IZRADA DIGITALNOG SUSTAVA ZA BLAGOSLOV KUĆA I OBITELJI

Vodnica, Tomo

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

IZRADA DIGITALNOG SUSTAVA ZA BLAGOSLOV KUĆA I OBITELJI

Tomo Vodnica

Zagreb, veljača 2023.

Predgovor

Zahvaljujem svom mentoru dr. sc. Aleksanderu Radovanu koji mi je pomagao pri rješavanju svih nejasnoća tijekom izrade završnog rada.

Zahvalan sam i svim ostalim profesorima i djelatnicima Visokog učilišta Algebra koji su mi prenijeli svoja znanja i iskustva na mom akademskom putu.

Zahvaljujem i svojoj obitelji koja mi je pružala potporu tijekom cijelog obrazovanja.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

U ovom radu kreirano je programsko rješenje koje digitalizira proces blagoslova kuća i obitelji. Blagoslov kuća i obitelji tradicionalan je običaj koji se u katoličkim sredinama prakticira od davnina, međutim u relativno nepromijenjenom obliku, zbog čega u današnje vrijeme rezultira određenim problemima. Novim sustavom riješeni su određeni problemi, ali je i dodatno olakšan cijeli proces svima, i svećenicima i vjernicima. Sustav tako između ostaloga vjernicima daje uvid o lokaciji svećenika i procijenjenom vremenu njegovog dolaska, a svećeniku pruža informacije o svim obiteljima i napretku blagoslova. Time korisnici imaju jednostavan pristup svim potrebnim informacijama i mogu maksimalno iskoristiti svoje vrijeme.

Ključne riječi: blagoslov kuća, blagoslov obitelji, Android, React, Firebase

Summary

In this paper a programming solution has been created which digitalizes the house and family blessing process. This process is a traditional custom which is practiced in catholic areas since ancient times, but it has seen little changes throughout it's history, so nowadays it produces certain problems. The new system primarily solves some of those problems, but additionaly makes the entire process easier overall, for both priests and believers. The system, among other things, displays to the believers the location of the priest and the approximate time of his arrival, and to the priest it displays all information regarding families and the blessing progress. This way users have a simple access to any information they might need and they are able to make maximum use of their time.

Key words: house blessing, family blessing, Android, React, Firebase

Sadržaj

1. Uvod

Svake godine tijekom blagdana u katoličkim kućanstvima održava se blagoslov doma i obitelji. Svećenik lokalne župe, sam ili u pratnji jednog ili više ministranata, sakristana ili nekih drugih osoba dolazi u domove župljana i blagoslivlja prisutne ukućane i njihov dom.

Ovaj običaj seže još do davnina, ali do danas se generalno nije mijenjao i u moderno doba izvodi se u više-manje izvornom, tradicionalnom obliku. Međutim osim tradicije vuku se i određeni problemi.

Kada bi se čitav proces digitalizirao možda bi se određeni problemi mogli riješiti, a možda bi i ostatak procesa bio jednostavniji za sve sudionike.

Na početku rada objašnjeno je koji su to uopće problemi koje bi trebalo rješavati. Potom je vidljivo kako bi se oni mogli riješiti i na temelju toga predložen je sustav s potrebnim funkcionalnostima. U sljedećem poglavlju definirana je arhitektura predloženog sustava kako bi se osigurala što jednostavnija izrada. Poglavlje nakon toga bavi se samom izradom sustava i svih potrebnih funkcionalnosti. Na kraju rada, kada je sustav spreman, odrađeni su posljednji testovi prije produkcije, a rad završava zaključkom čitavog procesa.

2. Razlike klasičnog i digitalnog sustava

Dobra stvar kod ideje modernizacije blagoslova je da temeljni koncept ostaje nepromijenjen. Za obaviti sami čin blagoslova svećenik fizički mora biti prisutan u domu obitelji koja se blagoslivlja. Takvi elementi koje nije moguće (ili potrebno) digitalizirati nisu dirani, nego je fokus bio samo na problemima. Stoga korisnici starije životne dobi i svi oni koji nisu toliko informatički pismeni neće imati nikakvih problema. Korištenje sustava nije obaveza, već samo mogućnost da proces bude jednostavniji. Ipak da bi vjernici mogli koristiti sustav, potrebno je da ga koristi svećenik. Međutim, ukoliko svećenik nije informatički pismen, s njime uvijek može ići netko tko je bolji s tehnologijom, i upravljati aplikacijom umjesto samog svećenika.

2.1. Nedostaci klasičnog blagoslova

S obzirom da najčešće nije poznato točno vrijeme dolaska svećenika, na dan blagoslova vjernici često iščekuju njegov dolazak nervozno i nestrpljivo. Da izbjegnu scenarij u kojem svećenik dođe, a nikoga nema doma, vjernici često cijeli dan stoje kod kuće što im onemogućava dobru organizaciju i iskorištavanje svog vremena. Za dobiti bar neke informacije, vjernici često zovu svoje susjede, prijatelje i sumještane ne bi li saznali gdje se svećenik nalazi i kada bi barem otprilike mogao doći. Ali čak i ako saznaju gdje se svećenik nalazi i kod koga je bio, ponovno im nije poznato točno vrijeme njegova dolaska. Negdje se može zadržati kraće, negdje dulje, a može otići i na neku pauzu, primjerice na ručak. Također, ukoliko svećenik završi s blagoslovima za taj dan, postoji mogućnost da ga vjernici ostanu čekati jer ne znaju da se neće pojaviti. Ukoliko vjernici nisu dostupni na dan blagoslova, a nisu to prijavili, svećenik može nepotrebno gubiti vrijeme na odlazak do njihovog doma. Ukoliko se blagoslov radi samo za prijavljene obitelji, svećenik također izdvaja puno svog vremena na izradu popisa obitelji za blagoslov. Ako se popis čuva na papiru, može biti nezgodno nositi sa sobom veliku količinu papira. Informacije na papiru mogu biti nepregledne, a neke informacije nije moguće ni imati, poput točne lokacije vjernika na mapi.

2.2. Prijedlog rješenja

Sve navedene probleme moguće je riješiti digitalizacijom čitavog procesa. Da bi se to postiglo, kreirane su web i mobilna aplikacija koje će koristiti i svećenik i vjernici. Obje aplikacije imaju iste funkcionalnosti, uz iznimku što tijekom samog blagoslova svećenik mora koristiti mobilnu aplikaciju.

Svaki svećenik ima svoj korisnički račun kojeg mu dodjeljuje administrator sustava. Samostalna registracija računa nije moguća kako bi se onemogućila izrada lažnih računa. Na svom korisničkom računu svećenik registrira župe koje predstavlja. Svaka župa ima popis obitelji za blagoslov. Obitelji na popis svećenik može unositi ručno, ali može i podijeliti poveznicu sa župljanima preko koje će se sami prijaviti. Time se osigurava točnost podataka, a svećenik ima manje posla. U svakom trenu svećenik može kontrolirati jesu li prijave otvorene ili ne, odnosno mogu li se vjernici sami prijavljivati. S obzirom da vjernici svoje podatke mogu i uređivati, na ovaj način osigurana je i ažurnost podataka.

Kada je popis spreman, svećenik može objaviti raspored. Obitelji na popisu može pridruživati u grupe, a za svaku grupu objaviti poseban raspored. Na ovaj način vjernici će znati kada blagoslov počinje i završava, ali i točno kada će biti blagoslov grupe kojoj pripadaju. Sve promjene statusa blagoslova bit će vidljive vjernicima u svakom trenu. Tako će se uvijek znati je li blagoslov u tijeku i je li svećenik na pauzi.

Dok god je blagoslov u tijeku, svi vjernici vide i lokaciju svećenika na mapi u stvarnom vremenu, s obzirom da svećenik na mobitelu ima obavezno uključen GPS koji čita njegovu lokaciju tijekom cijelog procesa blagoslova. Vjernici također vide i broj obitelji prije svoje koje isto tako čekaju na blagoslov. Na temelju podataka kao što su trenutna lokacija, udaljenost ili broj obitelji prije njihove, vjernici imaju uvid i u procijenjeno vrijeme dolaska svećenika. Ove funkcionalnosti omogućuju vjernicima da bolje organiziraju svoj dan i maksimalno iskoriste svoje vrijeme.

Tijekom blagoslova, svećenika aplikacija navodi kroz listu vjernika. Tako svećenik na jednom mjestu ima pregledan prikaz trenutno potrebnih informacija, uključujući i one koje na papiru ne bi bile dostupne, poput točne lokacije vjernika na mapi. Popis obitelji dostupan je cijelo vrijeme, uz brzo pretraživanje i jednostavno filtriranje po grupama, dostupnosti i informaciji je li blagoslov odrađen, ali i drugim parametrima koji bi svećeniku mogli biti korisni. Ovakav popis pregledniji je i jednostavniji za korištenje u odnosu na klasičan papir,

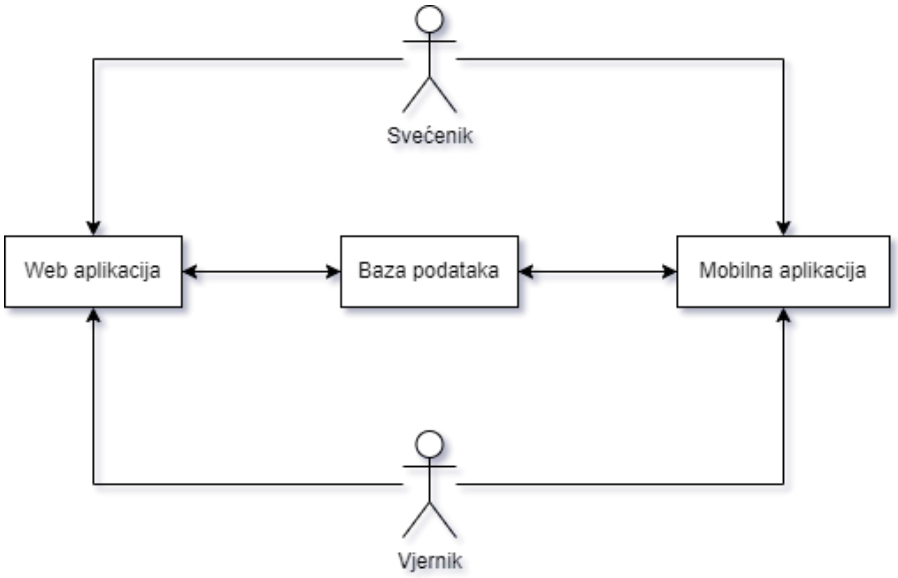
a donosi i dodatne mogućnosti poput uspostavljanja poziva s vjernikom u samo jednom kliku. Tijekom blagoslova svećenik može uređivati napomenu koja je vidljiva svim vjernicima.

3. Arhitektura sustava

Prije samog početka izrade sustava, prije nego započne programiranje, potrebno je dobro razmisliti što će se točno raditi, kako i na koji način. Potrebno je znati detaljno za svaku komponentu aplikacije što i kako radi, te kako se uklapa u sustav. Ovo je dio koji je možda manje zanimljiv, na trenutke čak i dosadan. Međutim, ovo je dio koji je od iznimne važnosti, a možda čak i najvažniji. Naime, ukoliko se ovaj dio odradi kvalitetno i odmah u startu se dobro definira arhitektura sustava, kasnije će biti neizmjerno lakše tijekom same izrade. Protivno, ukoliko se arhitektura ne definira dobro prije početka izrade, kasniji rad bit će otežan i vjerojatno će biti potrebno prepravljati sustav kasnije kada se shvati da je nešto pogrešno postavljeno. A u slučaju da se ovaj korak potpuno preskoči, kasniji problemi su neizbježni.

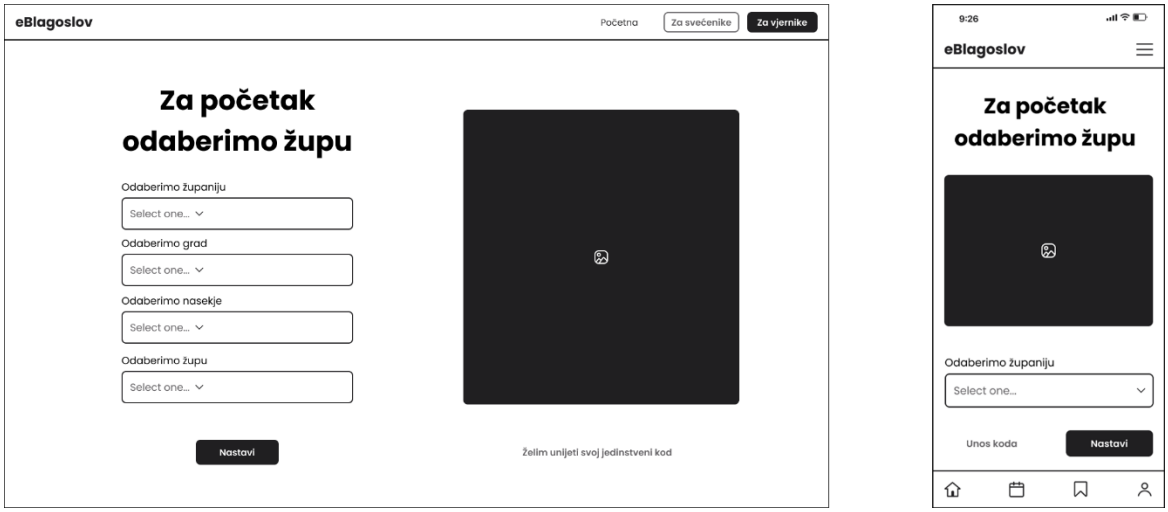
3.1. Skica i opis sustava

Kao što je ranije spomenuto, svim korisnicima, odnosno i svećeniku i vjernicima, na raspolaganju su dostupne mobilna aplikacija i web aplikacija. Obje aplikacije dijele iste funkcionalnosti, odnosno rade s istim podacima. Stoga se između njih nalazi zajednička baza podataka s kojom komuniciraju. Skica ovakvog sustava prikazana je slikom (Slika 3.1).



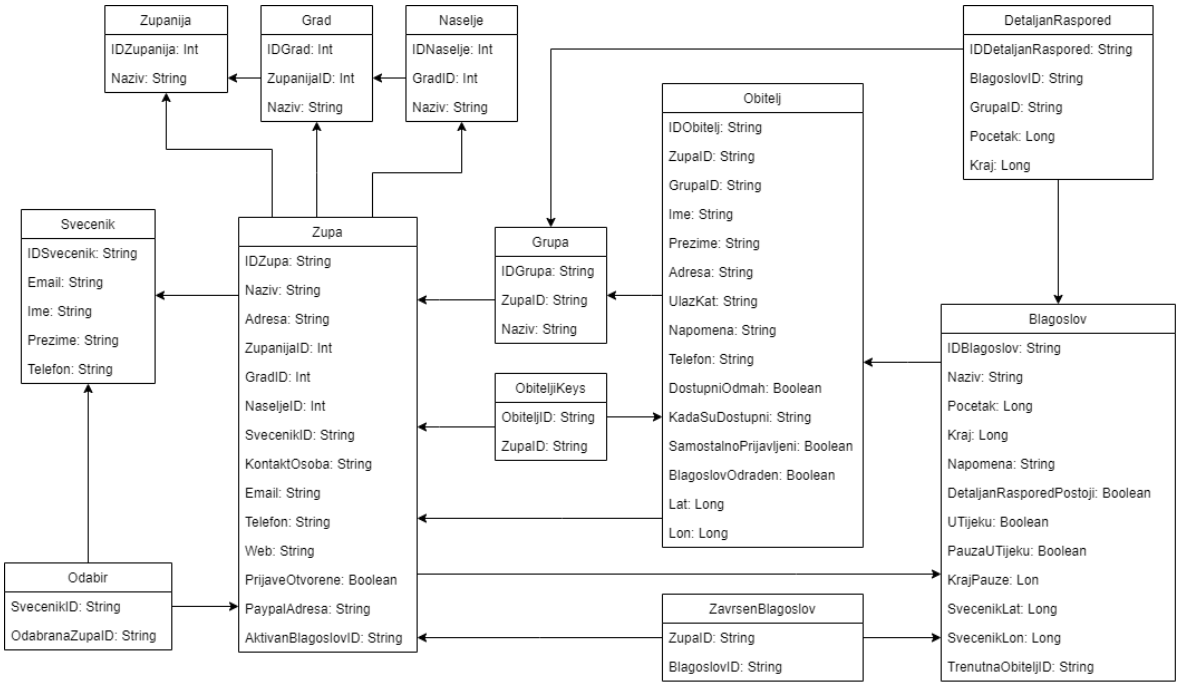
Slika 3.1 Skica arhitekture sustava

Kako bi se tijekom razvoja znalo točno što se treba raditi u svakom trenutku, kreiran je *wireframe*. *Wireframe* je pojednostavljeni vizualni prikaz aplikacije i njegovi elementi nisu interaktivni. Služi da bi se znalo kako će aplikacija izgledati, tako da se zna točno što raditi kada se krene razvijati. Kod izrade *wireframe*-a popisani su svi elementi i funkcionalnosti aplikacije i raspoređeni po zaslonima. Tako se zna točno koji zaslon što prikazuje i kako se veže na druge zaslone. Ovo omogućuje preglednu vizualnu prezentaciju svih elemenata sustava, kao što je vidljivo u primjeru na slici (Slika 3.2).



Slika 3.2 Dio *wireframe*-a aplikacije

Za jednostavniji rad s podacima na servisnom sloju, kreiran je dijagram baze podataka. Dijagram na jednostavan način vizualno prikazuje sve setove podataka koji se koriste u sustavu. Referencirajući se na ovaj dijagram tijekom rada s podacima, rad je olakšan i manje sklon pogreškama.



Slika 3.3 Dijagram baze podataka

3.2. Korištene tehnologije, jezici i razvojna okruženja

Za izradu *wireframe*-a korišten je alat koji se zove Figma. Figma je moćan alat za dizajniranje bogatih korisničkih sučelja. Omogućuje jednostavno kreiranje i oblikovanje svih potrebnih elemenata korisničkog sučelja. Na korištenje nudi brojne predloške zbog kojih nije potrebno crtati sve elemente ispočetka, pa time rad postaje brži i jednostavniji. Iako sadrži brojne značajke i vrlo je detaljna, Figma je jednostavna za korištenje, a povrh svega i besplatna [1].

3.2.1. Klijentski sloj

Za izradu vizualnih dijelova web aplikacije korišteni su jezici HTML i CSS.

HTML je jezik modeliranja i nema nikakvu programsku logiku već samo definira strukturu stranice. Sastoji se od niza elemenata kojim naznačuje pregledniku kako da prikaže sadržaj.

CSS je jezik za stiliziranje stranice. Definira točno kako HTML elementi trebaju izgledati. Osim uređivanja osnovnih parametara kao što su boja, veličina teksta, font i slično, CSS može raditi i složenije stvari kao na primjer mijenjati raspored elemenata ili kreirati

animacije. Za jednostavnije stiliziranje elemenata aplikacije korištena je biblioteka Bootstrap. Bootstrap je besplatna popularna biblioteka otvorenog koda koja olakšava stiliziranje stranice jer dolazi uz predefinirane stilove koje samo treba primijeniti na HTML elemente. Na taj način smanjuje se obujam posla jer se mora pisati manje vlastitih stilova.

U aplikaciji su korištene i neke druge biblioteke kao što su primjerice Aos ili Swiper. Aos je besplatna biblioteka za jednostavniji rad s animacijama. Dolazi s mnoštvom već ugrađenih animacija i HTML elementu na koji se primjenjuje potrebno je samo predati atribut s imenom animacije koja se koristi i rezultat će biti vidljiv. Swiper je također besplatna biblioteka koja je jednostavna za korištenje. Služi za prikaz elemenata u rotirajućem okruženju. Kod korištenja Swiper-a također treba samo predati atribut HTML elementima koji će se prikazivati u tom obliku, a Swiper će se pobrinuti za ostalo.

Programski jezik koji je korišten za izradu web aplikacije je Javascript. Javascript je jedan od najpopularnijih programskih jezika u svijetu, a kada se govori u kontekstu weba, vjerojatno i najpopularniji. Jedan je od temeljnih jezika razvoja web aplikacija uz HTML i CSS [2]. Iako se u radu često pisao i originalan Javascript kod, da bi se olakšala njegova uporaba korištena je biblioteka React.

React je trenutno jedna od najpopularnijih biblioteka za korištenje JavaScripta [3]. Razvio ga je Facebook 2013. godine. React koristi JSX sintaksu koja kombinira JavaScript kod i HTML. U srcu svake React aplikacije nalaze se komponente. Komponenta je jedan dio korisničkog sučelja i kada se razvijaju React aplikacije, ustvari se gradi više neovisnih, izoliranih i ponovno upotrebljivih komponenata od kojih se onda kreira složeno korisničko sučelje. Svaka React aplikacija ima barem jednu komponentu koja se naziva korijenska komponenta. Ona predstavlja internu aplikaciju i sadrži druge komponente, tako da je svaka React aplikacija u stvari stablo React komponenata. S obzirom da se komponente mogu koristiti više puta, elemente aplikacije koji se češće pojavljuju, kao što su zaglavlje i podnožje, moguće je kreirati i izmjenjivati na jednom mjestu, a prikazivati i koristiti gdje god se želi. Svaka komponenta ima svoje stanje i ne ovisi o drugim komponentama. Stoga prilikom promjene podataka React ažurira isključivo one elemente gdje su podaci promijenjeni. Na ovaj način ne mijenja se struktura cijelog dokumenta što odaje dojam brže i responzivnije stranice.

Razvojno okruženje koje je korišteno za ovaj segment je Visual Studio Code [4]. Visual Studio Code je vrlo lagan uređivač teksta i ne zahtjeva ozbiljne hardverske resurse.

Jednostavan je alat za korištenje, a njegove funkcionalnosti mogu se proširivati instaliranjem brojnih dodataka i proširenja.

Android aplikacija pisana je u programskom jeziku Kotlin [5]. Kotlin je moderan strogo tipiziran programski jezik temeljen na programskom jeziku Java. U odnosu na Javu, Kotlin donosi brojne korisne značajke kao što su pisanje manje količine koda i predefinirano onemogućavanje dodjele `null` vrijednosti varijablama čime se drastično smanjuju greške i iznimke u programu. Osim toga Kotlin je i službeno preporučeni jezik za izradu Android aplikacija [6].

Razvojna okolina koja je korištena je program Android Studio [7]. Također službeno preporučen, Android Studio je vrlo moćan alat za razvoj temeljen na programu IntelliJ IDEA. Brojne ugrađene značajke i pametan uređivač teksta znatno olakšavaju i ubrzavaju proces programiranja. U slučaju kada se rade promjene koje ne utječu na programsku logiku, primjerice vizualne promjene korisničkog sučelja, moguće je osvježiti aplikaciju bez potrebe za ponovnim sastavljanjem aplikacijskog koda.

Uz integrirano razvojno okruženje dolazi ugrađen i moćan emulator koji emulira rad stvarnih Android uređaja. Ovo omogućuje jednostavno testiranje na raznim Android uređajima i na raznim verzijama Android operacijskog sustava. Osim pametnih telefona moguće je emulirati i druge uređaje koji koriste Android kao što su pametni satovi, televizije, tableti ili sustavi u automobilima. Emulator podržava testiranja uz korištenje svih mogućih hardverskih komponenti uređaja kao što su kamera, GPS, žiroskop, senzor za otisak prsta i slično.

3.2.2. Servisni sloj

Za servisni sloj korišten je Google Firebase [8]. Firebase je BaaS sustav, odnosno sustav koji omogućuje korištenje usluga servisnog sloja bez potrebe pisanja i održavanja programskog koda. Firebase je besplatan za korištenje dok god radi s količinom podataka unutar određene granice. Nakon toga se korištenje usluge naplaćuje, međutim s obzirom da nije predviđena velika količina podataka potrebnu sustavu, imat će i više nego dovoljno slobodnog prostora na raspolaganju. Firebase na raspolaganje stavlja brojne svoje usluge, a neke koje sustav koristi su baza podataka, autentifikacija korisnika, analitika i hosting.

Postoje različiti tipovi baza podataka. Jedan od onih koji se češće koristi su relacijske baze podataka kod kojih se podaci spremaju u tablice koje imaju veze s drugim tablicama. Za rad s podacima koristi se SQL što je standardni jezik za pristupanje i manipuliranje podacima u ovakvim bazama podataka. Firebase za pohranu podataka koristi NoSQL nerelacijsku bazu podataka u kojoj se podaci ne spremaju u različite tablice nego u kontejnere kao što su dokumenti ili JSON datoteka. Ovakva baza omogućuje jednostavnije korištenje jer se ne mora pisati nikakav SQL kod. Umjesto toga svi zapisi u bazi odvijaju se direktno iz koda klijentskih aplikacija.

Firebase se pobrinuo i za vrlo jednostavnu, a ipak sigurnu autentifikaciju korisnika i pohranu njihovih podataka. U svega par klikova moguće je omogućiti registraciju i prijavu korisnika s klasičnom metodom kombinacije e-mail adrese i zaporke, ali također i koristeći račune drugih platforma kao što su Facebook, Google, Microsoft i drugi. Nakon prijave moguće je zatražiti i dodatnu potvrdu od korisnika u obliku verifikacijskog koda poslanog SMS porukom. Firebase automatski šalje korisnicima SMS i e-mail poruke i za ostale stvari poput promjene zaporke, promjene e-mail adrese, potvrde e-mail adrese i slično.

Hosting web aplikacije također može odraditi Firebase [9]. Za početni plan bez naknade na raspolaganju je 10 GB slobodnog prostora i do 360 MB dnevnog prometa što je za potrebe web aplikacije ovog sustava sasvim dovoljno. Automatski su dostupne i dodatne značajke poput prilagođene domene i SSL certifikata. Konfiguriranje je vrlo brzo i jednostavno, a puštanje aplikacije u pogon i izdavanje ažuriranja izvršava se u svega nekoliko naredbi u terminalu.

Za praćenje svih mogućih parametara u vezi sustava koristit će se ugrađeni alat za analitiku. Ovaj alat prikazuje brojne podatke kao što su rušenja i greške, akvizicija i broj korisnika, njihova aktivnost, uređaji koje koriste, efikasnosti poslanih obavijesti, kupnje unutar sustava i tako dalje. S obzirom na veliki broj detaljnih informacija, puno njih neće biti zanimljivo, ali isto tako puno njih pomoći će da rješenje bude bolje. Primjerice u slučaju grešaka, vidjet će se točno što nije u redu i popraviti se. Informacije poput broja aktivnih korisnika i njihove lokacije dat će uvid u uspješnost projekta, a pregled uređaja koji se češće koriste otvorit će mogućnost da se stave u prvi plan tijekom razvoja.

Firebase na raspolaganju ima još veliki niz usluga koje neće biti potrebne u sklopu ovog projekta. Međutim, ukoliko se naknadno odluči u projekt implementirati dodatne funkcionalnosti, uvijek se to može napraviti, i to vrlo jednostavno. Potrebno je svega par

klikova u kontrolnoj ploči Firebase-a, a ovisno o funkcionalnosti, možda i instaliranje odgovarajućih biblioteka u klijentskim aplikacijama. Međutim, u samom Firebaseu nikada se neće morati napisati niti jedna jedina linija koda.

4. Razvoj i implementacija sustava

Kada su definirani svi preduvjeti, stiglo je vrijeme da se prijeđe na ono najzanimljivije, a to je sama izrada sustava. Prvo su konfigurirane baza podataka i svi ostali elementi na servisnom sloju sustava, a onda su odmah tijekom izrade klijentskih aplikacija implementirane sve funkcionalnosti servisnog sloja tako da se već u startu radilo sa stvarnim podacima.

4.1. Izrada i konfiguracija servisnog sloja

Za početak rada kreiran je novi korisnički račun u Firebase-u, a potom i novi projekt kojem je dodijeljen naziv. Projekt koji je kreiran predefinirano ne koristi niti jednu od usluga koje nudi Firebase, stoga su ručno dodavane usluge koje su potrebne. Prva usluga s kojom se radilo je baza podataka. Firebase stavlja na raspolaganje dvije vrste baza: Realtime database i Cloud Firestore [10].

Realtime database je originalna inačica baze u sustavu Firebase koja sprema podatke u obliku jednog JSON stabla. Podatke je vrlo jednostavno spremati, ali kompliciranije podatke u velikim količinama teže je organizirati. Da bi performanse bile što bolje potrebno je strukturu podataka čuvati u što ravnijem obliku, odnosno sa što manje čvorova. Čitavu bazu moguće je preuzeti lokalno i koristiti čak i kada nema internetske veze, ali samo u Android i Apple aplikacijama, odnosno ne na webu.

Cloud Firestore sprema podatke u kolekcije dokumenata koji su slični JSON-u, ali omogućuju jednostavnije organiziranje kompleksnijih hijerarhijskih podataka u većim količinama. Ni velika količina ni struktura podataka ne utječu na performanse i nije potrebno održavati ravnu strukturu podataka. Ova vrsta baze podataka omogućuje lokalno korištenje bez internetske veze za sve aplikacije, uključujući i one na webu.

S obzirom da ovaj projekt nema prevelik ni prekompliciran skup podataka, a nije ni potrebna mogućnost korištenja baze bez internetske veze, odabrana je Realtime database baza. Međutim i da se odabrala Cloud Firestore baza, ne bi se previše pogriješilo.

Za kreiranje baze potrebno je samo pritisnuti tipku „Create database“. I to je to, baza je kreirana i spremna za korištenje. Odmah su i upisani prvi podaci kroz korisničko sučelje. Na ovaj način upisani su svi statični podaci jer se oni ne mogu mijenjati kroz klijentske aplikacije. Ovi podaci su županije, gradovi i naselja republike Hrvatske. Za dodavanje podataka dovoljno je pritisnuti tipku „+“ i upisati ključ i vrijednost podatka, što je vrlo jednostavno. Međutim ovih podataka ima jako puno tako da nije baš praktično ni vremenski efikasno unositi ih ručno. Da bi se stvar pojednostavnila, podaci su uvedeni koristeći funkciju „Import JSON“. Odabrana je lista županija, gradova i naselja koja je preuzeta s interneta32[17] u JSON formatu i svi podaci su u bazi u svega par klikova. Ostali podaci i njihovi tipovi definirani su kasnije kada se baza koristila u klijentskim aplikacijama. U ovom trenutku ograničenja ne postoje, već se u bazu može pisati bilo što.

Ipak, da bi se ograničile neželjene radnje, na razini baze definirana su određena pravila. Klikom na „Rules“ otvara se skup svih pravila također prikazan u JSON formatu. Ovdje su definirana prava čitanja i pisanja za sve skupine podataka u bazi. Primjerice, statičke podatke poput liste županija, gradova ili naselja nitko ne bi trebao imati pravo uređivati, ali s obzirom da su to javno dostupni podaci, bilo tko bi trebao imati pravo na čitanje. Župe, blagoslove i slično bi trebali moći uređivati samo prijavljeni korisnici, odnosno svećenici, a osobnim podacima obitelji određenog vjernika bi trebali imati pristup samo ta obitelj i svećenik one župe kojoj ta obitelj pripada. Za lakše pisanje sigurnosnih pravila, na istom zaslonu dostupan je alat za simuliranje upita prema bazi. Koristeći ovaj alat vrlo jednostavno se mogu slati sve vrste upita bazi i vidjeti njen odgovor kako bi se testirao rad napisanih pravila. Kada su pravila definirana, u odjeljku „Monitor rules“ može se pratiti statistika upita prema bazi i vidjeti točno koji upiti su prihvaćeni, koji odbijeni, a koji su bili neispravni i rezultirali pogreškom.

```
{
  "rules": {
    "zupanije":{
      ".read": true,
      ".write": "false"
    },
    "svecenici": {
      "$uid": {
        ".read": true,
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

Kôd 4.1 Primjer pravila na razini baze

Za sljedeći korak omogućena je autentifikacija korisnika i kreiran je prvi korisnik. Autentificirani korisnici u sustavu su svećenici, a vjernici za pristup svojim podacima koriste samo svoj jedinstveni identifikator kojeg će Firebase automatski generirati. Korisnici u Firebase-u ne čuvaju se u samoj bazi kao ostali podaci, već za to Firebase ima posebnu uslugu. Za postavljanje ove usluge potrebno je otići u sekciju „Authentication“ i pritisnuti „Get started“, a zatim odabrati način na koji će se korisnici moći prijavljivati. Sam Firebase na raspolaganju ima tri mogućnosti autentifikacije:

1. anonimno bez vjerodajnice
2. kombinacijom e-mail adrese i lozinke
3. brojem mobilnog telefona

Osim svojih metoda autentifikacije, Firebase nudi i autentifikaciju korisnika kroz sustave trećih strana kao što su Google, Facebook, Twitter, Apple, Microsoft, Yahoo i drugi. S obzirom da je najveća vjerojatnost da svaki korisnik ima e-mail adresu, ali i zato što je to najklasičniji pristup prijave u neki sustav, sustav koristi prijavu kombinacijom e-mail adrese i lozinke. Načina prijave može biti više, stoga ukoliko se u budućnosti odluči omogućiti i prijavu nekim drugim načinom, uvijek je moguće vratiti se i to učiniti.

Kada je odabran bar jedan način prijave, može se otići u sekciju „Users“ gdje su pohranjeni svi korisnici. Za potrebe testiranja sada je kreiran jedan korisnik. Jednostavnim upisom

željene e-mail adrese i lozinke te potvrdom na „Add user“ registriran je prvi korisnik. Registraciju je moguće implementirati u klijentskim aplikacijama tako da se korisnici sami registriraju, međutim da bi se onemogućilo kreiranje lažnih računa u sustavu, to nije omogućeno. Umjesto toga svaki račun kreirat će administrator kroz kontrolnu ploču Firebase-a.

Kao što je ranije napomenuto, Firebase korisnicima šalje automatske email ili SMS poruke za verifikaciju ili promjenu e-mail adrese, promjenu lozinke, dvostruku autentifikaciju ili prijavu brojem telefona. U odjeljku „Templates“ mogu se uređivati predlošci tih automatskih poruka. Za njihovo slanje potrebno je samo pozvati odgovarajuću metodu u klijentskoj aplikaciji, a Firebase će odraditi ostalo.

I na autentifikaciji korisnika još jednom je demonstrirana jednostavnost korištenja Firebase-a u kojem je praktički sve dostupno unutar nekoliko klikova.

4.2. Izrada web aplikacije

Prije nego se započne s izradom web aplikacije, potrebno je prvo pripremiti razvojno okruženje. Instalirani su Visual Studio Code i proširenja kojim je konfiguriran za lakši rad. Zatim je instaliran Node.js kako bi se mogao koristiti npm [12]. Node.js je platforma za izvođenje JavaScript koda izvan internetskog preglednika, a omogućuje i korištenje npm-a. Npm je najveći svjetski registar programske potpore i omogućuje brzo i jednostavno dodavanje raznih biblioteka u projekt koristeći samo jednu naredbu u konzoli. Besplatan je za korištenje. Određenim naredbama instalirat će React i kreirati aplikaciju. Na ovaj način instalirani su i drugi dodaci koji će se koristiti. Jedan od njih je primjerice Firebase za rad sa servisnim slojem, koji je nakon instalacije konfiguriran da radi s ovim projektom. Iz kontrolne ploče Firebase-a pokupljeni su konfiguracijski parametri koji su potom definirani u novokreiranom projektu te je tako aplikacija povezana s točno onim određenim Firebase projektom.

Kreirani React projekt sadrži direktorije “Public“, “Src“, i „Node_modules“.

U „Node_modules direktoriju nalaze se sve biblioteke koje su dodane u projekt. Ovi podaci se automatski preuzimaju i ovaj direktorij se najvjerojatnije nikada neće ručno modificirati.

Mapa „Public“ sadrži javne podatke koje čine temelj aplikacije. Između ostaloga to uključuje i „index.html“ datoteku koja je glavna stranica aplikacije.

„Src“ direktorij je najzanimljiviji direktorij. Sadrži React Javascript kod za projekt. Većina posla koji je rađen odvijao se ovdje. Tu se nalaze sve stranice i sve komponente aplikacije.

Rad je započeo s naslovnom stranicom. Naslovna stranica nema nikakve funkcionalnosti već služi za ulaz u aplikaciju i za prikaz informacija koje će korisnike upoznati s radom aplikacije. S obzirom da je naslovna stranica prvi doticaj korisnika s aplikacijom, važno je da bude dobro strukturirana i vizualno atraktivna kako bi se na korisnike ostavio što bolji prvi dojam. Posebice je bitan vrh stranice, odnosno onaj dio koji se nađe na zaslonu korisnika kada se stranica prvi put učita. Potrebno je da ne bude previše informacija da korisniku ne bude zamorno, već da se što kraće i jednostavnije opiše o čemu se radi u aplikaciji. Također je potrebno da korisnik može što jednostavnije krenuti koristiti ono što ga najvjerojatnije zanima, odnosno da postoji što jednostavnija mogućnost započeti s korištenjem aplikacije. U ovom slučaju kreiran je jednostavan naslov i tipka koja vjernike uvodi u dio aplikacije namijenjen njima s obzirom da će biti brojniji dio korisnika u odnosu na svećenike. Na ostatku stranice prikazane su ostale informacije o aplikaciji, a na samom vrhu nalazi se zaglavlje za lakšu navigaciju.

Da bi se manje vremena potrošilo na strukturu dokumenta, umjesto da se krene od nule, korišten je predložak već gotove stranice [13]. Na ovaj način izbjegnuto je pisanje čestih elemenata kao što su zaglavlje ili navigacijski izbornik, a za neke druge elemente preuzeti su već gotovi stilovi i modificirani po potrebi, čime se dodatno olakšao posao i uštedjelo vrijeme.

Call to action tipka na početnoj stranici uvodi korisnike dakle u dio aplikacije namijenjen vjernicima. Na ovom zaslonu vjernici će ili odabrati svoju župu ili unijeti svoj jedinstveni kod. Ukoliko odluče ručno odabrati svoju župu, u listi ponuđenih opcija prvo će odabrati županiju svoje župe, potom grad, potom naselje, a na kraju i samu župu. Kada se jedna opcija odabere, druge se dinamično nude ovisno o prethodnom odabiru, a podaci se, naravno, učitavaju iz baze podataka. Za dohvat podataka napravljen je prvi upit prema bazi. Kreirana je nova instanca baze i proslijeđena kao parametar u metodu kojoj se također definira i putanja podataka koje se želi dohvatiti. Metoda vraća set podataka koji imaju ključ i vrijednost, baš kao i na bazi. Preporučeni način dohvata podataka iz baze je korištenjem slušača (engl. *eventListener*) koji osluškuju na promjene u dohvaćenim podacima. Ovaj način se najčešće koristi i ovdje u aplikaciji, tako da će u slučaju promjene podataka u bazi, novi podaci biti automatski dohvaćeni u aplikaciju.

```
const db = getDatabase();
const dbRef = ref(db, 'zupe');
onValue(dbRef, (snapshot) => {
    var zupeList = [];
    snapshot.forEach(zupa => {
        var key = zupa.key;
        var value = zupa.val();
        zupeList.push({key, value});
    });
    this.setState({zupe: zupeList});
});
```

Kôd 4.2 Primjer dohvata podataka iz baze

Umjesto ručnog odabira župe vjernici imaju i mogućnost unijeti svoj jedinstveni kod. Ukoliko se unese ispravan kod, župa vjernika će također biti odabrana.

Nakon odabira župe, aplikacija odlazi na glavni prikaz blagoslova za vjernike. Ovdje su prikazane sve informacije o odabranoj župi i blagoslovu za tu župu. Za odabranu župu prikazane su osnovne informacije kao što su naziv, adresa, e-mail adresa, kontakt telefon, web stranica i osoba za kontakt. Što se tiče samog blagoslova, vjernici vide je li blagoslov trenutno u tijeku ili ne. Ukoliko nije vide kada je planirani početak, a ukoliko jest kada je planiran završetak. Detaljan raspored po grupama također je dostupan, tako da vjernici vide točno kada će koja grupa imati blagoslov. Tijekom samog blagoslova prikazuju se dinamični podaci o blagoslovu. Na mapi je vidljiva lokacija svećenika u stvarnom vremenu, prikazan je napredak blagoslova, odnosno broj blagoslovljenih obitelji zasada, prikazane su planirane pauze i stanke ukoliko ih ima, a vidi se i napomena koju svećenik može objaviti svim vjernicima. U bilo kojem trenutku svećenik može otići na pauzu. Ukoliko je svećenik na pauzi vjernici će to vidjeti, a ukoliko je svećenik definirao predviđeno trajanje pauze, vjernici će i to ovdje vidjeti.

Ukoliko je unesen jedinstveni kod vjernika, prikazane su i personalizirane informacije za tog vjernika. Na mapi se sada osim lokacije svećenika vidi i lokacija vjernika, dostupan je podatak o broju drugih obitelji koje čekaju na blagoslov prije, a prikazano je i predviđeno

vrijeme dolaska svećenika. Osobni podaci vjernika također su prikazani, a mogu se i uređivati.

Ukoliko je svećenik tako definirao, za odabranu župu moguće je izvršiti i donaciju. Potrebno je samo odabrati željeni iznos i izvršiti plaćanje. Plaćanje se vrši servisom PayPal. PayPal je internetski platni servis koji omogućuje sigurno slanje i primanje novca putem interneta. Za implementiranje ove funkcionalnosti u aplikaciju potrebno je samo dodati određeni kod preuzet od PayPal-a i konfigurirati određene parametre kao što su iznos, valuta i račun primatelja, a PayPal će se pobrinuti za ostalo. Nije potrebno brinuti o detaljima poput sigurnosnih rizika jer se tijekom plaćanja korisnik preusmjerava na službene stranice PayPal servisa.

Odabirom funkcije za svećenike u zaglavlju odlazi se u dio aplikacije namijenjen svećenicima. Da bi svećenici mogli upravljati svojim sadržajem, prvo se moraju prijaviti u sustav. Za prijavu svećenici unose e-mail adresu i zaporku koju aplikacija potom proslijedi u metodu za prijavu koju je omogućio Firebase [11]. Isto kao što se vidjelo i kod PayPal-a ranije, Firebase izvan aplikacije provjeri postoji li korisnik s unesenim podacima i vrati rezultat sukladno. Ukoliko je prijava uspješna sada je moguće dohvatiti podatke o prijavljenom korisniku i preusmjeriti ga na sljedeći zaslon, a ukoliko nije ispisati poruku o pogreški.

```
const auth = getAuth();
signInWithEmailAndPassword(auth, email, password)
          .then((userCredential) => {  // Prijava uspješna
            const user = userCredential.user;
          })
          .catch((error) => { // Prijava nije uspješna
            console.log(error.code + error.message);
          });
```

Kôd 4.3 Primjer prijave korisnika u sustav

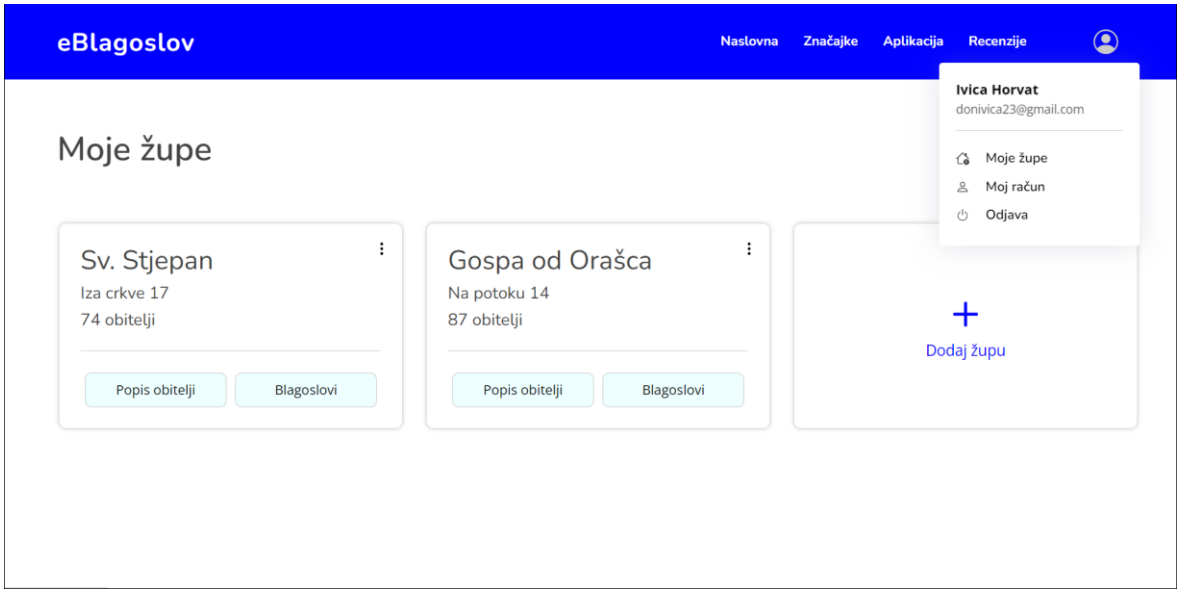
U slučaju zaboravljene lozinke moguće je zatražiti novu. Potrebno je samo unijeti odgovarajuću e-mail adresu i u poštanski sandučić stiže poruka s poveznicom za promjenu lozinke koja je pripremljena ranije u Firebase kontrolnoj ploči.

Nakon što se korisnik uspješno prijavi, Firebase čuva tu informaciju i ona je dostupna bilo gdje u aplikaciji. Potrebno je samo pozvati `firebase.auth().currentUser` za dohvatiti objekt prijavljenog korisnika. Ovaj objekt sadrži sve informacije o korisniku, čak i ako se prijavio kroz aplikaciju neke treće strane kao što je npr. Facebook. Ukoliko ovaj objekt ima vrijednost `null` znači da korisnik nije prijavljen.

Za odjavu iz sustava samo treba pozvati `firebase.auth().signOut()`.

Nakon uspješne prijave napokon se ulazi u dio aplikacije za svećenike. S obzirom da se u komponenti zaglavlja promijenilo stanje prijave, sada se iscrtavaju elementi za svećenike. Tu se nalazi izbornik za upravljanje župama, odlazak na detalje korisničkog računa ili odjavu.

Početna stranica uključuje upravljanje župama kao što je prikazano slikom (Slika 4.1). Na ovoj stranici svećenik može dodavati nove župe, a postojeće može uređivati i brisati. Svaka župa ima popis obitelji i blagoslova kojima svećenik također može upravljati.



Slika 4.1 Korisničko sučelje za svećenike

Na stranici popisa obitelji prikazane su sve obitelji koje su prijavljene na blagoslov u odabranoj župi. Za lakši pregled ovaj popis svećenik može sortirati po imenu, prezimenu, adresi, grupi ili dostupnosti. Za situacije kada se traži određena obitelj, moguće je koristiti i tražilicu za brzu pretragu obitelji po imenu, prezimenu, adresi, telefonu ili jedinstvenom

kodu. Za prikaz samo određenih stavki, dostupna je i opcija za filtriranje. Filtrirati se može prema načinu prijave, dostupnosti i grupi. Popis se može i izvesti u .csv formatu tako da se kasnije može urediti ili ispisati kroz neki drugi program kao npr. Microsoft Excel. U postavkama popisa svećenik može upravljati grupama te definirati jesu li prijave otvorene ili ne. Također, može dohvatiti poveznicu koju može podijeliti s vjernicima kako bi se sami prijavili na blagoslov, ukoliko su prijave otvorene. Prilikom prijave obitelji na popis, unose se osobni podaci kao što su ime i prezime, adresa i kontakt telefon. Da bi svećeniku dolazak bio što lakši moguće je upisati i dodatne informacije kao što su kat, način ulaska u zgradu i slično. Osim unosa standardnih podataka, moguće je odabrati i točnu lokaciju objekta na mapi. Ova lokacija bit će prikazana svećeniku tijekom blagoslova što može biti vrlo korisno. Nadalje, obitelj se prilikom prijave može označiti i kao nedostupna u određenom razdoblju, što također može biti bitno svećeniku. Ukoliko se unosi više obitelji odjednom, dostupan je poseban, pojednostavljen prikaz koji omogućuje brži unos podataka.

Kada stigne vrijeme za blagoslov, u sekciji za blagoslove svećenik kreira novi blagoslov. Prilikom kreiranja blagoslova svećenik definira generalni početak i kraj, a može definirati i detaljan raspored za svaku kreiranu grupu vjernika. Blagoslov se može uređivati dok god nije završio. Nakon završetka automatski se sprema u listu završenih blagoslova koja ostaje vidljiva za evidenciju.

Za započeti sam proces blagoslova nije moguće koristiti web aplikaciju. Za ovu radnju svećenik će morati koristiti aplikaciju za mobitel. Ovo je jedina razlika između dviju aplikacija.

4.3. Izrada mobilne aplikacije

Za razvoj aplikacije za Android sustav korišten je program Android Studio i odmah je kreiran novi projekt. Prilikom kreiranja projekta potrebno je odabrati minimalnu verziju sustava na kojem će aplikacija biti podržana. Za pomoć pri odabiru Android Studio za svaku verziju prikazuje postotak aktivnih uređaja na kojima će se aplikacija moći izvoditi. Što je podržana verzija starija, to je veći broj podržanih uređaja na kojima će aplikacija moći izvoditi. Međutim što je verzija novija, veći je broj funkcionalnosti koje su aplikaciji dostupne za korištenje. Potrebno je razmisliti koje funkcionalnosti aplikacija treba, a koje će žrtvovati i pronaći neki kompromis. S obzirom da je aplikacija prilično jednostavna, odnosno ne koristi nikakve posebne funkcionalnosti, odabrana je najniža moguća verzija kako bi se podržao

maksimalan broj uređaja. Od verzije broj 6.0, naziva Marshmallow, donesene su drastične promjene u vidu sigurnosti stoga je odabrana ova verzija. Postotak podržanih uređaja iznosi 97,2% što je jako dobro.

Nakon kreiranja projekta, odmah je dodana i podrška za Firebase. S obzirom da Android, odnosno Android Studio i Firebase dijele istog vlasnika, implementacija Firebase-a u projekt u Android Studiu moguća je u svega par klikova. Potrebno je samo odabrati koje usluge Firebase-a se žele dodati, a Android Studio će se pobrinuti za ostalo. Prilikom dodavanja prve usluge potrebno se autentificirati, odnosno prijaviti se u Firebase i odabrati željeni projekt kroz internetski preglednik koji će Android Studio otvoriti. U pozadini će se instalirati potrebne biblioteke, tj. dodati potreban kod, a podaci o projektu će se automatski konfigurirati. Automatski će se dogoditi ono što se u web aplikaciji radilo ručno.

Što se tiče strukture projekta, moguće je izdvojiti nekoliko sekcija.

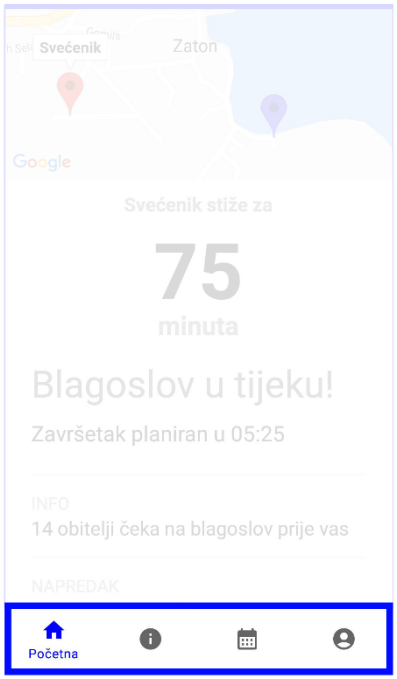
U mapi „java“ nalaze se sve datoteke koje sadrže programski kod, odnosno programsku logiku. Ovdje su organizirani svi paketi i datoteke u kojima se programira rad aplikacije. Mapa „res“ sadrži sve resurse i vizualne elemente aplikacije. Ovdje se nalaze slike, definirane boje, plan navigacije, datoteke u kojima se crta izgled zaslona i slično.

U korijenu projekta svake Android aplikacije obavezno se nalazi i „AndroidManifest.xml“ datoteka. Ova datoteka sadrži esencijalne informacije o aplikaciji kao što su sve njene komponente, dopuštenja ili hardverske značajke koje aplikacija koristi. Ovdje je između ostalog dodan i ključ za korištenje usluga Google mape.

Svaka aplikacija ima barem jedan *activity*. *Activity* predstavlja temeljni prikaz svake Android aplikacije. Da bi se *activity* mogao koristiti obavezno mora biti registriran u „AndroidManifest.xml“ datoteci. Srećom Android Studio to automatski radi u onom trenu kada se *activity* kreira. Da bi bilo lakše kreirati korisničko sučelje i da bi se ono bolje prilagođavalo različitim tipovima uređaja kao što su mobilni telefon ili tablet, unutar *activity*-a moguće je koristiti fragmente. Fragment je jedan dio korisničkog sučelja koji se može koristiti više puta i omogućava veću fleksibilnost nego *activity*. Fragment ima i svoj vlastiti životni ciklus, međutim ne može postojati samostalno nego se mora nalaziti unutar *activity*-a. Dobra je praksa koristiti *activity* što manje, a fragmente što više, stoga se i ovdje tako radilo. Svaka veća cjelina aplikacije je jedan *activity*, a svi ostali zasloni unutar te cjeline su fragmenti koji se izmjenjuju unutar svog roditelja.

Prilikom prvog ulaska u aplikaciju, otvara se prvi i glavni *activity*. Mobilna aplikacija nema informativni dio već korisnike odmah dočeka odabir župe. Kao i kod web aplikacije vjernici odabiru svoju župu ili unose kod, a ukoliko aplikaciju koristi svećenik tu je i zaslon za prijavu. Svaki od ovih zaslona je zaseban fragment koji je dio glavnog *activity*-a. Aplikacija ima točno jedan glavni *activity* i to je onaj koji se otvara prilikom starta aplikacije. Android zna koji je to *activity* jer je i to definirano u „AndroidManifest“ datoteci.

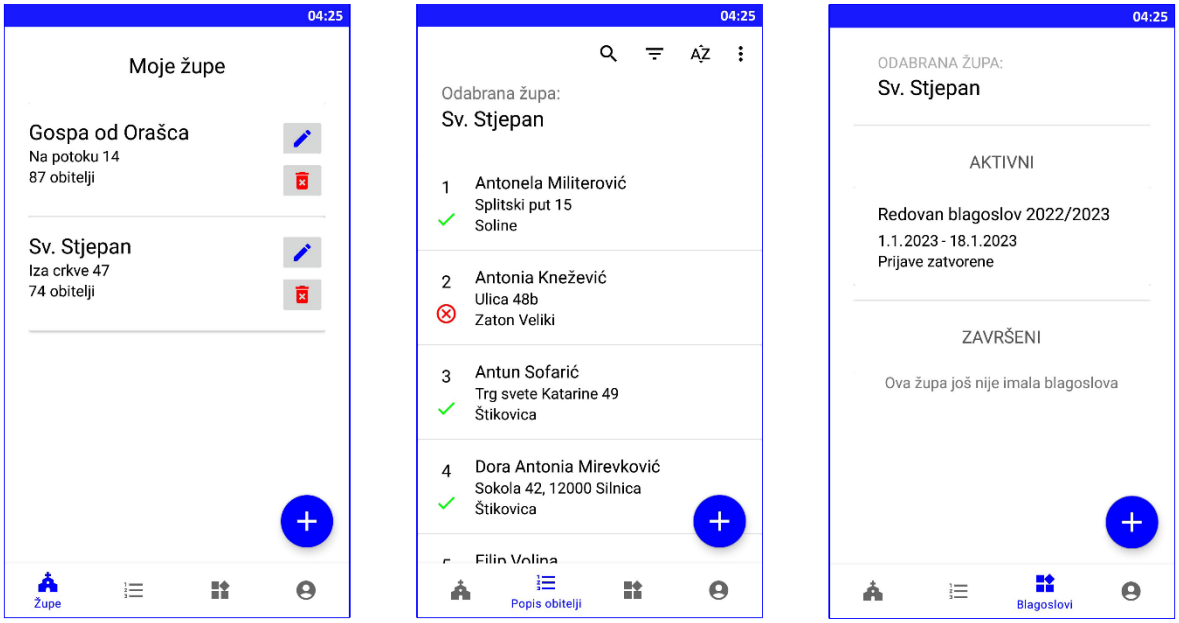
Nakon odabira župe, vjernici odlaze na zaslon blagoslova gdje prate informacije o samom blagoslovu. Ovaj prikaz je drugi *activity*. Koristi komponentu *bottom navigation* koja na dnu ekrana prikazuje izbornik koji vodi na različite zaslone, što je prikazano slikom (Slika 4.2). Prvi i glavni zaslon sadrži prikaz osnovnih informacija o blagoslovu, drugi zaslon sadrži informacije o odabranoj župi, treći zaslon prikazuje raspored blagoslova, a na četvrtom se nalaze osobne informacije vjernika i postavke. S obzirom da je izbornik dio *activity*-a on je vidljiv u svakom trenu, a svaki od navedenih zaslona jedan je fragment koji se prikazuje na zahtjev unutar *activity-a*.



Slika 4.2 *Bottom navigation* izbornik

Svećenici, kao i u web aplikaciji, nakon prijave ulaze u svoj dio aplikacije. I za ovaj dio ponovno se koristi jedan *activity* sa zajedničkim elementom izbornika, a fragmenti se

izmjenjuju na zahtjev. Prvi fragment sadrži sve u vezi župa, drugi popis obitelji, treći blagoslove, a na četvrtom su postavke korisničkog računa. Na ovaj način sve je pregledno i odvojeno jedno od drugog, a opet jednostavno dostupno (Slika 4.3).



Slika 4.3 Korisničko sučelje za svećenike

Na prva tri zaslona nalaze se liste koje prikazuju stavke s dinamičnim informacijama. Kod prikaza ovakvih lista u Androidu potrebno je podatke učitati u adapter i predati ga komponenti koja se zove *RecyclerView*. Adapter je klasa koja definira kako se podaci prikazuju i koliko ih ima. Na temelju tih podataka *RecyclerView* iscrtava listu na način da reciklira individualne elemente. Kada se lista pomakne i jedan element nestane s ekrana, njegovo mjesto se ne uništava, već na njega dolazi sljedeći element u listi koji je sada vidljiv na ekranu i tako se mjesto reciklira. Na ovaj način poboljšavaju se performanse i smanjuje potrošnja energije kod prikaza većih setova podataka [16].

Na dan blagoslova, u zakazano vrijeme početka, svećenik započinje proces blagoslova. Prije samog početka, aplikacija provjerava jesu li usluge lokacije na uređaju uključene te ne dozvoljava nastavak dok god nisu. Od iznimne je važnosti da aplikacija ima uvid u lokaciju svećenika u svakom trenu kako bi se ona mogla prikazivati vjernicima, ali i svećeniku, te kako bi procjene bile što preciznije. Da bi aplikacija mogla imati pristup GPS lokaciji uređaja, osim uključene usluge potrebno je imati i dopuštenje za njeno korištenje. Za dobiti

dopuštenje obavezno je u „AndroidManifest“ datoteci naznačiti da aplikacija koristi određenu uslugu, u ovom slučaju usluge lokacije. Kada se radi o korištenju usluga koje mogu biti opasne za sigurnost korisnika i podataka na uređaju, potrebno je i zatražiti dodatno dopuštenje od korisnika [14]. S obzirom da je čitanje lokacije uređaja radnja koja predstavlja sigurnosni rizik, prije nego se uđe u proces blagoslova, zatražit će se od korisnika dopuštenje za korištenje usluga lokacije. Korisnik zahtjev može prihvatiti ili odbiti, stoga je potrebno definirati reakciju za oba slučaja. S obzirom da je za korištenje ovog dijela aplikacije nužna lokacija uređaja, u slučaju da korisnik odbije zahtjev neće biti moguće ići dalje. Ukoliko aplikacija ipak dobije dopuštenje, izvođenje programa se nastavlja. Sada aplikacija provjerava postotak baterije na uređaju i upozorava svećenika da bude siguran da će imati dovoljno baterije za odraditi blagoslov. S obzirom da razina baterije uređaja nije informacija koja može ugroziti korisnika, za nju nije potrebno ni definirati korištenje ni tražiti dopuštenje. Ako svećenik potvrdi da je zadovoljan razinom baterije, proces blagoslova konačno započinje.

Aplikacija otvara novi *activity* koji ponovno koristi *bottom navigation* izbornik. Navigacija ima tri fragmenta koja su svećeniku potrebna u ovom trenu. Na prvom zaslonu prikazana je obitelj koja je trenutno na redu za blagoslov. Vidljive su sve osobne informacije uključujući i točnu lokaciju na mapi. Dostupna je i mogućnost upućivanja poziva jednim klikom. Na temelju ovih informacija svećenik odlazi u dom prikazane obitelji i ovisno o ishodu blagoslova označuje ga ili kao odrađen ili kao neodrađen. Nakon odabira oznake, aplikacija prikazuje podatke sljedeće obitelji na popisu koja je dostupna odmah, a ažurira se i brojač koji bilježi napredak na zaslonu.

Na drugom zaslonu nalaze se opcije blagoslova. Ovdje svećenik može uređivati napomenu koju vide svi vjernici, može upravljati pauzama ili završiti blagoslov. Kod pauze svećenik može odabrati otići na pauzu odmah, a može ju najaviti i za kasnije. Pauza može imati određeno ili neodređeno vrijeme trajanja, a ukoliko se najavljuje za kasnije, može postati aktivna nakon određenog broja blagoslova ili u određeno vrijeme. Kada je vrijeme za kraj blagoslova za taj dan, svećenik to označi ovdje. Proces blagoslova tada završava i svi vjernici vide da je kraj.

Na trećem zaslonu nalazi se popis svih obitelji u ovom blagoslovu. Uz jednostavnu pretragu i pregled svećenik još jednom ima uvid u sve informacije obitelji na popisu tako da vrlo jednostavno vidi za koje obitelji nije odrađen blagoslov, odnosno koje obitelji su preskočene

ili nisu dostupne. Potom se može organizirati odnosno napraviti plan kada će se vratiti u domove tih obitelji.

Tijekom rada, aplikacija ima potrebu znati određene informacije o korisniku, primjerice je li svećenik prijavljen, je li unesen kod vjernika, koji kod je unesen, koja župa je odabrana i slično. S obzirom da su ovi podaci značajni samo lokalnoj aplikaciji, ne čuvaju se u bazi podataka. Umjesto toga pohranjeni su u „SharedPreferences“ datoteci. „SharedPreferences“ [15] je lokalna datoteka koja sprema jednostavne podatke u obliku parova ključ vrijednost i javno je dostupna za čitanje i pisanje iz svakog dijela aplikacije. Automatski je dostupna u svakoj Android aplikaciji i dolazi s jednostavnim API-jem za uređivanje podataka.

U slučaju izlaska iz aplikacije u bilo kojem trenu, pamti se njeno stanje i ono se ponovno učitava kada se aplikacija ponovno otvori. Na ovaj način nije potrebno tražiti od korisnika ponavljanje istih radnji više puta. Primjerice ukoliko je vjernik već unio svoj kod i odabrao župu, aplikacija se odmah otvara u dijelu za vjernike s unesenim podacima. Ukoliko je svećenik bio prijavljen u trenutku izlaska, aplikacija otvara dio za svećenike, a svećenik ostaje prijavljen. A ukoliko je blagoslov u tijeku u trenu izlaska iz aplikacije, bit će učitan ponovno pri ulasku.

5. Testiranje i analiza rezultata

Nakon napornog rada, napokon je dostignut kraj. Uspješno su kreirani sustav i sve potrebne funkcionalnosti. No prije nego je potvrđeno da je sustav spreman za produkciju, odrađeno je testiranje svih njegovih funkcionalnosti i popravljene su sve pronađene greške. Otvorene su obje aplikacije u isto vrijeme, a i konzola u Firebase-u. Prošlo se kroz sve funkcionalnosti aplikacije i gledalo kako se podaci mijenjaju u realnom vremenu. Kreirani su novi korisnički računi za svećenika, registrirane župe, dodane nove obitelji na blagoslov… Započet je i sam proces blagoslova te su mijenjani podaci, a stanje praćeno na drugom uređaju kao vjernik. Svi mogući podaci su uneseni, dohvaćeni, izmijenjeni, a potom i obrisani kako bi se uvidjelo da sve funkcionira. Nakon što se dobila potvrda da sve radi, sustav je konačno spreman za produkciju.

Nakon puštanja sustava u produkciju nastavit će se pratiti njegov rad. Koristeći ugrađenu analitiku koju pruža Firebase vidjet će se informacije o broju korisnika i uspješnosti samog projekta. Vidjet će se i sve greške i detaljne informacije o njima, pa će ih se moći popraviti. S obzirom da se sustav koristi sezonski za očekivati je da neće biti velike aktivnosti tijekom godine, već samo tijekom blagdana. Malo dulje razdoblje korištenja može se očekivati za velike župe koje imaju veći popis obitelji, a ostale termine tijekom godine samo u iznimnim slučajevima.

Zaključak

Blagoslov kuća i obitelji proces je koji se već dugo vremena izvodi u istom obliku uz određene probleme. U ovom radu uspješno je kreiran sustav koji olakšava ovaj proces i rješava neke od problema kao što su nedostupnost informacija, gubljenje vremena, dugotrajan proces prikupljanja podataka ili korištenje zastarjelih i netočnih informacija. S obzirom da temeljni koncepti procesa ostaju isti, sustav nije obavezan za izvođenje samog blagoslova tako da manje informatički obrazovani korisnici neće zbog toga imati problema. Oni koji sustav ipak odluče koristiti uživat će brojne funkcionalnosti koje će im omogućiti jednostavniju izvedbu procesa. Kako sustav na korištenje nudi i mobilnu aplikaciju i onu za web, korisnici mogu pristupati potrebnim informacijama i uz veću preglednost na velikom zaslonu stolnog računala, a i uz luksuz mobilnosti koje pruža pametni telefon.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 28.03.2023.

_____

Popis kratica

API	*Application Programming Interface*	Sučelje za programiranje aplikacija
CSS	*Cascading Style Sheets*	Jezik za određivanje izgleda stranice
GPS	*Global Positioning System*	Globalni sustav pozicioniranja
HTML	*HyperText Markup Language*	Jezik za određivanje strukture stranice
JSON	*JavaScript Object Notation*	Format za prikaz seta podataka
NoSQL	*nonSQL*	Vrsta baze podataka bez pisanja koda
NPM	*Node package manager*	Menadžer paketa za Node
SQL	*Structured Query Language*	Jezik za relacijski model baze podataka
SSL	*Secure Socket Layer*	Protokol za sigurnu komunikaciju na internetu
UML	*Unified modeling language*	Jezik za unificirano modeliranje

Popis slika

Popis kôdova

Literatura

[1] Figma, https://www.figma.com/, preuzeto 18.01.2023.

[2] Javascript, https://www.w3schools.com/js/, preuzeto 18.01.2023.

[3] React biblioteka, https://reactjs.org/docs/hello-world.html, preuzeto 20.01.2023.

[4] Visual Studio Code, https://code.visualstudio.com/, preuzeto 15.01.2023.

[5] Kotlin, https://kotlinlang.org/, preuzeto 12.01.2023.

[6] Android i Kotlin, https://developer.android.com/kotlin, preuzeto 14.01.2023.

[7] Android Studio, https://developer.android.com/studio, preuzeto 12.01.2023.

[8] Firebase, https://firebase.google.com/, preuzeto 20.01.2023.

[9] Firebase hosting, https://firebase.google.com/products/hosting/, preuzeto 18.01.2023.

[10] Razlike baza u Firebaseu, https://firebase.google.com/docs/database/rtdb-vs-firestore/, preuzeto 18.01.2023.

[11] Firebase rad s korisnicima, https://firebase.google.com/docs/auth/web/manage-users/, preuzeto 18.01.2023.

[12] NPM, https://www.w3schools.com/whatis/whatis_npm.asp/, preuzeto 18.01.2023.

[13] BootstrapMade, https://bootstrapmade.com/, preuzeto 25.01.2023.

[14] Android dopuštenja, https://developer.android.com/guide/topics/permissions/overview/, preuzeto 18.01.2023.

[15] Android SharedPreferences, https://developer.android.com/training/data-storage/shared-preferences/, preuzeto 18.01.2023.

[16] Android RecyclerView, https://developer.android.com/develop/ui/views/layout/recyclerview/, preuzeto 18.01.2023.

[17] Popis naselja u Hrvatskoj, https://mfin.gov.hr/UserDocsImages/dokumenti/drzavna-riznica/upute-obrasci-zahtjeva-suglasnost/naselja.xls, preuzeto 15.01.2023.

Prilog

U prilogu ovog rada nalazi se CD sa svim datotekama u vezi ovog rada:

- programsko rješenje koje je kreirano sa svim korištenim materijalima
- *wireframe*
- ovaj rad u digitalnom formatu

Za korištenje Android aplikacije potrebno ju je instalirati na mobilni telefon koji ima Android operacijski sustav 6.0 ili noviji. Instalacijska datoteka ima naziv „android.apk“ i nalazi se u mapi „Produkcija“. Datoteku je potrebno pokrenuti i pratiti daljnje upute operacijskog sustava.

Web aplikacija može se pokrenuti tako da se datoteka „index.html“ otvori u internetskom pregledniku. Ova datoteka također se nalazi u mapi „Produkcija“.

Upravljanje servisnim slojem nije dostupno jer zahtjeva autentifikaciju. Umjesto toga u mapi „Produkcija“ dostupna je kopija čitave baze u „baza.json“ datoteci. U istoj mapi nalazi se i datoteka „korisnici.txt“ koja sadrži pristupne podatke korisnika za testiranje.