Analiza mobilnog oblaka

Milica Stanačić, Mirjana Mirković

***Sadržaj* — Rad analizira mobilno računarstvo u oblaku (MCC), njegove komponente, arhitekturu i bezbjednosne izazove, uz fokus na tri klaud (eng. cloud) platforme: Fajrbejz (Firebase), AVS amplifaj (AWS Amplify) i Ejžr (Azure). Karakteristike rešenja mobilnog oblaka su predstavljene i analizirane na primjeru aplikacije za analizu sastava kozmetike različitih proizvođača skeniranjem barkoda datog proizvoda.**

Lista simbola i skraćenica:

MCC - *Mobile Cloud Computing* - računarstvo u oblaku zasnovano na mobilnim uređajima
VM - *Virtual Machine* - virtuelna mašina
API - *Application Programming Interface* - programski interfejs za komunikaciju između sistema
DB - *Database* - baza podataka
IaaS - *Infrastructure as a Service* - infrastruktura kao servis
PaaS - *Platform as a Service* - platforma kao servis
SaaS - *Software as a Service* - softver kao servis
SSL - *Secure Socket Layer* - kriptografski protokol za sigurnu komunikaciju preko internet mreže
CDN - *Content Delivery Network* - mreža za isporuku sadržaja
SDK - *Software Development Kit* - razvojno okruženje softvera
CI/CD - *Continuous Integration / Continuous Deployment* - kontinuirana integracija i isporuka
A/B (testiranje) - metoda upoređivanja dvije verzije predmeta testiranja
MVP - *Minimum Viable Product* - minimalno funkcionalan proizvod
GDPR - *General Data Protection Regulation* - opšta uredba o zaštiti podataka
ISO - *International Organization for Standardization* - međunarodna organizacija za standarde iz različitih oblasti
HIPAA - *Health Insurance Portability and Accountability Act* - američki zakon koji reguliše privatnost i sigurnost zdravstvenih podataka

***Ključne riječi* — Firebase, Izazovi mobilnog oblaka, Mobilni oblak (Mobile Cloud Computing / MCC).**

M. S. Autor, Računarski fakultet, Srbija (telefon: 381-61-4081093; e-mail: mstanacic5024m@raf.rs)
M. M. Suautor, Računarski fakultet, Srbija (e-mail: mmirkovic@raf.rs)

I. UVOD

MOBILNI uređaji (npr. pametni telefoni, tablet računari, pametni satovi) su danas neizostavan dio ljudskog života kao najefikasniji i najpraktičniji alati za komunikaciju, neograničeni mjestom i vremenskom zonom. Baš iz tog razloga potekla je i motivacija za detaljnijim istraživanjem rada mobilnih aplikacija.

Brzi napredak mobilnog računarstva postaje snažan trend u razvoju informacionih tehnologija, kao i u oblasti trgovine i industrije. Međutim, mobilni uređaji se suočavaju sa brojnim izazovima u pogledu resursa (npr. trajanje baterije, skladišni prostor i propusni opseg) i komunikacije (npr. mobilnost i bezbjednost). Ograničeni resursi značajno otežavaju unapređenje kvaliteta usluga. Sa druge strane, računarstvo u oblaku (*CC - Cloud Computing*) pruža brojne prednosti omogućavajući korisnicima da koriste infrastrukturu (npr. servere, mreže i skladišta podataka), platforme (npr. operativne sisteme) i softvere (npr. aplikacije), koje nude pružaoci usluga računarstva u oblaku (npr. Gugl (*Google*), Amazon, itd.) po niskoj cijeni. Drugim riječima dobar dio nedostataka mobilnih uređaja danas je riješen usvajanjem principa iz računarstva u oblaku i iz tih razloga danas razvijamo mobilno računarstvo u oblaku.

Mobilno računarstvo u oblaku donosi nove tipove usluga i omogućava korisnicima mobilnih uređaja da u potpunosti iskoriste prednosti klaud tehnologija.

Cilj ovog rada je da pruži sveobuhvatnu analizu mobilnog oblaka, njegovih ključnih komponenti, prednosti i izazova, kao i mogućnosti njegove primjene u različitim industrijama. Ovaj rad izdvojiće 3 najzastupljenije platforme na polju mobilnog oblaka. Poseban akcenat biće stavljen i na pitanja bezbjednosti i privatnosti. Rad će se takođe dotaći i praktičnog primjera mobilne aplikacije koja uz pomoć mobilnog oblaka analizira sastav kozmetike koju korisnik koristi i grupiše potencijalno kancerogene sastojke.

Kroz analizu dostupne literature, aktuelnih istraživanja i primjera iz prakse, rad će pokušati da odgovori na danas veoma aktuelno pitanje: na koji način mobilni oblak transformiše način na koji koristimo mobilne tehnologije i kako utiče na digitalnu budućnost društva?

II. OSNOVNE KARAKTERISTIKE

Mobilni oblak je rešenje u kojem se podaci i aplikacije koje su ranije bile ograničene na hardver mobilnog uređaja, sada premještaju i izvršavaju na udaljenim *cloud* serverima, dok uređaj funkcioniše kao interfejs. MCC koristi

snagu platformi računarstva u oblaku kako bi mobilnim korisnicima obezbjedio dostupnost i performanse koje bi bile nemoguće postići lokalno zbog ograničenja mobilnog hardvera.

Koncept MCC-a zasniva se na osnovnoj ideji distribuiranog računarstva, gdje se obrada podataka, skladištenje i kompleksne kalkulacije ne obavljaju lokalno, već na udaljenim klaud (*cloud*) serverima. Korisnici putem interneta komuniciraju sa aplikacijama koje se izvršavaju na tim serverima, dok se korisnički interfejs prikazuje na njihovim mobilnim uređajima.

Glavne komponente MCC sistema uključuju:

- Mobilne uređaje: pametni telefoni, tableti, nosivi uređaji.
- Bežične mreže: 4G/5G, Wi-Fi, koje omogućavaju pristup internetu.
- Klaud (*Cloud*) infrastrukturu: udaljeni serveri koji obezbjeđuju procesore velike jačine i skladištenje.

Jedna od ključnih karakteristika MCC-a je elastičnost - mogućnost skaliranja resursa u skladu sa zahtjevima korisnika. Ovo omogućava aplikacijama visokih performansi, poput vještačke inteligencije, prepoznavanja glasa i slike, da budu dostupne i na uređajima sa ograničenim kapacitetima.

Mobilni oblak koristi se u različitim scenarijima - od jednostavnih aplikacija za skladištenje podataka (npr. Gugl drajv - *Google Drive,* Ajklaud - *iCloud*), do složenih aplikacija koje omogućavaju pristup velikim bazama podataka ili obradu u realnom vremenu, kao što su mobilne zdravstvene aplikacije, aplikacije za pametne gradove i još mnogo toga.

MCC predstavlja osnovu za mnoge moderne digitalne servise, omogućavajući globalnu dostupnost aplikacija, redukciju troškova, produženi vijek trajanja mobilnih uređaja i prije svega unaprijeđeno korisničko iskustvo.

III. ARHITEKTURA MOBILNOG OBLAKA

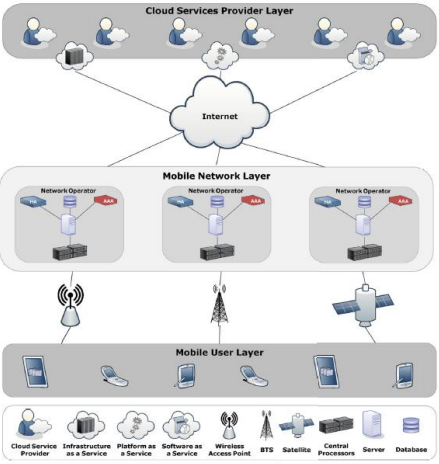
Arhitektura mobilnog oblaka definiše kako mobilni uređaji komuniciraju sa cloud infrastrukturom, kako se podaci prenose, skladište i obrađuju, i koje komponente omogućavaju realizaciju servisa. Ova arhitektura može se prikazati kao višeslojni model, pri čemu svaki sloj ima specifičnu funkciju u obradi korisničkih zahtjeva. U tom modelu, mobilni uređaji su povezani sa mobilnim mrežama putem baznih stanica, koje uspostavljaju i upravljaju vezom između mreže i uređaja.

Zahtjevi i informacije korisnika (npr. identitet i lokacija) prenose se centralnim procesorima povezanim sa serverima koji obezbjeđuju mobilne mrežne usluge (npr. to su serveri Telekoma, A1, Jetela (*Yettel)*, itd.).

Operator mobilne mreže može pružiti usluge poput autentifikacije i autorizacije (provjera identiteta korisnika putem SIM kartice ili broja telefona) i obračuna, zasnovane na podacima iz baza i domaćih agenata. Nakon toga, korisnički zahtjevi se prenose ka oblaku putem interneta, gdje klaud kontroleri obrađuju zahtjeve i pružaju odgovarajuće usluge iz oblaka.

Ove usluge su razvijene u skladu sa principima jutiliti kompjutinga (*utility computing* tzv. uslužno računarstvo), virtualizacije i servisno-orijentisane arhitekture [1].

Detalji arhitekture mogu varirati. Na primjer, neka rešenja opisuju četvoroslojnu arhitekturu, dok druga mogu opisati drugačije rešenje.



Sl. 1. Prikaz arhitekture klaud sistema, izvor [2].

U ovom radu fokus je na slojevitoj arhitekturi [2] klaud sistema:

1. Sloj korisničkog mobilnog uređaja (*Mobile User Layer*) Ovaj sloj uključuje sve mobilne uređaje (mobilne telefone, tablete, pametne satove itd.) sa kojih korisnici pristupaju cloud servisima. Uređaji šalju zahtjeve putem interneta i prikazuju rezultate koje vraća server. Klijentske aplikacije su često lagane i dizajnirane da minimizuju korišćenje resursa.

2. Sloj mrežne konekcije (*Mobile Network Layer*) Ovaj sloj omogućava prenos podataka između mobilnog uređaja i cloud servera. Uglavnom koristi bežične mreže (3G, 4G, 5G, Wi-Fi). Pouzdanost i brzina mreže presudni su za kvalitet korisničkog iskustva.

3. Klaud sloj (*Cloud Service Provider Layer*) Ovaj sloj predstavlja srž MCC sistema. Sastoji se od:

- IaaS (infrastruktura kao servis): pruža virtuelne mašine, skladište i druge

osnovne resurse. (npr. aplikacija Uber)

• PaaS (platforma kao servis): omogućava programerima da razvijaju, testiraju i distribuiraju aplikacije bez brige o osnovnoj infrastrukturi. (npr. aplikacija Duolingo)

• SaaS (softver kao servis): gotove aplikacije koje korisnici koriste direktno putem interneta (npr. Gugl doks (*Google Docs),* Dropboks (*Dropbox*)).

A. Komunikacija između slojeva

Kada korisnik sa mobilnog uređaja pošalje zahtjev (npr. za pristup dokumentu), zahtjev prolazi kroz mrežni sloj do cloud infrastrukture. Server obrađuje zahtjev, pristupa podacima, a zatim vraća rezultat nazad do korisnika. Efikasna komunikacija i minimizacija latencije ključni su za uspješan rad MCC sistema.

B. Mobilni agenti i virtualizacija

U MCC sistemima često se koriste mobilni agenti koji funkcionišu kao posrednici između uređaja i oblaka. Mobilni agenti su softverski entiteti koji mogu da se nezavisno kreću kroz mrežu između različitih čvorova (mobilni uređaji i oblak infrastruktura), izvršavajući zadatke u ime korisnika ili aplikacije. Njihova glavna prednost je to što mogu migrirati sa jednog sistema na drugi, noseći sa sobom stanje, podatke i kod koji treba da se izvrši. U kontekstu mobilnog oblaka oni prvenstveno optimizuju performanse, smanjuju latenciju, štede bateriju i smanjuju količinu prenesenih podataka. Pored mobilnih agenata, često se koriste i tehnike virtuelizacije koje omogućavaju dinamičko raspoređivanje resursa.

Virtualizacija u okviru MCC okruženja ima jedinstvene karakteristike koje je razlikuju od tradicionalne primjene virtualizacije u statičnim cloud sistemima. U MCC-u, mobilni uređaji su ograničeni resursima kao što su procesorska snaga, memorija i baterija, što nameće potrebu za specifičnim pristupom virtualizaciji.

Prije svega, MCC koristi pristup u kojem se teški zadaci aplikacija delegiraju (eng. *offloading*) sa mobilnog uređaja na udaljene virtuelne instance u oblaku. Ovaj pristup omogućava efikasnije korišćenje ograničenih resursa mobilnog uređaja, jer se zahtjevna obrada, kao što su analiza podataka, prepoznavanje glasa ili obrada slike, izvršava na serverima sa znatno većim kapacitetima.

Pored toga, MCC koristi dinamičku migraciju virtuelnih mašina (VM), gdje se virtuelna instanca pomjera bliže korisniku - npr. na edž (*edge)* servere - kako bi se smanjila latencija i poboljšalo korisničko iskustvo. Ova tehnika

je posebno važna za aplikacije koje zahtjevaju brz odziv u realnom vremenu.

Još jedna specifičnost MCC virtualizacije jeste korišćenje lakših oblika virtualizacije, poput kontejnera (npr. Doker (*Docker*)), koji omogućavaju brzo pokretanje i manju potrošnju resursa u poređenju sa klasičnim virtuelnim mašinama. Ovakav pristup je naročito efikasan za kratke zadatke i aplikacije koje se često pokreću i gase.

Na kraju, MCC sistemi često kombinuju mobilne agente sa virtuelnim mašinama, gdje agenti dinamički biraju instancu VM na kojoj će izvršiti zadatke, u zavisnosti od dostupnih resursa, sigurnosti i blizine.

C. *Arhitekture specifične za MCC virtualizaciju*

U okviru MCC-a, razvijene su specifične arhitekture koje podržavaju virtualizaciju prilagođenu mobilnim uslovima. Dvije istaknute arhitekture u ovom domenu su Klonklaud (*CloneCloud)* i Klaudlet (*Cloudlet)*.

Klonklaud [3] predstavlja arhitekturu u kojoj se aplikacija koja se izvršava na mobilnom uređaju djelimično „klonira“ u oblak. Na taj način, dio aplikacije koji je računski intenzivan se izvršava u virtuelnoj instanci u cloudu, dok se interfejs i osnovne funkcije zadržavaju na uređaju. Ova arhitektura omogućava korisničko iskustvo bez prekida i zastoja, jer se obrada odvija transparentno i bez direktne interakcije.

Klaudlet [4] arhitektura podrazumijeva postojanje lokalizovanih mini-cloud servera, najčešće smještenih u blizini krajnjih korisnika (npr. na Wi-Fi ruterima, baznim stanicama). Klaudleti koriste virtuelne mašine koje mogu brzo da se pokrenu putem tzv. VM overlej *(overlay)* tehnike, gdje se baza mašine već nalazi lokalno, a samo korisnički dio dolazi iz oblaka. Ovo omogućava gotovo trenutno izvršavanje aplikacija sa niskom latencijom.

U cjelini, arhitekture koje podržavaju virtualizaciju u MCC-u imaju za cilj da obezbjede nisku latenciju, efikasno korišćenje resursa, skalabilnost i poboljšano korisničko iskustvo, što ih čini osnovom savremenih aplikacija u mobilnom oblaku.

IV. FAJRBEJZ (FIREBASE)

Ovo poglavlje se zasniva na činjenicama iznijetim u [5]. Fajrbejz je platforma u vlasništvu kompanije Gugl koja omogućava razvoj, implementaciju i upravljanje mobilnim i veb aplikacijama korišćenjem klaud infrastrukture. Predstavlja jedan od najsveobuhvatnijih primjera mobilnog oblaka, pružajući različite servise koji pokrivaju backend, analitiku, autentifikaciju, bazu podataka, hosting itd. Glavna prednost Fajrbejza je što omogućava brzu izradu aplikacija bez potrebe za sopstvenim serverom.

A. Ključne komponente Firebase platforme

• Fajrbejz autentifikacija (*Firebase Authentication)*: Omogućava jednostavno upravljanje korisničkim nalozima putem maila, lozinke, Gugl naloga i drugih metoda autentifikacije. Takođe podržava dvofaktorsku autentifikaciju i integraciju sa trećim stranama. Jednostavna integracija - nema pisanja backend koda. Integrisana je sa bazama i tzv. „funkcije iz oblaka" komponentom *(Cloud Functions)*.

• Klaud fajrstor (*Cloud Firestore*) i Fajrbejz baza u realnom vremenu *(Firebase Realtime Database)*: Dvije opcije za skladištenje i sinhronizaciju podataka. Klaud fajrstor koristi strukturu dokumenata i kolekcija pogodnu za kompleksnije aplikacije, dok Fajrbejz baza u realnom vremenu omogućava trenutnu sinhronizaciju podataka idealnu za chat aplikacije i aplikacije sa velikom interaktivnošću. Fajrbejz baza u realnom vremenu je JSON bazirana NoSQL baza podataka u realnom vremenu. Klaud fajrstor je modernija i skalabilnija NoSQL baza sa podrškom za kolekcije i dokumente, brža je i bolja za složenije upite. Obije omogućavaju sinhronizaciju podataka između servera i klijenta u realnom vremenu. Podržavaju offline rad.

• Fajrbejz funkcije iz oblaka (*Firebase Cloud Functions)*: Serverles *(serverless)* funkcionalnost koja omogućava da se backend logika izvršava kao odgovor na događaje (npr. korisnik se registruje, podatak se promeni u bazi, HTTP zahtev, itd). Serverles funkcije se pišu u Javaskript/Tajpskriptu (*JavaScript/TypeScript*) i izvršavaju u Gugl klaud okruženju.

• Fajrbejz hosting(*Firebase Hosting)*: Omogućava brzo i bezbjedno hostovanje veb aplikacija i statičkog sadržaja, uz automatski SSL (*Secure Socket Layer*) i globalnu CDN (*Content Delivery Network*) podršku. Podržava mogućnost postavljanja (*deploy)* preko komandne linije (CLI - *Command Line Interface*). Brz, siguran, besplatan za osnovne potrebe.

• Fajrbejz klaud mesedžing (*Firebase Cloud Messaging* - FCM): Servis za slanje puš (*push)* notifikacija na Android, iOS i veb uređaje, sa mogućnošću targetiranja korisnika prema ponašanju ili lokaciji. Radi i kada aplikacija nije aktivna (u pozadini). Može se povezati sa komponentom „funkcije iz oblaka" (*Cloud Functions)* za dinamičko slanje poruka.

• Fajrbejz analitika (*Firebase Analytics - Google Analytics for Firebase*): Pruža uvid u ponašanje korisnika i korišćenje funkcionalnosti aplikacije. Koristi se za optimizaciju korisničkog iskustva i u svrhu marketinških kampanja.

• Fajrbejz skladište (*Firebase Storage)*: Klaud skladište za čuvanje i dijeljenje datoteka (npr. slike, video snimci). Bazirano na Gugl klaud skladištu *(Google Cloud Storage)*. Omogućava pravila pristupa i integraciju

sa autentifikacijom.

B. Kako teče proces instalacije i korišćenja?

Firebase se jednostavno integriše sa Android studiom i Ikskod *(Xcode)* razvojnim okruženjima, pružajući i emulator sa svojim okruženjem (*emulator suite)* za testiranje aplikacija lokalno. Takođe omogućava A/B testiranje, udaljene konfiguracije aplikacija, i povezivanje sa Bigkveri (*BigQuery)* platformom za naprednu analitiku. Fajrbejz koristi svoj komandni interfejs za upravljanje projektima iz terminala. Komanda „npm install -g firebase-tools“ vrši instalaciju. Zatim se prijavimo sa komadnom „firebase login“. Inicijalizacija projekta se vrši komandom „firebase init“. Izaberemo funkcionalnosti koje želimo da koristimo (Fajrstor, hosting, funkcije iz oblaka itd.). Fajrbejz *CLI* će pitati: koji projekat sa Fajrbejz konzole želimo da koristimo - postojeći ili novi, da li želimo da koristimo Tajpskript *(TypeScript)* za funkcije iz oblaka *(Cloud Functions)* itd. Zatim razvijamo funkcionalnosti lokalno. Možemo koristiti i Fajrbejz emulator za lokalni razvoj komandom: „firebase emulators:start“. Postavljanje *(deploy)* na Fajrbejz vrši se komandom „firebase deploy“.

C. Prednosti i ograničenja

Kada se preporučuje korišćenje Firebase-a? Kod razvoja:

• MVP-a / prototipa - Idealno za brzo razvijanje aplikacija bez postavljanja bekend *(backend)* servera. Brzo se postavlja: baza u realnom vremenu (*Realtime Database*), autentifikacija, hosting i klaud funkcije.

• Aplikacija u realnom vremenu - baza u realnom vremenu i Fajrstor omogućavaju trenutno sinhronizovanje podataka između korisnika.

• Aplikacija sa manjim timovima bez Devopsa (*DevOps*) - Ne zahtjeva podešavanje servera, baza, CI/CD infrastrukture - sve je serverles. Ugrađena autentifikacija, baza, skladište, klaud funkcije, puš notifikacije.

• Mobilnih aplikacija (Android/iOS) - Fajrbejz ima SDK integraciju sa Androidom, iOS-om i Flaterom (*Flutter)*. Dolazi uz Fajrbejz analitiku, A/B testiranje, krešlitiks (*Crashlytics)* i konfiguraciju na daljinu (*Remote Config)*.

• Ako koristimo Guglovu klaud platformu (*Google Cloud Platform* - GCP) Fajrbejz se savršeno integriše sa Bigkveri, Pub/Sub, klaud skladištem i drugim GCP servisima.

Kada se ne preporučuje korišćenje Fajrbejza?

• Kod kompleksne poslovne logike i SQL zahtjeva - Fajrbejz koristi NoSQL, bez podrške za: JOIN upite, kompleksne agregacije, transakcije između više kolekcija (djelimično podržano u Fajrstoru, ali ograničeno).

• Kod aplikacija sa velikim količinama podataka i složenim strukturama - teško se modeluju duboko ugnježdene strukture u Fajrbejzu. Ograničenje veličine dokumenata, broj zahtjeva u sekundi i indeksacija može biti usko grlo (eng. *bottleneck*).

• Kada ne želimo da zavisimo od jednog vendora (*vendor lock-in*) - teško se migrira sa Fajrbejza na neku drugu infrastrukturu.

• Kod aplikacija koje nemaju potrebe za funkcionalnostima u realnom vremenu (ili imaju male potrebe).

• Troškovi kod skaliranja - Fajrbejz djeluje jeftino na početku, ali sa visokim prometom može biti skup. Nema fiksne mjesečne cijene - sve se naplaćuje po potrošnji.

D. Plaćanje

Fajrbejz ima besplatan paket (Spark plan) dovoljan za većinu razvoja manjih aplikacija i njihovog testiranja. Ovaj paket ima ograničenja. Na plaćeni plan se prelazi ako postoji veći broj korisnika ili mnogo čitanja/pisanja u Fajrstoru tj. ako se pređu limiti iz besplatnog paketa. Zatim, ako se koriste napredne opcije kao što su Blejz funkcije (*Blaze Functions)*, klaud vižn (*Cloud Vision),* Automatsko mašinsko učenje *(AutoML)* itd. Kao i kada želimo prilagođen domen (*custom domain)* za hosting sa HTTPS-om. Plaćanje se vrši po „plati onoliko koliko potrošiš“ (*pay-as-you-go)* filozofiji.

V. AVS AMPLIFAJ (AWS AMPLIFY)

Veći dio sadržaja u ovom poglavlju preuzet je i prilagođen iz [5]. AVS amplifaj *(AWS Amplify)* je skup alata i servisa kompanije Amazon *(Amazon Web Services - AWS)* koji omogućava jednostavan razvoj i implementaciju skalabilnih mobilnih i veb aplikacija. Kao primjer platforme u okviru mobilnog oblaka, Amplifaj omogućava programerima da brzo povežu aplikacije sa klaud funkcionalnostima, uz minimalan napor i bez potrebe za upravljanjem infrastrukturom.

A. Ključne komponente AVS Amplifaj

• Autentifikacija (*Auth*): Integracija sa Amazon Kognito *(Cognito)* servisom za upravljanje korisnicima, registraciju, prijavu i višefaktorsku autentifikaciju.

• Dejtastor *(DataStore)*: Sinhronizovani i *offline-first* sloj za podatke koji koristi Grafkjuel (*GraphQL)* interfejs, sa automatskim sinhronizovanjem između lokalne baze i oblaka.

• API (REST i Grafkjuel): Jednostavna konfiguracija API-ja pomoću AVS Apsinka *(AppSync)* za Grafkjuel ili Amazon API gejtvej *(gateway)* za

REST.

• Skladište *(Storage)*: Omogućava korisnicima da upravljaju fajlovima (npr. slikama, dokumentima) pomoću Amazon S3 servisa.

• Hosting i CI/CD: Pruža hosting za statičke veb aplikacije i omogućava automatsko postavljanje *(deployment)* iz Git repozitorijuma.

• Analitika i monitoring: Praćenje ponašanja korisnika i događaja korišćenjem Amazon Pinpoint i klaudvoča (*CloudWatch)*.

B. Kako teče proces instalacije i korišćenja?

Instaliramo Amplifaj sa komandom: „npm install -g @aws-amplify/cli“. Inicijalizujemo projekat sa „amplify init“. Zatim dodajemo funkcionalnosti, npr. za autentifikaciju komanda glasi „amplify add auth“, slično za *storage* „amplify add storage“ itd. Postavljanje *(deploy)* na AVS se radi pomoću komande „amplify push“.

C. Prednosti i ograničenja

Kada se preporučuje korišćenje AVS Amplifaja?

• Fulstek *(full-stack)* razvoj u jednom alatu - Amplifaj je dizajniran za developere koji žele da upravljaju autentifikacijom (*Cognito*), koriste Grafkjuel/REST API-je *(AppSync/API Gateway + Lambda),* dodaju skladište i sve to bez potrebe da direktno konfigurišu AVS servise pojedinačno.

• Serverles i skalabilne aplikacije - Amplifaj koristi AVS Lambda, *AppSync* i druge serverles servise, što omogućava odličnu automatsku skalabilnost.

• Aplikacije koje koriste Grafkjuel API-je - Amplifaj ima izuzetnu podršku za Grafkjuel kroz AVS *AppSync*.

• Ukoliko smo već u AVS ekosistemu - Amplifaj se prirodno uklapa u postojeću infrastrukturu.

• CI/CD za frontend aplikacije - Amplifaj konzola omogućava automatski bild (*build*) i postavljanje (deploy) frontenda (React, Angular, Vue, Next.js) direktno iz Git repozitorijuma.

Kada se ne preporučuje korišćenje AVS Amplifaja?

• Ako ne koristimo AVS ekosistem i nemamo planove u daljoj ili bližoj budućnosti da ga koristimo.

• Aplikacije koje zahtjevaju potpunu fleksibilnost bekenda - Amplifaj automatizuje mnogo toga, ali to ograničava granularnu kontrolu. Ako želimo preciznu kontrolu nad arhitekturom, bolje je odabrati drugo rešenje.

• Kompleksne baze i relacione veze - Amplifaj (*AppSync + DynamoDB*) koristi NoSQL modele koji nisu prirodni za relacione podatke. Iako možemo

koristiti Amazonov servis za relacione baze podataka *(Relational Database Service)* uz dodatno podešavanje, to je mimo osnovnog Amplifaj toka.

• Visoka složenost i skaliranje bez predvidivog obrasca - Amplifajev sistem može biti „crna rupa" i skupo skalirati bez kontrole nad pozadinskim servisima.

• Veći timovi sa DevOps praksom - U organizacijama u kojima je praksa da se infrastruktura definise i upravlja uz pomoć programskog koda umjesto da se ručno konfiguriše, Amplifajev CLI pristup može biti previše automatizovan i neusklađen sa CI/CD politikama.

D. Plaćanje

AVS Amplifaj se može koristiti besplatno prvih 12 mjeseci nakon otvaranja naloga, ali postoje ograničenja u besplatnom (*Free Tier*) paketu. Nakon isteka 12 mjeseci korisnici plaćaju po filozofiji „plati onoliko koliko potrošiš" (*pay-as-you-go*).

VI. MAJKROSOFT EJŽR (MICROSOFT AZURE MOBILE APPS)

Razmatranja u ovom poglavlju takođe su zasnovana na [5]. Majkrosoft Ejžr eksositem za mobilne aplikacije *(Microsoft Azure Mobile Apps,* u daljem tekstu EMA*)* je komponenta šire Ejžr (*Azure)* platforme koja omogućava razvoj i upravljanje mobilnim aplikacijama korišćenjem skalabilnih klaud servisa uključujući: sinhronizaciju podataka, korisničku autentifikaciju, notifikacije, upravljanje podacima *offline/online*, povezivanje sa bazama podataka i drugim Ejžr servisima.

A. Ključne komponente Majkrosoft Ejžr ekosistema za mobilne aplikacije

• *Mobile App Backend (App Service):* Omogućava hostovanje REST API-ja za mobilne aplikacije koristeći ASP.NET, Node.js ili druge jezike podržane na Ejžr servisu za aplikacije *(Azure App Service)*.

• Autentifikacija/Autorizacija: Integracija Ejžr aktivnog direktorijuma (Azure Active Directory) sa Fejzbuk, Gugl, Tviter i Majkrosoft nalozima uz mogućnost prilagođene autentifikacije.

• Oflajn data sink (*Offline Data Sync)*: Klijentima omogućava da koriste podatke lokalno i da sinhronizuju promjene kada se uspostavi internet konekcija.

• Puš notifikacije: Koristi Ejžr hab *(Azure Notification Hubs)* za slanje notifikacija velikom broju korisnika preko različitih platformi (Android, iOS, *Windows*).

• Slakdište podataka *(Data Storage)*: Korišćenje Ejžr baza *(Azure SQL Database*, *Cosmos DB)* ili drugih Ejžr skladišta za bekend podatke aplikacija.

B. Kako teče proces instalacije i korišćenja?

Napravimo bekend mobilne aplikacije na Ejžr servisu za aplikacije. Povežemo ga sa bazom podataka, autentikacijom (npr. Fejzbuk login), puš servisom (*Azure Notification Hub*). Mobilna aplikacija koristi SDK (Android, iOS) da komunicira sa klaud bekendom.

C. Prednosti i ograničenja

Kada se preporučuje korišćenje Ejžr ekosistema za mobilne aplikacije?

• Ako koristimo *Ejžr* kao glavni klaud provajder - EMA se uklapa sa ostatkom ekosistema.

• Ako radimo sa aplikacijama koje zahtjevaju relacione baze i složenije upite. Za razliku od Fajrbejza i Amplifaja, koji koriste NoSQL pristup, EMA je *SQL-friendly*. Podržava: ORM modele (*Entity Framework*), SQL server, *stored* procedure itd.

• Enterprajz aplikacije sa zahtjevnom bezbjednošću - EMA nudi: Ejžr aktivni direktorijum autentifikaciju, RBAC (*Role-Based Access Control*), šifrovanje, poštovanje propisa/standarda *(compliance)* poput ISO, HIPAA, GDPR isl., privatne mreže i VNet integraciju.

• Oflajn *(offline)* sinhronizacija - *EMA* ima ugrađenu podršku za: oflajn podatke na klijentu kao i automatsku sinhronizaciju kad se uređaj ponovo poveže na mrežu.

• Ako koristimo .NET, C# itd. tj. već razvijamo aplikacije koristeći djelove Majkrosoftovog ekosistema, EMA je prirodan izbor za bekend.

Kada se ne preporučuje korišćenje EMA?

• Ako nam je potrebna funkcionalnost u realnom vremenu (puš poruke, sink *(sync),* čat *(chat*)) - EMA nema infrastrukturu koja podržava funkcionalnosti u realnom vremenu. Nema ugrađenu podršku za vebsoket (*WebSocket)* kao Fajrbejz baza u realnom vremenu ili Fajrstor.

• Ako želimo ultra-brz prototip ili MVP - EMA zahtjeva: Definisanje modela, povezivanje sa bazom, više podešavanja infrastrukture.

• Ako ne koristimo Ejžr ekosistem

• Kompleksna i intenzivna prilagođavanja funkcionalnosti - Za razliku od Amplifaj/Fajrbejza koji imaju gotove module (*Auth, Storage, Functions*), EMA često zahtjeva više ručne konfiguracije i programiranja.

• Cijena i kompleksnost u manjem timu - Ejžr servisi su robusni, ali i skuplji i komplikovaniji za održavanje ako tim nije iskusan u DevOps-u i infrastrukturi.

D. Plaćanje

Ejžr ima besplatan plan sa ograničenim mogućnostima. Kada se pređu limiti, plaćanje se vrši po „plati onoliko koliko potrošiš“ (*pay-as-you-go)* filozofiji.

VII. IZAZOVI I OGRANIČENJA

Iako mobilni oblak donosi brojne prednosti, njegova implementacija i korišćenje nose sa sobom određene izazove, posebno kada je riječ o bezbjednosti podataka, privatnosti korisnika i kvalitetu usluge.

A. Izazovi u mobilnoj komunikaciji

• Niska propusna moć - Propusna moć predstavlja jedan od najvećih izazova MCC-a jer su radio resursi za bežične mreže znatno ograničeniji u poređenju sa žičanim mrežama. Propusna moć (eng. *bandwidth*) označava količinu podataka koja može da se prenese kroz mrežu u određenom vremenskom periodu. Jedan od predloga rešenja povećanja propusne moći iz [1] jeste formiranje „zajednice korisnika“. Rešenje se odnosi na korisnike koji se nalaze na istoj lokaciji (npr. stanica, stadion) i koriste isti sadržaj. Korisnici formiraju zajednicu u kojoj je svaki član zadužen za dio sadržaja (npr. kod video zapisa to bi bili zvuk, slika, titlovi) i razmjenjuju ih sa ostalima, čime se poboljšava kvalitet reprodukcije. Međutim, ovo rešenje funkcioniše samo ako su korisnici zainteresovani za isti sadržaj i ne uzima u obzir pravičnu raspodjelu doprinosa među članovima tj. neko može da učestvuje više, neko manje, a svi imaju koristi.

Napredniji predlog - uvođenje politike raspodjele podataka koja odlučuje kada i koliko raspoložive propusnosti će se dijeliti među korisnicima i preko koje mreže (npr. *WiFi*, *WiMAX*). Sistem periodično prikuplja podatke o korisniku (npr. jačinu signala, stanje baterije itd.) i koristi Markovljev algoritam - statistički metod za donošenje odluka na osnovu vjerovatnoće budućih stanja. Na osnovu algoritma donose se zaključci - ko treba da pomogne drugim korisnicima, koliko propusne moći treba dodijeliti (npr. 10%) itd. Ova inteligentna raspodjela resursa se sprovodi preko clouda, u okviru sistema zvanog RACE (*Resource-Aware Collaborative Execution*). To je sistem koji koristi resurse oblaka da bi prikupio i analizirao prikupljene korisničke podatke. On takođe donosi odluke o raspodjeli propusne moći i saradnji među korisnicima.

• Dostupnost usluge - Dostupnost usluge je znatno važnija u MCC-u nego u CC-u. Korisnici mogu ostati bez pristupa oblaku zbog zagušenja, kvara mreže ili gubitka signala. Umjesto da čekaju povratak internet veze, autori [1]

predlažu da se izvrši detekcija obližnjih čvorova (npr. drugih mobilnih uređaja, laptopova) koji su u dometu korisnika. Dakle ovo rešenje se odnosi na dijeljenje sadržaja između mobilnih uređaja bez oslanjanja na fiksnu infrastrukturu (npr. bazne stanice, internet provajdere, itd.). Ako su ti čvorovi stabilni (npr. imaju dobru vezu i nisu u pokretu), omogućava se komunikacija u *ad hoc* režimu: to znači da uređaji komuniciraju direktno, bez posredovanja centralne infrastrukture. Na primjer, jedan telefon može slati podatke do cloud servera putem drugog telefona koji ima signal. Međutim, rešenje ne uzima u obzir mobilnost npr. šta ako se "obližnji čvor" pomjera ili nestane iz dometa. Takođe ne uzimaju se u obzir sposobnosti uređaja npr. da li drugi uređaj ima dovoljno baterije? Ne uzima se u obzir ni privatnost.

Drugi predlog jeste *WiFi multihop* sistem *MoNet*. *WiFi multihop* je mrežni sistem u kojem podatak putuje od jednog uređaja do drugog, u više "skokova" (eng. *hops*) dok ne stigne do cilja, dok je *MoNet* naziv konkretne implementacije jednog *WiFi multihop* sistema. Kako funkcioniše sistem? Svaki čvor periodično emituje kontrolne poruke sa različitim informacijama (npr. o signalu, o parametrima sistema), a zatim se na uređaju formiraju liste susjednih čvorova i dostupnog sadržaja. Na osnovu ovih informacija sistem procjenjuje koji su to najbolji posrednici. MoNet koristi tri tipa ključeva da bi zaštitio komunikaciju: *account key* - za identitet korisnika, *friend key* - za razmjenu podataka između poznatih uređaja, *content key* - za zaštitu konkretnog sadržaja koji se dijeli.

B. Izazovi u domenu obrade (offloading)

Kao što je prethodno objašnjeno, prebacivanje obrade je jedna od ključnih osobina MCC-a za poboljšanje trajanja baterije i performansi aplikacija. Međutim, postoje brojna pitanja vezana za efikasno i dinamično prebacivanje u promenljivim uslovima okruženja.

- Prebacivanje u statičnom okruženju. Eksperimenti iz [1] pokazuju da *offloading* nije uvijek energetski efikasno rešenje. Kod kompajliranja malih programa, lokalna obrada može biti jeftinija. Na primjer, ako veličina koda nakon kompajliranja iznosi 500 KB, offloading troši oko 5% baterije, dok lokalna obrada troši 10%. U tom slučaju ušteda iznosi 50%. Međutim, za 250 KB koda, učinkovitost pada na 30%, a kod manjih programa offloading troši više energije.

Ključni izazov je odlučiti kada i šta prebaciti kako bi se maksimizovala energetska efikasnost.

Jedan problem je to što se pretpostavlja da su vremena izvršavanja poznata unaprijed. To je nerealna pretpostavka, jer vrijeme izvršavanja u stvarnosti varira zbog: opterećenja mreže, promjene u jačini signala, razlike u uređajima

korisnika, promjene u zauzeću servera itd.

Autori [1] zato predlažu pametniji pristup koji se zasniva na stvarnom ponašanju sistema: vrši se prikupljanje statistike tokom izvršavanja tj. sistem bilježi koliko inače traju slični procesi, u različitim uslovima. Za svaki proces se postavi maksimalno vrijeme koje bi trebalo da protekne da bi se taj proces izvršio (npr. na osnovu prosjeka iz statistike). Ako se zadatak ne završi u tom roku, prekida se lokalno izvršavanje (na telefonu) i procesi se prebacuju na server u oblaku da se tamo izvrše brže i efikasnije. Time se postiže ušteda energije do 17% u poređenju sa postojećim pristupima.

• Prebacivanje u dinamičnom okruženju. Ovo poglavlje prikazuje nekoliko pristupa za rešavanje problema prebacivanja obrade u dinamičnim mrežnim uslovima (npr. promjena statusa veze i propusnog opsega). Promjene u okruženju mogu izazvati dodatne probleme, kao što su gubitak prenijetih podataka ili gubitak rezultata obrađenih na serveru.

U radu se dalje analiziraju performanse *offloading*-a sistema u bežičnim okruženjima. Razmatraju tri slučaja: kada se aplikacija izvršava lokalno (bez *offloading*-a), zatim slučaj „idealnog *offloading*-a“ kada se izvršava u idealnim uslovima (bez grešaka) i poslednji kada se vrši *offloading* uz mehanizme za oporavak od grešaka (u slučaju kada dođe do prekida veze, greške prilikom prenosa i sl.). U poslednjem slučaju, kada dođe do greške, aplikacija se ponovo *offload*-uje, ali samo djelimično - samo za one zadatke/procese koji nisu uspješno završeni, čime se poboljšava ukupno vrijeme izvršavanja i štede se resursi. Ograničenje ove analize je što se mobilno okruženje posmatra samo kao *ad hoc* mreža i takođe - svaki prekid veze se tretira kao greška, što u realnim sistemima ne mora biti greška - kratak prekid signala se automatski rešava.

Pristupi rešavanju problema prekida veze:

1. Bez dinamičkog pristupa - u slučaju prekida veze tokom izvršavanja, server periodično provjerava status veze i čuva informacije o zadacima. Kada se veza obnovi u unaprijed određenom vremenskom roku, rezultati se šalju korisniku. Ako se veza ne obnovi u unaprijed određenom vremenskom roku, zadaci se brišu. Prednost: sistem ne gubi podatke odmah, već čeka da se veza vrati. Međutim, ovi pristupi su opšti i ne razmatraju kako raspodijeliti djelove aplikacije u skladu sa trenutnim stanjem mreže, uređaja itd. Takođe ne postoji inteligentno tj dinamičko prilagođavanje - sve se radi po unaprijed definisanim pravilima bez fleksibilnosti.

2. Djelimično dinamično rešenje - sistem za particionisanje aplikacija u dinamičnim okruženjima koji se sastoji iz tri koraka: strukturiranje aplikacije (aplikacija se dizajnira tako da svi njeni moduli mogu raditi lokalno i u

oblaku; mobilni uređaj i cloud imaju kopije svih modula), izbor podjele (aplikacija sama odlučuje u realnom vremenu koje module da pokrene lokalno, a koje u oblaku) i zaštita podataka (osjetljivi moduli se izvršavaju isključivo lokalno). Nedostatak ovog sistema je što se podjele zasnivaju na predikcijama izvan mreže (*offline* analiza), to znači da sistem ne uči i ne reaguje dinamično na promjene u mrežnom okruženju, što smanjuje preciznost.

3. Napredni dinamički sistem - MAUI (*Mobile Assistant Using Infrastructure*). MAUI je konkretna arhitektura koja rešava upravo nedostatak prethodnog sistema. Radi u 3 koraka. Prvi: MAUI koristi prenosivost koda da kreira dvije verzije aplikacije - za lokalno i za udaljeno izvršavanje. Pošto pametni telefoni koriste različite arhitekture (ARM, x86), MAUI omogućava pokretanje istog programa na različitim procesorima bez pristupa izvornom kodu. Drugi korak: sistem koristi refleksiju (sposobnost programa da ispituje svoj kod) da identifikuje metode označene kao „*remoteable*“, i serijalizaciju za prenos samo potrebnog stanja programa u oblak čime štedi podatke i energiju. Treći: MAUI koristi linearnu programsku formulaciju da kombinuje troškove komunikacije, energije i status mreže (propusnost i kašnjenje) i donosi optimalne odluke o particionisanju. Ovaj pristup omogućava visoko dinamično prebacivanje koda uz minimalne izmjene aplikacije.

C. Bezbjednost

Zaštita privatnosti korisnika i njegovih podataka od napadača je ključna za izgradnju i održavanje povjerenja korisnika u mobilnu platformu, posebno u MCC-u. Ovdje se bezbjednosna pitanja u MCC-u razmatraju kroz dvije kategorije: bezbjednost mobilnih korisnika i bezbjednost podataka. Takođe, razmatraju se i neka rešenja za ove izazove.

- Bezbjednost mobilnih korisnika - Integracija GPS modula može ugroziti privatnost korisnika. Popularnost LBS (*Location-Based Service*) servisa tj aplikacija koje za svoj rad uključuju poznavanje lokacije korisnika, raste uz GPS uređaje, ali donosi rizik od otkrivanja stvarne lokacije korisnika (mnogim aplikacijama nije neophodno da znaju koja je tačna lokacija korisnika, već je dovoljan određeni opseg, što bi i zaštitilo privatnost korisnika). Lokacijski server povjerenja LTS (*Location Trusted Server*) je posrednik između korisnika i LBS servisa, koji štiti identitet korisnika tako što maskira njegovu pravu lokaciju. Problem se javlja kada aplikacije koriste GPS, one mogu precizno da odrede našu lokaciju. Ako te informacije stignu direktno do LBS servisa, postoji rizik da naša privatnost bude narušena i da se otkrije gdje se nalazimo, gdje idemo, kad smo kod kuće, kada ne itd.

Naučnici su predložili rešenje koje kombinuje LTS server i k-anonimlnost: lokacija uređaja se šalje LTS serveru, LTS server koristi koncept k-anonimnosti tj. lokacija korisnika se skirva među najmanje k korisnika. Ideja k-anonimnosti je da ako se naši podaci ne mogu razlikovati od podataka najmanje još k-1 korisnika, onda smo anonimni unutar grupe od k korisnika. U kontekstu lokacije, to znači: ako je naša lokacija neodvojiva od lokacija još bar k-1 drugih ljudi, tako niko ne može sa sigurnošću reći ko smo mi ili gdje se tačno nalazimo. LTS pravi zamaskirani region tj širu oblast u kojoj je korisnik i šalje je LBS-u. Međutim, postoji problem - ako LTS i LBS serveri međusobno sarađuju, mogu kombinovati podatke i otkriti stvarnu lokaciju korisnika.

Drugo rešenje je da se maskirani region generiše direktno na mobilnom uređaju uz korišćenje Casper algoritma. Casper algoritam kombinuje sopstvene podatke sa podacima drugih korisnika dobijenih iz oblaka radi efikasnosti, a zatim uređaj sam formira k-anonimnu oblast tj. zamaskirani region i šalje ga LBS servisu.

• Bezbjednost podataka u oblaku - Mobilni uređaji izloženi su brojnim bezbjednosnim prijetnjama poput zlonamjernih virusa. Jedno od rešenja je korišćenje antivirus softvera. Ipak, zbog ograničenih resursa mobilnih uređaja (procesorska moć i baterija), kontinuirano pokretanje takvog softvera nije održivo. Autori [1] predlažu premještanje detekcije prijetnji u oblak. U pitanju je proširenje postojeće CloudAV (*Cloud Anti Virus*) platforme za detekciju malvera u oblaku, sa komponentama: host agent (na uređaju) i mrežni servis (u oblaku). Host agent je lagana aplikacija koja nadgleda aktivnosti fajlova. Ako fajl nije u kešu prethodno analiziranih, šalje se na verifikaciju u oblak. Prednost ovog pristupa je paralelna upotreba više antivirus sistema (poput Avasta, Kasperskog isl.) kroz virtuelizovane kontejnere (svaki sistem je izolovan, nezavisan i skalabilan).

Tu se srećemo sa još jednim problemom bezbjednosti podataka u oblaku - integritet. I korisnici i programeri imaju koristi od čuvanja velikih količina podataka u oblaku, ali je neophodno osigurati integritet, autentifikaciju i digitalna prava nad tim podacima. Korisnike često brine da li su njihovi podaci izmijenjeni ili oštećeni. Većina postojećih rešenja u zaštiti podataka rade korektno, ali su energetski zahtjevna - troše puno baterije na mobilnim uređajima. Autori [1] predlažu šemu sa tri komponente: mobilni klijent, klaud skladište i treća pouzdana strana (TCC - *Trusted Crypto Coprocessor*). U tri faze (inicijalizacija, ažuriranje, verifikacija), datotekama se dodjeljuju *MAC* kodovi. Ti kodovi se čuvaju lokalno, dok se fajlovi skladište u oblaku. TCC upoređuje MAC vrijednosti za provjeru integriteta. Klijent izvršava samo

jednostavnu provjeru, dok TCC vrši verifikaciju, čime se štedi 90% procesorskih resursa mobilnog uređaja.

- Bezbjednost sistema zbog propusta kod korisnika - Čak i kada su *cloud* servisi dobro zaštićeni, bezbjednost MCC sistema može biti ugrožena ponašanjem korisnika, kao što su korišćenje slabe lozinke, instalacija sumnjivih aplikacija ili nepažljivo dijeljenje podataka.

D. Upravljanje resursima i latencija

U aplikacijama u realnom vremenu (npr. igrama, video pozivima, AR/VR sistemima), čak i minimalna latencija može imati značajan uticaj na korisničko iskustvo. Efikasna alokacija resursa i optimizacija ruta prenosa podataka ostaje izazov.

E. Kompatibilnost i standardizacija

Različiti mobilni operativni sistemi, cloud platforme i API-ji otežavaju razvoj univerzalno kompatibilnih rešenja. Nedostatak standardizovanih protokola dodatno komplikuje integraciju.

F. Pravna i regulatorna pitanja

Korišćenje servisa koji se nalaze van granica zemlje korisnika može biti u sukobu sa lokalnim zakonima o zaštiti podataka. Takođe, postavlja se pitanje odgovornosti u slučaju gubitka podataka ili u slučaju bezbjednosnog incidenta.

Da bi se ovi izazovi uspješno prevazišli, neophodna je saradnja između korisnika, programera i cloud provajdera, kao i implementacija sveobuhvatne bezbjednosne politike i zakonskih regulativa.

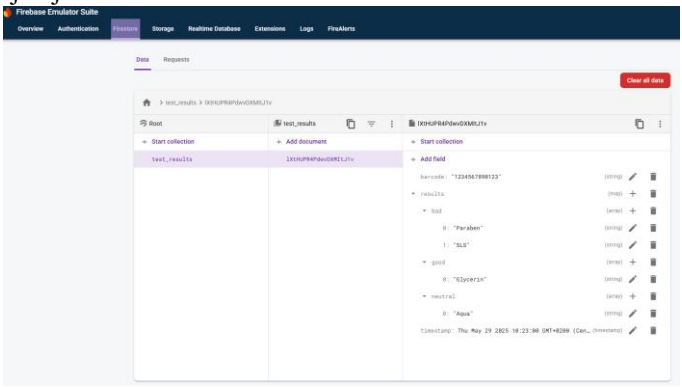
VIII. PRAKTIČAN PRIMJER

A. Opis aplikacije

Predmet analize je prototip mobile aplikacije koja korisnicima omogućava da skeniranjem barkoda sa kozmetičkih proizvoda, dobiju automatsku klasifikaciju sastojaka prema njihovoj bezbjednosti po ljudski organizam. Nakon unosa barkoda, aplikacija šalje podatke ka Fajrbejz funkciji iz oblaka (*Firebase Cloud Function* servisu), koja zatim analizira sastojke, klasifikuje ih kao „dobri“, „loši“ ili „neutralni“ i čuva rezultate u Fajrstor bazi podataka.

Osnovne funkcionalnosti uključuju: 1. Skener barkoda (npr. korišćenjem *ML Kit Barcode API* biblioteke, ali trenutno funkcioniše “ručno” zbog nemogućnosti testiranja na drugi način), 2. Klaud funkcija za analizu

sastojaka, 3. Upis rezultata u Fajrstor bazu podataka i 4. Lokalno testiranje putem Fajrbejz emulatora.



Sl. 2. Prikaz rezultata, nakon upisa u bazu, unutar Fajrbejz emulatora.

B. Izbor klaud platforme

U cilju implementacije ove aplikacije, analizirane su tri platforme: Fajrbejz *(Google Cloud),* AVS Amplifaj *(Amazon Web Services) i* Ejžr ekosistem za mobilne aplikcije *(Microsoft Azure)*.

Za potrebe izrade prototipa aplikacije, odabrana je platforma Fajrbejz zbog 1. brze i jednostavne integracije sa Android aplikacijama (posebno u Kotlin programskom jeziku), 2. podrške za Fajrstor bazu podataka u realnom vremenu, 3. mogućnosti definisanja klaud funkcija za serversku logiku, 4. autentikacije korisnika preko gotovih rešenja i 5. besplatnog plana koji je dovoljan za većinu razvoja i testiranja.

C. Arhitektura aplikacije

Arhitekturu aplikacije čine *Frontend* (mobilna aplikacija u Kotlinu), *Backend* (*Firebase Cloud Functions*) i baza podataka (*Cloud Firestore*). *Frontend* čini: modul za skeniranje barkoda, interfejs za prikaz rezultata pretrage (nije dovršen), komunikacija sa Fajrstor bazom. *Backend* vrši prijem zahtjeva sa barkodom, klasifikuje sastojake po unaprijed definisanoj bazi na dobre, loše i neutralne i upisuje rezultate u bazu. Baza podataka dinamički kreira dokumenat sa sledećim podacima: *barcode*, *results*, *timestamp* tj. barkod, rezultat (pozitivan, negativan, neutralan) i vrijeme upisa.

D. Ograničenja

Potpuno iskorišćenje mobilnog oblaka nije bilo moguće jer određene funkcionalnosti nisu bile besplatne. Ova činjenica je čak u nekim trenucima

otežavala implementaciju ovog rešenja.

LITERATURA

[1] Hoang Dinh Thai, Chonho Lee, Dusit Niyato, Ping Wang - A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing, 13(18), 1587-1605.
[2] Talal H. Noor, Sherali Zeadally, Abdullah Alfazi, Quan Z. Sheng - Mobile cloud computing: Challenges and future research directions - Journal of Network and Computer Applications 115 (2018) 70-85
[3] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik - CloneCloud: Boosting Mobile Device Applications Through Cloud Clone Execution. Intel Labs Berkeley, Princeton University.
[4] Dejan Kovachev, Yiwei Cao, Ralf Klamma - Mobile Cloud Computing: A Comparison of Application Models. Information Systems & Database Technologies, RWTH Aachen University, Ahornstr. 55, 52056 Aachen Germany
[5] Google Cloud Platform, AWS, Microsoft Azure - zvanična dokumentacija dostupna putem: https://cloud.google.com, https://aws.amazon.com, https://azure.microsoft.com.

ABSTRACT

Mobile Cloud Computing (MCC) is an emerging paradigm that integrates the advantages of mobile computing and cloud technologies to overcome the limitations of mobile devices such as limited processing power, storage, and battery life. This paper provides a comprehensive analysis of MCC, exploring its architecture, core technologies, benefits, and challenges. A particular focus is placed on the layered structure of MCC systems, including the mobile client, network, cloud infrastructure, and service management layers. The role of virtualization, mobile agents, and containerization in improving performance and reducing latency is also discussed.

The study compares three leading MCC platforms: Google Firebase, AWS Amplify, and Microsoft Azure Mobile Apps - highlighting their key functionalities, integration processes, and cost structures. A practical case study demonstrates the development of a prototype mobile application that scans cosmetic product barcodes and classifies ingredients based on their health safety, leveraging Firebase Cloud Functions and Firestore.

The findings indicate that MCC plays a vital role in the digital transformation of modern society, enabling scalable, accessible, and efficient mobile services. Nevertheless, ongoing advancements in security, legal frameworks, and collaborative efforts are essential for its sustainable and responsible deployment.

MOBILE CLOUD COMPUTING

Milica Stanačić, Mirjana Mirković