Višeplatformski razvoj mobilne aplikacije za prikaz aktivnosti srednje škole

Dijanović, Lovro

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

Sveučilišni studij

Višeplatformski razvoj mobilne aplikacije za prikaz aktivnosti srednje škole

Završni rad

Lovro Dijanović

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 12.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Lovro Dijanović
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4336, 22.07.2019.
OIB Pristupnika:	49583162997
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Dino Kurtagić,
Sumentor iz tvrtke:	
Naslov završnog rada:	Višeplatformski razvoj mobilne aplikacije za prikaz aktivnosti srednje škole
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	U završnom radu treba s načela programskog inženjerstva opisati i usporediti višeplatformske tehnologije za razvoj mobilnih aplikacija. Potrebno je objasniti i usporediti trenutno najpopularnije BaaS usluge i pružatelje istih, te objasniti kako koristiti bazu podataka kao uslugu u oblaku računala s mobilnim aplikacijama. Potrebno je predložiti tehničko rješenje i programski jezik (Dart), kao i razvojni okvir kojim bi se kreirala
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	12.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.09.2022.

Ime i prezime studenta:	Lovro Dijanović
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4336, 22.07.2019.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Višeplatformski razvoj mobilne aplikacije za prikaz aktivnosti srednje škole**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora Dino Kurtagić,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

U moderno doba mobilni su uređaji najkorišteniji tehnološki uređaj svakog učenika. U ovom radu bit će predstavljen višeplatformski razvoj mobilne aplikacije za Srednju školu Donji Miholjac. Cilj je ove aplikacije riješiti problem nedospijevanja bitnih informacija do učenika zbog male posjećenosti mrežne aplikacije i društvenih mreža škole. U razvoju su korištene višeplatformske tehnologije, što znači da se iz jedne kodne baze može u isto vrijeme izraditi aplikacija za više operacijskih sustava, konkretno Android i iOS, s time da se nastoje ispuniti očekivanja različitih operacijskih sustava.

Korisnici će imati pristup informacijama o školi, važnim obavijestima, opisima programa i zanimanja, rasporedu sati, kao i drugim bitnim informacijama vezanima uz školu. Tijekom izrade aplikacije korišten je Flutter, razvojni paket za izradu višeplatformskih aplikacija, Dart programski jezik te Firebase platforma. U razvoju se koriste višeplatformske tehnologije s ciljem omogućavanja pristupa aplikaciji svim studentima bez obzira na to koji mobilni uređaj koriste.

U poglavlju 2 bit će pružen osvrt na izazove u prikazu aktivnosti škole, bit će predstavljena neka postojeća rješenja te će se predstaviti višeplatformski razvoj i tehnologije. U poglavlju 3 bit će opisani razlozi izrade aplikacije, ciljana skupina korisnika, funkcionalni i nefunkcionalni zahtjevi na aplikaciji, model te građa mobilne aplikacije. Nakon toga, u poglavlju 4, bit će opisani alati i tehnologije korišteni u izradi aplikacije te predstavljeno programsko rješenje mobilne aplikacije. Naposljetku, u poglavlju 5, prolaskom kroz zaslone aplikacije bit će prikazano njeno korištenje.

1.1. Zadatak završnog rada

U završnom radu treba po načelu programskog inženjerstva opisati i usporediti višeplatformske tehnologije za razvoj mobilnih aplikacija. Potrebno je objasniti i usporediti trenutno najpopularnije BaaS usluge i pružatelje istih te objasniti kako koristiti bazu podataka kao uslugu u oblaku računala s mobilnim aplikacijama. Potrebno je predložiti tehničko rješenje i programski jezik (Dart), kao i razvojni okvir kojim bi se kreirala mobilna aplikacija (Flutter) za SŠ Donji Miholjac. Aplikacija treba sadržavati zaslone i funkcionalnosti: općenito o školi, važne obavijesti s *push* notifikacijama, opis svih programa i zanimanja koje škola pruža, raspored sati, događanja u obliku novosti preuzetih s API sučelja mrežne stranice škole, projekti škole, dio vezan uz učeničku zadrugu, informacije vezane uz upis u prvi razred, informacije vezane uz završne razrede, informacije i kontaktne podatke. Potrebno je obaviti nužno testiranje programske podrške te prezentirati rezultate.

2. PRIKAZ INFORMACIJA VEZANIH UZ ŠKOLE I OSVRT NA VIŠEPLATFORMSKE TEHNOLOGIJE

Ponuda različitih mobilnih uređaja u slobodnom tržištu podrazumijeva raznolikost operacijskih sustava koje ti uređaji koriste. Postoje brojni operacijski sustavi za mobilne uređaje, no danas Android i iOS zauzimaju više od 99% svjetskog tržišnog udjela. U svijetu dominira Android (72% naprema 28%) [1], kao i u Hrvatskoj (84% naprema 15.5%) [2], dok u SAD-u dominira iOS (57.5% naprema 42.3%) [3]. Ovo znači da ako se aplikacija razvija isključivo za jedan operacijski sustav, velika količina potencijalnih korisnika ostaje zanemarena.

Jedno od rješenja je razvijati svoju aplikaciju za oba operativna sustava, no ima nekoliko problema s ovim pristupom. Operacijski sustavi na mobilnim uređajima imaju različite standarde [4], koriste različite programske jezike za razvijanje aplikacija te imaju različita distribucijska tržišta, što znači da je potreban veći kapital i veći tim programera za razvoj željene aplikacije na više platformi [5]. Drugo je rješenje razvoj mrežnih aplikacija kojima će korisnici moći pristupiti na svim pametnim mobilnim uređajima, no pokazalo se da su uobičajeno nativne aplikacije brže u odnosu na mrežne aplikacije u čitanju iz datoteka, pisanju u datoteke, slanju zahtjeva za mrežnim informacijama, slanju zahtjeva za informacije od GPS-a, slanju zvučnih obavijesti i drugi nedostaci [6]. Dobro rješenje su tehnologije za višeplatformski razvoj kao Flutter, React Native, Ionic i ostali. Mobilni se okviri razvijaju kako bi ponudili rješenja koja omogućuju pojednostavljenje procesa razvoja kako bi se izradili proizvodi koji se mogu implementirati u različitim mobilnim operacijskim sustavima [7]. Prednosti su višeplatformnog razvoja relativno mali trošak za razvoj, kratko vrijeme razvoja te dostupnost korisnicima, dok su neki od nedostataka problemi s kompatibilnošću očvrsja, u određenim situacijama lošija izvedba te stabilnost [8].

U ovom poglavlju bit će opisani izazovi prikaza informacija vezanih uz aktivnosti škole, predstavit će se neka postojeća rješenja te će se opisati višeplatformske programske tehnologije i BaaS (engl. *Backend as a service)* usluge.

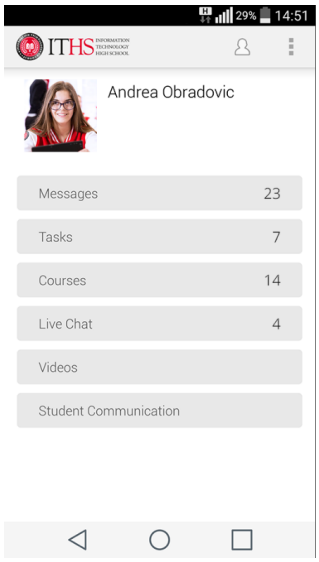
2.1. Izazovi prikaza informacija vezanih uz aktivnosti škole i prikaz postojećih rješenja

Glavni je izazov u prikazu informacija vezanih uz aktivnosti škole dolazak informacija do učenika. Društvene mreže pokazale su se kao dobar način slanja informacija učenicima, no nije sigurno da je svaki učenik aktivan na društvenim mrežama. Mrežne stranice još su jedno rješenje, no pokazalo se da učenici sve manje posjećuju mrežne stranice škole. Problem je i što učenici nisu ni na koji način obaviješteni kada stigne nova vijest ili događaj na mrežnu

stranicu škole.

2.1.1 Prikaz postojećih rješenja

ITHS aplikacija namijenjena je učenicima Srednje škole za informacijske tehnologije. Služi za lakši pristup *online* platofrmi za podršku u učenju. Učenici uz pomoć platforme, u okviru besplatnih tečajeva, imaju pristup programima koji nisu u programu škole, čime šire svoje znanje. Na slici 2.1 mogu se vidjeti neki od zaslona aplikacije ITHS.



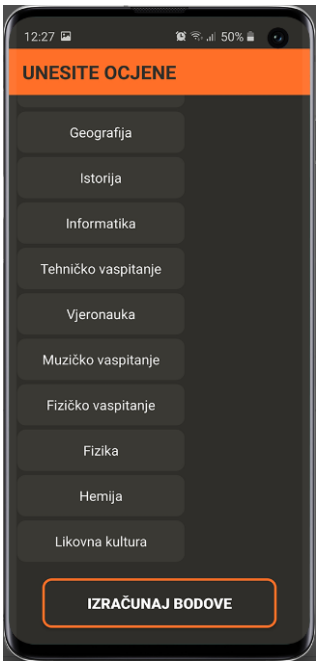
(a) *Zaslon mobilne aplikacije ITHS*

(b) *Zaslon mobilne aplikacije ITHS*

Slika 2.1: *Aplikacija ITHS*

Aplikacija Srednje škole "Nikola Tesla" Teslić služi učenicima za računanje broja bodova koji im je potreban za upis u srednju školu te za upoznavanje sa zanimanjima koja škola nudi. Učenici imaju lakši pristup lokaciji škole i mrežnoj stranici škole putem aplikacije. Također, uz pomoć aplikacije učenici se mogu lakše informirati o događanjima unutar škole. Na slici 2.2 vidljivi su neki od zaslona mobilne aplikacije.

(a) *Zaslon mobilne aplikacije*



(b) *Zaslon mobilne aplikacije*

Slika 2.2: *Aplikacija Srednje škole "Nikola Tesla" Teslić*

2.2. Višeplatformske programske tehnologije

Višeplatformski je mobilni razvoj stvaranje aplikacija koje su prilagođene za više mobilnih operacijskih sustava [9]. Višeplatformske aplikacije mogu koristiti gotovo identičan izvorni kod, što znači da se mogu razviti aplikacije za iOS i Android istovremeno bez velikih izmjena u kodu. Izvorno, razvoj mobilnih aplikacija bio je otežan izradom *backenda* koji bi radio na više platformi. Iako je bilo dugotrajno i skupo, često je bilo lakše izgraditi aplikacije za svaki mobilni operacijski sustav pojedinačno. Danas je puno lakša izrada višeplatformskih aplikacija zbog napretka tehnologija za višeplatformski razvoj. U idućem dijelu bit će opisano nekoliko takvih najzastupljenijih tehnologija.

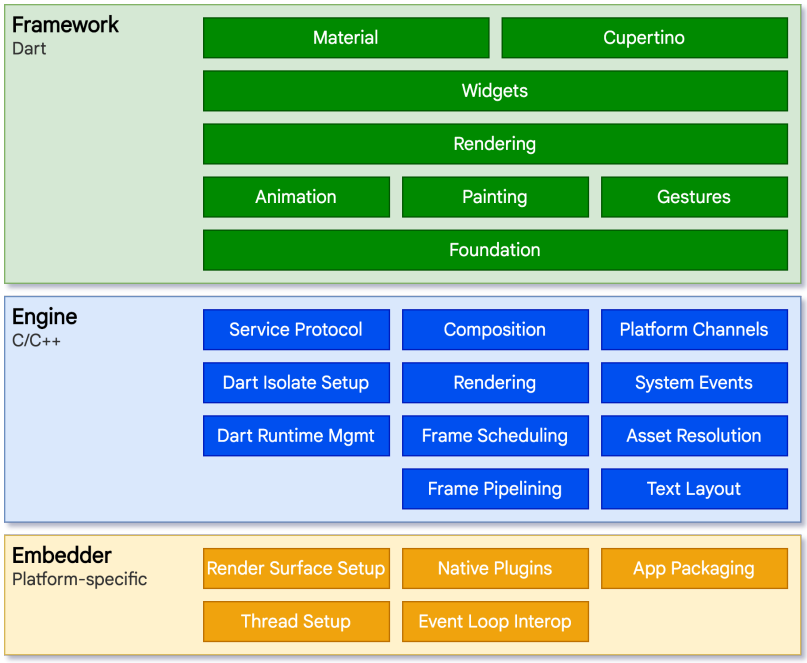
2.2.1 Flutter

Flutter je skup alata za višeplatformski razvoj koji je dizajniran da omogući ponovnu uporabu koda u operacijskim sustavima kao što su iOS i Android, dok također dopušta aplikacijama izravno sučelje s temeljnim uslugama platforme [10]. Cilj je Fluttera omogućiti programerima da isporuče aplikacije visokih performansi koje imaju prirodan osjećaj na različitim platformama, prihvaćajući razlike tamo gdje postoje, dok dijele što je više moguće koda.

Flutter je dizajniran kao slojevit i proširiv sustav. Sastoji se od međusobno neovisnih biblioteka koje skupno ovise o sloju ispod sebe. Niti jedan sloj nema povlašteni pristup sloju

ispod, a svaki dio okvira dizajniran je da bude opcionalan i zamjenjiv.

Flutter arhitektura može se podjeliti na tri cjeline: ugraditelj (engl. *embedder*), motor (engl. *engine*) i okvir (engl. *framework*). Korištenjem ugraditelja Flutter kod može se integrirati u postojeću aplikaciju kao modul ili kod može biti cijeli sadržaj aplikacije. U jezgri Fluttera je Flutter motor, koji je većinski napisan u C++ i sadrži sve potrebno za podršku svim Flutter aplikacijama. Okvir služi za komunikaciju između Fluttera i razvojnog inžinjera. Detaljna arhitektura nalazi se na slici 2.3.



Slika 2.3: *Flutter arhitektura [10]*

Flutter koristi takozvane *widgete* kao osnovne gradivne elemente. *Widget* je nadimak za svaku komponentu izgrađenu u Flutteru. Ovo može biti običan tekst, polje za unos podataka, slika unutar aplikacije i svi ostali elementi. Velika prednost *widgeta* je ta što su napravljeni da izgledaju nativno na fizičkim uređajima te što se mogu lako prilagoditi željama programera. *Widgeti* su gradivni blokovi korisničkog sučelja Flutter aplikacije. Svaki *widget* gnijezdi se unutar svog roditelja i može primiti kontekst od roditelja. Ova struktura je ista sve do korijenskog (engl. *root*) *widgeta*. *Widgeti* ne samo da kontroliraju i utječu na ponašanje aplikacije nego barataju i odgovaraju na korisnikove naredbe. Zbog toga je bitno da *widgeti* budu brzi, što uključuje *rendering* i animacije. Flutter sam odlučuje kada i kako su *widgeti renderani*, što omogućuje veću felksibilnost i prilagodljivost. Postoje dva tipa *widgeta* u Flutteru: *stateful* i *stateless* widgeti, čije razlike ilustrira tablica 2.1.

	Dinamična kompozicija	Nepromjenjiv	Podstanje objekta nepromjenjivo
Stateless widget	Ne	Da	Ne
Stateful widget	Da	Da	Da

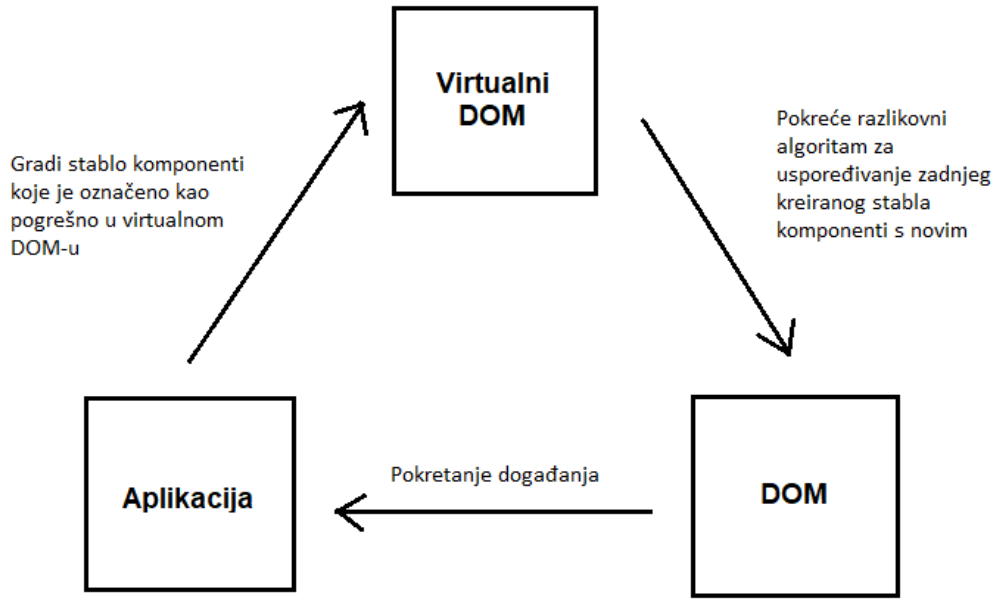
Tab. 2.1: *Razlike između tipova widgeta*

Stateful widget dolazi s objektom koji predstavlja stanje. Stanje predstavlja unutarnju strukturu *widgeta*. Stanje *widgeta* opisuje kako *widget* reagira na korisnikove upute, kao na primjer promjena rasporeda *widgeta*. Nasuprot tome, *stateless widgeti* obični su *widgeti* koji ne reagiraju na korisnikove radnje. Ovaj uzorak dizajna omogućuje samom *widgetu* da bude nepromjenjiv, što sprječava često *renderanje*.

2.2.2 React Native

React Native je tehnologija za višeplatformski razvoj koju je na tržište izbacio Facebook 2015. godine [5]. React Native koristi nativne skripte za stvaranje stvarnih nativnih komponenti. S obzirom na to da stvara nativne komponente, potrebna je određena količina koda specifična za platformu, no moguće je da većina koda bude dijeljena među platformama. React Native okvir promovira koncept "nauči jednom, piši svugdje", što znači da kada programer jednom nauči React Native okvir, može ga koristiti na više platformi. React okvir programeri koriste zbog njegove jednostavnosti i lakoće, ali i zbog njegovog učinkovitog procesa razvoja. Pri korištenju React Nativea treba biti svjestan njegovih prednosti i mana prije odluke je li okvir pogodan za razvoj željene aplikacije.

Jedan od glavnih razloga zašto React Native aplikacija može funkcionirati na različitim platformama jest korištenje Virtualnog DOM-a (engl. *Document Object Model*) [11]. Virtualni DOM omogućuje Reactu manipuliranje jednostavnim DOM stablom, koje je mapirano sa stvarnim DOM stablom, za dobivanje boljih performansi. Na slici 2.4 prikazane su relacije između Virtualnog DOM-a, DOM-a i aplikacije.



Slika 2.4: *Tijek rada Virtualnog DOM-a*

Svaki korisnikov unos u DOM, na primjer pritisak gumba, pokrenut će događaje u aplikaciji, što ima utjecaj na strukturu Virtualnog DOM-a. Aplikacija također pokreće algoritam za periodično osvježavanje DOM-a.

2.2.3 Razvojni okvir Ionic

Ionic je okvir koji omogućuje razvoj hibridnih mobilnih aplikacija uz pomoć mrežnih tehnologija poput HTML5, CSS i JavaScript [12]. Ionic pruža komponente s kojima je moguće izgraditi značajke slične nativnima za mobilne uređaje. Većina značajki nativnih aplikacija dostupna je u Ionicu i mogu se lako proširiti ovisno o potrebama aplikacije.

Ionic sam ne dopušta komunikaciju sa značajkama uređaja poput GPS-a i kamere, umjesto toga koristi Cordovu u te svrhe. Još jedna značajka Ionica je što su komponente vrlo slabo spregnute. Postoji mogućnost koristiti Ionic u malim količinama na već postojećoj hibridnoj aplikaciji.

Ionic okvir izgrađen je pomoću AngularJS, jednog od najpopularnijih i najkorištenijih JavaScript okvira. Ovo znači da je moguće koristiti sve mogućnosti Angulara unutar Ionic aplikacija. Angular također omogućuje laku organizaciju aplikacije za rad u timu, dok u isto vrijeme omogućava lako dodavanje novih značajki i biblioteka.

2.2.4 Flutter naspram React Nativea

Postoje brojne razlike u implementaciji Fluttera i React Native-a. Prema tome, kada bi se ista aplikacija razvijala u oba alata, dobio bi se različit rezultat. Flutter je, zbog svoje arhitekture, efikasniji i zahtijeva manje resursa, dok React Native ima širu zajednicu programera, što uključuje bolje biblioteke koje su u širokoj upotrebi. Prema istraživanjima [11], koristeći FPS (engl. *frames per second*), može se grubo usporediti koji alat ima bolje performanse. Za usporedbu alata u istraživanju koristilo se listanje zaslona te zauzeće memorije.

Vertikalna lista čest je zaslon jedne mobilne aplikacije. Uobičajno je da nativni Android i iOS imaju svoje specijalne kolekcije za manipuliranje listama kao što su RecyclerView i UICollectionView. Listanje zaslona zahtijeva brzi odgovor na akciju i brze animacije. U istraživanju se koristila jednostavna vertikalna lista s 1000 artikala. Zaključak je da Flutter i React Native jako dobro reagiraju na listanje zaslona. Prosječni je FPS u oba slučaja iznad 60. Flutterov je FPS poprilično stabilan, s pokojim skokom prema manjem FPS-u, što može biti uzrokovano korisnikovim radnjama i *renderanjem* animacija koje zahtijevaju puno resursa.

Još jedan bitan faktor za ocjenjivanje performansi je brzina ulaza i izlaza (I/O). Prijenos datoteka unutar uređaja može se mjeriti kao brzina IO diska. Čest je scenarij da mobilna aplikacija komunicira s uređajem s ciljem izmjene informacija. Obje tehnologije koriste biblioteke koje koriste nativnu optimizaciju datotečnog sustava uređaja na kojima je aplikacija pokrenuta. React Native ima prednost u prosječnom potrošenom vremenu i jedinstvenom potrošenom vremenu za pisanje u datoteku. Za ovo je zaslužno nativno optimiziranje procesa uz pomoć "react-native-fs" biblioteke. Brzina React Native aplikacije u ovom je slučaju usporediva s nativnom aplikacijom. Zanimljivo za primijetiti je da i Flutteru i React Nativeu treba puno više vremena za prvi zapis u datoteku.

React Native i Flutter uvelike su dokazali vrijednost višeplatformskog razvoja mobilnih aplikacija. Daljnjim razvojem ove tehnologije imaju potencijal doživjeti veliki rast i široku upotrebu. Efikasnost i pogodnost višepatformskog razvoja može povećati brzinu izbacivanja proizvoda na tržište. Nikada nije bilo jednostavnije razviti kvalitetnu i oku ugodnu aplikaciju na više mobilnih platformi. Kao negativna strana može se očekivati određen stupanj gubitka u performansama naspram nativnih aplikacija, što je razumljivo i očekivano.

2.2.5 Flutter naspram nativnih aplikacija

Mobilna aplikacija mora biti kvalitetno napravljena da bi mogla konkurirati na tržištu u koje svakog dana ulaze nove aplikacije [13]. Dobre performanse i izgled osnovni su zahtjevi razvoja mobilne aplikacije. Postoji mnogo opcija za razvoj, a jedna od njih je razvoj na-

tivnih aplikacija, za koje se smatra da imaju bolje performanse i održivost. Druga opcija je alat koji koristi jednu kodnu bazu za više platformi, što čini razvoj jednostavnijim. Na samom početku procesa razvoja mobilne aplikacije programeri trebaju odlučiti: razviti više nativnih aplikacija za različite operativne sustave ili jedinstvenu višeplatformsku aplikaciju. Najbolji način za usporediti ova dva pristupa jest razviti aplikaciju na oba načina i izmjeriti rezultate na temelju nekoliko bitnih značaljki. U jednom istraživanju [14] uspoređuju se uporaba procesora, trošenje memorije, veličina aplikacije i drugi pokazatelji performansi aplikacije. Također se provela anketa među programerima radi ocjenjivanja stavova prema višeplatformskom i nativnom razvoju.

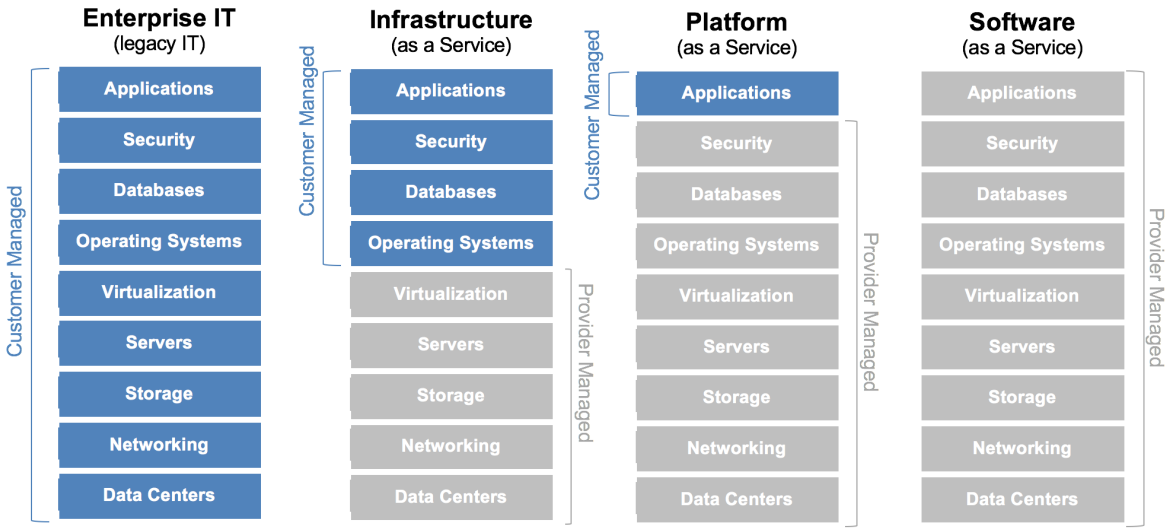
Za uspoređivanje performansi koristili su se sljedeći parametri: korištenje CPU (engl. *Central Processing Unit*), korištenje memorije, potrošnja energije, i FPS (engl. *Frames Per Second*). Razvijene su dvije aplikacije korištenjem Androida i Fluttera. Aplikacije su jednostave, sastoje se od prijave, registracije te početnog zaslona.

Iz rezultata može se zaključiti da Android ima prednost u parametrima korištenje CPU i memorije pošto ih koristi manje od Fluttera. Što se tiče potrošnje energije, obje tehnologije troše jednako. Po pitanju FPS-a, Flutter ima prednost. Anketa je bila provedena na 100 programera koji rade u različitim firmama s različitim vremenskim iskustvom (6 mjeseci do 8 godina). 51% ispitanika zadovoljno je s performansama Fluttera, dok 49% preferira Android. Što se tiče podrške programerske zajednice, većina se okreće Androidu, a razlog može biti što je stariji od Fluttera. Anketa također nalaže da je Flutter nešto lakši za naučuti od Androida.

2.3. Usluge BaaS (engl. *Backend as a service*)

Vrijednost podataka s vremenom sve više dobiva na značenju. Računala u oblaku šire se kao temelj za pružanje raznih programskih rješenja i usluga s bilo koje lokacije te za pohranu i upravljanje ogromnim količinama podataka. Koncept računarstva u oblaku idući je korak u svijetu distribuiranog računarstva. Cilj je ovog računalnog modela efikasnije iskoristiti distribuirane resurse, uklopiti ih radi postizanja boljih performansi i moći se suočiti s velikim i kompliciranim problemima [15].

Na slici 2.5 kategorizirane su arhitekture oblaka na temelju usluga koje pružaju. Generalno ih se može podijeliti na tri kategorije: infrastruktura kao usluga, platforma kao usluga i programska podrška kao usluga. Ove usluge dostupne su u stvarnom vremenu preko interneta. Usluge pružene određenim modelom označene su sivom bojom, dok je plavom označeno ono za što se korisnik mora sam pobrinuti.



Slika 2.5: *Arhitektura oblaka na temelju pruženih usluga [16]*

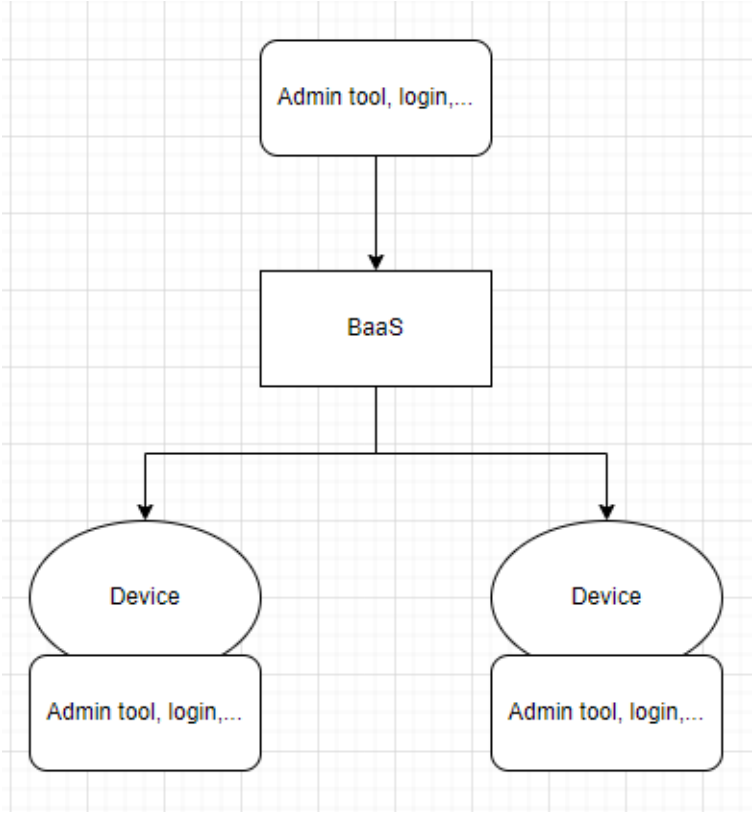
Usluge SaaS (engl. *Software as a service*) koriste zajedničke resurse i jednu instancu koda aplikacije kao i bazu podataka za pružanje usluga većem broju korisnika istovremeno. Nekada se koristi i naziv ASP (engl. *Application Service Provider*) model. Usluga je pružana korisnicima direktno, kao na primjer Dropbox i Google docs.

Usluge PaaS (engl. *Platform as a service*) pružaju programerima platformu koja uključuje sve sisteme i okruženja koja su potrebna za razvijanje programske podrške. U usporedbi s tradicionalnim razvijanjem programske podrške, ovaj pristup može uštediti vrijeme, pruža stotine stalno dostupnih alata i servisa te omogućuje rast velikom brzinom. Primjer je platforme kao usluge Microsoftov Azure.

Usluge IaaS (engl. *Infrastructure as a service*) su pružanje računalne infrastrukture kao usluge. Pored velike fleksibilnosti, ključna je prednost infrastrukture kao usluge plaćanje na temelju korištenja. Ovo omogućava korisnicima da plaćaju onoliko koliko koriste te im daje mogućnost plaćanja za više usluga u skladu s rastom njihovih projekata. Bitne su prednosti također i pristup najnovijim tehnologijama te odsutnost velikog prvotnog troška koji je prisutan bez korištenja IaaS usluga.

Uz ovo se u razvoju mobilnih aplikacija koristi i BaaS (engl. *Backend as a service*), koji se može opisati kao korištenje podataka u oblaku. Glavni razlog zašto programeri trebaju server pri razvijanju aplikacije jest slanje i primanje podataka. Na slici 2.6 može se vidjeti arhitektura BaaS usluge koja se koncentrira na komunikaciju između uređaja. Pri ovome se misli na prijavu i registraciju korisnika, spremanje podataka unutar aplikacije, prikupljanje zapisa i drugo. Kako bi se moglo slati i primati podatke, mora se odabrati odgovarajući okvir te bi programska podrška trebala biti optimizirana za podršku tog okvira. Da se iz-

bjegne ovaj mukotrpan zadatak, koriste se BaaS usluge koje rješavaju problem kombiniranja programske podrške i uređaja koje korisnici koriste. Primjer je BaaS usluge Firebase koji se koristi u sklopu ovog rada.



Slika 2.6: *Primjer BaaS usluge*

3. MOBILNA APLIKACIJA SREDNJA ŠKOLA DONJI MIHOLJAC

U ovom poglavlju bit će opisani razlozi izrade aplikacije, ciljana skupina korisnika, očekivanja od aplikacije, teoretski model aplikacije, funkcionalni zahtjevi aplikacije koje je definirala Srednja škola Donji Miholjac, nefunkcionalni zahtjevi aplikacije te građa aplikacije u vidu korištenih mapa i datoteka.

3.1. Razlozi izrade aplikacije

Škola ima potrebu redovno informirati učenike, što je danas najbolje činiti u digitalnom obliku. Budući da učenici rijetko posjećuju mrežnu stranicu škole, a profili škole na društvenim mrežama Facebook i Instagram sve su manje posjećeni zbog prelaska na nove društvene mreže, Srednja škola Donji Miholjac došla je na ideju izrade mobilne aplikacije s obzirom na to da je učenicima mobitel danas produžena ruka te bi im informacije na takav način bile stalno i brzo dostupne. Škola također od rujna 2022. godine novu školsku godinu započinje u novoizgrađenoj zgradi i sportskoj dvorani te aplikacijom želi dodatno osuvremeniti rad i uvesti inovacije poput školske aplikacije i tako započeti novo doba škole.

3.2. Ciljana skupina korisnika

Aplikacija je namijenjena prvenstveno učenicima, ali i njihovim roditeljima. Učenike kojima je ona namijenjena dijele se u dvije skupine:

- Učenici škole, s podskupinom unutar njih – učenicima završnih razreda, maturantima kojima su pored uobičajenih informacija potrebne i dodatne informacije vezane uz završne radove i državnu maturu.
- Učenici 8. razreda osnovnih škola koji bi putem aplikacije mogli dobiti sve potrebne informacije koje bi im olakšale odluku o odabiru zanimanja, programa koji žele upisati u srednjoj školi te sve informacije vezane uz upise.

3.3. Zahtjevi na mobilnu aplikaciju

Funkcionalni su zahtjevi značajke i funkcionalnosti proizvoda koje programeri moraju implementirati da bi korisnici mogli učinkovito koristiti proizvod [17]. Bitno je precizno definirati funkcionalne zahtjeve komunikacijom između korisnika i projektnih upravitelja. Aplikacija treba biti besplatno dostupna ciljanim skupinama i bez oglasa koji se u njoj pojavljuju. Osim informacija koje učenici mogu u njoj pronaći, aplikacija ih treba posebno informirati o najnovijim, važnim obavijestima (kao što su obavijesti o izmjeni rasporeda, događanjima u

školi, zamjenama za nastavnike...). Stavljanje obavijesti, dodatnih sadržaja i slično trebalo bi biti omogućeno administratoru određenom od strane škole na relativno jednostavan način.

3.3.1 Funkcionalni zahtjevi na mobilnu aplikaciju

Funkcionalni zahtjevi aplikacije Srednje škola Donji Miholjac, koji će se nadalje nazivati funkcionalnostima aplikacije, su sljedeći:

- Općenito o školi: kratka povijest, par fotografija stare i nove zgrade škole.
- Oglasna ploča: važne obavijesti poput izmjene rasporeda, zamjene za nastavnike i slično o kojima bi aplikacija učenike zasebno i obavještavala.
- Programi i zanimanja: opis svih programa i zanimanja za koje škole obrazuje učenike s informacijama o predmetima koji se uče, vještinama i ishodima koje će učenici usvojiti.
- Raspored: raspored sati, zvona te učionica i kabineta u novoj školi, raspored prostorija, kalendar škole, popis udžbenika.
- Događanja: kratke informacije o provedenim aktivnostima u školi – iste se objavljuju i na mrežnoj stranici škole.
- Projekti: prikaz projekata koje škola realizira, kratke informacije popraćene kojom fotografijom (poput ERASMUS+ projekata i slično).
- Učenička zadruga: prikaz rada učeničke zadruge kroz kratke informacije i s pratećim fotografijama.
- Upisi u 1.razrede: informacije vezane uz upise u 1.razred kao što su programi i zanimanja koja se upisuju u nadolazećoj školskoj godini, uvjeti za upise (predmeti koji se boduju i slično), rokovi.
- Završni razredi: teme završnih radova, rokovi, informacije o državnoj maturi.
- Knjižnica: informacije o radu knjižnice.
- Kontakt podaci: adresa, email, telefoni.

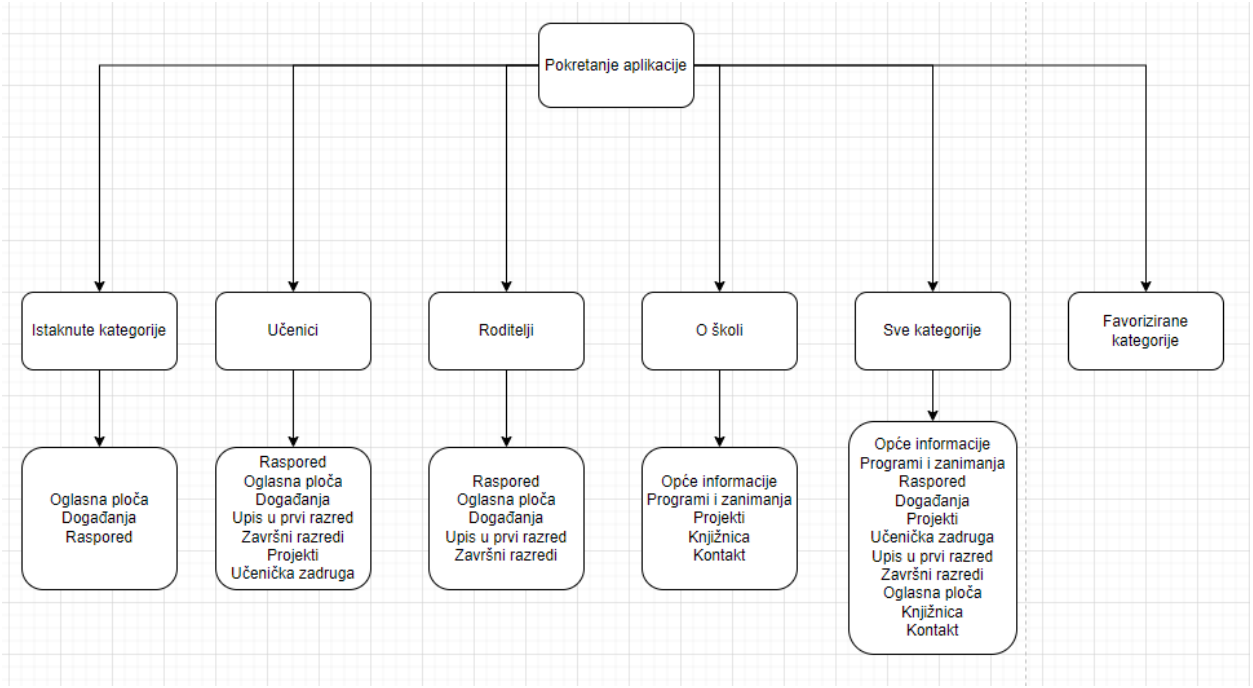
3.3.2 Nefunkcionalni zahtjevi na mobilnu aplikaciju

Nefunkcionalni zahtjevi su neizostavan dio svakog programskog rješenja i kritični su za uspjeh projekta [17]. Nefunkcionalni zahtjevi mogu se opisati kao zahtjevi koji definiraju atribute sustava kao što su sigurnost, pouzdanost, izvođenje, održivost, skalabilnost i upotrebljivost. Još ih se može opisati i kao zahtjeve koji određuju kriterije po kojima se ocjenjuje rad sustava, a ne određena ponašanja.

Aplikacija je dizajnirana da bude lako održiva i skalabilna. Funkcionalnosti aplikacije lako se mogu dodavati i uklanjati bez negativnih učinaka na druge dijelove sustava. Obavijesti i događanja implementirana su na način da se dodavanjem nove obavijesti ili događaja na mrežnu stranicu škole ona automatski pojavljuju u aplikaciji, što čini cijeli proces izravnim i jednostavnim. Aplikacija je brza i laka za korištenje svim korisnicima.

3.4. Model mobilne aplikacije

Funkcionalnosti aplikacije podijeljene su na četiri odjeljka s obzirom na to koje funkcionalnosti interesiraju određen tip korisnika, a to su: "Istaknuto", "Učenici", "Roditelji" i "O školi", no svaki tip korisnika ima pristup svim funkcionalnostima. Korisnici također mogu dodavati funkcionalnosti u "favorite" za što postoji poseban odjeljak, kao i poseban odjeljak za sve funkcionalnosti. U odjeljku "Istaknute kategorije" prikazane su fukncionalnosti koje će korisnici najčešće upotrebljivati. Na slici 3.1 prikazan je dijagram toka svih funkcionalnosti aplikacije.



Slika 3.1: *Dijagram toka aplikacije*

3.5. Građa mobilne aplikacije

Datoteke aplikacije podijeljene su na mape s obzirom na njihovu ulogu, što je vidljivo na slici 3.2. U mapi *app_functionalities* nalaze se zasloni svih funkcionalnosti aplikacije. U mapi *models* nalaze se često korištene klase koje služe kao modeli. U mapi *projects* nalaze se klase koje predstavljaju projekte škole. U mapi *widgets* nalaze se najbitniji *widgeti* koji služe kao kostur aplikacije.

(a) *Mapa app_functionalities*

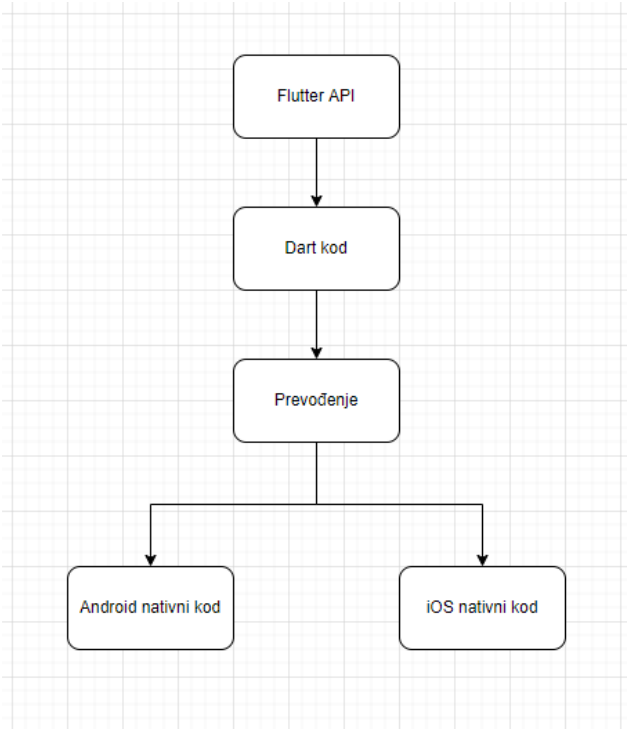
(b) *Mapa models*

(c) *Mapa projects*

(d) *Mapa widgets*

Slika 3.2: *Građa aplikacije*

Flutter SDK prevodi ovaj Dart kod u nativni Android i nativni iOS kod. Dijagram ovog procesa se može vidjeti na slici 3.3.



Slika 3.3: *Dijagram toka procesa prevođenja*

4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE

U ovom poglavlju bit će predstavljeni alati i tehnologije korišteni u izradi višeplatformske aplikacije te programsko rješenje aplikacije.

4.1. Korišteni alati i tehnologije

4.1.1 Flutter SDK (engl. *Software Development Kit)*

Flutter je [18] besplatan Googleov alat otvorenog koda pušten u svibnju 2017. godine. Dozvoljava programerima stvaranje nativnih mobilnih aplikacija sa samo jednom kodnom bazom. Ovo znači da se može koristiti jedan programski jezik za stvaranje dviju različitih aplikacija, jedne za iOS te jedne za Android. Flutter sačinjava dva bitna dijela: SDK (engl. *Software Development Kit*), kolekciju alata koji pomažu u izradi aplikacije, što uključuje alate za prevođenje koda u nativni strojni kod te okvir, kolekciju elemenata korisničkog sučelja za ponovnu uporabu kao što su dugma, polja za unos teksta, klizači i drugi.

4.1.2 Dart

U Flutteru [18] su sve aplikacije napisane u Dart programskom jeziku. Dart je programski jezik napisan i održavan od strane Googlea. Fokusira na *front-end* razvoj, a koristi se za stvaranje mobilnih i mrežnih aplikacija. Objektno je orjentiran jezik baziran na klasama s ugrađenim skupljanjem smeća. Dart je razvijen da nadomjesti i naslijedi JavaScript te zbog toga implementira većinu bitnih značajki JavaScripta. Da privuče programere koji nisu upoznati sa JavaScriptom, Dart ima sintaksu sličnu Javi.

4.1.3 Visual Studio Code

Visual Studio Code jednostavan je, no moćan alat za uređivanje izvornog koda dostupan za Windows, macOS i Linux. Ima ugrađenu podršku za JavaScript, Typescript i Node.js te bogat sistem nadogradnji za druge jezike kao što su C++, C#, Java, Python, PHP, Go, .NET, Dart i drugi.

4.1.4 Format RSS

RSS (engl. *RDF Site Summary*) format je koji služi za dijeljenje sadržaja mrežnih stranica koji su dostupni za distribuciju iz mrežnog izdavača korisnicima internetske mreže [19]. RSS je primjena XML-a (engl. *Extensible Markup Language*) koji se pridržava W3C-ovog (engl. *World Wide Web Consortium*) RDF-a (engl. *Resource Description Framework*). Izvorno je izrađen od strane Netscapea za svrhe svog mrežnog preglednika, no u današnje je vrijeme

otvoren za korištenje. Mrežna stranica koja želi objaviti neke od svojih sadržaja, kao na primjer naslove vijesti ili članke, stvara opis sadržaja i precizira gdje se taj sadržaj nalazi u obliku RSS dokumenta. Izdavačka stranica potom registrira svoj RSS dokument s jednim od nekoliko postojećih RSS izdavača. Korisnik s internetskim preglednikom ili posebnim programom za čitanje RSS sadržaja može čitati periodično dostupne distribucije. Stranice koje služe za čitanje RSS datoteka nazivaju se agregatori sadržaja. Primjer RSS dokumenta vidljiv je na slici 4.1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
 <title>RSS Title</title>
 <description>This is an example of an RSS feed</description>
 <link>http://www.example.com/main.html</link>
 <copyright>2020 Example.com All rights reserved</copyright>
 <lastBuildDate>Mon, 6 September 2010 00:01:00 +0000</lastBuildDate>
 <pubDate>Sun, 6 September 2009 16:20:00 +0000</pubDate>
 <ttl>1800</ttl>

 <item>
  <title>Example entry</title>
  <description>Here is some text containing an interesting description.</description>
  <link>http://www.example.com/blog/post/1</link>
  <guid isPermaLink="false">7bd204c6-1655-4c27-aeee-53f933c5395f</guid>
  <pubDate>Sun, 6 September 2009 16:20:00 +0000</pubDate>
 </item>

</channel>
</rss>
```

Slika 4.1: *Primjer RSS dokumenta [20]*

4.1.5 Firebase

Firebase [21] je NoSQL baza podataka u oblaku koja sinkronizira podatke za sve klijente u stvarnom vremenu i pruža iznavmrežnu funkcionalnost. Podaci se pohranjuju u JSON formatu, a svi klijenti dijele jednu instancu te automatski primaju ažuriranja s novim podacima. Korištenje Firebase baze podataka iz programerske je perspektive drugačije od tradicionalnih relacijskih baza podataka. Primarno, razlika je strukturiranje podataka, umjesto u tablice podaci se spremaju u JSON formatu. Također, za razliku od tradicionalnog građenja aplikacije, koje čini bazu podataka obavijenu pristupnim slojem na kojem se odvijaju pozivi i spremaju procedure, koju se onda poveže s mobilnom aplikacijom, kada se koristi Firebase baza podataka, sav kod se nalazi u mobilnom klijentu. Ovo pojednostavljuje građu i održavanje aplikacije.

4.2. Programsko rješenje mobilne aplikacije

Ovo poglavlje sadrži objašnjenja i primjere programskog koda koji je korišten radi ostvarivanja zadanih funkcionalnih zahtjeva. Datoteka *main.dart* početna je točka aplikacije. Ona sadrži funkciju *main* u kojoj se nalazi funkcija *runApp*, te klasu *MyApp*. Funkcija *runApp* služi za podizanje i stavljanje na zaslon predanog joj *widgeta*. Funkcija *main* sadrži i par linija koda odgovornih za čuvanje ekrana u uspravnom položaju. Klasa *MyApp* sadrži funkciju *build*. Funkcija *build* često je korištena funkcija koju okvir poziva kada se *widget* ubaci u stablo i svaki puta kada dođe do promjene stanja unutar klase, što omogućava prikaz dinamične promjene stanja, a temelji se na kontekstu predanom funkciji. Funkcija *build* vraća objekt klase *MaterialApp* koja sadrži većinu bitnih *widgeta* korištenih za razvoj aplikacije, no u ovom slučaju u njoj se definiraju naslov, tema i rute aplikacije, što se može vidjeti na slici 4.2.

```dart
void main() {
  WidgetsFlutterBinding.ensureInitialized();
  SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        title: 'Srednja škola Donji Miholjac',
        theme: appTheme,
        initialRoute: '/',
        routes: routes);
  }
}
```

Slika 4.2: *Funkcija main i klasa MyApp*

Na slici 4.3 može se vidjeti sadržaj datoteke *app_theme.dart*. U njoj je definirana varijabla *appTheme*, koja sadrži objekt klase *ThemeData* u kojoj su zadane pozadinska boja aplikacije, font te boja teksta.

```
var appTheme = ThemeData(
  canvasColor: HexColor('E5EFF3'),
  fontFamily: 'Raleway',
  textTheme: ThemeData.light().textTheme.copyWith(
        bodyText1: const TextStyle(
          color: Color.fromRGBO(20, 51, 51, 1),
        ), // TextStyle
        bodyText2: const TextStyle(
          color: Color.fromRGBO(20, 51, 51, 1),
        ), // TextStyle
      ),
); // ThemeData
```

Slika 4.3: *Varijabla appTheme*

Datoteka *routes.dart* sadrži varijablu *routes*, koja je objekt klase *Map* u kojoj su definirane rute do svih zaslona u aplikaciji. Kao što se može vidjeti na slici 4.4 u prvom retku, inicijalna je ruta ona do klase *Splash*, koja predstavlja *splash* zaslon aplikacije. *Splash* zaslon služi kao uvodni zaslon pri ulasku u aplikaciju. Najčešće sadržava sliku, logo ili trenutnu verziju aplikacije, ovisno o željama korisnika. Služi i kao zaslon koji korisnik vidi dok se učitavaju pozadinski elementi. Klasa *Splash* je *stateful widget*, a njezina *state* klasa sadrži metode *initState* i *navigateToHome*, koje se mogu vidjeti na slici 4.5. Metoda *initState* služi za ubacivanje objekta u *widget* stablo, a metoda *navigateToHome* definira koliko će vremena biti prikazan *splash* zaslon te koji će zaslon biti prikazan pri završetku tog vremena, to jest početni zaslon aplikacije.

```
var routes = {
  '/': (context) => const Splash(),
  'aboutSchool': (context) => const AboutSchool(),
  'programDescription': (context) => const ProgramDescription(),
  'scedule': (context) => const Scedule(),
  'events': (context) => const Events(),
  'projects': (context) => const Projects(),
  'studentAssociation': (context) => const StudentAssociation(),
  'firstGrade': (context) => const FirstGrade(),
  'lastGrade': (context) => const LastGrade(),
  'notifications': (context) => const Notifications(),
  'informationAndContact': (context) => const InformationAndContact(),
  'library': (context) => Library(),
  'favourites': (context) => const Favourites(),
  'allCategories': (context) => AllCategories(),
  'epas': (context) => const EpasSkola(),
  'eTwinning': (context) => const ETwinning(),
  'financialLiteracy': (context) => FinancialLiteracy(),
  'digitalMagazine': (context) => const DigitalMagazine(),
  'volunteers': (context) => const Volunteers(),
  'advancementSupportProgram': (context) => const AdvancementSupportProgram(),
  'schoolBees': (context) => const SchoolBees(),
  'internationalCooperation': (context) => const InternationalCooperation(),
};
```

Slika 4.4: *Varijabla routes*

```
void initState() {
  super.initState();
  _navigateToHome();
}

_navigateToHome() async {
  await Future.delayed(const Duration(milliseconds: 3000), () {});
  Navigator.pushReplacement(
      context, MaterialPageRoute(builder: (context) => MyNavigationBar()));
}
```

Slika 4.5: *Metode initState i navigateToHome*

Widget MyNavigationBar početni je zaslon aplikacije do kojeg vodi *navigateToHome* metoda. *MyNavigationBar* je *stateful widget* čija *state* klasa sadrži *build* metodu koja vraća objekt klase *Scaffold*. *Scaffold* klasa često je korištena klasa koja implementira osnovnu vizualnu strukturu materijalnog dizajna. Ova klasa nudi funkcionalnost navigacijske trake koja nam je potrebna. U klasi *BottomNavigationBar* definirane su ikone i naslovi modula za različite korisnike aplikacije. Klikom na jednu od stavki navigacijske trake poziva se metoda *onItemTapped* unutar koje se poziva metoda *setState*, koja služi za obavještavanje okvira da je došlo do promjene stanja unutar objekta. Pošto je varijabla *selectedIndex* postavljena na nulu, zaslon koji će korisnik vidjeti pri ulasku u aplikaciju onaj je s indeksom nula u listi *widgetOptions*, a to je zaslon "istaknute kategorije". Na slici 4.6 može se vidjeti *build* metoda klase *MyNavigationBarState*, a na slici 4.7 sama klasa *MyNavigationBarState* te *onItemTapped* metoda.

```dart
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: _widgetOptions.elementAt(_selectedIndex),
    ), // Center
    bottomNavigationBar: BottomNavigationBar(
      type: BottomNavigationBarType.fixed,
      items: const <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(Icons.star),
          label: 'Istaknuto',
        ), // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(PhosphorIcons.eyeglasses),
          label: 'Učenici',
        ), // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(PhosphorIcons.users),
          label: 'Roditelji',
        ), // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(PhosphorIcons.house_line),
          label: 'O školi',
        ), // BottomNavigationBarItem
      ], // <BottomNavigationBarItem>[]
      currentIndex: _selectedIndex,
      selectedItemColor: HexColor('#006D77'),
      onTap: _onItemTapped,
    ), // BottomNavigationBar
  ); // Scaffold
}
```

Slika 4.6: *Build metoda klase MyNavigationBarState*

```dart
class _MyNavigationBarState extends State<MyNavigationBar> {
  static final List<Widget> _widgetOptions = <Widget>[
    SelectionScreen(
        gridTitle: 'Istraži istaknute kategorije',
        shownFunctionalities: MyNavigationBar()._highlightedFunctionalities), // SelectionScreen
    SelectionScreen(
        gridTitle: 'Istraži kategorije',
        shownFunctionalities: MyNavigationBar()._studentFunctionalities), // SelectionScreen
    SelectionScreen(
        gridTitle: 'Istraži kategorije',
        shownFunctionalities: MyNavigationBar()._parentFunctionalities), // SelectionScreen
    SelectionScreen(
        gridTitle: 'Istraži kategorije',
        shownFunctionalities: MyNavigationBar()._schoolFunctionalities), // SelectionScreen
  ]; // <Widget>[]

  int _selectedIndex = 0;

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }
```

Slika 4.7: *Lista dostupnih modula i onItemTapped metoda*

Izgled osnovnih modula "Istaknuto", "Učenici", "Roditelji" i "O školi" implementirana je u klasi *SelectionScreen*. Atributi su ove klase naslov liste funkcionalnosti i lista prikazanih funkcionalnosti koji se mjenjaju ovisno o odabranom artiklu u navigacijskoj traci. *Build* metoda ove klase vraća objekt klase *Scaffold*, unutar kojeg su definirani gumb za prikaz favorita, gumb za prikaz svih funkcionalnosti te lista prikazanih funkcionalnosti koja je objekt klase *CategoryGrid*. Klasa *CategoryGrid* prikazana je na slici 4.8.

```dart
class CategoryGrid extends StatelessWidget {
  final List<Functionality> _shownFunctionalities;
  const CategoryGrid(this._shownFunctionalities, {Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Expanded(
      child: GridView(
        padding:
            const EdgeInsets.only(bottom: 30, left: 30, right: 30, top: 15),
        gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(
          crossAxisSpacing: 20,
          mainAxisSpacing: 30,
          maxCrossAxisExtent: 200,
        ), // SliverGridDelegateWithMaxCrossAxisExtent
        children: _shownFunctionalities
            .map((funData) => FunctionalityItem(
                functionality: funData)) // FunctionalityItem
            .toList(),
      ), // GridView
    ); // Expanded
  }
}
```

Slika 4.8: *Klasa CategoryGrid*

Ova klasa služi za prikaz liste prikazanih funkcionalnosti u obliku *grida*. Lista funkcionalnosti *mapira* se u listu objekata klase *FunctionalityItem*, što omogućuje jednostavno dodavanje i brisanje kategorija ovisno o željama korisnika.

Klasa *FunctionalityItem* ima kao atribut klasu *Functionality*, koju se može vidjeti na slici 4.9. Svaka je funkcionalnost jednoznačno određena svojim id-om, ima naslov, sliku te rutu do zaslona asociranog tom funkcionalnošću. Klasa *FunctionalityItem* predstavlja gumb koji vodi do zaslona svake funkcionalnosti te sadrži logiku dodavanja funkcionalnosti u favorite. U svrhu ovoga deklarirana je prazna lista u koju će se dodavati favorizirane funkcionalnosti. Metoda *isInFavourites* provjerava nalazi li se funkcionalnost u listi favorita te ju se može vidjeti na slici 4.10.

```dart
class Functionality {
  final String _id;
  final String _title;
  final SvgPicture _image;
  final String _route;

  const Functionality(this._id, this._title, this._image, this._route);

  String get getId {
    return _id;
  }
  String get getTitle {
    return _title;
  }
  SvgPicture get getImage {
    return _image;
  }
  String get getRoute {
    return _route;
  }
}
```

Slika 4.9: *Klasa functionality*

```dart
bool isInFavourites() {
  return favouriteFunctionalities.contains(functionalities[
      functionalities.indexWhere((element) => element.id == widget.id)]);
}
```

Slika 4.10: *Metoda isInFavourites*

Kada se stisne gumb za dodavanje funkcionalnosti u favorite, prvo se provjerava je li ona već u favoritima. Ako nije dodaje se, a ako je briše se iz favorita. U isto vrijeme se mijenja izgled ikone koja služi kao gumb za dodavanje u favorite, što se može vidjeti na slici 4.11.

```dart
IconButton(
    onPressed: _isInFavourites()
        ? () {
            setState(() {
              favouriteFunctionalities.remove(matchedFunctionality);
            });
          }
        : () {
            setState(() {
              favouriteFunctionalities.add(matchedFunctionality);
            });
          },
    icon: _isInFavourites()
        ? Icon(Icons.favorite,
            color: HexColor('#006D77'))
        : Icon(Icons.favorite_border,
            color: HexColor('#006D77')))
```

Slika 4.11: *Gumb za dodavanje funkcionalnosti u favorite*

Klasa *FunctionalityScreen*, koja se može vidjeti na slici 4.12, služi kao model zaslona svake funkcionalnosti. Kao atribute ima naslov prikazan na aplikacijskoj traci i tijelo koje se prikazuje na zaslonu korisnicima. Objekt klase *AppBar* koji se prikazuje na vrhu *Scaffolda* ima

funkcionalnost vraćanja na prijašnji zaslon.

```dart
class FunctionalityScreen extends StatelessWidget {
  final String pageTitle;
  final Widget child;

  const FunctionalityScreen({Key? key, required this.pageTitle, required this.child}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(pageTitle),
        backgroundColor: HexColor('#006D77'),
      ), // AppBar
      body: child,
    ); // Scaffold
  }
}
```

Slika 4.12: *Klasa FunctionalityScreen*

Problem automatskog prikazivanja obavijesti i događanja unutar aplikacije riješio se korištenjem RSS-a (engl. *RDF Site Summary*). U te svrhe izrađena je klasa *ParseRssToList* koja sadrži metodu *fetchNews*, vidljivo na slici 4.13, koja dekodira artikle iz RSS dokumenta te ih mapira u listu objekata klase *NotificationModel*, što se može vidjeti na slici 4.15.

```dart
Future fetchNews() async {
  final response = await http.get(rssUrl);
  if (response.statusCode == 200) {
    var decoded = RssFeed.parse(response.body);
    return decoded.items
        ?.map((item) => NotificationModel(
              title: item.title,
              description: item.description,
              link: item.link,
              pubDate: item.pubDate))
        .toList();
  } else {
    throw const HttpException('Failed to fetch the data');
  }
}
```

Slika 4.13: *Metoda FetchNews*

```dart
class NotificationModel {
  final String? title;
  final String? description;
  final String? link;
  final DateTime? pubDate;

  NotificationModel(
      {required this.title,
      required this.description,
      required this.link,
      required this.pubDate});
}
```

Slika 4.14: *Klasa NotificationModel*

Ovime je osigurano da će se, kada administrator mrežne aplikacije Srednja škola Donji Miholjac doda novu obavijest ili događaj, oni automatski prikazati u mobilnoj aplikaciji.

5. PRIKAZ KORIŠTENJA I ISPITIVANJE RADA MOBILNE APLIKACIJE

U ovom poglavlju bit će prikazano korištenje aplikacije, to jest neki od bitnijih zaslona koje će korisnici najčešće koristiti. Također će biti opisana analiza programskog rješenja.

5.1. Prikaz korištenja aplikacije

Nakon instaliranja i pokretanja mobilne aplikacije, korisnika dočekuje *splash* zaslon sa slikom i imenom škole. Potom aplikacija vodi korisnika do početnog zaslona, to jest zaslona s istaknutim kategorijama, što se može vidjeti na slici 5.1.

(a) *Splash zaslon* **(b)** *Zaslon istaknuto*

Slika 5.1: *Splash zaslon i početni zaslon aplikacije*

Korisnik potom ima nekoliko opcija: otići na zaslon sa svim kategorijama pritiskom na "Prikaži sve", otići na zaslon s favoriziranim kategorijama pritiskom na ikonu srca te prolazak po modulima na navigacijskoj traci, što je vidljivo na slici 5.2.

(a) *Zaslon "Sve kategorije"*

(b) *Zaslon "Učenici"*

(c) *Zaslon "Roditelji"*

(d) *Zaslon "O školi"*

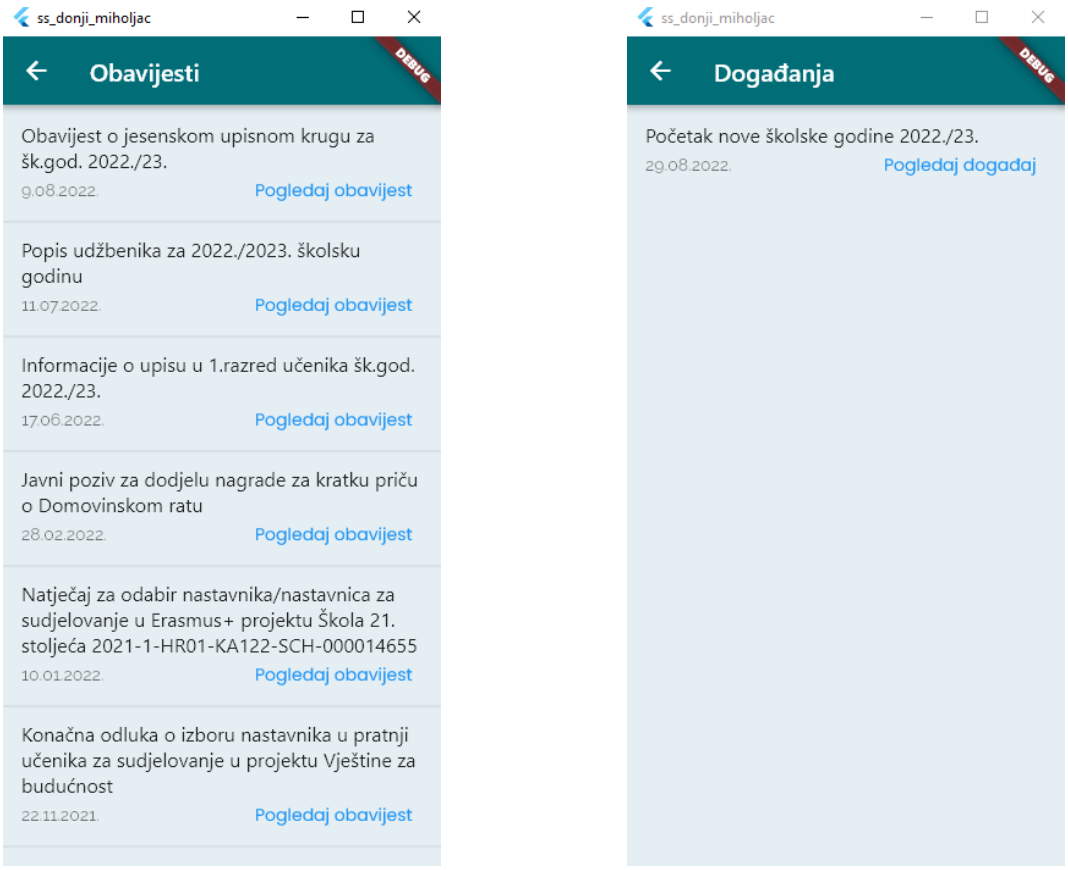
Slika 5.2: *Zasloni aplikacije*

Dodavanje kategorija u favorite je jednostavno. Klikom na ikonu srca pored naslova kategorije ta se kategorija dodaje u favorite te se ponovnim pritiskom briše iz favorita, što je prikazano slikom 5.3.

(a) *Odabir favorita* **(b)** *Zaslon "Favorizirane kategorije"*

Slika 5.3: *Biranje favoriziranih kategorija*

Odabirom kategorije "Oglasna ploča" ili "Događanja" korisniku se prikazuje zaslon s obavijestima ili događanjima poredanima od novijih prema starijima. Pritiskom na "Pogledaj obavijest" ili "Pogledaj događaj" korisnik je preusmjeren na obavijest ili događaj na mrežnoj stranici škole, što je vidljivo na slici 5.4.



(a) *Prikaz obavijesti* **(b)** *Prikaz događaja*

Slika 5.4: *Obavijesti i događanja*

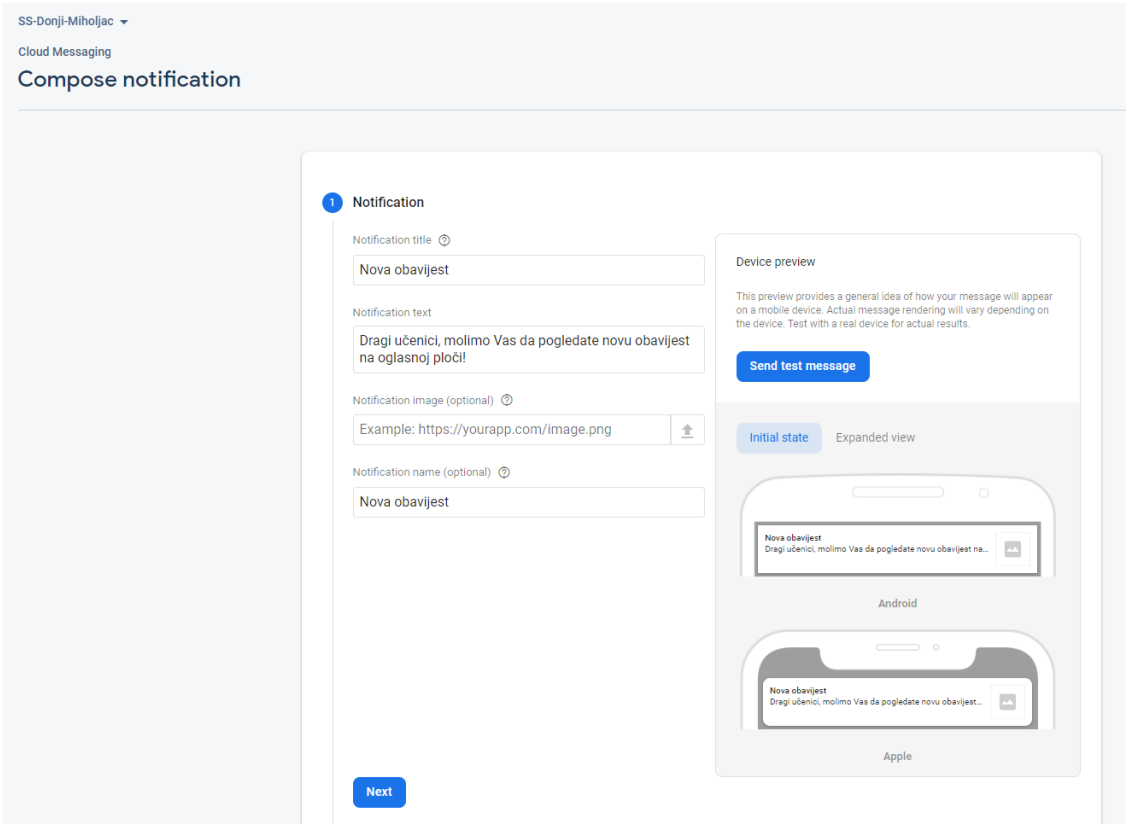
Odabirom kategorije "Projekti" prikazuje se zaslon koji sadrži izbornik za detalje o svim projektima u koje je škola uključena. Zasloni projekata u tekstualnom su obliku s linkovima na mrežne stranice ili listu događaja, ovisno o tome na koji način su implementirani na mrežnoj stranici škole. Programi i zanimanja koje nudi škola odvojeni su na trogodišnja i četverogodišnja zanimanja te pritiskom na link aplikacija vodi korisnika na PDF dokument s detaljima svakog programa i zanimanja, što je prikazano na slici 5.5.

(a) *Prikaz projekata*

(b) *Prikaz programa i zanimanja*

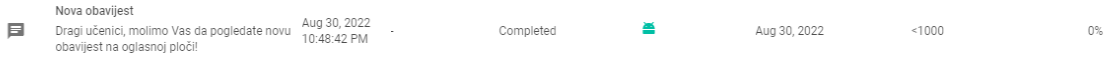
Slika 5.5: *Projekti, programi i zanimanja koje škola nudi*

U svrhu testiranja slanja *push* notifikacija stvoren je projekt na Firebaseu. Korištenjem *Firebase Cloud Messaging* servisa lako je slati *push* obavijesti svim korisnicima aplikacije. Primjer stvaranja obavijesti u Firebaseu prikazan je na slici 5.6.

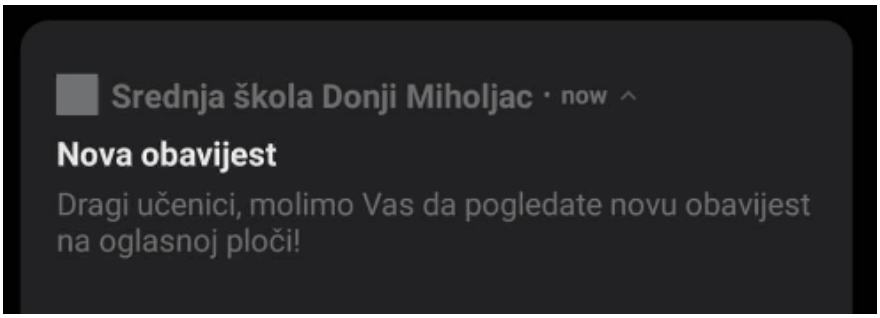


Slika 5.6: *Stvaranje nove push obavijesti u Firebase-u*

Push obavijest potom se pojavljuje na mobilnom uređaju, što se vidi na slici 5.8.



Slika 5.7: *Prikaz nove push obavijesti u Firebaseu*



Slika 5.8: *Prikaz push obavijesti na mobilnom uređaju*

5.2. Analiza programskog rješenja mobilne aplikacije

Analizom i testiranjem programskog rješenja ustanovljeno je da aplikacija pravilno dohvaća podatke s mrežne stranice škole te ih točno prikazuje unutar aplikacije. Pošto je aplikacija višeplatformska, testirana je na Android i iOS mobilnim operacijskim sustavima te ima jednake performanse na oba sustava. Aplikacija ne koristi nikakve osobne podatke o učenicima, stoga nema zabrinutosti oko sigurnosti podataka.

6. ZAKLJUČAK

U ovom završnom radu razvijena je višeplatformska mobilna aplikacija za Srednju školu Donji Miholjac. Aplikacija služi učenicima kao brz i pristupačan izvor svih informacija vezanih uz aktivnosti škole. Aplikacija je višeplatformna, što znači da je dostupna korisnicima i Andrioid i iOS mobilnih uređaja. Učenicima je omogućeno praćenje rasporeda, praćenje obavijesti i događaja, informacije vezane za upis u prvi razred, informacije vezane za završne razrede, mogućnost praćenja projekata u koje je škola uključena, informacije o programima i zanimanjima koje škola nudi, informacije vezane za učeničku zadrugu, knjižnicu i ostalo. Ovo pomaže učenicima u lakšem praćenju novosti vezanih uz školu te u snalaženju u novom srednjoškolskom okruženju.

Aplikacija je izrađena u Visual Studio Code uređivaču teksta. Za razvoj aplikacije korišten je razvojni paket Flutter s programskim jezikom Dart. U ostvarene funkcionalnosti aplikacije spada implementacija svih funkcionalnosti koje je škola propisala. U budućnosti postoji mogućnost nadogradnje i poboljšanja aplikacije u vidu dodavanja tražilice te mogućnost dodavanja novih rasporeda iz godine u godinu bez mijenjanja koda. Također se može implementirati mogućnost otvaranja obavijesti i događanja unutar aplikacije.

LITERATURA

[1] Tržišni udio mobilnih operativnih sustava u svijetu [online], dostupno na: https://gs.statcounter.com/os-market-share/mobile/worldwide, posjećeno: 30.7.2022.

[2] Tržišni udio mobilnih operativnih sustava u Hrvatskoj [online], dostupno na: https://gs.statcounter.com/os-market-share/mobile/croatia, posjećeno: 30.7.2022.

[3] Tržišni udio mobilnih operativnih sustava u SAD-u [online], dostupno na: https://gs.statcounter.com/os-market-share/mobile/united-states-of-america, posjećeno: 30.7.2022.

[4] L. Corral, A. Sillitti, and G. Succi. Mobile Multiplatform Development: An Experiment for Performance Analysis. *Procedia Computer Science*, Vol. 10, pp. 736-743, 2012.

[5] N. Hansson and T. Vidhall. Effects on Performance and Usability for Cross-platform Application Development Using React Native, Final Thesis, Linköpings university, Linköping, Sweden. 2016.

[6] W. Jobe. Native Apps vs. Mobile Web Apps. *International Journal of Interactive Mobile Technologies*, No.4, Vol. 7, pp. 27-29, October 2013.

[7] L. Corral, A. Janes, and T. Remencius. Potential Advantages and Disadvantages of Multiplatform Development Frameworks–a Vision on Mobile Environments. *Procedia Computer Science*, Vol. 10, pp. 1202-1207, 2012.

[8] Pros & cons of cross-platform mobile development: Is it right for your project? [online], dostupno na: https://surf.dev/advantages-and-disadvantages-of-cross-platform-mobile-development, posjećeno: 30.7.2022.

[9] TechTarget Contributor. What is cross-platform mobile development? [online], dostupno na: https://www.techtarget.com/searchmobilecomputing/definition/cross-platform-mobile-development, posjećeno: 12.8.2022.

[10] Flutter architectural overview [online], dostupno na: https://docs.flutter.dev/resources/architectural-overview, posjećeno: 12.8.2022.

[11] W. Wu. React Native vs Flutter, Cross-platform Mobile Application Frameworks, Final Thesis, Metropolia University of Applied Sciences, Helsinki, Finland. 2018.

[12] S. Yusuf. *Ionic Framework By Example.* Packt Publishing Ltd, Birmingham, United Kingdom, 2016.

[13] M. Olsson. A Comparison of Performance and Looks Between Flutter and Native Applications: When to Prefer Flutter Over Native in Mobile Application Development, Final Thesis, Faculty of Computing, Blekinge Institute of Technology, Karlskrona, Sweden, 2020.

[14] H. Hussain, K. Khan, F. Farooqui, Q. A. Arain, and I. F. Siddiqui. Comparative Study of Android Native and Flutter App Development. *Memory*, Vol. 47, pp. 36-37, December 2021.

[15] C. G. Kim. A Study of Utilizing Backend as a Service (baas) Space for Mobile Applications. In *International Conference on Intelligence Science*. Springer, Mount Pleasant, MI, USA, 2019.

[16] Who manages cloud iaas, paas, and saas services [online], dostupno na: https://mycloudblog7.wordpress.com/2013/06/19/who-manages-cloud-iaas-paas-and-saas-services/, posjećeno: 12.8.2022.

[17] M. Glinz. On Non-functional Requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)*. IEEE, Delhi, India, 2007.

[18] Flutter frequently asked questions [online], dostupno na: https://docs.flutter.dev/resources/faq, posjećeno: 12.9.2022.

[19] Z. Çelikbaş. *What Is RSS and How Can It Serve Libraries?* Istanbul University, Istanbul, Turkey, 2004.

[20] RSS [online], dostupno na: https://en.wikipedia.org/wiki/rss, posjećeno: 12.8.2022.

[21] L. Moroney. The Firebase Realtime Database. In *The Definitive Guide to Firebase*. Apress, Seattle, Washington, USA, 2017.

SAŽETAK

U ovom završnom radu razvijena je višeplatformska mobilna aplikacija za Srednju školu Donji Miholjac. Aplikacija je višeplatformska, što znači da je mogu koristiti korisnici različitih operacijskih sustava. U prvom dijelu rada opisane su postojeće, trenutno najkorištenije višeplatformske tehnologije te je napravljena usporedba tih tehnologija. Dana je razrada problema prikaza aktivnosti škole i prikaz postojećih rješenja. Također su opisane usluge *Backend as a service*. Nakon toga opisani su razlozi izrade aplikacije: lakše obavještavanje studenata i pružanje bitnih informacija, ciljana skupina korisnika: učenici škole i učenici 8. razreda osnovnih škola, model i građa aplikacije te funkcionalni i nefunkcionalni zahtjevi aplikacije, što obuhvaća želje škole u vidu svega što bi aplikacija trebala sadržavati. Opisani su korišteni alati i dano je programsko rješenje aplikacije. Aplikacija pruža učenicima sve bitne informacije koje su im potrebne za uspješno snalaženje u školskom životu.

Ključne riječi: Dart, Flutter, mobilna aplikacija, srednja škola, višeplatformski razvoj.

ABSTRACT

Title: Developing a multiplatform mobile application to display high school activities

This paper explains the process of developing a multiplatform application for Donji Miholjac highschool. In the first part of the paper, the existing, currently most used multi-platform technologies are described, and a comparison of these technologies is made. An elaboration is given of the problem of displaying school activities and existing solutions are presented. Backend as a service model is explained. After that, the reasons for creating the application: notifying students easier and displaying important information, the target group of users: students of the school and eight graders, the model and structure of the application, and the functional and non-functional requirements, which includes wishes of the school as to what the application has to contain. The tools used are described and the program solution of the application is given. The application provides students with all the essential information they need to successfully navigate school life.

Keywords: Dart, Flutter, mobile application, high school, cross platform development.

ŽIVOTOPIS

Lovro Dijanović rođen je 26. veljače 1998. godine u Osijeku. Pohađao je Osnovnu školu "Mladost" u Osijeku. Potom upisuje III. Gimnaziju u Osijeku koju završava 2017. godine. Upisuje preddiplomski studij računarstva 2019. godine na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

Prilog 1. Dokument završnog rada u .docx formatu
Prilog 2. Dokument završnog rada u .pdf formatu
Prilog 3. Dokument završnog rada u .tex formatu
Prilog 4. Programski kod izrađene aplikacije