Koncept razvoja mobilnih aplikacija korišćenjem rešenja u oblaku na primeru RAF Network mobilne aplikacije

Luka Petrović, dr Nemanja Radosavljević, docent i dr Mirjana Radivojević, profesor

***Sadržaj* —Rad istražuje razvoj mobilnih aplikacija korišćenjem rešenja u oblaku na primeru *RAF Network* aplikacije. Predstavljene su korišćene tehnologije i metodologije, uključujući *Flutter* za frontend i *Django* za backend, uz infrastrukturu na *DigitalOcean* platformi sa *Firebase PaaS* rešenjima. Opisan je koncept dokerizacije i upotrebe kontejnera, koji omogućavaju skalabilnost i efikasno upravljanje resursima. Kroz implementaciju *Traefik* reverznog proksija i orkestraciju kontejnera, aplikacija pruža fleksibilnost i visoku dostupnost. Rad obuhvata potencijalne pravce daljeg razvoja, kao što su integracija sa studentskim servisima i unapređenje komunikacije između studenata i profesora.**

***Ključne reči* — aplikacija, Android, cloud, Django, Docker, Firebase, Flutter, iOS, kontejner, mobilni uređaji, računarstvo u oblaku, REST API, skalabilnost**

I. Uvod

UBRZAN tehnološki napredak i sveprisutnost mobilnih uređaja transformisali su način na koji korisnici pristupaju informacijama i uslugama. Mobilne aplikacije postale su neophodne u raznim industrijama. Porastom očekivanja korisnika, raste i kompleksnost razvoja i održavanja aplikacija. Jedan od ključnih faktora koji omogućava savremeni razvoj mobilnih aplikacija jeste korišćenje tehnologija računarstva u oblaku. One pružaju programerima da razvijaju, distribuiraju i održavaju mobilne aplikacije, pružajući skalabilnost, fleksibilnost, pouzdanost i smanjenje troškova. U radu će biti istražen koncept razvoja aplikacija korišćenjem rešenja računarstva u oblaku, kroz studiju slučaja mobilne aplikaciju *RAF Network*.

L. P. Autor, Računarski fakultet, Srbija (e-mail: petrovic.luka99@gmail.com).
N. R. Autor, Računarski fakultet, Srbija (e-mail: nradosavljevic@raf.rs).
M. R. Autor, Računarski fakultet, Srbija (e-mail: mradivojevic@raf.rs).

Mobilna aplikacija *RAF Network* namenjena je aktivnim i budućim studentima, alumni članovima i svim zainteresovanim za dešavanja na Računarskom fakultetu. Aplikacija obuhvata širok spektar funkcionalnosti, uključujući pregled alumnista fakulteta i kompanija u kojima bivši studenti rade, personalizovani raspored predavanja, pregled kolokvijumskih nedelja i ispitnih rokova, te objave članova sa aktivnim diskusijama. Dobro osmišljena alumni mreža može poslužiti kao most između sadašnjih studenata i bivših studenata, podstičući veze koje povećavaju profesionalne mogućnosti i održavaju alumniste angažovane u aktivnostima institucije [1]. Ova aplikacija razvijena je korišćenjem *Flutter* framework-a za klijentsku stranu i *Django web framework*-a za serversku stranu, sa infrastrukturom postavljenom na *DigitalOcean* platformi za računarstvo u oblaku. Korišćenjem *REST API*-a, izloženog putem *Traefik* reverznog proksija, omogućena je efikasna komunikacija između mobilne aplikacije i web servisa. Ovaj pristup pruža visoku skalabilnost i fleksibilnost, omogućavajući lako proširenje aplikacije dodatnim funkcionalnostima kao što su integracija sa studentskim servisom za prijavu ispita, pregled rezultata kolokvijuma i ispita kao i komunikaciju sa profesorima. U ovom radu će biti predstavljene prednosti korišćenja tehnologija računarstva u oblaku pri razvoju mobilnih aplikacija sa gledišta skalabilnosti, fleksibilnosti i pouzdanosti.

II. PREGLED RAF NETWORK MOBILNE APLIKACIJE

Mobilna aplikacija *RAF Network* razvijena je sa ciljem da poveže aktivne i buduće studente, alumniste i sve zainteresovane za dešavanja na Računarskom fakultetu. Aplikacija pruža niz funkcionalnosti koje nude pristup informacijama i interakciju među korisnicima.

A. Ulaz i autentifikacija

Na ulazu u aplikaciju, korisnicima je omogućena prijava putem *Google Sign-in* opcije. Ovaj metod autentifikacije je jednostavan za korisnike i pruža visok nivo sigurnosti. Nakon prijave, aplikacija razlikuje korisnike na osnovu njihove email adrese. Korisnici koji se prijavljuju sa fakultetskom email adresom (@raf.rs) prepoznati su kao studenti dok su korisnici koji koriste druge email adrese prepoznati kao gosti.

B. Funkcionalnosti za gostujuće korisnike

Gostujući korisnici imaju ograničen pristup funkcionalnostima aplikacije. Mogu pregledati forumske objave koje su postavili drugi korisnici, ali nemaju mogućnost interakcije poput dodavanja sadržaja ili komentarisanja. Informacije o dešavanjima na fakultetu, uključujući događaje, seminare i druge

aktivnosti, takođe su im dostupne. Osim toga, gostujući korisnici mogu istraživati profile bivših studenata, saznati gde su radili, koje su smerove završili i dobiti uvid u druge relevantne podatke koji su prikazani na Slici 1.

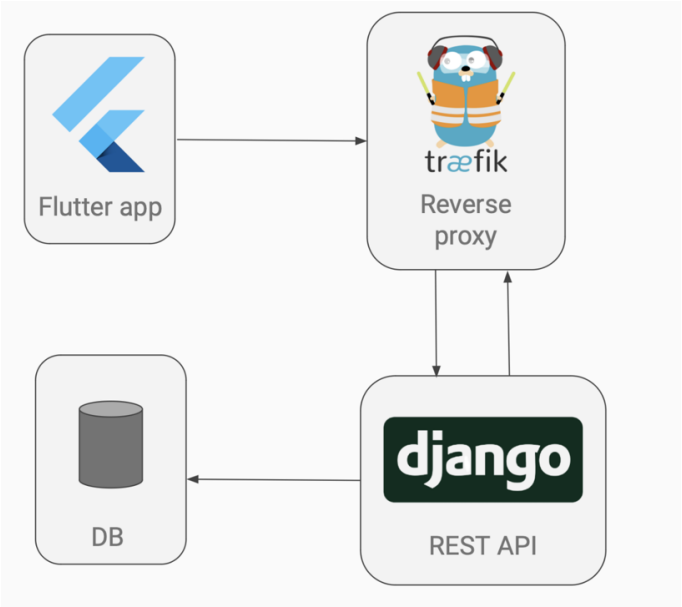
Sl. 1. Pregled alumni studenta u *RAF Network* aplikaciji

C. Funkcionalnosti za studente

Studenti, zahvaljujući fakultetskoj email adresi, imaju pristup dodatnim funkcionalnostima aplikacije, koje im omogućavaju veću interakciju i personalizaciju unutar platforme. Jedna od ključnih funkcionalnosti je mogućnost učestvovanja u diskusijama, gde studenti mogu postavljati objave na forumu i aktivno se uključivati u diskusije sa drugim korisnicima. Takođe, svakom studentu je dostupan personalizovani raspored predavanja, čime im se olakšava praćenje obaveza tokom semestra. Pored toga, studenti mogu pregledati termine kolokvijumskih nedelja i ispitnih rokova, što im pomaže u planiranju i organizaciji akademskih obaveza.

D. Odabir tehnologija i arhitektura sistema

RAF Network aplikacija koristi različite tehnologije kako bi pružila kvalitetno softversko rešenje. Frontend je razvijen korišćenjem *Flutter framework*-a otvorenog koda, razvijenog od strane *Google*-a, koji omogućava kreiranje nativnih mobilnih aplikacija za *Android* i *iOS* iz jedinstvenog koda. Zasnovan na programskom jeziku *Dart, Flutter* koristi *JIT* kompajliranje za razvoj i *AOT* kompajliranje za optimizovane performanse. Njegova arhitektura omogućava programerima da lako kreiraju prilagođene korisničke interfejse koristeći koncept vidžeta, što ga čini popularnim izborom za razvoj mobilnih aplikacija. *Flutter*-ova sposobnost da smanji vreme razvoja nudeći jednu bazu koda za *Android* i *iOS*, zajedno sa svojom pojednostavljenom strukturom, učinila ga je dobrim izborom tehnologije, posebno imajući u vidu važnost održavanja doslednog korisničkog iskustva na različitim platformama [2]. Backend deo sistema čine web servisi razvijeni u *Django web framework*-u, hostovani na *DigitalOcean Droplet* virtuelnoj mašini. *Django* je web framework visokih performansi zasnovan na programskom jeziku *Python*, dizajniran za brz razvoj složenih web aplikacija. Pruža širok spektar ugrađenih funkcionalnosti kao što su *ORM (Object Relational Mapper)* za rad sa bazama podataka, autentifikacija korisnika i zaštita od bezbednosnih pretnji. Zasnovan na *Model-View-Template* arhitekturi, *Django* omogućava jasno razdvajanje logike, prikaza i upravljanja podacima, čime se pojednostavljuje održavanje i proširenje aplikacija. Za projekat je iskorišćen i *Django REST Framework*, koji omogućava lako kreiranje i upravljanje *REST API*-ima za efikasnu komunikaciju sa frontendom. Komunikacija između klijentske i serverske strane ostvaruje se preko *REST API*-a, izloženog putem *Traefik* reverznog proksija, što omogućava sigurnu i efikasnu razmenu podataka. Na Slici 2. prikazana je arhitektura *RAF Network* sistema, uključujući mobilnu aplikaciju, reverzni proksi, web servis i bazu podataka, postavljeni na *DigitalOcean* platformi za računarstvo u oblaku.



Sl. 2. Arhitektura *RAF Network* sistema

III. POTENCIJALNI PRAVCI DALJEG RAZVOJA

Postoje razni pravci u kojima bi mobilna aplikacija *RAF Network* mogla dodatno da se proširi, pružajući korisnicima još veći spektar funkcionalnosti i unapređujući njihovo iskustvo. U nastavku su navedena neka od potencijalnih rešenja za unapređenje platforme.

A. Integracija sa studentskim servisima

Jedan od ključnih pravaca u daljem razvoju aplikacije je integracija sa postojećim studentskim servisima fakulteta. Ova integracija bi studentima omogućila pristup različitim akademskim servisima putem same aplikacije, čime bi se pojednostavili administrativni procesi i poboljšalo korisničko iskustvo. Na primer, funkcionalnost za prijavu ispita omogućila bi studentima da prijave ispite direktno kroz aplikaciju, eliminišući potrebu za korišćenjem drugih platformi. Takođe, integracija sa sistemom za ocenjivanje pružila bi mogućnost studentima da odmah po objavi pregledaju rezultate kolokvijuma i ispita, čime bi se značajno ubrzao pristup važnim informacijama. Sistem za anketiranje studenata o kvalitetu nastave je trenutno poseban servis koji bi isto mogao biti integrisan u samu aplikaciju.

B. Komunikacija sa profesorima i asistentima

Unapređenje komunikacije između studenata i profesora moglo bi značajno doprineti efikasnosti učenja i rešavanju problema. *RAF Network* aplikacija može integrisati funkcionalnosti koje omogućavaju direktnu i brzu komunikaciju. Na primer, sistem za slanje poruka omogućio bi studentima da direktno kontaktiraju profesore radi postavljanja pitanja i dogovaranja konsultacija, dok bi profesori mogli koristiti ovu funkcionalnost za slanje obaveštenja o predavanjima, ispitima i drugim važnim aktivnostima. Pored toga, implementacija alata za virtuelne konsultacije omogućila bi studentima da zakazuju i prisustvuju sastancima sa profesorima putem aplikacije, čime bi se smanjila potreba za fizičkim prisustvom i olakšala dostupnost profesora.

C. Dodatne informacije o alumni članovima

Proširenje funkcionalnosti aplikacije može uključivati dodavanje detaljnijih informacija o alumni članovima, što bi omogućilo efikasnije povezivanje i umrežavanje. Na primer, studenti bi pored informacija o karijerama bivših studenata, imali pristup i pozicijama koje trenutno zauzimaju i projekte na kojima rade. Ova funkcionalnost pružila bi korisnicima uvid u potencijalne karijerne puteve. Takođe, aplikacija bi omogućila studentima i gostima da se povežu sa alumni članovima, što bi otvorilo mogućnosti za savete, mentorstvo ili profesionalne prilike, unapređujući povezivanje unutar zajednice.

IV. Osnove tehnologije računarstva u oblaku

Tehnologija računarstva u oblaku predstavlja fundamentalni pomak u načinu na koji se računarski resursi koriste, omogućavajući organizacijama i pojedincima pristup snažnim računarskim kapacitetima putem interneta. Ova tehnologija nudi neprevaziđenu fleksibilnost, skalabilnost i pouzdanost, što je čini ključnom komponentom u razvoju savremenih mobilnih aplikacija. Računarstvo u oblaku je distribuirano okruženje koje omogućava organizacijama da pristupaju resursima na daljinu i upravljaju njima uz visoku skalabilnost, fleksibilnost i pouzdanost, dok istovremeno imaju koristi od isplativih modela plaćanja po utrošku [3]. Takođe, omogućava organizacijama da se usmere na razvoj aplikacija i servisa, dok oblak obezbeđuje skalabilne resurse za podršku njihovom rastu i poslovnim zahtevima. Na taj način, računarstvo u oblaku postaje suštinska tehnologija za unapređenje efikasnosti i inovacija u IT sektoru.

A. Prednosti tehnologija računarstva u oblaku

Računarstvo u oblaku pruža visok nivo fleksibilnosti programerima, omogućavajući brzo dodavanje novih funkcionalnosti, testiranje različitih

verzija aplikacija i agilno reagovanje na povratne informacije korisnika. Ova fleksibilnost podstiče brže inovacije i prilagođavanje aplikacija zahtevima tržišta, što je od suštinske važnosti u dinamičnom i konkurentnom poslovnom okruženju jer time omogućava ubrzanje razvojnog ciklusa i brže uvođenje novih funkcionalnosti.

Pouzdanost predstavlja ključnu komponentu računarstva u oblaku. Provajderi obezbeđuju visoku dostupnost svojih servisa kroz upotrebu redundantnih sistema [3] i automatskih mehanizama za oporavak od katastrofa. Ovi mehanizmi osiguravaju da aplikacije ostanu dostupne korisnicima 24/7, smanjujući rizik od gubitka podataka i prekida u radu. Uz visoku dostupnost, organizacije mogu garantovati stabilno iskustvo korisnicima.

Korišćenje tehnologija računarstva u oblaku može značajno smanjiti troškove razvoja i održavanja aplikacija. Organizacije plaćaju samo za resurse koje koriste, eliminišući potrebu za velikim početnim ulaganjima u hardversku i softversku infrastrukturu. Ova ekonomičnost omogućava efikasnije upravljanje budžetom, čime se oslobađaju sredstva za dalji razvoj i inovacije. Troškovi se tako optimizuju prema stvarnim potrebama aplikacija, omogućavajući organizacijama da povećaju profitabilnost i fleksibilnost u raspodeli resursa.

B. Tipovi usluga računarstva u oblaku

Usluge u oblaku se obično dele na tri glavne kategorije: Infrastruktura kao servis (*IaaS*), Platforma kao servis (*PaaS*) i Softver kao servis (*SaaS*), svaka od kojih pruža različite nivoe kontrole i funkcionalnosti u zavisnosti od potreba korisnika [3].

IaaS model omogućava korisnicima pristup osnovnim infrastrukturnim resursima kao što su virtuelne mašine, skladištenje podataka i mrežna infrastruktura. Korisnici imaju potpunu kontrolu nad konfiguracijom servera, operativnim sistemima i aplikacijama koje pokreću, ali ne upravljaju fizičkom infrastrukturom, jer o tome brine provajder oblaka. Ovaj model je posebno koristan za organizacije koje žele da prilagode svoje serverske i mrežne resurse specifičnim potrebama. Primeri *IaaS* platformi uključuju *Amazon Web Services (AWS)*, *Microsoft Azure* i *Google Cloud Platform*, koje omogućavaju fleksibilno korišćenje resursa uz plaćanje samo onoga što se zaista koristi.

PaaS platforma obezbeđuje kompletno okruženje za razvoj, testiranje i implementaciju aplikacija, gde se infrastrukturom u potpunosti upravlja od strane provajdera. Ovo omogućava programerima da se fokusiraju isključivo na razvoj i logiku aplikacija, bez brige oko upravljanja serverima, bazama podataka ili mrežnim konfiguracijama. Glavna prednost *PaaS* modela je što pojednostavljuje i ubrzava proces razvoja, omogućavajući timovima da brže

lansiraju nove funkcionalnosti i poboljšanja. Primer *PaaS* rešenja je *Google App Engine*, koji omogućava programerima da postave svoje aplikacije na *Google*-ovu infrastrukturu, dok provajder automatski upravlja svim potrebnim resursima kao što su balansiranje opterećenja, skaliranje aplikacija i praćenje njihovog rada.

SaaS model omogućava korisnicima da pristupaju gotovim softverskim rešenjima putem interneta, bez potrebe za instalacijom, održavanjem ili upravljanjem infrastrukturom. Ovaj model je pogodan za organizacije i pojedince koji žele da koriste softverske aplikacije sa minimalnim tehničkim znanjem i uz brz pristup. Primeri *SaaS* aplikacija uključuju *Google Workspace* i *Microsoft 365*, koje omogućavaju korisnicima da rade u aplikacijama za produktivnost kao što su tekstualni procesori, tabele i kolaborativni alati, bez potrebe za instalacijom ili lokalnim serverima.

V. IMPLEMENTACIJA RAF NETWORK APLIKACIJE U OBLAKU

Implementacija *RAF Network* aplikacije kroz tehnologije računarstva u oblaku je ključna za obezbeđivanje skalabilnosti, fleksibilnosti i pouzdanosti potrebnih za savremene mobilne aplikacije. Za projekat je primarno odabran tip usluge infrastruktura kao servis uz određene elemente *PaaS*, a kao provajder oblaka platforma *DigitalOcean*. *DigitalOcean* nudi niz usluga za pojednostavljeno upravljanje i razvoj aplikacija u oblaku. Upravljane baze podataka, poput *PostgreSQL*, *MySQL* i *Redis*, omogućavaju lako postavljanje, skaliranje i upravljanje, bez potrebe za ručnim održavanjem, uz automatske rezervne kopije i otpornost na greške. Za skladištenje velikih količina nestrukturiranih podataka, kao što su slike i video snimci, *DigitalOcean* nudi servis *Spaces*, koji je skalabilan i podržava *CDN* za bržu isporuku sadržaja. Takođe, platforma za orkestraciju kontejnera, *DigitalOcean Managed Kubernetes*, omogućava automatizovano upravljanje *Docker* kontejnerima, skaliranje i lakše rukovanje mikroservisima.

Docker predstavlja platformu za razvoj, isporuku i pokretanje aplikacija unutar kontejnera koji obezbeđuju izolovana okruženja za softver, omogućavajući doslednost u različitim razvojnim, testnim i produkcionim okruženjima [4]. U projektu, on je iskorišćen za sinhronizovanje svih komponenti sistema, uključujući *Django* web servis, *Redis* za keširanje i *Traefik* kao reverzni proksi. Korišćenjem *Traefik*-a, svaki servis je izolovan u sopstvenom kontejneru, što omogućava da aplikacija bude modularna i lako skalabilna. Izolacija servisa olakšava identifikaciju i rešavanje problema, jer svaki kontejner funkcioniše kao nezavisna jedinica sa sopstvenim resursima i okruženjem. *Docker Compose* je alatka koja dopunjuje *Docker* tako što omogućava orkestraciju više *Docker* kontejnera kao jedne aplikacije. U

projektu je korišćen za definisanje i upravljanje svim servisima aplikacija, uključujući njihove međusobne zavisnosti i mrežne konfiguracije. Dokerizacija projekta podrazumeva kontejnerizaciju svih njegovih komponenti kako bi se obezbedila konzistentnost, prenosivost i lakše upravljanje aplikacijom. Proces je uključivao više koraka.

A. *Dokerizacija web servisa*

Django web servis je dokerizovan da bi se obezbedilo jednostavno pokretanje i skalabilnost. Dokerizacija se realizuje kreiranjem *Dockerfile*-a koji definiše osnovnu sliku, instalira potrebne zavisnosti i pokreće aplikaciju.

Na Slici 3. je prikazan *Dockerfile* koji se koristi za kreiranje slike web servisa aplikacije time što instalira potrebne biblioteke i kopira izvorni kod u kontejner. Kada je servis upakovan u sliku, vrlo lako se može horizontalno skalirati kroz veliki broj kontejnera koji rade paralelno, što omogućava visoku dostupnost sistema.

```
FROM python:3.9.5-alpine

# set work directory
WORKDIR /usr/src/app

# set environment variables
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apk update \
    && apk add gcc python3-dev musl-dev libffi-dev g++

# install dependencies
RUN pip install --upgrade pip
COPY ./requirements.txt /usr/src/app/requirements.txt
RUN pip install -r /usr/src/app/requirements.txt

# copy project
COPY . /usr/src/app/
```

Sl. 3. *Dockerfile* korišćen za *RAF Network Django* web servis

B. *Konfiguracija virtuelne mašine*

Sledeći korak u implementaciji bilo je iznajmljivanje virtuelne mašine na platformi *DigitalOcean*. *DigitalOcean* je izabran zbog svoje jednostavnosti, pouzdanosti i fleksibilnosti. Na platformi kreirana je nova virtuelna mašina (*Droplet*) sa Ubuntu operativnim sistemom, sa odgovarajućim resursima potrebnim za pokretanje aplikacije. Domen *raf.code-dream.com* je konfigurisan da ukazuje na IP adresu novoiznajmljene mašine.

C. Konfiguracija Traefik reverznog proksija uz Let's Encrypt

Traefik je izabran za implementaciju reverznog proksija, koji je poznat po svojoj jednostavnosti, fleksibilnosti i mogućnosti automatskog prilagođavanja promenama u infrastrukturi. *Traefik* je posebno pogodan za okruženja u kojima se koristi *Docker*, jer ima mogućnost da automatski detektuje nove servise koji rade unutar *Docker* mreže i automatski ih integriše u konfiguraciju, bez potrebe za ručnom intervencijom. Jedna od sposobnosti svakog reverznog proksija jeste da efikasno balansira opterećenje između više instanci aplikacije, osiguravajući da su svi zahtevi ravnomerno raspoređeni i da nema zagušenja u sistemu [5]. Ova funkcionalnost je posebno važna za aplikacije koje očekuju veliki broj korisnika ili gde se očekuje nepredvidivo povećanje saobraćaja. Pored toga, *Traefik* omogućava jednostavno upravljanje *SSL/TLS* sertifikatima kroz integraciju sa *Let's Encrypt* sertifikacionim telom. Ova integracija omogućava automatsko dobijanje i obnavljanje *SSL* sertifikata, čime se obezbeđuje sigurnost komunikacije između korisnika i servera bez potrebe za ručnim ažuriranjem sertifikata. Ovo značajno smanjuje rizik od narušavanja bezbednosti do kojih može doći zbog zastarelih ili nevažećih sertifikata. *Traefik* takođe nudi napredne funkcije kao što su metrike u realnom vremenu koje omogućavaju administratorima aplikacija da prate performanse i identifikuju potencijalne probleme pre nego što počnu da utiču na korisnike.

D. Postavka docker-compose.yml datoteke

Docker Compose koristi jednostavnu *YAML* datoteku da definiše sve usluge potrebne za pokretanje aplikacije, uključujući informacije o mrežama, skladištima i portovima. To pojednostavljuje proces pokretanja aplikacije, jer omogućava pokretanje svih servisa jednom komandom. Ovo je posebno korisno u razvoju i testiranju, gde se čitava okruženja mogu lako kreirati i uništiti, obezbeđujući doslednost i brzinu u razvoju. Takođe, *Docker Compose* olakšava automatizaciju procesa implementacije u produkcionim okruženjima, gde se svi servisi mogu automatski pokrenuti i konfigurisati prema unapred definisanim podešavanjima. Dodatne instance servisa mogu se brzo pokrenuti, omogućavajući horizontalno skaliranje i visoku dostupnost aplikacija.

E. Implementacija privatne mreže

Django servis je konfigurisan unutar privatne *Docker* mreže koja takođe uključuje servis *Redis* keširanja. *Redis* je dostupan samo u okviru privatne mreže, što dodatno povećava bezbednost sistema. Ova konfiguracija omogućava *Django* servisu da komunicira sa *Redis*-om preko interne mreže, dok je *Traefik* odgovoran za izlaganje *Django* servisa internetu preko bezbednog *HTTPS* protokola. Ova mrežna arhitektura osigurava da su svi

kritični servisi izolovani i zaštićeni, dok je javni pristup ograničen na ono što je neophodno da bi aplikacija funkcionisala. Na ovaj način se postiže visok nivo sigurnosti i pouzdanosti čitavog sistema. Na Slici 4. je prikazan *docker-compose.yml* za postavku *Traefik* reverznog proksija dok je na Slici 5. prikazana postavka *docker-compose.yml* datoteke za *Django* i prateće servise sa implementiranom privatnom *Docker* mrežom.

```yaml
version: "3.9"

services:

  traefik:
    build:
      context: .
      dockerfile: Dockerfile.traefik
    restart: unless-stopped
    networks:
      - web-net
    command:
      - "--metrics.prometheus=true"
      - "--metrics.prometheus.buckets=0.1,0.3,1.2,5.0"
      - "--providers.docker=true"
      - "--providers.docker.watch"
    ports:
      - 80:80
      - 443:443
      - 8090:8090
    volumes:
      - "./traefik-public-certificates:/certificates"
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
    labels:
      - "traefik.enable=true"
      - "traefik.docker.network=web-net"

volumes:
  traefik-public-certificates:

networks:
  web-net:
    driver: bridge
    name: web-net
```

Sl. 4. *docker-compose.yml* konfiguracija za *Traefik* reverzni proksi

```yaml
version: "3.9"

services:

  alumninet:
    build:
      dockerfile: Dockerfile
      context: ./alumninet
    container_name: alumninet
    command: sh -c "python manage.py runserver --nostatic 0.0.0.0:80"
    env_file:
      - ./env
    ports:
      - '8070:80'
    restart: unless-stopped
    networks:
      - internal
      - traefik_proxy
    volumes:
      - ./alumninet:/usr/src/app
      - ./db:/usr/src/app/db
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.alumninet.rule=Host(`raf.code-dream.com`)"
      - "traefik.http.routers.alumninet.tls=true"
      - "traefik.http.routers.alumninet.entrypoints=websecure"
      - "traefik.http.routers.alumninet.tls.certresolver=letsencrypt"
      - "traefik.http.services.alumninet.loadbalancer.server.port=80"
      - "traefik.priority=101"
      - "traefik.docker.network=traefik_proxy"

  raf_service_consumer:
    build:
      dockerfile: Dockerfile
      context: ./alumninet
    command: sh -c "python manage.py consume_raf_service"
    env_file:
      - ./env
    networks:
      - internal
    volumes:
      - ./alumninet:/usr/src/app

  rabbitmq:
    image: "rabbitmq:management"
    restart: unless-stopped
    volumes:
      - ./rabbitmq:/var/lib/rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    networks:
      - internal

  redis:
    restart: unless-stopped
    image: redis:7.0.5-alpine
    ports:
    - '6379:6379'
    networks:
      - internal

networks:
  internal:
    driver: bridge
    name: internal
  traefik_proxy:
    external: true
```

Sl. 5. *docker-compose.yml* konfiguracija za *Django* i prateće servise

VI. BEZBEDNOST I ZAŠTITA PODATAKA U OBLAKU

Bezbednost i zaštita podataka su ključni aspekti u implementaciji rešenja koja koriste tehnologije računarstva u oblaku, posebno kada su u pitanju mobilne aplikacije koje obrađuju osetljive korisničke i biometrijske podatke. Jedan od mehanizama za obezbeđivanje bezbednosti podataka u oblaku jeste enkripcija [6]. Enkripcija podataka osigurava da su podaci zaštićeni od neovlašćenog pristupa tokom prenosa i skladištenja. U slučaju aplikacije *RAF Network* koristi se *HTTPS* protokol koji omogućava bezbednu komunikaciju između klijenata i servera. Ova konfiguracija omogućava enkripciju svih podataka koji se prenose između korisnika i aplikacije, obezbeđujući privatnost i integritet podataka.

Efikasna kontrola pristupa je takođe ključna za zaštitu podataka. Aplikacija *RAF Network* koristi *Google* prijavljivanje za autentifikaciju korisnika, što omogućava visoku sigurnost i jednostavnost korišćenja. Korisnici se prijavljuju koristeći svoje *Google* naloge, a aplikacija razlikuje korisnike na osnovu njihovih adresa e-pošte. Korisnici sa fakultetskim adresama (@raf.rs) imaju pristup dodatnim funkcionalnostima, dok su gosti ograničeni na osnovne funkcionalnosti. Ovaj pristup osigurava da samo ovlašćeni korisnici mogu pristupiti određenim delovima aplikacije.

Obezbeđivanje podataka u slučaju gubitka ili katastrofe je od vitalnog značaja. Platforme oblaka kao što je *DigitalOcean* pružaju mehanizme za automatsko pravljenje rezervnih kopija podataka i brzi oporavak u slučaju gubitka ili kvara sistema. Primena redovnih rezervnih kopija obezbeđuje da se podaci mogu brzo vratiti u slučaju nepredviđenih problema.

VII. FIREBASE REŠENJA U OBLAKU ZA MOBILNE APLIKACIJE

Firebase, razvijen od strane kompanije *Google* i implementiran kao *PaaS*, pruža niz integrisanih alata za mobilne i web aplikacije. Ova rešenja su deo tehnologija računarstva u oblaku koje omogućavaju programerima da prate i optimizuju aplikacije na osnovu podataka prikupljenih u realnom vremenu, što je ključno za poboljšanje korisničkog iskustva i performansi aplikacija. Većina *Firebase* servisa se svodi na jednostavnu integraciju u klijentsku aplikaciju kroz *plug-and-play* sistem unapred razvijenih *SDK*-ova za različite platforme.

A. Firebase Analytics

Firebase Analytics je besplatan servis namenjen praćenju interakcija korisnika sa aplikacijom. Ovaj alat omogućava programerima da prate ključne metrike, kao što su broj aktivnih korisnika, trajanje sesija, demografske podatke i ponašanje korisnika u okviru aplikacije. Svi podaci su dostupni kroz interaktivnu kontrolnu tablu koja omogućava detaljan uvid u način korišćenja

aplikacije od strane korisnika. *Firebase Analytics* takođe omogućava praćenje specifičnih događaja unutar aplikacije, poput kupovina, prijava ili preuzimanja datoteka, što pruža mogućnost dubinske analize korisničkog ponašanja. Analizom podataka o ponašanju korisnika, programeri mogu da poboljšaju uputstva i korisničke interfejse kako bi unapredili upotrebljivost, posebno za nove korisnike, obezbeđujući efikasno rešavanje uobičajenih grešaka i poboljšavajući celokupno korisničko iskustvo [7]. Podržana je integracija sa više platformi, uključujući *iOS*, *Android*, *Web*, *Flutter*, *Unity* i *C++*.

B. Firebase Performance Monitoring

Firebase Performance Monitoring je servis koji omogućava praćenje performansi mobilnih aplikacija u realnom vremenu. Ovaj alat prikuplja podatke o ključnim aspektima aplikacije, kao što su vreme učitavanja ekrana, brzina mrežnih zahteva i korišćenje resursa. Programeri mogu postaviti pragove performansi i definisati metrike koje omogućavaju identifikaciju zastoja ili potencijalnih problema u aplikaciji pre nego što ih krajnji korisnici uoče. Ova usluga omogućava brzu identifikaciju problema kao što su spor odziv aplikacije, kašnjenja u učitavanju ili nestabilnost mreže, čime pruža programerima mogućnost za pravovremenu optimizaciju i unapređenje performansi aplikacije.

C. Firebase Crashlytics

Jedan od najznačajnijih servisa koje *Firebase* pruža jeste *Firebase Crashlytics*, alat za praćenje i prijavljivanje grešaka u realnom vremenu. Ovaj servis automatski detektuje i beleži padove i greške aplikacija, pružajući programerima detaljne izveštaje o njihovim uzrocima. Izveštaji sadrže informacije o tipu greške, uređaju, operativnom sistemu i kontekstu u kome je greška nastala, što značajno olakšava dijagnostiku i otklanjanje problema. *Firebase Crashlytics* takođe omogućava rangiranje grešaka prema učestalosti i uticaju na korisnike, čime se programerima omogućava da prioritizuju rešavanje najkritičnijih problema.

VIII. ZAKLJUČAK

Primenom tehnologija računarstva u oblaku u razvoju *RAF Network* aplikacije demonstriran je napredak u oblasti mobilnih aplikacija, posebno u pogledu skalabilnosti, fleksibilnosti i bezbednosti. Ovakva rešenja omogućavaju programerima da brže odgovore na zahteve korisnika, prilagode se promenama u opterećenju i obezbede stabilnost aplikacije u različitim uslovima. *RAF Network* je primer kako tehnologija računarstva u oblaku može uspešno podržati složene sisteme, u kojima je važno održavanje visokog nivoa dostupnosti i zaštite podataka. Rad je takođe ukazao na potencijalne pravce za

budući razvoj aplikacije kroz unapređenje integracije sa postojećim studentskim servisima, čime bi se studentima omogućio lakši pristup studentskim servisima. Pored toga, aplikacija može postati značajan alat za poboljšanje komunikacije između studenata i profesora, kao i za umrežavanje i saradnju sa alumni zajednicom. U budućim istraživanjima i razvoju sličnih sistema, moglo bi se razmatrati unapređenje performansi kroz efikasnije upravljanje resursima u oblaku, kao i upotreba naprednih tehnologija kao što su veštačka inteligencija i mašinsko učenje. Ovo bi omogućilo još veću personalizaciju sadržaja, analizu podataka i predviđanje potreba korisnika.

LITERATURA

[1] Mona, E., & Sivakumari, S. (2021). Alumni Social Networking Site. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *7*, 467-472.

[2] Olsson, M. (2020). A Comparison of Performance and Looks Between Flutter and Native Applications: When to prefer Flutter over native in mobile application development.

[3] Gulati, M., Fulay, A., & Datta, S. (2013). *Building and Managing a Cloud Using Oracle Enterprise Manager 12c*. McGraw-Hill.E. H. Miller, "Periodical style—Accepted for publication," *IEEE Trans. Antennas Propagat.*, to be published.

[4] Mouat, A. (2015). *Using Docker: Developing and deploying software with containers*. " O'Reilly Media, Inc.".

[5] Tao, Y., & Chen, G. (2016, April). An extensible universal reverse proxy architecture. In *2016 International Conference on Network and Information Systems for Computers (ICNISC)* (pp. 8-11). IEEE.

[6] Sun, Y., Zhang, J., Xiong, Y., & Zhu, G. (2014). Data security and privacy in cloud computing. *International Journal of Distributed Sensor Networks*, *10*(7), 190903.

[7] Benbunan-Fich, R., & Benbunan, A. (2007). Understanding user behavior with new mobile applications. *The Journal of Strategic Information Systems*, *16*(4), 393-412.

ABSTRACT

This paper explores the concept of developing mobile applications using cloud solutions, focusing on the *RAF Network* mobile application. Through a detailed analysis of the technologies and methodologies employed in the development and implementation of this application, the paper demonstrates how cloud computing can significantly enhance the functionality, scalability, and reliability of modern mobile applications. The *RAF Network* application was developed using the *Flutter* framework for the frontend and the *Django* web framework for the backend, with cloud infrastructure hosted on the *DigitalOcean* platform. The application enables seamless communication between the client and server sides through a REST API exposed via a *Traefik* reverse proxy. This approach ensures high scalability and flexibility, allowing for the easy addition of features such as integration with student services and enhanced communication between students and professors. Additionally,

Firebase, a cloud-based platform developed by *Google*, was integrated to provide comprehensive tools for mobile app analytics, performance monitoring, and real-time error tracking. This enhances the app's ability to gather critical insights and optimize user experience. The paper highlights the advantages of using cloud technologies in mobile application development and proposes future improvements in functionality and user experience.

DEVELOPING MOBILE APPLICATIONS USING CLOUD SOLUTIONS: CASE STUDY OF THE RAF NETWORK MOBILE APPLICATION

Luka Petrović, dr Nemanja Radosavljević, docent i dr Mirjana Radivojević, profesor