Android mobilna aplikacija za dijeljenje nastavnih materijala

Milić, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij elektrotehnike, smjer Informatika

Android mobilna aplikacija za dijeljenje nastavnih materijala

Završni rad

Marko Milić

Osijek, 2021.

SADRŽAJ

1. UVOD

Pri prvom upisu na željeni fakultet studenti prve godine često budu izgubljeni i ne znaju odakle bi trebali učiti kako bi položili kolegije. Kako bi se olakšalo prilagođavanje na studentske dane studentima prve godine, ali i višim godinama osmišljena je Android aplikacija za dijeljenje nastavnih materijala. Kroz rad je prikazano kako je izvedeno čitanje, brisanje i spremanje u bazu podataka. Aplikacija će se sastojati od sljedećih dijelova:

- Prijava korisnika
- Registracija korisnika
- Odabir studija
- Odabir godine
- Preuzimanje materijala
- Dodavanje materijala

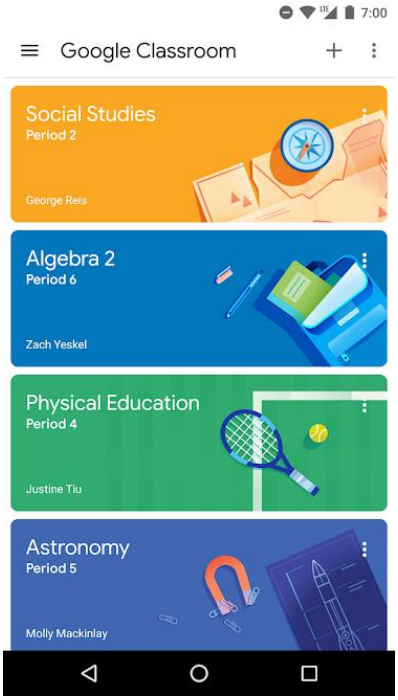
Pri registraciji korisnika, korisnik može odabrati vrstu računa, nastavnik ili student. Korisnik koji odabere nastavnika ima pravo ocjenjivanja i komentiranja materijala. Korisnik koji odabere studenta ima pravo na dodavanje materijala. Dodavanje materijala te registracija korisnika u aplikaciju se odvija preko obrasca koji se mora popuniti. Tehnologije korištene u izradi aplikacije su: Firebase za spremanje materijala, Kotlin programski jezik za implementaciju aplikacije, te Android na kojemu se testira funkcionalnost aplikacije i na kojem se aplikacija koristiti. Završni rad je podijeljen na pet poglavlja. Poglavlje dva predstavlja opis tehnologija koje su korištene za izradu završnog zadatka. Detaljno se prošlo kroz svaku tehnologiju. Poglavlje tri predstavlja opis procesa razvoja aplikacije. Upoznati ćemo se sa svim koracima procesa razvoja. Poglavlje četiri predstavlja opis izrade aplikacije te je kao zaključak rada poglavlje pet.

1.1. ZADATAK RADA

U završnom radu je potrebno razviti pozadinski servis za rad sa nastavnim materijalima te Android aplikaciju koja će koristiti taj servis. U Android aplikaciji je potrebno napraviti autentifikaciju i implementirati način za dohvaćanje, spremanje i preuzimanje nastavnih materijala sa pozadinskog servisa.

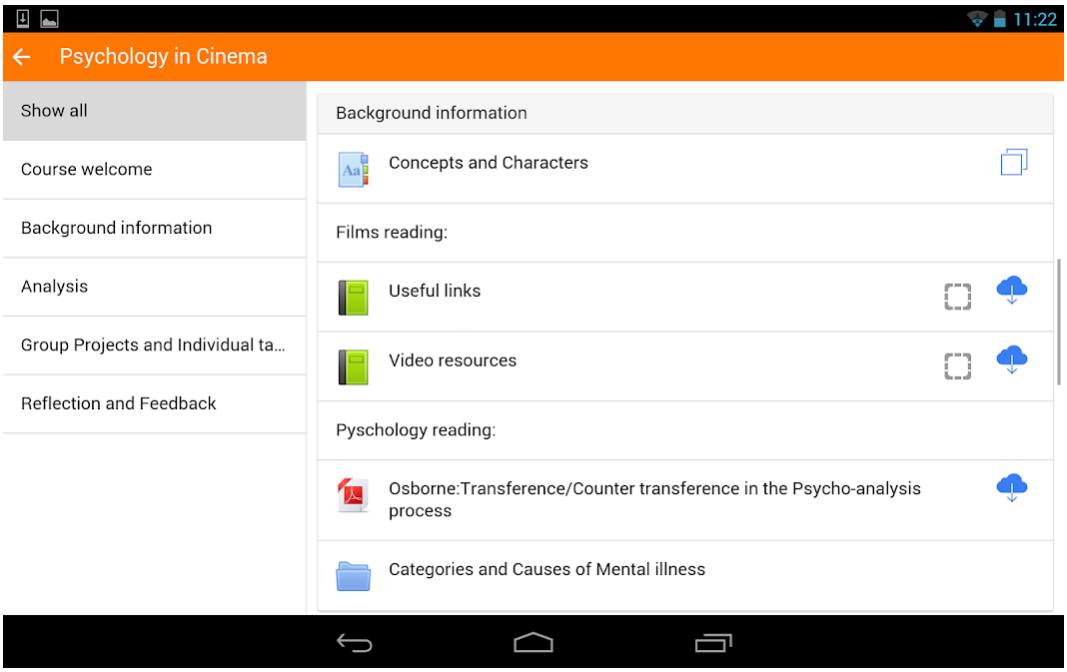
2. PREGLED PODRUČJA TEME RADA

Razmjena nastavnih materijala među studentima je jedna od najbitnijih stvari pri studiranju. Često studenti moraju na više mjesta tražiti nastavne materijale ili ispitivati kolege da im pošalju. Na Google Play Storeu postoje aplikacije za pomoć pri radu nastavnicima, ali je rijetkost naći aplikaciju koja olakšava način razmjene materijala među studentima. Među aplikacijama sa sličnom funkcionalnosti se nalazi Google Classroom. U Google Classroomu profesori mogu postaviti materijale za zadaću, video zapise, fotografije, dokumente te omogućava studentima da međusobno dijele zapise [1]. Izgled Google Classroom početne stranice se može vidjeti na slici 2.1..



Slika 2.1. – izgled Google Classroom aplikacije

Slijedeća aplikacija sa sličnom funkcionalnosti je Moodle. Moodle se može skinuti sa Google Play Storea te se na njoj mogu pregledavati materijali, slati poruke i polagati testove. Aplikacija ima mogućnost slanja obavijesti korisniku ako dođe do neke promjene. Korisnik može slati fotografije, video zapise i druge datoteke sa uređaja [2]. Izgled Moodle aplikacije se može vidjeti na slici 2.2..



Slika 2.2. – Izgled Moodle aplikacije

U praktičnom dijelu završnog rada je implementirana aplikacija koja je jednostavna za korištenje i na izravan način omogućava razmjenu nastavnih materijala te je u nekim segmentima slična ovim aplikacijama, ali je namijenjena samo razmjeni nastavnih materijala među studentima.

3. PREGLED KORIŠTENIH TEHNOLOGIJA

U ovom odlomku objašnjene su tehnologije i programski jezici korišteni u izradi praktičnog dijela završnog rada. Prva tri naslova u odlomku se odnose na tehnologije korištene pri izradi Android aplikacije za dijeljenje nastavnih materijala, dok se posljednja dva naslova odnose na tehnologije korištene u izradi pozadinskog servisa.

3.1. Kotlin

Kotlin je opće namjenski, besplatni programski jezik otvorenog koda. Dizajniran je za JVM (engl. Java Virtual Machine) i Android. JVM je kombinacija objektno orijentiranog programiranja te funkcionalnog programiranja. Fokusiran je na sigurnost te jasnoću koda. Kotlin je nastao u JetBrains kompaniji 2010. godine te je otvorenog koda od 2012., ali je prvu službenu objavu doživio u veljači 2016. godine sa 1.0 verzijom [3]. Trenutno ga vodi Kotlin Foundation, grupa kreirana od strane JetBrainsa i Googlea [4].

Kao programski jezik Kotlin je vrlo sličan Javi, ali je puno jasniji i s manje koda. Kada bi se gledala mogućnost, može se istaknuti kako u Kotlinu ima pametno „castanje“, funkcije višeg reda te lambda funkcije [5]. Kada se već uspoređuje Java i Kotlin može se reći da je moguće međudjelovanje između ta dva jezika, i u jednom i u drugom je moguće pozivanje koda drugog jezika. Također postoji automatizirani Java-u-Kotlin pretvarač ugrađen u IDE koji olakšava migraciju postojećeg koda. Kotlin može biti korišten za bilo kakvu vrstu razvoja, bilo to na strani poslužitelja ili na strani klijenta na webu ili Android razvoj. Također se radi na podršci za druge platforme kao što su macOS ili iOS. Za razvoj Android aplikacija koristi se Android Studio koji pruža prvorazrednu podršku za Kotlin [6].

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

Programski kôd 3.1. – Primjer Kotlin kôda

3.2. Firebase

Firebase je Baas (engl.Backend-as-a-Service). On donosi razvojnim inženjerima puno različitih servisa koji im mogu pomoći u razvitku kvalitetnih aplikacija te proširenju korisničke

baze. Razvijen je na Googleovoj infrastrukturi. Firebase je NoSQL program za bazu podataka koji sprema podatke u dokumente slične JSON-u [7].

Firebase se razvio iz Envolvea, startup tvrtke koji su financirali James Tamplin i Andrew Lee 2011. godine. Envolve je razvojnim inženjerima donosio API koji je omogućavao integraciju online chat funkcionalnosti na njihovim web stranicama. Razvojni inženjeri su koristili Envolve kako bi sinkronizirali podatke iz aplikacije u stvarnom vremenu svim korisnicima.

Tamplin i Lee su odlučili razdvojiti chat sistem od stvarno-vremenske arhitekture koja ju je pokretala. U rujnu 2011. su pokrenuli Firebase kao odvojenu tvrtku te su je pustili u javnost u veljači 2012. godine. Godine 2014. Google je preuzeo tvrtku Firebase koja je prije toga objavila dva proizvoda, Firebase Hosting i Firebase Authentication.

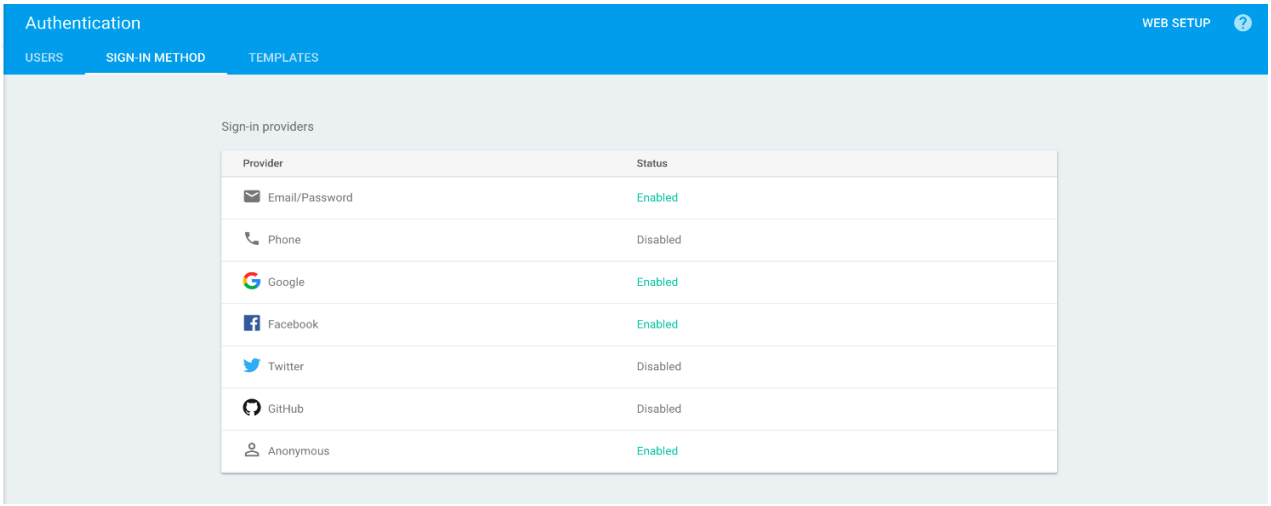
Servisi koje nudi Firebase:

- Google Analytics
- Firebase Cloud Messaging
- Firebase Authentication
- Firebase Realtime Database
- Cloud Firestore
- Firebase Storage
- Firebase Hosting
- ML Kit

Google Analytics je besplatna usluga koja nudi uvid u korištenje aplikacije. Firebase Cloud Messaging je više-platformsko rješenje za slanje poruka i obavijesti na Android, iOS i web aplikacije, također je besplatna. Firebase Authentication je servis koji provjerava autentičnost korisnika koristeći samo kod s klijentske strane. Primjer Firebase načina autentifikacije se može vidjeti na slici 3.2..

Firebase Realtime Database donosi stvarno-vremenu bazu podataka u pozadini kao servis. Servis razvojnim inženjerima donosi API koji omogućuje sinkronizaciju aplikacijskih podataka preko više klijenata i pohranu na Firebase oblak. Cloud Firestore je nasljednik originalne Firebase baze podataka, on omogućuje ugniježđeno raspoređene dokumente i polja umjesto pogleda preko stabla na stvarno-vremensku bazu podataka. Firebase Storage omogućuje sigurno učitavanje i skidanje datoteka za Firebase aplikacije, bez obzira na kvalitetu veze. Firebase Hosting je statični i

dinamični web hosting servis. ML kit je mobilan sistem za strojno učenje za razvojne inženjere, njegovi API-i omogućuju detekciju lica, skeniranje barkodova i slično [8].



Slika 3.2. – Primjer izgleda Firebase autentifikacije [9]

3.3. Android operacijski sustav

Android je operacijski sustav baziran na Linux kernelu. Treba napomenuti kako on nije verzija Linuxa, nego je u srodstvu s Linuxom. Android je operacijski sustav dizajniran za mobilne uređaje, sve što se vidi na ekranu uređaja je dio operacijskog sustava. Operacijski sustav je podijeljen na razne verzije, sa svakom verzijom se uvodi nekakav novitet. Kada se pogleda izgled može se vidjeti različite inačice Androida, tako postoji Stock Android koji je najosnovniji, ali svaki proizvođač Android uređaja može napraviti svoju verziju izgleda [10].

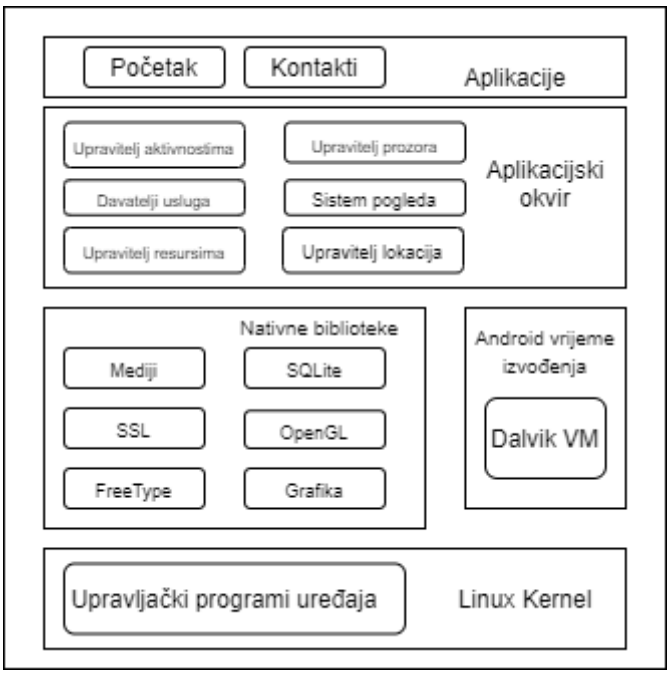
Android Incorporation, tvrtka zaslužna za Android operacijski sustav je stvorena 2003. godine u Sjedinjenim Američkim Državama. Njezin začetnik je Andy Rubin. Godine 2005. Google je preuzeo Android Incorporation te je postao dio Google Incorporationa. Android je u početku bio predviđen za kamere, ali su napravili zaokret na pametne telefone. Android je nadimak koji je dodijeljen Andy Rubinu zbog njegove ljubavi prema robotima. Godine 2007., Google najavljuje razvoj Android operacijskog sustava, dok se 2008. predstavio prvi Android pametni telefon tvrtke HTC [11].

Arhitektura androida:

- Linux kernel

- Nativne biblioteke
- Android vrijeme izvođenja
- Aplikacijski okviri
- Aplikacije

Primjer modela Android Arhitekture može se vidjeti na slici 3.3..



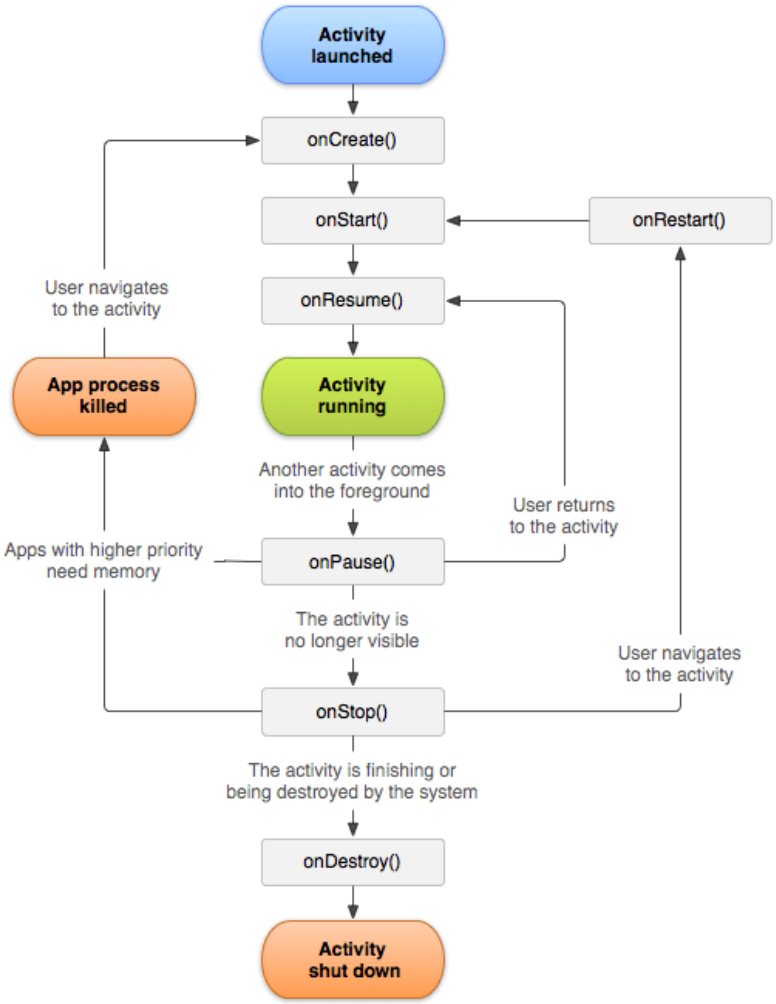
Slika 3.3. – Android arhitektura

Linux kernel je središte Android arhitekture te se nalazi u korijenu iste. Zadužen je za pogonitelje uređaja, upravljanje napajanjem, upravljanje memorijom, dostupnosti resursa te upravljanje uređajem. Nativne biblioteke se koriste kako bi se pristupalo fizičkim komponentama uređaja, kao što su senzori ili prepoznavanje dodira.

Android vrijeme izvođenja (engl. Android Runtime) je zaduženo za pokretanje Android aplikacija, koristi DVM (engl. Dalvik Virtual Machine) koji je sličan Java Virtual Machineu, ali je optimiziran za mobilne uređaje. Aplikacijski okviri uključuju Android API-je, kao što su UI, telefonija, resursi, lokacije itd.. Također pruža puno klasa i sučelja za razvijanje aplikacija. Iznad aplikacijskih okvira stoje aplikacije. Sve aplikacije kao što su home, kontakti, postavke itd., koje koriste Android vrijeme izvođenja i biblioteke [12].

3.3.1. **Android komponente**

Android komponenta je komad koda koji ima predefinirani životni vijek. Osnovne komponente Androida su aktivnost (engl. Activity), pogled (engl. View), namjera (engl. Intent), usluge (engl. Services), davatelji usluga (engl. Content Providers), fragmenti (engl. Fragments). Aktivnost je klasa koja predstavlja jedan zaslon. Aktivnost ima svoj životni vijek (Slika 3.4.) .



Slika 3.4. – Životni vijek aktivnosti [13]

Pogled je element UI-a (engl. User interface), npr. gumb, oznaka i sl. Namjera se koristi za pozivanje komponenti, uglavnom se koristi za pokretanje servisa, paljenje aktivnosti, prikazivanje web stranice i sl.. Primjer namjere se može vidjeti u programskom kôdu 3.2. Servis je pozadinski

proces koji može raditi dugo vremena. Davatelj usluga se koristi za dijeljenje podataka među aplikacijama. Fragmenti su kao dijelovi aktivnosti, jedna aktivnost može sadržavati više fragmenata [14].

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.javatpoint.com"));
startActivity(intent);
```

Programski kôd 3.2. – Primjer namjere

3.4. Java

Java je objektno orijentiran programski jezik zasnovan na rad sa klasama (eng. Class). Aplikacije napisane u Javi se kompajliraju u bytecode koji se može izvršiti na bilo kojem Java Virtual Machineu (JVM) koji je pokrenut na bilo kojoj računalnoj arhitekturi. Java sintaksa je slična C programskom jeziku, samo što nema previše nisko-razinskih operacija. Kako je prethodno napisano, u Javi su svi programi napisani pomoću klasa. Možemo zamisliti klasu kao nacrt i instrukcije za izradu objekta koju tvornica izrađuje [15]. Sun tvrtka je originalno razvila Javu, ali ju je 2009. Oracle Corporation preuzeo.

3.4.1. Java Spring

Spring je najpopularniji okvir za razvijanje aplikacija. Pomoću njega milioni programera diljem svijeta razvijaju dobro izvodeći, lagano testirajući i kod koji se može iznova koristiti. Spring je okvir otvorenog koda [16]. Pomoću Springa razvijanje Web pozadinskih servisa je postalo vrlo jednostavno jer sve što nam treba je dostupno na jednom mjestu. Od kada je nastao Spring Boot, sve stvari koje su bile komplicirane u Springu su riješene same po sebi pod time se misli na konfiguracije pomoću XML-a i praćenje verzija zavisnosti. Spring u sebi ima ugrađen Tomcat server tako da u njemu ima sve što treba da se aplikacija odmah nakon implementacije može staviti na korištenje na internet.

4. RAZVOJ APLIKACIJE ZA DIJELJENJE NASTAVNIH MATERIJALA

4.1. Korisnički zahtjevi

Kako bi se započelo s razvitkom aplikacije najprije treba utvrditi tko će ju sve koristiti. U aplikaciji postoje dvije vrste korisnika. Prvu vrstu korisnika čine studenti koji će svoje nastavne materijale učitavati u bazu podataka, a samim time i u aplikaciju.

Studenti će moći koristiti slijedeće mogućnosti:

- Odabir studija i godine
- Pregled nastavnih materijala
- Dodavanje nastavnih materijala
- Skidanje nastavnih materijala

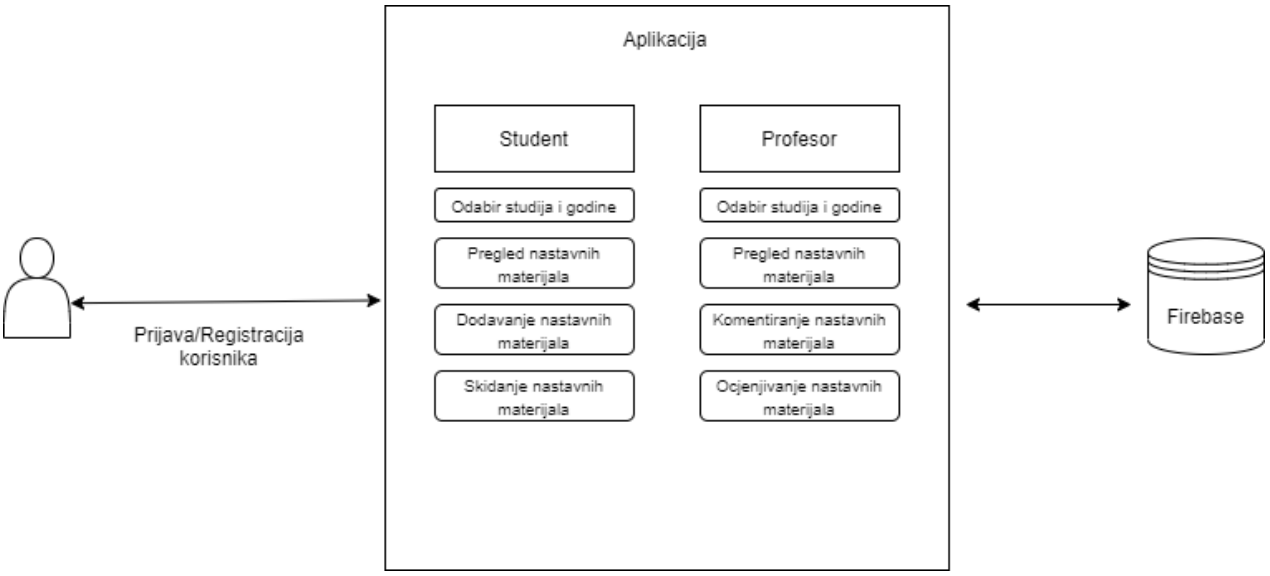
Drugu vrstu korisnika čine nastavnici. Korisnik koji je prijavljen u aplikaciju kao nastavnik će imati slijedeće mogućnosti:

- Odabir studija i godine
- Pregled nastavnih materijala
- Komentiranje nastavnih materijala
- Ocjenjivanje nastavnih materijala

4.2. Izgled sustava

Nakon utvrđivanja korisničkih zahtjeva potrebno je razraditi od kojih dijelova će se aplikacija sastojati. Najprije se utvrđuje izgled sustava te način na koji će sustav funkcionirati te na posljetku način implementacije korisničkih zahtjeva.

Ako se pogleda sliku 4.1., može se vidjeti arhitektura aplikacije. Aplikacija se sastoji od samog načina kako je izgled aplikacije zamišljen, te realizacije iste pomoću dvije baze podataka. Pomoću baza podataka pohranjujemo podatke koje želimo zadržati. Jedna baza pohranjuje podatke lokalno dok druga pohranjuje na cloud (Firestore).



Slika 4.1. – Izgled sustava Android aplikacije

Kada se razradi arhitektura aplikacije, posvećuje se rađenju UI izgleda. Pri izradi treba pripaziti na određene stvari. Kao prvo treba se pripaziti na korištenje riječi koje nisu dvosmislene, te kada se odabere koju riječ će se korist0iti držati se te riječi kod dijelova koji se mogu nazvati različitim riječima. Također vrijedi isto i za gumbe, treba odabrati gumb koji će imati jednu funkcionalnost i izgled.

Pripaziti treba na to da korisnik u svakom trenutku zna što se događa u aplikaciji, ako je stavio datoteku na učitavanje da bude jasno koliko je učitano. Dizajn aplikacije treba biti minimalističan, te u svakom prozoru treba biti gumb za nazad jedan korak te gumb za početni zaslon.

Pri odabiru boja treba izbjegavati zasićene boje. Također pri odabiru boje pozadine i teksta treba voditi računa o čitljivosti i vidljivosti. Ako smo se odlučili na boje, treba uzeti u obzir kako neki ljudi ne prepoznaju boje, te zbog toga treba izbjegavati korištenje boja za ključne funkcionalnosti aplikacije. Mockup aplikacije se može vidjeti na slici 4.2..

Slika 4.2. – Mockup aplikacije

4.3. Firebase autentifikacija

Korisnik prije korištenja aplikacije mora provesti registraciju i prijavu. Navedene radnje će biti provedene preko Firebase autentifikacije. Kako bi se uključila autentifikacija prvo što se mora napraviti je račun i projekt na Firebase stranici. Nakon kreiranja projekta omogućena je registracija putem email-a. U Firebase konzoli se mogu namjestiti različiti načini autentifikacije. Za potrebe završnoga rada biti će uključena samo Email/Password autentifikacija.

4.4. Izrada Cloud Firestore baze podataka

Kada je definiran izgled aplikacije te omogućena Firebase autentifikacija, slijedeći korak je napraviti bazu podataka Cloud Firestore. Kako bi se napravila baza mora se otići na Firebase stranicu, odabrati projekt te kreirati bazu podataka. Cloud Firestore je NoSQL baza podataka koja naše podatke sprema u dokumente koji su organizirani u kolekcije za razliku od tipične SQL baze podataka. Nakon što se kreira baza podataka, istu se mora uključiti u projekt.

4.5. Razvoj Java Spring Boot pozadinskog servisa

Kada su definirani izgleda aplikacije te utvrđene funkcionalnosti koje su potrebne treba razraditi izgled pozadinskog servisa na koji će biti slani upiti iz aplikacije te krajnje točke preko kojih će se vršiti dohvaćanje i spremanje nastavnih materijala u serversku bazu podataka. Pošto je u Spring Boot aplikaciji uključen Hibernate nema potrebe za izradom tablica u bazi već Java klasa predstavlja tablicu u bazi koja se kreira prvi prvom pokretanju servisa. Klasa koja će predstavljati tablicu u bazi može se vidjeti u kôdu 4.1..

```
@Entity
@Table(name = "files")
public class File {

	@Id
	@GeneratedValue(generator = "uuid")
	@GenericGenerator(name = "uuid", strategy = "uuid2")
	private String id;

	private String fileName;
	private String fileType;
	private String study;
	private String subject;
	private String comment;

	@Lob
	private byte[] data;
```

Programski kôd 4.1. – File klasa koja se sprema u bazu

Krajnje točke koje će biti potrebne su dohvaćanje, spremanje i izlistanje po studiju i godini te predmetu. Za dohvaćanje i izlistanje će se koristiti GET HTTP zahtjev, a za spremanje POST HTTP zahtjev.

5. IZRADA APLIKACIJE

5.1. Izrada Java Spring Boot servisa

Kako bi se počelo sa izradom servisa prvo je potrebno otići na Spring Initializr stranicu i odabrati potrebne zavisnosti kao i Java verziju te skinuti projekt i raspakirati ga na željeno mjesto. Nakon što je projekt importan u Spring Tool Suite potrebno je kreirati željene pakete po kojima će biti odvojene logične cjeline aplikacije (MVC model). Model paket se sastoji od File klase koja predstavlja tablicu u bazi te FileResource i FileResourceAssembler klasa koje služe za slanje podataka iz baze putem JSON objekata. U *controller* paketu se nalaze API krajnje točke servisa iz kojih se poziva Service metoda u kojoj se nalazi sva logika aplikacije. Funkcionalnosti koje se nalaze u servisu su dohvaćanje nastavnih materijala prema studiju i godini i predmetu, skidanje nastavnih materijala i dodavanje nastavnih materijala u bazu podataka.

```
@GetMapping("/files/{study}/{subject}")
public ResponseEntity<List<FileResource>> getByStudyAndYear(@PathVariable
String study, @PathVariable String subject) {
      try {
                  return new
ResponseEntity<List<FileResource>>(fileService.getByStudyAndSubject(study,
subject), HttpStatus.OK);
      } catch (Exception e) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
      }
}
```

Programski kôd 5.1. – Java kôd za kontroler metodu *getByStudyAndYear()*

Spring anotacije igraju veliku ulogu u kreiranju samih API metoda koje će biti pozivane iz android aplikacije. Za metodu 5.1. koristi se @GetMapping*(„/files/{study}/{subject}“)* koja govori da će se ta metoda pozvati ako dođe http poziv GET na krajnju točku */files/studij* i godina/predmet. *FileService* klasa ima definiranu metodu za svaku API krajnju točku i vraća željene podatke koje će Spring pretvoriti u JSON te ih proslijediti u android aplikaciju. *FileService* metodu za dohvaćanje datoteka po studiju i godini te po predmetu vraća listu uzlazno sortiranih podataka po imenu datoteke te se može vidjeti u kôdu 5.2..

```
public List<FileResource> getByStudyAndSubject(String study, String
subject) {
      List<File> fileFromDb =
      fileRepository.findByStudyAndSubjectOrderByFileNameAsc(study,
      subject);
      log.debug("Finding files with study: {}", study);
      List<FileResource> fileResource = new ArrayList<>();
      fileFromDb.forEach(f ->
      fileResource.add(resourceAssembler.toResource(f)));
      return fileResource;
}
```

Programski kôd 5.2. – Java kôd za *service* metodu *getByStudyAndSubject()*

Aplikacija uz JSON podatke vraća i pripadajući HTTP status (OK, Internal Server Error i Not Found). Uz *Controller*, *Model* i *Service* postoji *Repository* paket u kojem se nalazi *FileRepository* sučelje preko kojeg je riješeno pitanje SQL upita. Spring Data JPA preko *Repository* sučelja sastavlja upite prema ključnim riječima u nazivu metode. Spring Data JPA *repository FileRepository* u sebi metodu sučelja *findByStudyAndSubjectOrderByFileNameAsc()* može biti viđena u kôdu 5.3..

```
@Repository
public interface FileRepository extends JpaRepository<File, String> {
      List<File> findByStudyAndSubjectOrderByFileNameAsc(String study,
      String subject);
}
```

Programski kôd 5.3. – Java kôd za *repository* interface metodu *findByStudyAndSubject()*

Dohvaćanje nastavnih materijala ima svoju krajnju točku */download/{id}*, *id* predstavlja oznaku preko koje se različiti podaci u bazi razlikuju. *Id* koji je predan kontroler metodi se prosljeđuje u *FileService* metodu koja sadrži logiku. Kontroler metoda može biti viđena u kôdu 5.4..

```
GetMapping("/download/{id}")
public ResponseEntity<Resource> downloadFile(@PathVariable String id) {
      File dbFile = fileService.getFile(id);
      return
      ResponseEntity.ok().contentType(MediaType.parseMediaType(dbFile.getFi
      leType())).header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
      filename=\"" + dbFile.getFileName() + "\"").body(new
      ByteArrayResource(dbFile.getData()));
```

Programski kôd 5.4. – Java kôd za kontroler metodu *downloadFile()*

Pozivanjem download krajnje točke Service metoda *getFile()* se poziva, u njoj se nalazi logika za skidanje nastavnih materijala preko *fileRepositorya*. Nastavni materijal se traži preko Ida i ako nije nađen novi *Exception* se baca. Service metoda za *getFile()* se može vidjeti u kôdu 5.5..

```
public File getFile(String fileId) {
      log.debug("Finding file with id: {}", fileId);
      return fileRepository.findById(fileId).orElseThrow(() -> new
      RuntimeException("File not found"));
}
```

Programski kôd 5.5. – Java kôd za *service* metodu *getFile()*

Spremanje nastavnih materijala u serversku bazu podataka se odvija preko */upload/{study}/{subject}* krajnje točke iz koje se poziva Service metoda za spremanje. Kontroler prima uz *{study}* i *{subject}* parametar još i file parametar koji se preko tijela HTTP poziva predaje. Kôd za kontroler metodu je prikazan u kôdu 5.6..

```
@PostMapping("/upload/{study}/{subject}")
public ResponseEntity<File> uploadFile(@RequestParam("file") MultipartFile
file, @PathVariable("study") String study, @PathVariable("subject") String
subject) {
      try {
            return new ResponseEntity<>(fileService.storeFile(file, study,
            subject), HttpStatus.CREATED);
      } catch (Exception e) {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
      }
}
```

Programski kôd 5.6. – Java kôd za kontroler metodu *uploadFile()*

Pozivom */upload* krajnje točke se pokreće Service metoda *storeFile()* u kojoj je logika pomoću koje se nastavni materijali spremaju u bazu podataka pomoću *fileRepositorya*. *StoreFile()* metoda u sebi sadrži *try-catch* blok za upravljanje greškama u kojem se predani nastavni materijal sprema u bazu podataka. Metodu za spremanje nastavnih materijala u bazu se može vidjeti u kôdu 5.7..

```
public File storeFile(MultipartFile file, String study, String subject) {
String fileName = StringUtils.cleanPath(file.getOriginalFilename());
      try {
            if (fileName.contains("..")) {
                  throw new RuntimeException("Invalid path sequence");
            }

            File dbFile = new File(fileName, file.getContentType(), study,
            subject, file.getBytes());
            log.debug("Storing file with name: {}", dbFile.getFileName());
            return fileRepository.save(dbFile);
      } catch (IOException e) {
            throw new RuntimeException("Could not store file" + fileName);
      }
}
```

Programski kôd 5.7. – Java kôd za *service* metodu *storeFile()*

```
@PostMapping("/files/comment/{fileId}")
public ResponseEntity<FileResource> postFileComment(@PathVariable String
fileId, @RequestBody String comment){
      try {
            return new
            ResponseEntity<FileResource>(fileService.postFileComment(fileId
            , comment), HttpStatus.OK);
      } catch (Exception e){
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
      }

}
```

Programski kôd 5.8. – Java kôd za *controller* metodu *postFileComment()*

Postavljanje komentara se odvija preko */files/comment/{fileId}* krajnje točke. U metodi *postFileComment()* se poziva *fileService.postFileComment()* metoda u kojoj se vrši spremanje komentara u bazu podataka. Kontroler metoda *postFileComment()* prima *fileId* i u tijelu poziva sam komentar. Kontroler metoda se može vidjeti u kôdu 5.8..

```
public FileResource postFileComment(String fileId, String comment){
      log.debug("Finding file with id: {}", fileId);
      File file = fileRepository.findById(fileId).get();
      file.setComment(comment);
      log.debug("Saving comment to file with id: {}", fileId);
      return resourceAssembler.toResource(fileRepository.save(file));
}
```

Programski kôd 5.9. – Java kôd za *service* metodu *postFileComment()*

Unutar *postFileComment()* kontroler metode se poziva *service* metoda *postFileComment()*. U *service* metodi se odvija traženje datoteke u bazi po idu koji se predaje metodi te nakon što je datoteka pronađena postavlja se komentar za istu. Metoda za spremanje komentara u bazu se može vidjeti u kôdu 5.9..

```
@GetMapping("/files/{study}/{subject}/{findParam}")
public ResponseEntity<List<FileResource>>
getByStudyAndYearAndFindByParam(@PathVariable String study, @PathVariable
String subject, @PathVariable String findParam){
      try {
            return new
            ResponseEntity<List<FileResource>>(fileService.getByStudyAndYea
            rAndFindByParam(study, subject, findParam), HttpStatus.OK);
      } catch (Exception e) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
      }
}
```

Programski kôd 5.10. – Java kôd za *controller* metodu *getByStudyAndYearAndFindByParam()*

Kontroler metoda *getByStudyAndYearAndFindByParam()* služi za pretraživanje nastavnih materijala te vraća sve materijale koji odgovaraju unesenoj ključnoj riječi. Kontroler metoda se poziva kada se pošalje zahtjev na krajnju točku */files/{study}/{subject}/{findParam}*. Metodu za pretraživanje nastavnih materijala se može vidjeti u kôdu 5.10..

Unutar *getByStudyAndYearAndFindByParam()* kontroler metode se poziva *service* metoda istog imena koja sadrži logiku za pretraživanje nastavnih materijala po ključnoj riječi. Dohvaćanja materijala koji odgovaraju ključnoj riječi je riješene pomoću ugrađene JPA logike rađenja upita u

bazu pomoću *fileRepository* metode koja vraća listu odgovarajućih zapisa sortiranih uzlazno po imenu datoteke. Navedenu *service* metodu se može vidjeti u kôdu 5.11..

```
public List<FileResource> getByStudyAndYearAndFindByParam(String study,
String subject, String param){
      List<File> fileFromDb =
      fileRepository.findByStudyAndSubjectAndFileNameContainingIgnoreCaseOr
      derByFileNameAsc(study, subject, param);
      log.debug("Finding files with name like: {}", param);
      List<FileResource> fileResource = new ArrayList<>();
      fileFromDb.forEach(f ->
      fileResource.add(resourceAssembler.toResource(f)));
      return fileResource;
}
```

Programski kôd 5.11. – Java kôd za *service* metodu *getByStudyAndYearAndFindByParam()*

U Service klasi je implementiran *logger* pomoću kojeg se vide greške ako su se dogodile i može se pratiti izvođenje metoda u konzoli ili se može namjestiti da se zapisuje u log datoteku. *Logiranje* je riješeno pomoću slf4j *logger* klase.

5.2. Izrada korisničkog sučelja

Sučelje android aplikacije treba biti jednostavno i izravno kako bi bilo što lakše snaći se u njoj. Ulaskom u aplikaciju korisniku bude predstavljen zaslon za prijavu ili u slučaju da nema račun ima odabir registracije. Prijava i registracija se vrši preko emaila i zaporke. Način provjere za prijavu se nalazi u kôdu 5.12..

```
private fun onLoginClicked() {
      auth.signInWithEmailAndPassword(authEmail.text.toString(),
      authPassword.text.toString()).addOnCompleteListener {
            if (it.isComplete && it.isSuccessful) {
                startActivity(HomePageActivity::class.java)
            } else {
                Log.d("+++", "Login failed")
            }
        }
}
```

Programski kôd 5.12. – Kotlin kôd za login

Nakon uspješne registracije/prijave prikazuje se HomeFragment u kojem se nalaze studijski smjerovi u Recycler Viewu. Svaki studijski smjer na sebi ima onClick slušače te pritiskom na odabrani studijski smjer poziva se fragment u kojem se nalaze smjerovi za pojedini studijski smjer. Prethodno objašnjeno je prikazano u kôdu 5.13..

```
override fun setOnClickListeners() {
      strucni.setOnClickListener { onStrucniStudijClicked() }
      preddiplomski.setOnClickListener { onPreddiplomskiClicked() }
      diplomski.setOnClickListener { onDiplomskiClicked() }
 }
```

Programski kôd 5.13. – Kotlin kôd za HomeFragment onClick slušače

Nakon što je otvoren CourseFragment ponuđen je odabir smjera i godine sa onClick slušačima na njima. Smjerovi su prikazani pomoću Recycler Viewa. Odabirom smjera i godine otvara se SubjectFragment u kojem se nalazi popis predmeta za taj smjer i godinu. Opisani CourseFragment se može vidjeti u kôdu 5.14..

```
private fun setRecyclerViewData() {
        when(studijClicked){
            STRUCNI_STUDIJ ->
            adapter.setData(SubjectTitlesRepository().getTitlesStrucni())
            PREDDIPLOMSKI_STUDIJ ->
            adapter.setData(SubjectTitlesRepository().getTitlesPredDipl())
            DIPLOMSKI_STUDIJ ->
            adapter.setData(SubjectTitlesRepository().getTitlesDipl())
        }
}
```

Programski kôd 5.14. – Kotlin kôd za CourseFragment

Pokretanjem SubjectFragmenta otvara se Recycler View lista sa predmetima za odabrani smjer i godinu. Svaki predmet ima onClick slušače koji na pritisak otvara ScriptFragment u kojemu se nalaze nastavni materijali za odabrani studijski smjer, smjer i godinu te predmet. Prethodno opisano se može vidjeti u kôdu 5.15..

```
private fun setRecyclerViewData() {
        when(onSmjerClicked){
            RACUNARSTVO_1_GODINA -> {
            adapter.setData(SubjectTitlesRepository().getTitlesRacunarstvo
            1())
            SelectedStudyAndSubjectRepository().setSelectedStudy("Računars
            tvo stručni 1. godina")
            }
```

Programski kôd 5.15. – Kotlin kôd za SubjectFragment

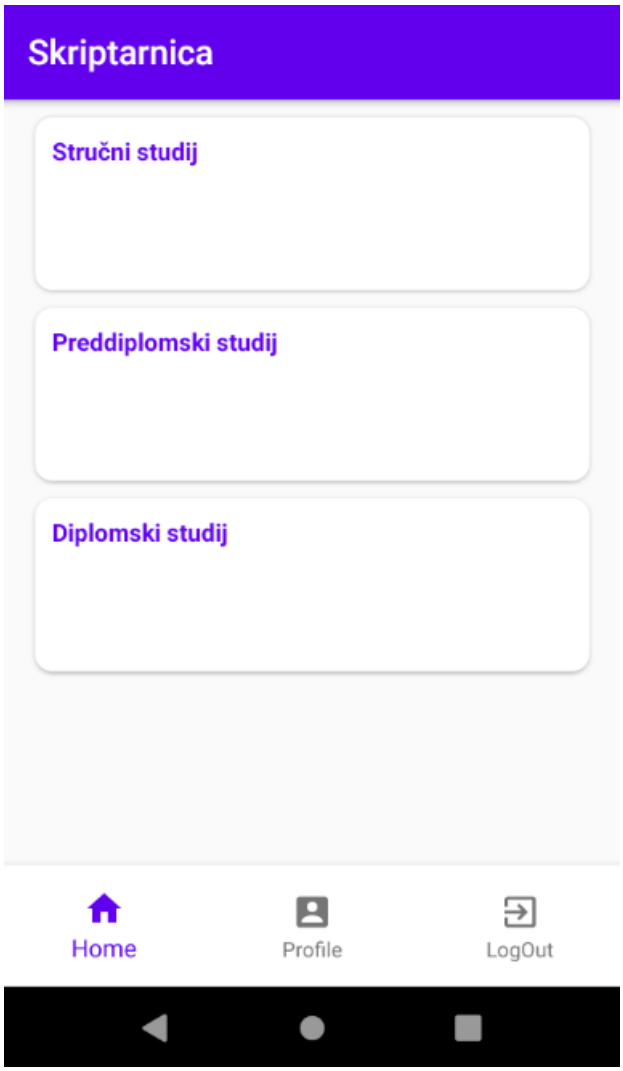
Otvorenjem ScriptFragmenta poziva se Retrofit zahtjev na Java Spring Boot API te se dohvaćaju nastavni materijali za odabrani smjer i godinu te predmet, način dohvaćanja će biti opisan u sljedećem poglavlju. U ScriptFragmentu je prisutan i gumb za spremanje nastavnih materijala na server, spremanje će također biti opisano u sljedećem poglavlju. Nastavni materijali koji su dohvaćeni se prikazuju u Recycler View listi te se pritiskom na njih poziva Retrofit zahtjev za skidanje materijala lokalno na uređaj. ScriptFragment može biti viđen u kôdu 5.16..

```
override fun setupUi() {
      Log.d("+++", selectedRepository.getSelectedStudy())
      Log.d("+++", selectedRepository.getSelectedSubject())
      scriptRecyclerView.layoutManager = LinearLayoutManager(context)
      scriptRecyclerView.adapter = adapter

      interactor.getFileByStudyAndYear(selectedRepository.getSelectedStudy
      (), selectedRepository.getSelectedSubject(), getScriptsCallback())
}
```

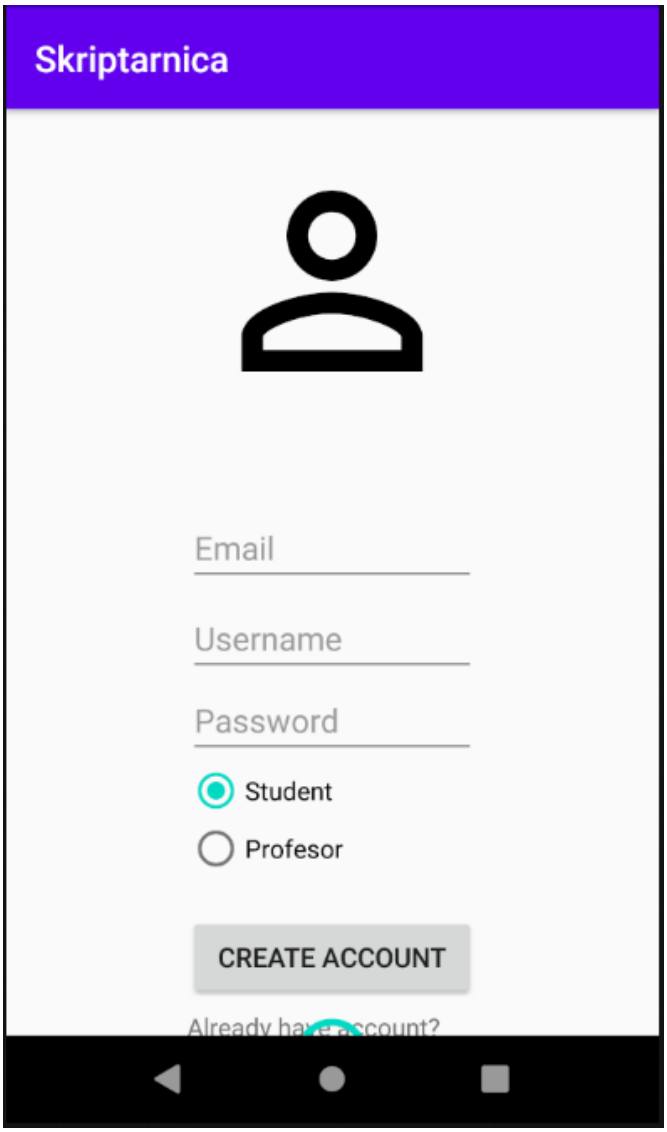
Programski kôd 5.16. – Kotlin kôd za ScriptFragment

U aplikaciji je implementira donja traka za navigaciju koja u sebi sadrži gumb za povratak na početnu stranicu, gumb za odjavljivanje iz aplikacije te gumb za profil korisnika. Kada korisnik uđe na svoj profil ima opciju za izmijeniti korisničke podatke kao što su email i šifra koji se koriste pri prijavi u aplikaciju. Korak unazad se ide pomoću ugrađenog android gumba za unazad. Izgled početne stranice možemo vidjeti u slici 5.1..



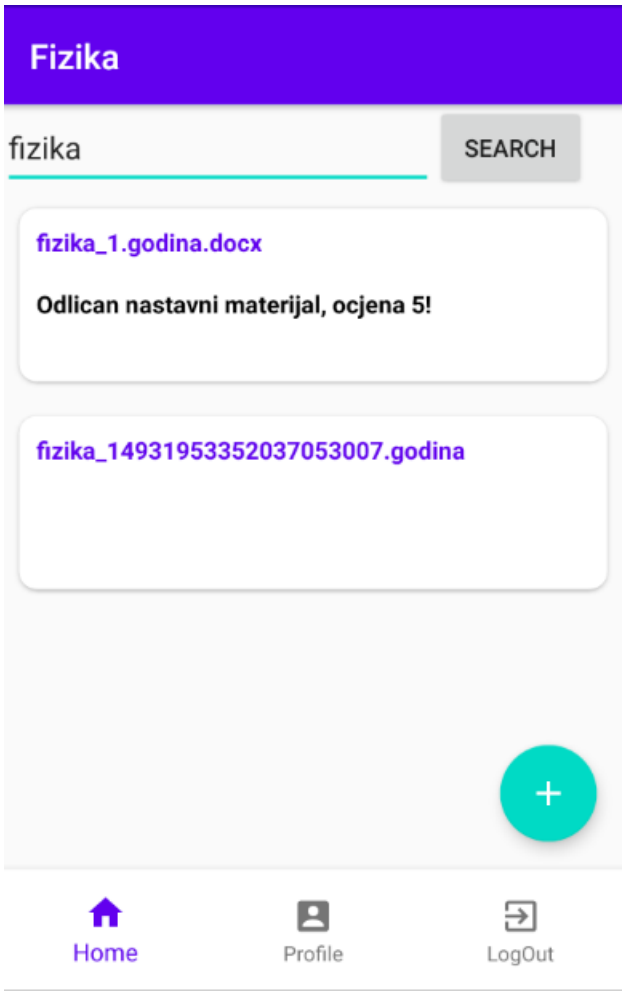
Slika 5.1. – Izgled početne stranice

Registracija se izvodi pomoću Registration stranice. Kako bi registracija bila uspješna mora se unijeti email koji nije u uporabi već. Također se pri registraciji treba odabrati vrsta profila (Nastavnik ili Student). Student ima prava na gledanje materijala, skidanje i dodavanje, dok nastavnik ima pravo na gledanje materijala, skidanje, dodavanje te komentiranje. Registracija se vrši pritiskom na gumb Create account. U slučaju da korisnik već ima račun može pritisnuti „Already have account?“. Izgled stranice za registraciju se može vidjeti na slici 5.2..



Slika 5.2. – Izgled stranice za registraciju

Nakon obavljene registracije i prijave, korisniku se prikaže stranica za odabir studija, pa smjera i godine, te predmeta i onda dobije na pregled nastavne materijale za odabrane uvjete. Ako je korisnik prijavljen kao student pritiskom na karticu sa željenim nastavnim materijalom pokreće se skidanje materijala i spremanje na uređaj. U podnožju stranice se nalazi + gumb na koji kada se pritisne bude otvoreno mjesto sa spremljenim datotekama na uređaju te na odabir datoteke pokreće se spremanje na Spring Boot servis. U slučaju da je korisnik prijavljen kao nastavnik pritiskom na nastavni materijal se otvara posebni pogled u kojem korisnik može napisati komentar ili skinuti nastavni materijal. Nastavni materijali su sortirani uzlazno po imenu te je moguće pretraživanje. Izgled stranice sa nastavnim materijalima može se vidjeti u slici 5.3..



Slika 5.3. – Izgled stranice sa nastavnim materijalima

5.3. Izrada rada s podacima pomoću Retrofita

Nakon što je izrađen izgled Android aplikacije i napravljen kostur za prikaz podataka, vrijeme je za napraviti spajanje na pozadinski Spring servis pomoću Retrofit biblioteke. Glavne zadatak je napraviti način na koji će se dohvaćati nastavni materijali za odabrani smjer, godinu i predmet, mogućnost skidanja odabranog nastavnog materijala te spremanje nastavnih materijala sa uređaja u pozadinsku bazu podataka. Napravljeno je Service Kotlin sučelje u kojem su predstavljeni zahtjevi za API poziv. U njemu je definirano sve što treba biti poslano u zahtjevu i preko koje Kotlin klase će odgovor biti vraćen. Za svaki API koji je razrađen na pozadinskoj strani potrebno je napraviti metodu koja će se pobrinuti za njega. U kôdu 5.17. se mogu vidjeti Retrofit Service sučelje.

```
interface ManageFileApiService {
      @Multipart
      @POST("/upload/{study}/{subject}")
      fun uploadFile(@Part file: MultipartBody.Part, @Path("study") study:
      String, @Path("subject") subject: String): Call<FileResponse>

      @GET("/download/{id}")
      fun downloadFile(@Path("id") id: String): Call<FileResponse>

      @GET("/files/{study}/{subject}")
      fun getFileByStudyAndYear(@Path("study") study: String,
      @Path("subject") subject: String): Call<List<BackendTask>>

      @POST("/files/comment/{fileId}")
      fun postFileComment(@Path("fileId") fileId: String, @Body comment:
      String): Call<BackendTask>

      @GET("/files/{study}/{subject}/{findParam}")
      fun getFilesByStudyAndYearFileNameContaining(@Path("study") study:
      String, @Path("subject") subject: String, @Path("findParam")
      findParam: String): Call<List<BackendTask>>
}
```

Programski kôd 5.17. – Kotlin kôd za Retrofit ManageFileApiService

Kako bi se API pozivi mogli napraviti, napravljen je način spajanja na pozadinski servis i to se odradi preko Retrofit graditelj kojem predamo klijenta (koji je OkHttpClient), dodamo URL na koji će se aplikacija spajati te na posljetku koji pretvarač će se koristi za prebacivanje iz JSON formata u Kotlin klasu. Nakon što je napravljeno sve što treba da bi se JSON podaci iz odgovora pozadinskog servisa mogli prikazati u ScriptFragmentu su napravljeni pozivi za Retrofit API metode (dohvaćanje svih nastavnih materijala po smjeru i godini te predmetu i spremanje u bazu) te pozivanje /download/id pozadinske metode preko Web preglednika. Kako bi se dohvatili nastavni materijali napravljen je getScriptsCallback() koji poziva getFilesByStudyAndYear() metodu. Poziv za povrat se sastoji od onFailure i onRespone metoda. U onResponse metodi se dohvaćeni podaci stavljaju u RecyclerView i prikazuju na ekranu. OnResponse metodu se može vidjeti u kôdu 5.18..

```
override fun onResponse(call: Call<List<BackendTask>>?, response:
Response<List<BackendTask>>) {
            if (response.isSuccessful) {
                  response.body()?.run {
                        adapter.setData(this)
                        }
                  }
      }
```

Programski kôd 5.18. – Kotlin kôd za callback onResponse()

Ovisno o tome je li korisnik prijavljen kao student ili nastavnik različiti se dijelovi koda pozivaju. Ako je student, na pritisak na nastavni materijal poziva se metoda za skidanje nastavnog materijala, a ako je nastavnik poziva se zasebni fragment u kojemu se nalaze metode za postavljanje komentara ili za skidanje skripte. OnClickListenere iz RegistrationFragmenta se može vidjeti u kôdu 5.19..

```
override fun setOnClickListeners() {
        commentButton.setOnClickListener {

            interactor.postFileComment(selectedFileIdRepository.getFileId(
            ), commentText.text.toString(), postCommentCallback())
        }
        downloadButton.setOnClickListener {
            val intent = Intent(Skriptarnica.instance,
            FileWebView::class.java)
            intent.putExtra("Url", "http://192.168.1.10:8080/download/" +
            selectedFileIdRepository.getFileId())
            startActivity(intent)
        }
    }
```

Programski kôd 5.19. – Kotlin kôd za RegistrationFragment onClickListeners

6. ZAKLJUČAK

Aplikacija za dijeljenje nastavnih materijala korisnicima pruža jednostavan način za razmjenu nastavnih materijala, te povratnu informaciju od nastavnika o kvaliteti materijala. Pomoću ove aplikacije studentima se olakšava način učenja pošto iz bilo kojeg mjesta gdje imaju pristup internetu mogu skinuti željene materijale i krenuti sa učenjem. Ako je student prijavljen u aplikaciji on ima mogućnost za pregled, skidanje i dodavanje materijala, dok nastavnik ima pregled, skidanje, dodavanje i komentiranje.

Izrada same aplikacije je zahtijevala znanje i istraživanje načina rada više tehnologija. U njoj su spojeni Android operacijski sustav i Java Spring Boot servis. U fazi planiranja aplikacije su određeni zahtjevi i način na koji će ti zahtjevi biti implementirani. Također, osmišljena je baza podataka, način spremanja nastavnih materijala te prijava korisnika u sustav. Izrađen je pozadinski servis na koji se spremaju materijali, ali također je izrađena Android aplikacija koja iskorištava taj servis. Korištene baze podataka su Firebase i SQL. Za testiranje Spring servisa korištena je Postman aplikacija.

LITERATURA

[1] https://play.google.com/store/apps/details?id=com.google.android.apps.classroom&hl=hr&gl=RU , rujan 2021.

[2] https://play.google.com/store/apps/details?id=com.moodle.moodlemobile , rujan 2021.

[3] InfoWorld, What is Kotlin? The Java alternative explained, https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html , srpanj 2020.

[4] Developers Android, Kotlin overview, https://developer.android.com/kotlin/overview, srpanj 2020.

[5] Kotlin, FAQ, https://kotlinlang.org/docs/reference/faq.html, srpanj 2020.

[6] Developers Android, Develop Android Apps with Kotlin, https://developer.android.com/kotlin, srpanj 2020.

[7] Edpresso, What is Firebase?, https://www.educative.io/edpresso/what-is-firebase, srpanj 2020.

[8] Wikipedia, Firebase, https://en.wikipedia.org/wiki/Firebase, srpanj 2020.

[9] Stackoverflow, https://stackoverflow.com/questions/45740671/facebook-authentication-not-working-with-firebase-ui-in-android, srpanj 2020.

[10] Android Authority, What is Android? Here's everything you need to know, https://www.androidauthority.com/what-is-android-328076/, srpanj 2020.

[11] JavaTpoint, History of Android, https://www.javatpoint.com/android-history-and-versions, srpanj 2020.

[12] JavaTpoint, Android Architecture, https://www.javatpoint.com/android-software-stack , srpanj 2020.

[13] JavaTpoint, Android Activity Lifecycle, https://www.javatpoint.com/android-life-cycle-of-activity, srpanj 2020.

[14] JavaTpoint, Android Core Building Blocks, https://www.javatpoint.com/android-core-building-blocks , srpanj 2020.

[15] Arnold, Ken, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[16] Mane, Dashrath, Namrata Ojha, and Ketaki Chitnis. "The spring framework: An open source java platform for developing robust java applications." *International Journal of Innovative Technology and Exploring Engineering* 3.2, 2013.

SAŽETAK

U završnom radu razvijena je aplikacija za dijeljenje nastavnih materijala, koja korisnicima omogućava razmjenu nastavnih materijala na lagan način. Aplikacija je osmišljena za studente koji će pomoću nje razmjenjivati nastavne materijale, ali će i nastavnici za pojedini predmet moći komentirati materijale. U teorijskom dijelu rada je opisan način na koji će aplikacija biti implementirana, te su opisane tehnologije koje će se koristiti. Praktični dio rada se sastoji od same implementacije aplikacije i utvrđivanja korisničkih zahtjeva te implementacije baze podataka. Za implementaciju aplikacije korišteni su Kotlin, Java Spring Boot, Firebase te SQL.

Ključne riječi: aplikacija, korisnički zahtjevi, nastavni materijali, tehnologije

ABSTRACT

Android mobile application for sharing teaching materials

In bachelor thesis an aplication for sharing of teaching materials was made which gives useres means to share teaching materials in easy way. Aplication was designed for students which will use it to share materials, but proffesors will be able to comment on materials also. In theory part of thesis the way in which the aplication will be implemented is described, also tehnologys that will be used are also described. Practical part of thesis is made of implementation of aplication but also from defining user requirements and implementation of database. For implementation of aplication Kotlin, Java Spring Boot, Firebase and SQL were used.

Keywords: aplication, teaching materials, technologies, user requirements

ŽIVOTOPIS

Marko Milić rođen je 6. svibnja 1995. godine u Osijeku. Od 2002. do 2010. pohađa Osnovnu školu Ljudevita Gaja u Osijeku. Godine 2010. upisuje Prirodoslovno-matematičku gimnaziju u Osijeku koju završava 2014. godine s položenim svim ispitima državne mature. Godine 2017. se upisuje na Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na stručni studij Informatike.

Marko Milić