Mobilna aplikacija za rezervaciju i naplatu najma pametnog suncobrana

Žiković, Mateo

SVEUČILIŠTE JURJA DOBRILE U PULI

Fakultet informatike u Puli

Mateo Žiković

MOBILNA APLIKACIJA ZA REZERVACIJU I NAPLATU NAJMA PAMETNOG SUNCOBRANA

Diplomski rad

Pula, 2023.

SVEUČILIŠTE JURJA DOBRILE U PULI

Fakultet informatike u Puli

Mateo Žiković

MOBILNA APLIKACIJA ZA REZERVACIJU I NAPLATU NAJMA PAMETNOG SUNCOBRANA

Diplomski rad

Studijski smjer: Informatika

Kolegij: Mobilne aplikacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc.dr.sc Siniša Sovilj

Pula, rujan 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Mateo Žiković, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, rujan 2023.

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Mateo Žiković dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom „Mobilna aplikacija za rezervaciju i naplatu najma pametnog suncobrana“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student

_____

U Puli, rujan 2023.

Sadržaj

1. Uvod

U ovom diplomskom radu opisuje se izrada aplikacije za rezerviranje i naplatu pametnih suncobrana. Aplikacija se sastoji od pregleda dostupnih plaža, opcije odabira ležaja na plaži te mogućnosti rezervacije tih ležaja.

Diplomski rad se sastoji od dva dijela. Prvi dio opisuje povijest Androida kao platforme, usporedbu sa drugim mobilnim operativnim sustavima i tehnologije koje se koriste pri razvoju aplikacija za Android platformu poput Java programskog jezika, Firebase baze za spremanje podataka, te Android Studio kao okruženje za razvoj aplikacija.

Drugi dio se sastoji od dokumentacije same aplikacije te se općenito opisuje razvoj aplikacije, od use-case dijagrama, sekvecijalnog dijagrama i dijagrama klasa.

2. Povijest Android operativnog sustava i alata

Android je mobilni operativni sustav koji je tvrtka Google predstavila 2008. godine. Od njegovog predstavljanja postaje najkorišteniji mobilni operativni sustav te 2023. godine ima preko 3,3 milijarde korisnika[1]. Jedan od glavnih razloga popularnosti Android platforme je zbog toga što je srce sustava otvorenog koda, takozvani Android Open Source Project ili skraćenica AOSP te radi toga se sustav može pokretati na različitom hardveru, iako većina uređaja danas pokreće Google-ovu inačicu sustava.

Android je baziran na Linux kernelu te je izrađen na kolekciji biblioteka koje pružaju osnovne mogućnosti sustava. Neke od tih biblioteka su Android Runtime (ART) koja omogućuje izvršavanje aplikacijskog koda, Android Framework koji je kolekcija API-jeva i alata koju developeri koriste za izradu aplikacija i ostalo.

Jedna od najvećih prednosti Android-a je mogućnost da pokreće aplikacije na velikom broju uređaja, od jeftinih mobilnih uređaja do skupljih tableta i mobilnih uređaja. Također omogućuje izradu aplikacija sa alatima poput Android Studio koji olakšava rad programera pošto ima mogućnosti pisanja koda, *debugger*-a i vizualne izrade *layout*-a aplikacije.

Android sustav ima ogroman utjecaj na razvoj mobilne industrije te velik utjecaj na razvoj društva zbog velikog broja korisnika. U ovom poglavlju će se ukratko opisati povijest Android operativnog sustava i nekih alata za razvoj aplikacija na Android sustavu.

1 https://www.demandsage.com/android-statistics/

2.1 Početak Android sustava

Povijest Android-a započinje u listopadu 2003. godine u Palo Altu, njegovi začetnici su Rich Miner, Nick Sears, Chris White i Andy Rubin. Originalna ideja za sustav je bila da se koristi u digitalnim kamerama, ali se nakon nekoliko mjeseci razvoj prebacio na razvoj sustava za mobilne uređaje.[2]

Google kupuje Android 2005. godine te Rubin i ostali nastavljaju raditi na projektu pod okriljem novog vlasnika te su za jezgru operativnog sustava odabrali Linux OS. Zbog odabira Linux sustava je platforma mogla raditi na mobilnim uređajima većeg broja proizvođača. Rubin je bio na čelu Android-a sve do 2013. godine.

U 2007. godini Apple izdaje svoj prvi iPhone mobilni uređaj te nedugo nakon toga Google najavljuje svoju platformu. Google formira takozvani Open Handset Alliance, skupinu proizvođača mobilnih uređaja i mikročipova za HTC, Motorola, Quallcom i ostale. Prvi uređaj sa Android operativnim sustavom je T-Mobile G1, poznatiji pod nazivom HTC Dream.

Slika 1: T-Mobile G1 (Izvor: https://upload.wikimedia.org/wikipedia/commons/b/be/HTC_Dream_Orange_FR.jpeg)

2 https://www.androidauthority.com/history-android-os-name-789433/

2.2 Verzije Androida kroz povijest

Android sustav od svog predstavljanja 2008. godine dobiva konstantne nadogradnje sustava koje su u ranijim danima nazivane tematski po slatkišima, dok se od Android 10 verzije nazivaju samo po brojci.

U tablici će se prikazat pregled verzija kroz godine te poboljšanja koja su uvedena sa svakom verzijom.[3]

Tablica 1: Verzije Androida

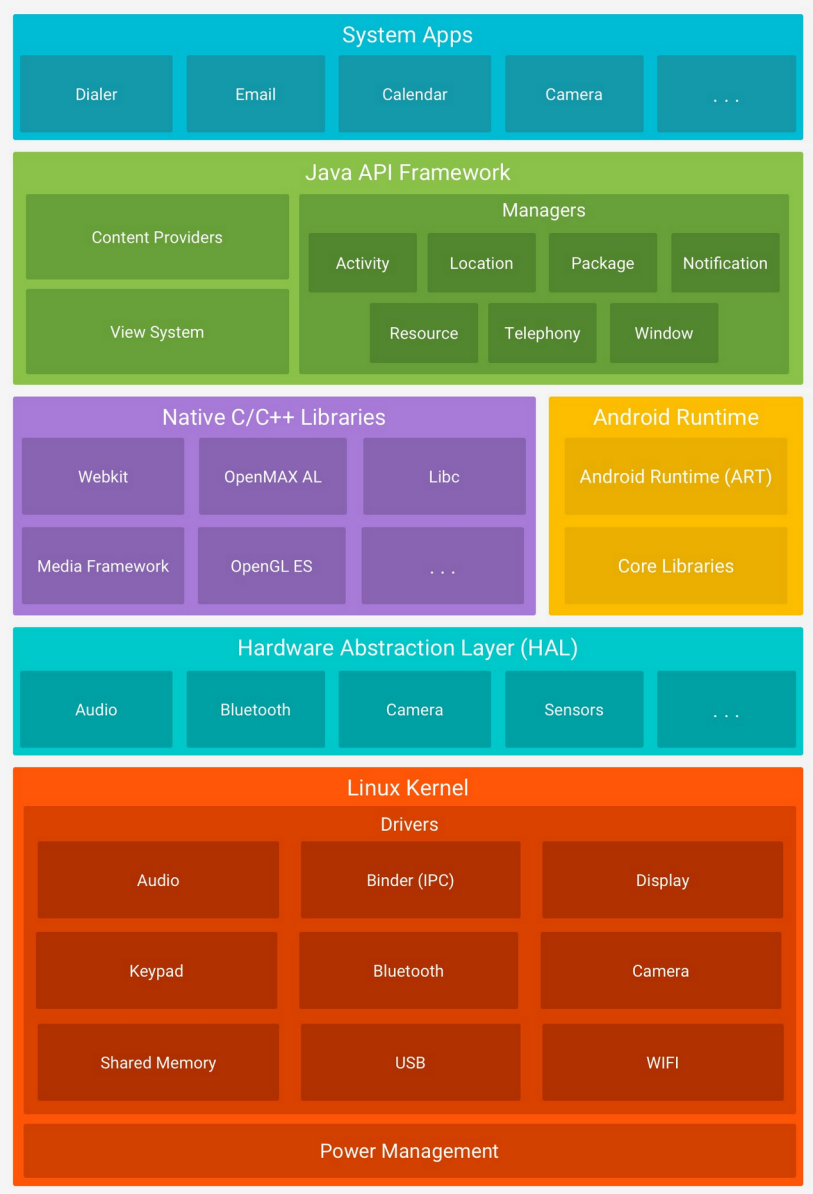
Naziv	Opis
Android 1.0	Prva verzija sadržavala je osnovne Google aplikacije poput Gmaila, Maps, Calendar i Youtube
Android 1.5: Cupcake	On-Screen tipkovnica, widgeti
Android 2.0 – 2.1: Eclair	Glasovna navigacija, speech-to-text, pinch-to-zoom
Android 2.2: Froyo	Poboljšanja sustava, dock sa ikonama, Flash mogućnosti
Android 2.3: Gingerbread	UI Redizajn
Android 3.0 – 3.2: Honeycomb	Sustav za tablete
Android 4.0: Ice Cream Sandwich	UI Redizajn
Android 4.1 – 4.2: KitKat	UI Redizajn
Android 4.1 – 4.3: Jelly Bean	Google Now, redizajn UI, multi-user support
Android 4.4: KitKat	Glasovna kontrola, redizajn sučelja
Android 5.0: Lollipop	Material dizajn, upravljanje notifikacijama
Android 6.0: Gingerbread	USB-c podrška, podrška za čitač otiska
Android 7.0: Nougat	Google Asistant – glasovni asistent
Android 8.0: Oreo	Picture-in-picture, gašenje notifikacija
Android 9.0: Pie	Upravljanje sustavom pomoću gesta
Android 10	Poboljšanje upravljanja gestama, Dark mode
Android 11	Poboljšanja privatnosti
Android 12	Redizajn sučelja – Material You, poboljšanja widgeta
Android 13	Poboljšanja uporabe sučelja za tablete
Android 14	Poboljšanja UI, optimizacija baterije

3 https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html?page=2

Nova verzija sustava izlazi otprilike jednom godišnje te zadnje verzije većinom donose inkrementalna poboljšanja performansi, promjene izgleda i optimizacije sustava.

2.3 Arhitektura Android operativnog sustava

Platforma Android-a sastoji se od niza različitih komponenti. Sastoji se od osnovnih aplikacija kao što su kontakti i elektronička pošta, skupa API-a koji će pomoći kontrolirati izgled i ponašanje aplikacije uz brojne podržavajuće datoteke i biblioteke.



Slika 2: Android arhitektura (https://developer.android.com/guide/platform)

Dijelovi Android platforme mogu se podijeliti na sljedeće dijelove:[4]

- **Linux kernel** – osnova Android platforme. Na primjer, Android runtime oslanja se na Linux kernel za osnovne funkcije kao što su upravljanje memorijom niske razine (eng. *low-level memory*) i dretvama.
- **Hardverski apstraktni sloj** – pruža standardna sučelja koja Java API-u izlažu harvderske mogućnosti uredaja. Sastoji se od više modula biblioteka, a svaki modul implementira specifičnu vrstu sučelja za određenu hardversku komponentu, kao što je kamera ili *Bluetooth*.
- **Android runtime –** temelj za aplikacijski okvir, pomoću njega se pokreću aplikacije i biblioteke. Sadrži komponente kao što su osnovne biblioteke te Dalvik, virtualni stroj.
- **Nativne C/C++ biblioteke** - biblioteke za pružanje podrške za Android razvoj.
  Neke od takvih biblioteka su: *Media library* za reprodukciju i snimanje audio i video formata, grafičke biblioteke *SGL* i *OpenGL* za 2D i 3DC grafiku, *SQLite* za podršku baze podataka, *FreeType* za podršku fontova i mnoge druge.
- **Java API** – gradivni blokovi koji su potrebni za izradu Android aplikacija. Neki od modularnih komponenti i usluga su: upravljač resursima (eng. *resource manager*), upravljač notifikacijama (eng. *notification manager*), upravljač aktivnosti (eng. *activity manager*), pružatelj sadržaja (eng. *content provider*) i slično.
- **Sistemske aplikacije** – gornji sloj Android arhitekture koji je vidljiv korisnicima. Neke od temeljnih aplikacija su aplikacija za elektroničku poštu, slanje SMS poruka, pregledavanje interneta, imenik i slično.

4 https://developer.android.com/guide/platform

2.4 Programiranje aplikacija za Android operativni sustav

Android aplikacije mogu biti pisane pomoću Java programskog jezika, Kotlin-a, i C++ jezika. Android SDK (Software Development Kit) pretvara programski kod u .apk datoteke koje potom mogu biti instalirani na Android uređaj.

U globalu, svaka aplikacija se vrti unutar svoje kutije (eng. *Sandbox)* u svrhu sigurnosti sustava te su neke od značajki aplikacija sljedeće[5]:

- Android sustav je Linux sustav sa više korisnika, gdje je svaka aplikacija unikatan korisnik.
- Android svakoj aplikaciji dodjeljuje poseban korisnički identifikator, o kojem aplikacija ne zna ništa te sustav svakom korisniku dodjeljuje dopuštenja o pristupu datoteka neke aplikacije.
- Svaki proces ima svoj virtualni stroj (eng. vi*rtual machine)* zbog čega se kod aplikacije vrti u izolaciji od ostalih aplikacija.
- Svaka aplikacija ima svoj Linux proces, koji se pokreće ovisno o komponentama neke aplikacije te se isto tako gasi kada više nije potreban ili kada sistem mora alocirati memoriju na ostale dijelove sustava.

Implementirano je *načelo namanje privilegije* što znači da svaka aplikacija ima pristup onim resursima koji su potrebni da funkcionira i ničemu više tako da se stvori sigurno okruženje gdje aplikacije ne mogu pristupati resursima sustava za koje nemaju dopuštenje.

Postoji način da se aplikacijama dopusti da imaju pristup drugim aplikacijama ili servisima:

- Dvije aplikacije mogu imati isti Linux korisnički identifikator te samim time mogu pristupati datotekama jedne i druge aplikacije te također mogu pokretati iste Linux procese i virtualne strojeve.
- Aplikacija može tražiti dopuštenje da koristi podatke ili usluge mobilnog uređaja, poput lokacije, kamere i Bluetooth konekcije. Korisnik mora eksplicitno tražiti dopuštenja sustava u kodu aplikacije.

5 https://developer.android.com/guide/components/fundamentals?hl=en

2.5 Odabir programskog jezika

Java je jedan od najpopularnijih programskih jezika koji seže dugo u povijest do 90ih godina te se koristi za razvoj web aplikacija, desktop i mobilnih aplikacija.

Kotlin je noviji programski jezik koji je postigao veliku popularnost te sada je popularniji od Jave za izradu Android aplikacija. Čak preko 50% Android developera koristi Kotlin kao primarni programski jezik, a 70% developera se slažu da ih Kotlin čini produktivnijima.[6]

Slika 3: Razlike u kodu između Jave i Kotlina (Izvor: https://mlsdev.com/blog/kotlin-vs-java)

U gornjem primjeru možemo vidjeti razliku u dužini koda. Dok se kod pisanja podatkovnih klasa (POJO – *Plain Old Java Object*) u Javi mora odrediti konstruktor, *getter* i *setter* metode te neke druge

6 https://kotlinlang.org/docs/android-overview.html

funkcije, Kotlin klase u sebi imaju ugrađene *getter* i *setter* metode pošto su konstruktori automatski zadani kao *public* i time zahtjevaju pisanje manje linija koda.

Također Kotlin nudi *null safety*, što znači da Kotlin ne može izbaciti greške koje se u Javi nazivaju *NullPointerExceptions* i rezultat toga je sigurniji kod te programer ne mora pisati dodatni kod kako bi se zaobišla ta greška.

Kotlin je dakle objektno orijentiran, statički programski jezik, koji se isto poput Jave vrti u JVM okruženju, koji se također može koristiti za izradu serverskog dijela (eng. *server-side*) i klijentskog dijela (eng. *client-side*) aplikacije. Google je 2019. godine deklarirao Kotlin kao glavni izbor programskog jezika za izradu Android aplikacija.

Kotlin je bolji izbor za izradu aplikacija u Androidu, pošto je jezik sigurniji, kod je više čitljiv i puno manje linija koda je potrebno napisati prilikom izrade iste aplikacije koristeći Javu.

U ovom radu je korištena Java pošto je jezik stariji i sadrži više resursa za učenje te se Java može koristiti u više svrha, na primjer u popularnom Spring okruženju za izradu web aplikacija.

2.5.1 Cross-platform jezici

Kod izrade aplikacija za mobilne uređaje, postoji izbor između *native* i *cross-platform* aplikacija. Kod *native* aplikacija, koriste se jezici Kotlin ili Java za Android sustav i Objective-C ili Swift za iOS aplikacije. Jedan od problema kod *native* pristupa je da se posebno mora pisati kod za svaku platformu što udvostručuje resurse potrebne za izradu aplikacija koje će se koristiti na oba dvije platforme.

Ako su resursi i vrijeme ograničeni onda se za taj problem koriste *cross-platform* jezici poput Flutter-a, React Native-a, Xamarin i ostali.

Cross-platform aplikacije imaju svoje prednosti poput dijeljenja koda što rezultira ubrzanom izradom aplikacija, no imaju i svojih mana. Pošto svaka platforma ima svoje određene značajke za korisničko sučelje, samo nativne aplikacije mogu iskoristiti sve opcije koje pruža sustav, npr. GPS, kamera i ostali senzori. Također nativne aplikacije u globalu imaju bolje performanse.[7]

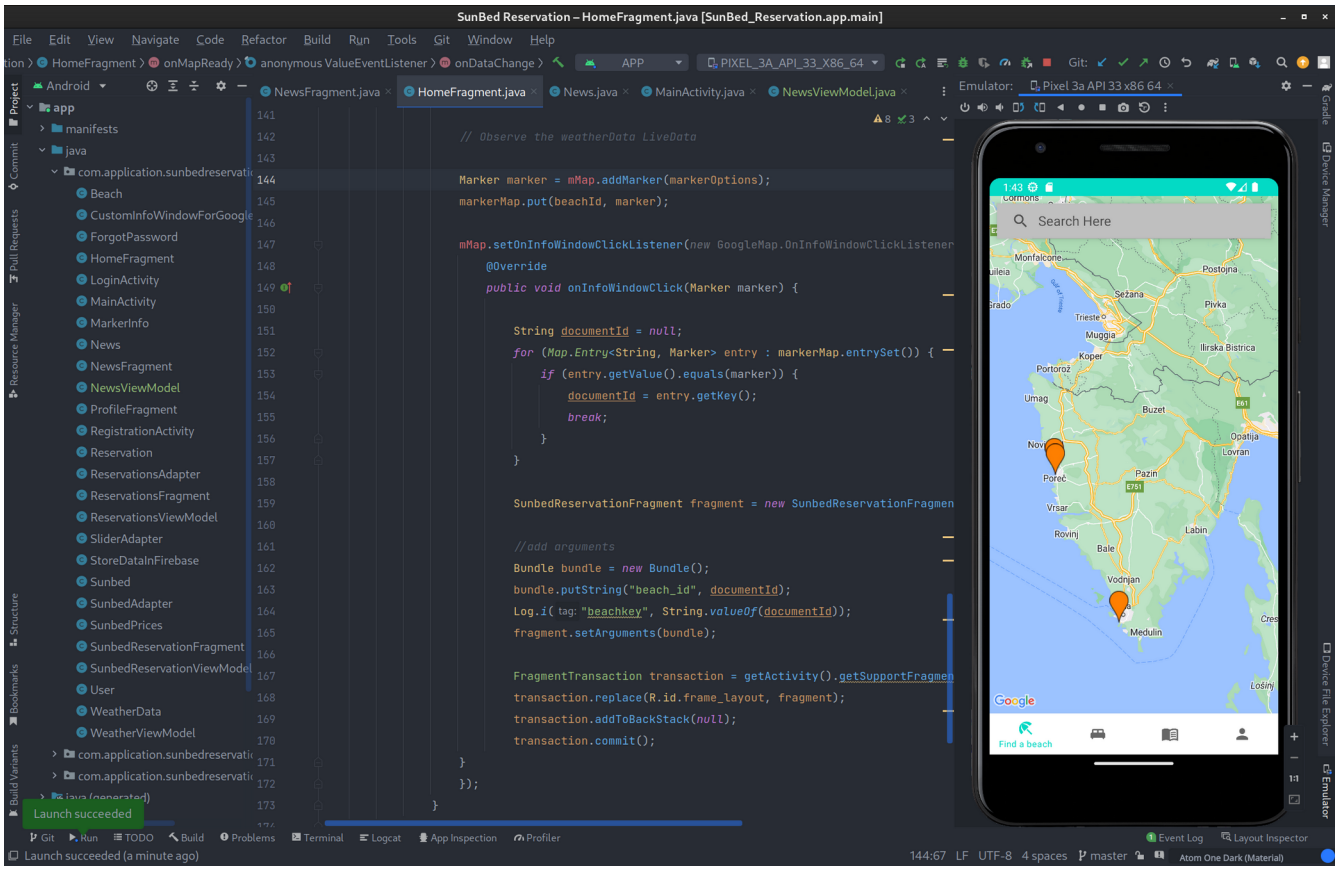
7 https://www.appschopper.com/blog/pros-cons-cross-platform-mobile-app-development/

2.6 Android Studio

Android Studio je službeni IDE (Integrated Development Environment) za izradu Android Aplikacija, koji je proizveden i održavan od strane tvrtke JetBrains.

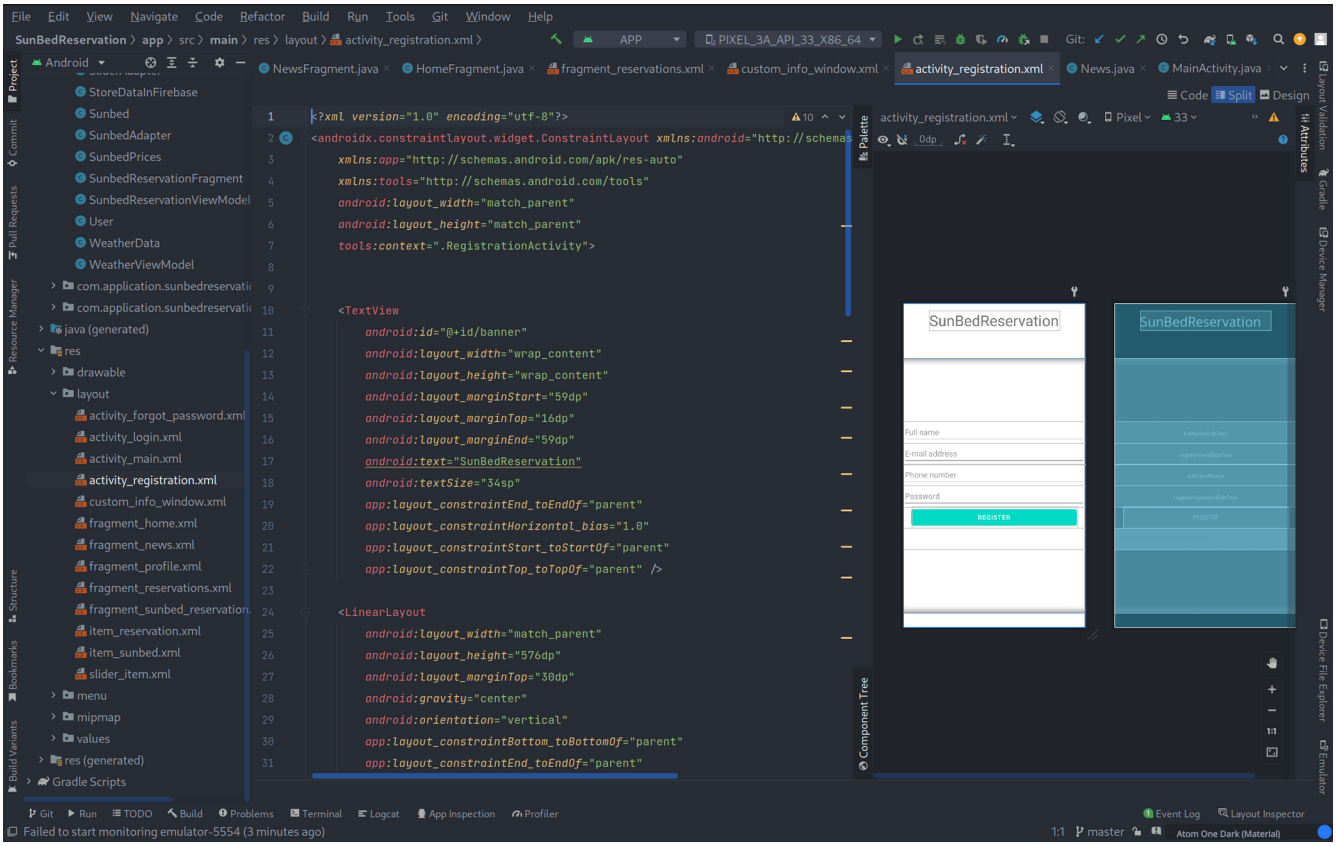
Android Studio ima više značajki:

- Gradle sustav za izradu aplikacija
- Emulator mobilnih uređaja
- Mogućnosti izrade aplikacija za više vrsta uređaja (Tableti i WearOS)
- GitHub integracija
- Lint alati za provjeru točnosti koda, verzija i ostalo.
- Alati za testiranje



Slika 4: Sučelje Android Studia (izvor: Autor)

Na gornjoj slici (slika broj 4) možemo vidjeti primjer sučelja Android Studio. Na sučelju sa lijeve strane imamo strukturu projekta sa svim datotekama aplikacije, u sredini imamo prozor za pisanje koda i na desnoj strani imamo pokrenut emulator za mobilni uređaj gdje možemo testirati aplikaciju.



Slika 5: Sučelje Layout Editora (Izvor: Autor)

Također u sebi ima ugrađen *Layout Editor*, gdje možemo pomoću XML-a i editora dizajnirati korisničko sučelje aplikacije.

Android Studio je trenutno najbolji alat za izradu Android aplikacija. U sebi ima ugrađene alate za početno postavljanje projekta, alate koji provjeravaju greške u kodu direktno dok pišemo kod, Gradle sustav za izgradnju projekta i korištenje mnogih *third-party* biblioteka, emulator, uređivač sučelja te integracija sa ostalim Google servisima.

2.7 Java programski jezik

Java je objektno orijentirani, klasno baziran jezik koji je kreiran 1995. godine u Sun Microsystems-u. Tvorac James Gosling bio je voditelj projekta koji je zajedno sa kolegama Mike Sheridanom i Patrickom Naughtonom pokrenuo projekt 1991. godine. Java je originalno trebao biti jezik koji pokreće niz elektroničkih uređaja za krajnje korisnike.

Prva službena verzija 1.0 izdana je 1995. godine, a jezik je zamišljen kao sustav koji isti kod može pokretati na više uređaja (*Write Once, Run Anywhere*). Izvršavanje koda na više uređaja omogućuje Java virtualni stroj (eng. *Java Virtual Machine*, skraćeno JVM).

Daljnji razvoj je uslijedio s uvodnom Java 2 platformom, Standard Edition (J2SE) 1998. godine, koja je donijela poboljšanja i nadogradnje programskog jezika. Java se i dalje razvija, a nova poboljšanja i značajke uvedene su u kasnijim verzijama, kao što je Java SE 8 iz 2014. godine, a posebno se ističe projekt OpenJDK koji je izdan 2006. godine kao *open-source* alat za razvoj Java aplikacija.

U ovom potpoglavlju će se opisati Java programski jezik, njegova arhitektura, kako se uklapa u Android sustav i usporedba sa Kotlin jezikom koji se također vrti na Java Virtual Machine-u.[8]

2.7.1. Java Virtual Machine

JVM je apstraktni stroj, koji ima svoj set instrukcija i koji pri svom radu manipulira sa memorijom poput fizičkog računala. Oracle-ove trenutne implementacije JVM-a rade na mobilnim, desktop i serverskim računalima. To znači da JVM nema fiksnu arhitekturu nego se njegov instrukcijski set kompajlira na određeni CPU ili se može direktno implementirati na silicij. [9]

8 http://objetjava.free.fr/JavaSE/Java_Technology-An_early_history.pdf
9 https://docs.oracle.com/javase/specs/jvms/se16/html/jvms-2.html

Java virtualni stroj ne prepoznaje Java jezik niti se Java jezik kompajlira u JVM-u. JVM prepoznaje samo *.class* fileove koji u sebi sadržavaju bajt kod (eng. *bytecode*). Java kod je sadržan u .java datotekama koje se potom pomoću Java kompajlera pretvaraju u bajt kod koji je sadržan u *.class* datotekama.

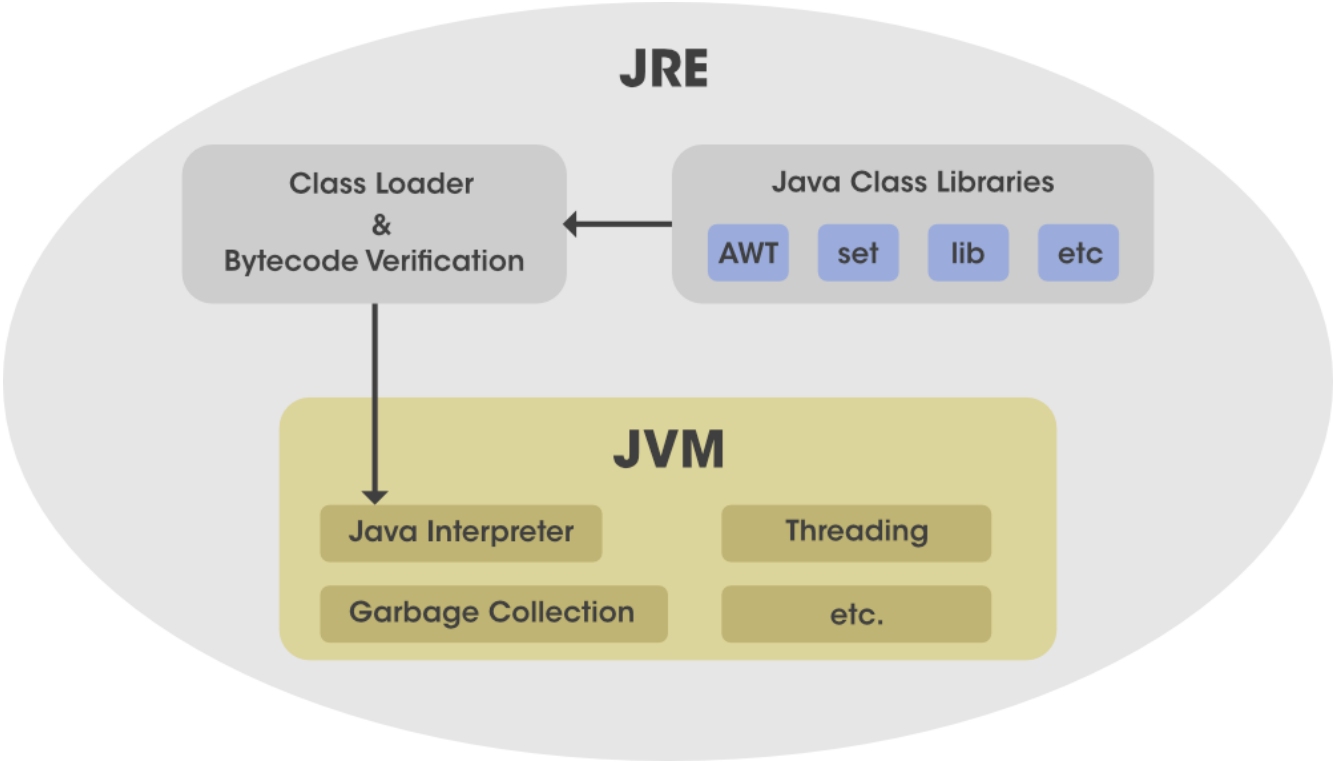
Funkcionalnosti JVM-a su sljedeće:

- Upravljanje memorijom: JVM upravlja dodjeljivanjem i oslobađanjem memorije putem svojih područja za pohranu podataka. Ta područja uključuju područje metoda, stog, stog za *native* metode i stog za pozive. Skupljač smeća (eng. *Garbage collector,* skraćeno GC) automatski upravlja memorijom tako da oslobađa neiskorištene objekte i oslobađa memoriju.
- JIT (*just-in-time*) kompilacija: JVM koristi kombinaciju interpretacije i dinamičke kompilacije za učinkovito izvršavanje Java bajt koda. Bajt kod se prvotno interpretira, a JVM prepoznaje često izvođene dijelove koda za daljnju optimizaciju. JIT kompilator dinamički prevodi te dijelove u stroj u obliku izvornog koda za poboljšanu izvedbu.
- Prenosivost i neovisnost o platformi: jedna od glavnih prednosti JVM-a je njegova neovisnost o platformi. Java kod koji je preveden u bajt kod može se izvršavati na bilo kojem sustavu s kompatibilnim JVM-om, bez obzira na hardver ili operacijski sustav.
- Sigurnost: JVM uključuje razne sigurnosne značajke za zaštitu od zlonamjernog koda. Strogo provjerava bajt kod kako bi osigurao sigurnost tipova i spriječio neovlašten pristup sistemskim resursima. Također, JVM pruža sigurnosni upravitelj koji ima precizne kontrole pristupa.

2.7.2. Java Runtime Environment

Java Runtime Environment (skraćeno JRE) je programsko okruženje koje omogućava izvršavanje programa na operativnim sustavima. Njegova uloga je da se ponaša kao most između Java aplikacije i operativnog sustava.

On se izvršava direktno na operativnom sustavu te u sebi sadrži resurse za Javu. U sebi sadrži instancu JVM-a te u globalu obuhvaća JVM i biblioteke koje omogućavaju rad programa. Učitavatelj klasa (eng. *Class Loader)* učitava klase potrebne za pokretanje, bajt kod verifikator provjerava kod prije nego se učita u *interpretator* ako se pridržava Java specifikacije koji tumači taj bajt kod i učitava ga u memoriju.[10]

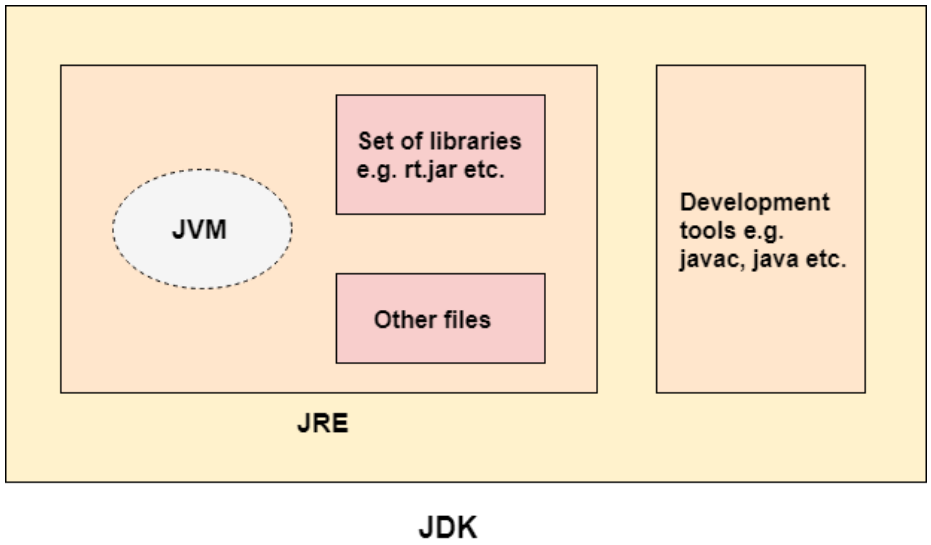


Slika 6: Java Runtime Environment (Izvor: https://www.geeksforgeeks.org/jre-full-form/)

10 https://www.geeksforgeeks.org/jre-full-form/

2.7.3. Java Development Kit

Java Development Kit ili skraćeno JDK je okruženje za razvoj aplikacija u Javi. U sebi sadrži sve potrebne alate za kompajliranje (*javac*), *Java Virtual Machine* i *Java Runtime Environment te* alat za pakiranje programa u .jar datoteke.[11]



Slika 7: Java Development Kit (Izvor: https://www.javatpoint.com/difference-between-jdk-jre-and-jvm)

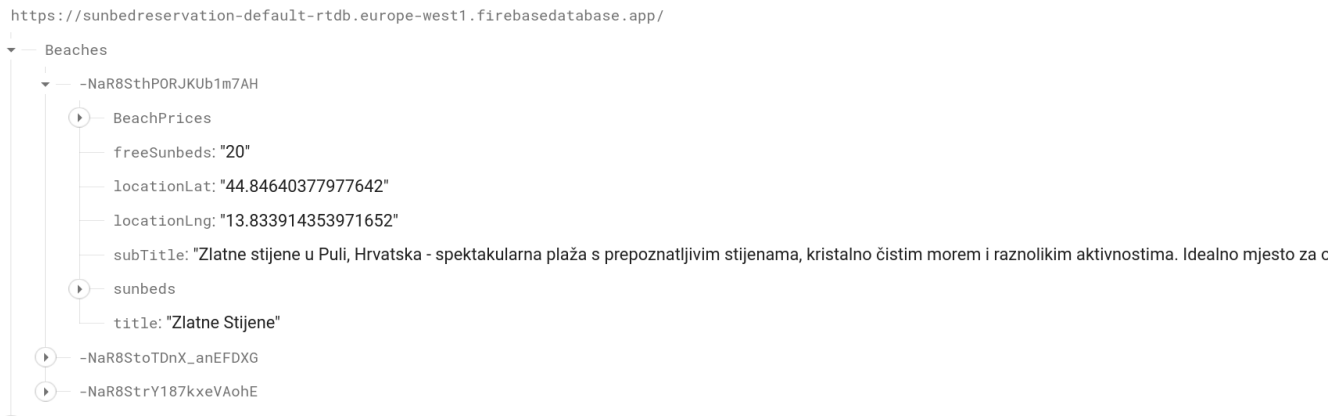
11 https://www.javatpoint.com/difference-between-jdk-jre-and-jvm

2.9 Firebase

Firebase je Google-ova platforma koja nudi alate za izradu aplikacija. Nudi jednostavnu i brzu integraciju sa aplikacijama te nudi direktnu integraciju sa platformama poput Android-a, iOS-a i web aplikacijama.[12]

Neke od usluga koje Firebase nudi su:

- Realtime Database: NoSQL baza podataka bazirana na oblaku, strukturirana kao velika JSON datoteka.
- Cloud Firestore: NoSQL baza podataka bazirana na oblaku, slično kao Realtime Database, razlikuje se u tome što su podaci strukturirani u više dIjelova nazvanih dokumenti (eng. *documents*) umjesto jednog velikog JSON dokumenta.
- Autentifikacija: alat za kreiranje korisničkih računa, developer ne mora razmišljati o kreiranju sustava za registraciju i autentifikaciju korisnika jer već postoji rješenje direktno na servisu.
- Analitika: nudi ugrađene alate za analitiku te mogućnosti mjerenja posebnih podataka, ovisno o potrebama korisnika ili poduzeća. Na primjer mjerenje broja aktivnih korisnika, najprodavanijih proizvoda i slično.



Slika 8: Izgled podataka u Firebase Realtime Database (Izvor: Autor)

12 https://www.geeksforgeeks.org/firebase-introduction/

Neke od glavnih prednosti Firebase-a su:

- besplatni planovi za početnike
- jednostavna integracija sa aplikacijom
- ugrađen sustav za autentifikaciju
- velik izbor usluga, kao što je na primjer Google Analytics.

Također, Firebase ima i svoje nedostatke:

- oslanjanje na Google servise, koji su podložni promjenama
- NoSQL struktura podataka koja nije pogodna za sve sustave.

NoSQL i SQL baze podataka bitno se razlikuju jedne od drugih. Neke od najpoznatijih NoSQL baza podataka su Firebase, MongoDB i ostale. Takve baze podataka koriste nestrukturirani način spremanja podataka.

Firebase sprema podatke u JSON dokument, gdje se ne mora imati predefinirana shema baze te su takve vrste baza pogodne za aplikacije gdje se podaci konstanto mijenjaju, poput društvenih mreža ili web stranica za serviranje sadržaja.

SQL baze podataka spremaju podatke u strukturirane tablice, gdje je potrebno imati predefinirane relacije između tablica. Neke od najpoznatijih SQL baza podatka su PostgreSQL, MySQL, Microsoft SQL Server i ostali. Takve baze podataka su pogodne za sustave gdje je integritet podataka vrlo bitan, poput financijskih sustava.

Za potrebe izrade aplikacije odabrana je Firebase baza podataka, pošto nudi integraciju sa Android aplikacijom te ima ugrađen sustav za registriranje i autentificiranje korisnika.

3. Modeliranje fukcionalnosti aplikacije

Pametni suncobrani su IoT uređaji koji u sebi sadrže broj senzora koji prikupljaju informacije o trenutnom vremenskom stanju poput temperature, vlažnosti, UV indeksa i ostale informacije. Također, mogu imati mogućnost zaključavanja stvari u sef koji je ugrađen u njih.

Opseg aplikacije treba sadržavati klijentski dio, koji uključuje:

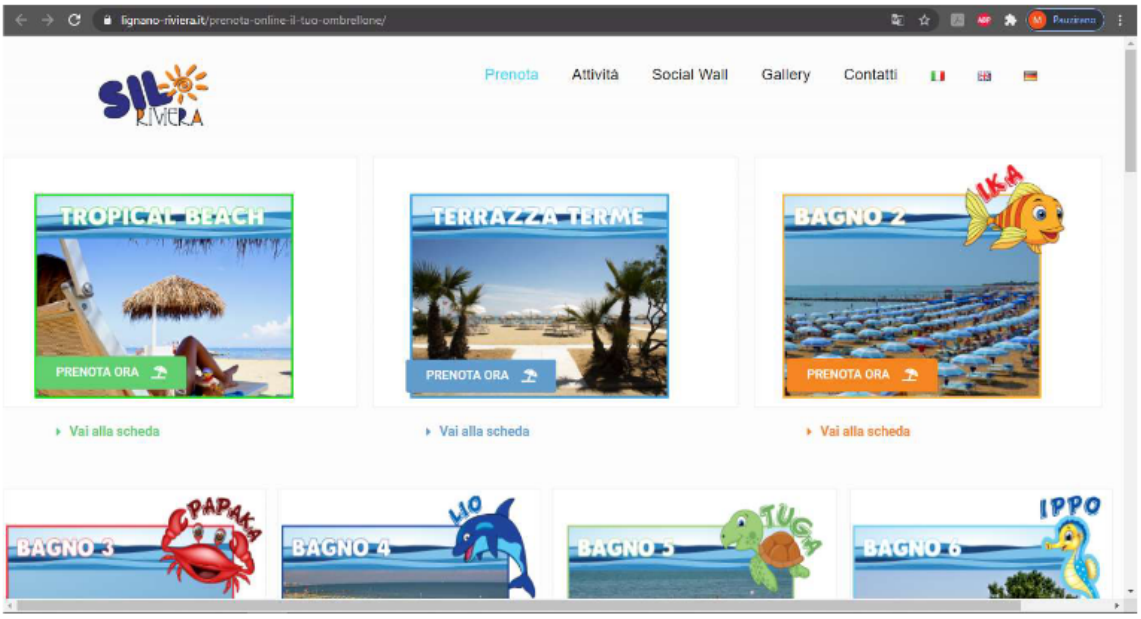
- registracija korisnika
- prijava korisnika
- popis plaža koje imaju suncobrane
- cjenici po plažama – različiti cjenici za različite periode godine
- vremenska prognoza po plažama
- rezervacije i prikaz slobodnih i zauzetih ležaljki
- novosti turističke zajednice

3.1 Istraživanje o postojećim rješenjima

Kao proces izrade aplikacije i prikupljanja informacija, pravljeno je istraživanje o postojećim rješenjima za rezervaciju i naplatu ležaljki na plažama.

Kao primjer uzet je Lignano, jedno od najpopularnijih odmarališta u Italiji, gdje se na njihovim plažama može rezervirati ležaljke i suncobrane.

Resort SIL Riviera u Italiji ima online sustav za rezervaciju ležaljki i suncobrana.[13]



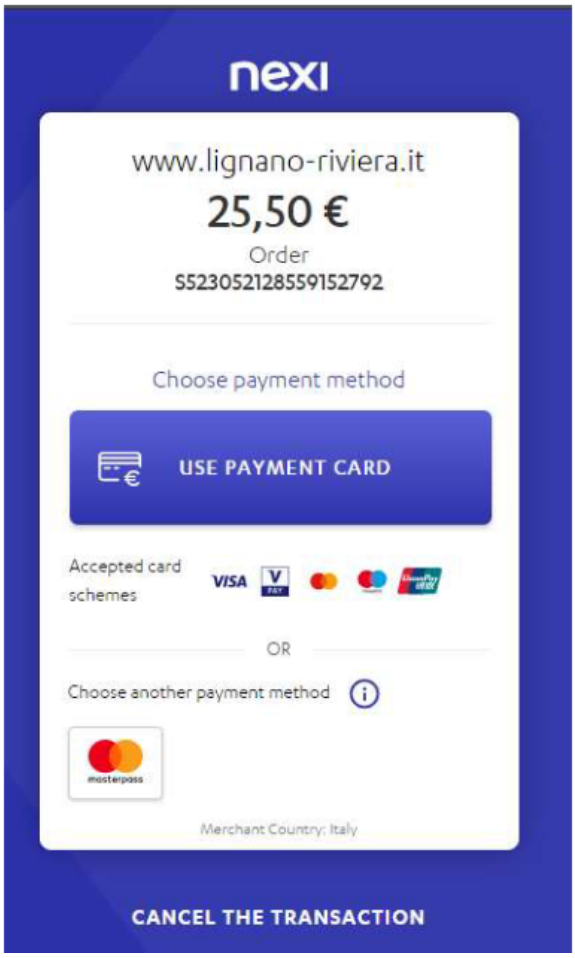
Slika 9: SIL Riviera stranica za rezervaciju suncobrana.(Izvor: Autor)

U ponudi imaju više lokacija plaža, svaka sa svojim nazivom. Svaka plaža ima određeni broj ležaljki te svoje cijene. Nakon izbora plaže, pojavljuje se izbornik s nacrtom plaže i rasporedom ležaljki. Dopuštena je rezervacija do 4 ležaljke i suncobrana, moguć je i izbor perioda te automatski obračunava cijenu za duži period. Redovi bliži moru imaju prosječnu cijenu od 25.50 eura po danu, dok zadnji imaju oko 16 eura po danu, no cijena ovisi i o plaži.

13 https://www.lignano-riviera.it/en/booking/

Slika 10: Menu za rezervaciju ležaljki (Izvor: Autor)

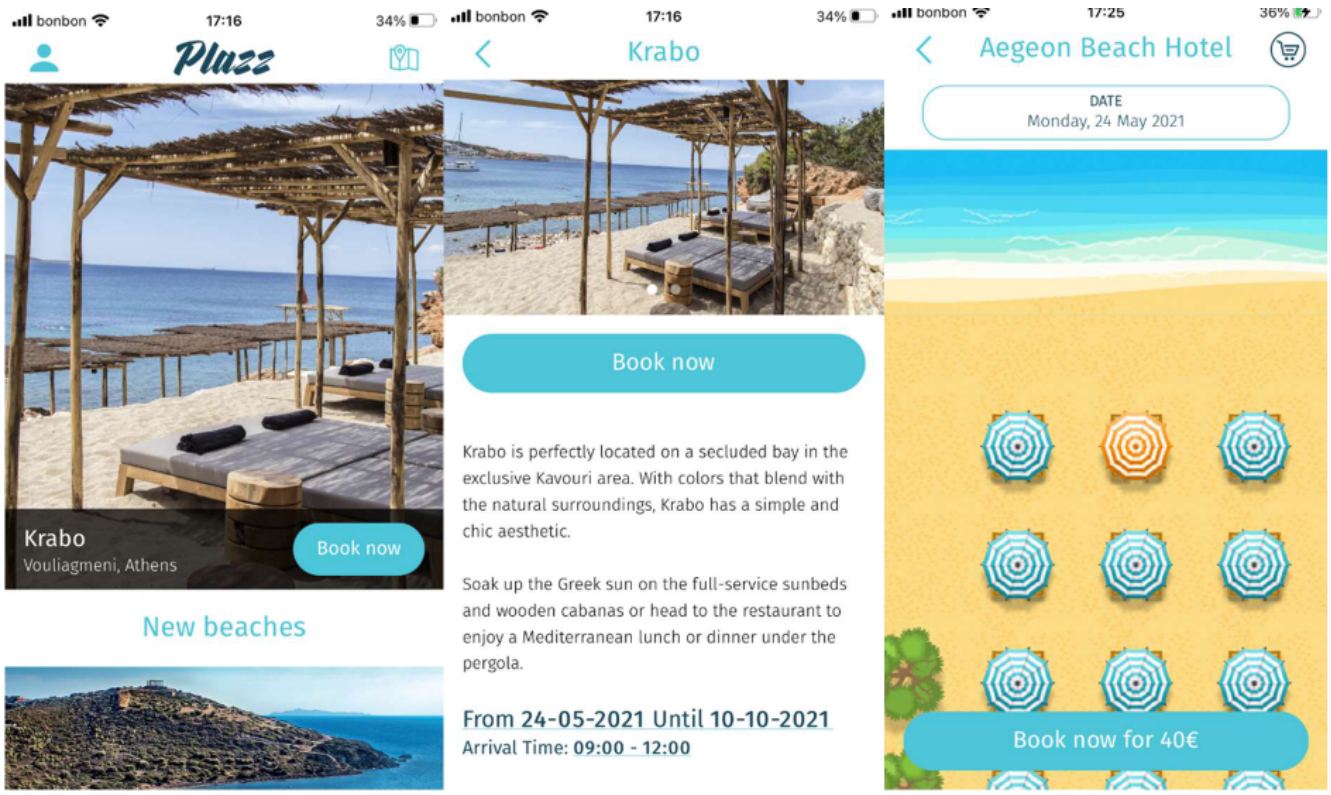
Nakon izbora datuma i ležaljki slijedi obračun troškova gdje je potrebna registracija te nakon toga se šalje na plaćanje.



Slika 11: Sustav za plaćanje (Izvor: Autor)

Na SIL Riviera stranicama postoje uvjeti online rezervacije. Online rezervacija omogućava izbjegavanje redova na plaži, klijent na e-poštu dobiva potvrdu o rezervaciji te tu potvrdu mora imati sa sobom kako bi ju predočio osoblju resorta. Napravljenu rezervaciju nije moguće otkazati, izmjeniti i nije moguća ni djelomična ni potpuna naknada rezervacije, čak ni u uvjetima loših vremenskih uvjeta.

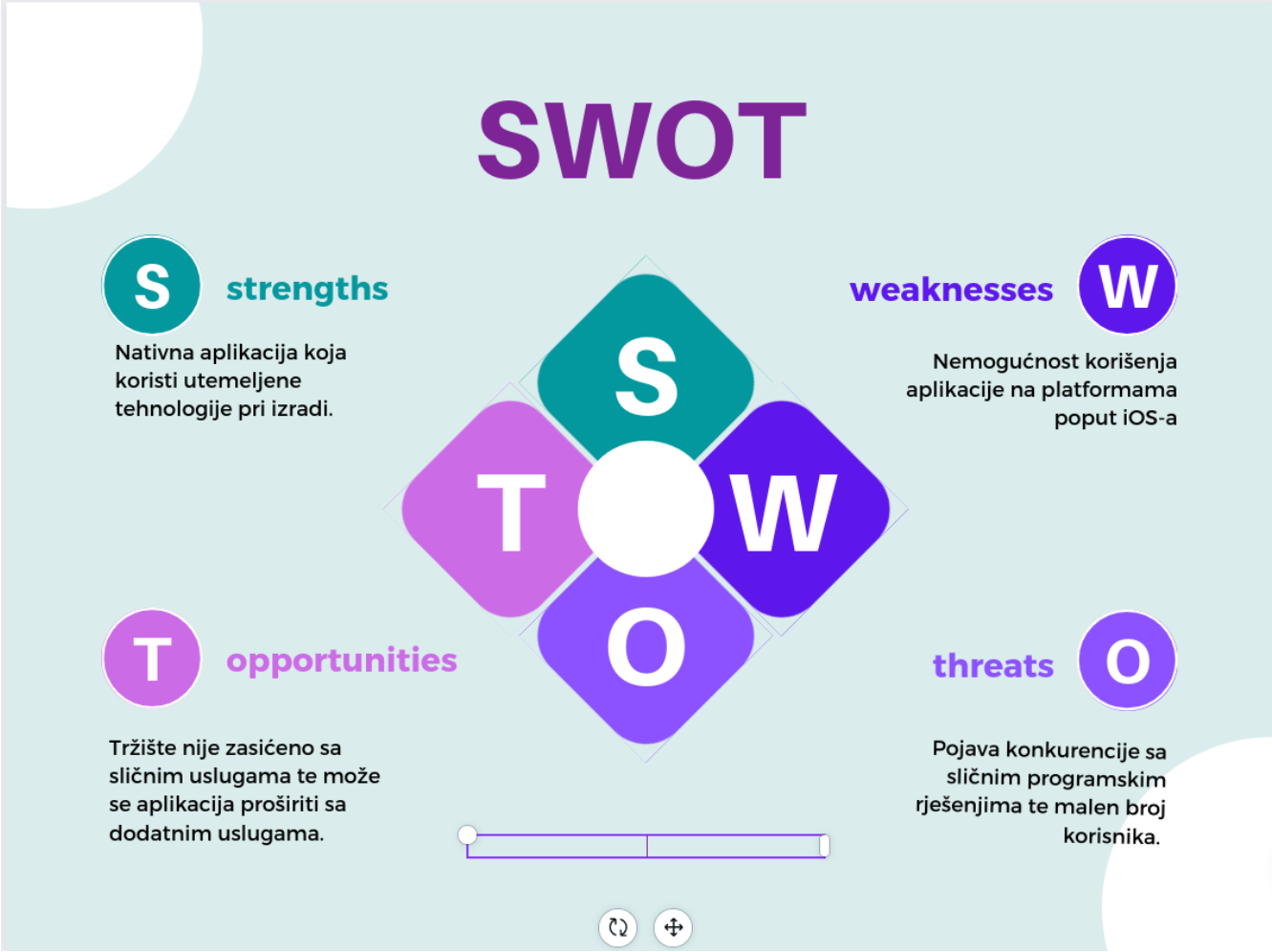
Na AppStore-u za iOS i na Google PlayStore-u za Android, može se naći aplikacija pod imenom Plazz, preko koje se može rezervirati suncobran i ležaljka na plažama u Grčkoj. Također ima izbor plaža na kojima se mogu vršiti rezervacije te opcija izbora ležaljke na plaži.



Slika 12: Korisničko sučelje aplikacije Plazz (Izvor: Autor)

3.2. Motivacija izrade aplikacije

Nakon provedenog istraživanja o postojećim rješenjima te nakon utvrđivanja funckionalnosti, potrebno je izraditi aplikaciju koja nudi rješenje za rezervaciju ležaljki i suncobrana na području Istre.



Slika 13: SWOT analiza izrade aplikacije. (Izvor: Autor)

Pomoću SWOT analize prikazane na gornjoj slici (br. 13) može se utvrditi koje su jačine, slabosti, prilike te prijetnje pri izradi aplikacije.

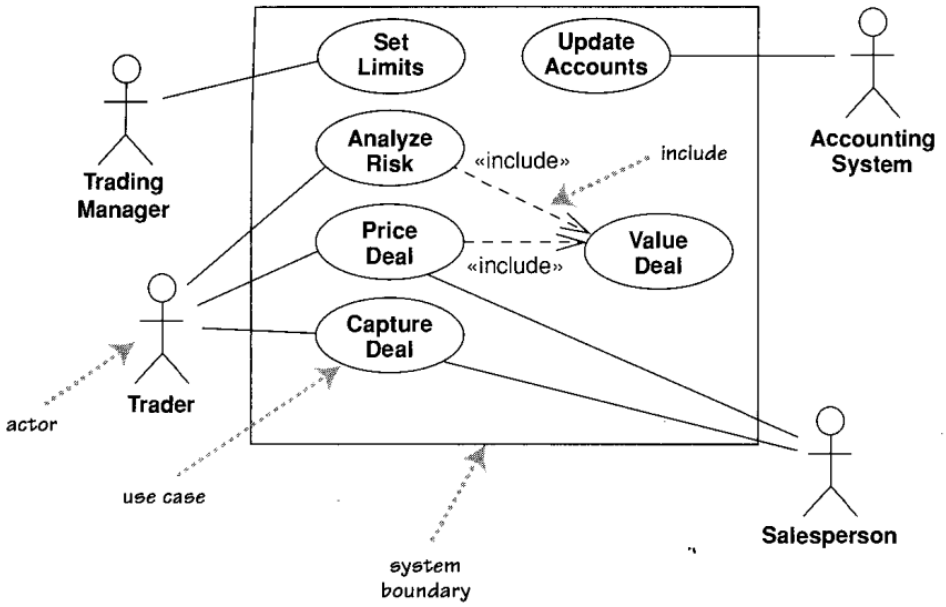
3.3 Dijagram slučaja upotrebe

Dijagrami slučaja upotrebe (eng. *Use-case diagram)* su dijagrami koji se koriste za opisati funkcionalne zahtjeve nekog sustava. Pomoću tih dijagrama opisuje se interakcija korisnika sustava i samog sustava, davajući pritom opis na koji način je sustav korišten. Opisuju se takozvani **scenariji** koji su koraci preko kojih se opisuje interakcija između sustava i korisnika.[14]

Aplikacija za rezervaciju ležaljki ima sljedeći scenarij:

Korisnik se mora registrirati na aplikaciju i nakon toga može se ulogirati u nju. Na zaslonu može birati pomoću mape na kojoj plaži želi rezervirati ležaljke. Nakon toga na odabranoj plaži može birati mjesto na plaži ili više njih koje želi rezervirati i potvrditi rezervaciju. Također, može pregledavati napravljene rezervacije i lokalne vijesti.

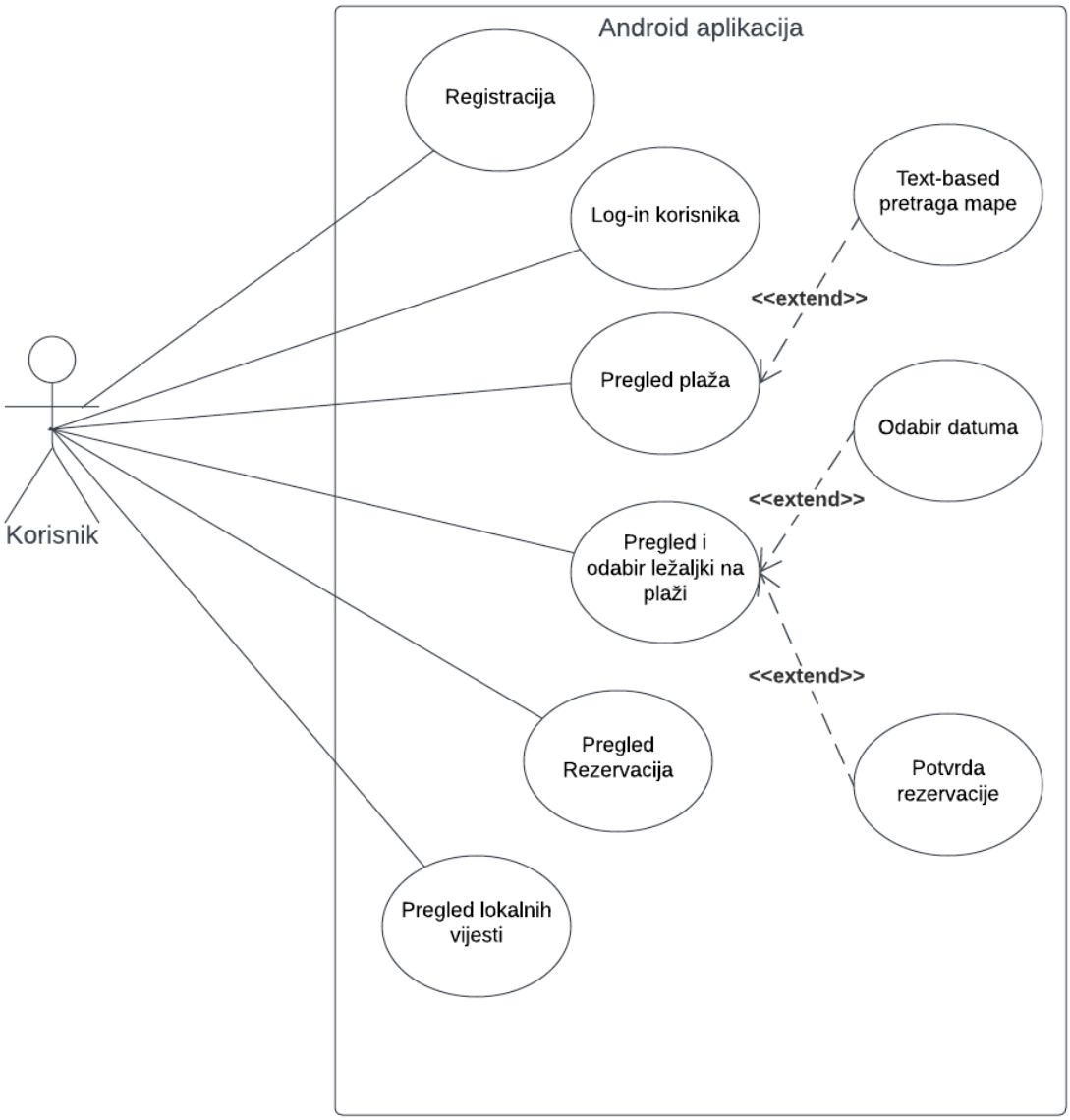
Dijagrams lučaja upotrebe prikazuje koji akteri imaju iterakciju sa sustavom. Čovječuljci prikazuju aktere, pravokutnik prikazuje granice sustava te ovalni krugovi prikazuju scenarije. Između scenarija mogu postojati veze koje mogu uključivati druge scenarije ili proširivati druge scenarije.



Slika 14: Dijelovi use-case dijagrama (Izvor: Fowler, 2003)

14 Fowler M., Scott K., UML Distilled, treće izdanje, Addison-Wesley, 2011.

Dijagram slučaja upotrebe za aplikaciju može se vidjeti na sljedećoj slici. Kao akter je opisan korisnik te kao sustav je opisana Android aplikacija.

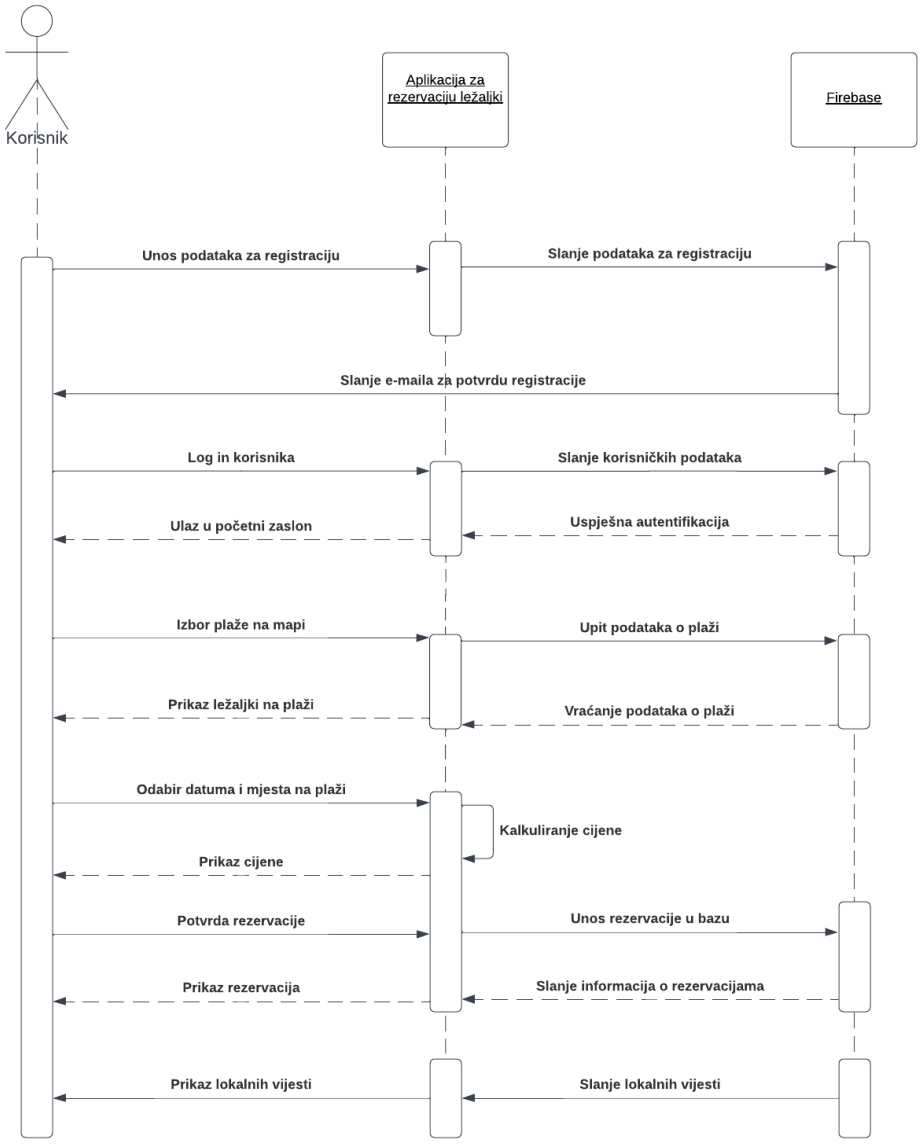


Slika 15: use-case dijagram aplikacije (Izvor: Autor)

`

3.4 Sekvencijalni dijagram

Sekvencijalni dijagrami se koriste kada se želi vidjeti ponašanje nekoliko objekata u nekom određenom scenariju, točnije vidjeti kakvu interakciju ti objekti imaju jedni sa drugima. Opisuju se vremenske sekvence kako objekti vrše interakciju jedni sa drugima u danom sustavu.[15]



Slika 16: Sekvencijalni dijagram aplikacije (Izvor: Autor)

15 Fowler M., Scott K., UML Distilled, treće izdanje, Addison-Wesley, 2011.

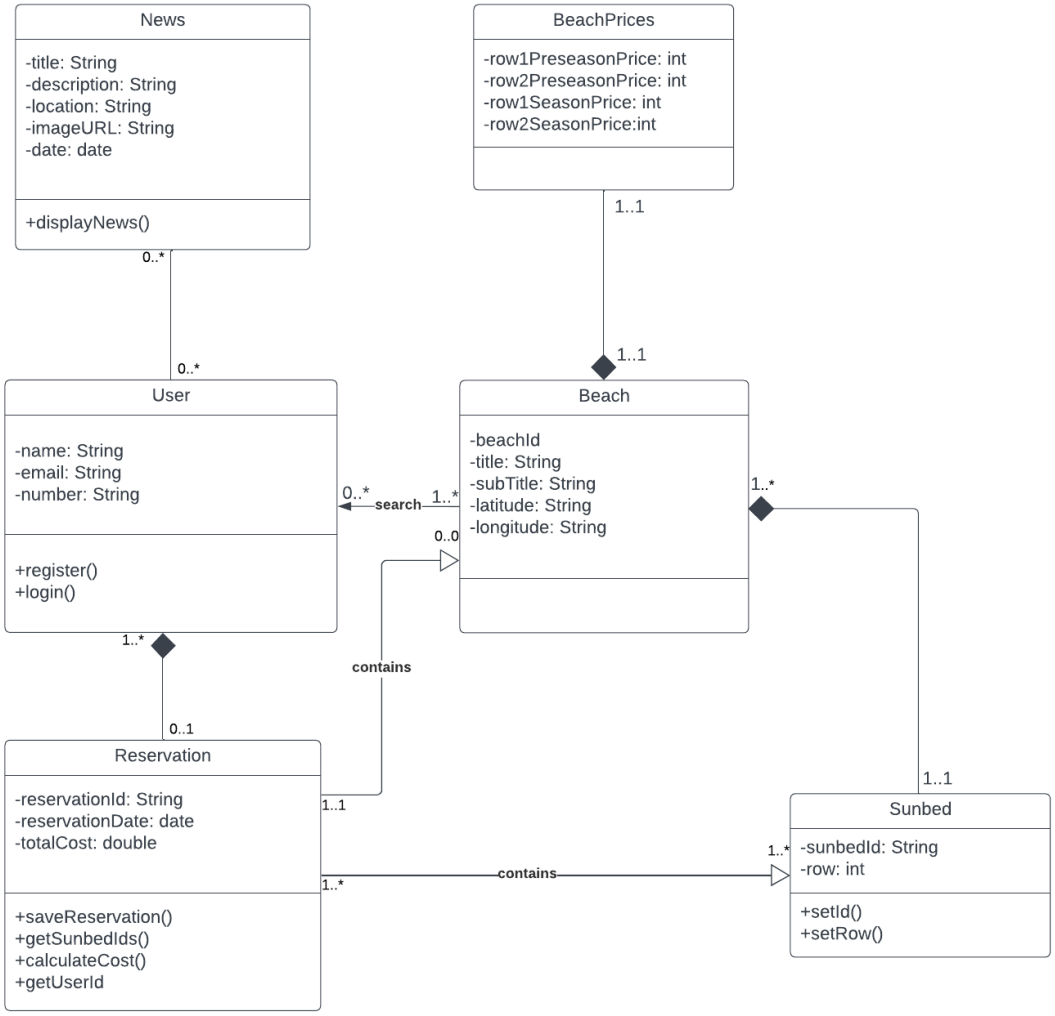
Na slici broj 16 može se vidjeti kako korisnik vrši interakciju sa aplikacijom i kako aplikacija vrši komunikaciju sa bazom. Da bi korisnik mogao koristiti aplikaciju potrebna je registracija gdje korisnik unosi potrebne podatke te aplikacija šalje te podatke u Firebase gdje se korisnik autentificira.

Nakon toga korisnik se može ulogirati u aplikaciju pomoću e-pošte i lozinke te se nakon toga korisnika šalje na početni zaslon gdje mu je prikazana mapa sa označenim plažama. Nakon odabira plaže korisnika se šalje na zaslon gdje su prikazani podaci o plaži i mjesta koje je moguće rezervirati. Korisnik zatim odabire željeni datum kada želi rezervirati i mjesta koja želi rezervirati te nakon pritiska na gumb za potvrdu, prikazuje se cijena i želi li konačno potvrditi rezervaciju. Nakon potvrde rezervacije, korisnik je upućen na zaslon sa rezervacijama.

Lokalne vijesti su spremljene u bazu i šalju se u aplikaciju gdje ih korisnik može pregledavati.

3.5 Klasni dijagram

Klasni dijagrami služe za opisivanje vrsta objekata i njihove statične veze koje se koriste u sustavu. Također pokazuju svojstva i metode objekata te njihova vezana ograničenja. Pravokutnici u dijagramu su klase; prvi dio je ime klase, drugi dio su atributi ili polja te treći dio su metode klase koje pokazuju operacije koje ta klasa može izvršiti.[16]



Slika 17: Klasni dijagram aplikacije (Izvor: Autor)

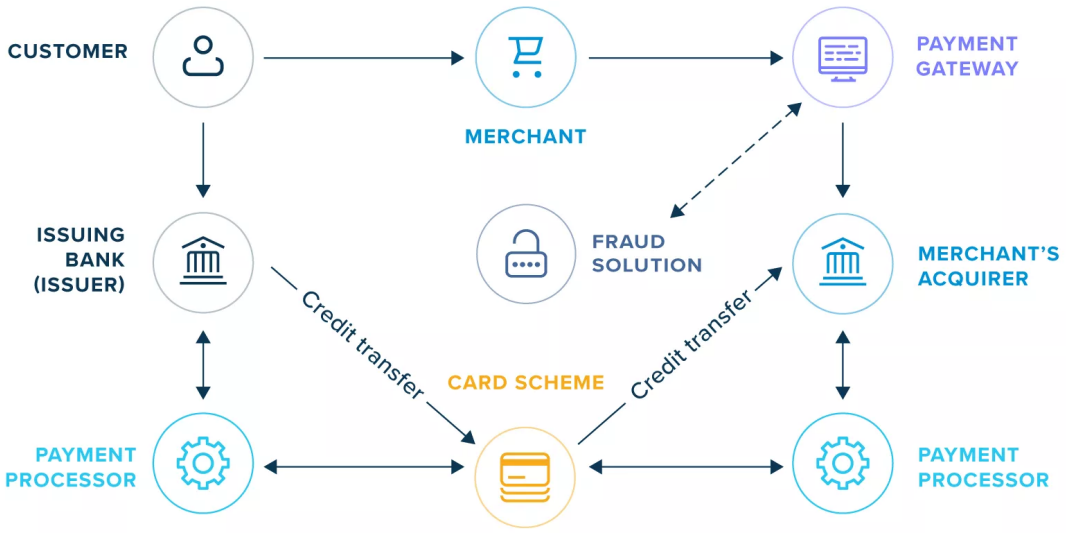
16 Fowler M., Scott K., UML Distilled, treće izdanje, Addison-Wesley, 2011.

Na slici 17. mogu se vidjeti klase i veze aplikacije. Također, dijagram sadrži nekoliko vrsta veza:

- User može pregledavati više klasa Beach
- Beach može imati više klasa Sunbed u sebi i ima jednu instancu klase BeachPrices
- Reservation sadrži atribute klase Beach i Sunbed
- User može pregledavati klasu News

3.7 Pristupnici plaćanja

Pristupnici plaćanja (eng. *payment gateways)* su servisi koji omogućavaju poduzećima sigurnosno procesiranje i menadžment online transakcija. Postoje fizički pristupnici u obliku POS (*Point of sale)* aparata u dućanima i *gateway* portali na raznim web i mobilnim aplikacijama.[17]



Slika 18: Pristupnici plaćanja (Izvor: https://truelayer.com/blog/payments/what-are-payment-gateways-how-do-they-work)

Rade na sljedeći način:

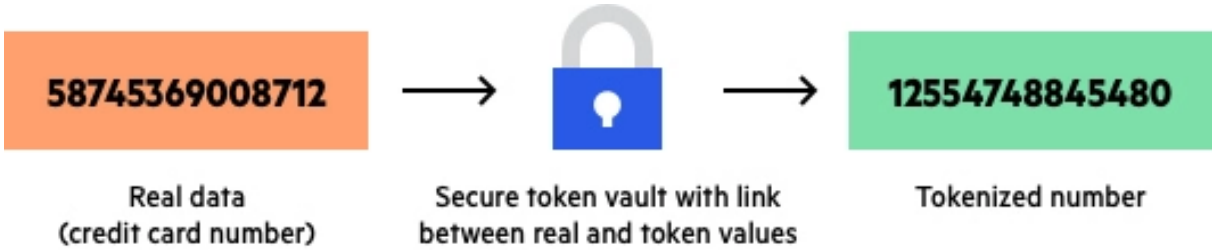
1. Korisnik unosi podatke kartice kod *checkout*-a, gdje forme za unos podataka hosta sam pristupnik ili su na siguran način podaci poslani prema pristupniku.
2. Pristupnik provjerava podatke i vjerodostojnost kartice i šalje podatke prema stjecatelju trgovca.
3. Stjecatelj zatim šalje podatke prema kartičnom procesoru poput MasterCard, Visa ili sličnom te prosljeđuje detalje transakcije prema banci korisnika.
4. Nakon toga banka korisnika šalje transfer novca preko kartičnog procesora.
5. Pristupnik prikazuje korisniku ako je transakcija uspješna.

17 https://truelayer.com/blog/payments/what-are-payment-gateways-how-do-they-work/

Neki od najpoznatijih pristupnika plaćanja su:

- PayPal – jedan od napoznatijih pristupnika plaćanja na svijetu, lagan je za implementaciju te velik broj ljudi ima PayPal račun. Jedna od boljki mu je da za korištenje je potrebna registracija.
- Stripe – namijenjen je velikim poduzećima te nudi asortiman alata kako bi poduzeća mogla prilagoditi plaćanje po potrebi. Podržava jezike poput Ruby, Java, Python i PHP.
- Adyen – koriste ga poduzeća poput eBay, Microsoft, Uber i ostali. Omogućava plaćanje sa više od 150 valuta i 250 vrsta plaćanja te nudi analitičke alate za uvid u ponašanje kupaca.[18]

Jedan od termina koji se spominje pri korištenju pristupnika plaćanja je tokenizacija. Tokenizacija je tehnika zamjene osobnih podataka sa tokenom. To je unikatan *string* koji sam nema nikakvu vrijednost te ne može biti dešifriran, ali prenosi informaciju o podatku bez da kompromitira sigurnost.



Slika 19: Tokenizacija (Izvor: https://www.imperva.com/learn/data-security/tokenization/)

Umjesto da se broj kartice šalje direktno, kreira se token, koji ne može biti probijen jer ne koristi enkripciju i ne mora biti ponovno dekriptiran te samo pristupnici mogu čitati tokene.[19]

18 https://www.techradar.com/best/best-payment-gateways
19 https://www.imperva.com/learn/data-security/tokenization

Pošto je moguće koristiti pristupnike plaćanja samo ako imamo PCI SSC certifikate, u aplikaciji nije moguće izvršiti plaćanje. To je set pravila koje su 2004. godine postavile tvrtke poput Visa, MasterCard, American Express, Discover Financial Services i ostali, kako bi se zaštitile online transakcije protiv krađa.[20]

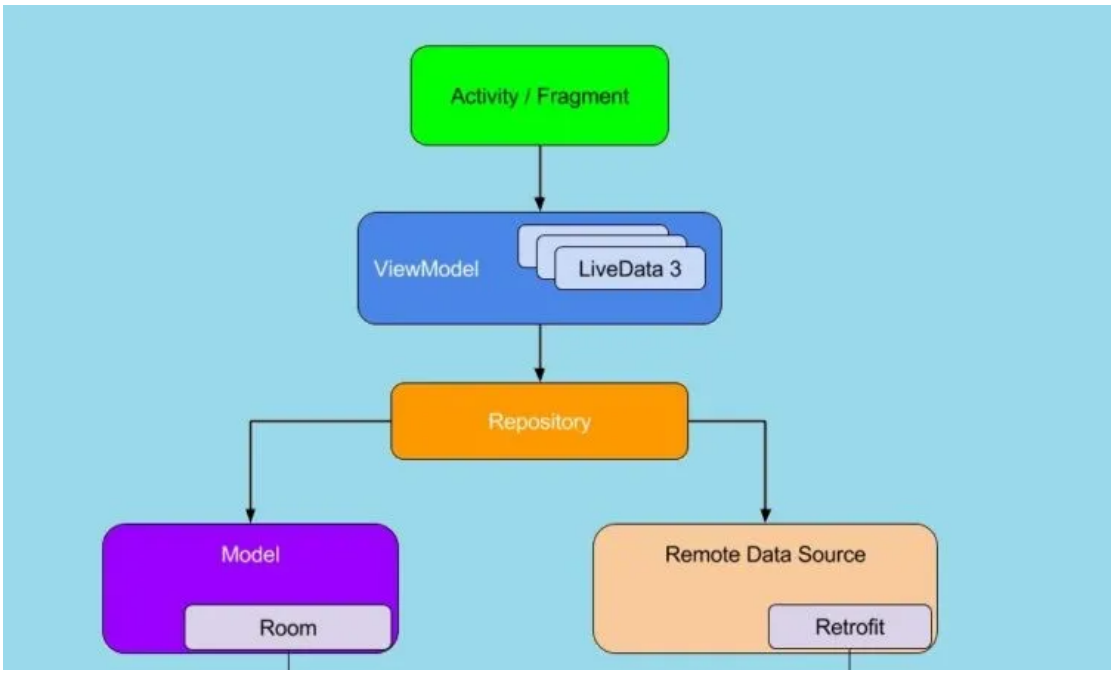
4. Implementacija

Nakon utvrđivanja korisničkih zahtjeva i modeliranja pomoću UML alata, slijedi izrada same aplikacije. U ovom poglavlju opisivat će se arhitektura aplikacije, koje su aplikacijske komponente korištene i bit će prikazani dijelovi koda.

Aplikacija koristi MVVM obrazac. Korištenjem tog obrasca omogućava se odvajanje poslovne logike aplikacije i korisničkog sučelja. Sastoji se od tri glavne komponente: Model, View i ViewModel.

- Model -predstavlja apstrakciju podataka i poslovnu logiku aplikacije.
- View - predstavlja komponentu korisničkog sučelja, korisniku prikazuje podatke i omogućava interakciju sa podacima.
- ViewModel - predstavlja vezu između View-a i Modela. Uzima korisnički unos i ažurira Model. Također prati promjene Modela te po potrebi ažurira korisničko sučelje.[21]

20 https://www.imperva.com/learn/data-security/pci-dss-certification/
21 https://blog.devgenius.io/a-guide-on-mvvm-architecture-in-android-development-3906a6c9bfc8



Slika 20: Prikaz MVVM obrasca (Izvor: https://blog.devgenius.io/a-guide-on-mvvm-architecture-in-android-development-3906a6c9bfc8)

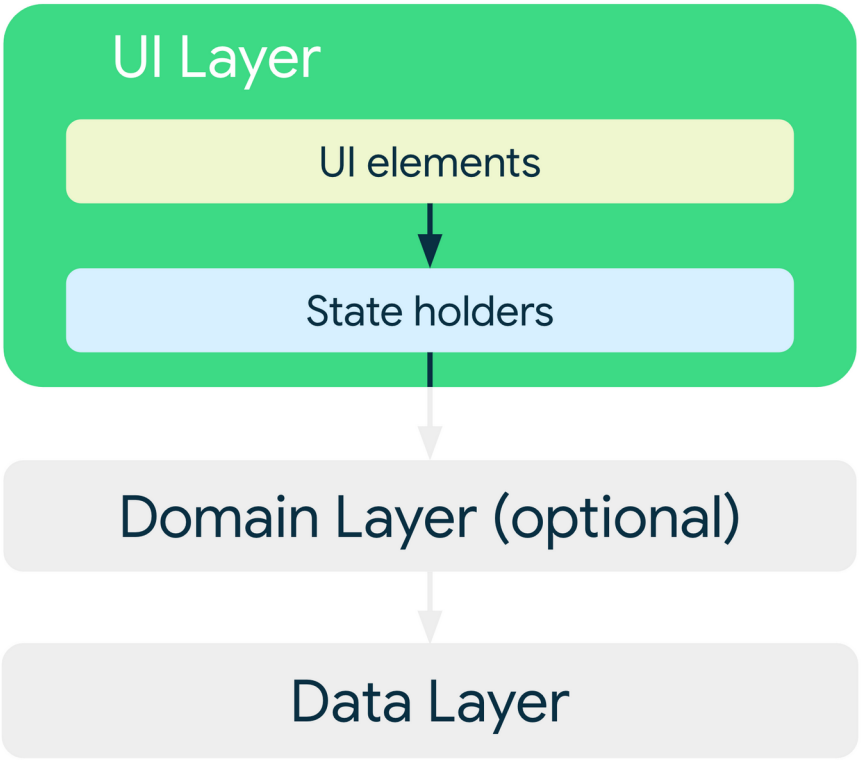
Neke od prednosti MVVM obrasca su[22]:

- ***Separation of concerns:*** razdvajaju se korisničko sučelje, poslovna logika i podaci. To omogućava da se kod lakše održava te modificira po potrebi.
- **Testiranje:** za svaki dio se može posebno pisati *unit* testovi, može se testirati ViewModel i Model neovisno o korisničkom sučelju.
- **Fleksibilnost:** lakše modificiranje korisničkog sučelja neovisno o ViewModel-u i Modelu bez da se aplikacija ne pokreće.
- **Ponovno korištenje:** nije potrebno ponavljati neke dijelove koda.

22 https://blog.devgenius.io/a-guide-on-mvvm-architecture-in-android-development-3906a6c9bfc8

4.1 Arhitektura aplikacije

Aplikacija se sastoji od sloja korisničkog sučelja te podatkovnog sloja. Sloj korisničkog sučelja se sastoji od klasa koje su potrebne za prikaz podataka korisniku i klasa koje upravljaju tim podacima. Podatkovni sloj se sastoji od klasa koje opisuju poslovnu logiku.[23]



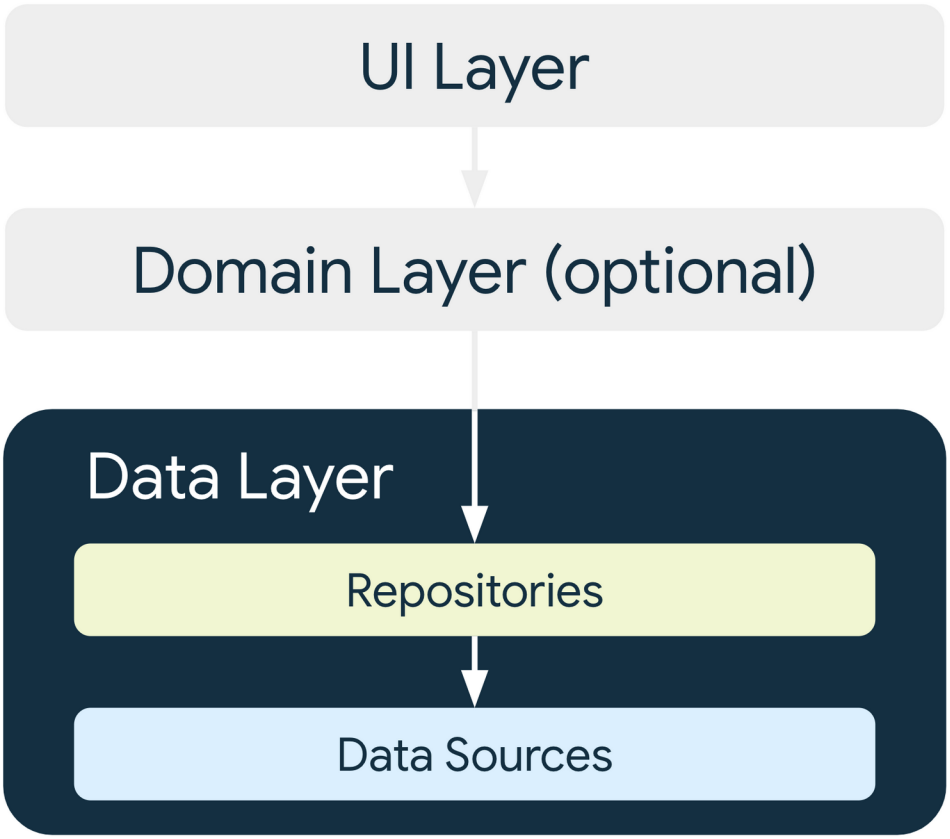
Slika 21: Sloj korisničkog sučelja (Izvor: https://developer.android.com/topic/architecture)

Korisnički sloj se sastoji od:

- elementi sučelja – prikaz podataka na sučelju korištenjem View funkcija.
- *state holders* – koriste se ViewModel klase koje komuniciraju sa podatkovnim slojem, izlažu podatke sučelju.

23 https://developer.android.com/topic/architecture

Podatkovni sloj sadrži klase poslovne logike. Te klase opisuju poslovnu logiku aplikacije, na primjer User klasa opisuje korisnika aplikacije.



Slika 22: Podatkovni sloj (Izvor: https://developer.android.com/topic/architecture)

Podatkovni sloj aplikacije se sastoji od Modela i izvora podataka. Modeli predstavljaju apstrakciju podataka potrebnih za rad aplikacije te izvor podataka je Firebase.[24]

24 https://developer.android.com/topic/architecture

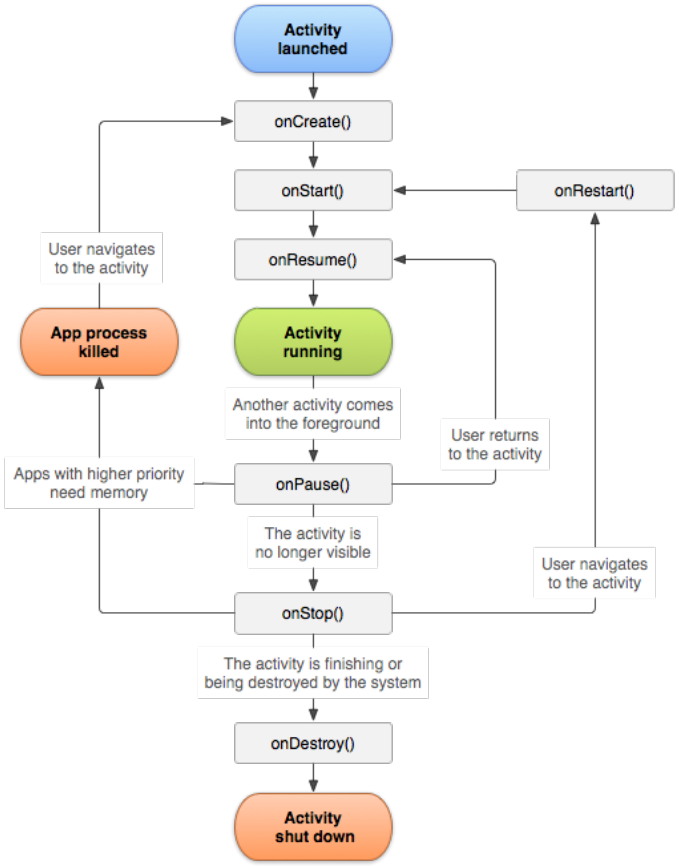
4.2 Android komponente

Aplikacijske komponente su sastavni dijelovi Android aplikacije koje korisnik ili sustav može koristiti za ulaz u aplikaciju. Neke komponente su ovisne o drugima. Postoje četiri tipa komponenti[25]:

Aktivnosti

Komponenta koja služi kao ulazak u sustav za korisnika. Prestavlja jedan zaslon sa korisničkim sučeljem. Na primjer, može predstavljati jedan zaslon koji prikazuje listu e-pošte, jedan zaslon gdje se sastavlja e-pošta te treći zaslon za čitanje e-pošte. Iako rade zajedno u jednom sustavu, svaka aktivnost je zasebna.

Aktivnosti implementiramo kao podklasu *Activity* klase u Android sustavu.



Slika 23: Životni ciklus aktivnosti (Izvor: https://developer.android.com/guide/components/activities/activity-lifecycle)

25 https://developer.android.com/guide/components/fundamentals

Na slici broj 23 može se vidjeti životni ciklus aktivnosti. Kao *callback* methode *Activity* klase, ponuđene su mogućnosti upravljanja ponašanja aktivnosti ovisno kako korisnik ulazi i izlazi iz njih.

Da bi upravljala prijelazima između faza životnog ciklusa aktivnosti, klasa *Activity* pruža osnovni skup od šest *callback* metoda: onCreate(), onStart(), onResume(), onPause(), onStop() i onDestroy(). Sustav poziva svaki od ovih povratnih metoda kako bi aktivnost ušla u novo stanje.

Ovisno o zahtjevima sustava, nije potrebno implementirati sve metode klase *Activity*, no potrebno je znati kako sustav reagira ovisno o stanju aplikacije. Na primjer, sustav će sam osloboditi memoriju ako je korisnik izašao iz aplikacije, ako neka druga aplikacija sa višim prioritetom zahtjeva memoriju.[26]

Servisi

Servisi se koriste za izvođenje aplikacije u pozadini. Oni se izvode u pozadini sustava te nemaju korisničko sučelje. Na primjer, servisi mogu pružati mogućnost izvođenja glazbe u pozadini dok je korisnik u drugoj aplikaciji ili hvatati podatke preko interneta bez da korisniku mora biti blokirana interakcija sa aktivnošću.

Prijemnici emitiranja

Prijemnici emitiranja (eng. *broadcast recievers)* su komponente koje omogućavaju da sustav korisniku javlja događaje. Ne prikazuju korisničko sučelje, ali mogu kreirati notifikacije na statusnoj traci poput alarma, slabe baterije i ostalih. Većinom su korišteni kao ulaz za ostale komponente te imaju minimalan obujam posla.

Pružatelji sadržaja

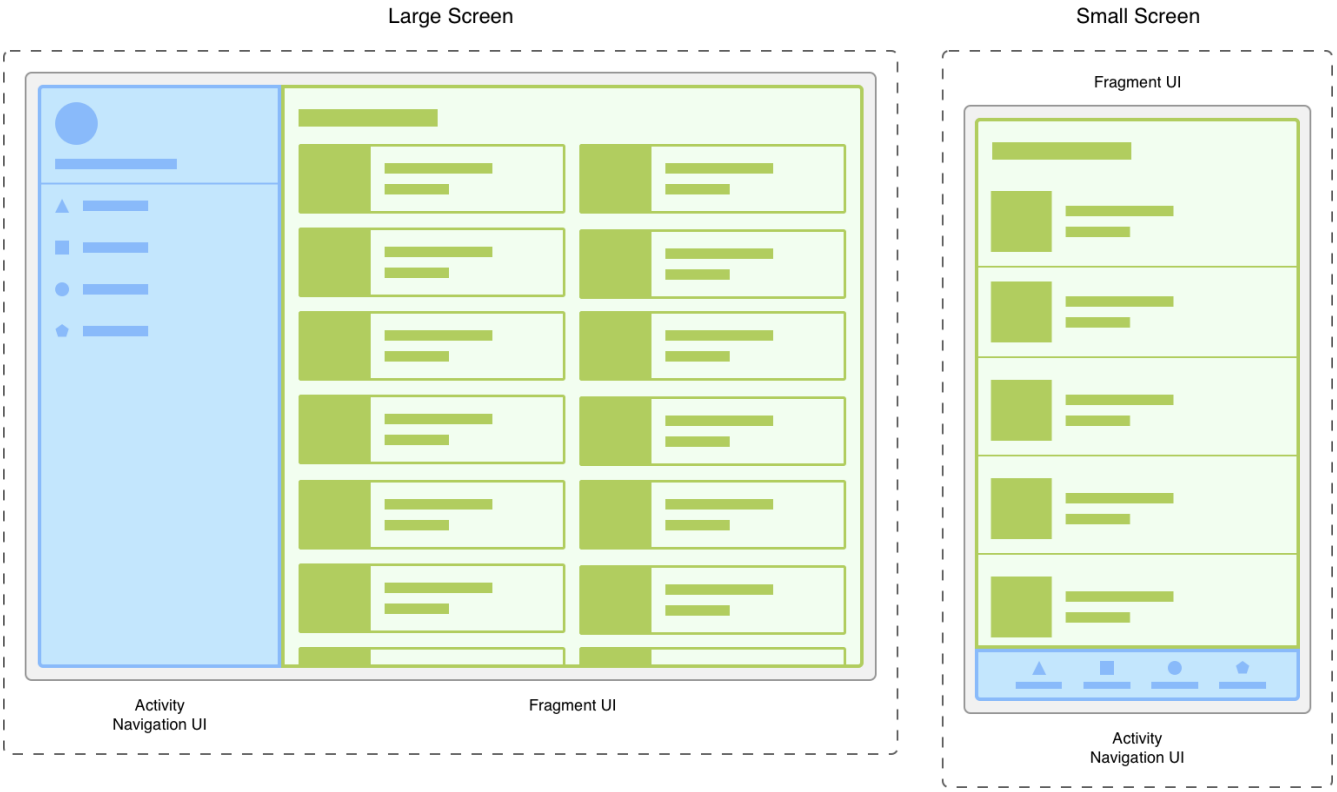
Pružatelji sadržaja se koriste kao kompontente koje mogu dijeliti sadržaj sa ostalim aplikacijama u sustavu. Sadržaj se može nalaziti u SQLite bazi, na internetu ili bilo kojem drugom spremištu podataka. Na primjer ako postoji aplikacija u kojoj treba koristiti kameru, može se koristiti već postojeća rješenja umjesto da se posebno mora raditi sustav za fotografiranje u aplikaciji.[27]

26 https://developer.android.com/guide/components/activities/activity-lifecycle
27 https://developer.android.com/guide/components/fundamentals

Za izradu glavnih komponenti koriste se dodatne aplikacijske komponente:

Fragmenti

Fragmenti predstavljaju dijelove korisničkog sučelja. U aktivnosti se stavljaju globalni dijelovi korisničkog sučelja poput navigacijske trake, dok fragmenti označavaju jedan dio sučelja. Na primjer, u aplikaciji koja koristi velike i male zaslone, na velikim zaslonima navigacijska traka bi se prikazala na lijevom dijelu sučelja u obliku izbornika, dok se na malim zaslonima navigacijska traka nalazi ispod sadržaja.[28]



Slika 24: Primjer fragmenata u sučelju (Izvor: https://developer.android.com/guide/fragments)

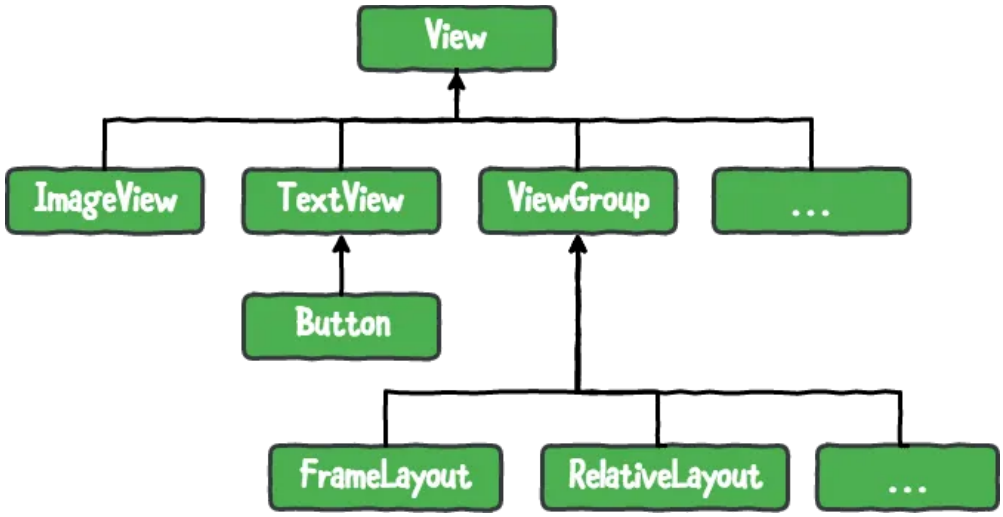
Također fragmenti mogu imati svoj životni ciklus te preko clase *Fragment* mogu koristiti metode onCreate(), onStart(), onResume(), onPause(), onStop() te onDestroy().

28 https://developer.android.com/guide/fragments

View

View su elementi sučelja koji prikazuju neki mali dio aplikacije te mogu reagirati na korisnički unos. Koriste se pomoću klase *View* i neki od elemenata su:[29]

- TextView – prikaz teksta
- EditText – prikaz za unost teksta
- Button – prikaz gumba
- ImageButton – prikaz gumba sa slikom
- DatePicker – prikaz za izbor datuma
- RadioButton – prikaz gumba za odabir
- ImageView – prikaz slike



Slika 25: Hijerarhija View klase (Izvor: https://www.kodeco.com/142-android-custom-view-tutorial)

29 https://medium.com/@huseyinozkoc/android-view-and-viewgroup-6667a20724c5

ViewGroup

ViewGroup je podklasa klase *View* koja se koristi kao nevidljiv spremnik za ostale *View* komponente u sučelju. Na primjer, *LinerLayout* je spremnik koji u sebi sadrži elemente poput Button, TextView i ostalih te također može u sebi sadržavati ostale *layout*-ove. Svaka podklasa ima neke specijalne karakteristike, kao primjer može se uzeti ListView koja služi za prikaz elemenata u listi.[30]

Neke od najkorištenijih *ViewGroup* podklasa su:

- FrameLayout
- ListView
- ScrollView
- GridView
- LinearLayout
- TableLayout

Intent

Intent je pojam koji se koristi za opisivanje operacije koja će se izvršiti. Pruža mogućnost za izvođenje kasnog povezivanja koda u različitim aplikacijama. Najviše se koristi za pokretanje aktivnosti, gdje se mogu različite aktivnosti povezivati jedna sa drugom.[31]

30 https://www.geeksforgeeks.org/difference-between-view-and-viewgroup-in-android/
31 https://developer.android.com/reference/android/content/Intent

Manifest

Manifest je datoteka koju mora sadržavati svaka Android aplikacija. U sebi sadrži informacije koje su potrebne kako bi Android sustav mogao točno pokretati aplikaciju i koja dopuštenja dati aplikaciji.[32]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.application.sunbedreservation">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SunBedReservation">

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyCS8MkQUYFzrCyASjST0ei19Pi0RU3wiSM"/>

        <activity
            android:name=".MainActivity"
            android:exported="false" />
        <activity
            android:name=".ForgotPassword"
            android:exported="false" />
        <activity android:name=".RegistrationActivity" />
        <activity android:name=".LoginActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

32 https://developer.android.com/guide/topics/manifest/manifest-intro

U globalu mora omogućiti sljedeće:

- Komponente aplikacije poput aktivnosti, servisa, prijemnika emitiranja i pružatelja sadržaja. U deklaraciji se odredi naziv klasa svakih komponenti i kako komponente trebaju započeti.
- Dopuštenja koja aplikacija zahtjeva od sustava kako bi mogla koristiti zaštićene dijelove sustava. Na primjer, dopuštenje da se aplikacija spaja na internet.
- Hardverske i softverske značajke koje aplikacija zahtjeva, što utječe na koje se uređaje aplikacija može instalirati. Na primjer, može se odrediti minimalna verzija Androida koju aplikacija dopušta.

Na isječku na stranici 40. može se vidjeti manifest datoteka aplikacije. U njoj su omogućena dopuštenja za pristup internetu te dopuštenja za pristup lokaciji uređaja. Također su u aplikaciji opisane aktivnosti, MainActivity je glavna aktivnost aplikacije u koju se ulazi nakon prijave korisnika i aktivnosti za registraciju, aktivnost za zaboravljenu zaporku te aktivnost za prijavu korisnika.

Također možemo odrediti koju će temu aplikacija koristiti i pomoću meta-data elementa je određen API ključ za korištenje Google Maps.

4.3 Implementacija u Java kodu

U ovom potpoglavlju će se opisati implementacija aplikacije u Java kodu, sa kratkim isječcima koda. Bit će objašnjeni glavni dijelovi aplikacije i kako funkcionira sa bazom podataka.

Android koristi Gradle sustav za izradu aplikacija. Preko njega mogu se koristiti vanjske biblioteke kako bi se ubrzao proces izrade aplikacija. U datoteci *build.gradle* upisujemo *dependencies* koje aplikacija koristi i u njemu se određuje verzija SDK koju aplikacija koristi.

```
dependencies {
    implementation 'com.google.code.gson:gson:2.10.1'
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    implementation 'com.google.firebase:firebase-database:20.0.5'
    implementation 'com.google.android.gms:play-services-location:20.0.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    implementation 'com.google.android.gms:play-services-places:11.0.2'
    implementation 'com.google.code.gson:gson:2.10.1'
    implementation("com.squareup.okhttp3:okhttp:4.10.0")
    implementation 'com.github.smarteist:autoimageslider:1.4.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'com.android.volley:volley:1.2.1'
    implementation 'com.squareup.picasso:picasso:2.71828'
    implementation 'androidx.cardview:cardview:1.0.0'
}
```

4.3.1 Registracija i prijava korisnika

Registracija i prijava korisnika koristi metode FirebaseAuth klase. Metoda za registraciju korisnika je createUserWithEmailAndPassword().

Ispod može se vidjeti isječak koda iz RegistrationActivity.java datoteke u kojoj se nalazi upis korisnika u bazu podataka. Jedna od prednosti korištenja Firebase baze podataka je da se ne mora brinuti o izradi sustava za registraciju i prijavu korrisnika, već sam Firebase nudi u sebi jednostavan sustav. Metodi createUserwithEmailAndPassword() prosljeđujemo dva *string*-a sa e-poštom i lozinkom, koje su prije prosljeđivanja ispitane pomoću *if* blokova kako bi validirali da je korisnik upisao točne podatke sa zahtjevima aplikacije poput dužine lozinke.

```
// Create User in Firebase
mAuth.createUserWithEmailAndPassword(email, password)
      .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
         @Override
         public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
               User user = new User(fullName, email, phone);

               FirebaseDatabase.getInstance().getReference("Users")
                     .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                     .setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                           if (task.isSuccessful()) {
                              Toast.makeText(RegistrationActivity.this, "User has been registered
succesfully!", Toast.LENGTH_LONG).show();
                              progressBar.setVisibility(View.GONE);
                              Log.i("login", "Succesfull");

                              startActivity(new Intent(RegistrationActivity.this, LoginActivity.class));
                              // redirect to login layout!
                           } else {
                              Toast.makeText(RegistrationActivity.this, "Failed to register... Try again!",
Toast.LENGTH_LONG).show();
                              progressBar.setVisibility(View.GONE);
                           }
                        }
                     });
            } else {
               Toast.makeText(RegistrationActivity.this, "Failed to register... Try again!",
Toast.LENGTH_LONG).show();
               progressBar.setVisibility(View.GONE);
            }
         }
      });
```

Nakon toga je moguća prijava korisnika, ako je korisnik potvrdio izrade korisničkog računa preko poveznice u dobivenoj e-pošti. Prijava se izvršava preko metode signInWithEmailAndPassword(), kojoj se također prenose dva *string*-a e-pošta i lozinka te se u metodi provjerava ako je korisnik verificiran. Ako je verificiran, preko *Intent* se otvara MainActivity aktivnost. [33]

```
if (user.isEmailVerified()) {
    //Redirect to user profile
    startActivity(new Intent(LoginActivity.this, MainActivity.class));
} else {
    user.sendEmailVerification();
    Toast.makeText(LoginActivity.this, "Check your email to verify your account!",
Toast.LENGTH_LONG).show();
}
```

4.3.2 Navigacija kroz aplikaciju

Nakon prijave korisnika pokreće se aktivnost MainActivity. U aktivnost se nalazi navigacijska traka i fragment koji se sastoji od *FrameLayouta* naziva frame_layout. Za navigaciju kroz fragmente koristi se FragmentManager.

```
private void replaceFragment(Fragment fragment) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.frame_layout, fragment);
    fragmentTransaction.commit();
}
```

Svaki gumb na traci ima svoj identifikator koji se prenosi u metodu replaceFragment preko *case* bloka nakon što korisnik klikne na gumb te se tako otvara novi fragment u prozoru.

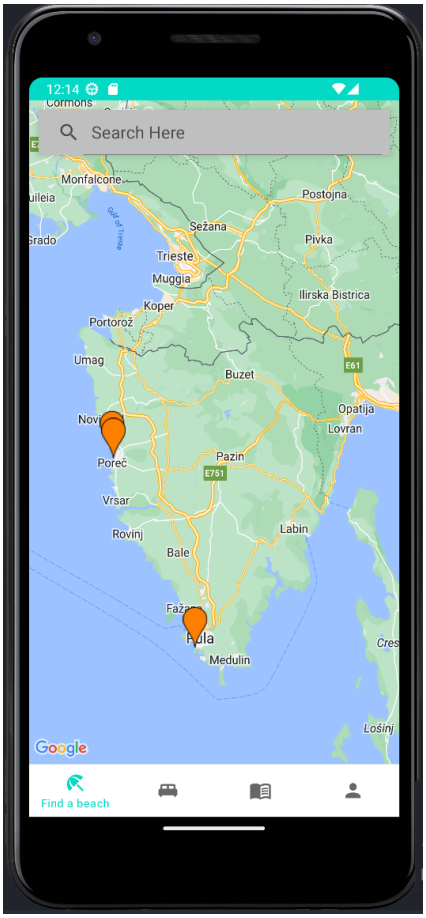
33 https://firebase.google.com/docs/auth/android/password-auth

4.3.3 Izbor plaže kroz Google Maps

Na slici broj 30 prikazan je prvi fragment aplikacije gdje se pomoću Google Maps prikazuje Istarski poluotok i lokacije plaže koje su označene markerima na mapi.

Implementacija je izvršena pomoću Google Maps SDK za Android. To je dio Google Play servisa koji omogućava korištenje Google Maps u aplikaciji gdje se mape prikazuju kao fragment preko XML koda. Neke od značajki su 3D mapa, satelitske snimake, terenske snimake, hibridne mape i ostalo. Moguće je dodavati markere na mapi koji označavaju točke interesa za korisnike i crtanje puteva. Također je moguće upravljati rotacijom, zumiranjem i perspektivom kamere.[34]

Mape su inicijalizirane pomoću SupportMapFragment klase, gdje se dodaje početna pozicija *cp* koja sadrži koordinate Istarskog poluotoka.



Slika 26: Izgled aplikacije sa navigacijskom trakom i fragmentom. (Izvor: Autor)

34 https://developer.android.com/training/maps/maps-and-places

```
FragmentManager fm = getChildFragmentManager();
SupportMapFragment mapFragment = SupportMapFragment.newInstance(new
GoogleMapOptions().camera(cp));
fm.beginTransaction().replace(R.id.map, mapFragment).commit();
//..
mapFragment.getMapAsync(this);
```

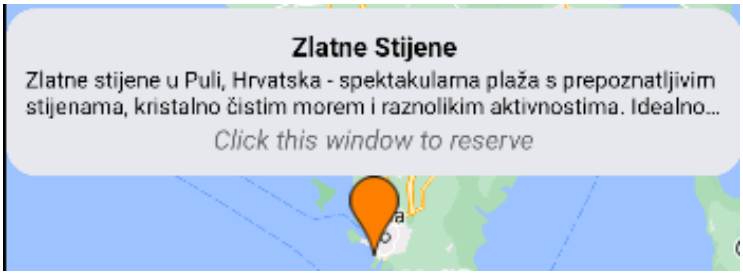
Nakon toga poziva se getMapAsync metoda iz klase SupportMapFragment, sa *this* ključnom riječi preko koje će se pokrenuti metoda onMapReady() kada mapa bude spremna za interakciju.

```
public void onMapReady(GoogleMap googleMap) {
   mMap = googleMap;
   mMap.setInfoWindowAdapter(new CustomInfoWindowForGoogleMap(getContext()));

//...
```

Metoda onMapReady se pokreće nakon što je spremna te inicijalizira instancu GoogleMap klase *mMap* koja omogućava interakciju sa mapom. Također, preko setInfoWindowAdapter postavlja se prilagođen InfoWindow kako bi se moglo upravljati izgledom info prozora kada korisnik klikne na markere na mapi.

Svaka plaža u bazi ima svoje koordinate koje se dohvaćaju iz baze pomoću Firebase Snapshot-a. Preko tih koordinata dodaju se markeri na mapu preko klase MarkerOptions. Pomoću metode addMarker postavljaju se markeri na mapu te se spremaju u mapu markerMap sa identifikatorom plaže i markerom.[35]



Slika 27: Izgled InfoWindow-a (Izvor: Autor)

35 https://firebase.google.com/docs/reference/kotlin/com/google/firebase/database/DataSnapshot

```
for(DataSnapshot beachSnapshot: snapshot.getChildren()) {
   Beach beach = beachSnapshot.getValue(Beach.class);
   String beachId = beachSnapshot.getKey();

   MarkerOptions markerOptions = new MarkerOptions();
   markerOptions.position(new
LatLng(Double.parseDouble(beach.locationLat),Double.parseDouble(beach.locationLng)))
         .title(beach.title)
         .snippet(beach.subTitle)
         .icon(BitmapDescriptorFactory.fromResource(R.drawable.map_pointer));
// ...
Marker marker = mMap.addMarker(markerOptions);
markerMap.put(beachId, marker);
```

Nakon toga za interakciju sa markerom koristi se klasa setOnInfoWindowClickListener, koja postavlja slušač (eng. *listener*) na cijeli InfoWindow. Kada korisnik klikne na prozor, vodi ga na sljedeći fragment za rezervaciju ležaljki na plaži. Preko Bundle klase upisuje se *beachID* plaže kako bi se prenio identifikator u sljedeći fragment koji se pokreće preko FragmentTransaction klase.[36]

```
mMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
   @Override
   public void onInfoWindowClick(Marker marker) {

// ... (ostali kod)

//add arguments
Bundle bundle = new Bundle();
bundle.putString("beach_id", documentId);
Log.i("beachkey", String.valueOf(documentId));
fragment.setArguments(bundle);
```

36 https://developer.android.com/reference/android/os/Bundle

Korisnik također može pretraživati lokaciju pomoću pretraživača, gdje upisuje tekst te se kamera fokusira na lokaciju. Koristi metodu onQueryTextChange koja preko Geocoder klase konvertira ime lokacije u geografske koordinate i kameru postavlja na te koordinate.[37]

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String s) {
        String location = searchView.getQuery().toString();
        List<Address> addressList = null;

        if (location != null || location.equals("")) {
            Geocoder geocoder = new Geocoder(getActivity());
            try {
                addressList = geocoder.getFromLocationName(location, 1);
            } catch (IOException e) {
                e.printStackTrace();
            }
             Address address = addressList.get(0);
            LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 10));
        }
        return false;
    }
```

Geocoder instanca, ako je upisana lokacija pronađena, uzima prvu adresu sa liste i preko metode animateCamera kamera se animirano pomiče na tu adresu.

4.3.4 Fragment rezervacije ležaljki

Fragment SunbedReservationFragment je odgovoran za upravljanje procesom rezervacije ležaljki na plaži. Sastoji se od galerije slika za plažu, vremenskih uvjeta na plaži, odabira datuma rezervacije, izbora ležaljki i potvrde rezervacije.

Galerija slika je implementirana pomoću Android Image Slider biblioteke koja je implementirana preko gradle.build datoteke sljedećom linijom koda:[38]

```
implementation 'com.github.smarteist:autoimageslider:1.4.0'
```

37 https://developer.android.com/reference/android/location/Geocoder
38 https://github.com/smarteist/Android-Image-Slider

Inicijalizira se galerija preko klase SliderView te se predaje XML identifikator image_slider pogledu. Kreiran je SliderAdapter kako bi se predala galerija slika koja je implementirana kao polje slika u kojem su sadržani identifikatori slika koje se nalaze u res/drawable mapi.
Također, može se upravljati ponašanjem galerije, postavljena je vrsta animacije .WORM i efekt kod promjene slika je postavljen kao .DEPTHTRANSFORMATION.

```
SliderView sliderView = view.findViewById(R.id.image_slider);
SliderAdapter sliderAdapter = new SliderAdapter(images);
sliderView.setSliderAdapter(sliderAdapter);
sliderView.setIndicatorAnimation(IndicatorAnimationType.WORM);
sliderView.setSliderTransformAnimation(SliderAnimations.DEPTHTRANSFORMATION);
sliderView.startAutoCycle();
```

SliderAdapter klasa služi za upravljanje sadržajem pogleda za galeriju. Preko konstruktora je predano polje identifikatora slika i pomoću LayoutInflatera pokreće se pogled galerije.

Rezervacija datuma implementirana je pomoću MaterialDatePicker biblioteke. Metoda za prikaz izbornika datuma je showDatePicker() u kojoj se inicijalizira MaterialDatePicker.Builder klasa.[39]

```
MaterialDatePicker.Builder<Long> builder = MaterialDatePicker.Builder.datePicker();
builder.setTitleText("Select Date");
builder.setCalendarConstraints(constraintsBuilder.build());
```

Postavlja se ograničenje za prikaz datuma od današnjeg datuma na buduće datume i postavlja se slušač na gumb: kada korisnik odabere datum, taj datum se sprema u varijablu formattedDate te se na gumbu sprema prikaz odabranog datuma.

```
SunbedReservationFragment.this.selectedDate = formattedDate;

Button selectDateButton = getView().findViewById(R.id.select_date_button);
selectDateButton.setText(formattedDate);
```

Detalji o plaži dohvaćaju se iz Firebase baze podataka pomoću funkcije fetchBeachData iz ViewModel-a. Detalji o plaži sastoje se od imena plaže i kratkog opisa.

```
public void fetchBeachData(String beachId) {
   DatabaseReference beachRef =
FirebaseDatabase.getInstance().getReference("Beaches").child(beachId);
   beachRef.addListenerForSingleValueEvent(new ValueEventListener() {
```

39 https://developer.android.com/reference/com/google/android/material/datepicker/MaterialDatePicker

```
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.exists()) {
            String title = dataSnapshot.child("title").getValue(String.class);
            String subTitle = dataSnapshot.child("subTitle").getValue(String.class);

            titleLiveData.setValue(title);
            subTitleLiveData.setValue(subTitle);
```

Podaci o vremenu se dohvaćaju pomoću WeatherViewModel klase. Pomoću LiveData varijable prate se promjene te se podaci ažuriraju.

```
public LiveData<WeatherData> getWeatherData() { return weatherData; }
```

Podaci se dohvaćaju pomoću metode fetchWeatherData preko API sučelja OpenWeatherMap.

```
public void fetchWeatherData(Context context, double latitude, double longitude) {
    String apiKey = "{API KLJUC}";
    String url = "https://api.openweathermap.org/data/2.5/weather?lat=" + latitude +
            "&lon=" + longitude + "&appid=" + apiKey;
//...
```

Povratno se dobivaju podaci u JSON strukturi i iz tih podataka se izvlače informacije o temperaturi, vlažnosti zraka te naoblaka.[40]

40 https://javadoc.io/doc/com.android.volley/volley/1.1.1/com/android/volley/toolbox/JsonObjectRequest.html

Prikaz ležaljki implementiran je pomoću RecyclerView-a koristeći SunbedAdapter klasu za prikaz i izbor ležaljki. Podaci se dohvaćaju iz Firebase baze podataka preko SunbedReservationViewModel klase. Podaci o ležaljkama se dohvaćaju pomoću metode fetchSunbeds kojoj kao argument predajemo identifikator plaže.

```
for (DataSnapshot childSnapshot : snapshot.child("sunbeds").getChildren()) {
      Sunbed sunbed = childSnapshot.getValue(Sunbed.class);
      if (sunbed != null) {
         int row = sunbed.getRow();
         if (!sunbedMap.containsKey(row)) {
            sunbedMap.put(row, new ArrayList<>());
         }
         sunbedMap.get(row).add(sunbed);
      }
   }
   sunbedRows.setValue(sunbedMap);
}
```

Podaci o ležaljkama organiziraju se u mapu gdje su sortirani po svojim redovima. Prvi i drugi red imaju različite cijene. Nakon toga vrši se ažuriranje LiveData objekta sunbedRows. Ležaljke možemo označiti jer je postavljen slušač na View predmet item_sunbed.xml. Odabrane ležaljke
Svaka ležaljka ima listu datuma u sebi kada je rezervirana te listu datuma može se dobiti pomoću *getter* metode getReservedDates() iz Sunbed objekta.

U SunbedAdapter klasi preko isSunbedReserved() metode vraća se podatak o tome je li napravljena rezervacija na određeni datum ležaljka.

```
private boolean isSunbedReserved(Sunbed sunbed) {
   List<String> reservedDates = sunbed.getReservedDates();
   boolean isReserved = reservedDates != null && reservedDates.contains(selectedDate);
   return isReserved;
}
```

Nakon toga, ovisno o statusu rezervacije ležaljke na određeni datum, mijenja se ikonica te je onemogućen izbor ležaljke.

```
if (isSunbedReserved(sunbed)) {
    holder.sunbedImageView.setImageResource(R.drawable.sunbed_taken);
} else {
    holder.sunbedImageView.setImageResource(R.drawable.sunbed_free);
}
//....
holder.itemView.setClickable(!isSunbedReserved(sunbed));
```

Nakon izbora datuma i izbora ležaljki, korisnik je u mogućnosti potvrditi rezervaciju. Implementirano je preko metode saveReservation() pomoću ViewModel-a.

Preko te metode generira se identifikator rezervacije, sprema se korisnik koji je tu rezervaciju potvrdio, datum, identifikator ležaljki koje su odabrane i cijena rezervacije.

Slika 28: Izgled Reservations dokumenta u Firebase bazi (Izvor: Autor)

Za rezervaciju postoji klasa Reservation, koja sadrži sve potrebne podatke o rezervaciji te getter i setter metode.

```
public class Reservation {
    private String userId;
    private String beachId;
    private String beachTitle;
    private String reservationDate;
    private List<String> sunbedIds;
    private double totalCost;
```

Cijene ležaljki se dohvaćaju iz baze gdje se u poddokumentu BeachPrices nalaze cijene ovisno o trenutnom mjesecu i redu ležaljki.

```
if (row != null) {
    if (row.equals(1)) {
        totalCost.updateAndGet(v -> v + row1price);
    } else if (row.equals(2)) {
        totalCost.updateAndGet(v -> v + row2price);
    }
}

///

updateTotalPrice(totalCost.get());
```

Na kraju se vrši update LiveData objekta preko metode updateTotalPrice();

Poslije toga se korisniku prikazuje poruka s ukupnim iznosom rezervacije i upitom želi li korisnik potvrditi rezervaciju. Ta mogućnost implementirana je pomoću metode showTotalPricePrompt(). Koristi se AlertDialog.Builder klasa za prikaz poruke.[41]

```
AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
builder.setTitle("Confirm Reservation");
```

41 https://developer.android.com/reference/android/app/AlertDialog.Builder

4.3.5 Fragment prikaza rezervacija

Fragment prikaza liste rezervacija je implementiran pomoću ReservationsFragment klase. Lista je prikazana pomoću RecyclerView klase koja sadrži liste rezervacija. Preko klase ReservationsAdapter, popunjava se RecyclerView sa rezervacijama te se podaci dohvaćaju preko ReservationsViewModel klase.

Kako bi se rezervacije prikazivale samo korisniku koji je ulogiran, svaka rezervacija sadrži u sebi identifikator korisnika koji je napravio tu rezervaciju. U metodi getCurrentUserId() povratno se dobiva identifikator trenutno ulogiranog korisnika te se u ViewModel klasi postavlja korisnik s metodom setCurrentUserId() kako bi se mogao točno poslati upit bazi.[42]

```
FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
```

LiveData objekt reservationsLiveData služi za ažuriranje UI komponenti pri promjeni koji sadrži listu rezervacija.

```
private MutableLiveData<List<Reservation>> reservationsLiveData;
```

U metodi loadReservations() dobivaju se podaci iz baze te se filtriraju podataci po korisničkom ID-u. Nakon toga se podaci spremaju u LiveData varijablu reservationsLiveData u obliku liste.

```
DatabaseReference reservationsRef = FirebaseDatabase.getInstance().getReference("Reservations");
Query query = reservationsRef.orderByChild("userId").equalTo(currentUserId);
```

Pomoću onBindViewHolder spajaju se podaci sa View elementima sučelja, koji se dobivaju iz liste reservationsList te poziva metodu bind() za spajanje podataka.

```
void bind(Reservation reservation) {
    beachTitleTextView.setText(reservation.getBeachTitle());
    reservationDateTextView.setText("Reservation date: " + reservation.getReservationDate());

    List<String> sunbedIds = reservation.getSunbedIds();
    String sunbedsText = TextUtils.join(", ", sunbedIds);
    sunbedsTextView.setText("Reserved sunbeds: " + sunbedsText);
    String totalCost = String.valueOf(reservation.getTotalCost());
    totalPriceTextView.setText("Total cost: " + totalCost + "€");

//.....
```

42 https://firebase.google.com/docs/auth/android/manage-users

Za brisanje rezervacije koristi se metoda onDeleteReservationClick u kojoj pozivamo AlertDialog.Builder za prikaz upozorenja.[43]

Za brisanje rezervacija prvo se referencira rezervaciju koju se želi pobrisati preko DatabaseReference te se onda preko metode setValue() postavlja vrijednost na *null* koja briše dokument u bazi.

```
if (reservationId != null) {
    DatabaseReference reservationsRef =
FirebaseDatabase.getInstance().getReference("Reservations");
    reservationsRef.child(reservationId).setValue(null);
}
```

4.3.6 Fragment vijesti

Prikaz vijesti implementiran je pomoću dva fragmenta. Prvi fragment je NewsFragment koji prikazuje listu vijesti, dok je drugi fragment NewsDetailFragment koji sadrži slike i prikazuje detaljno vijesti. Nakon što se klikne na jednu od vijest u prvom fragmentu, otvara se drugi fragment za tu određenu vijest. Preko LinearLayoutManager klase postavlja se upravljač kako bi se podaci prikazivali linearno u RecyclerView-u.

```
LinearLayoutManager layoutManager = new LinearLayoutManager(requireContext());
newsRecyclerView.setLayoutManager(layoutManager);
```

Pomoću NewsAdapter-a spajaju se podaci sa elementima UI sučelja. U unutarnjoj klasi ViewHolder pomoću bind() metode spajaju se podaci te se postavlja slušač za klik na svaki element vijesti.

```
titleTextView.setText(news.getTitle());
```

Pomoću biblioteke Picasso dohvaćaju se slike preko Interneta, dok je poveznica slike sadržana u Firebase dokumentu.[44]

```
Picasso.get().load(news.getImageURL()).placeholder(R.drawable.default_news_image).into(photoIma
geView);
```

43 https://developer.android.com/reference/android/app/AlertDialog.Builder
44 https://square.github.io/picasso/

Nakon klika na element otvara se novi fragment koji sadrži sliku i tekst vijesti. Koristi se FragmentManager za transakciju.[45]

```
FragmentManager fragmentManager = ((AppCompatActivity)
context).getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
fragmentTransaction.replace(R.id.frame_layout, detailFragment);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

Preko Bundle klase šalju se podaci u sljedeći fragment i prikazuju se u fragmentu NewsDetailFragment te se pomoću Picasso biblioteke opet prikazuje slika.[46]

```
Bundle arguments = getArguments();
if (arguments != null) {
   String imageUrl = arguments.getString("image_url");
   String title = arguments.getString("title");
   String description = arguments.getString("description");

   Picasso.get().load(imageUrl).into(photoImageView);
   titleTextView.setText(title);
   descriptionTextView.setText(description);
}
```

Za povratak je postavljen gumb koji preko FragmentManager-a vodi natrag na prijašnji fragment.

4.3.6 Fragment profila

Fragment profila prikazuje korisničke podatke i sadrži gumb za odjavu. Prikazuje se adresa e-pošte, ime korisnika i telefonski broj. Pomoću metode getCurrentUser() dohvaća se trenutno prijavljeni korisnik te se nakon toga dohvaćaju podaci pomoću reference na FirebaseDatabase instancu gdje postoji dokument Users sa podacima.

```
user = FirebaseAuth.getInstance().getCurrentUser();
reference = FirebaseDatabase.getInstance().getReference("Users");
```

45 https://developer.android.com/guide/fragments/fragmentmanager
46 https://developer.android.com/reference/android/os/Bundle

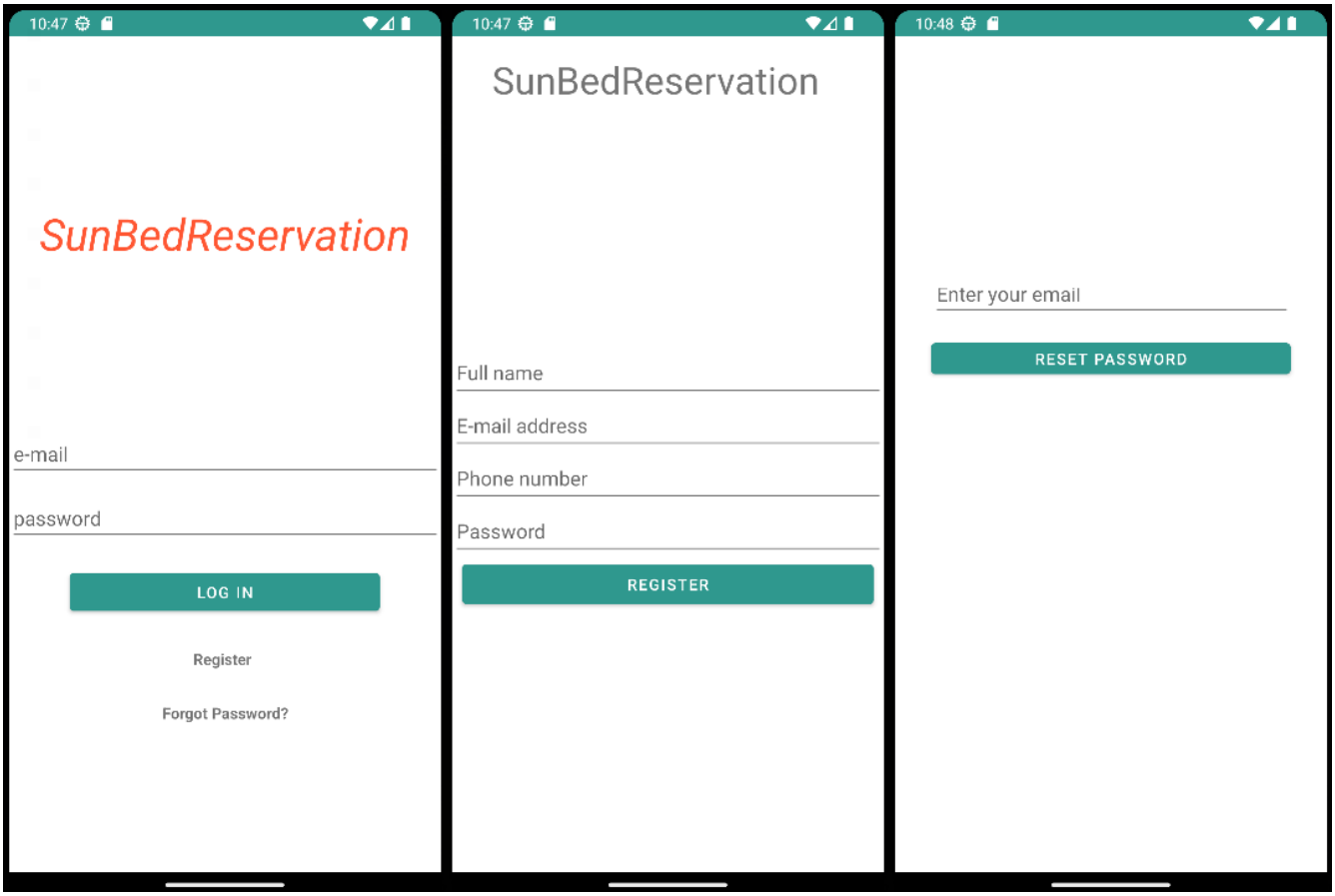
Za odjavu na gumb se postavlja slušač i kad korisnik klikne na gumb pokrene se signOut() metoda prkeo koje se korisnik odjavljuje.[47]

```
FirebaseAuth.getInstance().signOut();
```

47 https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseAuth

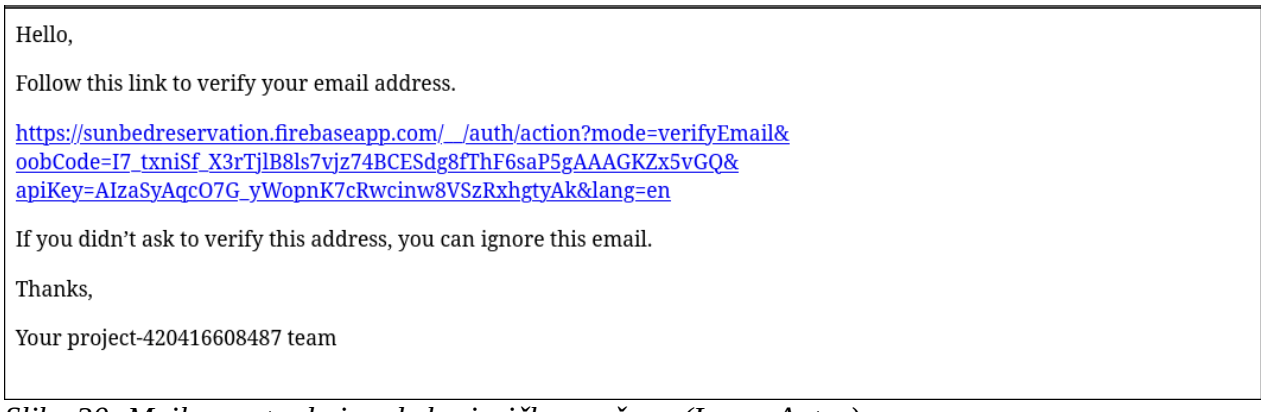
5. Korisničke upute

Prilikom ulaska u aplikaciju pojavljuje se forma za prijavu korisnika. Ako korisnik nema profil, može kliknuti na gumb za registraciju gdje se zatim pojavljuje forma za upis podataka za registriranje. Ako je zaboravljena lozinka, može se korisniku poslati mail za postavljanje nove zaporke.



Slika 29: Registracija i login korisnika (Izvor: Autor)

Nakon registracije se korisniku šalje mail te mora potvrditi svoj mail na poveznici kako bi se zatim mogao prijaviti u aplikaciju.


Hello,

Follow this link to verify your email address.

https://sunbedreservation.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=I7_txniSf_X3rTjlB8ls7vjz74BCESdg8fThF6saP5gAAAGKZx5vGQ&apiKey=AIzaSyAqcO7G_yWopnK7cRwcinw8VSzRxhgtyAk&lang=en

If you didn't ask to verify this address, you can ignore this email.

Thanks,

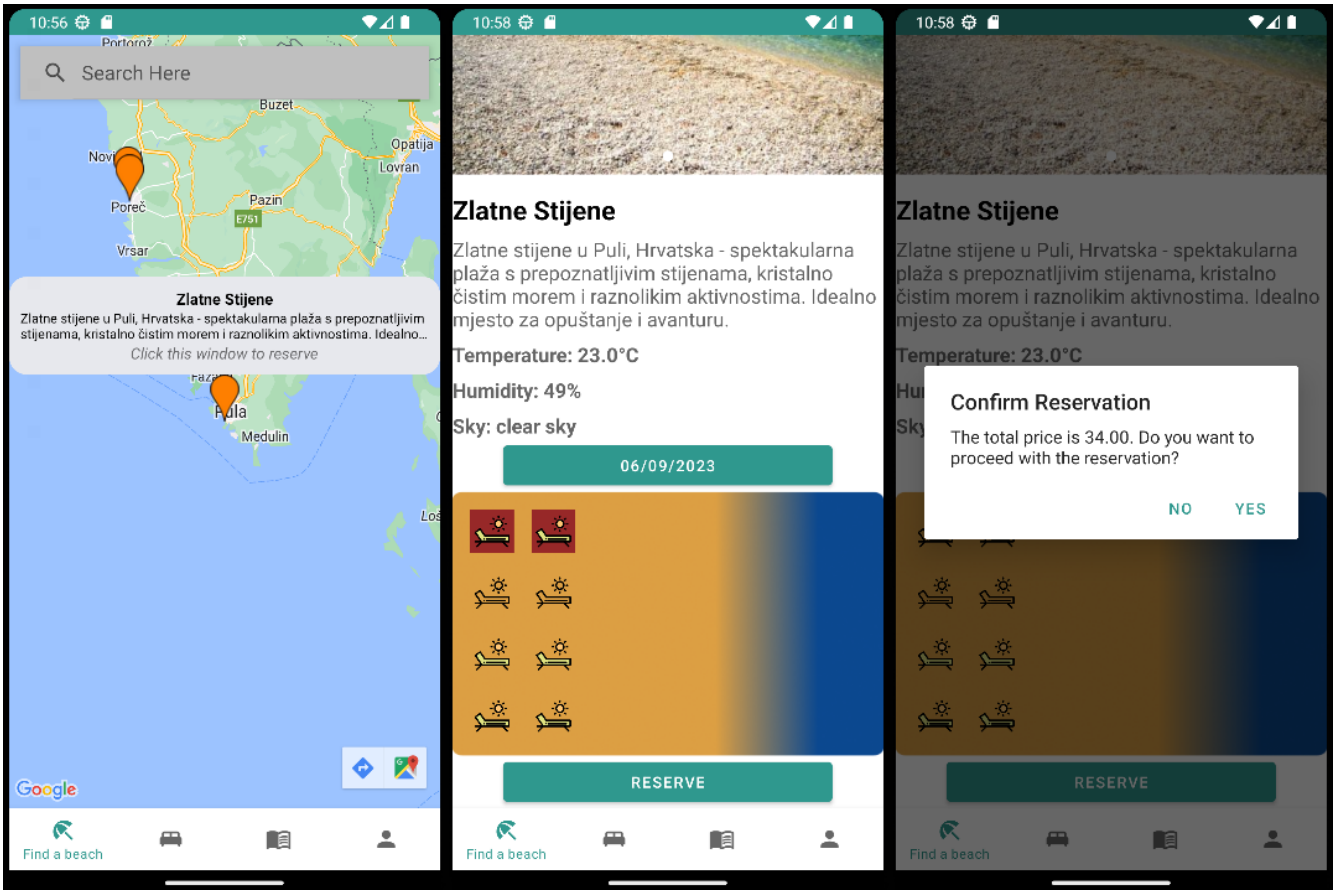
Your project-420416608487 team

Slika 30: Mail za potvrdu izrade korisničkog računa (Izvor: Autor)

Nakon prijave u aplikaciju, na zaslonu se pojavljuje mapa sa markerima plaža. Korisnik može pretraživati mapu pomoću tražilice i kliknuti na marker koji mu prikazuje ime plaže. Nakon klika na informativni prozor, korisnika se vodi na prozor gdje mu se prikazuje galerija slika plaže, ime, detalji i vremenski uvjeti.

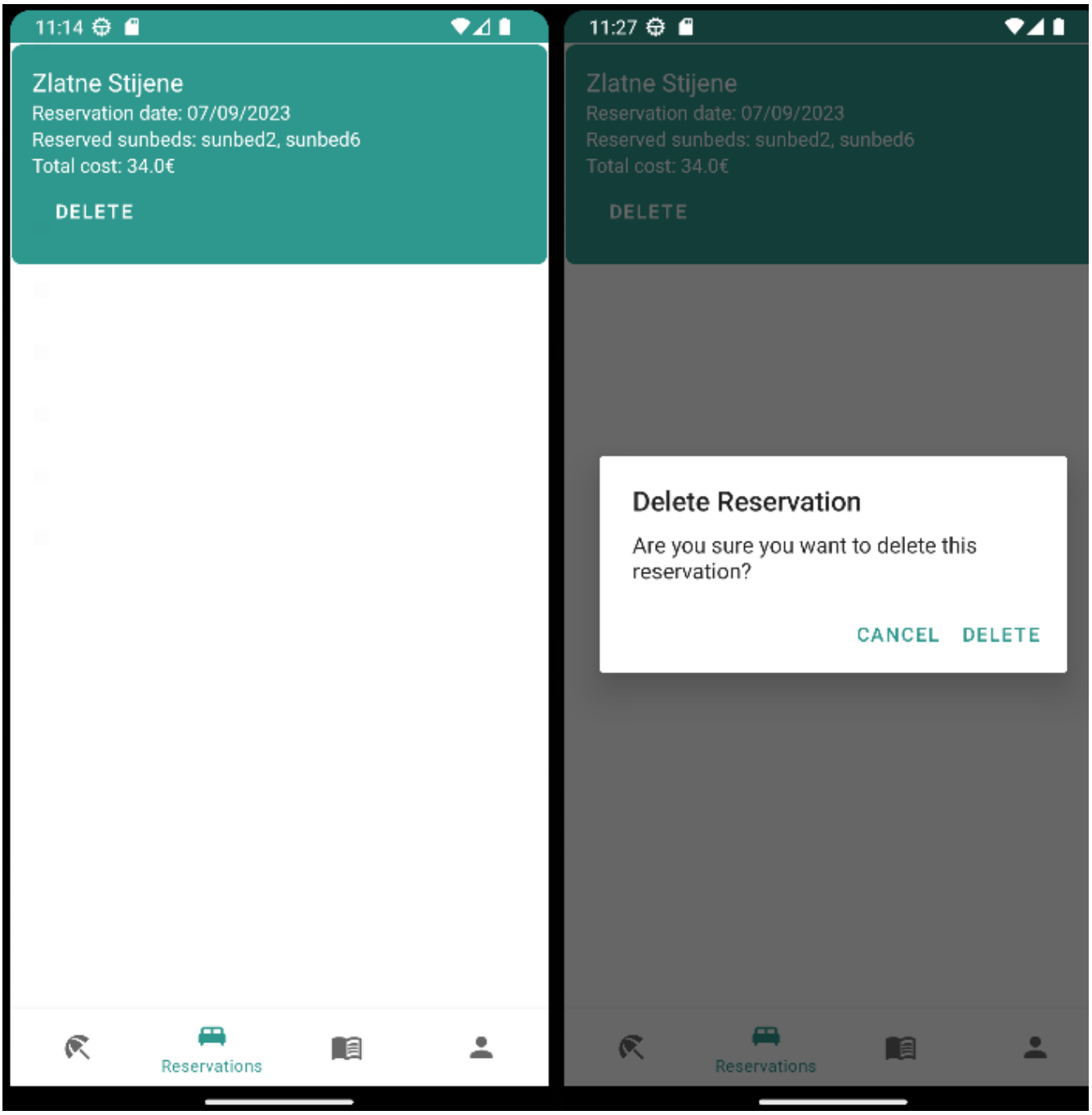
Ispod informativnog dijela, korisniku se nudi izbor datuma i može odabrati koje ležaljke želi rezervirati. Ležaljke koje su na taj dan zauzete ne može odabrati te je ikona zauzetih ležaljki bez ispune boje.

Klikom na gumb za rezervaciju korisniku se prikazuje obavijest s ukupnom cijenom te upitom želi li potvrditi rezervaciju.



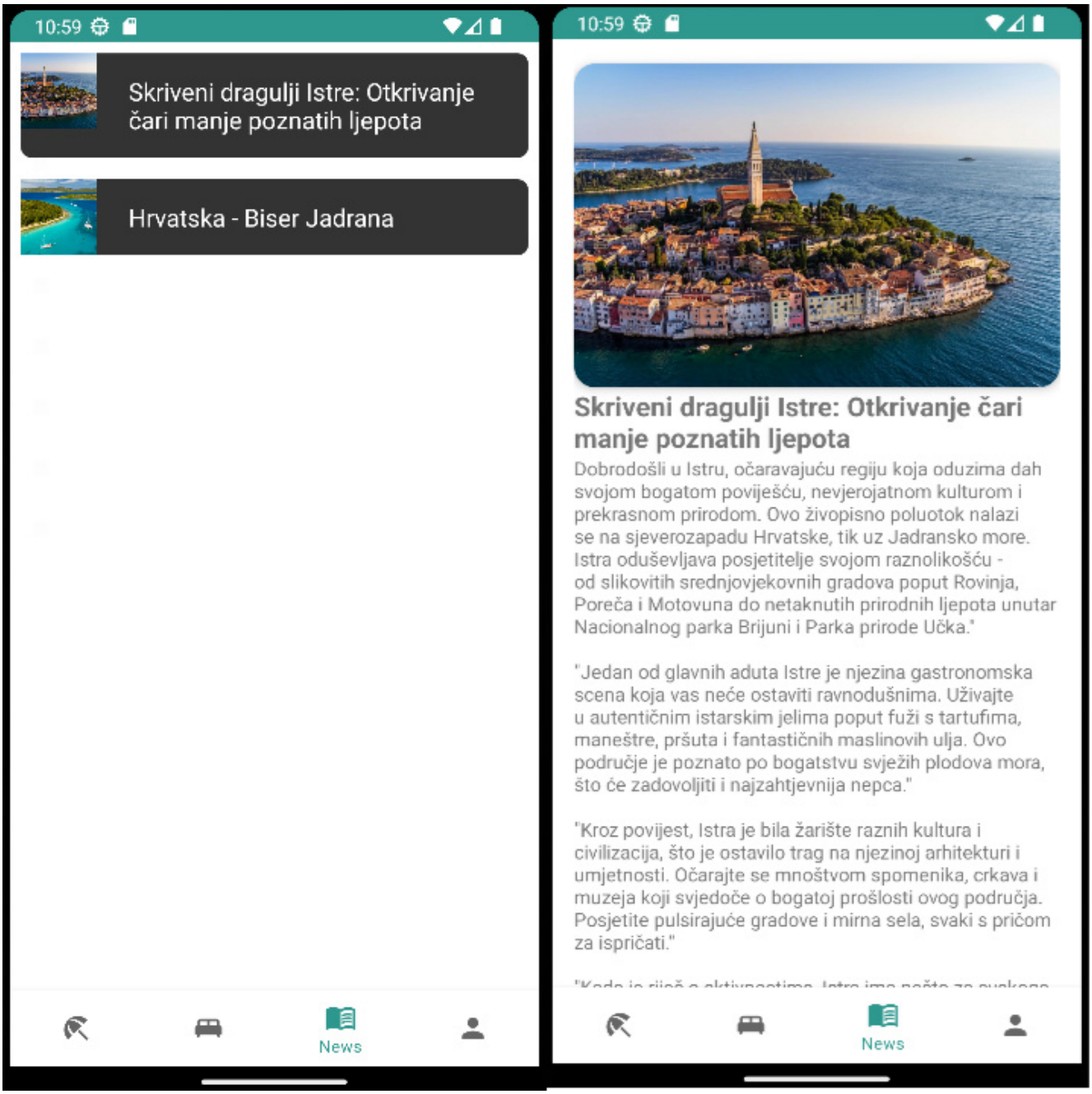
Slika 31: Pretraživanje i rezervacija plaže (Izvor: Autor)

Nakon potvrde, rezervacija se prikazuje na listi rezervacija, na njoj je prikazan naziv plaže, datum rezervacije, koje su ležaljke rezervirane i ukupan iznos rezervacije. Također je moguće rezervaciju izbrisati pritiskom na gumb „Delete”.



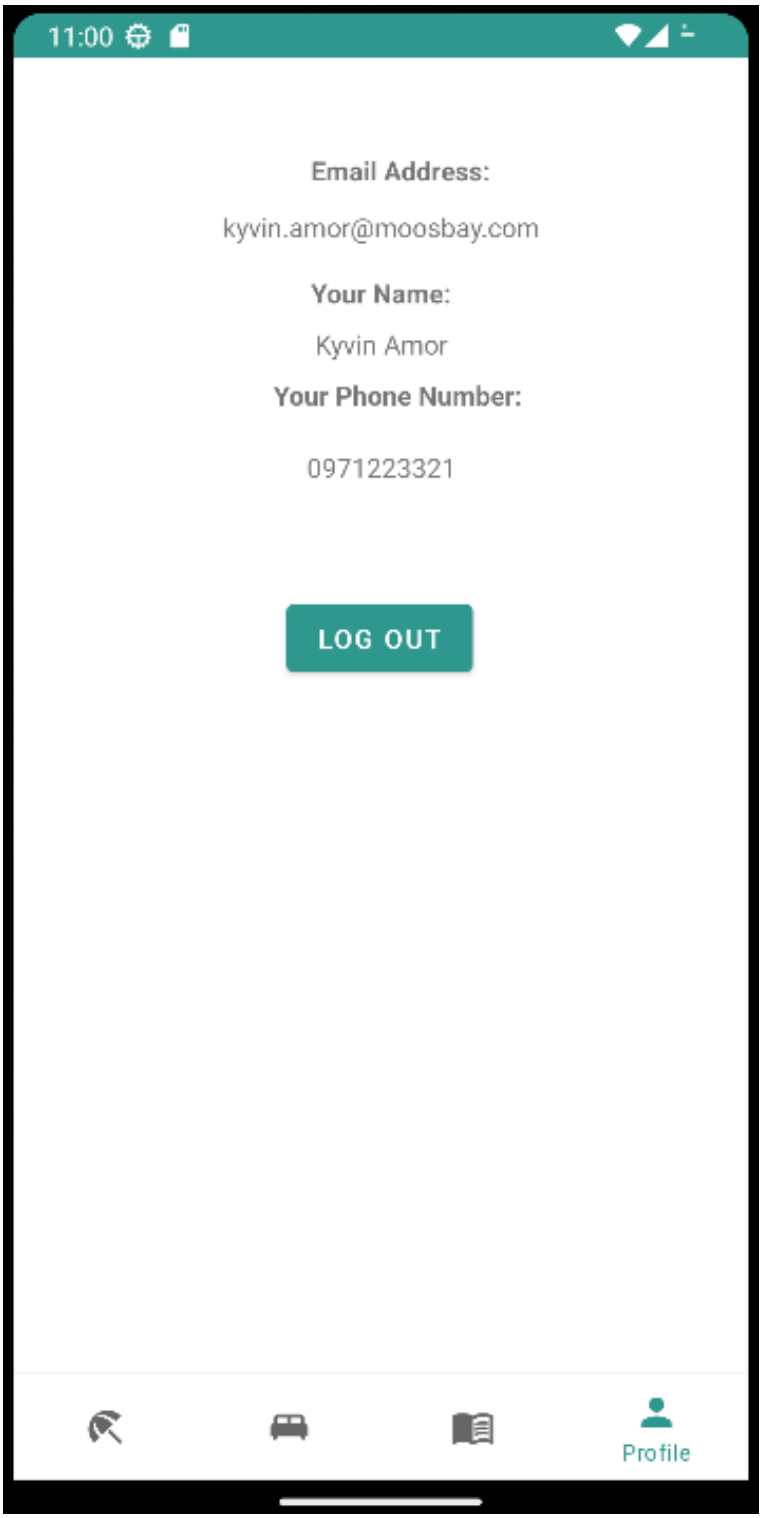
Slika 32: Lista rezervacija (Izvor: Autor)

Pritiskom na gumb za vijesti korisniku je prikazana lista vijesti na koje može kliknuti. Nakon klika, prikazuje se slika, naslov i tekst vijesti. Za povratak na listu može kliknuti na gumb koji se nalazi ispod teksta.



Slika 33: Pregled lokalnih vijesti (Izvor: Autor)

Pritiskom na gumb za profil, prikazuju se korisnički podaci te gumb za odjavu. Pritiskom na gumb se korisnika odjavljuje iz aplikacije i vrati na početni zaslon za prijavu.



Slika 34: Korisnički profil te gumb za logout (Izvor: Autor)

6. Zaključak

Svrha ovog diplomskog rada bila je izrada aplikacije po zadanim zahtjevima gdje se pritom istraživao proces izrade aplikacije te moguća poboljšanja procesa rezervacije i naplate u mobilnom sučelju. Za modeliranje podataka aplikacije korišten je UML, standardni jezik za modeliranje za softvere i razvojne sustave te su izrađeni dijagram slučaja, sekvecnijalni dijagram i dijagram klasa. Provedeno je istraživanje postojećih rješenja za online rezervaciju ležaljki i suncobrana te su uspoređene mogućnosti tih rješenja.

Zahtjevi aplikacije su ispunjeni te aplikacija ima mogućnost registracije i prijave, pregleda plaži, ponudu i pregled više cjenika za različita razdoblja turističke sezone, mogućnost pregleda zauzetosti ležaljki te pregled lokalnih vijesti. Aplikacija koristi MVVM obrazac gdje se u kodu odvaja upravljanje korisničkim sučeljem te poslovna logika.

U radu je opisan na koji način funkcionira online naplata, no tu mogućnost nije bilo moguće implementirati u aplikaciju pošto je potrebno imati bankovne certifikate koje koriste ovlašeni *payment gateway*-i.

Aplikacija bi u budućnosti mogla imati poboljšanja poput automatskih promjena cijena ovisno o postotku bookinga, poboljšanje vizualnog dijela aplikacije, mogućnosti uvođenja naplate i slično.

Iako su moguća različita poboljšanja, može se zaključiti da je sama aplikacija funkcionalna te su zadovoljene osnove za rezervaciju pametnog suncobrana, no za naplatu je potrebno posjedovati posebne certifikate.

Popis slika

Popis Tablica

Literatura

1. 20 Android Statistics In 2023 (Market Share & Users), Daniel Ruby, https://www.demandsage.com/android-statistics/ (06.07.2023.)
2. The history of Android: The evolution of the biggest mobile OS in the world, John Callaham, https://www.androidauthority.com/history-android-os-name-789433/ (06.07.2023.)
3. Android versions: A living history from 1.0 to 14, JR Raphael, https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html?page=2 (07.07.2023)
4. Android, Platform Architecture, https://developer.android.com/guide/platform, https://developer.android.com/guide/platform (08.07.2023.)
5. Android, Application Fundamentals, https://developer.android.com/guide/components/fundamentals?hl=en (09.07.2023.)
6. Kotlin, Kotlin for Android, https://kotlinlang.org/docs/android-overview.html (10.07.2023.)
7. AppsChopper, The Pros and Cons of Cross-Platform Mobile App Development – [2022 Updated], https://www.appschopper.com/blog/pros-cons-cross-platform-mobile-app-development/ (10.07.2023.)
8. Java Technology: An Early History, Jon Boyus, http://objetjava.free.fr/JavaSE/Java_Technology-An_early_history.pdf (10.07.2023.)
9. Oracle, The Structure of the Java Virtual Machine, https://docs.oracle.com/javase/specs/jvms/se16/html/jvms-2.html (12.07.2023.)
10. GeeksForGeeks, JRE Full Form, https://www.geeksforgeeks.org/jre-full-form/ (12.07.2023)
11. JavaTpoint, Difference between JDK, JRE and JVM, https://www.javatpoint.com/difference-between-jdk-jre-and-jvm (14.07.2023.)
12. GeeksForGeeks, Firebase Introduction, https://www.geeksforgeeks.org/firebase-introduction/ (14.07.2023)
13. SIL Riviera, https://www.lignano-riviera.it/en/booking/ (05.05.2022.)
14. Fowler M., Scott K., UML Distilled, treće izdanje, Addison-Wesley, 2011.

15. Truelayer, What are payment gateways and how do they work?, https://truelayer.com/blog/payments/what-are-payment-gateways-how-do-they-work/ (17.07.2023.)
16. Techradar, Best payment gateways of 2023, https://www.techradar.com/best/best-payment-gateways (18.07.2023.)
17. Imperva, Tokenization, https://www.imperva.com/learn/data-security/tokenization(18.07.2023)
18. Imperva, PCI DSS Certification, https://www.imperva.com/learn/data-security/pci-dss-certification/ (18.07.2023)
19. Sharon Rose, A guide on MVVM architecture in Android development, https://blog.devgenius.io/a-guide-on-mvvm-architecture-in-android-development-3906a6c9bfc8 (20.07.2023)
20. Android, Guide to app architecture, https://developer.android.com/topic/architecture (21.07.2023.)
21. Android, The activity lifecycle, https://developer.android.com/guide/components/activities/activity-lifecycle (01.08.2023.)
22. Android, Fragments, https://developer.android.com/guide/fragments (02.08.2023.)
23. Medium, Android View and ViewGroup, https://medium.com/@huseyinozkoc/android-view-and-viewgroup-6667a20724c5 (02.08.2023.)
24. Geeksforgeeks, Difference between View and ViewGroup in Android, https://www.geeksforgeeks.org/difference-between-view-and-viewgroup-in-android/ (02.08.2023.)
25. Android, Intent, https://developer.android.com/reference/android/content/Intent (03.08.2023)
26. Android, App manifest overview, https://developer.android.com/guide/topics/manifest/manifest-intro (04.08.2023.)
27. Firebase, Authenticate with Firebase using Password-Based Accounts on Android, https://firebase.google.com/docs/auth/android/password-auth (06.08.2023.)
28. Android, Maps and places, https://developer.android.com/training/maps/maps-and-places (07.08.2023.)
29. Firebase, DataSnapshot, https://firebase.google.com/docs/reference/kotlin/com/google/firebase/database/DataSnapshot (08.08.2023.)

30. Android, Bundle, https://developer.android.com/reference/android/os/Bundle (09.08.2023.)
31. Android, Geocoder, https://developer.android.com/reference/android/location/Geocoder (11.08.2023.)
32. Github, Android Auto Image Slider, https://github.com/smarteist/Android-Image-Slider (13.08.2023)
33. Android, MaterialDatePicker, https://developer.android.com/reference/com/google/android/material/datepicker/MaterialDatePicker (14.08.2023.)
34. Javadoc, JsonObjectRequest, https://javadoc.io/doc/com.android.volley/volley/1.1.1/com/android/volley/toolbox/JsonObjectRequest.html (15.08.2023)
35. Android, AlertDialog.Builder, https://developer.android.com/reference/android/app/AlertDialog.Builder (16.08.2023.)
36. Firebase, Manage Users in Firebase, https://firebase.google.com/docs/auth/android/manage-users (15.08.2023.)
37. Picasso, https://square.github.io/picasso/ (16.08.2023.)
38. Android, Fragment manager, https://developer.android.com/guide/fragments/fragmentmanager (19.08.2023.)
39. Firebase, FirebaseAuth, https://firebase.google.com/docs/reference/android/com/google/firebase/auth/FirebaseAuth (19.08.2023)

Sažetak

U ovom diplomskom radu izrađen je praktični dio u obliku aplikacije za Android. Funkcija aplikacije je rezervacija ležaljki i pametnih suncobrana. Mogućnosti aplikacije su pretraga plaža na kojima je dostupna rezervacija, odabir datuma rezervacije, pregled zauzetosti mjesta na plaži, pregled potvrđenih rezervacija te čitanje lokalnih vijesti iz turizma. Aplikacija je izrađena u Java programskom jeziku, sa Firebase bazom podataka. Opisan je proces izrade aplikacije te su opisani alati koji su korišteni za izradu.

Ključne riječi: Android, Java, Firebase, Android Studio, rezervacija, ležaljke, pametni suncobrani

Abstract

In this master's thesis, a practical part in the form of an Android application has been developed. The function of the application is the reservation of sunbeds and smart umbrellas. The capabilities of the application include searching for beaches where reservation is available, selecting reservation dates, viewing occupancy status on the beach, reviewing confirmed reservations, and reading local tourism news. The application is developed in the Java programming language, with a Firebase database. The process of developing the application is described, and the tools used for development are outlined.

Keywords: Android, Java, Firebase, Android Studio, reservation, sunbeds, smart umbrellas.