_ocjena,izmjene_odbor,misljenje_povjerenstva)
    call.enqueue(object: Callback<Any> {
        override fun onResponse(call: Call<Any>, response: Response<Any>) {
            Toast.makeText(applicationContext, "Ocjenjivanje success!",
Toast.LENGTH_LONG).show()
        }

        override fun onFailure(call: Call<Any>, t: Throwable) {
            Toast.makeText(applicationContext, "Ocjenjivanje error:
${t.message.toString()}", Toast.LENGTH_LONG).show()
        }

    })
}
```

Sl. 5.30. Nastavak onClickListenera za submit tipku u PreddiplomskiOcjenjivacActvityju

Ako je riječ o studentu na diplomskom ili stručnom studiju, klikom na studenta u *OcjenjivacActivity*ju otvorit će se aktivnost *ObranaRadaActivity*. U njemu ocjenjivač može unijeti mjesto, vrijeme i datum obrane rada, ocjenu na usmenom dijelu, kao i ukupnu ocjenu na ispitu. Uz njih unosi predsjednika povjerenstva obrane, kao i pitanja i mišljenja članova povjerenstva. Klikom na tipku Submit okida se POST zahtjev za obranu rada, a *onClickListener* te tipke se nalazi na slikama 5.31. i 5.32.

```
submitObrana.setOnClickListener {
    val makApi = ServiceGenerator.createObranaRada()
    val predsjednikImeBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), getPredsjednikIme())
    val predsjednikOznakaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), getPredsjednikOznaka())
    val datumObraneBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), datumObrane.text.toString())
    val satiObraneBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), vrijemePocetkaObrane.text.toString())
    val prostorijaObraneBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), prostorijaObrane.text.toString())
    val misljenjePovjerenstvaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), misljenjePovjerenstvaObrane.text.toString())
    val ukupnaOcjenaBody = RequestBody.create("multipart/form-
data".toMediaTypeOrNull(), ukupnaOcjenaObranaSpinner.selectedItem.toString())
val usmeniOcjenaBody = RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
ocjenaUsmenogObranaSpinner.selectedItem.toString())
val pitanjaBody =  RequestBody.create("multipart/form-data".toMediaTypeOrNull(),
pitanjaClanovaObrane.text.toString())
val call = makApi.obranaRada(token,user,id,
datumObraneBody,satiObraneBody,prostorijaObraneBody,usmeniOcjenaBody,ukupnaOcjenaBod
y, pitanjaBody,misljenjePovjerenstvaBody, predsjednikOznakaBody, predsjednikImeBody)
```

Sl. 5.31. OnClickListener za submit tipku u ObranaRadaActvityju

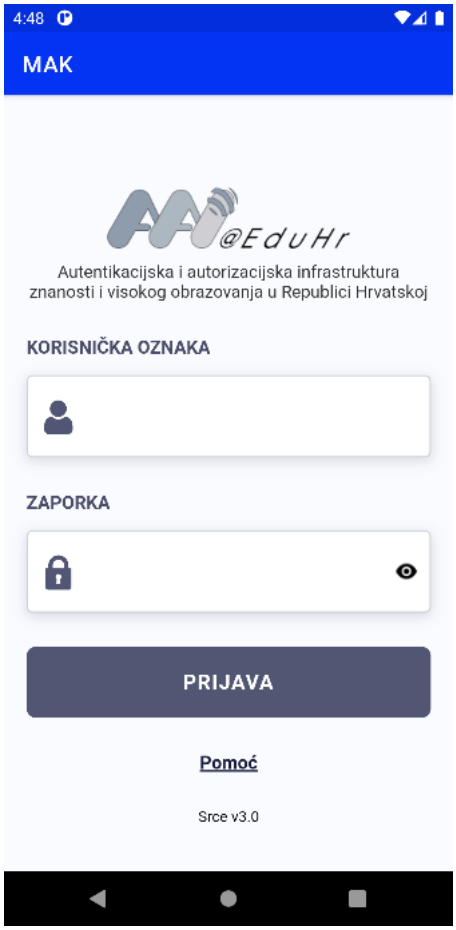
```
call.enqueue(object: Callback<Any> {
    override fun onResponse(call: Call<Any>, response: Response<Any>) {
        Toast.makeText(applicationContext, "Obrana finished!",
Toast.LENGTH_LONG).show()
    }

    override fun onFailure(call: Call<Any>, t: Throwable) {
        Toast.makeText(applicationContext, "Obrana error: ${t.message.toString()}",
Toast.LENGTH_LONG).show()
    }
})
```

Sl. 5.32. Nastavak onClickListenera za submit tipku u ObranaRadaActvityju

5.4. Ulaz i izlaz iz programa

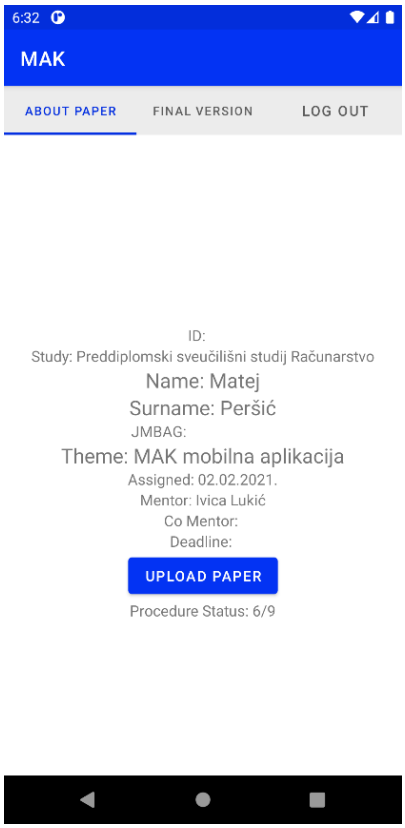
Svaki korisnik prilikom prvog korištenja aplikacije dolazi na WebView unutar kojeg se može prijaviti u MAK sustav. Početno stanje aplikacije se može vidjeti na slici 5.33.



Sl. 5.33. Prijava u MAK sustav

5.4.1. Student

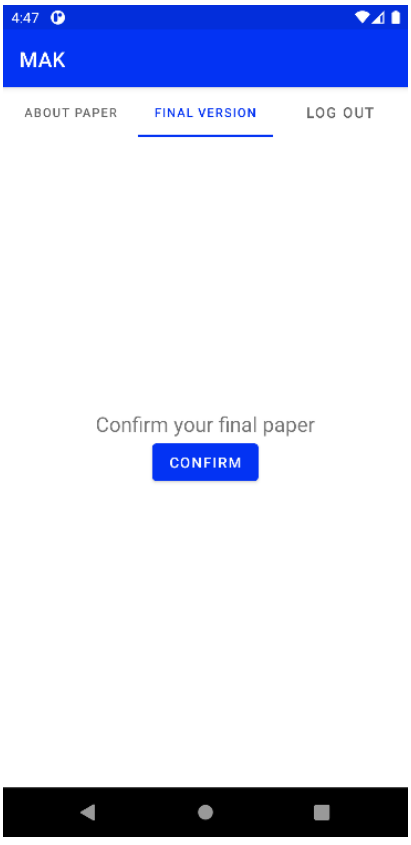
Kada se student ulogira, dolazi na *StudentsMainActivity* i vidi fragment *StudentFinalPaperInfo*, koji je na slici 5.34. Na njemu može vidjeti neke informacije o svom radu, kao i pokrenuti proceduru pritiskom na tipku Upload paper.



Sl. 5.34. Fragment StudentFinalPaperInfo

Student klikom na upload paper pokreće proces odabiranja završnog rada. Za primjer je uzet proces stavljanja početne verzije rada, za koju student dolazi u *StudentFinalPaperInitialVersionActivity*. Prije svega, odabire datoteku u sučelju kao na slici 5.12. Nakon toga dobiva Tost poruku o uspješnosti zahtjeva kao na slici 5.35. , gdje se može vidjeti kako izgleda sučelje nakon što je student odabrao početnu verziju rada i dobio je pozitivan odgovor. Slično izgleda i sučelje za odabir konačne verzije rada *StudentFinalPaperFinalVersionActivity*, a jedina razlika između aktivnosti za početnu i za konačnu verziju je u tekstu koji se nalazi u TextViewu iznad tipke Back i zahtjevu koji se šalje. Na slici 5.36. može se vidjeti *StudentConfirmsFinalPaper*, u kojem student može potvrditi svoj rad kada je on prijavljen u MAK i ISVU sustavu.

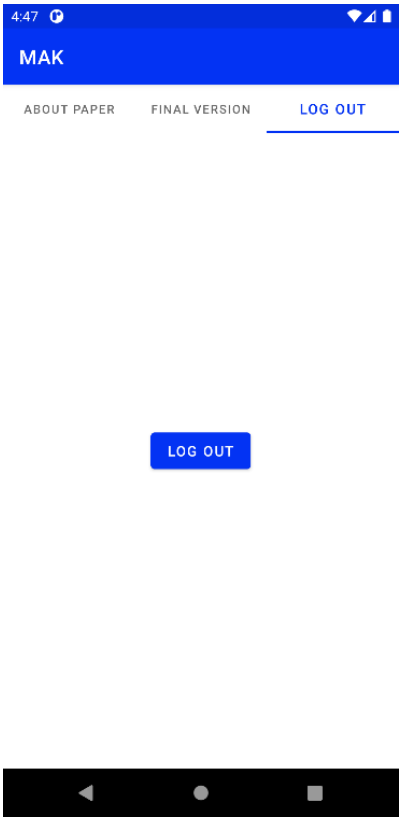
Sl. 5.35. Uspješan response u StudentFinalPaperInitialVersionActivityju



Sl. 5.36. StudentConfirmsFinalPaper fragment

Na slici 5.37. može se vidjeti *StudentLogOutFragment*, u kojem se student može odjaviti iz sustava

i vratiti se na *MainActivity*.



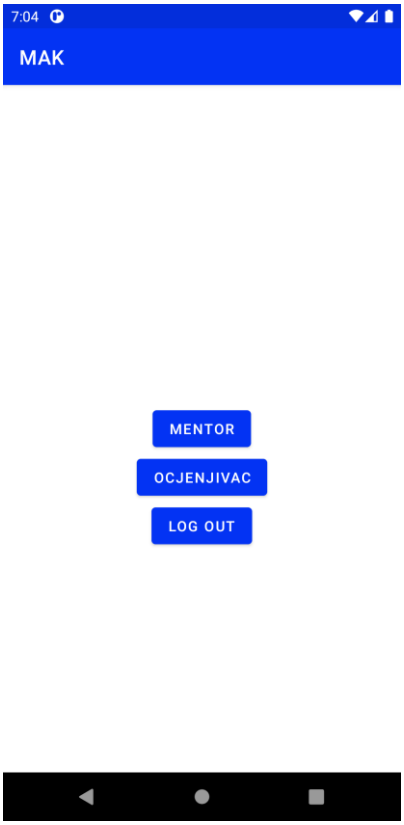
Sl. 5.37. StudentLogOutFragment

5.4.2. Mentor

Ostali korisnici nakon uspješne autentifikacije dolaze u *OtherUserMainActivity* gdje mogu dalje birati što će raditi. Primjer te aktivnosti se može vidjeti na slici 5.38.

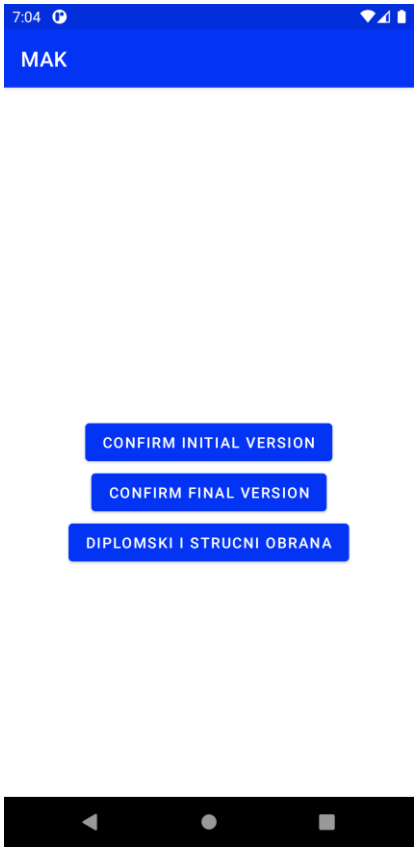
Klikom na tipku Mentor otvara se aktivnost *MentorsRecyclerActivity* u kojoj mentor može vidjeti svoje studente. Ako ih nema, dobiva odgovarajuću Tost poruku. Primjer *MentorsRecyclerActivityja* može se vidjeti na slici 5.39.

Kada mentor pritisne na jednog od studenata, otvori mu se aktivnost *MentorChoosesActivity*, u kojoj ima izbornik kao na slici 5.40. Klikom na tipku Confirm Initial Version mentoru se otvara aktivnost *ConfirmInitialStudentsPaper* koja služi za potvrđivanje početne verzije, i može se vidjeti na slici 5.41. Klikom na tipku Confirm Final Version mentoru se otvara aktivnost za potvrđivanje konačne verzije kao na slici 5.42. Klikom na tipku Diplomski i stručni obrana, mentoru se otvara aktivnost *DiplomskiStrucniMentorPovjerenstvoActivity*, u kojoj mentor odabire povjerenstvo za obranu rada. Izgled navedene aktivnosti može se vidjeti na slici 5.43.

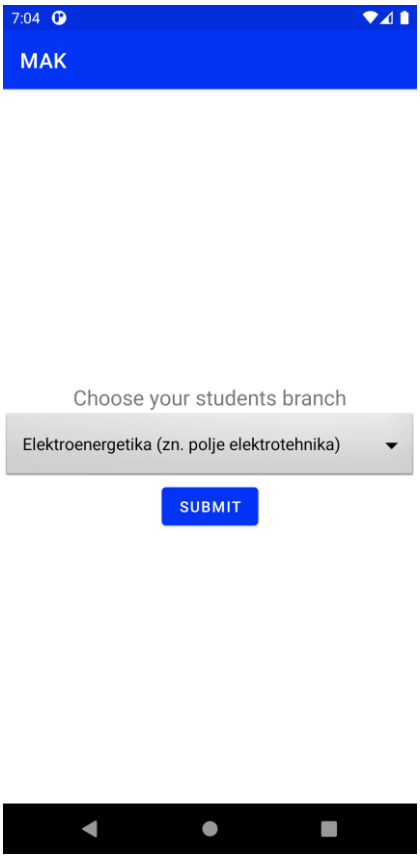


Sl. 5.38. OtherUserMainActivity

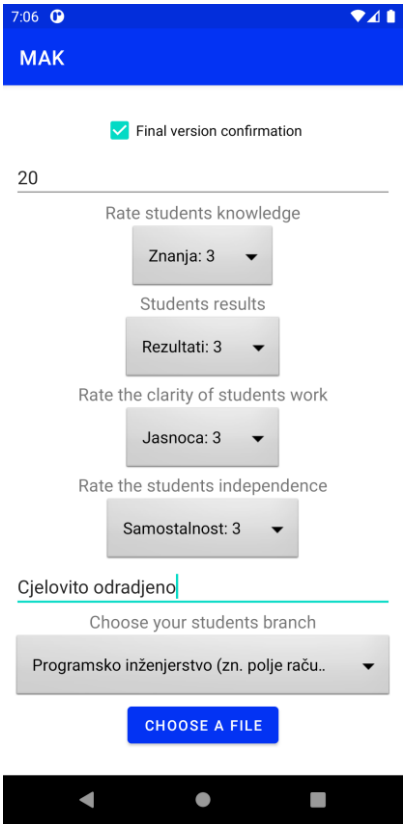
Sl. 5.39. MentorsRecyclerActivity



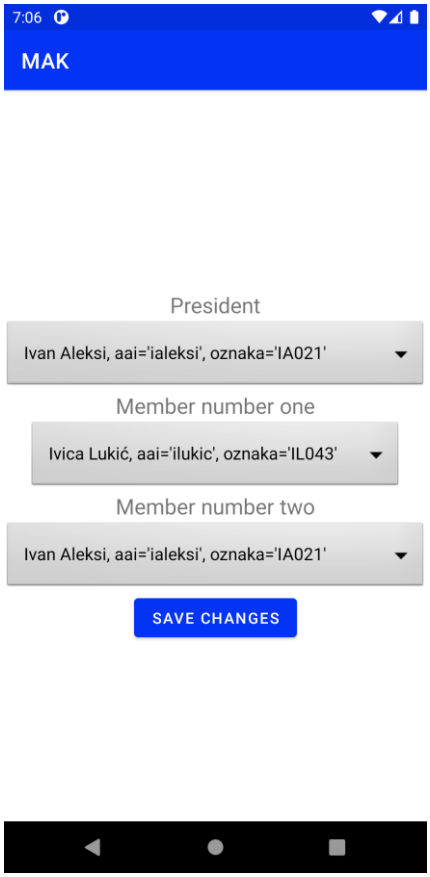
Sl. 5.40. MentorChoosesActivity



Sl. 5.41. ConfirmInitialStudentsPaper



Sl. 5.42. ConfirmFinalStudentsPaper

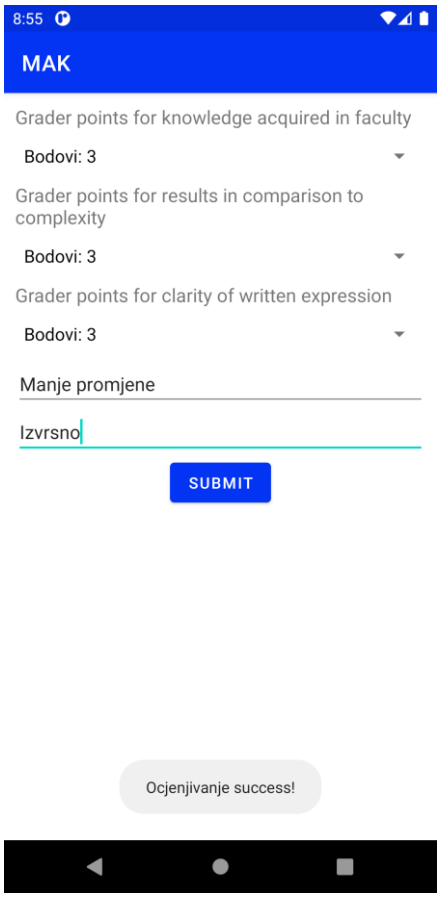


Sl. 5.43. DiplomskiStrucniMentorPovjerenstvoActivity

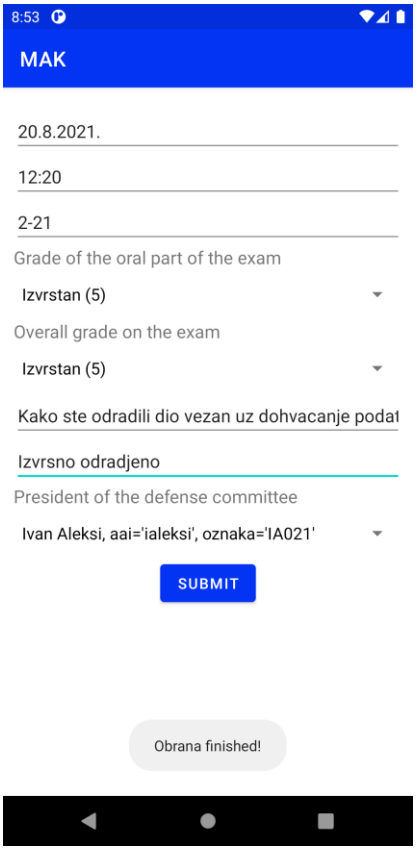
5.4.3. Ocjenjivač

Kao i mentor, ocjenjivač nakon uspješne autentifikacije dolazi u aktivnost *OtherUserMainActivity* koja se može vidjeti na slici 5.38. Klikom na tipku Ocjenjivac, ocjenjivaču se otvara aktivnost *OcjenjivacActivity* unutar koje ocjenjivač može vidjeti studente čije radove ocjenjuje. Ako ih nema, dobiva odgovarajuću poruku.

Kao i u *MentorsRecyclerActivityju* na slici 5.39. ocjenjivač vidi svoje studente u RecyclerViewu, i sučelje za njih je isto. Kada ocjenjivač pritisne na studenta koji je na preddiplomskom studiju, otvara mu se aktivnost *PreddiplomskiOcjenjivacActivity* koja se može vidjeti na slici 5.44. Kada ocjenjivač pritisne na druge studente, otvara mu se aktivnost *ObranaRadaActivity*, koja se može vidjeti na slici 5.45.



Sl. 5.44. PreddiplomskiOcjenjivacActivity



Sl. 5.45. ObranaRadaActivity

6. ZAKLJUČAK

Cilj završnog rada bio je osmisliti i implementirati programsko rješenje kojim će korisnici MAK sustava moći odrađivati aktivnosti i sa svojih mobilnih uređaja. Osmišljeno rješenje trebalo je implementirati za mobilne uređaje Android, u programskom jeziku Kotlin.

Izgledom jednostavna aplikacija omogućuje studentima, mentorima i ocjenjivačima obavljanje složenijih zadataka kroz jednostavno sučelje. Tijek aplikacije je jednostavan i korisnik može obavljati mnoge zadatke kao i u Web rješenju sustava MAK.

Programski jezik Kotlin moderni je programski jezik koji olakšava programiranje Android aplikacija s malo gubitaka u odnosu na Javu. Priložena su sva ključna znanja i funkcionalnosti programskog jezika Kotlina korištena pri izradi aplikacije. Najprivlačnije svojstvo programskog jezika Kotlin je lakša čitljivost i manje pisanja za razliku od Jave.

Stvorena aplikacija omogućuje studentima ažuriranje verzija svojih radova i jednostavnu kontrolu njihovih procedura. Mentorima i ocjenjivačima omogućuje jednostavno nadgledanje svojih studenata i rad s njihovim radovima, sve s njihovih mobilnih uređaja.

Moguće su dorade aplikacije za dodavanje ostalih uloga iz sustava MAK. Cilj završnog rada uspješno je ostvaren.

LITERATURA

[1] Povijest mobilnih uređaja

https://www.uswitch.com/mobiles/guides/history-of-mobile-phones/ [8.6.2021.]

[2] Zmija

https://theprint.in/features/nokias-snake-the-mobile-game-that-became-an-entire-generations-obsession/462873/ [14.6.2021.].

[3] Scott Galloway: The four, London, 2017.

[4] Profit u tržištu mobilnih uređaja

https://www.forbes.com/sites/johnkoetsier/2019/12/22/global-phone-profits-apple-66-samsung-17-everyone-else-unlucky-13/ [8.6.2021.]

[5] Android razvoj

Neil Smyth: Android 9, Kotlin i Android Studio 3.2 u jednoj knjizi, Beograd, 2018.

[6] Operacijski sustav android

https://www.androidauthority.com/history-android-os-name-789433/ [8.6.2021.]

[7] Retrofit

https://square.github.io/retrofit/ [8.6.2021.]

[8] Osnovni tipovi

https://kotlinlang.org/docs/basic-types.html [10.6.2021.]

[9] Kolekcije

https://kotlinlang.org/docs/collections-overview.html [10.6.2021.]

[10] FileAndFolderPicker

https://android-arsenal.com/details/1/8131#!description [8.6.2021.]

SAŽETAK

Cilj ovog rada bio je napraviti mobilnu verziju sustava MAK. Aplikacija je trebala omogućiti obavljanje zadataka korisnicima student, mentor i ocjenjivač sustava MAK. Aplikacija preko tokena i korisničkog imena koje korisnik dobije nakon uspješne autentifikacije, određuje je li riječ o studentu ili ne. Sukladno tome, korisniku prikazuje odgovarajuće sučelje. Unutar svake aktivnosti korisnik može raditi različite aktivnosti unutar sustava koje služe kao ulaz u sustav, a nakon njih dobiva odgovarajuće Tost poruke kao izlaz programa, koje mu govore o uspješnosti procedure ili grešku vezanu uz stanje rada, koje onemogućuje prijelaz u sljedeću aktivnost.

Ključne riječi: GET zahtjev, Kotlin, POST zahtjev, Retrofit, sustav MAK

ABSTRACT

Title: MAK mobile application

The goal of this project was to make a mobile version of the MAK portal. The application had to enable the students, mentors and reviewers to do their tasks inside of the portal. The application uses a token and user name each user is assigned with after a successful authentification and with it it decides if the user is a student or not. With that in mind, the user gets the appropriate user interface. Inside every activity the user can do various activities in the MAK portal, where users input is the input to the system. After an action, the user gets a Toast message which is used as an output from the system. Messages show the state of the procedure or an error which is related to the state of the students paper, and disables navigation to the next activity.

Keywords: GET request, Kotlin, POST request, Retrofit, portal MAK

ŽIVOTOPIS

Matej Peršić rođen je 20.8.1999. godine u Našicama. Završio je Osnovnu školu Josipa Kozarca Slatina. Godine 2018. završio je S.Š. Marka Marulića Slatina, smjer opća gimnazija te je iste godine upisao Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek. Student je treće godine preddiplomskog sveučilišnog studija računarstva.