Sustav za upravljanje biljnim kulturama pomoću mobilnih uređaja

Kekelić, Bernard

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

Sveučilišni studij

SUSTAV ZA UPRAVLJANJE BILJNIM KULTURAMA POMOĆU MOBILNIH UREĐAJA

Diplomski rad

Bernard Kekelić

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 29.08.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Bernard Kekelić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-991R, 17.09.2019.
OIB studenta:	40866687143
Mentor:	Doc.dr.sc. Josip Balen
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	Doc.dr.sc. Josip Balen
Član Povjerenstva 2:	Dr.sc. Bruno Zorić
Naslov diplomskog rada:	Sustav za upravljanje biljnim kulturama pomoću mobilnih uređaja
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu rada potrebno je proučiti i opisati tehnologije vezane za izradu mobilnih aplikacija za Android platformu, arhitekturalni dizajn sustava, strukturu programskog koda i mrežne baze podataka. U praktičnom djelu rada potrebno je izraditi algoritam za obavještavanje korisnika na temelju praćenja vremenskih parametara i karakteristika biljnih kultura. Algoritam je potrebno ugraditi u Android mobilnu aplikaciju koja će omogućiti poljoprivrednicima učinkovitije upravljanje biljnim kulturama (ratarstvo, voćarstvo, vinogradarstvo, povrtlarstvo), spremanje i dohvaćanje svih informacija o pojedinoj kulturi na temelju geografske lokacije, prikaz na karti, obavještavanje, upravljanje dokumentima, kalendarom i
Prijedlog ocjene pismenog dijela ispita (diplomskog 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: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	29.08.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2020.

Ime i prezime studenta:	Bernard Kekelić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-991R, 17.09.2019.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Sustav za upravljanje biljnim kulturama pomoću mobilnih uređaja**

izrađen pod vodstvom mentora Doc.dr.sc. Josip Balen

i sumentora

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:

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

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Bernard Kekelić_____, OIB: __40866687143_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Diplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Sustav za upravljanje biljnim kulturama pomoću mobilnih uređaja**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima/icama i djelatnicima/ama ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 23.09.2020._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Razvojem znanosti pojavljuje se potreba i prilika za ugradnju i integriranje tehnologije u nove, do tada ne toliko tehnološki zastupljene industrijske grane. Jedna od njih je i poljoprivreda. Gospodarska djelatnost koju njegujemo još od pamtivijeka, koja je ugrađena u naše korijene, koja nas hrani i održava na životu. Samim time poljoprivreda ima određenu težinu, treba napredovati, rasti i ići u korak s vremenom koje je tehnološki snažnije nego ikada prije. Nadogradnjom poljoprivredne djelatnosti tako da se sustavno organiziraju djelatnosti, bilo to na obiteljskom poljoprivrednom gospodarstvu ili u nekoj većoj organizaciji, znatno bi se povećala efikasnost rada i sami prihodi, što je u konačnici i cilj. Mobilni uređaji današnjice posjeduju mnoštvo funkcionalnosti koje bi omogućile prikupljanje, analizu i obradu podataka vezanih za pojedine poljoprivredne kulture, predviđanja vremenske prognoze i slično. Korisnik bi bio pravovremeno obaviješten o vremenskoj neprilici, o bolesti koja može poharati njegov usjev i plod te tako uništiti sav njegov trud ili ga pak obavijestiti kako njegova biljka preživljava nepovoljne vremenske uvijete koji su ju snašli. Korisnik bi mogao pravovremeno reagirati, zaštititi svoj usjev, i to sve uz pomoć nove tehnologije koju svatko posjeduje – mobilni uređaj. Znatno bi se povećala kvaliteta a samim time i kvantiteta krajnjih proizvoda.

Gospodarstva koja obrađuju mnoštvo zemlje, računala se ona u nekoliko hektara ili nekoliko stotina hektara, trebaju nekako voditi bilješke, zapise, događaje o svojim postupcima, i to sve za svako zemljište posebno koje obrađuju. Mobilni uređaji i njihova prenosivost, funkcionalnost i dostupnost čine se kao savršeno rješenje za navedeni problem. Kao dio ovog rada bit će kreirana Android mobilna aplikacija koja će korisniku omogućiti spremanje i dohvaćanje svih informacija o pojedinoj kulturi na temelju geografske lokacije, prikaz informacija na karti, upravljanje dokumentima i događajima, uvid u vremensku prognozu za pojedino zemljište te algoritam koji će korisnika obavještavati o mogućim vremenskim neprilikama, bolestima ili štetnim uvjetima za pojedinu kulturu koju posjeduje temeljeno na njenim karakteristikama. Osim toga, korisniku treba biti omogućena prenosivost podataka pa će se u tu svrhu koristiti mrežna baza podataka. Samim time, korisnik se neće morati bojati da će njegovi podatci biti izgubljeni.

U drugom poglavlju opisane su glavne značajke sustava za upravljanje poljoprivrednim djelatnostima, što su ti sustavi te koje probleme oni rješavaju. Također, dana je usporedba sa sličnim postojećim rješenjima i sustavima. Treće poglavlje sadržava opis tehnologija i alata korištenih u izradi ovog rada. Dani su razlozi korištenja određenih tehnologija te usporedba s njima sličnima. Navedena su svojstva Android mobilnih uređaja, arhitektura Android platforme, te kratki uvid u lokacijske senzore. Također, opisane su prednosti programskog jezika Kotlin i Postman

alata za testiranje aplikacijskog programskog sučelja (engl. *API*). Četvrto poglavlje sadržava detaljan opis arhitekture cjelokupnog sustava i arhitekture programskog koda. U posljednjem, petom poglavlju opisana je mobilna aplikacija i rješenje koje je dobiveno kreiranjem sustava te su dani primjeri zaslona s detaljnim opisom.

2. SUSTAVI ZA UPRAVLJANJE POLJOPRIVREDNIM DJELATNOSTIMA

Napretkom civilizacije, omogućuje se lakši i bezbolniji život pojedinca te se samim time ljudska populacija povećava. Prema [1], trenutna ljudska populacija broj 7.8 milijardi ljudi te iz dana u dan raste. Od 1960. godine, ljudska populacija se povećava eksponencijalno za milijardu ljudi svakih 13 godina, dok se do 2100. godine očekuje gotovo 11 milijardi ljudi na svijetu. Sve veći broj ljudi na zemlji zahtjeva i veću količinu hrane koju je potrebno proizvesti. Kako je navedeno u [2], preko 800 milijuna ljudi na svijetu je trenutno gladno, što znači da jedan od devet ljudi trenutno gladuje. Hrana i proizvodnja hrane predstavlja sve veći izazov ljudskoj civilizaciji.

Pomoć poljoprivredniku u današnje vrijeme je neophodna te se u tu svrhu razvijaju razni programski alati i sustavi koji će olakšati vođenje i upravljanje poljoprivrednim aktivnostima. Današnja programska rješenja korisniku ostavljaju više vremena za samu poljoprivrednu djelatnost te minimiziraju nepotrebne aktivnosti, a vrijeme i djelovanja usmjeruju prema korisnikovim željama. Poljoprivredne djelatnosti su razne, pa samim time i programsko rješenje mora biti fleksibilno, sposobno se prilagoditi raznim uvjetima i potrebama. Prema [3], bitni zahtjevi na programsko rješenje za upravljanje poljoprivrednim djelatnostima su:

- znanje o poljoprivredi
- upravljanje resursima
- upravljanje poslovanjem
- stupanj autonomnosti poljoprivrednog gospodarstva
- ekonomski aspekt

Prvotni ciljevi sustava za upravljanjem poljoprivrednim djelatnostima su veća profitabilnost i učinkovitost, a do njih se dolazi efikasnim i ispravnim korištenjem datih funkcionalnosti sustava. Kako bi sustav bio funkcionalan, mora implementirati barem nekoliko bitnih funkcionalnosti i zahtjeva navedenih u prethodnom odlomku, dok se kao glavni i najbitniji zahtjev predstavlja općenito znanje o poljoprivredi. Većina sustava za upravljanje poljoprivrednim djelatnostima trebaju uključiti upravljanje zemljištima i usjevima, upravljanje prihodima i rashodima, financijska izvješća, upravljanje poslovima i zadatcima te njihovo praćenje, praćenje vremenske prognoze, aktivnosti na zemljištima i usjevima, upravljanje pokretnom i nepokretnom imovinom, prikaz zemljišta na karti, praćenje aktivnosti i stanja vezano za životinje te upravljanje rizikom, navodi se u [3, 4].

Sve to vrijeme utrošeno na zapisivanje i vođenje papirologije, događaja, troškova i postupaka bi se moglo upotrijebiti na nešto korisnije i svrsishodnije, pa se u novije vrijeme sve više poljoprivrednika odlučuje koristiti takva ili slična programska rješenja. Ne samo da povisuje efikasnost i profitabilnost već i smanjuje mogućnost pogreške i krivog interpretiranja i zapisa. Kako je navedeno u [5], sustav za upravljanjem poslovanja je sustav koji pomaže korisnicima donositi odluke i rješavati probleme. Dijelovi jednog takvog sustava su: korisnici, programi, podaci i računala. Korisnici kao krajnji i početni sudionici u tom sustavu donose odluke na gospodarstvu na temelju rezultata koji su dobili od programa. Kako bi se program izvršavao ispravno, potrebno je koristiti točne i valjane podatke do kojih je danas teško doći, jer poljoprivreda je širom rasprostranjena u svijetu, no ipak, razlikuje se od države do države, pa posebice za svaku kulturu, tip zemlje koji se obrađuje, klimatski utjecaj i tako dalje. Sustav za upravljanje poljoprivrednim djelatnostima rješava taj problem. On omogućuje podjelu znanja, prilagodbu različitim uvjetima i načinima proizvodnje i poljoprivrede. Korisnici ne samo što mogu donijeti odluku na temelju nekih podataka, već sada mogu i nešto novo naučiti, a stečeno znanje i informacije primijeniti na svojem gospodarstvu.

Kako bi sustav bio upotrebljiv potrebno ga je prilagoditi korisniku i njegovom načinu poslovanja i poljoprivrede. Poljoprivreda nije centralizirana djelatnost, u većini slučajeva se ne obavlja iz jednog mjesta, već su djelatnosti rasprostranjene po zemljištima koja su nerijetko udaljena i po nekoliko desetaka kilometara jedno od drugog te bi sustav trebao omogućiti korisniku neku razinu mobilnosti. U tu svrhu standardno računalo može se zamijeniti pametnim mobilnim uređajima koji u današnjici posjeduju sve funkcionalnosti kao i računalo.

Takav način izvedbe programskog rješenja prepoznale su i razne računalne industrije te se fokusiraju na spoj mobilnosti, sigurnosti i funkcionalnosti. Pametni mobilni uređaji i njihove mogućnosti u kombinaciji sa spremanjem podataka na oblaku (engl. *cloud*) te raznim internetskim servisima uvelike su doprinijele razvoju sustava za pomoć i upravljanje poljoprivrednim djelatnostima. Jedna od vodećih svjetskih tvrtki u toj branši je i Agrivi, hrvatska tvrtka koja je osnovana 2013. godine, a do sada posluju u više od 150 zemalja s poljoprivrednim kooperacijama, prerađivačima hrane, vladama, neprofitnim organizacijama i malim poljoprivrednicima, kako je navedeno u [6]. Njihovo rješenje sastoji se od upravljanja nasadima i usjevima, upravljanje financijama, vremenska prognoza i detekcija bolesti i štetnika, upravljanje resursima i zalihama te analiza proizvodnje i izvještaji. Za razliku od Agrivi sustava koji nudi programsko rješenje u obliku web i mobilne aplikacije, u ovom radu kreirana je Android mobilna aplikacija s funkcionalnostima za upravljanje zemljištima, nasadima i usjevima, uvid u vremensku prognozu te obavještavanje o mogućim vremenskim nepogodama ovisno o karakteristikama biljnih kultura,

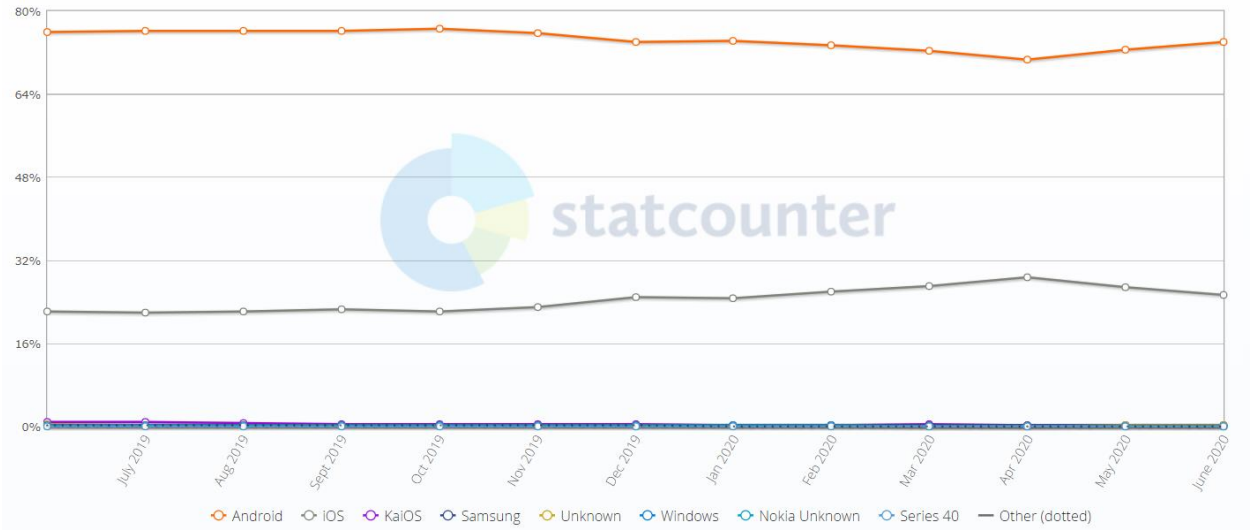
upravljanje događajima na temelju geografske lokacije, upravljanje dokumentima i drugo. Cilj rada je povezati zemljišta s događajima, jer je poljoprivredna djelatnost događaj koji se izvršava na zemljištu. Osim toga, aplikacija podržava različite korisnike, te višekorisnički rad s određenim podacima nekog gospodarstva. Slično rješenje donosi ruska tvrtka ExactFarming, koja korisnicima nudi korištenje modernih tehnologija za povećavanje prihoda i profita. Nude programsko rješenje u obliku web aplikacije i aplikacije za mobilne uređaje. Prema [7], također, kao i aplikacija kreirana u ovom radu, nude upravljanje zemljištima, te podatcima vezanim za geografski položaj, te uvid u relevantne informacije vezane za vrijeme, vegetaciju, temperaturu i slično. Kako je navedeno u [8], ostala značajnija rješenja su: Granular, Trimble, FarmERP, FarmLogs i drugi.

3. TEHNOLOGIJE I ALATI KORIŠTENI U IZRADI RADA

Pametni mobilni uređaji su promijenili način života i svakodnevicu ljudi. Danas, bez puno truda i muke šaljemo poruke, razmjenjujemo informacije, dokumente i fotografije. Sve te tehnologije su prije bile pojedinačne, dok su danas ukomponirane u jednu revolucionarnu tehnologiju koja postaje pametnija i privrženija nam iz dana u dan.

3.1. Android mobilni uređaji

Android mobilni uređaji mogu se opisati kao uređaji koje pokreće Android operacijski sustav, imaju zaslon osjetljiv na dodir raznih dimenzija (pametni uređaji i tableti) i sadrže razne senzore i programske alate. Operacijski sustav baziran je na Linux jezgri čije su karakteristike: brzina, sigurnost i performanse, što je jedan od glavnih prednosti nad ostalim uređajima slične kategorije. Rasprostranjenost i dobra korisnička upoznatost s Android uređajima jedna je od glavnih svojstava zbog koje je ova vrsta uređaja odabrana kao kandidat za implementaciju u sustav za upravljanje biljnim kulturama. Na slici 3.1. prikazana je usporedba rasprostranjenosti Android operacijskih sustava u usporedbi s ostalim operacijskim sustavima na pametnim uređajima (iOS, Windows i drugi) u razdoblju od svibnja 2019. godine do svibnja 2020. godine.



Slika 3.1. *Podjela operacijskih sustava mobilnih uređaja na tržištu [9]*

Kako slika 3.1. prikazuje, Android mobilni uređaji su preplavili tržište i široko dominiraju njime. No, početci i nisu bili tako bajni. Borba između Applea i tada Googleovog Android sustava za prevlast na tržištu pametnih uređaja datira još od 2007. godine, kada je Apple pustio u prodaju

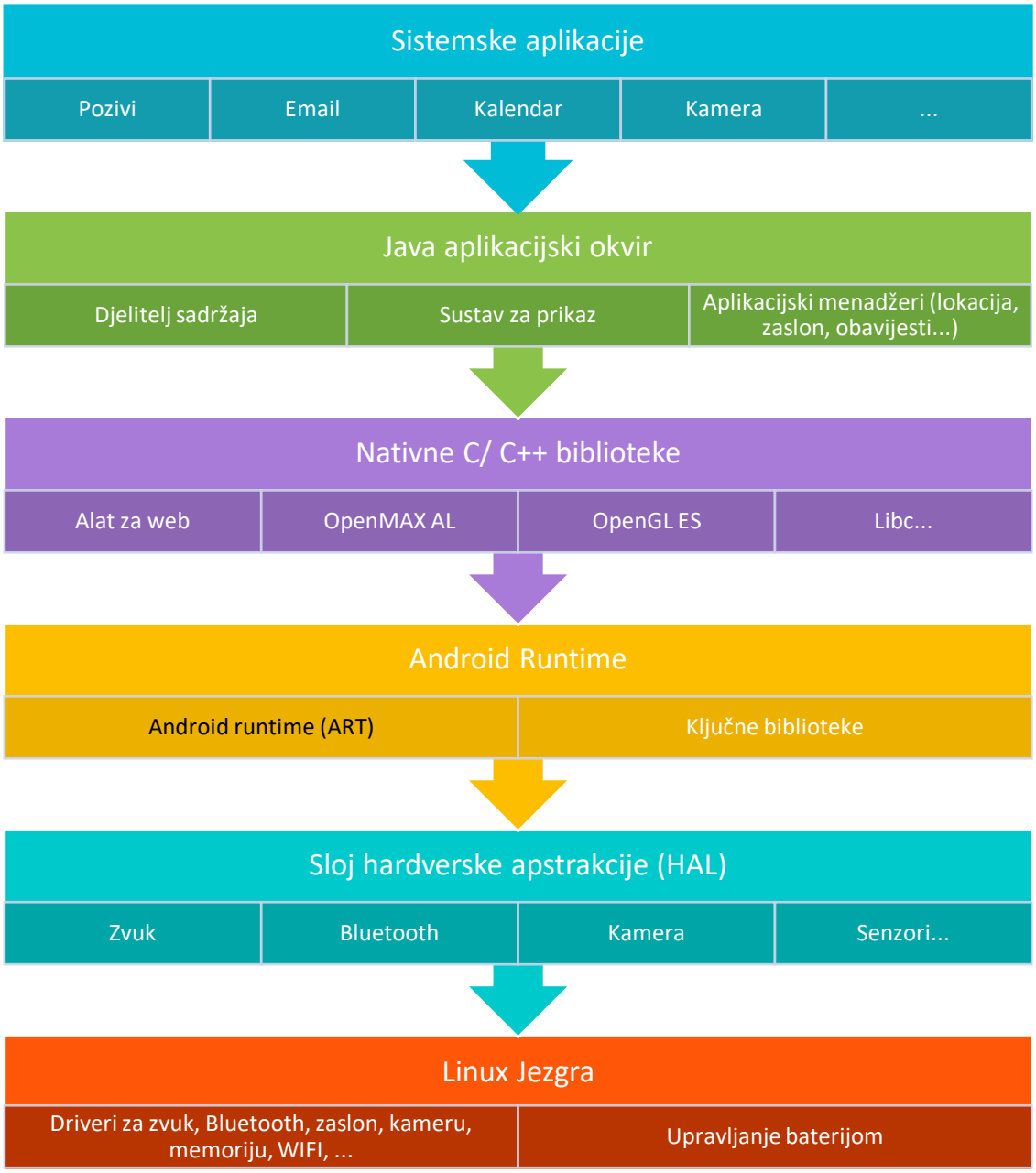
svoj prvi iPhone, dok je Google u tajnosti još radio na Android sustavu. Naime, Android operacijski sustav isprva nije bio zamišljen kao podloga ta pametne mobilne uređaje. 2003. godine, puno prije nego su postojali izrazi kao pametni uređaji (engl. *Smartphone*), osnivači Android sustava osmišljavali su kako kreirati operacijski sustav za digitalne kamere, kako ih povezati s internetom te omogućiti spremanje podataka u oblak računala (engl. *Cloud*). Ipak, kako je vrijeme odmicalo, tržište digitalnih kamera je propadalo te se drastično smanjivalo te dolazi do prekretnice u povijesti Android operacijskog sustava implementiranjem tog istog operacijskog sustava u mobilne uređaje, navedeno je u [10].

Definiranjem nove ponude na tržištu, bilo je potrebno ponuditi nešto jedinstveno i novo, pružiti kupcima bolje i nadasve jeftinije mogućnosti. Kako ih je Apple pretekao u tome i uspio prvi pustiti u prodaju svoj mobilni uređaj – iPhone, u Googleu su trebali drugačiju ideju. Tu nastupa njihova dobitna karta, Linux. Android operacijski sustav kao bazu koristi Linux operacijski sustav koji ima besplatan i svima dostupan izvorni programski kod (engl. *Open source*). Tim potezom, Android sustav može biti ponuđen raznim trećim stranama, posebice svjetski poznatim i priznatim proizvođačima mobilnih uređaja, kako bi besplatno ugradili Android operacijski sustav u svoje proizvode. Naravno, ovu odluku većina je brzo prihvatila, a oni koji nisu, ubrzo su propali. Jedan takav primjer je i Nokia, koja je izgubila bitku s Googleom i Appleom na tržištu mobilnih uređaja jer su gurali svoju tehnologiju a nisu dovoljno promatrali i osluškivali tržište. Naravno, bitno je napomenuti kako su u konačnici, kao slamku spasa, odlučili implementirati Android operacijski sustav u svoje mobilne uređaje.

3.1.1. Arhitektura Android platforme

Kako je rečeno u prethodnom odlomku, Android operacijski sustava je zasnovan na Linux jezgri, što mu daje nebrojene mogućnosti i bolje performanse. Samim time, moguće ga je implementirati na razne hardverske arhitekture i komponente. Prikaz arhitekture Android platforme dan je na slici 3.2.

Android sustav u većini slučajeva dolazi s instaliranim ključnim aplikacijama kako bi sam sustav mogao funkcionirati a i pružiti korisniku dodatne mogućnosti. To su na primjer aplikacije za SMS i pozive, kalendar, aplikacija za elektroničku poštu, Internet pretraživač, alarm i drugo. Osim već instaliranih, korisnik može instalirati željene mu aplikacije preko raznih aplikacija koje to nude, a najpoznatija je Google Play Store. U novijim verzijama Android sustava omogućeno je pozivanje jedne aplikacije iz druge. Pa tako ako primjerice želimo uputiti poziv, razvojni inženjer ne mora osmišljavati cijelu logiku i razvijati novu aplikaciju za pozive, već jednostavno pozove već instaliranu aplikaciju s očekivanim parametrima, navedeno je u [11].



Slika 3.2 *Arhitektura Android platforme*

Kako bi razvojni inženjer mogao koristiti funkcionalnosti Android operacijskog razvijen je sloj koji predstavlja razvojni okvir za programsko sučelje, a napisan je u programskom jeziku Java (engl. *Java API Framework*). Sastoji se od raznih modula koje razvojni inženjer koristi kada želi koristiti neke od sklopovskih ili programskih mogućnosti Android operacijskog sustava. Sastoji se od modula koji se brine za prikaz (engl. *View System*) putem kojeg se kreira korisničko sučelje, tekst, gumbovi, liste i slično. Modul za upravljanje resursima (engl. *Resource Manager*) omogućuje pristup raznim ilustracijama, slikama, ikonama, sučeljima te omogućuje prevođenje tekstova na više jezika. Nadalje, ovaj sloj upravlja obavijestima (engl. *Notification Manager*),

upravlja životnim ciklusom aplikacija (engl. *Activity Manager*) i brine se o razmjeni podataka između aplikacija (engl. *Content Provider*), npr. dohvaćanje spremljene slike iz aplikacije za slanje elektroničke pošte.

Prema istraživanju provedenom u [12], programi pisani Java programskim jezikom ne daju izrazite performanse, pa se u tu svrhu koristi C ili C++ programski jezik koji je u raznim situacijama i do nekoliko puta brži i efikasniji, pogotovo gledajući korištenje računalnih resursa. Zato Android posjeduje sloj koji sadrži biblioteke napisane u C i C++ jeziku. Takve biblioteke lako upravljaju 2D i 3D grafikama, a dostupne su preko Java sučelja.

Od Android sustava 5.0 (API razina 21), svaka aplikacija se pokreće u jedinstvenom procesu i sa svojom instancom koja se brine o radu Android aplikacija (engl. *Android Runtime -ART*). Glavna svojstva Android Runetime procesa su kompiliranje koda u 2 načina: prijevremeno (engl. *Ahead of time*) i pravovremeno (engl. *Just in time*), optimiziranje prikupljanje memorijskog otpada (engl. *Garbage collection*) koje predstavlja velik problem u sustavima s malom količinom memorije, bolja podrška prilikom pronalaženja pogrešaka u kodu, dijagnostika i slično. Prije verzije 21, umjesto Android Runetime procesa koristio se Dalvik proces.

Sloj koji je pruža sučelje prema sklopovskim funkcionalnostima uređaja naziva se sloj apstrakcije sklopovlja (engl. *Hardware Abastraction Layer*). Sastoji se od više modula koji implementiraju sučelje prema određenoj sklopovskoj komponenti, kao na primjer kameri ili Bluetooth funkcionalnosti. Kada aplikacija zatraži određeni dio sklopovlja na korištenje, Android sustav učita biblioteke tog modula u memoriju te joj omogući korištenje i pristup.

Kako je rečeno u potpoglavlju 3.1., Linux jezgra je temelj Android platforme. Korištenje Linux jezgre definira veću razinu sigurnosti u sustavu, a i proizvođačima sklopovlja kreiranje programa za upravljanje sklopovljem za jako poznatu jezgru. Osim toga, Linux jezgra omogućuje Android Runetime procesu višenitnost i upravljanje memorijom.

3.1.2. Lokacijski senzori

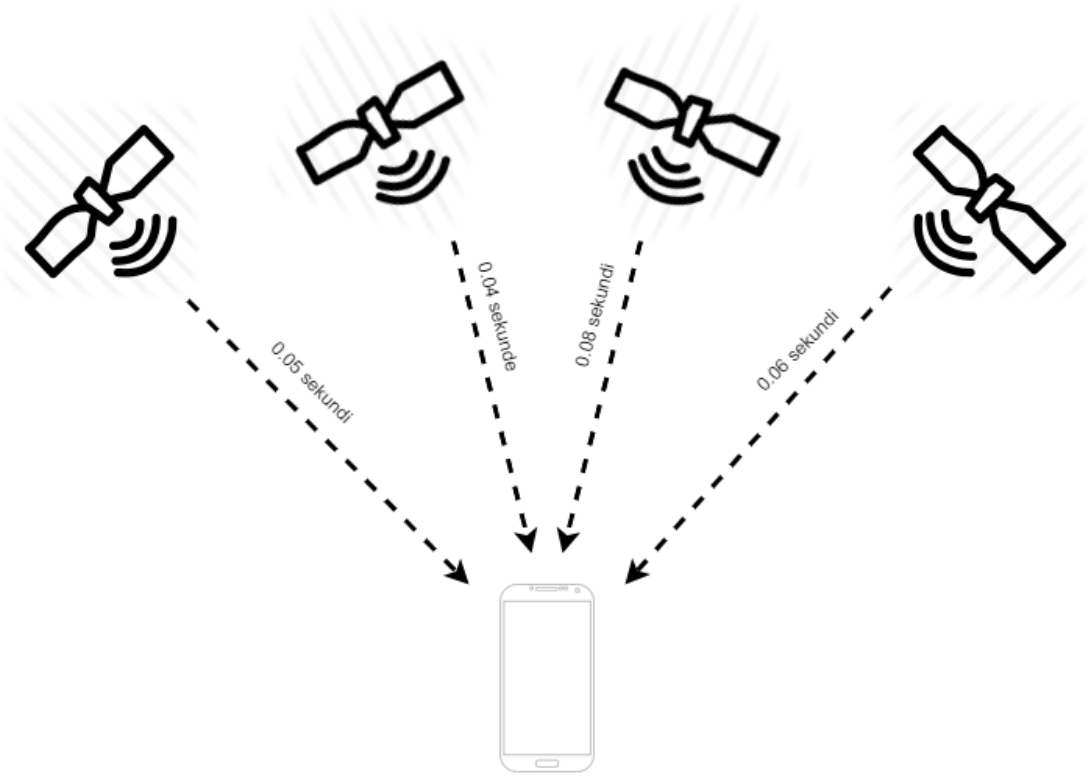
Prvi mobilni uređaji pogonjeni Android operacijskim sustavom, razlikovali su se od ostalih prvenstveno po svojoj cijeni, bili su mnogo jeftiniji od drugih. To svojstvo zasigurno i danas ima veliku ulogu u samoj preraspodjeli tržišta mobilnih uređaja, koji je višestruko na strani Androida. Prema [13, 14], nije samo cijena prevladavala pri odabiru, Android 1.0 nudio je mnogo toga što je temelj današnje tehnologije mobilnih uređaja: sinkronizacija podataka s oblakom, obavijesti, organizacija, Google Play Store i preuzimanje aplikacija, kalendari, mape i slično. Sva ova tehnologija koristi se i danas samo je naprednija i svrsishodnija kada se kombinira s internetom i velikim brzinama razmjena podataka, korištenjem raznih senzora i tehnologija. Tako će se i u

ovom radu koristiti razne mogućnosti i sklopovske funkcionalnosti Android uređaja. Kako je navedeno u [15], Android platforma podržava tri kategorije senzora i oni se mogu prikazati kao:

- Senzori pokreta
- Senzori okoline
- Lokacijski senzori

Senzori koji omogućuju praćenje i detektiranje pokreta mjere sile ubrzavanja i rotacijske sile na 3 glavne osi pomoću žiroskopa, akcelerometra, senzora gravitacije i rotacijskih senzora. Barometri, termometri i fotometri koriste se kao senzori okoline a mjere razne parametre okoline kao što su temperatura zraka, vlažnost, atmosferski tlak i svjetlost. Kako bi korisnik mogao uvidjeti svoju lokaciju, Android uređaji posjeduju orijentacijski senzor i magnetometar.

O ovom radu najveću ulogu imat će senzor lokacije. On koristi sustav za globalno pozicioniranje (engl. *GPS - Global Positioning System*) koji je prema [16] trenutno u vlasništvu vlade SAD-a no besplatan je za korištenje. GPS je osmišljen tako da šalje radio valove sa satelita koje kruže oko zemlje do uređaja koji zahtjeva svoju lokaciju. Naime, uređaj ne treba slati satelitu nikakve podatke, već samo mora moći primiti signale s minimalno 3 satelita, dok se četvrti koristi kao provjera ispravnosti sustava i za određivanje nadmorske visine. Ilustracija rada GPS-a prikazana je na slici 3.3.



Slika 3.3. *Prikaz rada GPS-a [18]*

Svaki satelit posjeduje svoj interni atomski sat čije vrijeme šalje u pravilnoj frekvenciji. Senzor na mobilnom uređaju određuje koji sateliti su mu vidljivi te prikuplja njihove podatke o lokaciji i vremenske informacije. Znajući da mora postojati kašnjenje između poslanog i primljenog signala, uređaj određuje svoju lokaciju na temelju 4 odabrana satelita. Kada bi bio korišten samo jedan satelit, na zemlji bi se mogla ocrtati kružnica s točnim lokacijama. Korištenjem 2 satelita, dobivene su dvije kružnice s dvije zajedničke točke, što znači kako korisnik može biti na bilo koje od te dvije točke. Dodavanjem još jednog satelita eliminira se lažna lokacija te je lokacija uređaja otkrivena. Četvrti satelit koristi se za detektiranje mogućih grešaka u sustavu ili pak za određivanje nadmorske visine uređaja.

3.2. Razvojno okruženje Android Studio

Za razvoj Android aplikacija potrebno je razvojno okruženje u kojem razvojni inženjer piše programski kod te samim time i razvija aplikaciju. Danas se najčešće koriste razvojna okruženja Android Studio, Eclipse ili Unity. Eclipse razvojno okruženje (engl. *IDE - Integrated Development Environment*) je u neka prošla vremena bio glavni IDE za razvoj Android aplikacija, dok im je danas fokus na Java aplikacije i slično. Ako se želi razvijati igra onda je pravi odabir Unity IDE.

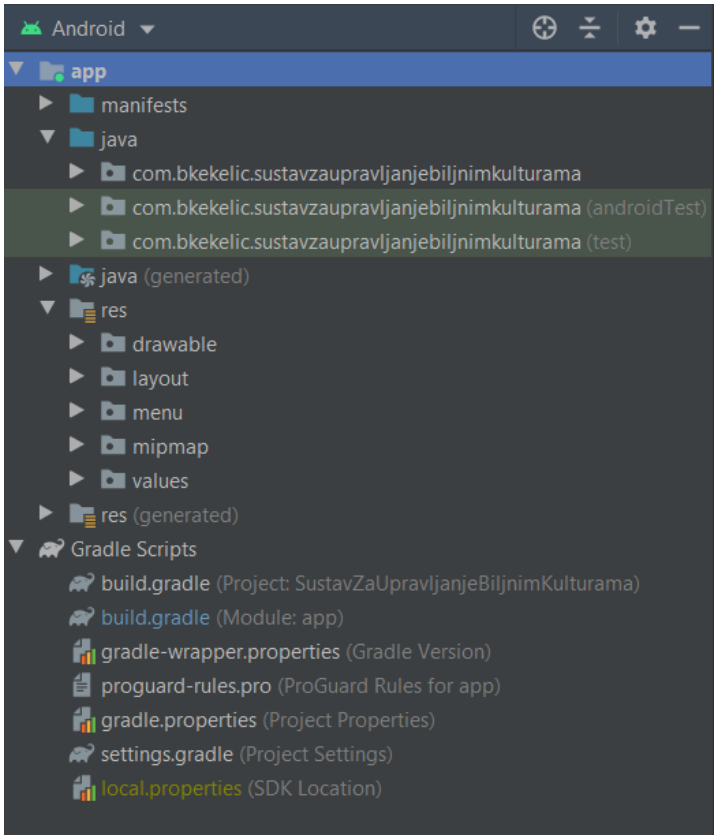
Kako je navedeno u [17], gotovo sve Android mobilne aplikacije razvijaju se u razvojnom okruženju naziva Android Studio. Android Studio je službeno razvojno okruženje za razvoj Android mobilnih aplikacija, a mnoge sličnosti ima s IntelliJ IDEA razvojnim okruženjem, na kojem je i baziran. Osim uređivača koda (engl. *Code editor*) i razvojnih alata (engl. *Developer tools*) koji su bazirani na IntelliJ IDEA razvojnom okruženju, razvojno okruženje Android Studio nudi dodatne mogućnosti za razvoj Android aplikacija, a to su:

- Moćan i fleksibilan sustav za kreiranje aplikacija - Gradle
- Brzi i funkcionalni emulatori koji simuliraju Android mobilne uređaje
- Jedinstveno sučelje za razvoj Android aplikacija za sve vrste Android uređaja
- Prikaz promjena na aplikaciji bez ponovnog pokretanja aplikacije (engl. *Instant Run*)
- Dobra integracija sa sustavima za verzioniranje koda
- Proširivi alati i razvojni okviri za testiranje
- Lint alati za praćenje performansi, potrošnje resursa i slično
- Podrška za razvoj aplikacija u C++ programskom jeziku preko alata za razvoj Android aplikacija u nativnim jezicima (engl. NDK - *Native Development Kit*) i drugo.

Android projekt sastoji se modula koji sadrže izvorni kod i modula s resurs datotekama. Kako je vidljivo na slici 3.4., glavna dva modula od kojih se sastoji Android projekt je modul s aplikacijom – *app* i modul s Gradle skriptama (engl. *Gradle Scripts*). Modul aplikacije sadrži nekoliko ključnih direktorija aplikacije, a među njima su: manifest direktorij, java direktorij, direktorij s resursima i direktoriji s generiranim kodom za resurse i java kod.

Ključna datoteka projekta je AndroidManifest.xml datoteka koja sadrži sve bitne postavke aplikacije, kao što su: popis dopuštenja aplikacija, ikona i naziv aplikacije, lista zaslona i popis servisa, tema aplikacije i drugo. Unutar java direktorija nalazi se sav programski kod raspoređen u pakete koje razvojni inženjer raspoređuje po određenim obrascima. Direktorij s resursima (engl. *res*) sadrži sve resurse koji su potrebni aplikaciji: ikone aplikacije u raznim dimenzijama, XML datoteke s podatcima za kreiranje zaslona i elemenata zaslona te resursi u obliku tekstova, boja, dimenzija i slično.

Android Studio koristi Gradle kao alat za kreiranje aplikacije, a može se koristiti direktno preko grafičkog sučelja (engl. GUI - *Graphical User Interface*). Gradle datoteke su tekstualne datoteke pisane Groovy sintaksom a svaki projekt sadrži jednu datoteku za konfiguraciju na projektnoj razini i na razini svakog modula. Također Gradle se brine za korištenje raznih vanjskih paketa koje korisnik želi koristiti za izradu aplikacije.



Slika 3.4. *Struktura Android projekta*

Nakon napravljene promjene u Gradle datoteci potrebno je izvršiti sinkronizaciju koja se može izvršiti preko grafičkog sučelja ili pokretanjem odgovarajuće Gradle skripte. Gradle skripte su iznimno bitne za konfiguraciju sustava, a neka od svojstva su:

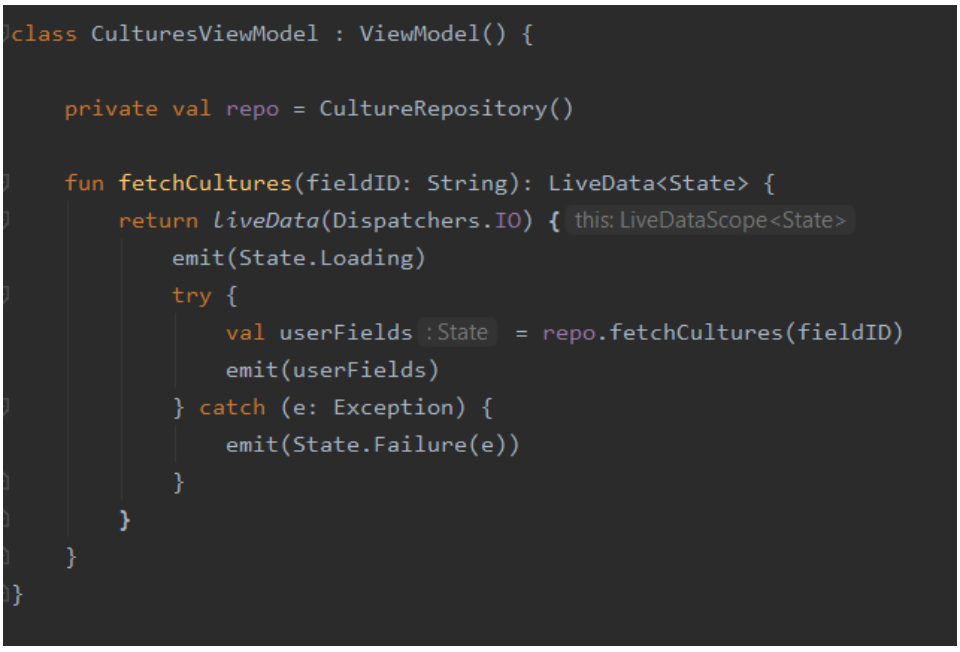
- Prilagodba, konfiguracija i proširenje procesa za kreiranje aplikacija (engl. *build*)
- Kreiranje višestrukih verzija aplikacije, s drugačijim svojstvima. Korisno kada se želi kreirati besplatna i verzija koja se plaća, ili pak ako se aplikacija želi prilagoditi određenom tržištu.
- Korištenje resursa i programskog koda bilo gdje u izvornom kodu aplikacije

Osim toga, u Gradle datoteke pohranjuju se i druge bitne postavke aplikacije pri razvoju. Neke od bitnijih su: minimalna verzija razvojnog alata (engl. *minSdkVersion*), ciljana verzija razvojnog alata (engl. *targetSdkVersion*) i verzija razvojnog alata prema kojoj će se aplikacija kreirati (engl. *compileSdkVersion*). Također, Gradle je veoma moćan alat za kreiranje *.jar* datoteka što omogućuje izdvajanje kompletnog projekta u zasebnu datoteku koja se može učitati u drugi projekt te tako koristiti funkcionalnosti, klase i metode iz učitanog projekta.

3.3. Programski jezik Kotlin

Za razvoj Android mobilnih aplikacija najčešće se koriste programski jezici Java i Kotlin. Iako je programski jezik Java dugo bio prvi izbor za razvoj Android aplikacija, Google je 2019. godine predstavio Kotlin kao prvi jezik za razvoj Android mobilnih aplikacija. Također, kako je navedeno u [18], 55 Googleovih aplikacija trenutno je napisano u Kotlinu, a to uključuje aplikacije: Maps, Home, Pay, Play i Drive. Korištenjem programskog jezika Kotlin, smanjili su količinu programskog koda za 33 %, dok se broj grešaka i ispada u aplikaciji smanjio za 30 %.

Prema [19], Kotlin je statički pisani programski jezik (engl. *Statically typed programming language*) što znači da je svaka ekspresija u programu poznata kompajleru (engl. *Complier*) za vrijeme izvođenja (engl. *Compile time*). Time kompajler može predložiti i ispravljati pogreške u metodama, klasama i varijablama. Jezici koji ne sadrže takve prijedloge i ispravke tijekom pisanja koda nazivaju se dinamički pisani programski jezici (engl. *Dynamically typed programming languages*) a jedan od njih je i Groovy, koji se koristi u Gradle sustavu, a spomenut je u potpoglavlju 3.3. Primjer programskog koda napisanog u Kotlin programskom jeziku dan je slikom 3.5., a predstavlja klasu koja implementira metodu za dohvaćanje spremljenih kultura određenog zemljišta na temelju identifikatora zemljišta.

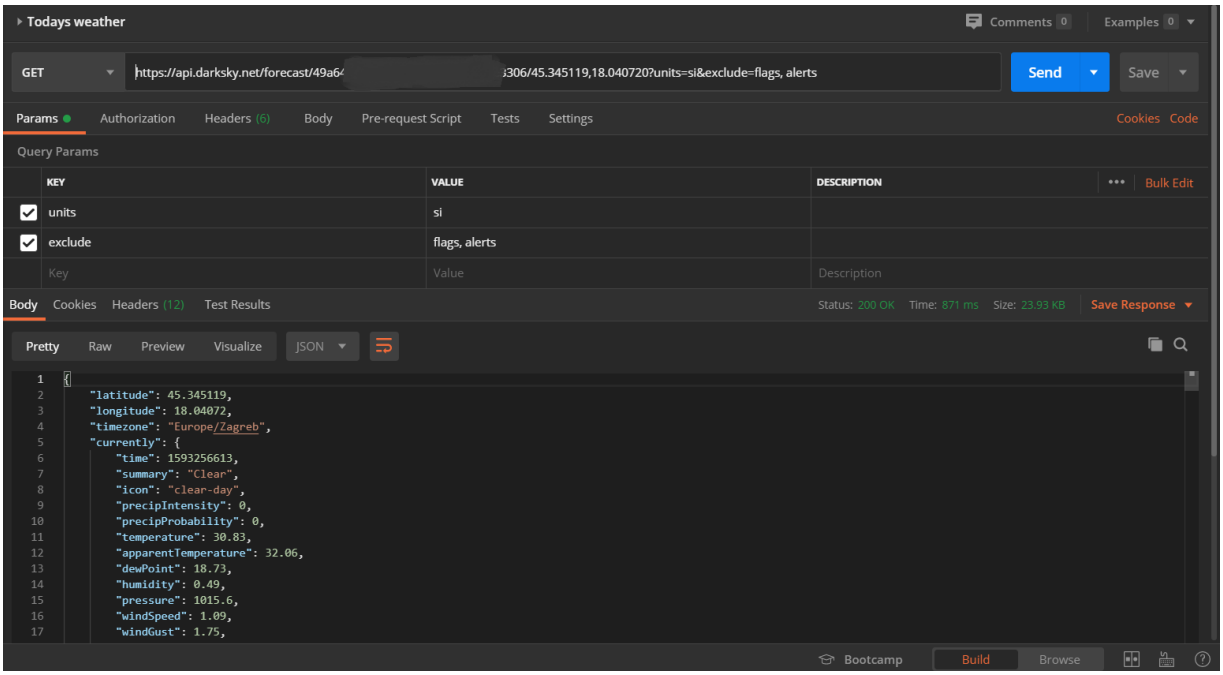


Slika 3.5. *Programski jezik Kotlin*

Jedna od glavnih karakteristika Kotlin programskog jezika je podrška provjere nulabilnosti tipova (engl. *Nullable types*) podataka koje omogućuje pisanje programskog koda koji je siguran od ispada uzrokovanih iznimkama null vrijednosti (engl. *Null pointer exceptions*) korištenjem Elvis operatora (*?:*). Osim toga Kotlin podržava i lambda izraze (engl. *Lambda expressions*) koje se baziraju na anonimnim funkcijama, sigurnu višenitnost, lakše testiranje, podatkovne klase (engl. *Data classes*) te razne API-je za standardne biblioteke. Osim navedenih razloga, Kotlin programski jezik je korišten za izradu Android aplikacije u ovom radu i zato što je besplatan i otvorenog koda, s puno raznih biblioteka i alata koje ga podržavaju.

3.4. Programski alat Postman

Programski alat Postman je platforma za razvoj i testiranje API-ja. Prema [20], Postman omogućuje brzo i jednostavno slanje REST, SOAP i GraphQL zahtjeva te prikaz dobivenih rezultata korisniku. U ovom radu Postman će biti korišten za testiranje i kreiranje API zahtjeva za vremensku prognozu pomoću REST stila. REST se temelji na HTTP-u, pa su dostupne sljedeće metode i tipovi poziva: GET, PUT, POST, HEAD, PATCH, DELETE i drugo. Kako će u radu biti potrebno jedino dohvaćati podatke s poslužitelja, bit će korištena jedino GET metoda. Primjer GET metode i rezultata vraćenog od strane server prikazan je na slici 3.6.



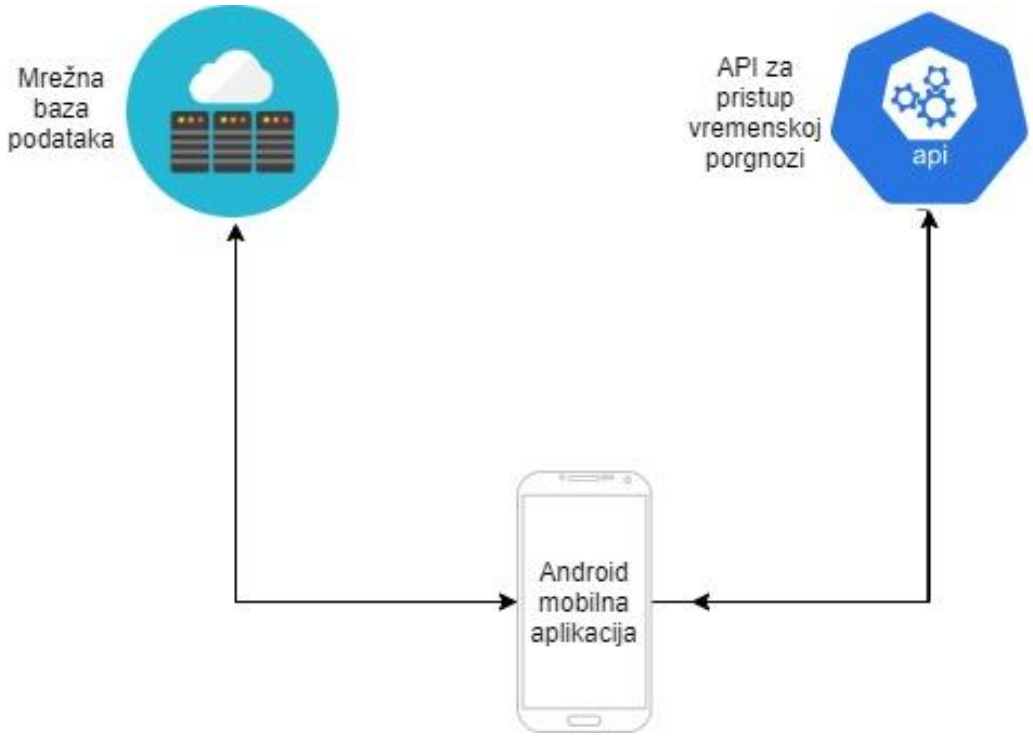
Slika 3.6. *Prikaz zahtjeva i rezultata u programskom alatu Postman*

Kako je vidljivo na slici 3.6., osim direktnog navođenja parametara unutar URL-a, moguće je dodati potrebne parametre i preko programskog sučelja. Moguće je dodati općenite parametre, autorizacijske podatke, zaglavlja, tijela dokumenta i slično. Rezultat s poslužitelja automatski prepoznaje te odabire valjani format: JSON, XML, HTML ili tekst te omogućuje pravilno strukturalno formatiranje rezultata. Osim toga, prikazuje i bitne podatke o statusu zahtjeva, vrijeme koje je potrebno za izvršavanje, obradu zahtjeva i prikaz rezultata te veličinu primljenog dokumenta.

4. ARHITEKTURA SUSTAVA PROGRAMSKOG RJEŠENJA

U svrhu ovog rada razvijena je Android mobilna aplikacija koja poljoprivrednicima omogućuje učinkovitije upravljanje biljnim kulturama u raznim poljoprivrednim granama: ratarstvo, voćarstvo, vinogradarstvo ili povrtlarstvo. Također, sustav omogućuje spremanje i dohvaćanje svih informacija o pojedinoj kulturi na temelju geografske lokacije, prikaz zemljišta na karti s odabranim granicama i pripadnim točkama. Sustav omogućuje uvid u vremensku prognozu za pojedino zemljište, kako buduću tako i prošlu. Nadalje, sustav implementira upravljanje korisnicima i njegovim dokumentima, kalendarom i događajima. Izrađen je i algoritam za obavještavanje korisnika o mogućim neprilikama i bolestima za pojedinu kulturu kako bi korisnik na vrijeme mogao reagirati.

Sustav izrađen u ovom radu sastoji se od nekoliko dijelova: Android mobilne aplikacije, mrežnog servisa za spremanje podataka i aplikacijskog programskog sučelja (engl. *API - Application Programming Interface*) za dohvaćanje podataka s mrežnog servisa. Implementiranjem mrežne baze podataka sustav postaje fleksibilniji te je lakša prenosivost i dijeljenje podataka. Arhitektura sustava kreirana u ovom radu prikazana je na slici 4.1.



Slika 4.1. *Arhitektura sustava programskog rješenja*

Slika koja prikazuje arhitekturu sustava programskog rješenja kreirana je pomoću mrežnog alata Draw.io. Iz slike 4.1. vidljivo je kako svaki element sustava ima poveznicu s Android

mobilnom aplikacijom u oba smjera, što znači kako aplikacija i šalje i prima podatke od svakog pojedinog elementa unutar sustava. Detaljan opis svakog pojedinog elementa sustava naveden je u sljedećim potpoglavljima.

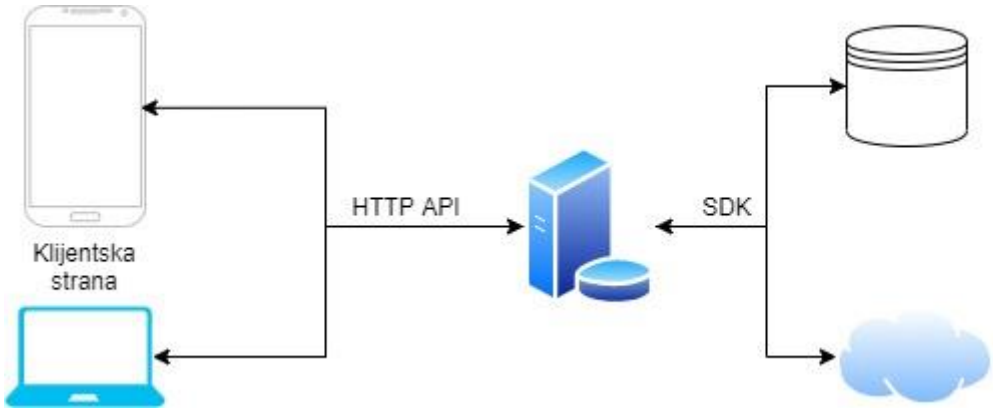
4.1. Mrežna baza podataka

U svrhu prenosivosti i dostupnosti podataka na različitim uređajima korištena je mrežna platforma naziva Firebase. Prema [21, 22], Firebase je Googleova platforma za razvoj mobilnih aplikacija koja pomaže pri rastu, razvoju i poboljšanju same aplikacije. Firebase zadrži set raznih alata i servisa koji su smješteni u oblaku računala, a omogućuju programeru implementaciju njih samih pomoću programskih razvojnih alata (engl. *SDK - Software Development Kit*). To je tipičan primjer BaaS (engl. *Backend as a Service*) servisa u oblaku računala.

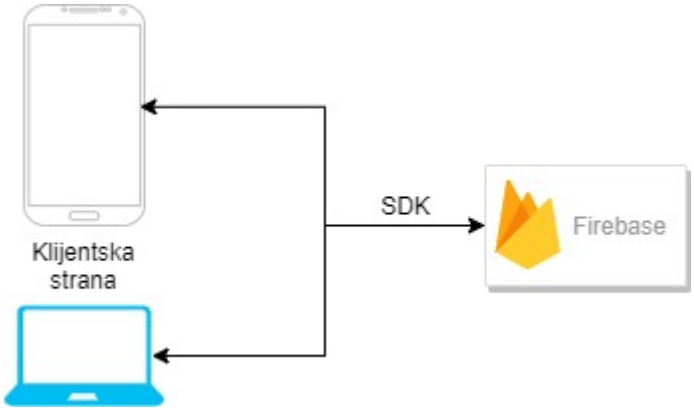
Kako je navedeno u [23], BaaS je servis koji razvojnom inženjeru omogućuje lakše upravljanje centraliziranom bazom podataka te omogućuje korisnicima dijeljenje podataka i sadržaja spremljenog u bazi podataka preko oblaka računala. Također, korištenjem BaaS servisa, razvojni inženjer lakše, brže i jednostavnije kreira i implementira backend mobilne aplikacije, korištenjem API-ja i raznih programskih alata. Za razliku od softvera kao servisa (engl. *SaaS - Software as a Service*) koji je namijenjen krajnjem korisniku, BaaS je namijenjen razvojnim inženjerima.

Korištenjem BaaS servisa, razvojni inženjeri mogu uštedjeti dosta vremena i novaca zbog nepotrebnog razvijanja dijelova sustava ili alata koji već negdje postoje. Naravno, ako je to moguće, korištenjem BaaS sustava izbjeglo bi se pisanje frontend programskog koda, koji općenito poziva API krajnje točke bilo za dohvat ili slanje podataka prema backendu koji onda odrađuje taj posao. Naravno, potrebno je implementirati i backend stranu sustava. Kakogod, korištenjem Firebase platforme, pisanje backend programskog koda je izbjegnuto. Na slici 4.2. prikazan je tradicionalni pristup razvoja mrežnog sustava, dok je na slici 4.3. prikazana prednost korištenja Firebase platforme.

Na slici 4.3., vidljivo je kako nedostaje i nije implementiran cijeli jedan dio sustava koji se bavi backendom te je to jedan od glavnih razloga korištenja Firebase platforme u izradi ovog rada. Korištenjem Firebase platforme, razvojnom inženjeru se nude mnogi alati pa je potrebno odabrati željene, no potrebno je obratiti pažnju i na cijenu jer se većina usluga plaća prekoračenjem određenog broja besplatnih jedinica usluga.



Slika 4.2. *Tradicionalni način razvoja mrežnog sustava*

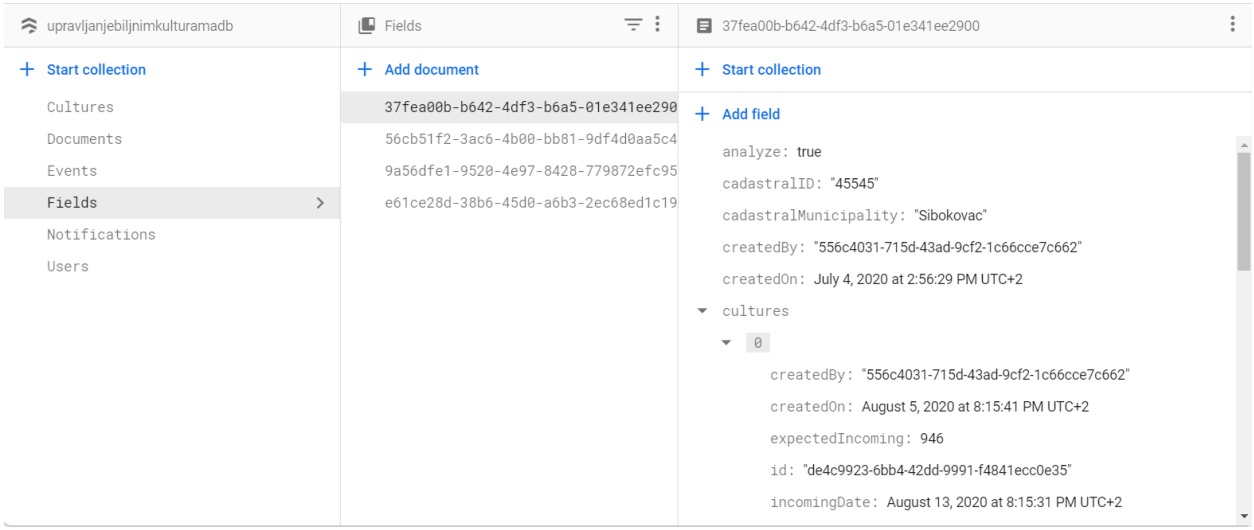


Slika 4.3. *Razvoj mrežnog sustava korištenjem Firebase platforme*

Upravljanje projektima, servisima i alatima vrši se preko alata Firebase Console. Potrebno je registrirati se preko Google računa, te kreirati korisnički profil. Također, korištenjem Firebase Console razvojni inženjer može vidjeti statistiku posjećenosti i uporabu podataka u odabranom prošlom razdoblju te još puno toga. Temeljeno na tim činjenicama, u ovom radu bit će korištena dva alata za spremanje podataka koja nudi Firebase, a to su Cloud Firestore i Cloud Storage, no više o tome u nastavku.

4.1.1. NoSQL baza podataka Cloud Firestore

Cloud Firestore je NoSQL baza podataka koja omogućuje relativno jednostavno spremanje, sinkroniziranje i dohvaćanje podataka za mobilne i web aplikacije na globalnoj razini, kako je rečeno u [24]. Za razliku od SQL baza podataka koje su relacijske, sadrže tablice i entitete, primarne i strane ključeve, NoSQL baza podataka sprema zapise u obliku dokumenta koji sadrži određene podatke. Takav način zapisa omogućuje strukturiranje povezanih podataka po želji u kolekcije i dokumente. Primjer zapisa u Cloud Firestore NoSQL bazi podataka dan je slikom 4.4.



Slika 4.4. *Primjer zapisa zemljišta u NoSQL bazi podataka*

Prema slici 4.4., vidljivo je da je baza podataka podijeljena u 6 kolekcija: *Cultures*, *Documents*, *Events*, *Fields*, *Notifications* i *Users*. Kolekcija, kao na primjer *Fields*, sadrži popis dokumenata zemljišta koji su spremljeni u bazi. Korištenjem načina pristupanja Firebase alatima pomoću Firebase SDK-a, prikazanog na slici 4.3., moguće je dodati novi zapis u kolekciji *Fields* te dohvatiti spremljeni. Programskim kodom 4.1. dana je struktura klase *Field*.

```
class Field(
    var id: String,
    val name: String,
    val surface: Double,
    val cadastralID: String,
    val cadastralMunicipality: String,
    var listOfCoordinates: MutableList<LatLng> = mutableListOf(),
    var cultures: MutableList<Culture> = mutableListOf(),
    val createdOn: Date,
    val createdBy: String,
    val analyze: Boolean
)
```

Programski kod 4.1. *Klasa Field*

4.1.2. Servis za pohranu dokumenata Cloud Storage

Cloud Storage je servis koji omogućuje spremanje dokumenata, fotografija, videozapisa i drugih korisničkih resursa, navedeno je u [25]. U ovom radu, Cloud Storage je korišten za spremanje korisničkih fotografija koje su direktno povezane sa zemljopisnom lokacijom i nekim događajem. Također, nakon spremanja određenog resursa u Cloud Storage, dostupan je link za preuzimanje dokumenta, koji se zatim sprema u Cloud Firestore NoSQL bazu podataka kako bi kasnije bilo moguće pristupiti spremljenom dokumentu.

4.2. API za pristup vremenskoj prognozi

Među bitnijim zahtjevima na aplikaciju i sustav bilo je implementirati mogućnost uvida u vremensku prognozu. Tako je u aplikaciji implementirana mogućnost uvida trenutne i dnevne prognoze te prognoze za idućih 7 dana. Ova funkcionalnost je ostvarena korištenjem DarkSky aplikacijskog programskog sučelja, koji prema [26] omogućuje pregled vremenske prognoze i relevantnih podataka za bilo koju lokaciju na zemlji. Općenito, API je podijeljen na dva dijela: onaj koji daje podatke za vremensku prognozu u budućnosti i onaj koji vraća podatke o vremenu iz nekog prošlog trenutka. Detaljnije, API vraća sljedeće tipove prognoza:

- Trenutni vremenski uvjeti i podatci
- Prognoza za svaku minutu u sljedećem satu
- Sat po sat i dan po dan vremenska prognoza za idućih 7 dana
- Sat po sat i dan po dan vremenska prognoza za trenutak u prošlosti desetljećima unatrag
- Upozorenja za nevremena za određene lokacije na zemlji: SAD, Kanada, Europska Unija i Izrael

API vraća podatke u JSON formatu za obje vrste vremenske prognoze (buduću i prošlu). Također, potrebno je naglasiti kako je svaki dan dozvoljeno 1000 besplatnih API poziva te se svaki idući naplaćuje $0.0001. Programskim kodom 4.2. dan je uvid u vrstu podataka koja je primljena od strane API-ja za trenutnu vremensku prognozu.

```
data class WeatherCurrently (
    val time: Long,
    val summary: String,
    val icon: String,
    val precipIntensity: Double,
    val precipProbability: Double,
    val temperature: Double,
    val apparentTemperature: Double,
    val dewPoint: Double,
    val humidity: Double,
    val pressure: Double,
    val windSpeed: Double,
    val windGust: Double,
    val windBearing: Long,
    val cloudCover: Double,
    val uvIndex: Long,
    val visibility: Double,
    val ozone: Double,
    val precipType: String? = null
)
```

Programski kod 4.2. *Klasa koja sadrži podatke trenutne vremenske prognoze*

Kako je vidljivo iz programskog koda 4.2., dostupni su gotovo svi podatci koji se mogu trenutno izmjeriti za vremensku prognozu. Pa tako postoje: vrijeme, opis, ikona, intenzitet padalina, vjerojatnost padalina, temperatura, osjećaj temperature, temperatura kondenzacije, vlažnost zraka, tlak zraka, brzina vjetra, udari vjetra, smjer vjetra, pokrivenost oblacima, UV indeks, vidljivost, ozon i tip padalina. Gore navedenu podatci spremljeni su u klasu naziva *WeatherCurrently* koja se nalazi unutar klase *WeatherBaseResponse*, čija je implementacija dana programskim kodom 4.3.

```
data class WeatherBaseResponse (
    val currently: WeatherCurrently,
    val hourly: WeatherHour,
    val daily: WeatherWeak
)
```

Programski kod 4.3. *Klasa WeatherBaseResponse*

Osim objekta WeatherCurrently, moguće je implementirati i dohvatiti objekte tipa *WeatherHour*, koji sprema podatke o vremenu po satima i objekt tipa *WeatherWeak* koji sprema podatke po tjednima. U klasu *WeatherBaseResponse* sprema se rezultat koji je dohvaćen preko API-ja, čiju implementaciju je potrebno kreirati pomoću biblioteke Retrofit.

4.2.1. Retrofit biblioteka

Retrofit je REST klijent za Java programski jezik i Android. Relativno je jednostavno ga koristiti za slanje i primanje podataka u raznim formatima (JSON, XML, …) prema web servisima koji su temeljeni na REST-u, navedeno je u [27]. Kako je navedeno u potpoglavlju 4.2., DarkSky API vraća podatke u JSON formatu pa je potrebno koristiti neki od JSON pretvarača koji zapisuju JSON zapis unutar klasa aplikacije. U tu svrhu bit će korišten Gson biblioteka. Za HTTP zahtjeve Retrofit koristi OkHttp biblioteku. Implementacija Retrofit klase dana je programskim kodom 4.4.

```
object WeatherNetworking {
    val weatherService: WeatherApi = Retrofit.Builder()
        .addConverterFactory(ConverterFactory.converterFactory)
        .client(HttpClient.client)
        .baseUrl(WEATHER_API_BASE_URL)
        .build()
        .create(WeatherApi::class.java)
}

object ConverterFactory {
    val converterFactory = GsonConverterFactory.create()
}

object HttpClient {
    val client = OkHttpClient.Builder()
```

```
.addInterceptor(HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY)
)
        .build()
}
```

Programski kod 4.4. *Retrofit klasa*

Osim klase koja sadrži instancu Retrofit biblioteke, potrebno je implementirati još minimalno dvije klase. Jedna od njih je model, u koji se spremaju podatci dohvaćeni preko Retrofit biblioteke, koji je prikazan programskim kodom 4.3. Kako bi podatci bili dohvaćeni potrebno je implementirati i metode koje dohvaćaju podatke, a implementiraju se unutar sučelja koje je predano kao klasa u kreatoru (engl. *Builder*) instance Retrofita naziva *WeatherAPI*. Primjer je dan programskim kodom 4.5.

```
interface WeatherApi {

    @GET("{latitude},{longitude}")
    suspend fun getForecastRequest(
        @Path("latitude") latitude: Double,
        @Path("longitude") longitude: Double,
        @Query("units") units: String,
        @Query("exclude") excludes: String
    ): WeatherBaseResponse

    @GET("{latitude},{longitude},{timestamp}")
    suspend fun getTimeMachineRequest(
        @Path("timestamp") timestamp: Long,
        @Path("latitude") latitude: Double,
        @Path("longitude") longitude: Double,
        @Query("units") units: String,
        @Query("exclude") excludes: String
    ): WeatherBaseResponse
}
```

Programski kod 4.5. *Sučelje WeatherAPI*

Sučelje *WeatherAPI* implementira metodu naziva *getForecastRequest* koja prima 4 parametra i metodu *getTimeMachineRequest* koja prima 5 parametra. Anotacijom *@GET* označava se kako se radi GET HTTP metoda koja će u URL dodati 2 parametra: *latitude* i *longitude* za metodu *getForecastRequest* te za metodu *getTimeMachineRequest* još jedan parametar naziva *timestamp*, što mora biti označeno anotacijom *@Path* odmah uz navedene parametre. Anotacijom *@Query* označava se kako će se u URL dodati 2 upita: *units* i *exclude*, koji označavaju mjerne jedinice u kojima će vraćeni podatci biti napisani i vrijednosti koje se ne dohvaćaju. Samim time smanjujemo utrošeni internetski promet te povećavamo brzinu primanja odgovora.

4.3. Arhitektura mobilne aplikacije

Razvojem aplikacije i sustava broj datoteka projekta se povećava te se mnogo vremena troši na pretragu datoteka, izmjenu i slično. Također, ako su datoteke velike i složene, sama izmjena programskog koda zahtjeva dodatan trud, a ponekad i mukotrpne izmjene i implementacije. U tu svrhu koriste se razni obrasci koje razvojni inženjer treba poštovati kako bi u konačnici projekt bio organiziraniji. Osim toga, implementacijom određenih obrazaca programski kod će uvijek biti oblikovan određenim pravilima, izgledom i funkcionalnostima što predstavlja veliku prednost prilikom razvoja sustava u kojem sudjeluje više inženjera. Oni tada s lakoćom mogu prepoznati nekakav oblik programskog koda i pretpostaviti njegovu funkcionalnost.

Osim korištenja obrazaca pri pisanju programskog koda, za bolje organiziranje projekta dokumenti se strukturiraju unutar određenih direktorija i paketa prema obrascima arhitekture. Trenutno se najčešće koriste 3 sljedeća obrasca: MVP (engl. *Model-View-Presenter*), MVC (engl. *Model-View-Controller*) i MVVM (engl. *Model-View-ViewModel*). Prilikom razvijanja mobilne aplikacije za ovaj rad, korišten je MVVM obrazac. Kako je navedeno u [28], arhitektura MVVM projekta sastoji se od 3 ključne komponente:

- modela (engl. *Model*),
- prikaza (engl. *View*) i
- modela prikaza (engl. *ViewModel*)

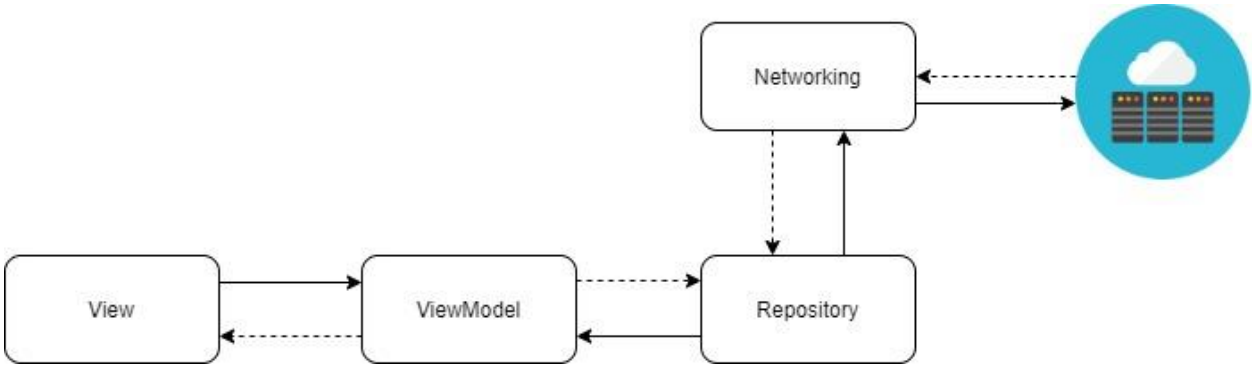
Komunikacija između navedenih komponenti prikazana je slikom 4.5. Model sadrži sve podatkovne klase, klase baze podataka, implementacije API-ja i sučelja, repozitorij i ostale slične implementacije. View predstavlja korisničko sučelje i sve one klase i stanja koja su vidljiva korisniku. ViewModel dohvaća podatke koji su potrebni za prikazivanje na zaslonu, tj. koji su potrebni View-u, iz model klasa te ih prosljeđuje na View kako bi se podatci prikazali na zaslonu.

Slika 4.5. *MVVM arhitektura sustava*

Kako je vidljivo na slici 4.5., Model obostrano komunicira s ViewModel podsustavom, dok View jedino pristupa ViewModel podsustavu. Točnije, View zatraži nekakav podatak za prikaz ili se pretplati na određeni podatak i njegovu promjenu te se promjena automatski prikazuje na

zaslonu. To je omogućeno korištenjem LiveData funkcionalnosti koja je korištena u izradi ovog rada. Osim korištenja LiveData funkcionalnosti, moguće je povezati View i ViewModel i preko DataBinding načina povezivanja.

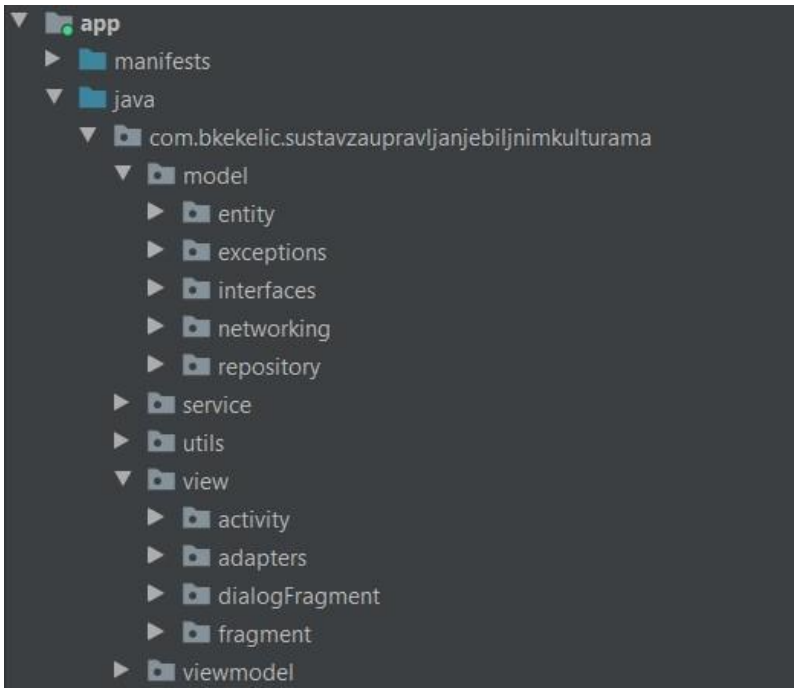
Implementiranjem opisanog MVVM dizajna može se uvidjeti kako Model dio sadrži većinu programskog koda sustava. To se može spriječiti dodavanjem repozitorij (engl. *Repository*) klasa koje se brinu o komunikaciji s mrežom (engl. *Networking*) i eventualno klasa za rad s bazom podataka. Također, ako projekt sadrži puno zaslona i elemenata zaslona, poželjno ih je razdvojiti po njihovim funkcionalnostima. Prikaz detaljnijeg MVVM dizajna prikazano je slikom 4.6.



Slika 4.6. *Proširena MVVM arhitektura sustava*

Kako je prikazano na slici 4.6., Model je zamijenjen s repozitorijem i mrežnim dijelom. U ovakvoj raspodijeli sustava repozitorij je odgovoran za dohvaćanje podataka koji će se zatim proslijediti do ViewModel dijela, a zatim i do View dijela. Repozitorij je prikladno koristiti kada se podatci mogu dohvatiti iz više izvora, npr. podatci za vremensku prognozu se dohvaćaju svakih 5 minuta, a ako nije prošlo 5 minuta potrebno je prikazati prethodno spremljene podatke. Pa tako u navedenom slučaju repozitorij prvo provjerava je li prošlo 5 minuta od prošlog dohvaćanja podataka, ako je, dohvati podatke preko Networking modula, koji jedini zna i brine se o implementaciji dohvaćanja podataka s mreže. Nakon što dohvati podatke spremi ih u bazu i vrati na prikaz. Ako se dogodi pogreška ili nije prošlo 5 minuta, repozitorij odlučuje dohvatiti podatke iz lokalne baze podataka preko objekata za pristup bazi podataka koji znaju kako se pristupa bazi i imaju implementirane potrebne metode.

U izradi ovog rada projekt je organiziran i strukturiran kako bi se lakše i jednostavnije proširio i nadogradio. Temeljem toga, na slici 4.7. dane je prikaz organizacije projektne strukture. Ovako organiziranim projektom razvojni inženjer kasnije može lakše ubaciti i implementirati lokalnu bazu podataka bez velikih izmjena i konflikata, kako je navedeno u prošlom odlomku.



Slika 4.7. *Raspodjela paketa i direktorija Android projekta*

Kako je vidljivo na slici 4.7., projekt je organiziran korištenjem MVVM obrasca. Postoje 3 glavna direktorija (*model*, *view* i *viewmodel*) te *service* direktorij u kojemu se nalaze klase servisa te jedan u kojemu su smještene klase koje se koriste u raznim dijelovima aplikacije i nije ih moguće točno smjestiti - *util*. Prošireni prikaz paketa *model* sadrži sljedeće pakete:

- *entity* – u kojem su spremljene podatkovne klase
- exceptions – podatkovne klase za iznimke
- *interfaces* – sučelja
- *networking* – mrežni dio aplikacije
- *repository* - repozitorij

Nadalje, vidljivo je kako je *view* direktorij podijeljen u nekoliko paketa: zasloni (engl. *Activity*), fragmenti (engl. *Fragments*), fragmenti u skočnim prozorima (engl. *dialogFragment*) i adapteri (engl. *Adapters*). Korištenjem MVVM-a razdvojena je poslovna logika od prezentacijskog dijela, što znači da prezentacijski dio ne treba znati što treba prikazati već se sva logika za pripremu podataka za prikaz vrši unutar ViewModel modula, što ga čini ponovno upotrebljivim. Korištenjem repozitorija, ViewModel jedino zna kako napraviti upit za podatke, dok se sva logika obrađuje unutar repozitorija. Još jedna prednost ViewModel modula je i njegova svjesnost životnog ciklusa (engl. *Lifecycle Awareness*) jer se proteže kroz sve životne cikluse zaslona, što znači da su podatci u njemu postojani i očuvani prilikom rotiranja zaslona i slično.

5. ANDROID MOBILNA APLIKACIJA

U ovom radu razvijena je mobilna aplikacija za Android platformu koja je trenutno najzastupljenija platforma za operacijski sustav mobilnih uređaja na tržištu. Aplikacija omogućuje poljoprivrednicima učinkovitije upravljanje biljnim kulturama (voćarstvo, ratarstvo, povrtlarstvo, vinogradarstvo…) te omogućuje spremanje i dohvaćanje svih informacija o pojedinoj kulturi na temelju geografske lokacije. Osim toga, aplikacija omogućuje korisnicima prikaz zemljišta na karti, upravljanje dokumentima, kalendarom i događajima. Također, sustav je izgrađen tako da omogućuje upravljanje korisnicima, njihovu registraciju, prijavu i odjavu iz sustava, te korištenje aplikacije na različitim mobilnim uređajima.

Aplikacija također omogućuje obavještavanje korisnika na temelju praćenja vremenskih parametara i raznih karakteristika biljnih kultura te mogućih razvoja bolesti. Korisnik bi bio pravovremeno obaviješten o vremenskoj neprilici, o bolesti koja može poharati njegov usjev i plod te tako uništiti sav njegov trud ili ga pak obavijestiti kako njegova biljka preživljava nepovoljne vremenske uvijete koji su ju snašli. Korisnik bi mogao pravovremeno reagirati, zaštititi svoj usjev, a kao rezultat povećala bi se kvaliteta a i samim time i kvantiteta krajnjih proizvoda. Funkcionalnosti aplikacije bit će opisane u sljedećim potpoglavljima.

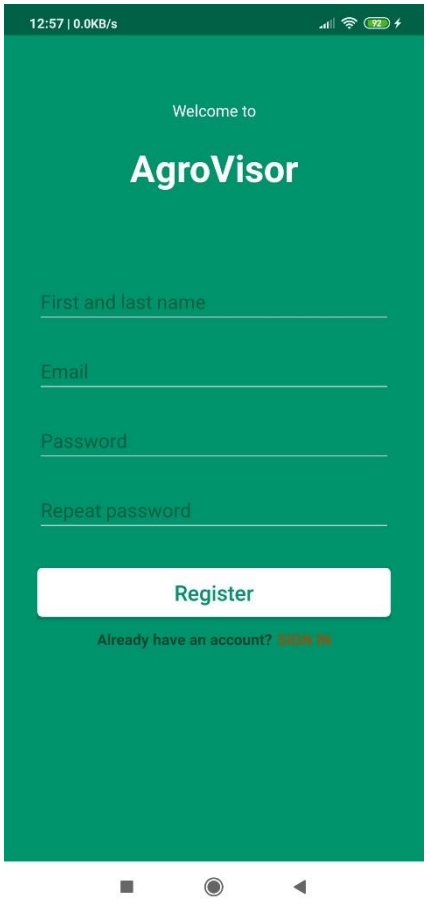
5.1. Upravljanje korisnicima

Pri pokretanju aplikacije, aplikacija provjerava postoji li trenutno prijavljen korisnik. Ako je korisnik prijavljen prikazuje se početni zaslon, a ako trenutno nema prijavljenih korisnika u sustavu, prikazuje se zaslon za prijavu. Prije nego što korisnik krene koristiti aplikaciju, potrebno je registrirati se u sustav. Zaslon za registraciju korisnika u sustav prikazan je na slici 5.1.

Korisnik se registrira u sustav upisivanjem svog imena i prezimena te emaila i lozinke koju je potrebno ponoviti. Također, potrebno je naglasiti kako postoje određena ograničenja prilikom registracije. Pa je tako naravno, potrebno popuniti sva polja, lozinka i ponovljena lozinka trebaju biti identične, s više od 6 znakova. Email adresa korisnika mora biti jedinstvena u sustavu, što znači kako se u sustav ne mogu registrirati 2 korisnika s istim emailom.

Prilikom spremanja lozinke u bazu podataka ona se kriptira sa SHA-256 algoritmom kako lozinka ne bi bila vidljiva administratoru sustava ili nekoj trećoj osobi u slučaju nezakonitog i nevaljanog pristupa bazi podataka. Kako je navedeno u [29], rezultat je 128 bitna *hash* vrijednost. Java implementira *MessageDigest* klasu koja aplikacijama nudi korištenje algoritama za kriptiranje poruka (engl. *Message Digest Algorithm*) koje kao ulaznu vrijednost primaju

proizvoljnu duljinu teksta a uvijek vraćaju konstantnu duljinu od 128 bitova. Implementacija kriptiranja dana je programskim kodom 5.1.

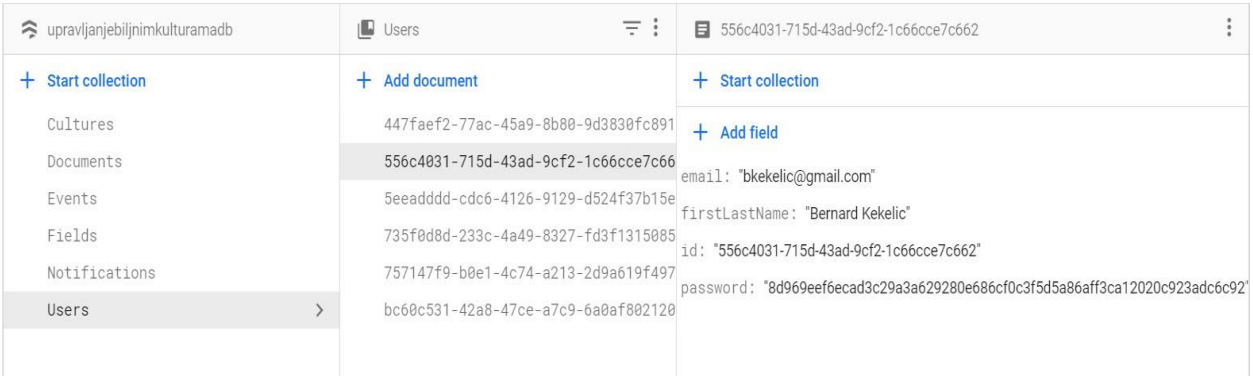


Slika 5.1. *Zaslon za registraciju korisnika u sustav*

```
fun String.SHA256(): String {
    val md = MessageDigest.getInstance("SHA-256")
    return BigInteger(1, md.digest(toByteArray())).toString(16).padStart(32, '0')
}
```

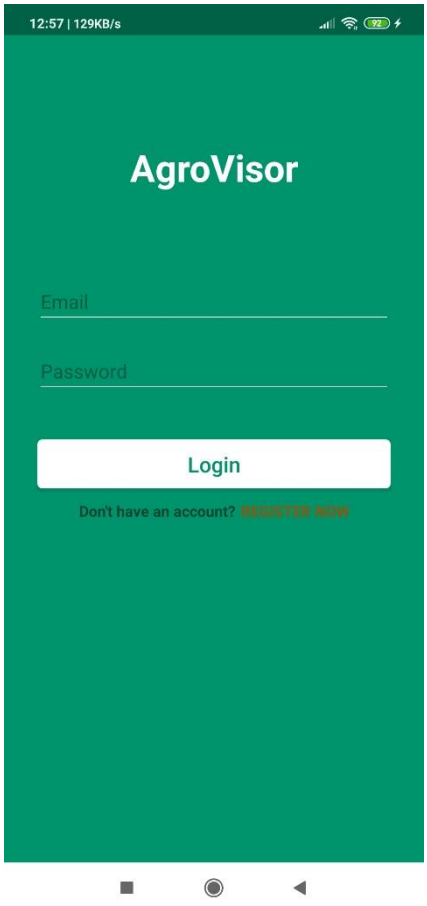
Programski kod 5.1. *Algoritam za kriptiranje lozinke*

Kako je vidljivo u programskom kodu 5.1., implementirana je funkcija proširenja (engl. *Extension Function*) nad klasom *String*. Ovakav način implementacije omogućuje pozivanje funkcije *SHA256* nad bilo kojim tipom podataka *String* te smanjuje redundantni kod i dodatne implementacije a veoma je jednostavan i učinkovit. Pritiskom na gumb *Register* korisnik se registrira u sustav, a podatci se zapisuju u mrežnu bazu podataka. Primjer spremljenih podataka dan je slikom 5.2.



Slika 5.2. Primjer korisničkih podataka spremljenih u mrežnu bazu podataka

Osim registracije novih korisnika, aplikacija implementira i mogućnosti prijave i odjave iz mobilne aplikacije. Ako korisnik trenutno nije prijavljen u sustav, prilikom pokretanja aplikacije bit će prikazan ekran za prijavu. Na slici 5.3. dan je prikaz zaslona za prijavu u sustav.



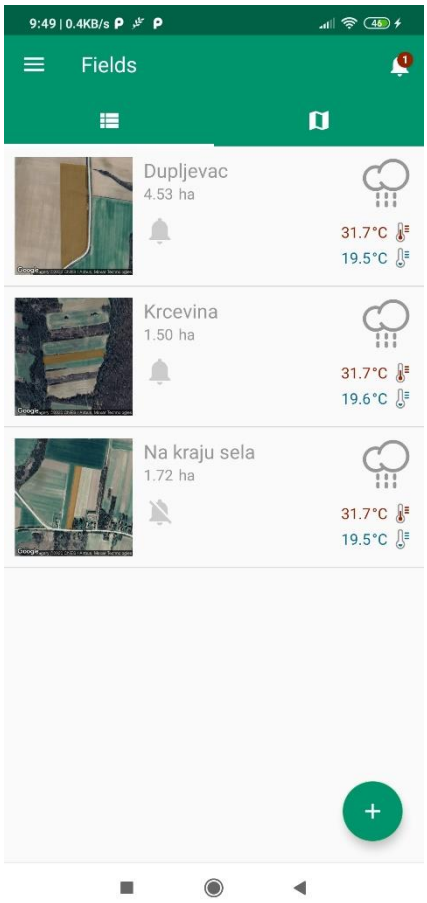
Slika 5.3. *Zaslon za prijavu u sustav*

Kako je vidljivo na slici 5.3., korisnik se u sustav prijavljuje upisivanjem valjane email adrese i lozinke. Pritiskom na gumb *Login* pokreće se provjera prijave. Prvo se provjerava jesu li svi

potrebni podatci uneseni te zatim postoji li u sustavu registrirani korisnik s tom email adresom. Ako je valjana, lozinka se kriptira na isti način kao i kod registracije te se uspoređuje sa spremljenim 128 bitnim zapisom u mrežnoj bazi podataka. Ako su oba zapisa identična, može se sa sigurnošću utvrditi kako su lozinke identične te je prijava u sustav završena te se prikazuje početni zaslon aplikacije. Ako su podatci neispravni o tome se valjano obavještava korisnik.

5.2. Upravljanje zemljištima

Sve poljoprivredne aktivnosti vezane su direktno za zemljišta pa je tako i u ovom radu fokus prvenstveno na povezivanje informacija sa zemljištima. Početni zaslon aplikacije prikazuje korisnikova zemljišta u dva načina prikaza na dva *fragmenta* u *view pageru*: prikaz popisa zemljišta u listi i prikaz zemljišta na karti. Osim toga, u početni zaslon aplikacije implementirana je i *drawer* funkcionalnost koja omogućuje izvlačenje dijela ekrana te prikaz dodatnih funkcionalnosti. Početni zaslon aplikacije prikazan je na slici 5.4.



Slika 5.4. *Početni zaslon aplikacije*

Na početnom zaslonu aplikacije prikazan je popis zemljišta. Iz slike 5.4. vidljivo je da trenutno korisnik ima spremljena 3 zemljišta naziva *Dupljevac, Krcevina* i *Na kraju sela*. Zemljišta se imenuju po domaćim imenima, dok se identificiraju po Katastarskom (ARKOD) identifikatoru. Osim imena zemljišta, prikazuju se još i površina, dnevna prognoza te zemljište na karti. Ikona zvona označava hoće li se zemljište i njegove trenutno aktivne kulture analizirati algoritmom za detekciju mogućih bolesti i nepovoljnih vremenskih uvjeta.

Prikaz zemljišta na karti omogućen je korištenjem statičnih mapa. Naime, svaki član liste treba prikazati zemljište na karti. Ako bi bile korištene Google mape, u slučaju velikog broja zemljišta lista bi se jako dugo učitavala a i sam koncept i svrha prikaza bila bo pogrešna. Tako se prema [30], za prikaz karte u listi koriste statične Google mape. Korištenjem Google Maps Static API-ja razvojni inženjer može kreirati mapu na temelju parametara koje šalje HTTP zahtjevom. Rezultat je URL koji sadrži sliku koju se može dohvatiti i prikazati na sučelju. Također, bitno je napomenuti kako ova usluga nije besplatna već se naplaćuje. Primjer programskog koda koji se koristi za dohvaćanje statičnih mapa dan je programskim kodom 5.2.

```
val url = StaticGoogleMapsUrlBuilder.Builder()
    .mapType("satellite")
    .size(300)
    .scale(2)
    .pathColor(polygonColor)
    .pathWeight(3)
    .pathFillColor(polygonColor)
    .coordinates(field.listOfCoordinates)
    .markerColor(markerColor)
    .buildUrl()

Picasso.get()
    .load(url)
    .fit()
    .into(itemView.home_list_field_map)
```

Programski kod 5.2. *Dohvaćanje statičnih mapa*

Danim programskim kodom 5.2. prikazano je dohvaćanje statične mape s određenim parametrima te na kraju prikaz slike u listi pomoći Picasso biblioteke. Ova biblioteka ima mogućnost dohvaćanja slika preko njihovih URL-ova, što je u ovom slučaju iznimno korisno. Prilikom dohvaćanja slike moguće je postaviti različite parametre, kao na primjer:

- *mapType* – vrsta mape (*roadmap, satellite, hybrid* i *terrain*)
- *size* – dimenzija slike (npr. 300 x 300)
- *scale* – skaliranje, predstavlja broj piksela (2 = dva puta više piksela na slici)

- *pathColor, pathWeight i pathFillColor* – parametri korišteni za crtanje poligona koji predstavlja zemljište i njegovu ispunu
- *coordinates* – lista koordinata koje označavaju granice zemljišta
- *markerColor* – ako korisnik odabere samo jednu točku kao granicu zemljišta onda se to zemljište prikazuje s markerom pa ovaj parametar označava njegovu boju

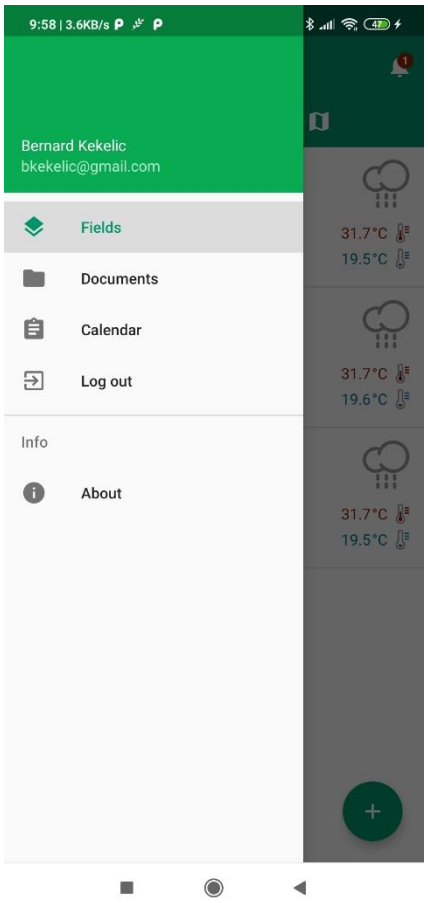
Osim prikaza zemljišta na karti, svako zemljište je prikazano s dnevnom meteorološkom prognozom. Naime, kako je vidljivo na slici 5.4., u desnom kutu maksimalna i minimalna dnevna temperatura za lokaciju zemljišta te odgovarajuća ikonica dnevnog vremena. Meteorološki podatci se, kako je rečeno u potpoglavlju 4.2., dohvaćaju preko API-a s DarkSky poslužitelja. Prikaz svih zemljišta na mapi prikazuje se povlačenjem ekrana u lijevo ili pritiskom na gornju desnu ikonicu u alatnoj traci. Na slici 5.5. prikazana su zemljišta na Google mapi.

Slika 5.5. *Prikaz svih zemljišta na Google mapi*

Zemljišta su na Google mapi prikazana i ispunjena smeđom bojom. Potrebno je bilo kreirati algoritam koji će zumirati i fokusirati ekran kako bi sva zemljišta bila prikazana a dovoljno vidljiva jer Google Maps API podržava zumiranje samo za jedan poligon. Algoritam je osmišljen tako da

odredi 4 točke koje će opisivati koordinate zemljišta te na temelju novokreiranog poligona zumira i postavi prikaz.

Na početni zaslon dodana je funkcionalnost prikaza nove obavijesti u gornjem desnom kutu koja prikazuje broj nepročitanih obavijesti generiranih od strane algoritma za detekciju bolesti i nepovoljnih vremenskih uvjeta, no više o tome u sljedećim potpoglavljima. Nadalje, na početni zaslon je implementiran i *drawer* prikaz koji omogućuje dodatne funkcionalnosti ali su skrivene na prvi pogled. Klikom na ikonicu u gornjem lijevom kutu ili povlačenjem s lijeve strane prema desnoj otvara se navedeni prikaz koji implementira funkcionalnosti navigacije po aplikaciji do početnog zaslona (*Fields*), dokumenata (*Documents*), kalendara (*Calendar*), o nama (*About*) i odjava (*Log out*). Detaljan prikaz dizajna ekrana dan je slikom 5.6.

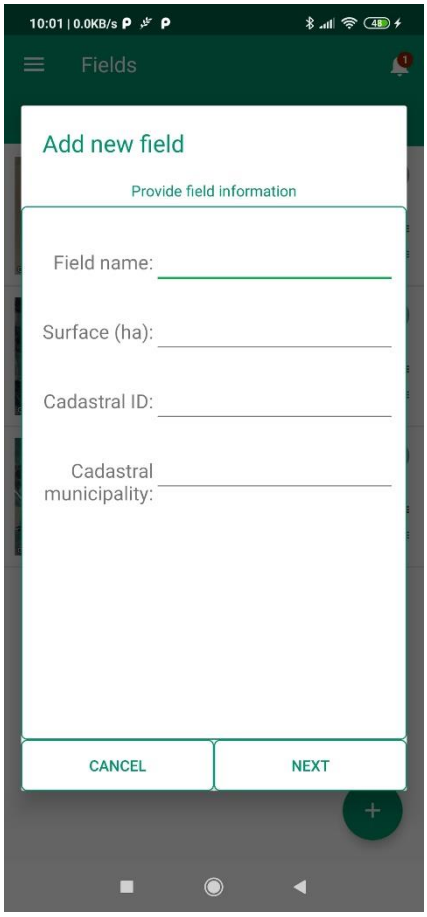


Slika 5.6. *Drawer prikaz na početnom zaslonu*

5.2.1. Dodavanje novog zemljišta

Korisnik ima mogućnost dodati novo zemljište pritiskom na gumb u donjem desnom kutu ekrana za prikaz zemljišta, bilo kao popis ili na mapi. Gumb se može vidjeti na slikama 5.4., 5.5. i 5.6. Pritiskom na gumb otvara se skočni prozor (engl. *Dialog*) s 2 *fragmenta*. Jedan za unos

podataka o zemljištu a drugi s prikazom Google mape na kojoj korisnik može označavati i iscrtavati granice novog zemljišta. Slikom 5.7. dan je prikaz zaslona za unos informacija o zemljištu.

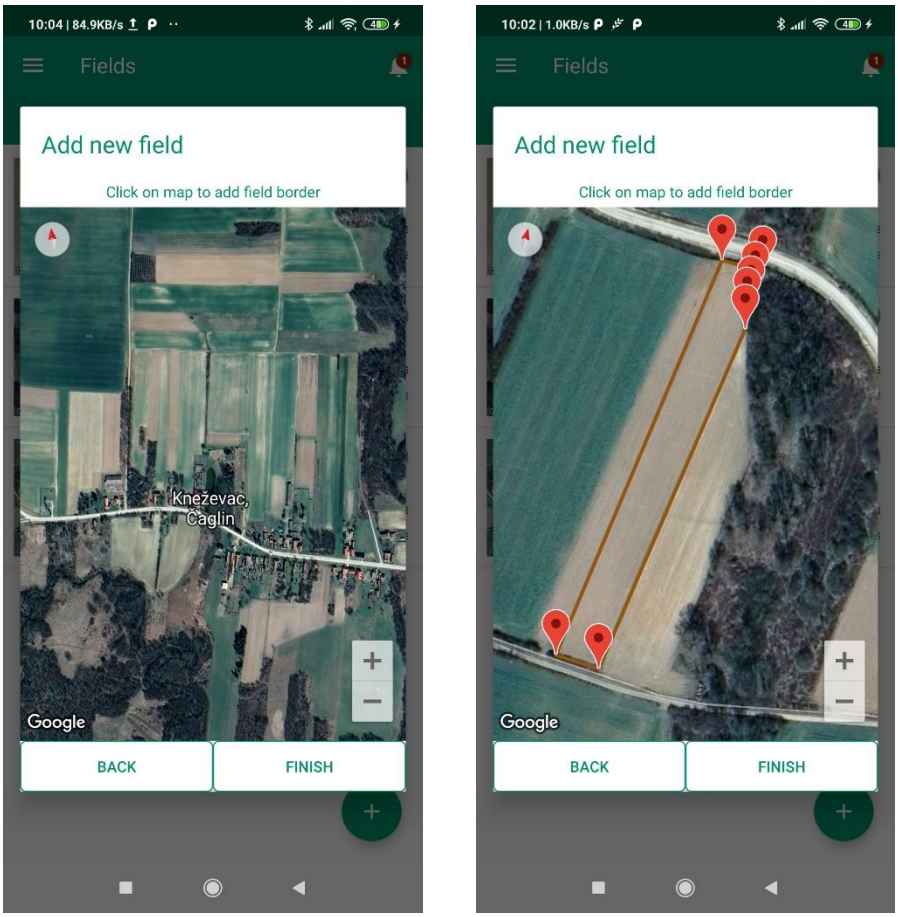


Slika 5.7. *Dodavanje novog zemljišta – unos informacija*

Pri kreiranju novog zemljišta, korisnik mora unijeti naziv zemljišta (engl. *Field name*), površinu u hektarima (engl. *Surface*), katastarski ID (engl. *Cadastral ID*) i katastarsku općinu (engl. *Cadastral municipality*). Ako su svi podatci uneseni, korisnikom klikom na gumb *Next* otvara sljedeći prikaz koji omogućuje crtanje granica zemljišta. Zaslon za crtanje zemljišta dan je slikama 5.8. i 5.9.

Prilikom otvaranja mape, aplikacija automatski zumira do korisnikove posljednje poznate približne lokacije. Ova funkcionalnost omogućena je korištenjem *fused location* servisa. Aplikacija ne treba znati točnu lokaciju korisnika već približnu, kako bi mu predložila zemljišta u njegovoj blizini, a i smanjila nepotrebno trošenje baterije i mrežnih podataka. Korisnik dodaje točku koja označava granicu zemljišta klikom na određeni dio ekrana te se automatski kreira poligon od trenutno odabranih točaka. Dodavanjem više točaka korisnik može dodati točne granice svoga zemljišta, pogotovo ako zemljišta nisu pravokutnog ili sličnog oblika, kao što je vidljivo u

gornjem desnom kutu na slici 5.9. Klikom na postojeću točku na karti korisnik ju briše te se nove granice automatski crtaju. Spremanje zemljišta vrši se klikom na gumb *Finish,* dok se povratak na prethodni zaslon vrši pritiskom na gumb *Back.*



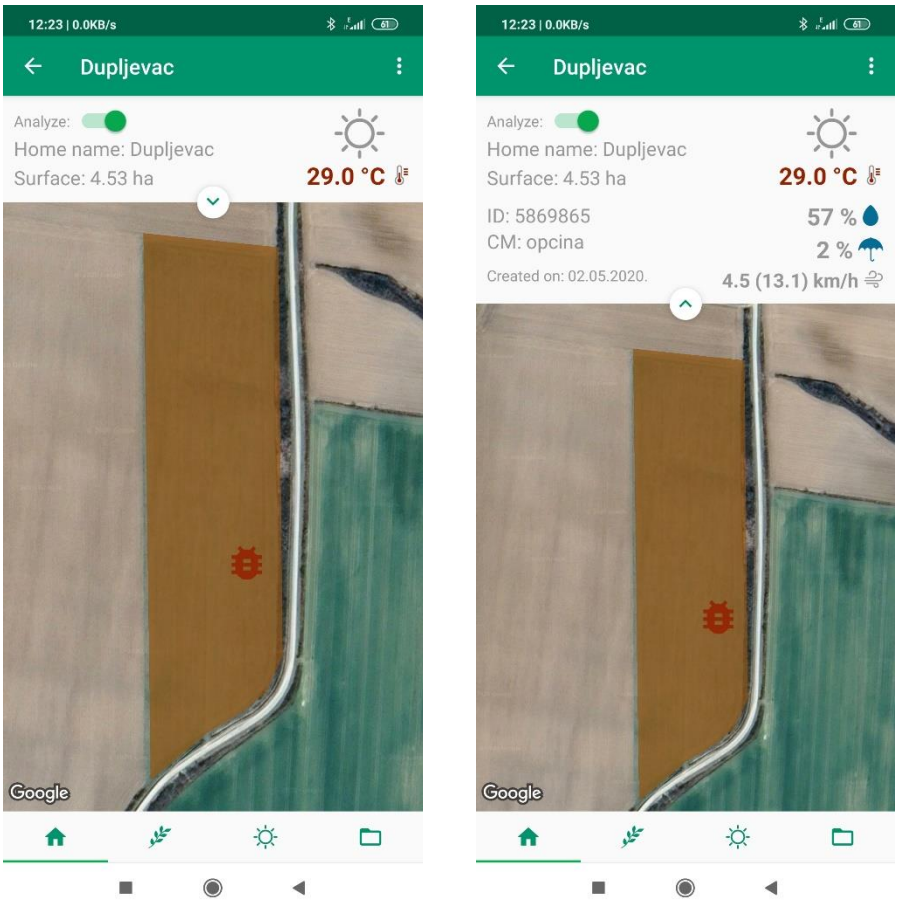
Slike 5.8. i 5.9. Dodavanje novog zemljišta - crtanje granica

5.2.2. Detalji o zemljištu

Korisnik može uvidjeti detalje o svakom zemljištu odabirom odgovarajućeg zemljišta iz liste zemljišta na početnom zaslonu ili klikom na iscrtano zemljište na Google karti. Odabirom zemljišta, otvara se novi zaslon (engl. *Activity*) koji sadrži *view pager* s nekoliko *fragmenata*. Početni zaslon detalja o zemljištu dan je slikama 5.10. i 5.11.

Na početnom zaslonu detalja o zemljištu, prema slici 5.10., prikazani su općeniti podatci o zemljištu: površina i naziv, prikaz zemljišta na karti smeđom bojom, te trenutna temperatura i sličica koja prikazuje trenutno meteorološko stanje. Također, korisnik ima mogućnost upaliti ili ugasiti buduće analize kultura odabranog zemljišta. Klikom na padajuću ikonu na sredini ekrana iznad Google mape prikazuju se dodatne informacije o zemljištu i trenutnom vremenu, prikazano na slici 5.11. Za zemljište: katastarski ID, općina i datum kreiranja zemljišta, dok za trenutnu

vremensku prognozu: vlažnost zraka, vjerojatnost oborina te brzina i udari vjetra. Ovakav prikaz omogućuje korisniku brz i jednostavan uvid u trenutno stanje na zemljištu bilo ono od njega udaljeno nekoliko stotina metara ili kilometara. Klikom na meni u gornjem desnom kutu korisnik ima mogućnost obrisati zemljište.



Slike 5.10. i 5.11. *Detalji o zemljištu*

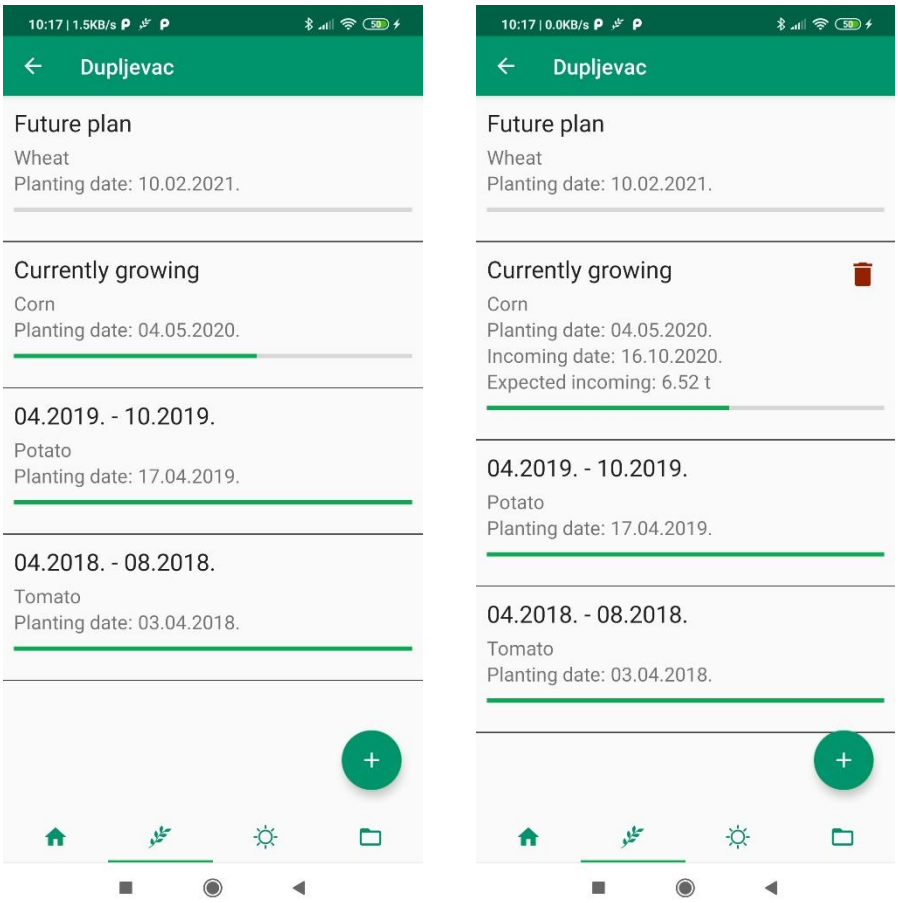
5.3. Upravljanje kulturama

Aplikacija kreirana u ovom radu zamišljena je tako da je baza svih događanja zemljište. Odabirom odgovarajućeg zemljišta korisnik vidjeti popis kultura na tom zemljištu, kako prošlih tako i budućih. Primjer liste koja predstavlja popis kultura dan je slikama 5.12. i 5.13.

Aplikacija automatski sortira i prikazuje naslove određene kulture ovisno o razdoblju kada je kultura bila aktivna. Moguća su 3 status kultura koje se automatski izračunavaju ovisno o datumu:

- *Future plan* – ovakav tip kulture ili aktivnosti još nije započeo, on je u planu za sljedeću sezonu ili buduće vrijeme
- *Currently growing* – tip kulture ili aktivnosti koji je trenutno aktivan u ovoj sezoni

- Kultura ili aktivnost koja je završila. Prikazuje se mjesec i godina početka i kraja aktivnosti

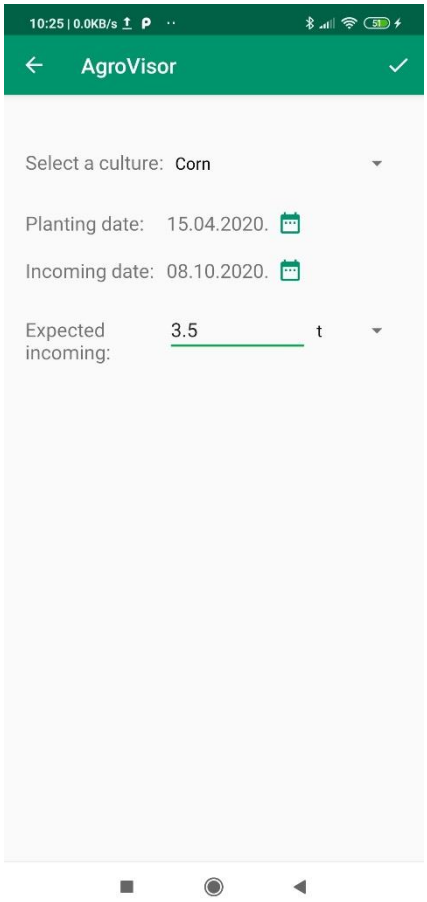


Slike 5.12 i 5.13. *Popis kultura na zemljištu*

Kako je vidljivo na slici 5.12. svaka kultura osim naslova sadrži još i tip kulture, te datum sadnje ili početka cvjetanja kod određenih kultura i indikator napretka. Indikator napretka označen je zelenom bojom te predstavlja postotak provedbe te aktivnosti u ovisnosti o datumu. Pa tako, one aktivnosti koje još nisu započele nemaju nikakav napredak, a dok one završene imaju popunjen napredak do 100 %, a one koje trenutno traju ovise o trenutnom datumu. Odabirom pojedine kulture, kako je prikazano na slici 5.13., animiranim prikazom na ekran se prikazuju i dodatne informacije o kulturi kao na primjer: očekivani datum roda i očekivani prinos. Također, vidljivo je kako je napredak na trenutnoj kulturi koja je i odabrana otprilike 60 % što i odgovara odnosu između početnog, trenutnog i krajnjeg datuma te aktivnosti. Osim toga, korisnik ima mogućnost brisanja odabrane aktivnosti klikom na ikonu za brisanje. Također, potrebno je napomenuti kako se zaslon automatski ažurira ako se doda ili obriše neka od kultura.

5.3.1. Dodavanje nove kulture

Korisnik u svakom trenutku može dodati novu kulturu na zemljištu. Tako može imati u isto vrijeme dvije ili više kultura na istom zemljištu. To je u svakodnevici normalno ako se radi o raznim povrtlarskim kulturama gdje korisnik ne želi ili ne može podijeliti zemljište u više manjih jer su oni upisani pod jedinstvenim katastarskim identifikatorom. Dodavanje nove kulture vrši se pritiskom na gumb u donjem desnom kutu na zaslonu koji prikazuje trenutne kulture na zemljištu te se zatim otvara novi zaslon prikazan na slici 5.14.



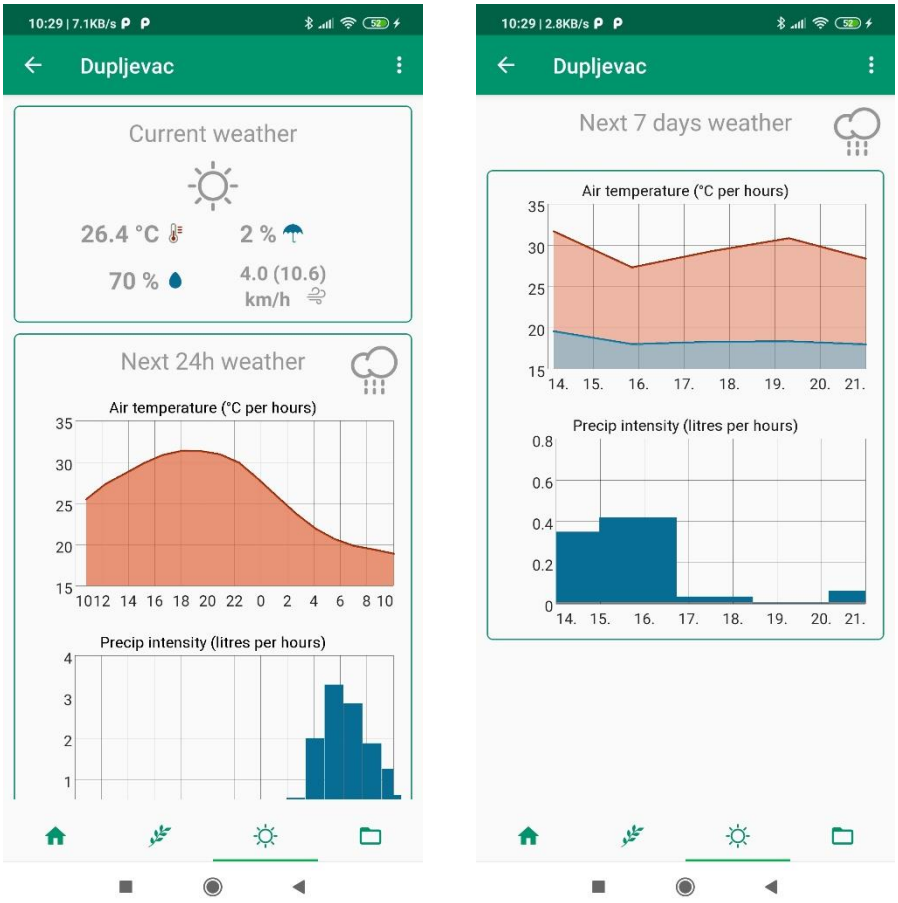
Slika 5.14. *Dodavanje nove kulture*

Kako je vidljivo na slici 5.14., prikazan je zaslon s mogućnošću dodavanja nove kulture na zemljište. Korisnik iz padajuće liste odabire jednu od ponuđenih kultura koju želi dodati (lijeska, orah, vinova loza, pšenica, kukuruz, rajčica ili krumpir), datum sadnje ili početka cvjetanja ovisno o aktivnosti, očekivani datum prinosa te očekivanu količinu prinosa. Za odabir datuma korišten je *dialog* s mogućnošću odabira datuma koji se zatim sprema i prikazuje na zaslonu. Za očekivani prikaz korisnik može odabrati jednu od ponuđenih mjernih jedinica: kg, t, l, hl, m i km te upisati određeni iznos koji će služiti kao mjerilo koliko je sezona bila uspješna. Klikom na kvačicu u

gornjem desnom kutu navigacijske trake kultura se sprema u bazu podatka ako su svi podatci popunjeni.

5.4. Vremenska prognoza za zemljište

Detalji o pojedinom zemljištu omogućuju i prikaz detaljne trenutne i dnevne vremenske prognoze te vremenske prognoze za idućih 7 dana. Ovakav način organiziranja i prikaza vremenske prognoze za svako pojedino zemljište korisniku omogućuje veću točnost i preciznost podataka nego da obrađuje informacije koje su za određenu regiju ili veće područje. Zaslon vremenske prognoze zadrži podatke o trenutnom meteorološkom stanju te grafove koji prikazuju temperaturu i intenzitet padalina ovisno o odabranom prikazu. Vremenska prognoza za zemljište dana je slikama 5.15. i 5.16.



Slike 5.15. i 5.16. *Vremenska prognoza za zemljište*

Kako je vidljivo iz slike 5.15. gornji dio aplikacije prikazuje trenutno vrijeme te bitne meteorološke podatke kao što su: trenutna temperatura, trenutna vlažnost zraka, trenutna vjerojatnost oborina te trenutna brzina i udari vjetra. Osim toga prikazana je sličica koja prikazuje

trenutno meteorološko stanje. Donji dio slike 5.15. i slika 5.16. prikazuju 2 grafa koji se odnose na vremensku prognozu za iduća 24 sata/idućih 7 dana. U gornjem desnom kutu prikazana je sličica koja prikazuje meteorološko stanje za odabrani budući period.

Prikaz grafova omogućen je korištenjem biblioteke *GraphView* koja prikazuje graf s određenim podatcima ovisno o vremenu na zaslonu. Moguće je kreirati proizvoljan tip grafa s proizvoljnim bojama i naslovima. Kako je vidljivo iz slika 5.15. i 5.16., temperaturni graf je prikazan crvenom bojom, dok je graf koji prikazuje intenzitet padalina prikazan plavom bojom.

Analizom grafova može se uvidjeti kako noć biti kišna te da će pasti mnogo kiše, čak 3 litre po metru kvadratnom, dok će temperatura zraka tijekom dana rasti a u kasnim noćnim satima se smanjivati zbog najavljene promijene vremena. Ovakva detaljna analiza uvelike pomaže planirati poljoprivredne aktivnosti, koje u većini slučajeva ovise o vremenskim prilikama koje se ne mogu obavljati po kiši. Uvidom u ovaj izvještaj poljoprivrednik može isplanirati sutrašnje aktivnosti bez većih poteškoća te pravodobno reagirati i zaštititi svoje usjeve i nasade.

5.5. Algoritam za obavještavanje korisnika na temelju praćenja vremenskih parametara i karakteristika biljnih kultura

Jedna od bitnijih i većih funkcionalnosti mobilne aplikacije je algoritam koji prati vremenske parametre i karakteristike biljnih kultura te na temelju izračunatih vrijednost obavještava korisnika ako je to potrebno. Analiza se vrši maksimalno jednom dnevno prilikom pokretanja aplikacije, a implementirana je pomoću Android servisa. Sam algoritam odvija se asinkrono te ne utječe na rad aplikacije te nije vidljiv korisniku. Asinkrona funkcionalnost implementirana je korištenjem Kotlin *Coroutines* funkcionalnosti. Analiza vremenskih parametara na temelju karakteristika biljnih kultura vrši se za svako zemljište kojemu je postavka *Analyze* uključena. Korisnik tu postavku može isključiti ili uključiti unutar zaslona koji prikazuje detalje o zemljištu. Sam proces analize se sastoji od 3 dijela:

1. Analiza vremenskih uvjeta prethodnih 30 dana
2. Analiza vremenske prognoze u ovisnosti o parametrima pojedine kulture za sljedećih 7 dana
3. Analiza moguće pojave bolesti na određenoj kulturi za sljedećih 7 dana

Prije analize podataka potrebno je dohvatiti vremenske podatke koji će se obrađivati. Oni se dohvaćaju pomoću Dark Sky API-a, koji, kako je rečeno u potpoglavlju 4.2., nudi podatke kako za trenutnu i buduću, tako i za prošlu vremensku prognozu te je to glavni razlog odabira Dark Sky

API-a u odnosu na mnoštvo drugih. Aplikacija podržava analizu i detektiranje mogućih neprilika za sljedeće kulture: lijeska, orah, vinova loza, pšenica, kukuruz, rajčica i krumpir. Svaka kultura ima sebi karakteristične parametre koji se koriste prilikom izračunavanja. Pa tako primjerice kultura kukuruz sadrži sljedeće podatke:

- idealna mjesečna količina padalina
- mjeseci u kojima se izračunava količina padalina
- minimalna podnošljiva temperatura i pripadajući mjeseci
- minimalna temperatura za klijanje i pripadajući mjeseci
- maksimalna podnošljiva temperatura za ljetne mjesece i pripadajući mjeseci
- popis mogućih bolesti

Prvi korak je analiza vremenskih uvjeta za prošlih 30 dana, pa bi tako analiza kulture kukuruza uključivala provjeru količine padalina za prošlih 30 dana te usporedba s nominalnom vrijednošću. Ako je ta razina padalina veća ili manja u određenom postotku, korisnik se obavještava s određenom porukom. Nadalje, dohvaća se prognoza za idućih 7 dana te se analizira svaki pojedini dan u ovisnosti o trenutnom mjesecu te karakteristikama pojedine kulture. Tako bi se za kulturu kukuruz analizirala minimalna podnošljiva temperatura, minimalna temperatura za klijanje (ako je trenutno razdoblje klijanja) te maksimalna podnošljiva temperatura. Osim navedenih karakteristika, za druge kulture se analizira još i vlažnost zraka te druge minimalne i maksimalne temperature zraka u ovisnosti o mjesecu.

Zadnji korak analize je analiza mogućeg razvijanja bolesti kulture. Aplikacija podržava detekciju i analizu parametara za tri bolesti: pepelnicu, plamenjaču i sivu pjegavost. Većina kultura može razviti neke od dvije od navedenih bolesti, pa tako primjerice vinova loza može razviti bolesti pepelnice i plamenjače, dok primjerice pšenica može obolijevati od sive pjegavosti i pepelnice. Za pojavu i širenje većine bolesti u poljoprivredi potrebna su razdoblja u kojima se izmjenjuje kiša, sunce i vjetar te povoljna vlažnost zraka. Na temelju dohvaćene vremenske prognoze analiziraju se podatci pojedinih bolesti koji su ugrađeni u mobilnu aplikaciju te se detektira slaba ili visoka mogućnost razvoja pojedine bolesti za određenu kulturu. Detaljniji prikaz algoritma dan je pseudokodom 5.1.

```
funkcija izvrši_analizu(){
        početak;
        Dohvati zemljišta korisnika za analizu;
        Za i = 0 do broj_kultura_korisnika - 1 (korak 1)
                Ako je kultura_u_tijeku onda:
                        funkcija analiziraj_prošlu_vremensku_prognozu;
```

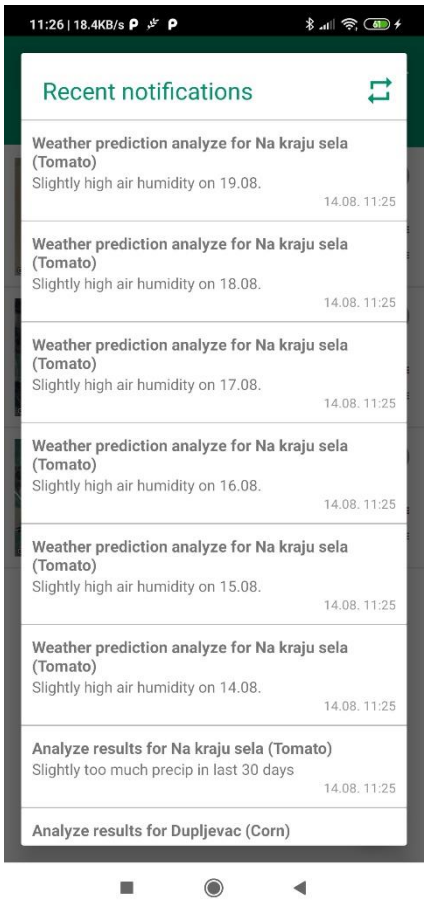
```
                    funkcija analiziraj_buduću_vremensku_prognozu;
        kraj;
}

funkcija analiziraj_prošlu_vremensku_prognozu(){
        početak;
        Dohvati vremensku prognozu za prošlih 30 dana;
        Ovisno o kulturi:
                Analiziraj padaline u prošlih 30 dana:
                        Ako je analiza negativna:
                                funkcija obavijesti_korisnika_određenom_porukom;
                Analiziraj sunčane dane u prošlih 30 dana:
                        Ako je analiza negativna:
                                funkcija obavijesti_korisnika_određenom_porukom;
        kraj;
}

funkcija analiziraj_buduću_vremensku_prognozu(){
        početak;
        Dohvati vremensku prognozu za budućih 7 dana;
        Analiziraj parametre kulture ovisno o vremenskoj prognozi:
                Ovisno o kulturi:
                        Analiziraj min i maks prihvatljivu temperaturu te vlažnost zraka:
                                Ako je analiza negativna:
                                        funkcija obavijesti_korisnika_određenom_porukom;
        Analiziraj moguću pojavu razvoja bolesti:
                Ovisno o kulturi:
                        Ovisno o mogućoj bolesti:
                                Ako su zadovoljenji uvijeti temperature, vjetra i vlage:
                                        funkcija obavijesti_korisnika_određenom_porukom;
        kraj;
}

funkcija obavijesti_korisnika_određenom_porukom(){
        početak;
        Kreiraj Android obavijest i prikaži ju;
        Spremi podatke o obavijesti u mrežnu bazu podataka;
        Ako je korisnik trenutno u aplikaciji:
                Ažuriraj broj novih obavijesti;
        kraj;
}
```

Pseudokod 5.1. *Algoritam za obavještavanje korisnika na temelju praćenja vremenskih parametara i karakteristika biljnih kultura*

5.5.1. Obavještavanje korisnika

Nakon što algoritam detektira mogućnost razvoja neke bolesti ili nepovoljnih prošlih ili budućih vremenskih uvjeta korisnik se obavještava valjanom porukom. Implementirana su dva načina obavještavanja: obavještavanje Android obavijestima i obavještavanje unutar aplikacije.

Nakon prikaza Android obavijesti korisnik klikom na obavijest pokreće aplikaciju i otvara zaslon odabranog zemljišta. Kako je vidljivo na slici 5.4. u gornjem desnom kutu prikazan je broj nepročitanih obavijesti. Odabirom ikone zvona otvara se zaslon koji prikazuje nepročitane obavijesti, prikazan na slici 5.17.



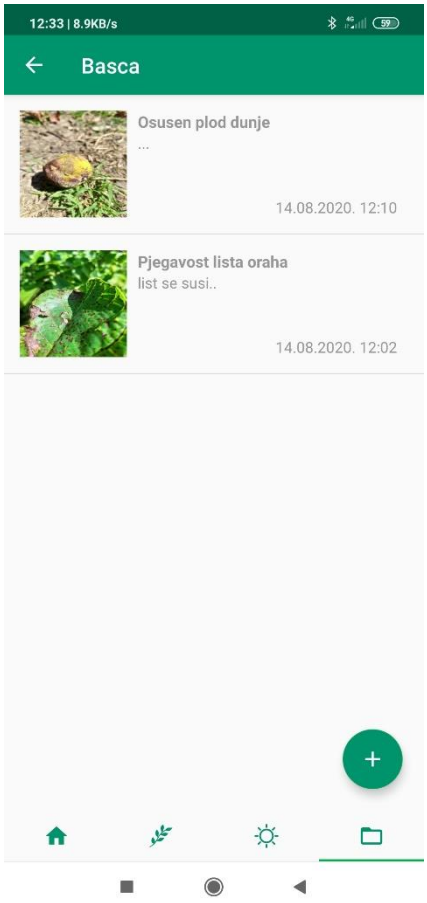
Slika 5.17. *Nepročitane obavijesti*

Kako je prikazano na slici 5.17., nepročitane obavijesti su prikazane u listi. Svaka obavijest sadrži naslov koji opisuje koji tip obavijesti je, za koje zemljište i koju kulturu. Tijelo obavijesti detaljno pojašnjava vremensku nepriliku ili moguću bolest u određenim razinama i za određeni datum. U donjem desnom kutu prikazan je datum kreiranja obavijesti. Pritiskom na ikonu u gornjem desnom kutu korisnik može ručno pokrenuti analizu iako možda nije prošlo 24 sata od prethodne analize.

5.6. Upravljanje dokumentima

Dodavanje novog dokumenta prvenstveno se odvija kao dodavanje nove slike s određenim opisom. Ako korisnik dozvoli dijeliti lokaciju, trenutna korisnikova lokacija će biti dohvaćena te

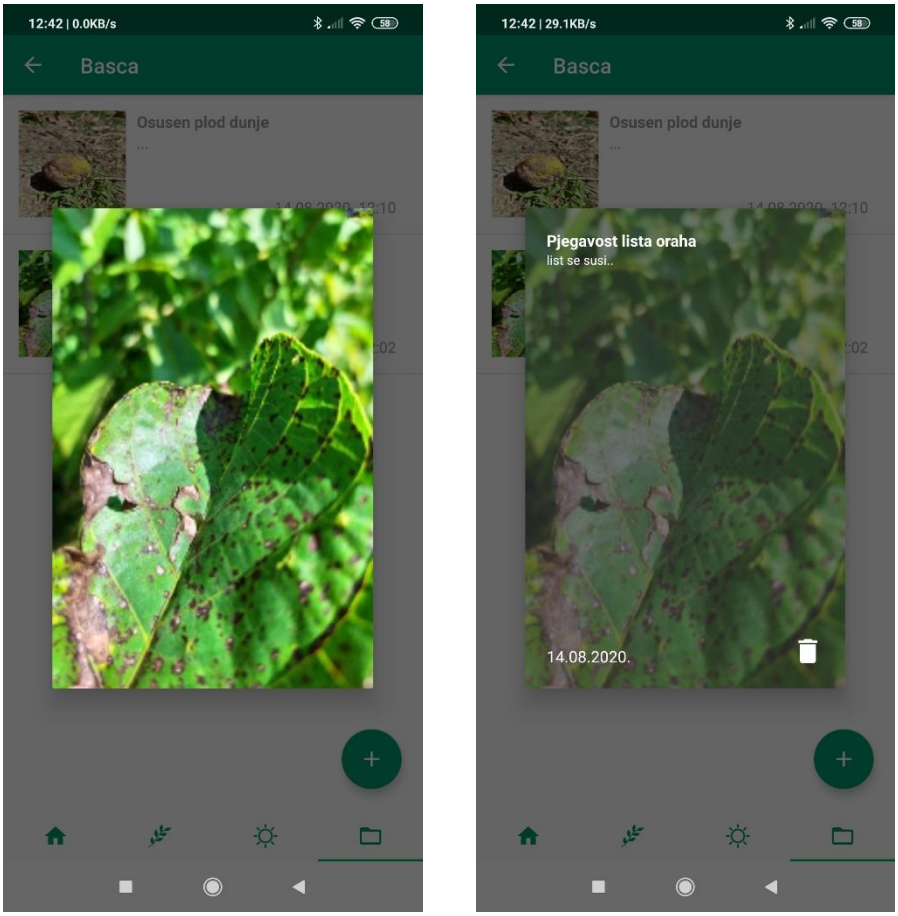
će se spremiti s dokumentom. U aplikaciju su implementirana dva načina prikaza i dodavanja dokumenata: oni koji ovise i koji se odnose na pojedino zemljište te oni koje se ne može svrstati u pojedino zemljište već korisniku služe samo kako bi spremio nekakvu informaciju. Odabirom opcije prikaza dokumenata na početnom zaslonu u *drawer* prikazu prikazuju se svi dokumenti korisnika, neovisno o lokaciji. Ako korisnik odabere željeno zemljište te otvori prikaz dokumenata za to zemljište, prikazat će mu se samo dokumenti za to zemljište. Prikaz je vidljiv na slici 5.18.



Slika 5.18. *Prikaz popisa dokumenata za pojedino zemljište*

Svaka obavijest sadrži sliku, naslov, opis te vrijeme spremanja dokumenta. Slike se spremaju na servise *Cloud Storage* dok su informacije spremljene u NoSQL bazu podataka *Cloud Firestore*. Dokumenti se automatski ažuriraju kako se dodaju ili brišu. Odabirom pojedinog dokumenta otvara se prikaz slike s informacijama, vidljiv na slikama 5.19. i 5.20. Klikom na učitanu sliku prikazuju se informacije o slici. Također, prikazuje se ikona za brisanje dokumenta te ako je odabrana dokument se briše iz baze podataka. Ovakav koncept spremanja dokumenata u ovisnosti o točnoj i preciznoj lokaciji uvelike pomaže poljoprivrednicima ako otkriju nekakvu bolest na

svom nasadu ili usjevu te znaju točnu lokaciju kako bi ju mogli precizno detektirati u slučaju tretiranja bolesti.



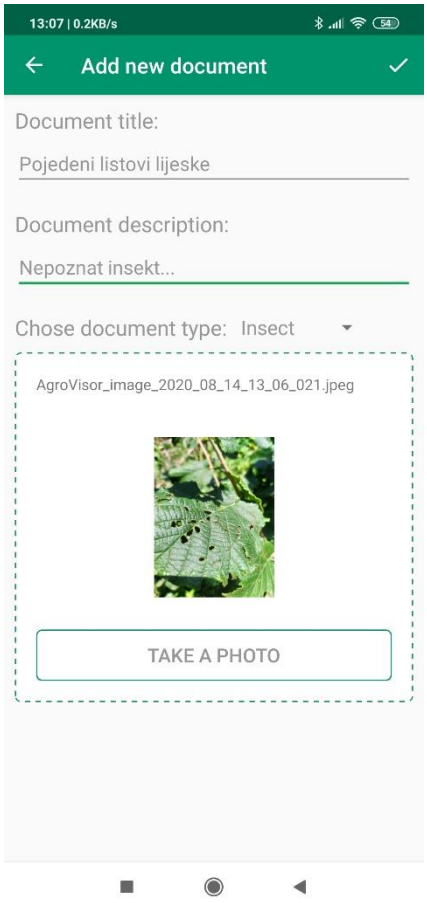
Slike 5.19. i 5.20. *Detaljan prikaz dokumenta*

5.6.1. Dodavanje novog dokumenta

Dodavanje novog dokumenta vrši se klikom na gumb u donjem desnom kutu, vidljiv na slici 5.18. Nakon odabira, otvara se zaslon za dodavanje novog dokumenta. Potrebno je upisati naslov dokumenta, njegov opis, te odabrati jedan od ponuđenih tipova dokumenata. On prvenstveno služi kako bi se na karti iscrtala drugačija ikonica. Trenutno su ponuđeni sljedeći tipovi dokumenata:

- Insekt
- Bolest
- Priroda
- Papirnati dokument
- Ostalo

Nakon odabira tipa dokumenta, klikom na gumb *Take a photo* otvara se kamera te korisnik ima mogućnost snimanja fotografije. Nakon potvrde snimljene fotografije, ona se prikazuje unutar okvira za sliku, vidljivo na slici 5.21. Prilikom učitavanja slike, nakon potvrđivanja dopuštenja aplikacije za pristup točnoj lokaciji ona se automatski dohvaća te se kasnije sprema s dokumentom. Ako korisnik ne dozvoli aplikaciji dopuštenje za pristup lokaciji ona se neće spremiti s lokacijom te se neće prikazati na mapi.

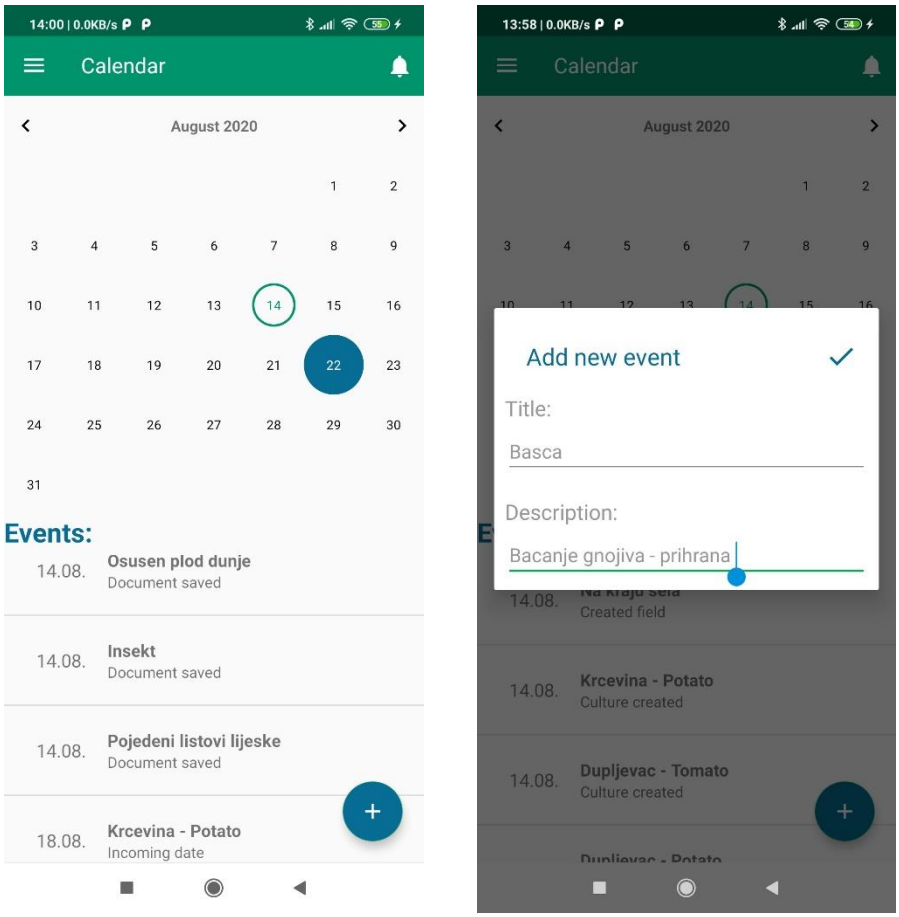


Slika 5.21. *Dodavanje novog dokumenta*

5.7. Upravljanje kalendarom

Kako je navedeno u potpoglavlju 5.2., svaki događaj je povezan s određenim zemljištem. Također, svaki događaj je potrebno zapisati i ispravno prikazati. Upravljanje kalendarom u mobilnoj aplikaciji omogućuje korisniku uvid u sve njemu bitne događaje za svaki mjesec prikazom događaja kao sortirani popis. Zaslon za prikaz događaja prikazuje se odabirom stavke unutar *drawer* funkcionalnosti aplikacije. Na slici 5.22. prikazan je početni zaslon prikaza dokumenta koji se sastoji od kalendara te popisa događaja za prikazani mjesec. Svaki događaj prikazan u listi sastoji se od naslova, opisa i datuma odvijanja događaja. Naslov je obično naslov

dokumenta ili zemljišta i kulture kojem pripada taj događaj, dok opis događaja može biti: dokument kreiran, zemljište kreirano, kultura kreirana, datum sadnje kulture, datum očekivanog prinosa kulture ili pak događaj može biti kreiran od strane korisnika za određeni datum što je vidljivo na slici 5.23.



Slike 5.22. i 5.23. *Prikaz događaja i dodavanje novog događaja*

Mijenjanjem mjeseca na kalendaru automatski se ažurira popis događaja s novim događajima za odabrani mjesec. Tako korisnik ima bolji pregled i uvid u događaje koji ga zanimaju. Odabirom određenog datum na kalendaru, datum se boja u plavo te se prikazuje gumb za dodavanje novog događaja u donjem desnom kutu, vidljiv na slici 5.22.

Odabirom gumba za dodavanje novog događaja, otvara se zaslon za unos informacija o događaju: naslov i opis događaja što je vidljivo slikom 5.23. Klikom na ikonu za spremanje događaj se sprema u mrežnu bazu podataka ako su podatci valjani. Ovakav način upravljanja događajima korisniku nudi određenu razinu fleksibilnosti i prilagodljivosti njegovim potrebama i radu jer je moguće spremiti bilo kakav događaj koji je korisniku bitan, a u poljoprivrednim djelatnostima događaji mogu biti različiti i potpuno neovisni jedan o drugom.

6. ZAKLJUČAK

Razvojem gospodarstva pojavljuje se mnoštvo problema s kojima se poljoprivrednik do tada nije susretao. Također, sve je više zapisa, papira, bilješki a u konačnici i sama njegova djelatnost je veća i šira. Gospodarstvo se širi na nova zemljišta, razvijaju i se nove kulture, ostaju neke stare. Sve je to potrebno imati na umu i pratiti. Uvođenjem sustava za upravljanje i organiziranje poljoprivrednih djelatnosti bilo na obiteljskom poljoprivrednom gospodarstvu ili u nekoj većoj organizaciji unapređuje se i olakšava rad i djelatnost općenito. Korisnik ima više vremena djelovati i raditi ono što mu je bitno. Napretkom današnje tehnologije napreduje i poljoprivreda. Mobilni uređaji današnjice posjeduju razne funkcionalnosti koje omogućuju prikupljanje, analizu i obradu podataka koje se mogu iskoristiti u poljoprivrednim djelatnostima, dok njihova prenosivost omogućuje direktnu interakciju korisnika s informacijama na mjestu događaja i u pravom trenutku.

U sklopu ovog rada kreiran je sustav za upravljanje biljnim kulturama. Sastoji se od Android mobilne aplikacije, mrežne baze podataka te pristupa vremenskoj prognozi putem API-ja. Korištenjem mrežne baze podataka omogućena je visoka fleksibilnost i prenosivost podataka između više uređaja, pa je tako moguće aplikaciju koristiti na različitim uređajima u isto vrijeme. Osim toga, podatci neće biti izgubljeni prilikom mijenjanja uređaja i ponovne instalacije aplikacije.

Android mobilna aplikacija je osmišljena tako da je baza svih događaja i aktivnosti zemljište, tj. svaka aktivnost koju poljoprivrednik obavlja direktno ili indirektno utječe na neko od njegovih zemljišta. Tako korisnik ima mogućnost dodavanja novog te uvid u postojeća zemljišta, bilo preko popisa zemljišta ili prikaza na karti. Za svako pojedino zemljište, korisnik može dodati novu kulturu ili aktivnost te odrediti trajanje aktivnosti te očekivani prihod. Nadalje, korisnik može uvidjeti trenutnu i dnevnu vremensku prognozu te prognozu se sljedećih 7 dana. Osim trenutne temperature, vlage zraka i brzine vjetra, prikazuje se i trenutna vjerojatnost padalina te pojedinačno, po svakom satu za idućih 24 sata ili danu za tjedni prikaz, temperatura zraka i količina padalina. Nadalje, korisnik ima mogućnost uvida u dokumente te mogućnost dodavanja novog snimanjem fotografije s prigodnim opisom te trenutnom korisnikovom lokacijom. Također, korisnik može uvidjeti događaje u kalendaru kreirane od strane aplikacije te dodati svoj proizvoljan.

Implementiranjem navedenih funkcionalnosti omogućeno je kreiranje i razvoj algoritma koji na temelju karakteristika biljnih kultura te vremenske prognoze obavještava korisnika o mogućim nedaćama za njegov nasad ili usjev. Algoritam je osmišljen tako da prati vremensku prognozu za prošlih 30 i budućih 7 dana. Korištenjem aplikacije, korisnik je pravovremeno obaviješten o teškim

vremenskim neprilikama koje proživljava njegova biljka ili pak kako postoji mogućnost razvoja i širenja bolesti kod biljke koja može poharati njegov usjev i plod te tako smanjiti ili uništiti prinose. Pravovremeno obaviješten, korisnik može reagirati, zaštititi svoj usjev i povećati kvantitetu i kvalitetu roda i prinosa, a sve to uz korištenje tehnologije u pravu svrhu, kao pomoć.

LITERATURA

[1] A. D. Hwang, 7.5 billion and counting: How many humans can the Earth support?, The Conversation, 09.6.2018., dostupno na: https://theconversation.com/7-5-billion-and-counting-how-many-humans-can-the-earth-support-98797 [13.6.2020.]

[2] World Vision, Why are so many people in the world hungry?, dostupno na: https://www.worldvision.com.au/global-issues/work-we-do/famine/why-are-so-many-people-in-the-world-hungry#top-scroll [13.6.2020.]

[3] I. Russel, What is the best farm management software?, Quora, 20.1.2019., dostupno na: https://www.quora.com/What-is-the-best-farm-management-software [14.6.2020.]

[4] N. K Van Alfen, Encyclopedia of agriculture and food systems, University of California, Davis, CA, USA, 2014. godina

[5] R. Panneerselvam, Database Management Systems, Pondicherry University, Pondicherry, 2018. godina

[6] Agrivi, O nama, Agrivi, dostupno na: https://www.agrivi.com/pocetna/o-nama [14.6.2020.]

[7] ExactFarming, About, ExactFarming, dostupno na: https://www.exactfarming.com/en/exactfarming-company/ [14.6.2020.]

[8] Imanuel, Top 9 Farm Management Software, Predictive Analitics Today, dostupno na: https://www.predictiveanalyticstoday.com/top-farm-management-software/ [14.6.2020.]

[9] Stat Counter, Mobile Operating System Market Share Worldwide May 2019 – May 2020, dostupno na: https://gs.statcounter.com/os-market-share/mobile/worldwide [14.6.2020.]

[10] J. Callaham, The history of Android OS: its name, origin and more, Android Authority, 18.8.2019., dostupno na: https://www.androidauthority.com/history-android-os-name-789433/ [21.6.2020]

[11] Android Developers, Platform Architecture, 07.5.2020., dostupno na: https://developer.android.com/guide/platform [21.6.2020.]

[12] BenchmarksGame, Java Versus C++ g++ fastest programs, dostupno na: https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/java.html [21.6.2020.]

[13] JR Raphael, Android versions: A living history from 1.0 to 11, Computerworld, 22.2.2020., dostupno na: https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html [21.6.2020.]

[14] C. Scott Brown, Android turns 10 today: Here are 5 features we still use from Android 1.0, Android Authority, 23.9.2018., dostupno na: https://www.androidauthority.com/android-1-0-features-905923/ [21.6.2020.]

[15] Android Developers, Sensors Overview, 27.12.2019., dostupno na: https://developer.android.com/guide/topics/sensors/sensors_overview [21.6.2020.]

[16] J. Hildenbrand, How does GPS work on my phone?, Android Central, 24.8.2018., dostupno na: https://www.androidcentral.com/how-does-gps-work-my-phone [21.6.2020.]

[17] Android Developers, Meet Android Studio, 03.6.2020., dostupno na: https://developer.android.com/studio/intro [22.6.2020.]

[18] Android Developers, Android's Kotlin-first approach, 10.6.2020., dostupno na: https://developer.android.com/kotlin/first [27.6.2020.]

[19] D. Jemerov i S. Isakova, Kotlin in Action, Manning Publications, 02.2017.

[20] Postman, What is Postman?, dostupno na: https://www.postman.com/ [27.6.2020.]

[21] Firebase, A comprehensive app development platform, dostupno na: https://firebase.google.com/ [29.6.2020.]

[22] D. Stevenson, What is Firebase? The complete story abriged, Medium, 25.9.2018., dostupno na: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 [29.6.2020.]

[23] H. Fatima, Why is Firebase the best Mobile Backend-as-a-Service, ResellerClub, 12.12.2017., dostupno na: https://blog.resellerclub.com/why-is-firebase-the-best-mobile-backend-as-a-service/ [29.6.2020.]

[24] Firebase, Cloud Firestore, dostupno na: https://firebase.google.com/products/firestore [29.6.2020.]

[25] Firebase, Cloud Storage, dostupno na: https://firebase.google.com/products/storage [30.6.2020.]

[26] DarkSky, Dark Sky API – Overview, dostupno na: https://darksky.net/dev/docs [30.6.2020.]

[27] L. Vogel, S. Scholtz, D. Weaiser, Using Retrofit 2.x as REST client, Vogella, 05.6.2018., dostupno na: https://www.vogella.com/tutorials/Retrofit/article.html [30.6.2020.]

[28] A. Bishit, MVVM (Model View ViewModel) + Kotlin + Google JetPack, Medium, 02.5.2019., dostupno na: https://medium.com/@er.ankitbisht/mvvm-model-view-viewmodel-kotlin-google-jetpack-f02ec7754854 [01.7.2020.]

[29] C. Tech, MD5 and SHA256 in Java Kotlin and Android, CodeMonk, 26.12.2019., dostupno na: https://www.javacodemonk.com/md5-and-sha256-in-java-kotlin-and-android-96ed9628 [04.7.2020.]
[30] Developers, Maps Static API, 19.6.2020., dostupno na: https://developers.google.com/maps/documentation/maps-static/intro [04.7.2020.]

SAŽETAK

Razvojem poljoprivrede, poljoprivrednici sve više vremena ne provode obavljajući poljoprivrednu djelatnost. Sustav za upravljanje biljnim kulturama kao cilj ima poboljšati i unaprijediti poljoprivrednu djelatnost. Također, osim unapređivanja određenih procesa, sustav treba dati na uvid potrebne informacije korisniku u bilo kojem trenutku. Zahtjevi na aplikaciju rješavani su tako da je prvo osmišljen cjelokupni sustav koji se sastoji od mobilne aplikacije i mrežne baze podataka. Kreiranjem i implementiranjem Android mobilne aplikacije u programskom jeziku Kotlin, korisniku je omogućeno upravljanje zemljištima, upravljanje kulturama i kalendarom, uvid u vremensku prognozu te dohvaćanje svih potrebnih informacija na temelju geografske lokacije i prikaz na karti. Korištenjem mrežne baze podataka omogućena je prenosivost podataka te je tako moguće koristiti aplikaciju na više uređaja istovremeno. U konačnici osmišljen je i razvijen algoritam za obavještavanje korisnika na temelju praćenja vremenskih parametara i karakteristika biljnih kultura. Pravovremeno obaviješten, korisnik može na vrijeme reagirati te zaštiti svoj nasad ili usjev te povećati kvalitetu i kvantitetu krajnjeg proizvoda.

Ključne riječi: Android, biljne kulture, Kotlin, mobilna aplikacija, upravljanje

ABSTRACT

Smartphone-based plant management system

With the development of agriculture, the farmers are spending a lot of time doing side jobs instead of performing agricultural activities. The goal of building a plant management system is to improve and enhance agricultural activities of such a company. In addition to improving certain processes within a company, the system should provide the necessary information to the user at any time. The application requirements were solved by first designing a complete system consisting of a mobile application and network database. By creating and implementing the Android mobile application in the Kotlin programming language, the user is enabled to manage land, manage crops and calendar, view weather forecast and retrieve all necessary information based on geographical location and display it on a map. Using a network database enables data portability, so it is possible to use the application on multiple devices simultaneously. Finally, an algorithm for informing users based on monitoring the weather forecast parameters and characteristics of plant crops was designed and developed. Informed in a timely manner, the user can react in time and protect his plants and increase the quality and quantity of the final product.

Keywords: Android, Kotlin, management, mobile application, plants

ŽIVOTOPIS

Bernard Kekelić rođen je 10. prosinca 1996. godine u Požegi. 2011. godine završava Osnovnu školu Stjepana Radića u Čaglinu, dok srednju tehničku školu u Požegi, smjer Tehničar za računalstvo završava 2015. godine. Po završetku srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Računarstvo na preddiplomskom sveučilišnom studiju, koji završava tri godine kasnije, 2018. godine. Odmah nakon završetka preddiplomskog studija, na istom fakultetu upisuje diplomski studij, smjer Programsko inženjerstvo, koji trenutno pohađa.

Bernard Kekelić

_____