Mobilna aplikacija za rezervaciju kazališnih ulaznica

Klasan, Kristijan

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Kristijan Klasan

MOBILNA APLIKACIJA ZA REZERVACIJU KAZALIŠNIH ULAZNICA

Diplomski rad

Pula, rujan 2021. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Kristijan Klasan

MOBILNA APLIKACIJA ZA REZERVACIJU KAZALIŠNIH ULAZNICA

Diplomski rad

JMBAG: 0303069664, redoviti student
Studijski smjer: Informatika

Predmet: 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 2021. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisan Kristijan Klasan, 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 nedozvoljen 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,_____,_____. godine.

IZJAVA

o korištenju autorskog djela

Ja, Kristijan Klasan dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom Mobilna aplikacija za rezervaciju kazališnih ulaznica 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.

U Puli,_____.

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 5. ožujka 2021.

DIPLOMSKI ZADATAK

Pristupnik:	**Klasan Kristijan (0303069664)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Mobilna aplikacija za rezervaciju kazališnih ulaznica**
Naslov (eng.):	Mobile application for booking theater tickets
Opis zadatka:	Zadatak je izraditi mobilnu aplikaciju za rezervaciju kazališnih ulaznica, te rezervaciju sjedala u kazališno-koncertnoj dvorani. Cilj izrade aplikacije je poboljšati dosadašnju online rezervaciju karata tako da mobilnu aplikaciju mogu koristiti korisnici i administratori. U korisničkom dijelu, korisnik treba izvršiti prijavu ili izradu novog računa. Korisnički dio omogućuje: rezervaciju kazališnih ponuda, pregled i upravljanje rezerviranim kartama, upravljanje korisničkim računom, te upute o korištenju. Administratorski dio omogućuje unos i upravljanje ponudama (predstavama), te prikaz rezerviranih karata za određenu ponudu. Aplikacija se treba temeljiti na tehnologijama: Android operacijski sustav, programski jezik Java/Kotlin, SQL baza podataka, Node.js za *backend*. Istražiti način rada operacijskog sustava Android te razlike u odnosu na operacijski sustav iOS. Proučiti neke od sličnih postojećih aplikacija. Opisati razvijeni sustav (korisničke scenarije, funkcionalnosti, izraditi potrebne UML dijagrame - klasne, *use case*, *use sequence*, opisati implementaciju te na kraju izraditi kratke korisničke upute).

Zadatak uručen pristupniku: 5. ožujka 2021.
Rok za predaju rada: 5. veljače 2022.

Mentor:

Siniša Sovilj

_____

doc.dr.sc. Siniša Sovilj

Sadržaj

1. Uvod 1

2. Načini rada operacijskih sustava Android i iOS 2

2.1 Općenito o operacijskom sustavu Android 2

2.2 Android arhitektura 3

2.3 Programski jezici razvoja mobilnih aplikacija 4

2.4 Općenito o MVC arhitekturi 4

2.5 Korištenje razvojnog okruženja Android Studio 5

2.6 Aplikacijske komponente 7

2.7 Apple iOS 9

2.8 Razlike operacijskih sustava Android i iOS 11

3. Opis i analiza sustava 12

4. Motivacija za izradu aplikacije 13

4.1 Uvod u SWOT analizu 13

4.2 Izrada i opis SWOT analize 13

4.3 Budući razvoj 14

4.4. Postojeće aplikacije 16

5. Funkcionalnosti aplikacije 19

5.1 Dijagram slučajeva korištenja 19

5.2 Dijagram slijeda obrazaca uporabe 23

5.3 Klasni dijagram 30

5.4 Prototipiranje korisničkog sučelja 32

6. Implementacija i korištene tehnologije 36

6.1 Implementacija baze podataka 36

6.2 Implementacija pozadinskog sustava 42

6.3 Implementacija Android aplikacije 44

6.3.1 Implementacija i korištenje REST servisa 44

6.3.2 Prijava i registracija korisnika u aplikaciju 45
6.3.3 Popis glavnih funkcionalnosti (korisnik) 51
6.3.4 Pregled predstava 51
6.3.5 Rezervacija ulaznica 54
6.3.6 Pregled rezervacija i favorita 58
6.3.7 Upravljanje korisničkim računom 60
6.3.8 Pregled cijena 61
6.3.9 Pregled najčešćih pitanja 61
6.3.10 Pregled informacija o aplikaciji 62
6.3.11 Administratorski prikaz predstava 62
6.3.12 Administratorski pregled rezerviranih ulaznica 63
6.3.13 Administrator upravljanja profila 66
6.3.14 Unos predstave i ponuda 67
6.3.15 Unos i pregled cijena 68
7. Korisničke upute 69
8. Zaključak 95
9. Literatura 96
10. Popis slika 99
11. Popis programskog koda 102
12. Popis tablica 103
SAŽETAK 104
SUMMARY 104

1. Uvod

Još od antičkih vremena do danas cilj kazališta je ostao isti. Izvođači su kroz pjesmu, ples, različite načine interpretacije teksta izražavali i prenosili emocije, energiju, iznenađenja, otkrića, sve ono čemu je čovjek težio. Scenskim nastupom doticali su unutarnji svijet i ostavljali tragove ljepote u čovjeku, tako da je gledatelj još dugo vremena bio pod dojmom predstave. Sama svečanost i način izvođenja su bili uvijek na prvom mjestu, dok je prostor za gledatelje dugo bio zanemaren. U staroj Grčkoj, kolijevci kazališta, gledatelji su prvotno kružno stajali oko izvođača. S vremenom se počelo mijenjati. Prvo su ugradili sjedeća mjesta. Vremenski uvjeti ubrzali su natkrivanje pojedinih dijelova. Nedostaci natkrivenih mjesta potaknuli su gradnju objekata, dvorana s krovovima. Kroz stoljeća, dvorane su mijenjale oblik i veličinu, nastajala su kazališta koja danas posjećujemo.

Odlazak u kazalište često se smatra društevnim činom koji uključuje gledanje novih predstava u društvu prijatelja, obitelji ili samostalni odlazak. Posjetitelji se suočavaju s velikim redovima čekanja na blagajnama za kupnju ulaznica uoči izvođenja predstave što često rezultira kašnjenjem ili odustajanjem od gledanja. Dodatni problem može biti i pristup podacima na web stranici preko mobilnog uređaja, jer nema odgovarajuće mobilne verzije za prikaz. Tu su i nedostaci neoptimiziranih slika, nepostojanja pravila skaliranja, te nepodržane vrste medija. Navedeni nedostaci ponekad administratore i korisnike prisiljavaju koristiti fizičko računalo kako bi došli do određenih informacija. S druge strane, administratorima bi olakšalo postojanje mobilne aplikacije koja bi podržala sve funkcionalnosti kao i standardna web aplikacija. Time bi se rješio problem pristupa fizičkom računalu radi izvršavanja određenog zadatka. Idealno rješenje je razvoj mobilne aplikacije koju će istovremeno moći koristiti administratori i korisnici. Aplikacija treba omogućiti svim korisnicima online rezervaciju ulaznica koje potom mogu jednostavno platiti i podignuti. Adminsitratori mogu izvršavati jednake funkcionalnosti kao i preko web aplikacije. Aplikacija bi podržavala slogan mobilnosti i dostupnosti: *„U bilo koje vrijeme na bilo kojem mjestu“*. Sve što je potrebno za sigurno korištenje je Android mobilni uređaj povezan s internetom.

Cilj rada je izraditi takvu mobilnu aplikaciju, približiti je, pojasniti i omogućiti njenu uporabu ljubiteljima kazališta. Osim uvoda i zaključka sve navedeno u cilju je realizirano kroz šest detaljno opisanih poglavlja koji slijede u nastavku rada.

2. Načini rada operacijskih sustava Android i iOS

2.1 Općenito o operacijskom sustavu Android

Android je operacijski sustav zasnovan na Linux jezgri dizajniranog za mobitele i ostale uređaje osjetljive na dodir. Google ga je kupio 2005. godine, te ga razvija do danas. Sadrži otvoreni kod (Apache licenca) dostupan svima. Android OS je podijeljen na različite brojeve verzija što podrazumijeva promjene u značajkama, radu i stabilnosti, a koje obično imaju kodna imena. Prva novija verzija kojoj je dodijeljen numerički naziv je Android 10. API predstavlja cjelobrojnu vrijednost koja jedinstveno identificira reviziju API okvira koju nudi verzija Android Platforme. [15] Najnovija verzija je Android 11 koja je izašla u rujnu 2020. godine. Moguće je razvijati aplikacije u različitim okruženjima poput: Android Studio, Eclipse, Netbeans, Unreal Engine itd.

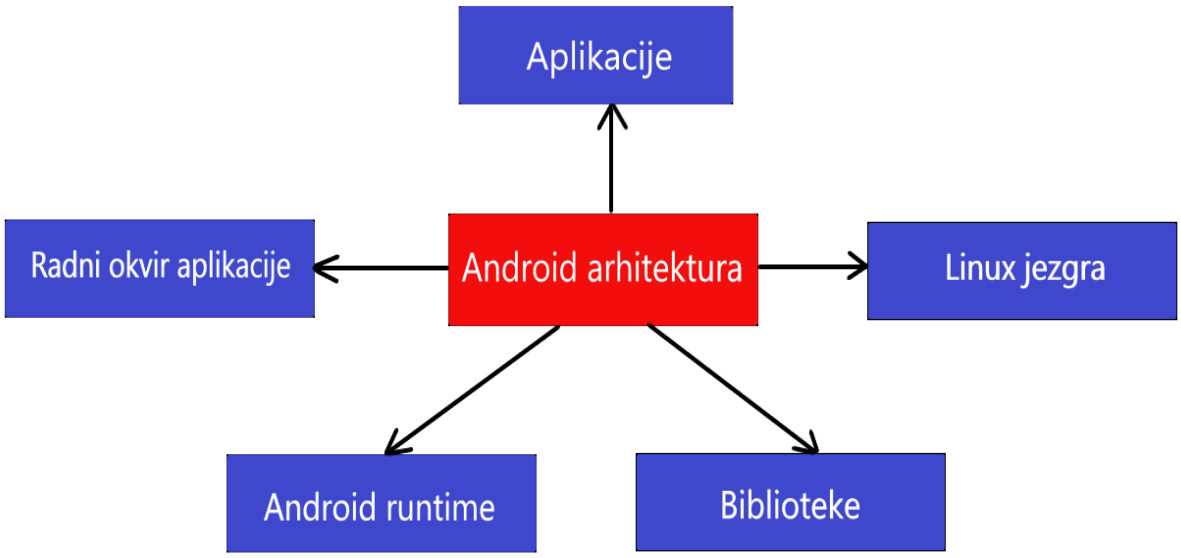
Code name	Version numbers	API level	Release date
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010
Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014
Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

Slika 1. Prikaz dosadašnjih verzija Android sustava [15]

Android je najprodavaniji OS na svijetu. Od svibnja 2021. godine ima preko tri milijarde aktivnih korisnika mjesečno, a od siječnja iste godine trgovina Google Play sadrži više od 3 milijuna aplikacija. Google je razvio različite slučajeve uporabe, uključujući Android Wear za nosive uređaje poput pametnih satova, Android TV za televizore, te Android Things za pametne telefone. Maskota Androida je mali zeleni robot. [16]

2.2 Android arhitektura

Način rada Androida se najbolje vidi iz njegove arhitekture. Arhitektura sadrži pet osnovnih dijelova koji tvore operacijski sustav.



Slika 2. Prikaz Android arhitekture

Sastavni dijelovi opisuju:

1. Linux jezgra - predstavlja osnovni sloj na kojem je izgrađen Android. Sloj sadrži drajvere uređaja za različite hardverske komponente svakog pojedinačnog Android uređaja. [21] Odgovoran je za upravljanje programom potrošnje, fotoaparata, audio programa, memorije, tipkovnice itd.
2. Biblioteke – sadrže kod koji omogućuje osnovne funkcije Android operacijskog sustava. [21] Primjeri biblioteka su: SQLite (baza podataka), OpenGL (grafika), Web kit (web sadržaj), Media Framework (audio, video), FreeType (font), itd.
3. Android runtime – obuhvaća skup biblioteka koje omogućuju programerima pisanje Android aplikacija korištenjem Java programskog jezika. Okruženje sadrži Dalvik virtualnu mašinu koja omogućuje svakoj Android aplikaciji da se izvršava u vlastitom procesu, s vlastitom instancom Dalvik mašine. [21]
4. Radni okvir aplikacije – omogućava korištenje različitih mogućnosti Android operacija, tako da ih programeri mogu koristiti za razvoj aplikacija. [21]
5. Aplikacije – na sloju se nalaze aplikacije koje se isporučuju Android uređajima. [21] Aplikacije izrađene ili preuzete iz trgovine će biti instalirane na ovom sloju.

2.3 Programski jezici razvoja mobilnih aplikacija

Slijede dva najpoznatija programska jezika namijenjena razvoju mobilnih aplikacija:

- **Java** je objektno orijentirani programski jezik opće namjene. Najčešće se koristi za izradu stolnih i mobilnih aplikacija, te obradu veće količine podataka. Danas radi na preko 3 milijarde uređaja diljem svijeta, što je čini najpopularnijim programskim jezikom. Pruža nekoliko bitnih značajki, kao što su automatsko upravljanje memorijom i sigurna platforma za razvoj. [17] Java posjeduje jedinstveni moto koji glasi: *„Napiši jednom, pokreni bilo gdje (eng. Write Once, Run Everywhere)“* čime se naglašava njena višeplatformska rasprostranjenost.

- **Kotlin** je programski jezik sličan Javi. Radi zajedno s Javom unutar koje se mogu uvesti promjene koda i migracije. Glavna prednost proizlazi iz kompaktne sintakse i čistog koda, te lakšeg održavanja projekata. Sintaksa koda napisana u Kotlinu je manja u odnosu na kod u Javi. Smatra se da će u budućnosti postati glavni programski jezik namijenjen razvoju Android aplikacija, pa prebacivanje s Jave na Kotlin može predstavljati veliki izazov za programere.

2.4 Općenito o MVC arhitekturi

MVC (Model-View-Controller) je najpoznatija i najčešće korištena arhitektura koja je gotovo prisutna u svim programskim okvirima. Prilikom izrade aplikacija, potrebno je organizirati programski kod s obzirom na klase koje definiraju sučelje i događaje na njemu. Zadatak arhitekture je podjela aplikacije s obzirom na poslovnu, ulaznu i korisničku logiku, ali istovremeno ostavljajući efekat povezanost među elementima. Zbog toga se aplikacija dijeli na tri dijela: Model, View, Controller.

- **Model**: sadrži poslovnu logiku aplikacije. Često je unutar modela prisutna enkapsulizacija koja skriva manipulaciju atributa iz klase. Pristup atributima se postiže postavljanjem odnosno dohvaćanjem vrijednosti (metode Get i Set).

- **View**: koristi se za prikaz podataka na sučelju. Iste podatke korisnik pregledava.

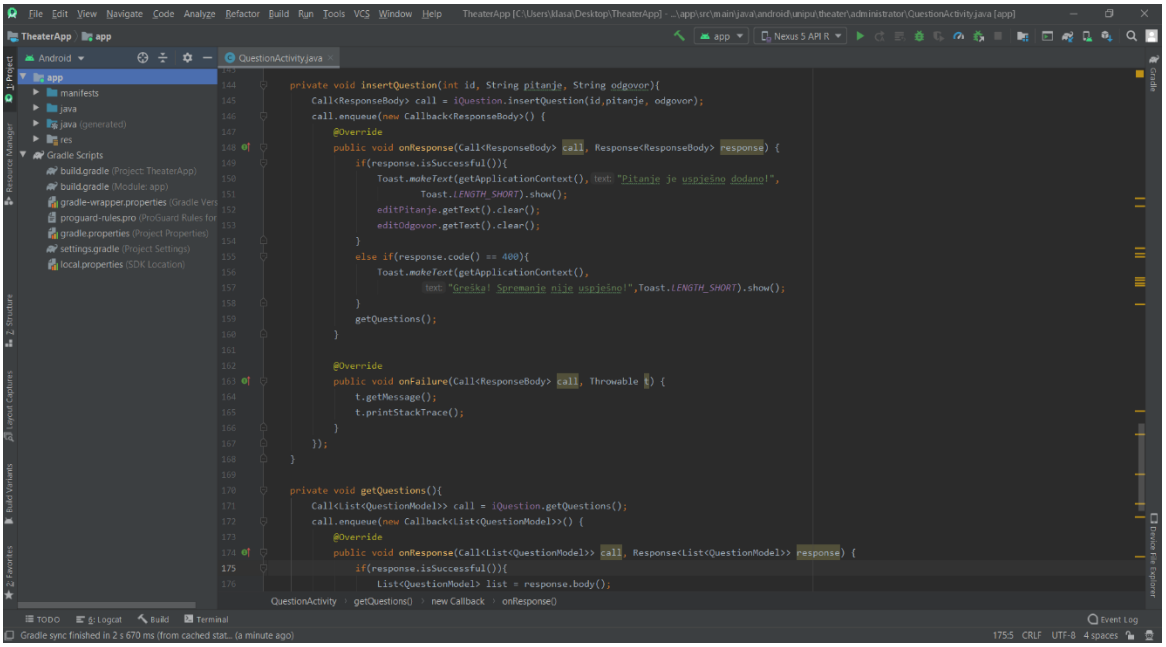
- **Controller**: se rabi za upravljanje svih događaja nad kojima je postavljen slušač. On je posrednik između Modela i Viewa. U većini slučajeva poziva metode iz Modela za izvršavanje nekog zadatka. Primjer je dohvat podataka iz Modela i prikaz istih na sučelju koristeći View komponentu.

2.5 Korištenje razvojnog okruženja Android Studio

Android Studio je integrirano razvojno okruženje (IDE) za razvoj Android aplikacija. Temelji se na IntelliJ IDEA, Java integriranom razvojnom okruženju, te uključuje dodatne alate za uređenje koda. [13] Android SDK predstavlja skup klasa i pripadnih alata koji zajedno tvore temeljnu jedinicu razvoja programskih proizvoda za uređaje. Aplikacije programirane u Android Studiu se kompajliraju u APK format, te se mogu postaviti u trgovinu (npr. Google Play ili Samsung Galaxy Store). Pokretanje aplikacije se može postići na dva načina, korištenjem emulatora ili stvarnog fizičkog uređaja. Emulator je AVD (eng. Android Virtual Device) virtualni uređaj koji simulira način rada mobitela. Sadrži sve funkcionalnosti kao i fizički mobitel. Dobra praksa je pokretanje na što više fizičkih uređaja čime se postiže siguran dolazak do stvarnih rezultata. Slijedi usporedba pokretanja aplikacija na emulatoru s obzirom na prednosti i nedostatake.

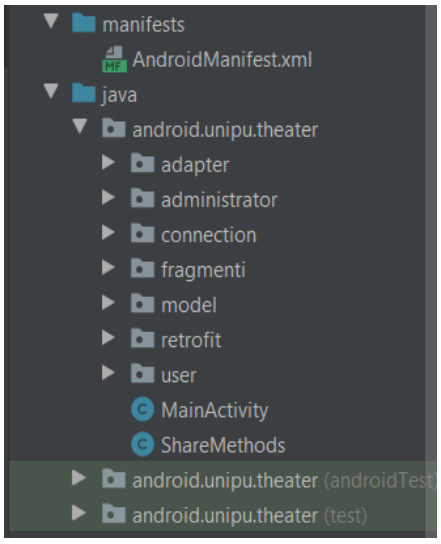
- **Prednosti**: dostupnost u bilo kojem trenutku, jednostavnost korištenja, jeftino rješenje, mogućnost pronalazaka anomalija vezanih za sučelje, različite verzije.
- **Nedostatci**: spora brzina izvođenja, nema vođenja evidencije o pregrijavanju, drugačiji izgled aplikacije, za potrebe pokretanja je potrebno jače računalo.

Sučelje prikazano slikom 3 je podijeljeno na prozore koji se mogu prikazivati i zatvarati od strane programera. Na vrhu se nalazi alatna traka, s lijeve strane je struktura aplikacije, na desnoj strani su skriveni alati, centar je predstavljen radnom površinom, a dno je realizirano popratnim statusom prilikom izvršavanja.

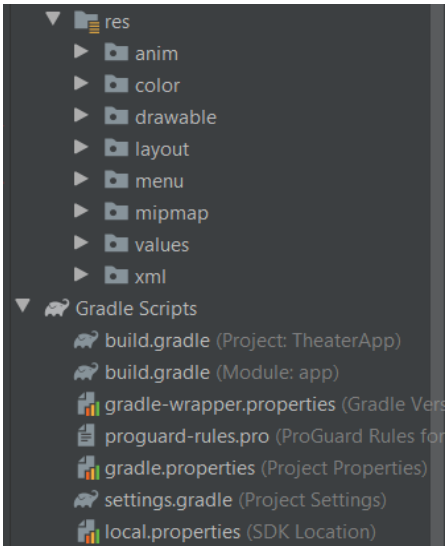


Slika 3. Sučelje razvojnog okruženja Android Studio (izvor. autor)

Bitno je naglasiti kako je programska logika u potpunosti odvojena od korisničkog dizajna sučelja. Ona za opis koristi programske jezike poput Jave i Kotlina, dok je XML korišten za izradu sučelja. Hijerarhiju je moguće vidjeti iz rasporeda smještenosti klasa i vanjskih resursa. Vanjski resursi su smješteni u mapi *„res“*, dok se klase nalaze u mapi *„java“*. Sve mape unutar strukture se često nazivaju čvorovima. Sadržaj projekta je podijeljen na četiri glavne cijeline. U mapi ***„manifests“*** se nalazi datoteka AndroidManifest.xml koja sadrži sve informacije i uvjete vezane za način rada aplikacije. Glavni element <user-sdk> sadrži kompatibilnost najviše i najniže verzije koje su odabrane prilikom kreiranja projekta. Ostali primjeri informacija su: popis deklariranih aktivnosti, naziv i ikona aplikacije, dozvole, orijentacija itd.



Slika 4. Prikaz strukture mape java



Slika 5. Prikaz strukture mape res

U mapi ***„java“*** se nalazi popis svih kreiranih klasa uključujući i preostale testne klase. Drugim riječima, unutar nje se nalazi cjelokupni programski kod aplikacije. Ova mapa sadrži proizvoljno kreirane podmape namijenjene rasporedu klasa prilikom izrade, to su: adapter, connection, model, administrator, retrofit, user i fragment.
Unutar mape **„res“** se nalazi popis svih vanjskih resursa koji obuhvaćaju značajke vezanih za izgradnju korisničkog sučelja aplikacije. Slijedi popis značajki mape:

1. **Drawable**: mapa sadrži sve slike koje se koriste u aplikaciji. Programer može samostalno prenijeti sliku različitih formata s računala ili može koristiti postojeće predloške (eng. Vector and Image Assets). Korištenjem predložaka najčešće opcije su promjene boje ikona, te dimenzija s obzirom na veličinu ekrana.

2. **Layout**: sadrži popis XML datoteka koje predstavljaju korisničko sučelje.

3. **Menu**: sadrži XML datoteke kreirane za potrebe sučelja i navigacije izbornika.
4. **Mipmap**: sadrži ikone koje se koriste prilikom pokretanja aplikacije. Svaka ikona ima unaprijed definiranu veličinu (primjer. hdpi, mdpi, xhdpi, xxhdpi,xxxhdpi). Na taj način je realizirano prilagođavanje slike zaslonu ovisno o veličini uređaja.
5. **Values**: obuhvaća XML datoteke za korištenje: boja, stringova, te stilova. Unutar mape boja se navode sve boje koje se planiraju koristiti u aplikaciji. Definiranje boje se postiže navođenjem heksadecimalne vrijednosti i proizvoljnog naziva. Mapa stringova sadrži popis tekstova. Uporabom stilova je moguće izvršiti nekoliko funkcija, primjeri su: promjena boje i veličine fonta, promjena stila fonta, postavljanje naslova, promjena pozadine itd.

Prozvoljno izrađene mape unutar resursa su:

1. **Anim**: sadrži XML datoteke unutar kojih su definirana obilježja animacija.
2. **Xml**: sadrži XML datoteku unutar koje se nalazi postavka mrežne sigurnosti.

Automatski generirana klasa R: sadrži sve identifikatore (ID) resursa navedenih prilikom izgradnje sučelja od strane programera u XML-u. Tom klasom je omogućeno povezivanje resursa s programskim kodom. Postojanje reakcije odnosno definiranje izgleda sučelja, te izvršavanje zadataka korisničkom interakcijom na element rezultira povezivanje elemenata sučelja s programom. Shodno tome, po potrebi se postavljaju odgovarajući slušači događaja. Promjena resursa automatski ažurira klasu.

Gradle Script sadrži razvojne skripte (datoteke) unutar kojih su definirane biblioteke potrebne razvoju i testiranju programskog rješenja. Osim biblioteka tu se nalaze podaci o verziji aplikacije, verzijama SDK, te preostaloj konfiguraciji. Potrebno je s vremena na vrijeme ažurirati biblioteke kako bi se uvijek koristile najnovije i nadograđene verzije.

2.6 Aplikacijske komponente

Aplikacijske komponente odnosno elementi predstavljaju vrijednosti na temelju kojih je izgrađena aplikacija. Slijedi opis nekoliko najpoznatijih korištenih komponenata:

1. Aktivnost (eng. Activity): predstavlja jedan zaslon korisničkog sučelja pomoću kojega korisnik može izvršavati jedan određeni zadatak. Aplikacija može imati jednu ili više aktivnosti te se zbog toga smatra glavnim elementom. Sve aktivnost su realizirane u datoteci AndroidManifest.xml unutar koje su povezane s programskim rješenjem.

Većina aktivnosti posjeduje vlastito sučelje, te imaju vlastiti životni ciklus. Najpoznatija metoda je „onCreate()“ koja omogućuje stvaranje i instanciranje objekata aplikacije. Prilikom pokretanja aktivnosti aplikacije, poziva se „onCreate()“ metoda.

2. Fragment: predstavlja podaktivnost koja se koristi u svrhu prilagodbe različitim zaslonima (npr. razvoj aplikacija za tablet uređaje). Aktivnosti mogu sadržavati jedan ili više različitih fragmenata. U najviše slučajeva, jedna aktivnost sadrži jedan fragment. Fragment ima vlastiti životni ciklus i korisničko sučelje. Prednosti njihovog korištenja je: dinamičko dodavanje, zamjena i uklanjanje, te mogućnost ponovnog korištenja. Aktivnost je posrednik u komunikaciji između fragmenata. [10] Izvrsno se prilagođavaju različitim zaslonima uređaja na kojima se izvršava aplikacija.

```
getSupportFragmentManager().beginTransaction()
     .replace(R.id.containerUnos,fragment).commit();
```

Programski kod 1. Prikaz dinamičke zamjene fragmenata

3. Namjere (eng. Intent): predstavlja poruku koja se prenosi između komponenata kao što su: aktivnosti, pružatelji sadržaja, usluge itd. [11] Postoje dvije vrste namjera, implicitne i eksplicitne. Implicitna namjera se koristi kada je poznat zadatak koji se treba izvršiti, ali se ne zna koja aktivnost to omogućava. Eksplicitna namjera se koristi kada znamo u kojem trenutku je potrebno aktivirati i prikazati aktivnost na zaslonu, njeno korištenje je realizirano unutar aplikacije. Metoda „startActivity“ se rabi za iniciranje aktivnosti, te se potom aktivnost stavlja na vrh stoga. Zatvaranje aktivnosti se postiže pozivanjem metode finish() koja istu uklanja sa stoga. [1] Uklonjenosti sa stoga se može provjeriti pritiskom na gumb „Natrag“ (eng. Back) na mobitelu gdje se vidi nemogućnost povratka na prethodnu aktivnost.

```
toolbar.setNavigationOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       Intent intent = new Intent(PerformancesActivity.this,AdminWindow.class);
       startActivity(intent);
       finish();
   }
});
```

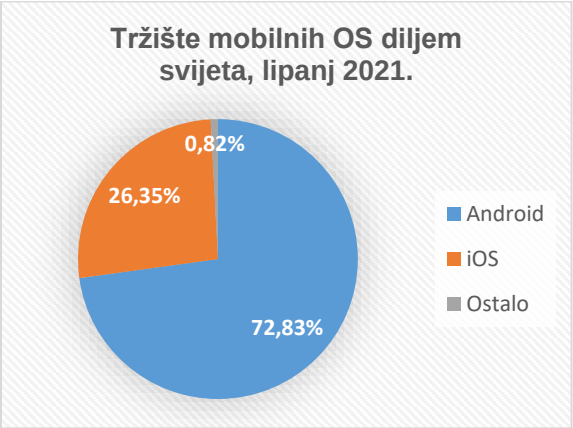
Programski kod 2. Pokretanje eksplicitne namjere nove aktivnosti

4. Usluge (eng. Services): komponenta koja radi u pozadini za obavljanje dugotrajnih operacija bez potrebe za interakcijom s korisnikom. Obuhvaća metode koje se implementiraju u svrhu praćenja promjena stanja usluge i izvođenja posla u odgovarajućoj fazi. Usluge se odvijaju uvijek u pozadini. [12]

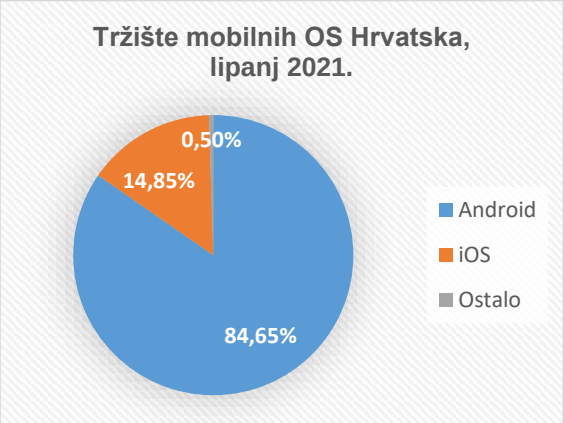
2.7 Apple iOS

Apple iOS je operacijski sustav zasnovan na Mac OS-u koji pokreće liniju stolnih i prijenosnih Apple računala, mobitela, tableta itd. Prva verzija je objavljena 2007. godine kada ujedno debitira na tržištu predstavljajući iPhone. Verzije od 1 do 3 su se nazivale Apple OS, a u sljedećim verzijama su uveli naziv iOS koji se i danas koristi. Operacijski sustav se temelji na Unixu koji pokreće sve Apple mobilne uređaje, te se do danas smatra drugom najpopularnijom platformom odmah iza Androida. [18] Godišnje izlazi po jedna nova verzija, a zadnja je bila 2020. godine kada je predstavljen iOS 14. Svakom novom verzijom postižu se bolje značajke i performanse korištenja uređaja. Uvođenjem pametnih aplikacija korisnicima se pomaže u izvršavanju svakodnevnih zadataka. iOS omogućava korištenje Siri virtualnog asistenta koji na temelju glasovnih upita izvršava određeni zadatak u obliku odgovora na traženi upit. Siri je u kratkom roku postala hit među korisnicima diljem svijeta. Nakon toga je došlo do razvoja virtualnih asistenata koji se mogu pokretati na preostalim operacijskim sustavima, a razvijeni su od strane drugih proizvođača. Neke od najpoznatijih aplikacija razvijenih samo za iOS su: Timepage, Overcast, Lose It itd.

Prikaz evidencije korištenja mobilnih operacijskih sustava s obzirom na područje:



Slika 6.Tržište mobilnih OS – svijet



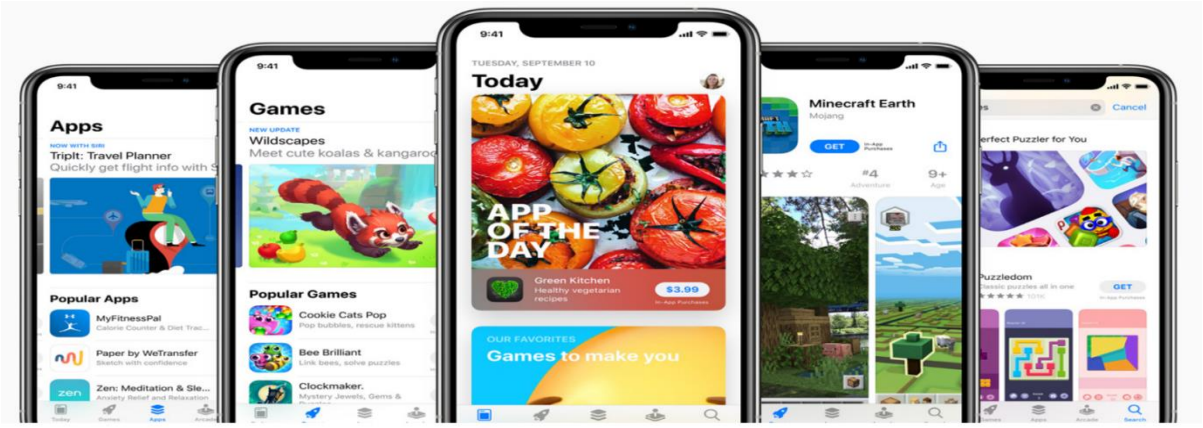
Slika 7. Tržište mobilnih OS – Hrvatska

Prema podacima prikupljenih diljem svijeta moguće je vidjeti da čak cca. 72% se odnosi na operacijski sustav Android, dok na iOS 26%. [19] Postotak potvrđuje rezultat uporabe Androida kao primarnog operacijskog sustava u mobilnim uređajima. Podaci iz Hrvatske su slični podacima iz svijeta. Android OS zauzima približno 85%, dok iOS znatno manje. Iz ovoga se može zaključiti kako se u Hrvatskoj u većini slučajeva koriste Android uređaji zbog široko ponuđenog spektra po prilagođenim cijenama.

Prije izrade aplikacije potrebno je definirati sve funkcionalnosti i zadatke. Nakon toga slijedi odabir programskog jezika i razvojnog okruženja. Najpoznatiji programski jezici za razvoj iOS aplikacija su: Swift, C#, Objective-C. Odabir programskog jezika proizlazi iz više kuteva gledanja poput: jednostavnosti korištenja, traženost i dostupnost na tržištu, dosljednosti, brzine razvoja i izvršavanja. Zbog svega toga se Swift smatra trenutno najpopularnijim jezikom za razvoj modernih iOS aplikacija. Mnoga poduzeća traže programere s iskustvom u tom programskom jeziku. Definiranjem programskog jezika, pomno se odabire razvojno okruženje unutar kojega će razvijati aplikacija. Dva najpoznatija podržana razvojna okruženja za iOS su: Xcode i AppCode. Po završetku izrade aplikacije, ista se može distribuirati na specijaliziranim trgovinama poput Apple App Store-a.

Testiranje aplikacije se provodi na iOS simulatoru koji se izvodi na Macintosh računalu. Unutar okruženja se odabire želi li se aplikacija pokrenuti na fizičkom uređaju ili simulatoru. Odabirom simulatora, okruženje automatski pokreće iOS simulator.

Apple App Store je internetska trgovina unutar koje se mogu kupiti i besplatno preuzeti mobilne aplikacije i igrice razvijene za Apple iOS. U trgovini je moguće pronaći aplikacije koje su potrebne za kolaboraciju s ostalim mogućnostima mobilnog uređaja. Primjerice, za upravljanje glazbom je potrebno imati instaliranu aplikaciju iTunes. Pored vodećeg Androida, iOS postiže velike uspjehe na američkom tržištu. Do sada je zabilježeno nekoliko milijuna aktivnih aplikacija unutar trgovine. Apple i Google Play imaju manji broj aplikacija koje se plaćaju, što znači da kod oboje prevladavaju besplatne aplikacije. Međutim Apple posjeduje veći udio u aplikacijama koje se plaćaju za njihovo korištenje u odnosu na Google Play.



Slika 8. Prikaz trgovine App Store (Izvor: Screenshot [20])

2.8 Razlike operacijskih sustava Android i iOS

Postoje svakodnevni mnogobrojni problemi s kojima se susreću Android i iOS operacijski sustavi. Prvi problemi nastaju prilikom razvoja proizvoda. Programeri se suočavaju s grešakama u kodu, rušenjem aplikacije, limitiranosti prilikom korištenja besplatnih servisa. Različitost cijena i distribucija na tržištu aplikacija može pomoći u odlučivanju koji će se operacijski sustav početi koristiti. Daljnji problem nastaje prilikom testiranja programskog rješenja. Jedan od ključnih potencijalnih problema je sigurnost aplikacije. Trgovine nastoje zaštiti korisnike na način da unaprijed provjere i testiraju aplikaciju koja se želi postaviti. Međutim, nitko ne može garantirati sigurnost korisniku prilikom korištenja aplikacije. Mnoge distribuirane aplikacije ne rade na svim verzijama mobilnih uređaja, pa je tu postavljeno ograničenje za korištenje. U tablici se mogu vidjeti osnovne razlike između navedenih operacijskih sustava.

Tablica 1. Tablični prikaz razlika operacijskih sustava Android i iOS

Broj	Opis	Android	iOS
1.	Početak rada	2008. godina	2007. godina
2.	Programski jezici	Java, Kotlin	Objective-C, Swift
3.	Trgovina	Google Play Store	Apple App Store
4.	Razvojno okruženje	Android Studio	XCode, AppCode
5.	Vlasništvo	Google	Apple Inc.
6.	Virtualni asistenti	Google Assistant	Siri
7.	Prosječna cijena	Niska	Visoka
8.	Sigurnost	Stalna ažuriranja i zakrpe	Rjeđa ažuriranja
9.	Hardver	Nepoznato dobavljača	Samo Apple
10.	OS	Linux	Unix
11.	Zadnja verzija	Android 11	iOS 14
12.	Razvojna linija	Mobiteli, tableti, TV, satovi	iPhone, iPad, iPod, računala

3. Opis i analiza sustava

Kazalište je kulturno-umjetnička ustanova namijenjena održavanju priredbi scenske umjetnosti, tu se podrazumijevaju profesionalna i amaterska kazališta. U radu kazališta sudjeluju djelatnici podijeljeni na: umjetničko, tehničko i administrativno osoblje. [6]

U doba bez interneta, informacije su bile dostupne samo preko pisanih medija, novina ili časopisa. Danas je situacija promijenjena. Porastom broja osoba koje koriste internet dovela je do povećanja razvoja kazališta i njegovih usluga.

Visoke cijene ulaznica su jedan od glavnih razloga izbjegavanja odlaska u kazalište. Osim cijene ulaznica, većina posjetitelja mora plaćati parkirno mjesto za osobno vozilo. Cijena mjesta za parkiranje raste ovisno o broju sati. Duže predstave često znače plaćanje unaprijed za nekoliko sati. Veliki problem je i relativno mali broj takvih mjesta. Navedeni problemi rješavaju se smanjenjem cijena i uvođenjem različitih popusta, te izgradnjom novih mjesta za parkiranje blizu kazališta. Problemima nikad kraja.

Nedostaci i problemi nisu obeshrabrili ljubitelje kazališta, njihov broj se povećava.

Tablica 2. Prikaz podataka kazališta u razdoblju od 2010. do 2018. godine [6]

Godina	Kazališta		
	Broj	Predstave	Posjetitelji
2010	39	4928	926 134
2011	53	5045	1 057 846
2012	60	4984	1 044 273
2013	80	5976	1 145 516
2014	81	5965	1 139 094
2015	89	5770	1 044 845
2016	84	6018	1 380 648
2017	90	6369	1 237 664
2018	95	6729	1 258 631

Iz priložene tablice se vidi porast broja kazališta, interes za predstave raste iz godine u godinu. Prodaja ulaznica proporcionalno se povećava s brojem kazališta, što primjećujemo u podacima o posjetiteljima.

4. Motivacija za izradu aplikacije

4.1 Uvod u SWOT analizu

Postoje mnogi izazovi s kojima se aplikacije gotovo svakodnevno susreću. Potrebno je poduzeti određene radnje kako bi se proizvod što duže održao na tržištu. Radnje obuhvaćaju pregled svih prednosti i nedostataka gotovog proizvoda kako bi se donio zaključak. Postizanje poboljšanja se očitava procjenom, stoga su potrebni svi aspekti njegovog poslovanja. Jedan od najpoznatijih alata za analizu trenutnog poslovanja je SWOT analiza. Analiza se definira kao tehnika koja procjenjuje uspješnost poslovanja i pomaže u donošenju strateških odluka. Oslanja se na resurse podijeljene u četiri glavne kategorije: snagu (eng. Strength), slabost (eng. Weakness), priliku (eng. Opportunity) i prijetnju (eng. Threat).

4.2 Izrada i opis SWOT analize

Prilikom izrade SWOT analize potrebno je pronaći nekoliko glavnih stavki koje pripadaju navedenim kategorijama. Treba proučiti vanjske i unutarnje strane aplikacije, kako bi se realizirala kvalitetna analiza koja će u budućnosti unaprijediti proizvod. Neke od dobrih strana aplikacije su besplatnost i dostupnost svakome tko posjeduje Android mobilni uređaj. Aplikacija sadrži sučelje koje omogućuje jednostavno kretanje kroz aplikaciju čime je postignut brz i efikasan dolazak do željenih informacija. Sve što je potrebno za korištenje ovakve aplikacije je pristup internetu. Povezivanje uređaja s internetom se može realizirati tako što korisnik preuzima mobilne podatke ili Wi-Fi s odgovarajućim ključem i lozinkom. Pristupom internetu dobivaju se podaci u realnom vremenu. Na taj način se sprječava korištenje zastarjelih podataka, kao što su podaci o izvedenim predstavama. Istu aplikaciju mogu koristiti administratori i korisnici jer su odvojeni različitim ulazima. Time je riješen problem velike količine resursa koji su potrebni razvoju zasebnih aplikacija. Korisničke glavne funkcionalnosti su: pregled predstava, rezerviranje ulaznica, te dodavanje predstava u favorite. Sporedne funkcionalnosti su: pregled cijena, upravljanje korisničkim profilom i pripadnim podacima, te pregled najčešćih pitanja i informacija vezanih uz samu aplikaciju. Administratorski dio se odnosi na: dodavanje i pregled predstava, unos i brisanje ponuda predstava, upravljanje profilom, dodavanje i brisanje cijena, te obrada najčešćih pitanja u vezi korištenja aplikacije. Ovakvom aplikacijom bi se znatno smanjili redovi za fizičku kupnju ulaznica jer bi se korisnicima ponudila online rezervacija.

Jednom rezervirane ulaznice se plate i podignu uoči početka predstave. Aplikacija se može smatrati EKO aplikacijom jer se pomoću nje smanjuje potreba tiskanja letaka i plakata za promidžbu predstava. Brzina učitavanja podataka ovisi o kvaliteti internetske veze i predstavlja jednu od negativinijih osobina aplikacije. Aplikaciju nije moguće koristiti u Offline načinu rada. Pitanje sigurnosti i nepoznatosti bitno utječu na njezin opstanak na tržištu. Često postoji rizik od zamjene aplikacije inačicom drugih proizvođača. Trenutno je razvijena aplikacija samo na jednom jeziku i jednoj platformi čime je smanjena njezina uporaba. Kako bi se dodatno poboljšala, potrebno je pronaći investitore koji su spremni uložiti sredstva u njen razvoj. Drugim riječima, otvorena je za unaprijeđenja i razna proširenja.

Slika 9. Prikaz SWOT analize (Izvor: autor)

4.3 Budući razvoj

Mobilne aplikacije su u potpunosti promijenile živote ljudi u relativnom kratkom periodu. Na tržištu se danas mogu pronaći različite ali slične konkurentske aplikacije koje mogu nadomjestiti nedostatke ove aplikacije. U najgorem slučaju takve aplikacije mogu u

potpunosti nadomjestiti postojeći proizvod čime počinje negativan trend organizacije. Kako bi se izbjegao takav scenarij, potrebno je u budućnosti poboljšati i unaprijediti aplikaciju na način da je može koristiti i šira populacija. Prije poboljšanja je potrebno provesti istraživanja unutar kojih bi korisnici iznijeli svoja mišljenja i iskustva vezana za uporabu. Istraživanje bi se provodilo preko anketa i recenzija ostavljenih unutar trgovine aplikacijama. Sve prikupljene rezultate istraživanja je potrebno statistički obraditi i prikazati kako bi se znalo u kojem smjeru se treba kretati u budućnosti. Redovita ažuriranja poboljšavaju samu aplikaciju i podižu njezinu razinu kvalitete, te ih treba često raditi. Prvi korak unaprjeđenja bio bi razvoj hibridne aplikacije za oba mobilna operacijska sustava, Android i iOS. Na taj način korisnici ne bi morali razmišljati mogu li koristiti aplikaciju na svom uređaju. Također, aplikacija bi trebala biti prevedena na nekoliko svjetskih jezika, koji bi omogućili stranim državljanima jednostavniju uporabu. Prije samog razvoja preporučeno je koristiti vlastite pozadinske tehnologije (eng. Backend Technology) koje pružaju glavne funkcionalnosti kao što je primjer povezivanja mobilnih aplikacija s bazom podataka koristeći upite.

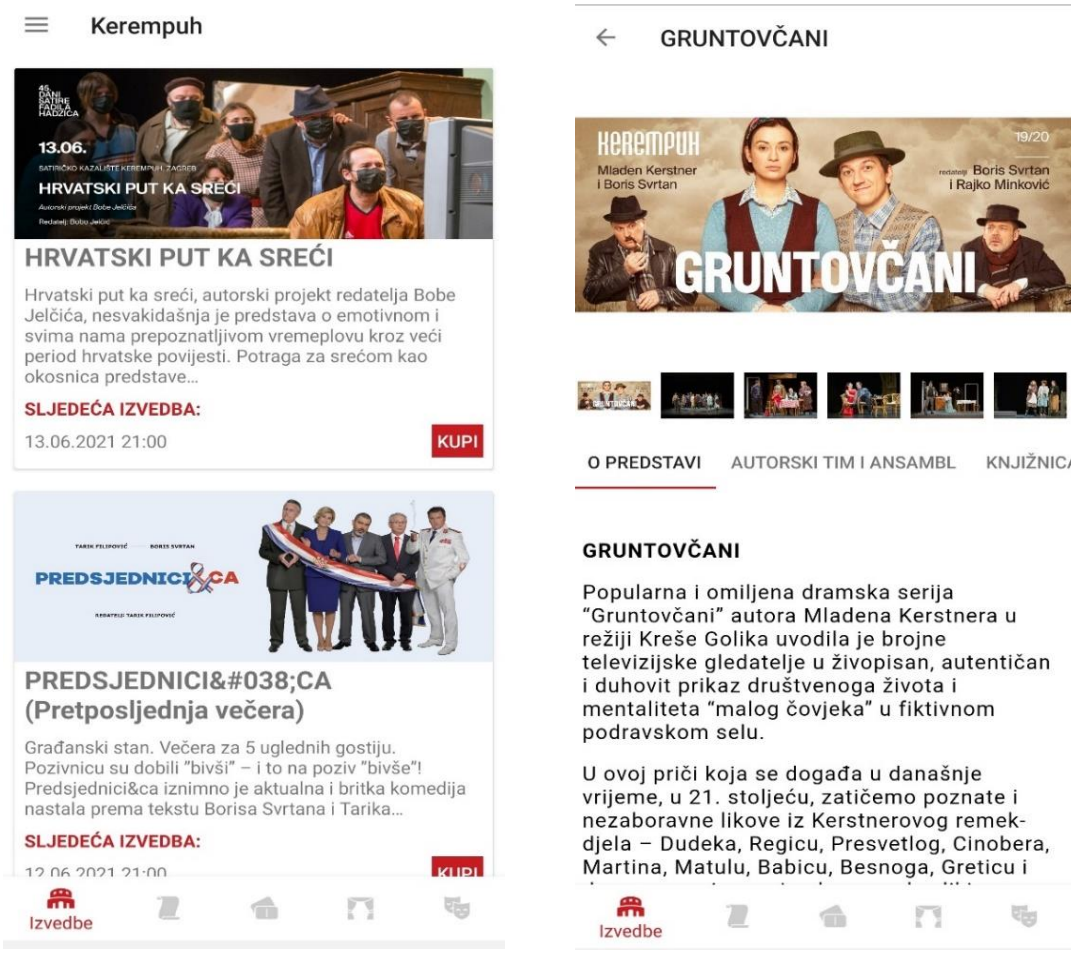
Pitanje privatnosti i sigurnosti predstavljaju glavni problem većine današnjih aplikacija. Besplatni servisi osim svoje ograničenosti, ne garantiraju sigurnost i dostupnost podataka u bilo kojem trenutku. Idealan slučaj bi bio korištenje vlastite baze podataka kako bi se održala određena razina privatnosti i sigurnosti od gubitaka odnosno krađe podataka. Navedeno zahtjeva i velike investicije, financijska sredstva. Sredstva se mogu pronaći u obliku županijskih, gradskih ili kazališnih donacija. Kazalište može samostalno pronaći investitore odnosno sponzore koji bi bili spremni investirati novac u proizvod. Kako bi se privukli potencijalni investitori potrebno je izraditi reklamu koja bi se besplatno promovirala preko društvenih medija. Razvojem društvenih medija privlače se i potencijalni korisnici aplikacije koji bi samim time bili pravovremeno obaviješteni o svakoj predstavi koja igra u kazalištu. Razvojem YouTube kanala mogu se postaviti zanimljivi intervjui s glumcima i isječci s predstava koji se mogu nadodati u aplikaciju.

U slučaju korisničkog zaborava za predstavu, dobra praksa je uvesti obavijesti koje bi podsjetile korisnika o vremenu izvođenja rezervirane predstave. Dodatak tome može biti i pregled ulaznice koja pruža jednostavnu navigaciju dolaska do kazališta prema trenutnoj lokaciji korisnika dobivenoj GPS-om. Tržišni ključni detalj može biti uvođenje SEO optimizacije koja poboljšava vidljivost aplikacije na tržištu, primjer Google Play.

4.4. Postojeće aplikacije

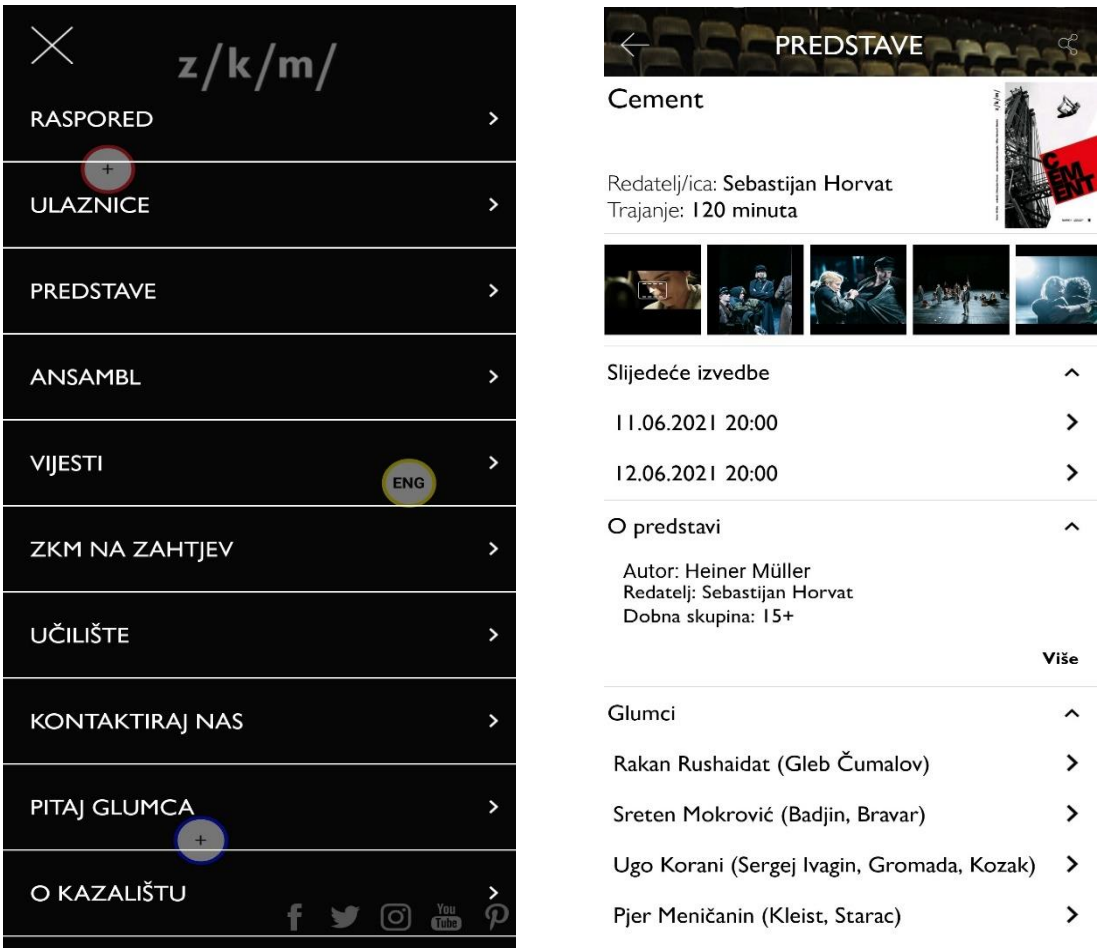
U ovom dijelu je potrebno istražiti slične postojeće aplikacije, te ih usporediti i opisati. Cilj je istražiti domaća i strana tržišta koja obuhvaćaju rezervaciju kazališnih ulaznica.

1. Satiričko kazalište Kerempuh posjeduje besplatnu aplikaciju namijenjenu prodaji i rezervaciji ulaznica. Kazalište je osnovano davne 1964. godine u Zagrebu. Od tada, pa sve do danas uveseljava svoje gledatelje, te omogućuje gledanje najpoznatijih predstava. Ovakva aplikacija je prva u Hrvatskoj koja može za sebe reći da osim internetske stranice posjeduje mobilne aplikacije za iOS i Android. [2] Aplikacija je razvijena od strane tvrtke BITWARE, te sadrži jednostavno sučelje koje omogućuje lak dolazak do željenih informacija. Ona pruža razne opcije, od kojih su najvažnije: pregled novosti, prikaz dostupnih predstava, pregled podataka o kontaktima, informacije o kazalištu, te popis rezerviranih karata. Moguće je pogledati i raspored izvođenja predstava, te njihov repertoar. Dodatno se dobiva detaljan uvid u ansambl koji prikazuje sve dramske prvakinje i prvake kazališta.



Slika 10. Popis i prikaz detalja predstava (Izvor: Screenshot [2])

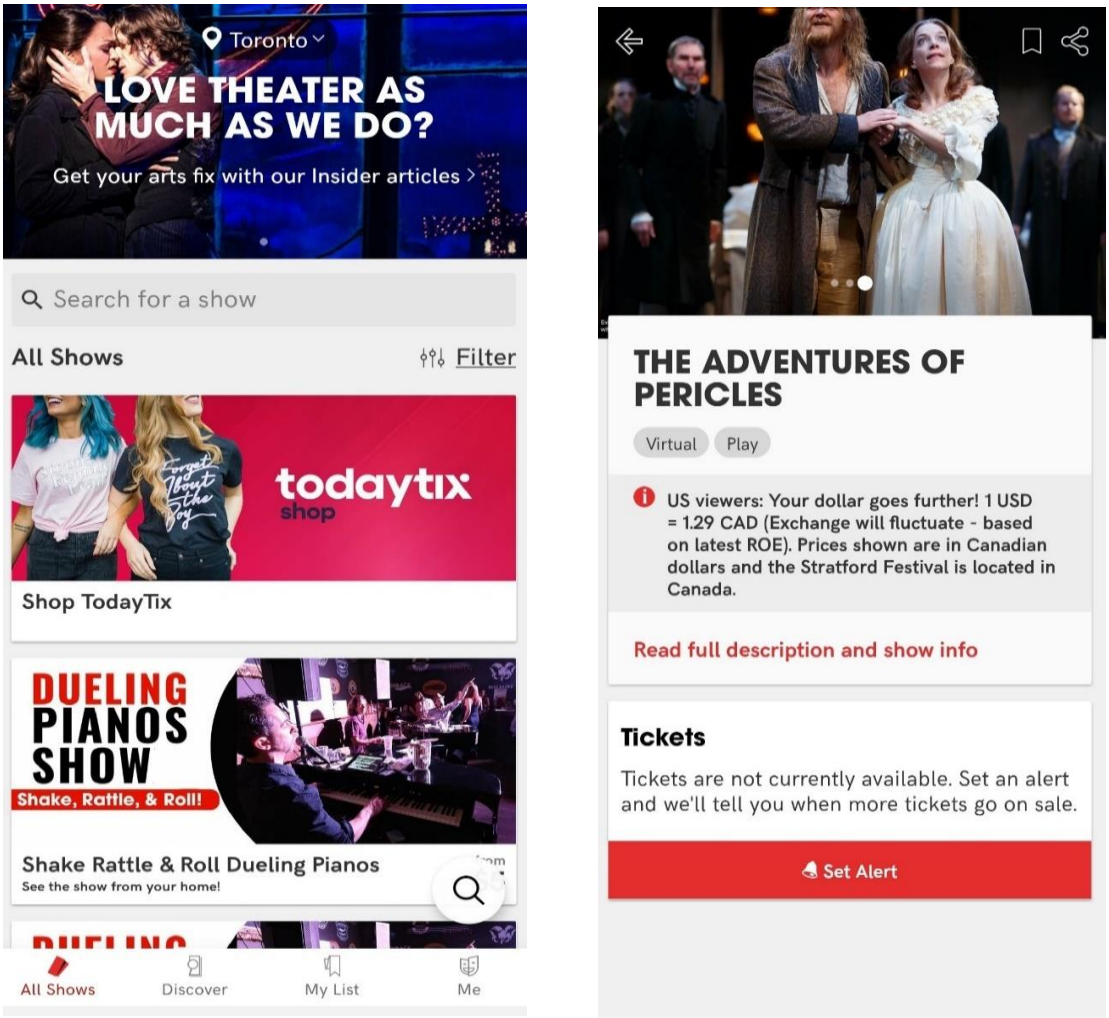
2. **Zagrebačko kazalište mladih** skraćeno ZKM (izvorno ime Pionirsko kazalište) je osnovano 1948. godine. Aplikacija je namijenjena rezervaciji i prodaji kazališnih ulaznica. Dodatne mogućnosti koje aplikacija pruža su pregledi: rasporeda prikazivanja, novosti, ansambla, informacije o kazalištu, prikaz kontakata, te informacije o aplikaciji. Korisnicima je stavljena na raspolaganje i opcija „Pitaj glumca“ unutar koje korisnik može direktno postaviti pitanje svojem omiljenom glumcu. Aplikacija je razvijena od strane tvrtke Bitware. Jednostavno i dinamično sučelje osigurava brzo snalaženje i uporabu od strane korisnika. Aplikacija je prevedena i na engleski jezik, te je na taj način proširenо područje uporabe. Svi sadržaji vezani za kazalište su dostupni na društvenim mrežama.



Slika 11. Izbornik i sadržaj predstave (Izvor: Screenshot [3])

3. TodayTix je aplikacija koja daje: pregled, rezerviranje i kupnju kazališnih ulaznica. Aplikacija je razvijena 2013. godine te pokriva značajnija kazališta u većim gradovima, kao što su: Toronto, New York, San Francisco, Los Angeles, Washington DC, Chicago.

Cilj ove aplikacije je približiti korisniku, omogućiti mu: uključivanje, otkrivanje i istraživanje umjetnosti i kulture. Glavne opcije ove aplikacije su: prikaz predstava, dodavanje predstava na listu želja, pregled dodanih predstava, pregled osobnih podataka, te postavke. Korisnik može odabrati predstavu s popisa predstava ili je pretražiti preko njihove tražilice. Prilikom pretraživanja moguće je i filtrirati predstave uz pomoć ugrađenih filtera. Filtriranje je moguće po: datumu, kategoriji predstave, po lokaciji kazališta, te pripadnim cijenama ulaznica. Nakon odabira predstave, mogu se pregledati svi detalji vezani za tu predstavu kao i pripadne rezervacije. Za rezervaciju ulaznica je potrebno posjedovati korisnički račun s pripadnom karticom za plaćanje rezervacije. U slučaju korisničkih poteškoća, aplikacija posjeduje i tehničku podršku gdje se isto pitanje može postaviti korisničkoj službi.

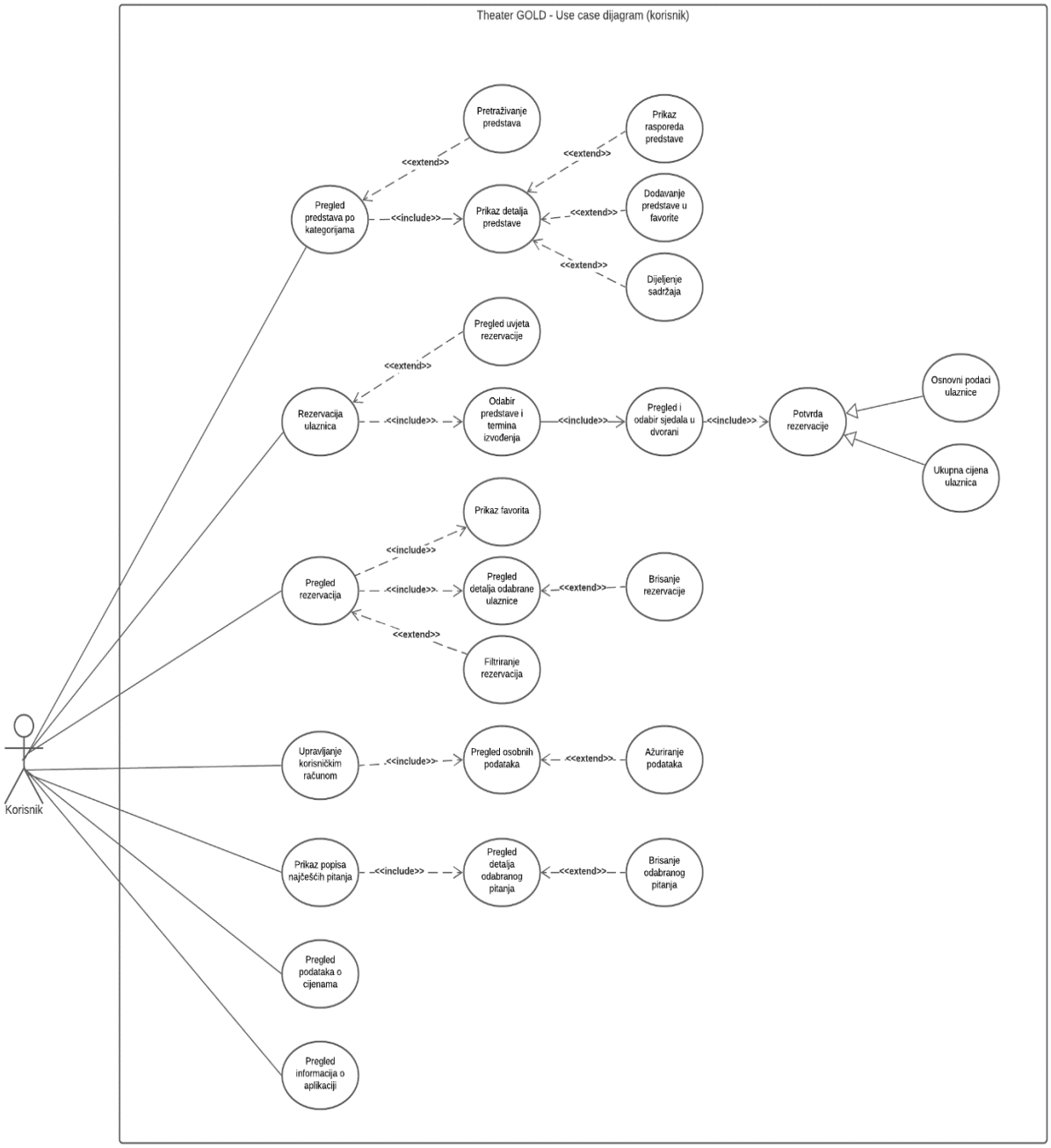


Slika 12. Početni zaslon i pojedinosti predstave (Izvor. Screenshot [30])

5. Funkcionalnosti aplikacije

U sljedećoj cjelini su prikazani i opisani UML dijagrami temeljeni na glavnim funkcionalnostima aplikacije. U prvom dijelu su prikazani Use Case dijagrami za administratora i korisnika. Drugi dio obuhvaća Use Case Sequence dijagrame koji detaljno opisuju pojedine slučajeve uporabe. Na kraju će cjelokupni dio biti zaokružen s klasnim dijagramom koji na pojednostavljen način prikazuje korištene klase.

5.1 Dijagram slučajeva korištenja

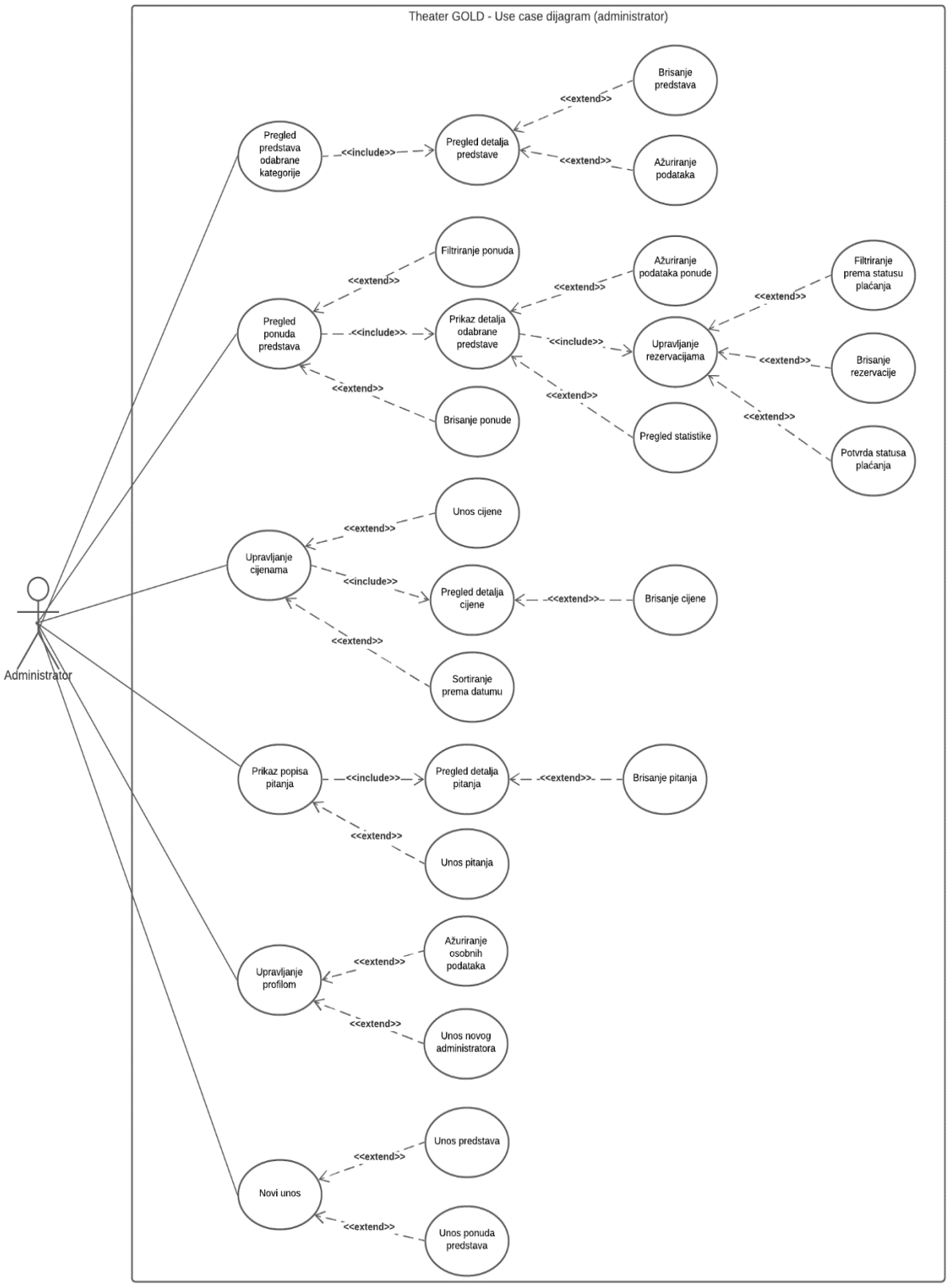


Slika 13. Dijagram slučajeva korištenja – Korisnik

Dijagram slučajeva korištenja (eng. Use Case diagram) se sastoji od dvije glavne tematske cjeline, korisnik i aplikacija. Glavna primjena dijagrama je prikaz definirane interakcije između njih. U nastavku teksta su pružene upute, te mogućnosti aplikacije. Korisnik predstavlja svaku punoljetnu fizičku osobu koja ispunjavanjem podataka u registraciji postaje punopravni član organizacije. Registracijom mu je dodijeljena uloga klijenta koji prihvaća uvjete rada. Korisnik (akter) je vanjski entitet povezan sa sustavom i pokretač svih akcija. Svaki aktivni, registrirani korisnik može: pregledavati, rezervirati i otkazivati rezervirane ulaznice. Dužan je čuvati osobne podatke koji su upisani prilikom regsitracije ili ažuriranja. Sudionicima se dodjeljuju imena koja ne bi smjela biti povezana s organizacijom poduzeća i ona su prikazana pojednostavljenim prikazom ikone čovjeka. Korisnik aplikacije treba upoznati njenu funkcionalnost.

Tablica 3. Pregled i opis funkcionalnosti korisnika u aplikaciji

NAZIV FUNKCIONALNOSTI	OPIS
PREGLED PREDSTAVA	Pretraživanje kategorija, dodavanje u favorite, dijeljenje sadržaja s prijateljima, pregled datuma izvođenja.
REZERVACIJA ULAZNICA	Odabir predstave za rezervaciju ulaznica, pregled detalja, odabir količine ulaznica i sjedala u dvorani, spremanje rezervacije.
PREGLED REZERVACIJA	Pregled detalja i otkazivanje rezerviranih ulaznica, pregled favorita, filtriranje favorita po datumu.
KORISNIČKI RAČUN	Pregled, izmjena/ažuriranje osobnih korisničkih podataka.
CIJENE ULAZNICA	Pregled najnovijih cijena prema datumu.
NAJČEŠĆA PITANJA	Pregled čestih pitanja vezanih za rezerviranje karata i njihovo otkazivanje.
INFORMACIJE O APLIKACIJI	Pregled tehničkih detalja aplikacije.



Slika 14. Dijagram slučajeva korištenja – Administrator

Administrator predstavlja glavno i odgovorno lice koje posjeduje sva prava na aplikaciju. Osigurava obavljanje administratorsko-tehničkih poslova koji obuhvaćaju osiguranje podataka na mobilnoj aplikaciji i održavanja informatičke infrastrukture

kazališta. Oni moraju poznavati konfiguraciju računala, bazu podataka i općenito sve mogućnosti aplikacije. Unos novog administratora u sustav se odobrava tek nakon što polaznik zadovolji sve uvjete za izvršavanje poslova. Osobine svakog administratora su: dobra komunikacija, samostalni rad, poštivanje dogovorenih rokova, te učinkovito upravljanje velikim brojem kazališnih projekata.

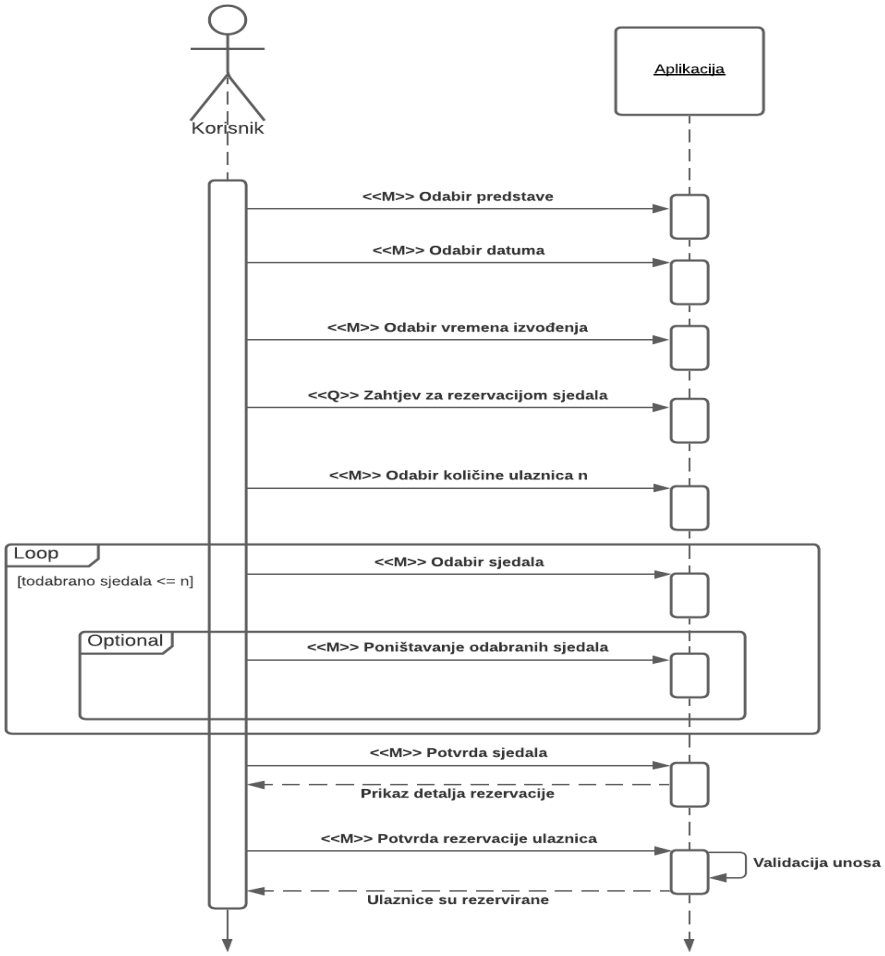
Prije crtanja treba razmisliti o cilju koji se želi postići dijagramom, potrebno je definirati želje administratora u pogledu aplikacije. Tijekom crtanja bitno je naznačiti granice sustava kako bi se znala sustavna pripadnost. Potom se realizira rješenje koje na najbolji način prikazuje željene funkcionalnosti. Administrator inicira komunikaciju i ujedno je glavni korisnik predviđenog sustava. Na raspolaganju ima šest glavnih funkcionalnosti tablično predstavljene.

Tablica 4. Pregled funkcionalnosti administratora u aplikaciji

NAZIV FUNKCIONALNOSTI	OPIS
PREGLED PONUDA	Pretraživanje prema kategorijama, pregled i izmjena/ažuriranje podataka predstave
REZERVIRANE KARTE	Filtriranje, pregled i brisanje ponuda, izmjena/ažuriranje podataka ponude, filtriranje, pregled, brisanje rezervacija, izmjena/ažuriranje statusa plaćanja, pregled grafičkih podataka (statistika).
NOVE PONUDE	Unos novih predstava i pripadnih ponuda
UPRAVLJANJE CIJENAMA	Pregled svih cijena, dodavanje nove cijene, brisanje postojeće cijene
POSTAVKE PROFILA	Pregled, izmjena/ažuriranje osobnih administratorskih podataka, dodavanje novog administratora u sustav
UPRAVLJANJE PITANJIMA	Unos, pregled i brisanje pitanja

5.2 Dijagram slijeda obrazaca uporabe

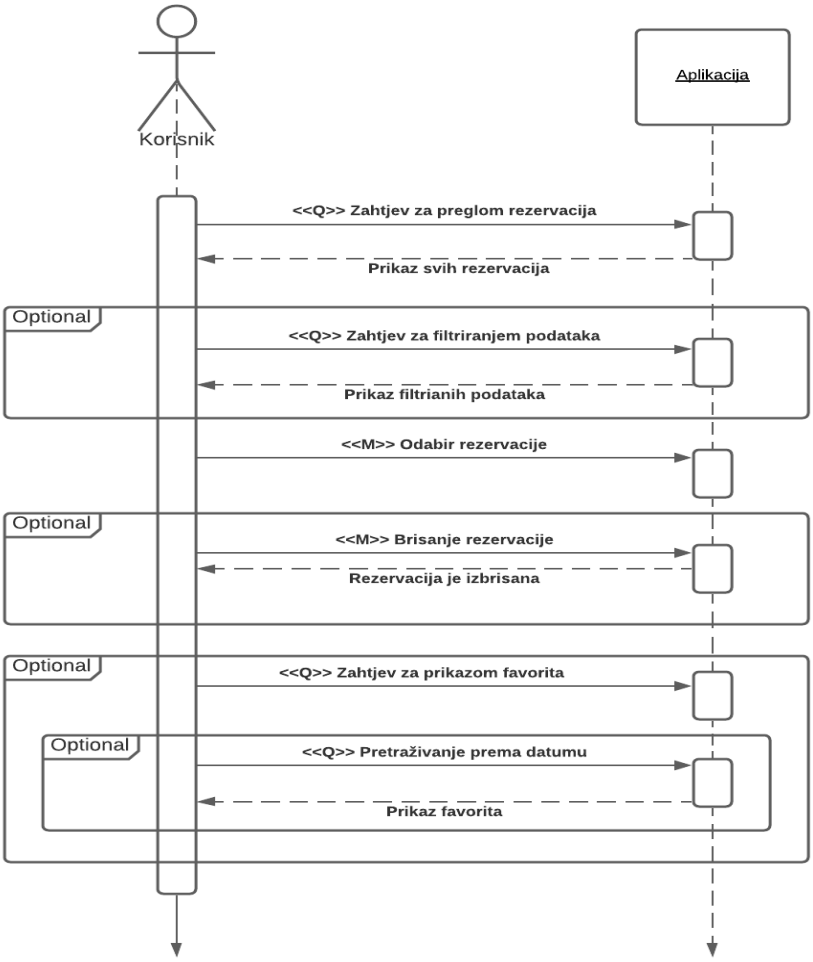
Zadatak je, uz pomoć dijagrama slijeda obrazaca uporabe (eng. Use Case Sequence diagram), detaljno predstaviti i opisati interakciju između korisnika i aplikacije. Komunikacija se može izraziti u obliku slanja poruka odnosno aktiviranja funkcionalnosti. Glavni elementi dijagrama su: korisnik, vremenski pravac, te aplikacija. Postoje dva glavna tipa poruka koji se koriste prilikom izrade, Query i Mutation event. *Query event* se koristi u obliku zahtjeva za prikaz određenih informacija. *Mutation event* omogućuje promjene podataka unutar samog sustava. Vertikalni raspored poziva određuje njihov vremenski redoslijed. U nastavku slijede dijagrami s glavnim ulogama administratora i korisnika.



Slika 15. Sekvencijski dijagram rezervacije ulaznica

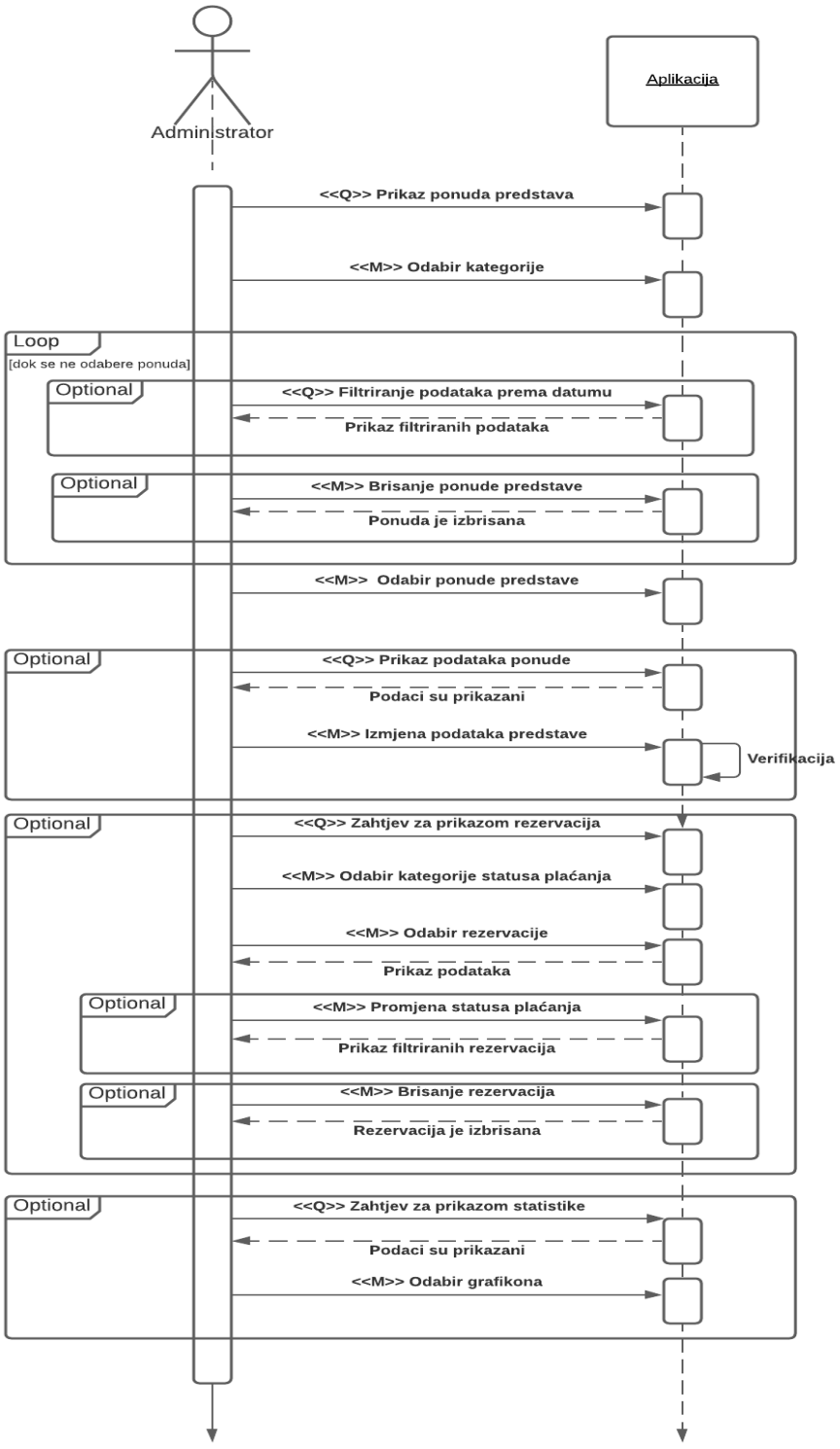
Rezervacija ulaznica predstavlja glavnu korisničku funkcionalnost aplikacije. Korisnik na početku bira predstavu, datum i vrijeme izvođenja predstave koju želi pogledati. U nastavku je potrebno odabrati količinu ulaznica koja se želi rezervirati. Nakon toga

korisnik bira sjedala s obzirom na njihovu zauzetost unutar dvorane. U svakom trenutku korisnik može poništiti označeno sjedalo. Proces odabira sjedala se izvršava onoliko puta koliko korisnik naznači rezervaciju. Nakon odabira broja sjedala prelazi se na završni korak rezervacije. Potvrdom odabranih sjedala, otvara se novi prozor unutar kojeg se nalaze detalji rezervacije. Pregledom svih detalja, korisnik pokreće proces rezervacije i aplikacija daje odgovor o pohrani odabranog.



Slika 16. Sekvencijski dijagram pregleda rezervacija i favorita

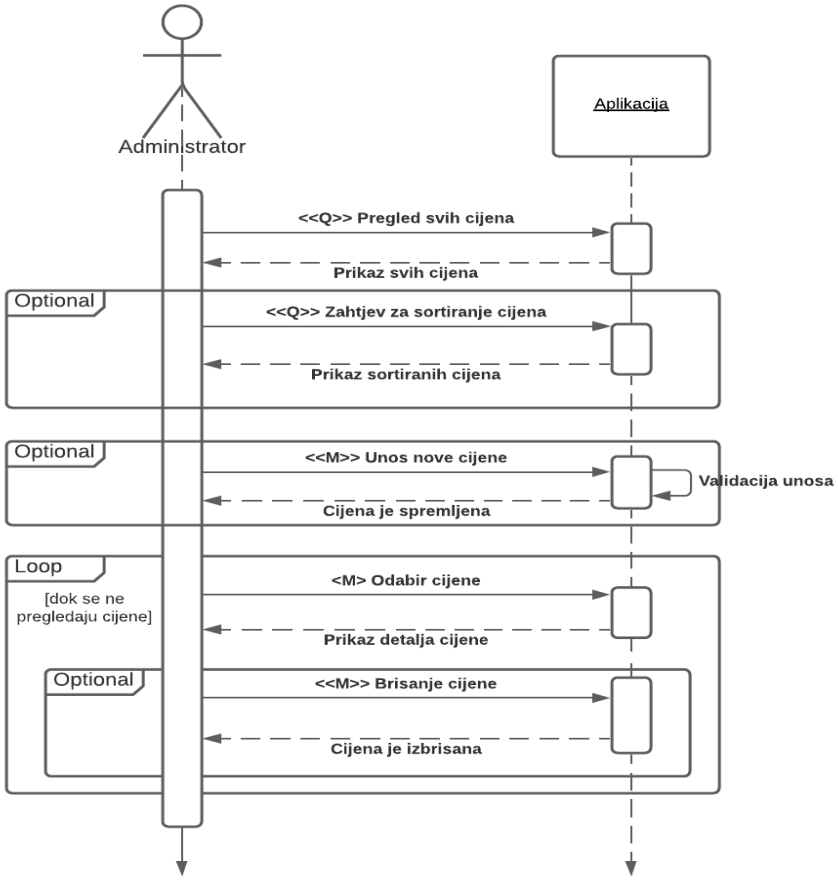
Otvaranjem sučelja za uvid u rezervacije, šalje se zahtjev za prikazom podataka. Aplikacija vraća podatke na traženi upit. Korisnik ima opciju filtriranja rezervacija s obzirom na predstavu i status izvođenja: aktivno, prošlo i prikaz svih rezervacija. Detalji rezervacije se dobivaju odabirom željenih rezervacija. Dodatna opcija obuhvaća pregled favorita koji uključuju popis predstava koje korisnik može pretraživati po datumu. Naknadno se nudi opcija brisanja rezervacije.



Slika 17. Sekvencijski dijagram upravljanja rezervacijama

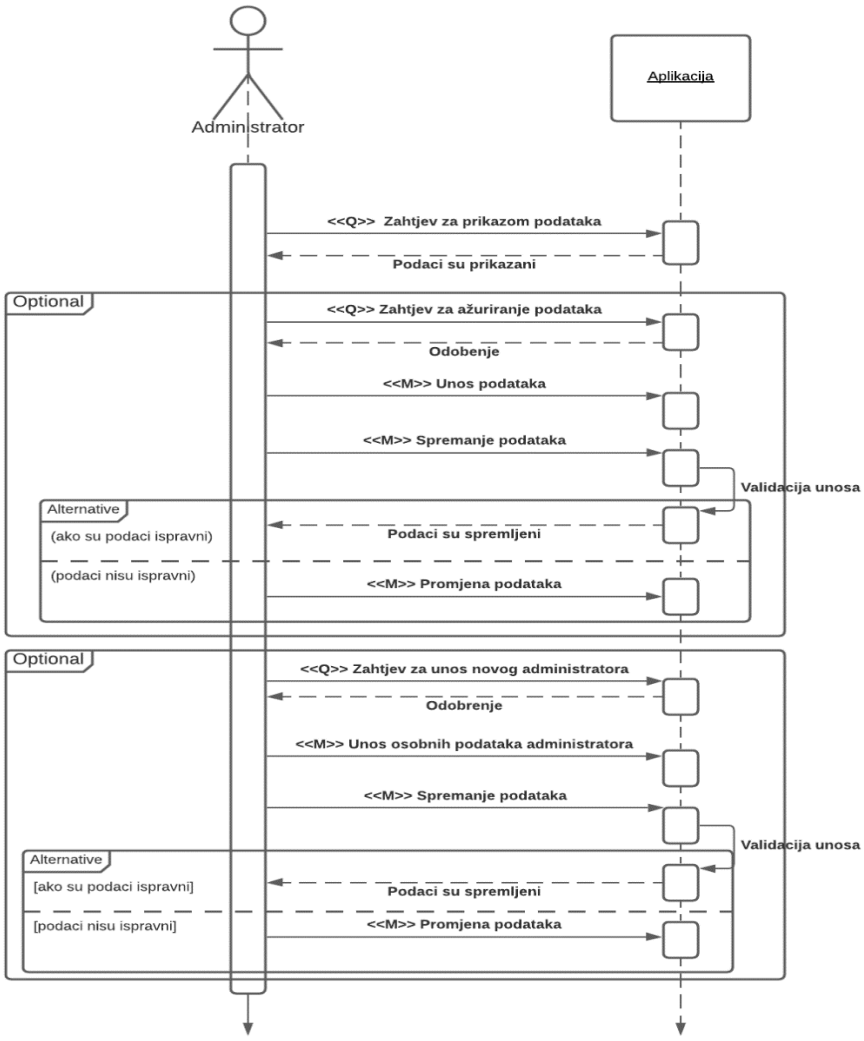
Administrator šalje zahtjev za prikazom podataka vezanih uz raspored ponuda predstava. Prikazane ponude se mogu filtrirati kako bi se došlo do onih ponuda koje su administratoru od interesa. Dodatno, moguće je izbrisati ponudu i sve pripadne rezervacije. Za nastavak je potrebno odabrati jednu ponudu. Odabirom ponude

predstave otvara se prozor koji omogućuje traku s tri glavne opcije: pregled i ažuriranje podataka ponude, pregled i ažuriranje podataka rezerviranih ulaznica i pregled statistike. Pregledavanje se može izvršavati onoliko puta koliko to administrator želi. Pregled ponude omogućuje prikaz svih osnovnih podataka. Iste je moguće promijeniti, posebno pozornost usmjeriti na definiranje datuma i vremena izvođenja predstave. Pregled rezerviranih ulaznica obuhvaća sučelje namijenjeno jednostavnom upravljanju rezervacija. U slučaju korisničke želje za pokazom rezervacije, potrebno je poslati zahtjev za prikazom podataka. Dobivene podatke je tada moguće pregledavati, izbrisati, te ažurirati podatak o statusu plaćanja ulaznice. Radi lakšeg pregleda, može se odabrati kategorija prema statusu plaćanja. Na taj način bi se određenim filterom prikazivale samo one rezervacije koje su vezane uz odabrani status. Ukoliko administrator želi prikazati samo rezervacije koje su podignute i plaćene, potrebno je odabrati kategoriju: *„Da – plaćeno“.* Zadnja opcija se odnosi na pregled podataka uz pomoć statistike koja ima namjenu grafički predočiti količinu rezerviranih ulaznica koristeći različite dinamične grafikone.



Slika 18. Sekvencijski dijagram obrade cijena

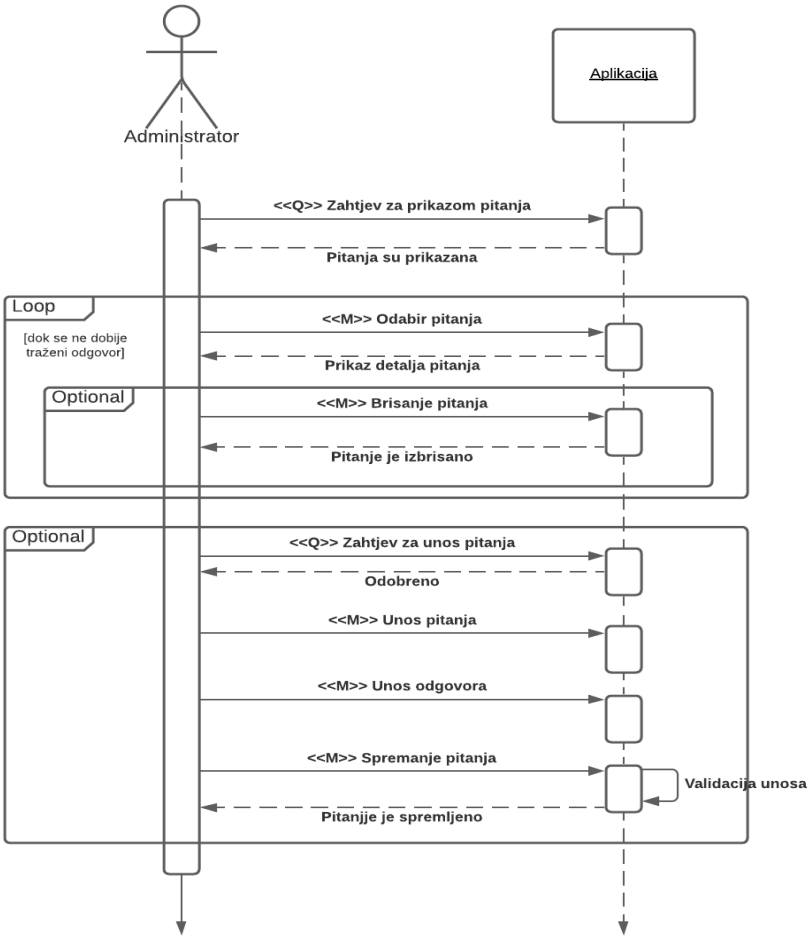
Administrator šalje zahtjev za pregledom cijena ulaznica. Pregled se izvodi onoliko puta koliko administrator želi. Odabirom jedne od ponuđenih cijena, prikazuju se detalji odabrane cijene. Opcionalno, moguće je izbrisati otvorenu cijenu. Osim pregleda, omogućen je unos novih cijena i sortiranje prema datumu. Prilikom unosa je potrebno popuniti sva polja različitih kategorija cijena. Prije spremanja podataka administrator provjerava njihovu točnost i pravilnost. Izvršenim procesom spremanja, aplikacija vraća poruku o uspješnom spremanju.



Slika 19. Sekvencijski dijagram upravljanja računom

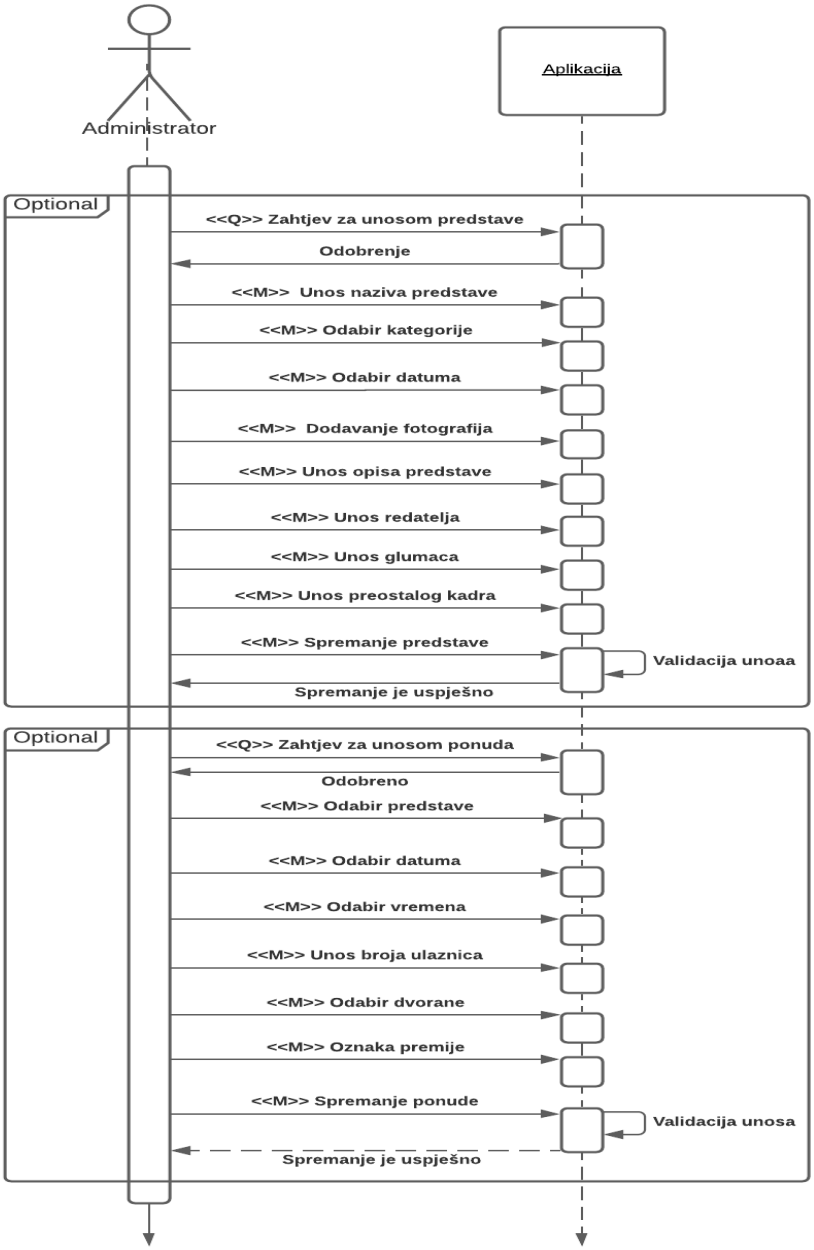
Upravljanje računom obuhvaća dvije glavne funkcionalnosti koje se mogu po potrebi izvršavati, to su: pregled i promjena osobnih podataka, te promjena administratora. U prvom koraku administrator šalje zahjev za prikazom osobnih podataka. Aplikacija dohvaća podatke i prosljeđuje administratoru. U slučaju ažuriranja podataka šalje se novi zahjtev koji omogućava izmjenu podataka. Odobrenjem izmjene, unose se novi

podaci. Prije spremanja provjerava se ispravnost upisanih podataka kako je unaprijed definirano. Spremanjem podataka vraća se poruka administratoru. U suprotnom, ako nisu spremljeni, administrator mora podatke naknadno izmjeniti i spremiti. Promjena administratora je dodatna mogućnost koju aplikacija nudi. Ako se administrator odluči na takav korak potrebno je unijeti odgovarajuće podatke novog administratora. Prije svakog spremanja se provjerava valjanost podataka.



Slika 20. Sekvencijski dijagram obrade najčešćih pitanja

Administrator šalje zahtjev za prikazom svih pitanja. Aplikacija zatim prikazuje pitanja koja postoje. Kako bi se pregledali detalji pitanja, potrebno je odabrati jedno pitanje. Otvaranjem pitanja, sustav prikazuje njegove detalje i nudi mogućnost brisanja istog. Osim pregledavanja pitanja, moguće je unijeti novo pitanje. Administrator šalje zahtjev za unosom, te unosi podatke vezane za naslov pitanja i pripadni odgovor. Prije potvrde spremanja pitanja, aplikacija provjerava popunjenost polja i ako je sve u redu sprema pitanje i vraća poruku o pohranjenosti.

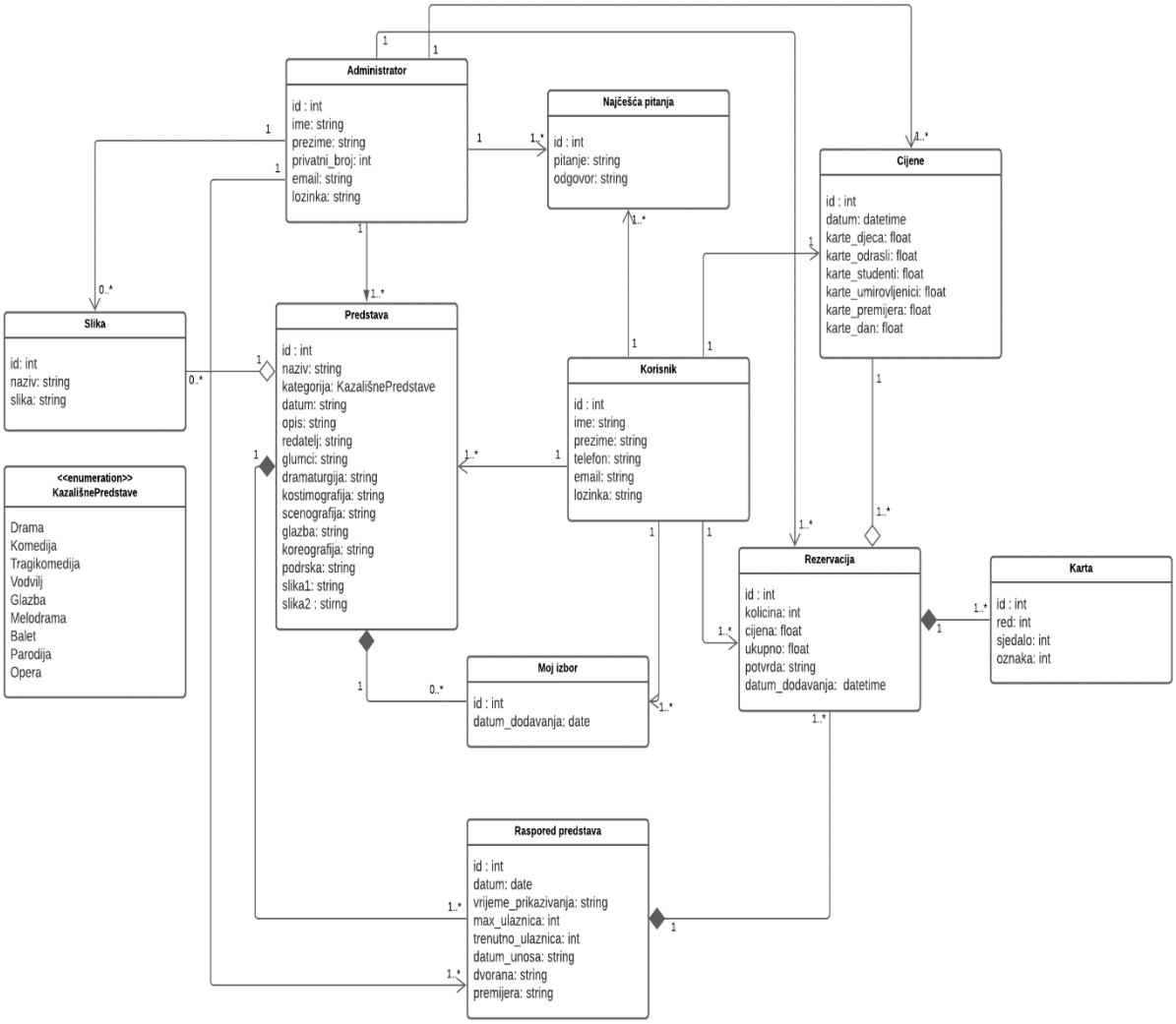


Slika 21. Sekvencijski dijagram unosa predstave i ponude

U ovom dijelu administrator unosi predstave i njihove pripadne ponude. Prilikom upisa, potrebno je unijeti sve podatke koji su vezani za njeno izvođenje i tehničku podršku. Podaci obuhvaćaju unos: kategorije, datuma premijere, fotografije, opis predstave, redatelja, glumce i preostalog tehničkog kadra. Unosom potrebnih podataka, vrši se provjera i spremanje podataka, te aplikacija vraća poruku. Opcija dodavanja ponude predstave sadrži unose podataka: naziva predstave, datuma i vremena izvođenja, maksimalnog broja ulaznica za prodaju i dvorane za izvedbu. Spremanjem se provjerava ispravnost upisanih podataka. Shodno tome, aplikacija vraća poruku o statusu spremanja.

5.3 Klasni dijagram

Klasni dijagram je UML dijagram koji prikazuje međusobne odnose i interakciju među klasama u razvijenome sustavu. Svaka klasa u posjedu sadrži atribut i tip podataka. Atributi pobliže opisuju klasu, dok su tipovi podataka namijenjeni spremanju različitih podataka. Kardinalnost pomaže u dočaravanju veze između klasa, a odnosi su realizirani: asocijacijom, agregacijom, te kompozicijom.



Slika 22. Prikaz klasnog dijagrama

Sve klase prikazane dijagramom se mogu pronaći unutar baze podataka. Iz ponuđenog se može zaključiti da je za potrebe rada aplikacije potrebno koristiti deset klasa. Glavnu ulogu u sustavu imaju korisnik i administrator. Njihove tablice sadrže osnovne podatke koji opisuju korisnika aplikacije. Tablica „Korisnik“ sadrži osnovne podatke o korisniku. Podaci o korisniku su zabilježeni prilikom registracije. Tablica je

direktno povezana s tablicom „Rezervacija“. Na temelju kardinalnosti korisnik može rezervirati jednu ili više ulaznica za predstavu.

Tablica „Rezervacija“ predstavlja ključnu tablicu klasnog dijagrama. Osim standardnih podataka sadrži i podatke vezane za raspored predstava, cijene rezerviranja, te popis rezerviranih sjedala. Tablica „Karta“ je izvedena tablica koja sadrži popise rezerviranih sjedala odgovarajuće rezervacije. Njeno postojanje nema smisla bez tablice „Rezervacija“. Kompozicijom je realizirano da se brisanjem rezervacija direktno brišu i određeni retci s jednakim ključevima u tablici „Karta“. Dodatno je naznačena jedna cijena rezervacije na temelju koje se izračunava ukupna cijena.

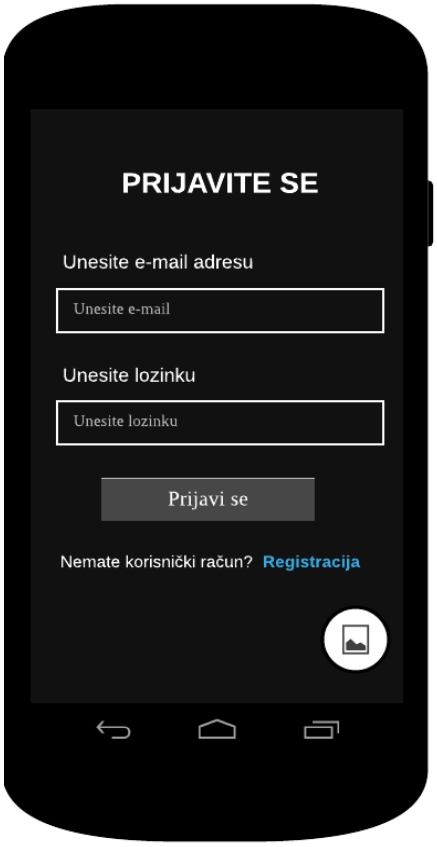
Korisnik po želji dodaje jednu ili više predstava kao favorite. Broj favorita odgovara jednoj predstavi. Brisanjem određene predstave, brišu se i pripadni favoriti. Omogućen je pregled pisanih i slikovnih detalja pojedine predstave. Dodatno je pružen pregled cijena i najčešćih pitanja korisnika.

Administrator može dodavati jednu ili više predstava i njihovih pripadnih ponuda. Svaka predstava ima točno jednu kategoriju kojoj pripada, a prikazana je uporabom enumeracije. Brisanjem redaka iz tablice „Predstava“ brišu se povezane ponude iz tablice „Raspored_predstava“. Upravljanje jednom ili više rezervacija predstavlja jednu od glavnih administratorskih funkcionalnosti. Također je administratoru omogućen pregled i unos najčešćih pitanja i cijena.

5.4 Prototipiranje korisničkog sučelja

U ovom dijelu je prikazan prototip aplikacije koji pruža uvid u njezine funkcionalnosti. Prototip se definira kao preliminarni vizualni model koji izgleda kao prava aplikacija, pokazuje temeljni dizajn i funkciju aplikacije, ali ne sadrži programski kod. [5] Bitno je naglasiti kako prototip ne mora izgledati u potpunosti jednako kao završna aplikacija. Moguće je izostaviti neki element koji će kasnije postojati u konačnoj verziji. Prototip se najčešće koristi za predstavljanje aplikacije potencijalnim investitorima koji su spremni uložiti novac u njen budući razvoj. Za potrebe izrade koristila se platforma Lucidchart. U nastavku je izdvojeno nekoliko bitnih ekrana.

Pokretanjem aplikacije, otvara se prozor za prijavu korisnika. Korisnik popunjava polja e-mail adrese i lozinke. Aplikacija dodatno pruža: registraciju korisnika, te prelazak na dio za administratore. Prijavom u sustav moguće je koristiti sve opcije aplikacije. Pretraživanje i pregled predstava pružaju uvid u detalje odabrane predstave. Dodatno, korisnik može dodati predstavu u favorite ili je podijeliti s prijateljima.



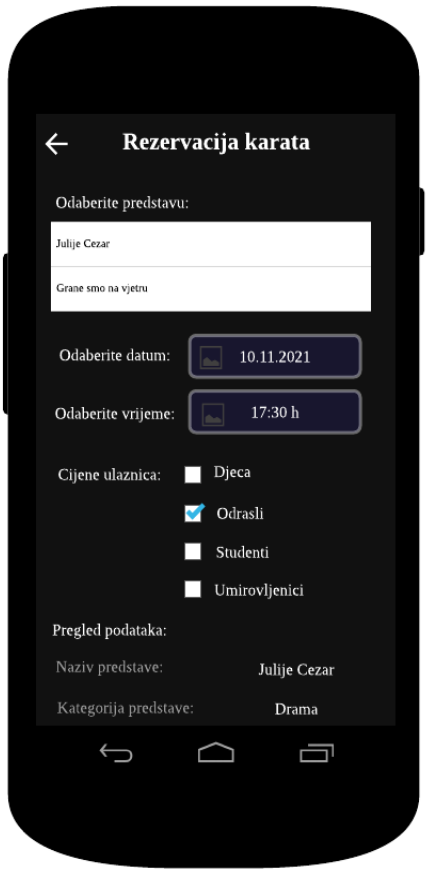
Slika 23. Prototip prijave korisnika

Slika 24. Prototip detalja predstave

Rezervacija ulaznica zahtjeva od korisnika odabir predstave, datuma i vremena izvođenja. Dodatno se odabire dobna kategorija radi obračuna cijene rezervacije.

Potvrdom trenutnih podataka, korisnik mora odabrati sjedala u kazališnoj dvorani koju želi rezervirati. Zadnji korak rezervacije obuhvaća: dodatni pregled svih prethodnih podataka, prikaz obračuna cijena rezervacije, te ukupni iznos. Klikom na gumb „Rezerviraj“ potvrđuju se svi uvjeti kupnje, te su ulaznice uspješno rezervirane.

Rezervirane ulaznice je moguće pronaći u pregledu rezervacija. Sve prikazane rezervacije se filtriraju prema datumu. Odabirom jedne od njih, otvara se novi prozor koji prikazuje pripadne detalje i omogućuje otkazivanje/brisanje rezervacije.



Slika 25. Prototip odabir podataka

Slika 26. Prototip rezervacija ulaznica

Opcija upravljanja korisničkim računom omogućava korisniku pregled i izmjenu osobnih podataka uključujući opciju ažuriranja. Korisnik popunjava ona polja za koja želi izmjeniti podatke. Na kraju mu se nudi opcija zadržavanja postojeće e-mail adrese koju potvrđuje označavanjem kvačice. Ukoliko ne želi zadržati adresu, upisuje novu. Spremanje se izvršava klikom na gumb „Spremi“.

Slika 27. Prototip prikaza rezervacija

Slika 28. Prototip upravljanja računom

Administratorski dio obuhvaća opcije: upravljanje rezervacijama, pregled i unos cijena, pregled i izmjena podataka predstave, unos i brisanje pitanja. Administratori se prijavljuju u sustav posebnim sučeljem. Nakon toga mogu u potpunosti koristiti sve funkcionalnosti aplikacije. Prototipom su prikazana sučelja unosa i pregleda ponuda.

Slika 29. Prototip unosa ponude

Slika 30. Prototip prikaza ponuda

Odabirom jedne ponude prikazuju se u novom prozoru detalji ponude. Osim detalja moguće je pregledati rezervirane ulaznice za tu predstavu, te prikazati statistiku preko različitih grafikona. Popis rezervacija obuhvaća: pregled svih rezerviranih ulaznica, pregled statusa plaćanja, te po potrebi brisanje/otkazivanje rezervacije. Odabirom jedne rezervacije otvara se novi prozor koji prikazuje detalje rezervirane ulaznice. Detalji se odnose na sve podatke koji su vezani uz rezervaciju. Administrator može ažurirati status plaćene ulaznice, te izbrisati rezervaciju.

Opcija *„Unos cijene“* omogućuje pregled, brisanje i unos novih cijena u sustav. Na temelju cijena se izračunava ukupni iznos rezervacija ulaznica. Unos cijena je realiziran skočnim prozorom koji se otvara kada korisnik odabere ikonu unosa iz alatne trake. Prilikom unosa potrebno je unijeti cijene za šest različitih kategorija. Administrator može izbrisati cijene koje su zastarjele i nemaju se više namjeru koristiti. Klikom na gumb *„Spremi cijene“* sve unešene cijene se spremaju u bazu podataka.

Slika 31. Prototip prikaza detalja rezervacije

Slika 32. Prototip unosa cijene

6. Implementacija i korištene tehnologije

Temeljem izrađenih dijagrama i korisničkog sučelja, razvija se aplikacija s tehnologijama za rad. Mobilna aplikacija za spremanje podataka koristi relacijsku bazu podataka. Komunikacija aplikacije s bazom zahtjeva izradu pozadinskog sustava. U ovom dijelu rada je prikazana i kratko opisana implementacija potrebnih tehnologija.

6.1 Implementacija baze podataka

Za izradu aplikacije koristilla se MySQL relacijska baza podataka. Relacijske baze koriste strukturu koja omogućuje identificiranje i pristup podacima baze podataka koji su organizirani u tablice. [8] MySQL koristi strukturirani jezik upita SQL (eng. Structured Query Language) koji ujedno predstavlja primarni jezik komunikacije s bazama podataka. Uporabom baze evidentiraju se informacije vezane za kazalište, a upitima jednostavniji i brži dolazak do njih.

Tablice odnosno relacije mogu sadržavati jednu ili više kategorija podataka u stupcima ili atributima. Svaki redak sadrži jedinstveni primarni ključ, a svaki stupac sadrži atribute. Odnos između tablica se postavlja uporabom stranih ključeva koji imaju ulogu povezivanja s primarnim ključevima drugih tablica. Korištenjem relacijskih baza olakšava se soritranje i organizacija podataka što pomaže organizacijama u donošenju poslovnih odluka. [7] U bazama nije moguće imati dva primarna ključa, dozvoljen je samo jedan, preostali atributi mogu biti kandidati ključa. Korisnici aplikacije imaju kontrolirani pristup čitanju podataka i dijelu uvođenja promjena. Administratori posjeduju oba prava u obliku neograničenih pristupa. Uvođenjem integriteta osigurava se korisnicima unos ispravnih podataka unaprijed definiranim ograničenjima. Prilikom izrade baze potrebno je procijeniti potrebna ograničenja koja se mogu zamijeniti odgovarajućim provjerama u aplikaciji. Većina atributa ima klauzulu NOT NULL, što znači da niti jedan podatak ne smije biti prazan, već mora biti ispunjen s vrijednošću. Ograničenje ON DELETE CASCADE briše određene retke iz tablice djeteta kada se izbriše redak iz tablice roditelja. Prilikom brisanja gleda se podudaranost ključeva tablice roditelja i djeteta. Preostali primjeri ograničenja su: tipovi podataka, jedinstvenost ključeva, indeksiranje, itd. Na sve primarne ključeve je postavljeno automatsko povećanje prilikom unosa novog reda.

Pristup bazi podataka se realizira korištenjem alata MySQL Workbench. Spajanje na bazu zahtjeva otvaranje nove konekcije gdje je potrebno ispuniti obrazac s podacima.

Podaci za vezu su dobiveni prilikom instalacije odnosno stvaranjem baze na udaljenom poslužitelju. Uspješnim povezivanjem omogućuje se korištenje. Spremanje informacija se postiže kreiranjem i korištenjem deset baznih tablica rezervacije ulaznica. Sve tablice i upiti su izrađeni i definirani unutar *backenda*.

Tablica 5. Pregled popisa tablica u bazi podataka

Popis tablica	
Administrator	Korisnik
Predstava	Raspored_predstava
Moj izbor	Cijene
Rezervacija	Karta
Najčešća pitanja	Slika

1. Administrator: sadrži popis osobnih podataka administratora. Nad svim podacima osim broja telefona je postavljeno ograničenje NOT NULL čime se postiže atributsko sadržavanje neke vrijednosti. Ograničenje na broj nije postavljeno zbog mogućnosti da kazalište nije dodijelilo službeni broj telefona novom administratoru.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
ime	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐
prezime	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐
privatni_broj	INT	☐	☐	☐	☐	☐	☐	☐
email	VARCHAR(60)	☐	☑	☐	☐	☐	☐	☐
lozinka	VARCHAR(60)	☐	☑	☐	☐	☐	☐	☐

Slika 33. Prikaz tablice „Administrator“

2. Korisnik: tablica sadrži popis osobnih podataka korisnika. Unutar tablice su svi znakovni podaci definirane duljine, a najvažniji su podaci e-mail adrese i lozinke s kojima se korisnik prijavljuje u aplikaciju. Lozinka je kriptirana s SHA-256 kriptografskom hash funkcijom čime se sprječava krađa podataka od strane neovlaštenih korisnika. Nad svim podacima je postavljena ograničenje NOT NULL.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
ime	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐
prezime	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐
telefon	VARCHAR(10)	☐	☑	☐	☐	☐	☐	☐
email	VARCHAR(60)	☐	☑	☐	☐	☐	☐	☐
lozinka	VARCHAR(60)	☐	☑	☐	☐	☐	☐	☐

Slika 34. Prikaz tablice „Korisnik“

3. Predstava: sadrži sve osnovne i tehničke podatke vezane uz predstavu. Svih petnaest atributa je predstavljeno s VARCHAR tipom podataka i nad svima je postavljeno ograničenje NOT NULL. Unutar tablice nema stranih ključeva.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
naziv	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
kategorija	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
datum	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
opis	VARCHAR(1000)	☐	☑	☐	☐	☐	☐	☐
redatelj	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
glumci	VARCHAR(200)	☐	☑	☐	☐	☐	☐	☐
dramaturgija	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
kostimografija	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
scenografija	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
glazba	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
koreografija	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
podrska	VARCHAR(80)	☐	☑	☐	☐	☐	☐	☐
slika1	VARCHAR(300)	☐	☑	☐	☐	☐	☐	☐
slika2	VARCHAR(300)	☐	☑	☐	☐	☐	☐	☐

Slika 35. Prikaz tablice „Predstava“

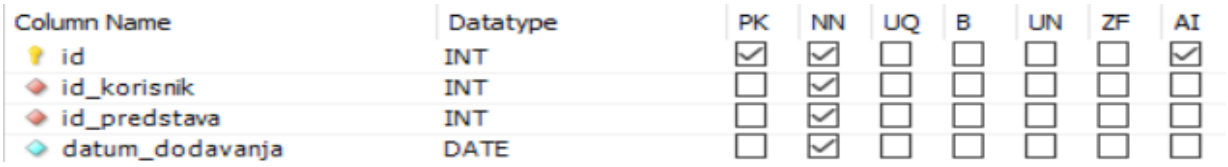
4. Raspored predstava: tablica sadrži podatke vezane uz raspored izvođenja predstava. Atribut datuma je predstavljen DATETIME tipom podataka. Podaci o maksimalnim i trenutnim ulaznicama su realizirani s INT tipom. Preostali podaci se odnose na atribute dvorane i premijere predstave. U tablici postoje dva strana ključa koja su povezana s tablicama *„Predstava“* i *„Korisnik“.* Tablica nema smisla postojanja bez tablice *„Predstava“* jer se na temelju njenih podataka kreiraju rasporedi izvođenja pripadnih predstava. Nad svim atributima je postavljeno ograničenje NOT NULL.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
id_predstava	INT	☐	☑	☐	☐	☐	☐	☐
id_korisnik	INT	☐	☑	☐	☐	☐	☐	☐
datum	DATE	☐	☑	☐	☐	☐	☐	☐
datum_prikazivanja	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
vrijeme_prikazivanja	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐
max_ulaznica	INT	☐	☑	☐	☐	☐	☐	☐
trenutno_ulaznica	INT	☐	☑	☐	☐	☐	☐	☐
datum_unosa	VARCHAR(50)	☐	☑	☐	☐	☐	☐	☐
dvorana	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐
premijera	VARCHAR(2)	☐	☑	☐	☐	☐	☐	☐

Slika 36. Prikaz tablice „Raspored_predstava“

5. Moj izbor: sadrži podatke o predstavama koje je korisnik postavio na listu želja za gledanje. Korisnika podsjeća na pregled detalja predstave koje je označio na listi. U tablici postoje dva strana ključa koja su povezana s tablicama *„Korisnik“* odnosno *„Predstava“.* Korištenjem stranih ključeva postiže se dosljednost koja održava bazu

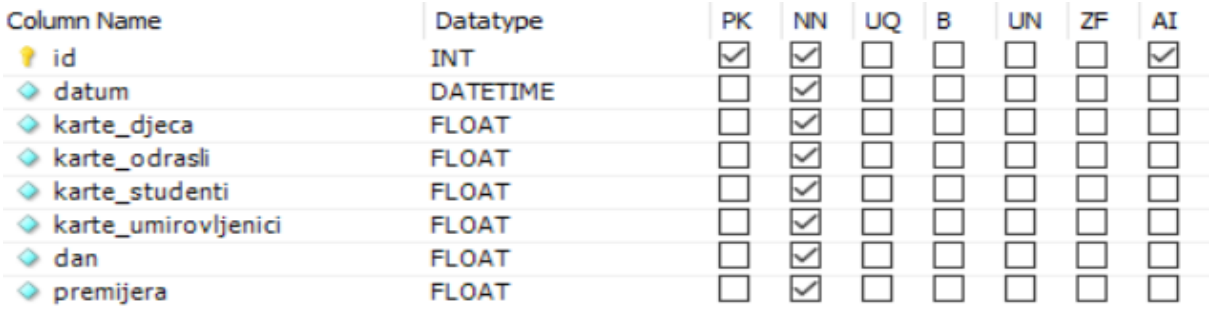
podataka čistom. Atribut datuma dodavanja označava datum kada je predstava dodana u bazu, nad atributom je postavljeno ograničenje NOT NULL.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
id_korisnik	INT	☐	☑	☐	☐	☐	☐	☐
id_predstava	INT	☐	☑	☐	☐	☐	☐	☐
datum_dodavanja	DATE	☐	☑	☐	☐	☐	☐	☐

Slika 37. Prikaz tablice „Moj izbor“

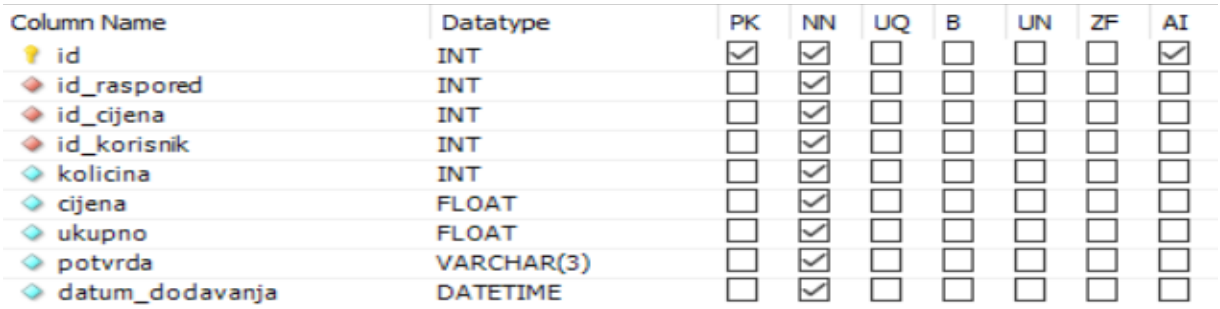
6. Cijene: tablica sadrži podatke za šest kategorija cijena. Od toga se četiri kategorije odnose na cijene s obzirom na dobnu granicu. Preostala dva se odnose na naknade za rezervaciju ulaznica na dan izvedbe predstave ili premijeru predstave. Zbog preciznosti cijena, koristi se tip podataka FLOAT. Polje datuma u kombinaciji s vremenom je realizirano tipom podataka DATETIME. Tu je ograničenje NOT NULL.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
datum	DATETIME	☐	☑	☐	☐	☐	☐	☐
karte_djeca	FLOAT	☐	☑	☐	☐	☐	☐	☐
karte_odrasli	FLOAT	☐	☑	☐	☐	☐	☐	☐
karte_studenti	FLOAT	☐	☑	☐	☐	☐	☐	☐
karte_umirovljenici	FLOAT	☐	☑	☐	☐	☐	☐	☐
dan	FLOAT	☐	☑	☐	☐	☐	☐	☐
premijera	FLOAT	☐	☑	☐	☐	☐	☐	☐

Slika 38. Prikaz tablice „Cijene“

7. Rezervacija: tablica sadrži podatke o rezerviranim ulaznicama. Tu su podaci o količini, cijenama ulaznica i ukupnom iznosu. Količina je predstavljena cjelobrojnim INTEGER tipom podataka. Cijene i ukupni iznos koriste FLOAT radi postizanja preciznosti. Atribut datuma se odnosi na točno vrijeme rezervacije ulaznica. U tablici postoje tri strana ključa povezana s tablicama: *„Raspored_ponuda*“, *„Cijena“ i „Korisnik“.* Nad svim podacima je postavljeno NOT NULL ograničenje.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
id_raspored	INT	☐	☑	☐	☐	☐	☐	☐
id_cijena	INT	☐	☑	☐	☐	☐	☐	☐
id_korisnik	INT	☐	☑	☐	☐	☐	☐	☐
kolicina	INT	☐	☑	☐	☐	☐	☐	☐
cijena	FLOAT	☐	☑	☐	☐	☐	☐	☐
ukupno	FLOAT	☐	☑	☐	☐	☐	☐	☐
potvrda	VARCHAR(3)	☐	☑	☐	☐	☐	☐	☐
datum_dodavanja	DATETIME	☐	☑	☐	☐	☐	☐	☐

Slika 39. Prikaz tablice „Rezervacija“

8. Karta: izvedena tablica koja se koristi radi spremanja podataka vezanih za sjedala rezervacije. Jedna rezervacija može sadržavati više sjedala, pa je dobra praksa umjesto ugradnje u tablicu „Rezervacija“ kreirati zasebnu tablicu koja će služiti prikazu detalja rezervacije. Tablica sadrži atribute reda, sjedala i oznake sjedala realiziranih cjelobrojnom numeričkom vrijednosti INT. Strani ključ je povezan s tablicom *„Rezervacija“.* Nad svim podacima je postavljeno NOT NULL ograničenje jer nema smisla postojanje rezervacije bez rezerviranih sjedala.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
id_rezervacije	INT	☐	☑	☐	☐	☐	☐	☐
red	INT	☐	☑	☐	☐	☐	☐	☐
sjedalo	INT	☐	☑	☐	☐	☐	☐	☐
oznaka	INT	☐	☑	☐	☐	☐	☐	☐

Slika 40. Prikaz tablice „Karta“

9. Najčešća pitanja: tablica služi unosu i prikazu pitanja i odgovora vezanih za rad aplikacije. Unutar tablice postoji strani ključ administratora. Polja za pitanje i odgovor su realizirani VARCHAR tipom podataka. Nad svim podacima je postavljeno ograničenje NOT NULL.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
id_administrator	INT	☐	☑	☐	☐	☐	☐	☐
pitanje	VARCHAR(250)	☐	☑	☐	☐	☐	☐	☐
odgovor	VARCHAR(1000)	☐	☑	☐	☐	☐	☐	☐

Slika 41. Prikaz tablice „Najcesca_pitanja“

10. Slika: tablica sadrži naslov i detalje slike kako bi se znalo kojoj predstavi slika pripada. Naslov je realiziran s VARCHAR tipom podataka koji omogućava unos teksta. Unutar tablice je za sve podatke definirano ograničenje NOT NULL.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	INT	☑	☑	☐	☐	☐	☐	☑
naziv	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐
slika	VARCHAR(300)	☐	☑	☐	☐	☐	☐	☐

Slika 42. Prikaz tablice „Slika“

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length
administrator	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes
cijene	InnoDB	10	Dynamic	1	16384	16.0 KiB	0.0 bytes
karta	InnoDB	10	Dynamic	22	744	16.0 KiB	0.0 bytes
korisnik	InnoDB	10	Dynamic	1	16384	16.0 KiB	0.0 bytes
mojizbor	InnoDB	10	Dynamic	3	5461	16.0 KiB	0.0 bytes
najcesca_pitanja	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes
predstava	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes
raspored_predstava	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes
rezervacija	InnoDB	10	Dynamic	10	1638	16.0 KiB	0.0 bytes
slika	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes

Slika 43. Prikaz popisa tablica iz baze podataka

Indexes in Table

Visible	Key	Type	Uni...	Columns
☑	PRIMARY	BTREE	YES	id

Columns in table

Column	Type	Nullable	Indexes
id	int	NO	PRIMARY
naziv	varchar(50)	NO	
kategorija	varchar(50)	NO	
datum	varchar(50)	NO	
opis	varchar(1000)	NO	
redatelj	varchar(50)	NO	
glumci	varchar(200)	NO	
dramaturgija	varchar(50)	NO	
kostimografija	varchar(50)	NO	
scenografija	varchar(50)	NO	
glazba	varchar(50)	NO	
koreografija	varchar(50)	NO	
podrska	varchar(80)	NO	
slika1	varchar(300)	NO	
slika2	varchar(300)	NO	

Slika 44. Struktura tablice predstava u bazi podataka

Don't Limit

```
SELECT p.naziv, p.kategorija, o.id, o.datum_prikazivanja, o.vrijeme_prikazivanja
                FROM database.Predstava p, database.Raspored_predstava o
                WHERE o.id_predstava = p.id AND p.naziv = 'Julije Cezar'
                ORDER BY o.datum ASC
```

Slika 45. Primjer SQL upita dohvaćanja rasporeda prema nazivu predstava

6.2 Implementacija pozadinskog sustava

Zbog potrebe razvoja pozadinskog sustava, koristio se programski jezik JavaScript u kombinaciji s **Node.js** okruženjem. Node.js pokreće V8 JavaScript *engine*, jezgru Google Chroma izvan preglednika. Aplikacija radi u jednom procesu, bez stvaranja nove niti za svaki zahtjev. Njime je moguće obratiti tisuće istovremenih veza s jednim poslužiteljem. [22] Jedan od razloga korištenja ovog okruženja leži u jednostavnosti izrade REST sučelja i upravljanje s zahtjevima. Glavna svrha njegove uporabe je posredničko povezivanje baze podataka s mobilnom aplikacijom putem HTTP metoda.

REST (eng. Representational state transfer) je arhitekturni stil web usluga koje se koriste za razmjenu podataka putem HTTP protokola. Skup API –ja (eng. Application Programming Interface) s principima RESTful arhitekture naziva se RESTful API-ji. To je arhitektura bez stanja jer se veza između klijenta i poslužitelja ne čuva. [23]

Prije početka uporabe, potrebno je preuzeti i instalirati razvojno okruženje. Upraviteljom npm je moguće instalirati i uključivati biblioteke potrebne za izradu pozadinskog sustava. Jedan od najčešće korištenih modula je modul baze podataka. Upravljanje HTTP upitima zahtjeva uporabu Express okvira koji se po završetku instalacije uključuje u projekt. Popis svih instaliranih modula je moguće pronaći u mapi „node_modules“. Unutar datoteke *package.json* se nalaze metapodaci aplikacije.

Glavni dio razvoja *backenda* obuhvaća izradu ruta preko kojih komunicira s mobilnom aplikacijom. Prosljeđivanje parametara unutar HTTP-a se postiže definiranjem vrste, adrese i putanje upita. Komunikacija se realizira pozivom metode s udaljenog poslužitelja. Unutar metode se prosljeđuju parametri (dio URLa) na temelju kojih se izvršava određena vrsta zahtjeva, te se na koncu prima rezultat tj. odgovor upita koji može biti prikazan kao XML ili JSON.

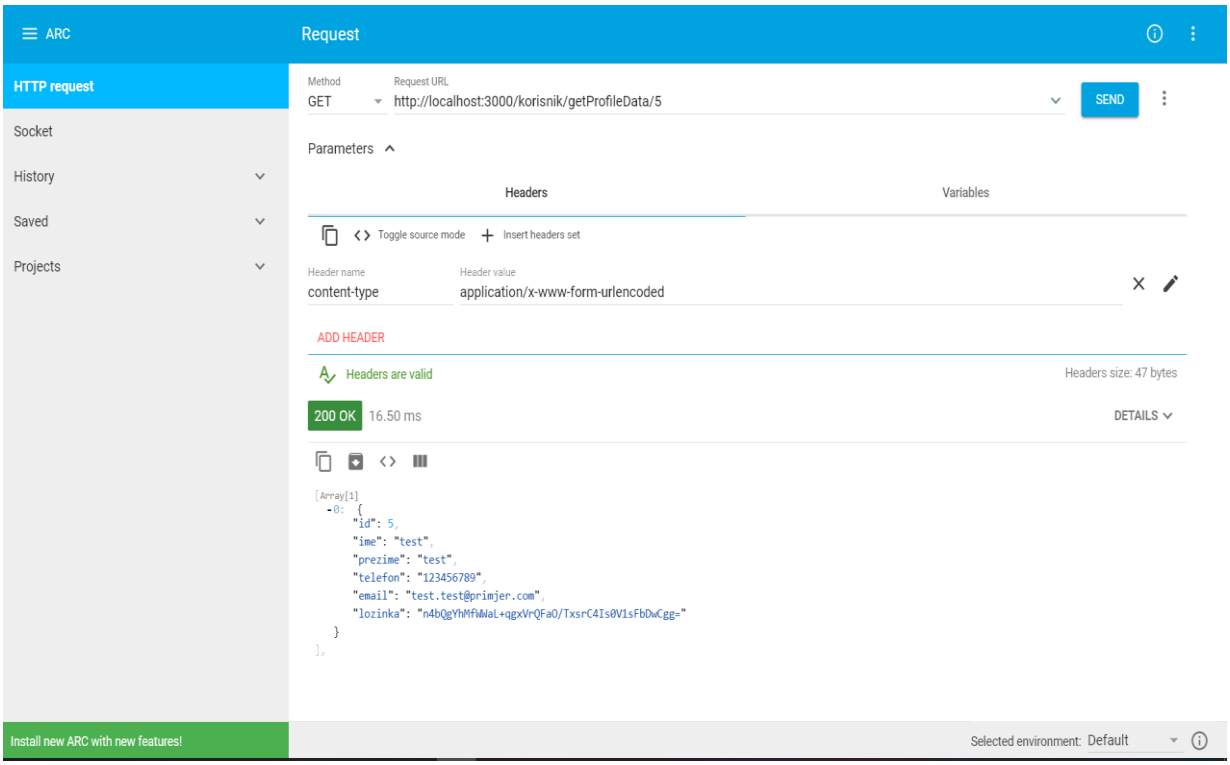
Tablica 6. Tablični prikaz osnovnih korištenih HTTP metoda

Metoda	**Opis**
GET	Dohvaćanje (čitanje) resursa
POST	Izrada resursa
PUT	Modifikacija resursa
DELETE	Brisanje resursa

Prosljeđivanje koristi dinamičke rute kod kojih svaki parametar tvori zaseban dio putanje. Vraćanje odgovora na postavljeni upit u JSON formatu je postignut pozivom metode *„res.json()“*, u obliku teksta s metodom *„res.send()“*.

Primjer procesa prijave korisnika u sustav započinje od mobilne aplikacije i unosa osnovnih podataka o korisniku. Pritiskom na gumb za prijavu se šalje HTTP zahtjev (eng. Request) poslužitelju koristeći REST API. Web poslužitelj REST protokolom omogućuje pristup bazi podataka. Primanjem zahtjeva on prosljeđuje upit bazi. Unutar baze se izvršava obrada podataka shodna postavljenom upitu. U ovom slučaju se pretražuju podaci o korisniku. Završetkom pretraživanja, baza vraća odgovor web poslužitelju, koji u JSON formatu šalje natrag aplikaciji (eng. Response). Mobilna aplikacija pretvara JSON format u Java objekt, te odgovor ispisuje na zaslon mobitela.

Prije izrade mobilne aplikacije potrebno je testirati ispravnost API-ja slanjem HTTP upita. Za takvu potrebu se koristio program „Advanced REST Client“. U primjeru je prikazano dohvaćanje korisničkih detalja na temelju prosljeđenog parametra primarnog ključa (ID) korisnika. Dohvaćanje podataka je realizirano GET metodom, te su rezultati vraćeni u JSON formatu. Kvaliteta ovisi o vremenu odaziva i propusnosti.



Slika 46. Primjer korištenja alata „Advanced REST client“ (Izvor: Screenshot)

6.3 Implementacija Android aplikacije

U ovom dijelu rada je objašnjeno programsko rješenje mobilne aplikacije. Unutar projekta se definira naziv paketa, biblioteke, te klase s aktivnostima. Česti primjeri View elemenata su: gumbi (eng. Button), polja za unos (eng. EditText), polja za prikaz teksta (eng. TextView), komponente izbora (eng. Radio Button), itd. Sučelja su izrađena unutar XML datoteka koje se mogu pronaći u mapi „layout“. U svakoj klasi aktivnosti, metoda „onCreate“ se izvršava prilikom stvaranja aktivnosti, dok se metodom „setContentView()“ postavlja korisničko sučelje.Potpoglavlja su opisana osnovnim implementacijama mobilne aplikacije prema najznačajnijim funkcijama.

6.3.1 Implementacija i korištenje REST servisa

Baza podataka je najvažniji segment rada aplikacije. Komuniciranje s bazom zahtjeva povezivanje mobilne aplikacije s pozadinskim sustavom koji radi na serveru. Pristup internetu je moguće ostvariti dodavanjem dopuštenja u datoteku manifesta.

```
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Programski kod 3. Prikaz dopuštenja unutar datoteke manifest

Postoje standardne i dodatne biblioteke koje omogućuju rad s web servisima. Standardne obuhvaćaju izvorne biblioteke iz Androida (klasa HttpUrlConnection), dok dodatne koriste vanjske biblioteke koje su naknadno ugrađene (Retrofit, Google Volley itd). Rad sa servisima je realiziran dodatnom biliotekom Square Retrofitom (REST klijent) koja sadrži gotove metode razmjene podataka, dok se za obradu koristi GSON.

GSON pretvara JSON objekt u Java objekt koristeći klase modela izrađene za potrebe aplikacije. Prilikom dohvaćanja odgovora, svaki dio treba biti pokriven odgovarajućim klasama iz mape „Model“. Klasa se sastoji od atributa i konstruktora. Prije korištenja, potrebno je dodati biblioteke u Gradle: RxJava, Retrofit2, GSON.

```
implementation 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'

implementation 'com.squareup.retrofit2:converter-scalars:2.3.0'

implementation 'com.squareup.retrofit2:converter-gson:2.2.0'
```

Programski kod 4. Prikaz referenci na Retrofit i GSON biblioteke

Definiranjem klase Builder i URL-a na kojem radi pozadinski sustav omogućuje se slanje zahtjeva API-ju. Ako se za lokalnu uporabu aplikacije koristi fizički uređaj treba paziti na povezanost uređaja i računala u mreži s istom IP adresom. Uporaba emulatora zahtjeva unos IP adrese unutar BASE_URL-a, koja glasi: 10.0.2.2. U slučaju smještaja servera na udaljeni poslužitelj, tada se unosi URL web stranice.

```
private static Retrofit instance = null;

public static Retrofit getInstance2(){
    instance = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
            .addConverterFactory(GsonConverterFactory.create())
            .build();
    return instance;
}
```

Programski kod 5. Primjer kreiranja instance Retrofit objekta GSON biblioteke

6.3.2 Prijava i registracija korisnika u aplikaciju

Aplikacija zahtjeva unos e-mail adrese i lozinke za pristup. Sučelje je realizirano s dva polja za unos podataka, gumba preko kojeg se izvršava proces prijave, gumba za prelazak u administrativni dio, te polja kojim se otvara novi prozor sučelja za registraciju. Prilikom upisa korisnik treba paziti na unos podataka. Ukoliko se preskoči polje ili pogriješi, aplikacija upozorava. U polju unosa podataka se javlja crveni obrub s upozorenjem korisniku o promjeni e-mail adrese. Procesom prijave, aplikacija provjerava ispravnost upisane e-mail adrese ili postojanju iste u bazi podataka. U slučaju postojanja, vraća povratnu informaciju korisniku, te je potrebno unijeti novu. Međutim, ako su uvjeti ispunjeni izvršava se proces prijave korisnika u aplikaciju.

```
private void checkArray(String email, String password){
    if(TextUtils.isEmpty(email)){
        emailText.setError("Unesite e-mail adresu!");
        emailText.setBackgroundResource(R.drawable.edit_text_red_border);
    }else emailText.setBackgroundResource(R.drawable.textfield);

    if(TextUtils.isEmpty(password)){
        passwordText.setError("Unesite lozinku!");
        passwordText.setBackgroundResource(R.drawable.edit_text_red_border);
    }else passwordText.setBackgroundResource(R.drawable.textfield);

    if(!(TextUtils.isEmpty(email))){
        if(!(TextUtils.isEmpty(password)))
            loginUser(email,password);
    }
}
```

Programski kod 6. Provjera ispunjenosti polja e-mail adrese i lozinke

U prvom dijelu se dohvaća instanca retrofit objekta, te se stvaraju metode web servisa prema sučelju. Kreiranjem call objekta uz pomoć metode „enqueue“ se asinkrono poziva web servis. Potom definira pozivnu metodu callback koja je zadužena za zaprimanje odgovora s web servisa (onResponse i onFailure). Metoda „onResponse“ provjerava je li HTTP odgovor uspješno primljen. Potvrdnim odgovorom se dohvaća tijelo koje sadrži JSON poruku i pokreće nova aktivnost. U slučaju neuspješnog poziva, ispisuje se poruka o pogrešci koristeći klasu Toast.

```
private void loginUser(String email, String lozinka){

    Retrofit retrofitClient = RetrofitClient.getInstance2();

    // Metoda web servisa prema IKorisnik sučelju
    iLogin = retrofitClient.create(IKorisnik.class);

    // Asinkrono povezivanje web servisa s parametrima metode
    Call<List<UserModel>> call = iLogin.getUser(email,lozinka);

    // Implementacija događaja
    call.enqueue(new Callback<List<UserModel>>() {

        // U slučaju da je poziv uspješan
        @Override
        public void onResponse(Call<List<UserModel>> call,
                               Response<List<UserModel>> response) {
            if(response.isSuccessful()) {
                // Podaci korisnika postoje u bazi podataka
                Intent intent = new Intent(UserLogin.this, MainWindow.class);
                intent.putExtra("email", emailText.getText().toString());
                startActivity(intent);
                finish();
            }
            // Nema podataka o korisniku, te se ispisuje poruka o pogrešci
            else if(response.code() == 400){
                Toast.makeText(getApplicationContext(),
                        "Pogrešna e-mail adresa ili lozinka!",Toast.LENGTH_SHORT).show();
            }
        }

        // U slučaju da poziv nije uspješan
        @Override
        public void onFailure(Call<List<UserModel>> call, Throwable t){ ;
            Toast.makeText(getApplicationContext(),
                    "Pogreška, poziv nije uspješan!", Toast.LENGTH_SHORT).show();
            t.printStackTrace();
            t.getMessage();
        }
    });
}
```

Programski kod 7. Implementacija poziva web servisa

Iz primjera se vidi metoda poziva web servisa za prijavu korisnika koja sadrži dva anotirana parametra. Ista metoda prikazuje put korištenja GET metode prilikom slanja zahtjeva. Metoda „getUser“ vraća objekt s listom objekata „UserModel“.

```
@GET("korisnik/login/{email}/{lozinka}")
Call<List<UserModel>> getUser(@Path("email") String email,
                              @Path("lozinka") String lozinka);
```

Programski kod 8. Metoda za poziv web servisa

Realizacija prijave korisnika u sustav je postignuta stvoranjem ruta. Rute su jednako definirane kao i unutar mobilne aplikacije. Zahtjev za GET dohvat podataka je postignut metodom „app.get“. Unutar funkcije su postavljeni parametri e-mail adrese i lozinke. U prvom koraku se uzima trenutna lozinka te se kriptira uporabom SHA-256 hash funkcije. Kriptiranje je potrebno kako bi se generirani hash i hash iz baze podataka mogli usporediti. Upitom se provjerava postoje li podaci u bazi podataka. Ukoliko podaci postoje u bazi, isti će biti vraćeni u JSON formatu. U protivnom, ako podataka nema, ispisuje se poruka s upozorenjem o nepostojećim podacima.

```
app.get('/korisnik/login/:email/:lozinka', (request, response)=>{
    var hash = crypto.createHash('sha256')
     .update(request.params.lozinka).digest('base64');
      connection.query(queries.loginUser,[request.params.email,hash],(err,res)=>{
        if(err) throw err;
        if(res.length > 0)  response.json(res);
        else response.status(400).send({message: "Nema podataka!"});
    })
});
```

Programski kod 9. Definiranje ruta za prijavu korisnika koristeći Node.js


```
[Array[1]
  -0:  {
        "id": 5,
        "ime": "test",
        "prezime": "test",
        "telefon": "123456789",
        "email": "test.test@primjer.com",
        "lozinka": "n4bQgYhMfWWaL+qgxVrQFaO/TxsrC4Is0V1sFbDwCgg="
    }
],
```


Slika 47. Prikaz dobivenog odgovora u JSON formatu

Dobiveni podaci označavaju uspješan poziv, u većini slučajeva popunjavaju elemente sučelja koristeći model. Model odgovara klasi s atributima jednakim onima iz JSON formata. GSON tada izvršava zadatak i rezultat obrade prosljeđuje podacima modela. Model tvori poslovnu logiku aplikacije i sadrži enkapsulizaciju. Postavljanje vrijednosti u model i iz modela se postiže metodama „get“ i „set“.

```
public class UserModel {

    @SerializedName("ime")
    @Expose
    private String ime;

    public void setIme(String ime) {
        this.ime = ime;
    }

    public String getIme() { return ime;}

    @SerializedName("prezime")
    @Expose
    private String prezime;

    public void setPrezime(String prezime){
        this.prezime = prezime;
    }

    public String getPrezime(){ return prezime;}

    @SerializedName("telefon")
    @Expose
    private String telefon;

    public void setTelefon(String telefon){
        this.telefon = telefon;
    }

    public String getTelefon(){ return telefon;}

    @SerializedName("email")
    @Expose
    private String e_mail;

    public void setEmail(String email){
        e_mail = email;
    }

    public String getEmail(){ return e_mail;}

    @SerializedName("lozinka")
    @Expose
    private String lozinka;

    public void setLozinka(String lozinka){
        this.lozinka = lozinka;
    }

    public String getLozinka(){return lozinka;}

}
```

Programski kod 10. Primjer modela klase „UserModel“

Plutajući akcijski gumb (eng. Floating Action Button) sučelja omogućuje prelazak s korisničkog u administracijski dio aplikacije. Nad gumbom je postavljena animacija namijenjena kretanju lijevo-desno s obzirom na os X. Na parametar „fromXDelta“ se postavlja vrijednost od -10%, a na „toXDelta“ 20%. Parametar „*repeatMode“* sadrži opciju *„reverse“* koja ponavlja animaciju od zadnje iteracije, a „repeatCount“ iznosi 25.

Sučelje **registracije** korisnika sadrži pet polja za unos podataka. Popunjavanjem polja korisnik pokreće proces registracije. Metoda *„registerUser“* koristeći Retrofit šalje POST zahtjev s podacima prema pozadinskom sustavu koji ih sprema u bazu podataka. Ukoliko su svi zahtjevi ispunjeni, namjera se pokreće i otvara novu aktivnost. U suprotnom se ispisuje poruka o postojanju korisnika u bazi.

```
private void registerUser(String ime, String prezime, String telefon, String email,
String lozinka) {
    Retrofit retrofitClient = RetrofitClient.getInstance2();
    iKorisnik = retrofitClient.create(IKorisnik.class);

    Call<Object> call = iKorisnik.registerUser(ime, prezime, telefon, email, lozinka);
    call.enqueue(new Callback<Object>() {
        @Override
        public void onResponse(Call<Object> call, Response<Object> response) {
            if(response.isSuccessful()){
                String responseBody = response.body().toString();
                if(responseBody.equals("true")) {
                    Intent intent = new Intent(UserRegistration.this, MainWindow.class);
                    intent.putExtra("email", emailText.getText().toString());
                    startActivity(intent);
                    finish();
                }
            }
            else if(response.code() == 404) {
                Toast.makeText(getApplicationContext(),
                        "Korisnik već postoji!", Toast.LENGTH_SHORT).show();
            }
            else{
                Toast.makeText(getApplicationContext(),
                        "Poziv nije uspješan!", Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onFailure(Call<Object> call, Throwable t) {
            t.printStackTrace();
            t.getMessage();
        }
    });
}
```

Programski kod 11. Prikaz registracije korisnika koristeći Retrofit

Napomena: @FormUrlEncoded se koristi za slanje podataka na poslužitelj uporabom POST metode. Prilikom definiranja svakog parametra, potrebno je dodati napomenu @Field s ključem koji je bitan radi budućeg raspoznavanja podataka na poslužitelju.

```
@POST("register")
@FormUrlEncoded
Call<Object> registerUser(@Field("ime") String ime,
                          @Field("prezime") String prezime,
                          @Field("telefon") String telefon,
                          @Field("email") String email,
                          @Field("lozinka") String lozinka);
```

Programski kod 12.Slanje podataka na server pomoću rute

Dohvaćaju se prosljeđeni parametri i poziva se asinkrona funkcija koja provjerava postojanje korisnika u bazi podataka prema e-mail adresi. Ukoliko ne postoji, pomoćna funkcija vraća logičku vrijednost „true“ i podaci se unose. U slučaju postojanja, aplikacija vraća „false“, unosi se nova e-mail adresa i ponavlja postupak registracije. Ukoliko server ne odgovara na postavljeni upit, ispisuje se poruka o pogrešci.

```
var query = function(sql, par){
    return new Promise(function(reslove,reject){
        var result = false;
        db.query(sql,par,function(err,res){
            if(err) throw err;
            if(res[0].email == 0){
                result = true;
            }else result = false;
            reslove(result);
        })
    })
};
```

Programski kod 13. Pomoćna funkcija koja provjerava postojanje korisnika

```
app.post('/register', (request, response)=> {
    let post = {ime: request.body.ime, prezime: request.body.prezime,
        telefon: request.body.telefon, email: request.body.email,
        lozinka: crypto.createHash('sha256')
          .update(request.body.lozinka).digest('base64')};

        var getResult = async function(){
            let result = await query(queries.registrationCheck,post.email);
            console.log(result)
            if(result == true){
                connection.query(queries.insertKorisnik,post,(err) =>{
                    if(err) throw err;
                    response.send(true);
                });
            }else if(result == false){
                response.status(404).send({message: "Korisnik postoji!"})
            }else response.status(400).send({message: "Greška!"});
        }
        getResult();
});
```

Programski kod 14. Definiranje rute registracije korisnika koristeći Node.js

6.3.3 Popis glavnih funkcionalnosti (korisnik)

Korijenski element je ScrollView koji omogućuje pristup podacima prelaskom prsta preko zaslona. Predstavlja dobru praksu kada se ne zna dimenzija uređaja na kojoj će raditi buduća aplikacija. Unutar njega je smješten LinearLayout sa zadatkom vertikalnog rasporeda elemenata. Svaka funkcionalnost je predočena s CardView-om, raspolaže s nekoliko horizontalnih i vertikalnih LinearLayout rasporeda. Nad karticama se postavlja slušač događaja koji reagira kada korisnik klikne na nju.

U prvom koraku je potrebno pronaći mapu *„res“* i kreirati novu mapu *„menu“*. Unutar mape se radi nova XML datoteka. Stavke unutar nje se definiraju kroz element </item>, a sadrži atribute *„title“* i *„icon“*. Za izradu glavnog dijela potrebno je uzeti NavigationView u kombinaciji s DrawerLayout-om. NavigationView je podijeljen na dva dijela, zaglavlje i navigaciju. Zaglavlje se kreira kao zaseban layout unutar kojeg se nalazi slika, naslov i podnaslov. Za potrebe obrade događaja u navigacijskom dijelu, potrebno je implementirati metodu *„onNavigationItemSelected()“* unutar koje se dohvaća ID. Grananjem switch pokreće se namjera otvaranja nove aktivnosti.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
    drawerLayout.closeDrawer(GravityCompat.START);

    switch(menuItem.getItemId()){
        case R.id.pregledRezervacija:{
            Intent intent = new Intent(MainWindow.this, ReservationOverview.class);
            startActivity(intent);
            overridePendingTransition(R.anim.slide_out_left,R.anim.slide_out_right);
            finish();

        }break;

        case R.id.popisPredstava:{
            Intent intent = new Intent(MainWindow.this, UserPerformances.class);
            startActivity(intent);
            overridePendingTransition(R.anim.slide_out_left, R.anim.slide_out_right);
            finish();

        }break;
}
```

Programski kod 15. Primjer pokretanja nove aktivnosti koristeći namjere

6.3.4 Pregled predstava

Sučelje je dizajnirano kako bi omogućilo pretraživanje i odabir predstave od strane korisnika. AppBarLayout uključuje alatnu traku s efektom pomicanja prema dolje.

TabLayout nudi implementaciju vodoravnih kartica koje predstavljaju kategorije predstava. Nove kartice se kreiraju pozivom metode „newTab()“, te se unosi kategorija. LinearLayout pruža vertikalni prikaz elemenata, a TextView prikazuje tekst odabrane kategorije predstave. Sučelje nudi opcije pretraživanja, pregleda i odabira predstava.

```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbarUserPerformance"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/pozadina"
        android:fitsSystemWindows="true"
        android:minHeight="?attr/actionBarSize"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:title=" ">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:contentDescription="@string/name_app"
            android:src="@drawable/logo" />
    </androidx.appcompat.widget.Toolbar>

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tabsUser"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/pozadina"
        android:backgroundTint="@color/pozadina"
        android:visibility="visible"
        app:tabGravity="fill"
        app:tabMode="scrollable"
        app:tabTextColor="#FFFFFF" />
</com.google.android.material.appbar.AppBarLayout>
```

Programski kod 16. Prikaz isječka XML koda za pregled kategorija predstava

1) **Pretraživanje predstava** se postiže metodom GET koja dohvaća sve podatke o predstavi iz baze podataka. Podaci pune listu (ArrayList) koja se prosljeđuje adapteru. Adapter prikazuje njen sadržaj ovisno o korisničkom zahtjevu. Tražilica prilikom unosa svakog slova provjerava podudaranost s naslovom predstave. U slučaju da predstave nema, aplikacija vraća poruku o nepostojanju predstave.

2) **Pregled i odabir predstave**: odabirom kategorije dolazi do slanja GET zahtjeva za podacima iz baze podataka koji odgovaraju traženoj kategoriji. Odabirom predstave pokreće se nova namjera unutar koje se postavlja ID predstave kako bi se u sljedećoj aktivnosti mogli prikazati podaci iste predstave. Otvaranjem nove aktivnosti preuzima

se prosljeđena Id vrijednost, te se ponovno šalje GET zahtjev vezan za odabranu predstavu. Metoda *„setData()“* popunjava ImageSlider sa slikama koje su dohvaćene iz baze podataka, te unosi naziv naziv predstave koji se nalazi na njemu. Naslov se pozicionira tako da bude na kraju ImageSlidera, a to se postiže naredbom *„layout_top“* = -40dp. TabLayout omogućuje odabir prikaza detalja predstave ili rasporeda prikazivanja. Svaka od opcija zahtjeva novi fragment s pripadajućim rasporedom i elementima. Fragment *„ShowDetailFragment“* koristi CardView na koji se postavljaju elementi prikaza detalja predstave. Prethodno dobiveni rezultati pune preostale widgete koji pripadaju fragmentu. Alternativno, fragment *„PerformanceTickets“* je sačinjen od tri RadioButton pogleda koji imaju ulogu filtera, RecyclerView se puni s podacima iz baze, te koristi layout namijenjen prikazu reda po podatku rasporeda.

Implementacija favorita: daje pregled favoriziranih predstava. Ikona favorita je izrađena koristeći ImageView, te je postavljena u alatnu traku. Ona koristi dvije slike, prazno odnosno puno srce. Prazno srce označava predstavu koja nije dodana na listu želja, dok puno srce predstavu na listi. Odabirom predstave, program gleda postoji li predstava na listi odnosno u tablici *„Moj Izbor“*. Pregled se završava stanjima *postoji* (true) i *ne postoji* (false). Program označava ikonu u skladu s trenutnim stanjem. Na ikonu favorita se postavlja slušač događaja koji poziva metodu nakon korisničke interakcije na ikonu. Interakcijom se predstave dodaju odnosno brišu s liste. Prikaz uspješnog ili neuspješnog dodavanja, te brisanja koristi snackbar (widget preuzet iz biblioteke Material Design).

```
imageFavorites.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(favoritesModel.getDostupnost()) {
            imageFavorites.setBackgroundResource(R.drawable.ic_favorite_black_24dp);
            deleteFavorites();
            setNegativeSnackbar(v);
        }else {

imageFavorites.setBackgroundResource(R.drawable.ic_favorite_border_black_24dp);
            insertFavorites();
            setPozitiveSnackbar(v);
        }
    }
});
```

Programski kod 17. Dodavanje predstave u favorite

Implementacija dijeljenja sadržaja: obuhvaća jednostavnu razmjenu podataka među omiljenim aplikacijama i društvenim mrežama. Za realizaciju je potrebno koristiti

namjere. Namjere su organizirane s ciljem slanja sadržaja van aplikacije, izravno drugom korisniku. Slanje podataka iz jedne aktivnosti u drugu koristi akciju *„Intent.ACTION_SEND“.* Metodom „setType()“ je definiran tip podataka oblika MIME-a. Prosljeđivanje podataka se postiže metodom *„putExtra()“.* Naslov predstave je dodan u *„EXTRA_SUBJECT“,* a opis poruke zajedno s naslovom predstave u *„EXTRA_TEXT“.* Slijedi dodavanje naslova na dijaloški okvir za odabir. Metodom *„finish()“* aktivnost se zatvara i uklanja sa stoga. U slučaju iznimaka, ispisuje se tekst pogreške s opisom o nemogućnosti podjele sadržaja koristeći klasu Toast.

```
imageShare.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
        try { Intent i = new Intent(Intent.ACTION_SEND);
            i.setType("text/plain");
            i.putExtra(Intent.EXTRA_SUBJECT, "Premijera predstave");
            i.putExtra(Intent.EXTRA_TEXT,
                   theaterModel.getNaziv() + " dostupno u Theater GOLD aplikaciji!");
            startActivity(Intent.createChooser(i, "Podijeli s prijateljima!"));
            finish();
        }catch(Exception e){
            Toast.makeText(getApplicationContext(),
                   "Nije moguće podijeliti sadržaj!",Toast.LENGTH_SHORT).show();
        }
    }
});
```

Programski kod 18. Programski kod dijeljenja sadržaja predstave

6.3.5 Rezervacija ulaznica

Ideja o rezervaciji ulaznica se proteže od odabira predstave do potvrde rezervacije. Interakcija između aplikacije i korisnika se odvija kroz četiri standardne opcije odabira: predstave, vremena, datuma prikazivanja i količine ulaznica. Dizajn je započet s izgradnjom ScrollView-a pomoću kojeg sadržaj bilježi pomak prema dolje. Nadalje, odabir predstave u aplikaciji koristi tri padajuća izbornika Spinner.

1.Punjenje prvog izbornika se odvija GET zahtjevom i dohvatom predstava koje imaju definiran termin izvođenja i čiji je datum veći/jednak datumu pristupanja iz aplikacije. Prethodno se postiže tako da u SQL upit osim naredbe SELECT dodaje naredba DISTINCT koja sprječava kopiranje predstava istog naziva. Podaci se pohranjuju u listu i ona je prosljeđena metodi *„setSpinner(view)“* za punjenje izbornika

2.Sljedeći izbornik je namijenjen odabiru svih datuma prikazivanja predstave koju je korisnik prethodno odabrao. Cjelokupni postupak je identičan prethodnom postupku.

3.Posljednji izbornik nudi odabir vremena prikazivanja predstave. Opcija se odnosi na slučaj većeg broja izvođenja predstave istog datuma. Tim načinom korisnik bira termin predstave koju želi pogledati odnosno rezervirati ulaznice.

Završetkom odabira vremena predstave, potrebno je kliknuti na dobnu kategoriju postignutu CheckBox pogledom. Postavlja se uvjet programu o uzimanju u obzir samo cijena odabranih dobnih kategorija. Prilikom postavljanja slušača na događaj potrebno je postaviti upit u kojem se provjerava koji je od CheckBox pogleda odabran. Prema odabranoj kategoriji se povlače cijene iz baze podataka, te se postavljaju u tablicu s podacima o predstavi. Gumb *„Rezerviraj sjedala“* vodi na novu aktivnost koja uključuje odabir sjedala u dvorani. Namjerom se šalje ID (primarni ključ) predstave za koju želimo rezervirati ulaznice.

```
private void setSpinner(View view){
    String [] arrayNaziv = new String[theaterModel.getNazivPredstave().size()];
    arrayNaziv = theaterModel.getNazivPredstave().toArray(arrayNaziv);

    ArrayAdapter<String> adapter = new ArrayAdapter<String>(view.getContext(),
            android.R.layout.simple_spinner_item, arrayNaziv);
    adapter.setDropDownViewResource(android.R.layout.simple_list_item_single_choice);
    spinnerPredstava.setAdapter(adapter);

    for(int i=0; i<spinnerPredstava.getCount(); i++){
        if(spinnerPredstava.getItemAtPosition(i).toString().equals(podaci[0])){
            spinnerPredstava.setSelection(i);
            break;
        }
    }
    spinnerPredstava.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
{
        @Override
        public void onItemSelected(AdapterView<?> parent,
                                   View view, int position, long id) {
            ((TextView) parent.getChildAt(0)).setTextColor(Color.BLACK);
            if(parent.getItemAtPosition(position).equals("")){ }
            else{
                String item = parent.getItemAtPosition(position).toString();
                getDate(item);
            }
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) { }
    });
}
```

Programski kod 19. Prikaz punjenja Spinnera i odabira stavke predstave

Rezervacija sjedala u dvorani je predstavljena aktivnošću čije je sučelje realizirano CollapsingLayout rasporedom. U svome naslovu je sadržan naziv predstave čime je postignuto da naslov u layoutu bude veći kada je izgled u potpunosti vidljiv, odnosno

manji kada se raspored pomiče izvan zaslona. On se ujedno sastoji od dijelova, AppBarLayout i NestedScrollView. U AppBar se postavlja ikona strelice za povratak na prethodnu aktivnost i ikona za rotiranje zaslona u NestedScrollView. Prije rotacije se grananjem provjerava trenutni položaj zaslona, te shodno tome zakreće zaslon.

```
private void setElements(){
    createCheckBox();
    bookingToolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(BookingTickets.this,UserReservation.class);
            startActivity(intent);
            finish();
        }
    });
    imageRotate.setOnClickListener(new View.OnClickListener() {
        @SuppressLint("SourceLockedOrientationActivity")
        @Override
        public void onClick(View v) {
            final int screenOrientation=
BookingTickets.this.getResources().getConfiguration().orientation;

            switch(screenOrientation){
                case Configuration.ORIENTATION_PORTRAIT: {
                    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                }break;

                case Configuration.ORIENTATION_LANDSCAPE:{
                    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                }break;
            }
        }
    });
}
```

Programski kod 20. Postavljanje slušača radi mogućnosti zakretanja zaslona

Podaci o predstavi se dohvaćaju iz baze prema ID-u odnosno primarnom ključu tablice rasporeda predstave preuzetog od prethodne aktivnosti. Lista s dobivenim podacima se raspoređuje po modelu ponude predstave. U konačnici se u naslov postavlja naziv metodom *„collapsingLayout.setTitle(offerModel.getNaziv())“*.

NestedScrollView se koristi za pomičan prikaz sjedala u dvorani. Svojom strukturom je sličan rasporedu ScrollView, a glavna razlika je korištenje kada postoji prikaz u već drugom pomičnom prikazu. Spinner je namijenjen odabiru količine sjedala koje korisnik može rezervirati. Kazališna dvorana je izrađena koristeći CardView. Unutar njega se postavlja novi vertikalni LinearLayout koji pruža vertikalni raspored sjedala po redovima. Za potrebe izrade sjedala, potrebno je uvrstiti horizontalni raspored redova.

Kratke upute prikazuju status rezerviranih sjedala. Upute su predočene s tri ImageView pogleda unutar kojih su postavljene boje koje odgovaraju trenutnom statusu sjedala.

Sjedala su predstavljena CheckBox-om, te su izrađena programski unutar petlje. Status rezerviranih sjedala se provjerava kada je aktivnost kreirana i na kraju kada korisnik odabere sjedala koja želi rezerviratil. Na početku se postavlja crvena boja za sva sjedala koja su zauzeta i takva sjedala nije moguće odabrati. Nad svaki CheckBox koji odgovara zauzetom sjedalu se poziva metoda *„setEnabled(false)“* čime se sprječava ponovni odabir istog sjedala. Klikom na gumb, podaci o odabranim sjedalima i količina ulaznica se postavljaju u liste, te se šalju narednoj aktivnosti.

```
private void checkSeat(){
    Retrofit retrofit = RetrofitClient.getInstance2();
    iReservation = retrofit.create(IReservation.class);
    checkBoxes = new CheckBox[121];
    Call<List<ReservationModel>> call =
          iReservation.checkAvailability(reservationModel.getId_raspored());
    call.enqueue(new Callback<List<ReservationModel>>() {
        @Override
        public void onResponse(Call<List<ReservationModel>> call,
         Response<List<ReservationModel>> response) {
            if(response.isSuccessful()){
                List<ReservationModel> list = response.body();

                for(int i=0; i<list.size(); i++){
                    reservationModel.setOznaka(list.get(i).getOznaka());
                    int oznaka = reservationModel.getOznaka();
                    checkBoxes[oznaka] = (CheckBox) findViewById(oznaka);
                    checkBoxes[oznaka].setTextColor(Color.RED);
                    checkBoxes[oznaka].setEnabled(false);
                    checkBoxes[oznaka].setClickable(false);
                    CompoundButtonCompat.setButtonTintList
                            (checkBoxes[oznaka], ColorStateList.valueOf(Color.RED));
                }
            }
        }
        @Override
        public void onFailure(Call<List<ReservationModel>> call, Throwable t) {
            t.getMessage();
            t.printStackTrace();
        }
    });
}
```

Programski kod 21. Provjera zauzetosti sjedala u dvorani

Završetak rezervacije ulaznica

Radi postizanja boljeg dizajna se koristi CardView unutar kojeg je smješten vertikalni LinearLayout. Zatim se postavlja onoliko horizontalnih LinearLayout rasporeda koliko je potrebno prikazati podataka. Na kraju se dodaje raspored koji prikazuje račun rezervacije s odgovarajućim stavkama i njihovim cijenama. Korisnik završava s rezervacijom klikom na gumb *„Rezerviraj“*. Podaci rezervacije se spremaju u tablicu „Rezervacija“, dok se red i sjedalo spremaju u tablicu „Karta“ koja je povezana stranim

ključem na tablicu „Rezervacija“. Spremanjem aplikacija pomoću namjere pokreće aktivnosti koja sadrži popis trenutnih rezervacija.

6.3.6 Pregled rezervacija i favorita

Otvaranjem nove aktivnosti osigurava se zaslon na kojem se nalaze fragmenti namijenjeni prikazu rezervacija i favorita. Kako bi korisnik mogao pregledavati sadržaj koji se nalazi na fragmentima, potrebno je osmisliti način prebacivanja s fragmenta na fragment. BottomNavigationView mijenja poglede najviše razine jednim dodirom.

Postupak pripreme obuhvaća otvaranje glavne mape *„res“* i odabir pripadne mape *„menu“*. U mapi se kreira datoteka pod nazivom: *„bottom_reservation.xml“.* Unutar nje je potrebno definirati programski kod koji sadrži: ID, naslov i ikonu izbornika.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/pregledRezervacije"
        android:icon="@drawable/ticket"
        android:title="Lista rezervacija"
        app:showAsAction="always">
    </item>

    <item
        android:id="@+id/mojIzbor"
        android:icon="@drawable/ic_favorite_border_black_24dp"
        android:title="Moj izbor"
        app:showAsAction="always"></item>
</menu>
```

Programski kod 22. Prikaz XML koda navigacije

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/navigation_pregledRezervacija"
    android:layout_alignParentBottom="true"
    android:background="#323232"
    android:foregroundGravity="bottom"
    app:itemTextColor="#FFFFFF"
    app:itemIconTint="#FFFFFF"
    app:menu="@menu/bottom_reservation"
    tools:ignore="MissingConstraints">
</com.google.android.material.bottomnavigation.BottomNavigationView>
```

Programski kod 23. Prikaz dodavanja izbornika u donji navigacijski prikaz

Realizacija se postiže dodavanjem metode *„OnNavigationItemSelectedListener“* unutar aktivnosti. Njegova je namjena da kada korisnik dodirne ikonu dođe do prebacivanja fragmenata. Prije toga se provjerava je li odabrani fragment instanca postojećeg fragmenta. Time je spriječen pokušaj dodavanja jednog te istog fragmenta

uzastopnim pritiskom na istu opciju. Dodatno je riješeno i eventualno rušenje aplikacije koje bi moglo uslijediti zbog interakcije s bazom podataka, jer ne bi mogla vraćati odgovore ako bi administrator brzo dodavao jedan te isti fragment. Na kraju, kao rezultat je moguće mijenjati poglede pri odabiru opcije navigacijskog izbornika. Skup promjena unutar operacije se naziva transakcijom, pa se može reći kako se zamjena fragmenata izvršava transakcijski.

```
private BottomNavigationView.OnNavigationItemSelectedListener navListener = new
BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem menuItem){
        List<Fragment> fragments = getSupportFragmentManager().getFragments();
        for(Fragment fragment :fragments) {
            switch (menuItem.getItemId()) {
                case R.id.pregledRezervacije: {
                    if (!(fragment instanceof ReservationList)) {
                        fragment = new ReservationList();
                    }
                }break;

                case R.id.mojIzbor: {
                    if (!(fragment instanceof FavoritesOverview)) {
                        fragment = new FavoritesOverview();
                    }
                }break;
            }
            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.contentReservation, fragment).commit();
        }
        return true;
    }
};
```

Programski kod 24. Prikaz odabira fragmenata koristeći navigaciju

Slika 48. Prikaz navigacije za prebacivanje fragmenata

1. Sučelje fragmenta za korijenski element koristi FrameLayout čija je uloga rezervacija potrebnog prostora na zaslonu bilo kojeg objekta klase View. Unutar njega su definirani vertikalni odnosno horizontalni LinearLayout rasporedi. Horizontalni prikazuje podatke broja rezervacija, te omogućuje filtriranje prikaza. Postavljanjem slušača, moguće je prikazati ili sakriti filter prikaza. Filter je realiziran elementom CardView na koji se dodaje obrub i zaobljenje i u njemu nalazi *spinner* koji sadrži nazive predstava za koje postoji rezervacija. Podatke je potrebno dohvatiti iz baze podataka, te listu napunjenu s podacima proslijediti metodi za punjenje izbornika.

Dodatan filter prikazuje predstave s obzirom na aktivnost. Aktivnosti su realizirane s elementom RadioGroup koji grupira tri RadioButton pogleda. RecyclerView pruža prikaz rezervacija koje postoje s obzirom na odabrani filter. Korištenje zahtjeva izradu klase adaptera i datoteke unutar koje će biti izrađeno sučelje vezano za prikaz rezervacija. Podaci za RecyclerVIew se dohvaćaju GET zahtjevom iz baze podataka, te pune listu koja se šalje adapteru. Adapter tada preuzima podatke i poziva razmještaj. Podaci se u metodi *„onBindViewHolder()“* raspoređuju po elementima. Konačan rezultat je prikaz liste rezervacija odabranog filtera. Otvaranje detalja rezervacije se postiže fragmentom CardView i LinearLayout rasporedom.

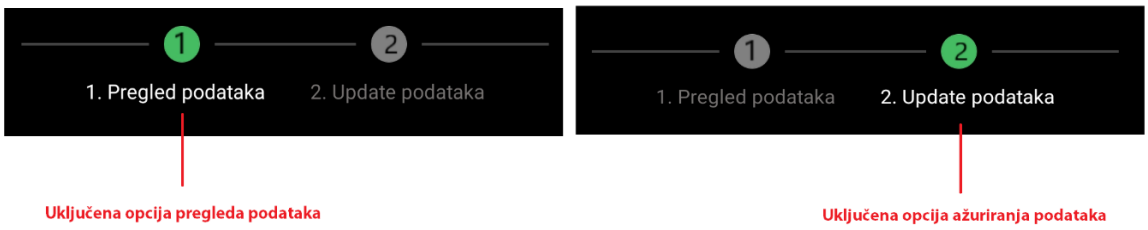
2. Korijenski element sučelja je ScrollView koji sadrži vertikalni LinearLayout raspored. Tražilica za pretraživanje predstava je realizirana s dodatna dva LinearLayout rasporeda čija vidljivost ovisi o promjeni stanja na prekidaču. Odabir datuma koristi DatePicker pogled pomoću kojeg korisnik bira ispravan unaprijed formatirani datum. Prikaz podataka u RecyclerView-u zahtjeva izradu sučelja koje će predstavljati jedan red u popisu. Potrebni podaci se dohvaćaju iz baze, te pune listu. Lista se prosljeđuje adapteru koji raspoređuje na sučelje prema definiranim metodama.

6.3.7 Upravljanje korisničkim računom

Upravljanje profilom obuhvaća pregled i ažuriranje podataka korisnika. Na početku se dohvaća ID korisnika, te se njime GET zahtjevom dohvaćaju podaci iz baze podataka. Dobiveni podaci se postavljaju u model odnosno instancu klase „UserModel“. Istoimena klasa sadrži getter i setter metode koje postavljaju odnosno dohvaćaju vrijednosti. Na taj način, dohvaćanjem vrijednosti iz modela se pune polja za prikaz podataka. Korijenski element sučelja je CoordinatorLayout, a prikaz podataka je postignut s elementom CardView. Unutar njega se definira TextView za opis pojedinog podatka. EditText služi ispuni s podacima, te se nad svaki postavlja uvjet o nemogućnosti unosa sve dok korisnik ne odabere opciju ažuriranja.

Pomoću horizontalnog LinearLayout-a, dva plutajuća akcijska gumba i dva polja za prikaz teksta kreirana je navigacija koja ima zadatak prikazati status korištenja funkcionalnosti. Korisničkim pregledom podataka, unutar prvog gumba se postavlja zelena pozadina kako bi signaliziralo da se radi o pregledu podataka. Uključivanjem ažuriranja podataka pritiskom na element prekidača Switch, slušač događaja reagira, te se mijenja pozadina prvog gumba odnosno postavlja se siva ispuna pozadine. Na

drugom gumbu za prikaz ažuriranja se javlja zelena pozadina. Postupak se iterativno izvršava s obzirom na količinu uporabe korisnika.



Slika 49. Usporedba prikaza aktivnosti odabrane funkcionalnosti

Korisnik ažurira sve podatke osim podatka o šifri. Opcije zadržavanja e-mail adrese i unosa lozinke su realizirane uporabom CheckBox pogleda. U slučaju da korisnik ne želi zadržati e-mail adresu tada će se prije spremanja vršiti provjera nepostojanja unesene e-mail adrese. Ukoliko adresa postoji u bazi, aplikacija javlja korisniku grešku koristeći dijaloški okvir biblioteke *„Sweet Alert“.* Prilkom unosa lozinki treba paziti na identičnost lozinke, jer će u protivnom aplikacija obavijestiti o neispravnosti skočnom porukom, definirane duljine i vremena trajanja.

6.3.8 Pregled cijena

Aktivnost omogućuje prikaz cijena ulaznica različitih kategorija. Podaci se dohvaćaju iz baze podataka slanjem GET zahtjeva pozadinskom sustavu. Zahtjevom se dohvaćaju cijene s najnovijim datumom unosa, te se dobiveni podaci postavljaju u model gdje se raspoređuju po elementima. Unutar korijenskog elementa sučelja ScrollView je definiran vertikalni LinearLayout. Na alatnoj traci se postavlja ImageView s logom i strelica za povratak na prethodnu aktivnost.

6.3.9 Pregled najčešćih pitanja

Česta pitanja obuhvaćaju upite koje korisnici postavljaju u svezi rada i poslovanja aplikacije. Cilj je izraditi jednostvano sučelje koje će preko liste prikazivati popis pitanja i pripadnih odgovora. Implementacija obuhvaća ExpandableListView koji se koristi za grupiranje podataka vezanih uz pitanja. Sastoji se od dvije glavne razine odnosno grupe. U glavnom izborniku (zaglavlje) su prikazani naslovi pitanja, dok se u podizborniku (stavka) nalazi odgovor. Proširivanje odnosno sažimanje se postiže dodirom na neki od elemenata. Za korištenje liste je potrebno kreirati adapter koji će učitavati podatke u elemente vezanih za prikaz. U prvom koraku GET zahtjevom se dohvaćaju svi podaci vezani za najčešća pitanja. Dobiveni podaci pune liste koje se

prosljeđuju konstruktoru adaptera. Metodom iz adaptera *„getGroupView“* se postavlja prikaz zaglavlja grupe, dok se metoda *„getChildView“* koristi za dobivanje pogleda odabrane stavke. Sučelje za korijenski element koristi ScrollView. Na kraju se kreira lista koja ima ulogu prikaza podataka koristeći adapter.

6.3.10 Pregled informacija o aplikaciji

Informacije o aplikaciji omogućuju pregled tehničkih podataka o aplikaciji. Sučelje je realizirano s CoordinatorLayout-om koji sadrži alatnu traku i NestedScrollView za postizanje dinamičkog prikaza podataka.

6.3.11 Administratorski prikaz predstava

Cilj je izraditi sučelje unutar kojeg administratori mogu pregledavati predstave prema različitim kategorijama. Zaslon je podijeljen na dva dijela, to su: aktivnost i fragment. Aktivnost sadrži alatnu traku unutar koje se nalazi strelica za povratak na prethodnu aktivnost i logo aplikacije. Povratak na prethodnu aktivnost se izvodi preko eksplicitne namjere. Aktivnost sadrži HorizontalScrollView unutar kojeg se nalazi izbornik koji sadrži horizontalni popis kategorija predstava. Logika aplikacije se svodi na administratorov odabir kategoriju i vraćanjem odgovora kroz popis svih predstava iste kategorije. Primjer: administrator odabire određenu kategoriju i podatak se prosljeđuje pozadinskom sustavu koji uz pomoć upita SELECT provjerava predstave. Potom baza vraća rezultat tog upita i on se pohranjuje u listu.

```
private void sendCategory(String kategorija){
    Bundle bundle = new Bundle();
    bundle.putString("kategorija", kategorija);
    PerformancesFragment fragment = new PerformancesFragment();
    fragment.setArguments(bundle);
    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    ft.replace(R.id.fragmentPerformances, fragment);
    ft.commit();
}
```

Programski kod 25. Prosljeđivanje podataka sljedećem fragmentu

Fragment sadrži RecyclerView i CardView i za potrebe punjenja je potrebno koristiti adapter. Napunjena lista se prosljeđuje adapteru koji puni CardView s podacima slike i naziva predstave. Nad RecyclerView-om je dodana i animacija koja će poredati elemente liste od vrha ka dnu unutar definiranog vremena. Odabirom neke od ponuđenih predstava, pokreće se eksplicitna namjera nove aktivnosti unutar koje će

se prikazati detalji iste predstave. Potrebno je postaviti slušač na razmještaj koji će pod pritiskom na bilo koji dio otvoriti novu aktivnost i proslijediti ID narednoj aktivnosti.

Detalji predstave služe prikazu informacija odabrane predstave. Uzima se prosljeđeni ključ iz prethodne aktivnosti, te se njime dohvaćaju podaci o predstavi. Svi podaci su predočeni u EditTextu radi naknadne izmjene od strane administratora. Klikom na prekidač, uključuje se opcija ažuriranja. U slučaju promjene slike, to se čini odabirom jedne od ponuđenih iz galerije ili se opcionalno može dodati put slike u bazu, čime bi se ujedno spremila u galeriju. Moguće je kasnije brisanje slike.

```
if(flag.equals(true)){
    naslovEdit.setFocusableInTouchMode(true);
    naslovEdit.setSingleLine(false);
}else if(flag.equals(false)){
    naslovEdit.setFocusable(false);
}
```

Programski kod 26. Prikaz ažuriranja podataka

Ažuriranje se provodi tako što se sav EditText zablokira (flag = **true**) prilikom kreiranja aktivnosti. Pritiskom na prekidač Switch, vrijednost atributa se mijenja u **false** što omogućuje promjenu podataka. Metodom „setSingleLine“ omogućen je unos podataka unutar jednog reda čime se sprječavaju velike količine teksta.

6.3.12 Administratorski pregled rezerviranih ulaznica

Pregled sadrži popis svih ponuda predstava koje su pohranjene u bazi. Namijenjena je bržem pregledavanju ponuda pomoću različitih filtera za sortiranje. Administrator može izbrisati ponudu kao i sve rezervacije koje se odnose na nju. Aktivnost se sastoji od alatne trake, filtera ponuda, stanja prikaza, te liste s popisom ponuda predstava.

1) Alatna traka je realizirana pomoću AppBarLayouta unutar kojeg se nalazi Toolbar koji sadrži logotip aplikacije i strelicu za povratak na prethodnu aktivnost.

2) Filter ponuda daje pregled i brže dohvaćanje ponuda predstava. Prilikom pokretanja aktivnosti aplikacija dohvaća sve podatke iz baze za koje postoji ponuda. Iste ponude popunjavaju padajući izbornik Spinner s predstavama. Odabirom predstave se prosljeđuje primarni ključ predstave prema kojem će se u novoj aktivnosti dohvaćati podaci vezani za tu predstavu. Stoga treba biti dobra povezanost između tablica *„Ponuda_predstava“* i *„Predstava“*. Drugim riječima, tablica *„Raspored_predstave“* sadrži strani ključ jednak primarnom ključu tablice *„Predstava“.*

3) Stanje prikaza je filter koji sadrži tri stanja prema kojima administrator može filtrirati ponude. Stanja obuhvaćaju opcije: sve, aktivno i prošlo. Realizirani su elementom RadioGroup unutar kojeg se nalaze tri RadioButton pogleda koji odabirom jedne od ponuđenih opcija poziva metodu „onCheckedListener()“.Ona pomoću dodatne metode „isChecked()“ provjerava koji je od pogleda odabran, te prema njemu filtrira.

4) Lista s predstavama koristi ListView napunjen s podacima iz baze podataka. ListView zahtjeva adapter koji će dobivati podatke preko liste, te puniti odgovarajuće elemente. Postoji i stavka koja briše ponudu i pripadne rezervacije. Prikaz detalja predstave se postiže slanjem primarnog ključa predstave idućoj aktivnosti koja će na temelju ključa povući podatke o detaljima iz baze te ih prikazati.

Prikaz detalja predstave pruža podatake odabrane ponude predstave. Aktivnost sadrži TabLayout koji posjeduje kartice namijenjene prikazu tri fragmenta za pregled: detalja ponude, rezervacije i statistike. Pokretanjem aktivnosti, dohvaća se proslijeđeni ključ ponude predstave i ponovno šalje odabranom fragmentu.

1) Prikaz detalja o predstavi zahtjeva preuzimanje vrijednosti primljenog ključa. GET zahtjevom se povlaće podaci iz baze, te se postavljaju na elemente fragmenata.

2) Pregled rezerviranih ulaznica odabrane ponude su realizirane fragmentom. Popis rezervacija su predstavljene SwipeLayout-om. Moguće je iste filtrirati prema statusu plaćanja. Na taj način imamo TabLayout s opcijama: prikaz rezervacija neovisno o statusu plaćanja, prikaz samo plaćenih rezervacija, te prikaz neplaćenih rezervacija. Aplikacija pri učitavanju broji količinu rezervacija, te je prikazuje na zaslonu. Detaljan pregled se postiže postavljanjem slušača na SwipeLayout radi otvaranja posebnog fragmenta s pojedinostima rezervacije. Dodani fragment je napravljen radi pregleda podataka, ažuriranja statusa plaćanja i brisanja rezervacije .

3) **Statistika** služi prikazu podataka koristeći grafikone iz MPAndroidChart biblioteke. Svaki dio grafikona je predstavljen zasebnom bojom radi isticanja različitosti podataka. Ispod se nalazi legenda koja pomaže njegovom boljem razumijevanju. Prvi je predstavljen stupčastim grafikonom (eng. BarChart). On se odnosi na pregled rezerviranih sjedala po redovima unutar dvorane određene predstave. Drugi je predstavljen kružnim grafikonom (eng. PieChart). Unutar njega je prikazana količina slobodnih mjesta koje korisnik može rezervirati, broj plaćenih rezervacija, te broj neplaćenih rezervacija.

```
private void setElements(){
    changeSwitch.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(changeSwitch.isChecked()){
                Bundle bundle = new Bundle();
                bundle.putInt("id",offerModel.getId());
                PieChartFragment fragment = new PieChartFragment();
                fragment.setArguments(bundle);
                getFragmentManager().beginTransaction()
                            .replace(R.id.fragmentReservation,fragment).commit();
            }
        }
    });
}
```

Programski kod 27. Promjene grafikona i slanje identifikacijskog ključa

U kodu 27 prikazana je promjena grafikona uporabom fragmenata koristeći prekidač. Na njega se postavlja slušač koji reagira pod pritiskom i ako je uključen primarni ključ. Ponude predstave se šalju novom fragmentu radi dohvaćanja podataka za grafikon.

```
private void createGraph(int razlika, int poz, int neg){
    List pieData = new ArrayList<>();
    ArrayList label = new ArrayList();

    pieData.add(new Entry(razlika, Color.BLUE));
    pieData.add(new Entry(poz, Color.GREEN));
    pieData.add(new Entry(neg, Color.RED));

    PieDataSet dataSet = new PieDataSet(pieData, "Rezervirana sjedala");
    label.add("Slobodna mjesta");
    label.add("Plaćeno");
    label.add("Neplaćeno");

    PieData data = new PieData(label, dataSet);
    data.setValueTextSize(9f);
    dataSet.setColors(ColorTemplate.COLORFUL_COLORS);
    pieChart.animateXY(5000, 5000);

    PieDataSet bardataset =
         new PieDataSet(pieData, "Odnos plaćenih i neplaćenih sjedala");

    bardataset.setColors(ColorTemplate.COLORFUL_COLORS);
    bardataset.setValueTextColor(Color.WHITE);
    bardataset.setValueTextSize(12f);
    data.setValueTextColor(Color.WHITE);
    pieChart.getLegend().setTextColor(Color.WHITE);
    pieChart.setData(data);
}
```

Programski kod 28. Prikaz izrade tortnog grafikona

6.3.13 Administrator upravljanja profila

Omogućuje pregled i promjenu osobnih podataka i dodavanje novog administratora. Funkcionalnost je realizirana s aktivnosti i dva fragmenta. Fragmenti su namijenjeni prikazu osobnih podataka i sučelja za dodavanje administratora. Aktivnost sadrži TabLayout unutar kojeg se nalaze dvije kartice koje odgovaraju dodavanju definiranih fragmenata. Fragment prikaza osobnih podataka se sastoji od CardViewa na kojem se nalaze elementi koji su ispunjeni osobnim podacima dohvaćenim iz baze podataka. Podatak s oznakom može sadržavati vrijednost 1 ili 2 ovisno o odabiru korisnika, želi li ažurirati podatke s lozinkom ili bez nje. Oznaka 1 označava ažuriranje podataka bez lozinke, dok oznaka 2 ažurira sve podatke. Uspješnim ažuriranjem ispisuje se poruka o pozitivnom spremanju podataka, u protivnome će se ispisati poruka o pogrešci.

```
// Azuriranje administratorskih podataka

private void updateData(int oznaka, boolean potvrda,
                                    int id, String ime,String prezime,
                                    int broj, String email, String lozinka){
    Call<ResponseBody> call = null;
    if(oznaka == 1) {call = iAdmin.updateAdminAllData(potvrda,
                            id,ime,prezime,broj,email,lozinka)};
    else if (oznaka == 2) {call =
                    iAdmin.updateAdminData(potvrda, id, ime, prezime, broj, email)};

    call.enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse(Call<ResponseBody> call,
                                     Response<ResponseBody> response){
            if(response.isSuccessful()){
                new SweetAlertDialog(getContext(),SweetAlertDialog.SUCCESS_TYPE)
                        .setTitleText("Uspješno promijenjeno!")
                        .show();
            }
            else if(response.code() == 400){
                Toast.makeText
                   (getContext(),"Ažuriranje nije uspjelo!",Toast.LENGTH_SHORT).show();

                new SweetAlertDialog(getContext(),SweetAlertDialog.ERROR_TYPE)
                        .setTitleText("Provjerite e-mail adresu!")
                        .setConfirmText("U redu")
                        .show();
            }
        }

        @Override
        public void onFailure(Call<ResponseBody> call, Throwable t) {
            t.getMessage();
            t.printStackTrace();
        }
    });
}
```

Programski kod 29. Prikaz ažuriranja administratorskih podataka

6.3.14 Unos predstave i ponuda

Administratorima je omogućen unos predstava i pripadnih ponuda. Aktivnost sadrži navigaciju koja omogućava prikaz fragmenata. U ovom slučaju, aktivnost prikazuje jednu od dva izrađena fragmenta unosa.

Unos predstave podrazumijeva ispunu svih podataka o predstavi. Jednom izrađena predstava omogućuje unos ponude izvođenja. Sučelje je izrađeno CardView karticom na koju se slažu preostale komponente unosa. Svi podaci vezani za detalje predstave su predstavljeni EditText pogledima. Unutar njih administrator unosi tekstualne podatake vezane za predstave. Slike su realizirane ImageVIew pogledima. Na kraju se postavlja gumb preko kojeg je omogućen unos predstave. Program prije unosa provjerava popunjenost polja detalja predstave.

Izgled sučelja za unos ponuda prikazuje informacije rabeći CardView. CardView postiže dosljedan izgled dodavanjem okruglih kuteva i sjena. Na njega se dodaju ostali grafički elementi namijenjeni prikazu i unosu ponuda. Popis predstava je predstavljen padajućim izbornikom (eng. Spinner) koji pruža odabir jedne stavke iz skupa. Lista se puni s podacima o nazivima predstava iz baze podataka. Funkcionalnosti odabira vremena i datuma izvođenja predstave su realizirani pogledima TimePicker i DatePicker. Polja za unos podataka su postignuta elementom EditText. Opcije premijere su predočene elementom prekidača (eng. Switch). Ispunjavanjem podataka, ponuda se sprema klikom na gumb „Spremi“.

Program izvršava provjeru jesu li svi podaci ispunjeni i provjerava dostupnost dvorane za odabrani termin. Algoritam radi na principu uspoređivanja odabranog datuma i vremena prikazivanja s postojećima iz baze podataka. Dostupnost dvorane se procijenjuje na 2h (cca. vrijeme trajanja predstave + vrijeme nakon završetka koji je predviđen za odlazak posjetitelja) što znači da se ne mogu unijeti predstave koje imaju manji razmak između prikazivanja. Stoga se prvo provjerava jesu li jednaki datumi. Ukoliko jesu, gleda se razlika između vremena prikazivanja trenutne ponude s onom iz baze koja ima jednak datum. Razlika vremena se pretvara u milisekunde i uspoređuje s unaprijed definiranim propisom između predstava. Ukoliko su svi zahtjevi ispunjeni, aplikacija otvara skočni prozor u kojem dodatno provjerava i potom sprema u bazu. U slučaju ne ispunjenosti zahtjeva, aplikacija šalje poruku o zauzetosti dvorane i poruku administratoru o promjeni termina prikazivanja predstave.

6.3.15 Unos i pregled cijena

Aktivnost sadrži popis svih cijena koje se koriste za izračun konačne cijene rezervacije ulaznica. Glavni dio se odnosi na prikaz spremljenih cijena po datumima. Spremljene cijene su prikazane pomoću Expandable RecyclerView-a. Detalji cijena su izrađeni u posebnom odvojenom layoutu koji sadrži CardView. CardView je primjer widgeta pomoću kojega se postiže dosljedan prikaz podataka tako što je moguće postaviti zaobljenost kuteva i sjenu. CardView je ugrađen u postojeći RecyclerView čime je određen konačni prikaz detalja o cijenama. Kako bi prikazali podatke potreban je adapter i ViewHolder koji imaju ulogu prikaza podataka odnosno informacija na sučelju. Izradom adaptera je potrebno nadjačati tri ključne metode:

1. onCreateVIewHolder(): metoda inicijalizira ViewHolder.
2. onBindViewHolder(): metoda dohvaća podatke i popunjava elemente. Primjer: potrebno je popuniti tekstualna polja (TextView) s podacima o cijenama ulaznica po dobnim skupinama.
3. getItemCount(): dobiva se broj podataka.

1) Dodavanje cijena je realizirano skočnim prozorom koji se prikazuje nad istom aktivnosti. Prije unosa novih cijena, poziva se metoda *„clearEditText()“* koja briše prijašnje podatke iz tekstualnih polja. Po završetku unosa potrebno je spremiti cijene. Spremanjem program automatski dohvaća trenutni datum i vrijeme, te ga prosljeđuje prema bazi podataka s preostalim unesenim podacima.

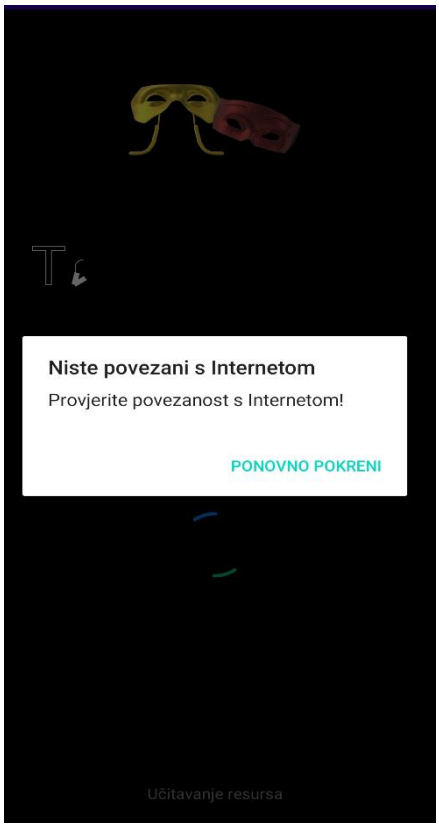
2) Brisanjem postojećih cijena dolazi do programskog odabira najnovije aktualne cijene. Izabrana cijena će biti s najnovijim datumom unosa. Svakim izbrisanim podatkom je potrebno ažurirati RecyclerView radi izbacivanja retka s popisa.

3) Sortiranje pruža silazno odnosno uzlazni prikaz cijena s obzirom na datum unosa.

7. Korisničke upute

Pokretanjem aplikacije otvara se početni zaslon aplikacije unutar kojega se učitavaju resursi. Dodatno se dinamički ispisuje tekst naziva aplikacije. Shodno tome se provjerava povezanost s internetom. Provjera se vrši na dva načina. Prvi način je ispitivanje aplikacije o postojanju povezanosti s internetom putem Wifi-a, a u drugom koraku povezanost s mrežom preko mobilnih podataka. Ukoliko ne postoji povezanost, aplikacija javlja pogrešku, te nije moguće nastavit s radom.

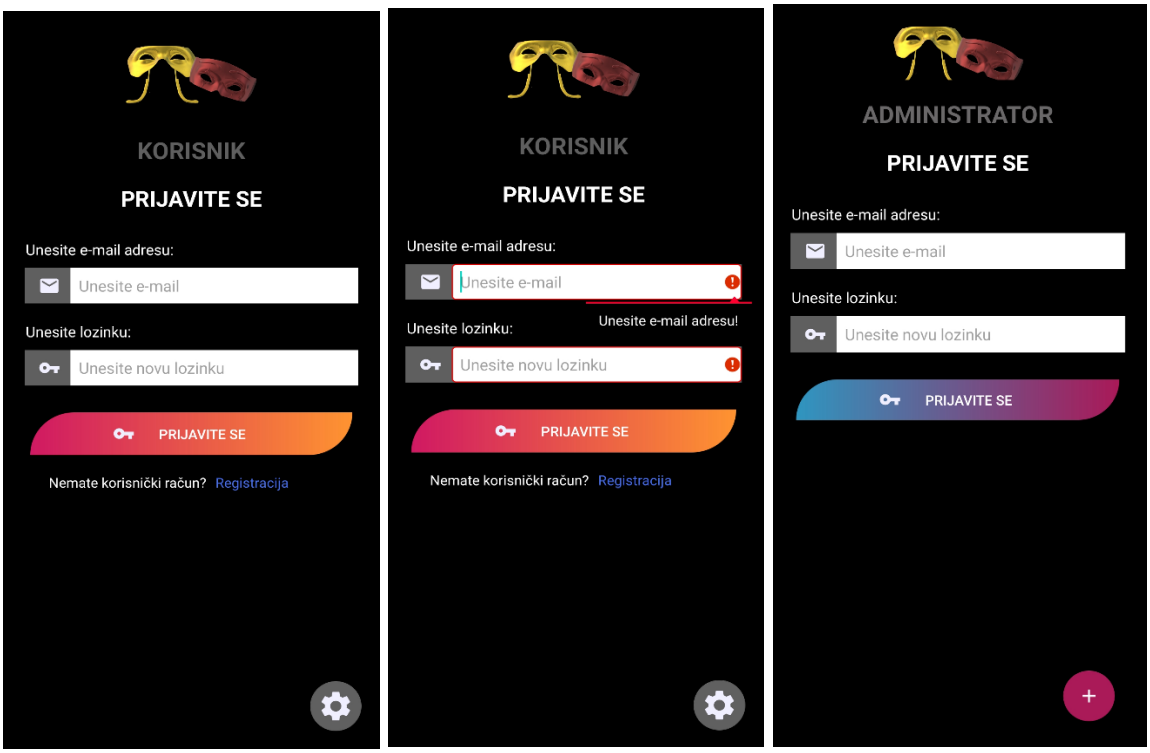
Slika 50. Početni zaslon aplikacije



Slika 51. Nema internetske veze

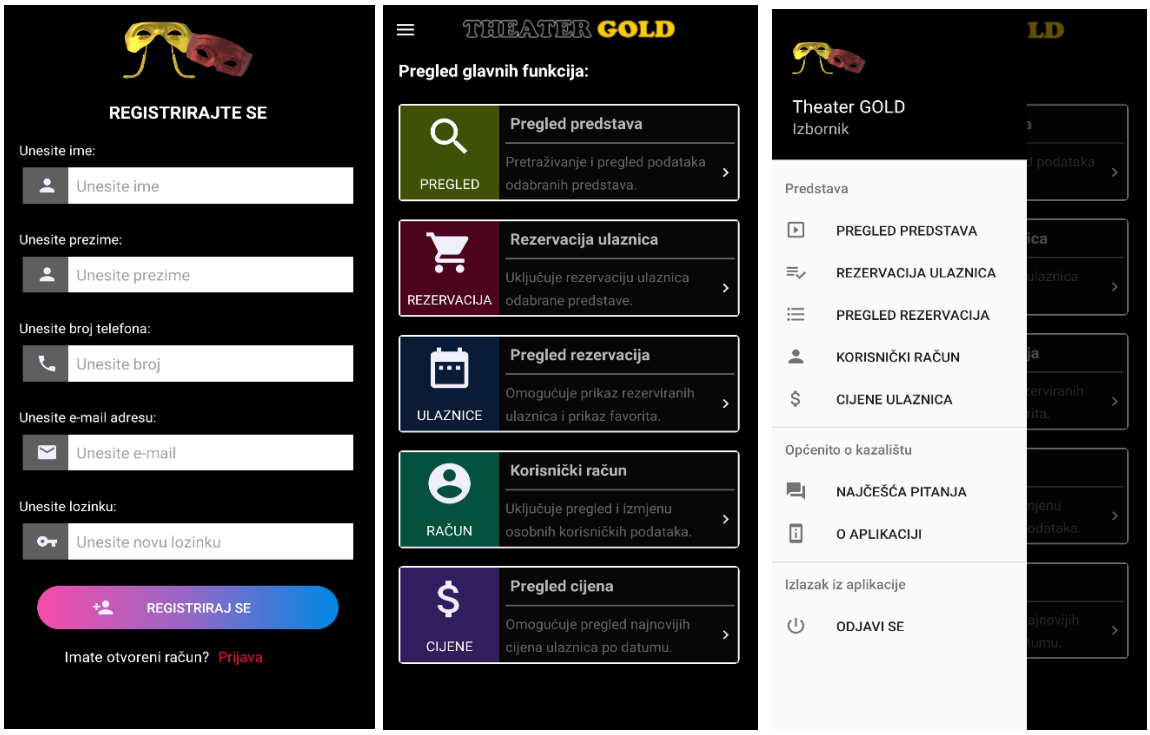
1. Početni zaslon, prijava i registracija korisnika

Ulaskom u aplikaciju, pojavljuje se zaslon za prijavu korisnika u sustav. Ukoliko korisnik ima otvoreni račun, potrebno je ispuniti podatke elektroničke pošte i odgovarajuće lozinke. Aplikacija ne dozvoljava prazna polja, te e-mail adresa mora biti napisana u ispravnom obliku. U protivnom će aplikacija javiti poruku o pogrešci, te će biti potrebno unijeti podatke odnosno zamijeniti one koji nisu ispravno upisani. Donji pomični gumb služi za prelazak na administratorski dio prijave. U slučaju da korisnik nema otvoreni račun, potrebno je izvršiti proces registracije. Klikom na tekst „Registracija“ otvara se novi zaslon s podacima za registraciju.



Slika 52. Prikaz zaslona za prijavu korisnika i administratora

Korisnik popunjava podatke vezane za: ime, prezime, broj telefona, e-mail adresu i lozinku. Ispunom svih polja pokreće se proces registracije klikom na gumb „Registriraj se“. Rezultat uspješne registracije je ulazak u aplikaciju. U slučaju korisničkog vraćanja na zaslon prijave, potrebno je kliknuti na tekst „Prijava“.

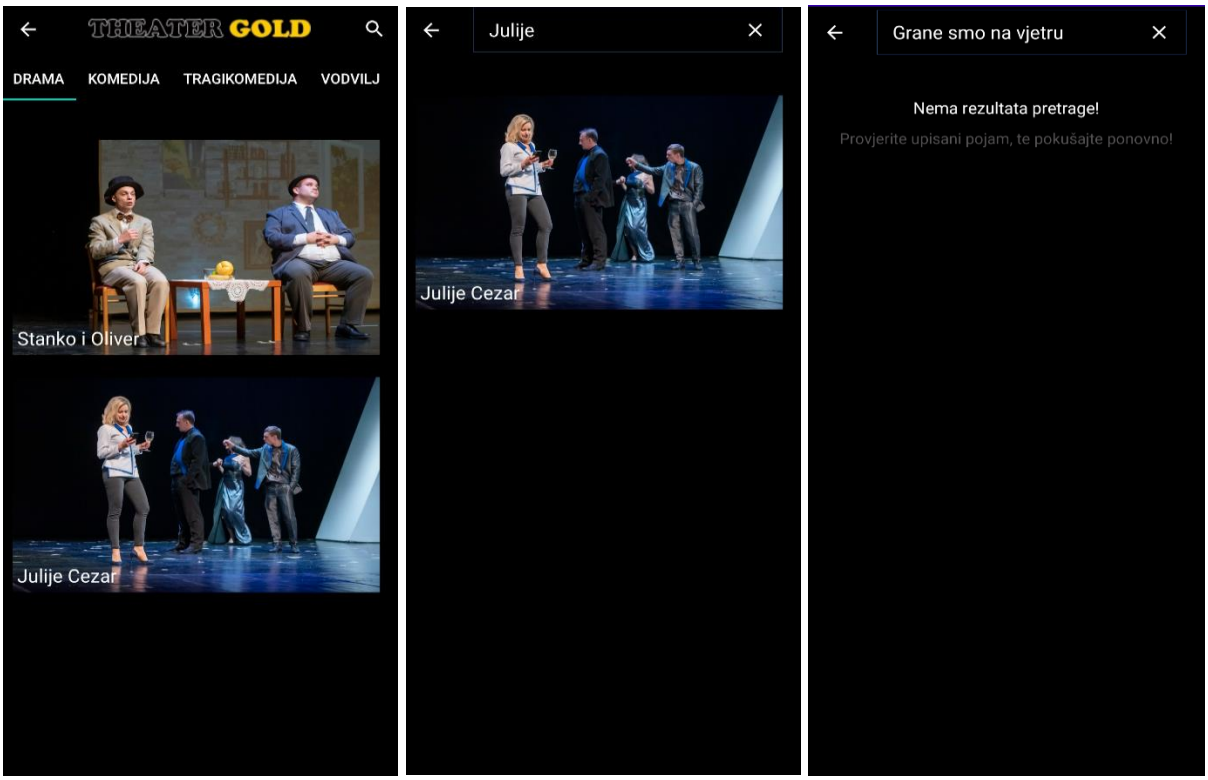


Slika 53. Prikaz zaslona registracije, početnog zaslona i izbornika aplikacije

3. Pregled predstava – korisnik

Odabirom opcije o pregledu predstave otvara se zaslon u kojem se mogu pregledati sve predstave po kategorijama. Kategorije se odnose na različite žanrove: drama, komedija, tragikomedija, vodvilj, glazba, melodrama, balet, parodija i opera. Odabirom jedne od ponuđenih, dohvaćaju se predstave u toj kategoriji.

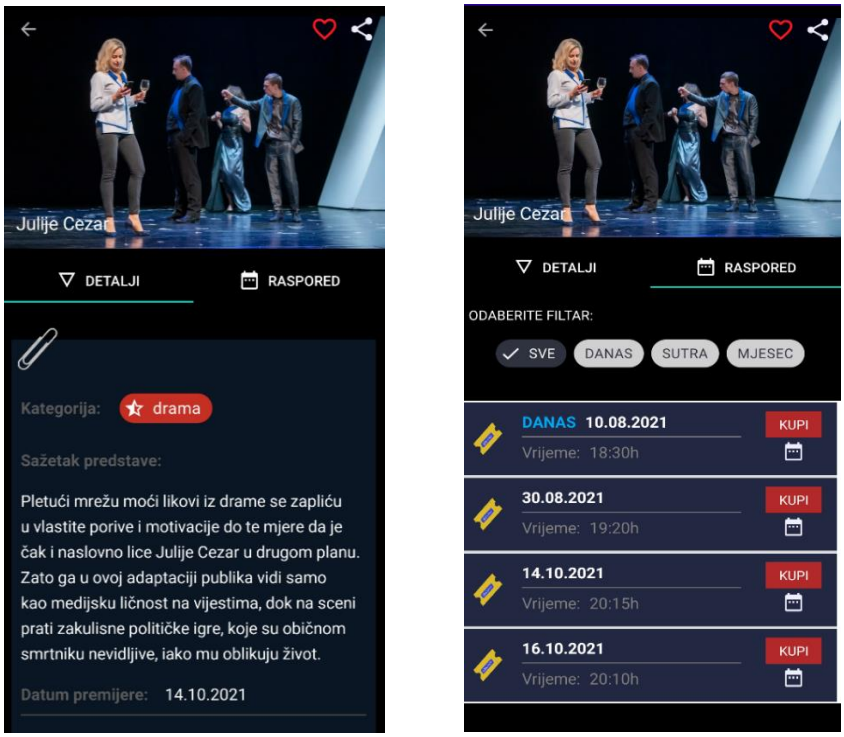
Dodatna pomoć prilikom traženja predstava je realizirana u obliku tražilice upisom zahtjeva naziva tražene predstave. Potvrdom unosa dohvaćaju se predstave s jednakim nazivom. Ako naziv nije pronađen, aplikacija javlja da nema rezultata.



Slika 54. Pretraživanje po kategorijama i naslovu

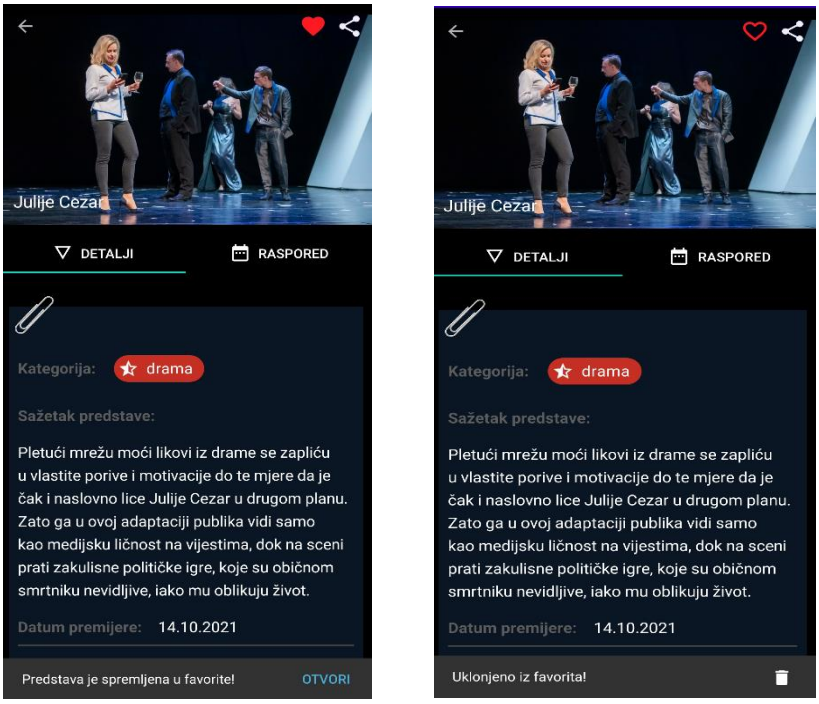
Korištene slike iz predstava su preuzete sa stranice kazališta „Zorin dom“. [32] [33]

Odabirom jedne od predstava otvara se zaslon s prikazom detalja te predstave. Korisniku su stavljene na raspolaganje dvije mogućnosti: pregled detalja i pregled rasporeda izvođenja predstave. Detalji obuhvaćaju sve podatke vezane uz predstavu. Dodatne opcije su dodavanje predstave u favorite i dijeljenje s prijateljima preko elektroničke pošte i društvenih mreža. Raspored prikazuje sve datume izvođenja odabrane predstave. Dodatno je moguće filtrirati raspored izvođenja. Filtriranje pruža odabir filtera: sve, danas, sutra i mjesec. Klikom na jedan od ponuđenih, otvara se zaslon s s podacima vezanih za rezervaciju ulaznica odabrane predstave.



Slika 55. Pregled detalja i rasporeda predstave

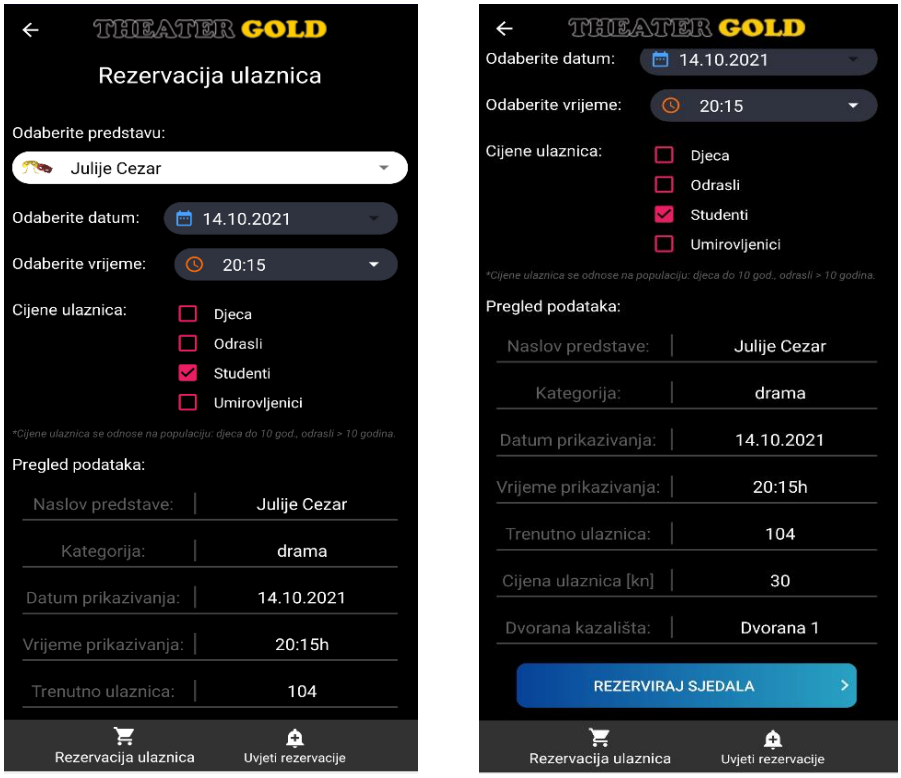
Prilikom spremanja predstave u favorite, program gleda je li takva već spremljena na listu od strane korisnika. U slučaju da nije, ikona favorita će biti prazno srce u kojem korisnik nakon klika može spremiti predstavu na listu. Međutim, ako je predstava već spremljena, ikona favorita je ispunjena i korisnik klikom na srce može brisati.



Slika 56. Dodavanje i uklanjanje predstave iz favorita

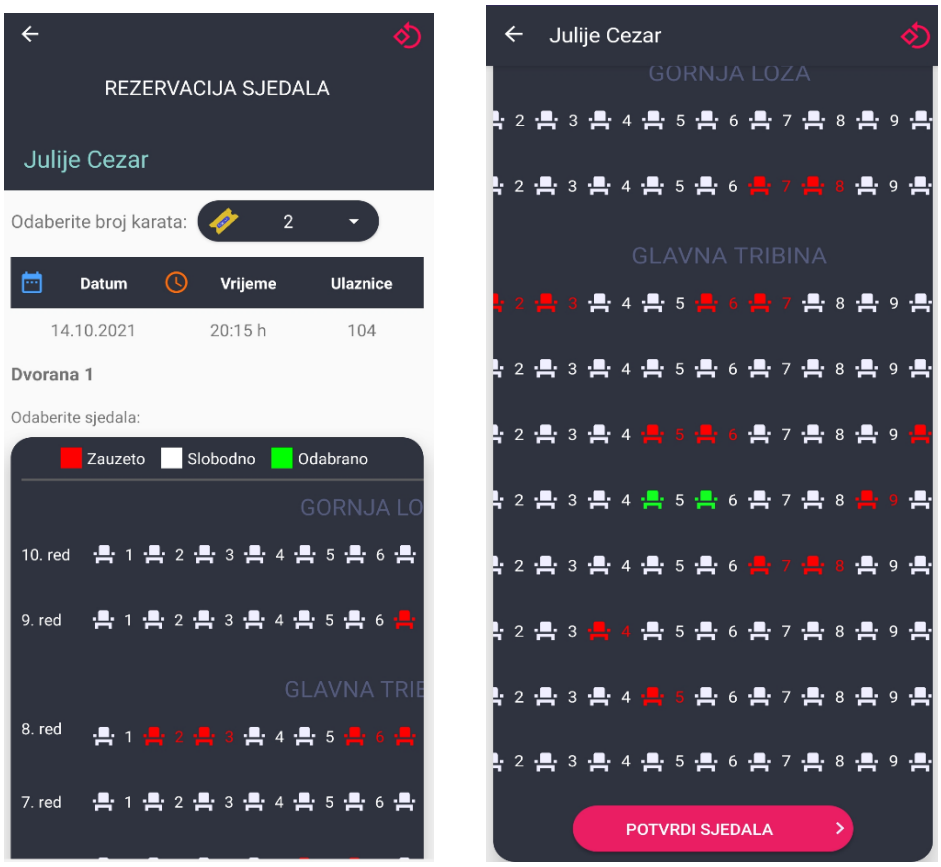
4. Rezervacija karata

Unutar aplikacije postoje tri glavna dijela kroz koja korisnik treba proći kako bi rezervirao ulaznice za predstavu. Prvi dio podrazumijeva odabir: predstave, datuma i vremena izvođenja, te dobne kategorije. Lista s predstavama se odnosi na sve predstave za koje postoji raspored izvođenja. Datum i vrijeme predstavljaju informaciju o početku izvedbe. Dobna kategorija služi za obračun cijene rezerviranja ulaznica. Odabirom navedenih stavki, korisniku se stavlja na raspolaganje pregled detalja ponude. Pregled prikazuje informacije o naslovu predstave, kategoriji, datumu i vremenu izvođenja, broju trenutnih ulaznica, cijeni ulaznica, te dvorani. Klikom na gumb „Rezerviraj sjedala“ potvrđuje se istinitost podataka, te se odobrava prelazak na drugu fazu rezervacije.



Slika 57. Prikaz postupka rezerviranja ulaznice

Drugi dio se odnosi na rezervaciju sjedala u kazališno koncertnoj dvorani. Dvorana sadrži 120 sjedećih mjesta raspoređenih na 10 redova po 12 sjedala koje su dostupne korisniku za rezerviranje. Jednostavniji pregled kazališne dvorane i pripadnih sjedala omogućeni su korištenjem vertikalnog i horizontalnog zakretanja zaslona. U alatnoj traci je moguće pronaći gumb koji će izvršiti operaciju zakretanja. Na korisniku je da sam odluči kako želi postaviti zaslon.



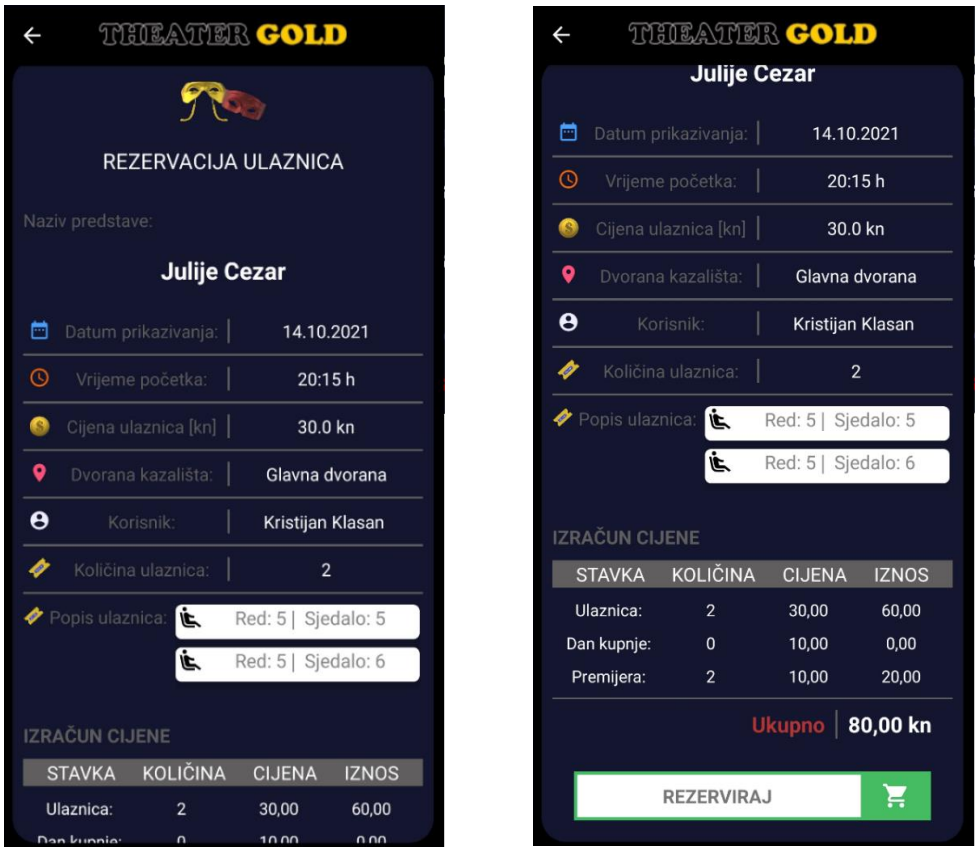
Slika 58. Vertikalni prikaz odabira sjedala

Slika 59. Horizontalni prikaz zaslona odabira sjedala u dvorani

Definiranjem položaja zaslona, prvi korak je odabir količine ulaznica. Količina ulaznica je limitirana, pa stoga korisnik može rezervirati četiri ulaznice odjedanput. U slučaju da se želi odabrati više od četiri ulaznice, onda je potrebno ponoviti postupak rezervacije. Prilikom odabira količine ulaznice treba se paziti na trenutni broj ulaznica. U slučaju

zauzetosti svih mjesta, aplikacija obavještava korisnika o nemogućnosti rezerviranja ulaznica za tu predstavu. Prije odabira sjedala, potrebno je proučiti legendu koja ukazuje na pravila statusa sjedala. Crvena boja označava zauzetost sjedala, bijela slobodno mjesto, a zelena trenutno odabrano sjedalo. Odabirom slobodnog sjedala, korisnik može uočiti promjenu boje iz bijele u zelenu, što označava trenutno odabrano mjesto za rezervaciju. Korisnik potvrđuje mjesta klikom na gumb „Potvrdi sjedala“. U slučaju da korisnik nije odabrao sjedalo, aplikacija obaviještava o potrebi odabira barem jednog sjedala za nastavak rezervacije.

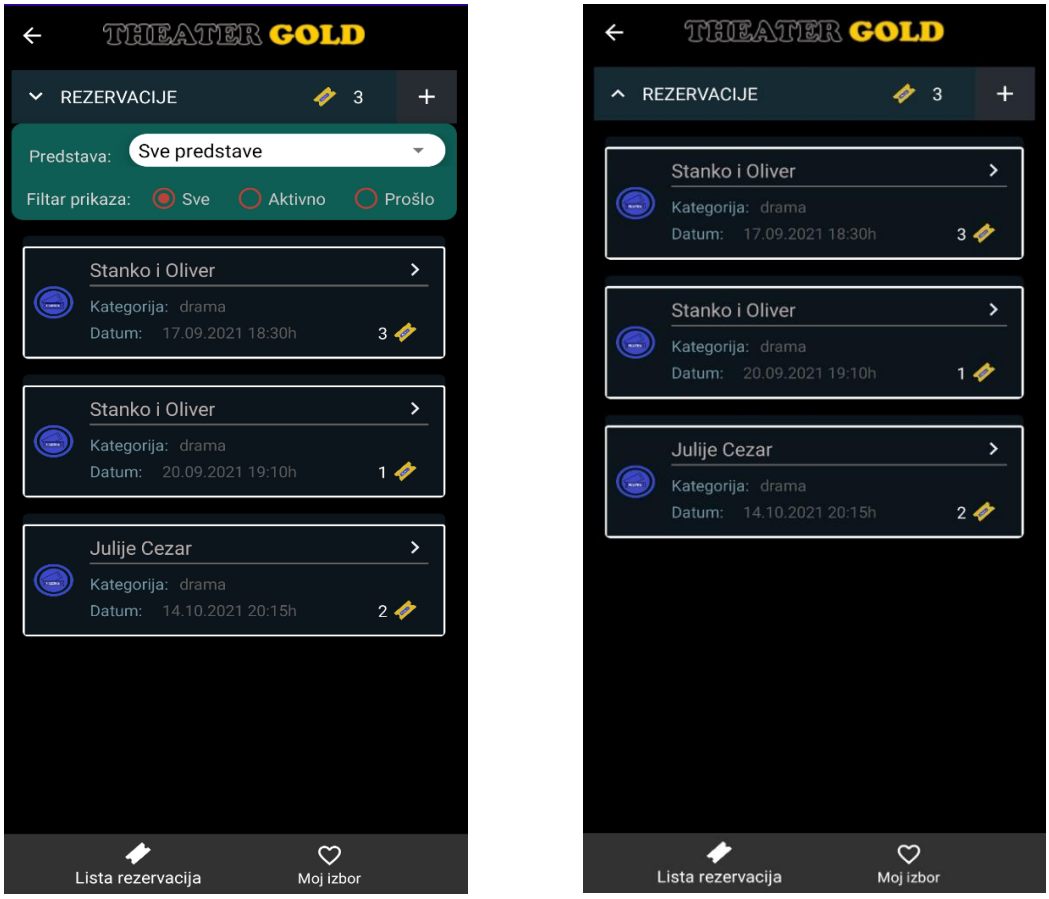
Treći dio se odnosi na pregled svih podataka rezervacije. Podaci počinju od naslova predstave, pa se prosežu sve do popisa rezerviranih sjedala. Poslije detalja se nalazi obračun cijena koji se sastoji od količine ulaznica, cijena s obzirom na odabranu dob, te ukupnog iznosa rezervacije. Ukupni iznos se može pronaći ispod obračuna. Klikom na gumb „Rezerviraj“ pokreće se proces rezervacije ulaznica, te aplikacija potom informira korisnika o uspješnoj rezervaciji.



Slika 60. Pregled detalja i završetak rezervacije

5. Pregled rezervacija

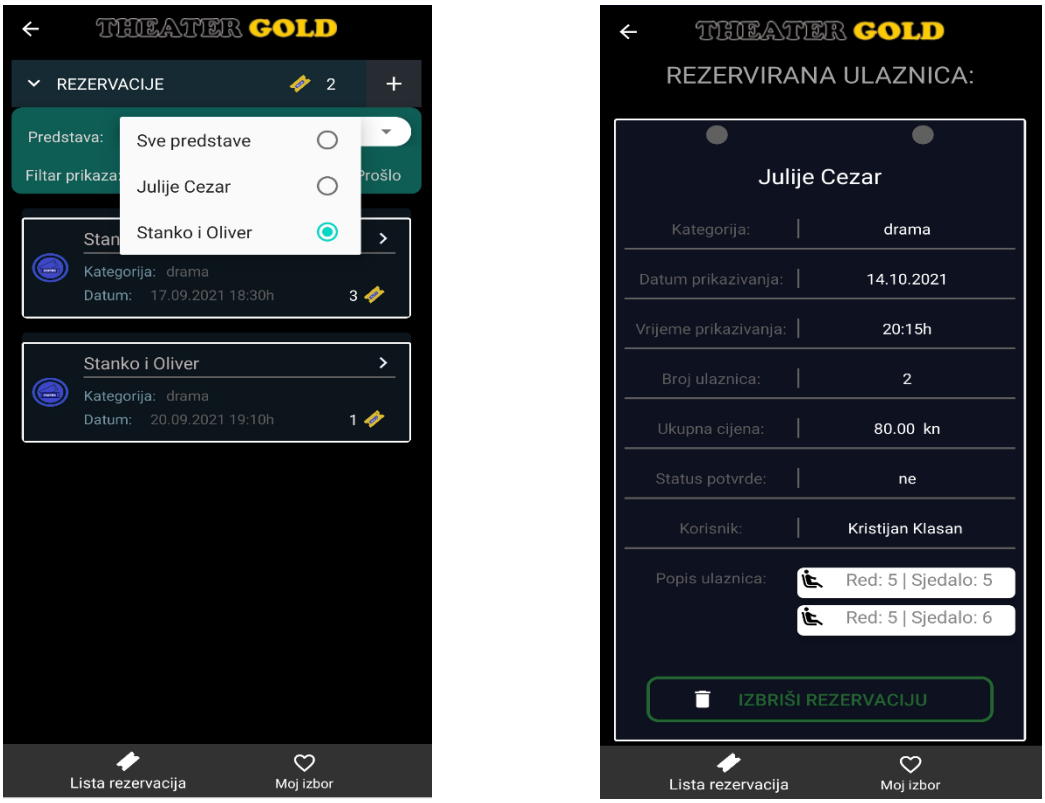
Pregled rezervacija daje prikaz svih rezerviranih ulaznica s obzirom na datum izvođenja predstave. Radi lakšeg pregleda postavljeni su filtri preko kojih korisnik filtrira predstave. Klikom na znak donje strelice, mogućnost filtriranja se uključuje odnosno isključuje. Ikona sa znakom + omogućuje prelazak na zaslon za rezervaciju.



Slika 61. Prikaz rezerviranih ulaznica i isključivanja opcije filtriranja

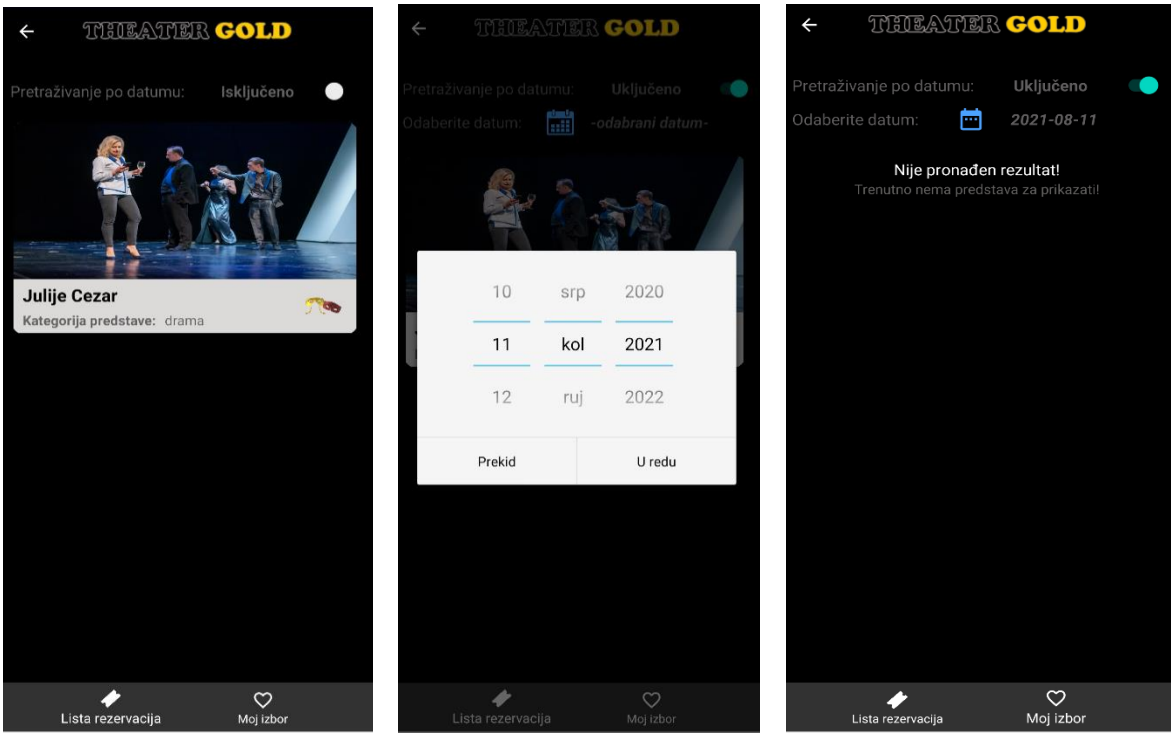
Filtri pružaju opcije prikaza: svih, aktivnih i prošlih (završene predstave) rezervacija. Dodatno je stavljeno na raspolaganje i filtriranje uporabom naziva predstave. Filter naziva predstave sadrži popis predstava za koje postoji rezervacija. Odabirom jedne od ponuđenih predstava, dohvaćaju se i prikazuju rezervacije samo za tu predstavu.

Korisnik može pregledati detalje svoje rezervacije, te po želji otkazati rezervaciju. Detalji obuhvaćaju sve one podatke vezane za rezerviranu ulaznicu počevši od kategorije do popisa sjedala. Otkazivanjem rezervacije, otvara se prozor u kojem aplikacija pita korisnika je li zaista siguran u svoju odluku. Potvrdnim odgovorom, rezervacija se briše i rezervirano mjesto ponovno se stavlja na raspolaganje za rezervaciju. U suprotnome, korisnik uvijek može odustati od otkazivanja rezervacije.



Slika 62. Filtriranje rezervacija prema nazivu te prikaz detalja

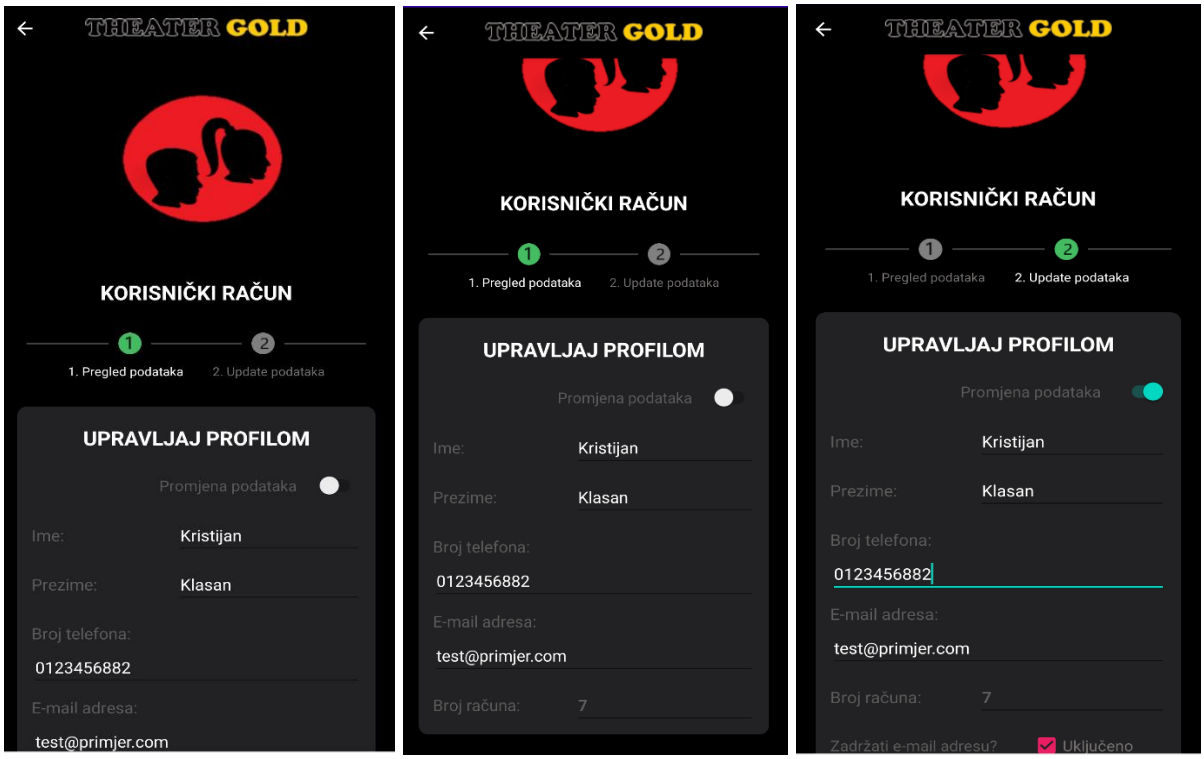
Moj Izbor odnosno favoriti je opcija koja omogućuje spremanje predstava na posebnu listu namijenjenoj kasnijem pregledu predstava. Ukoliko korisnika zainteresira neka predstava, a nije odlučio odmah rezervirati, istu može spremiti na listu.



Slika 63. Prikaz favorita i filtriranje favorita prema datumu

6. Korisnički račun

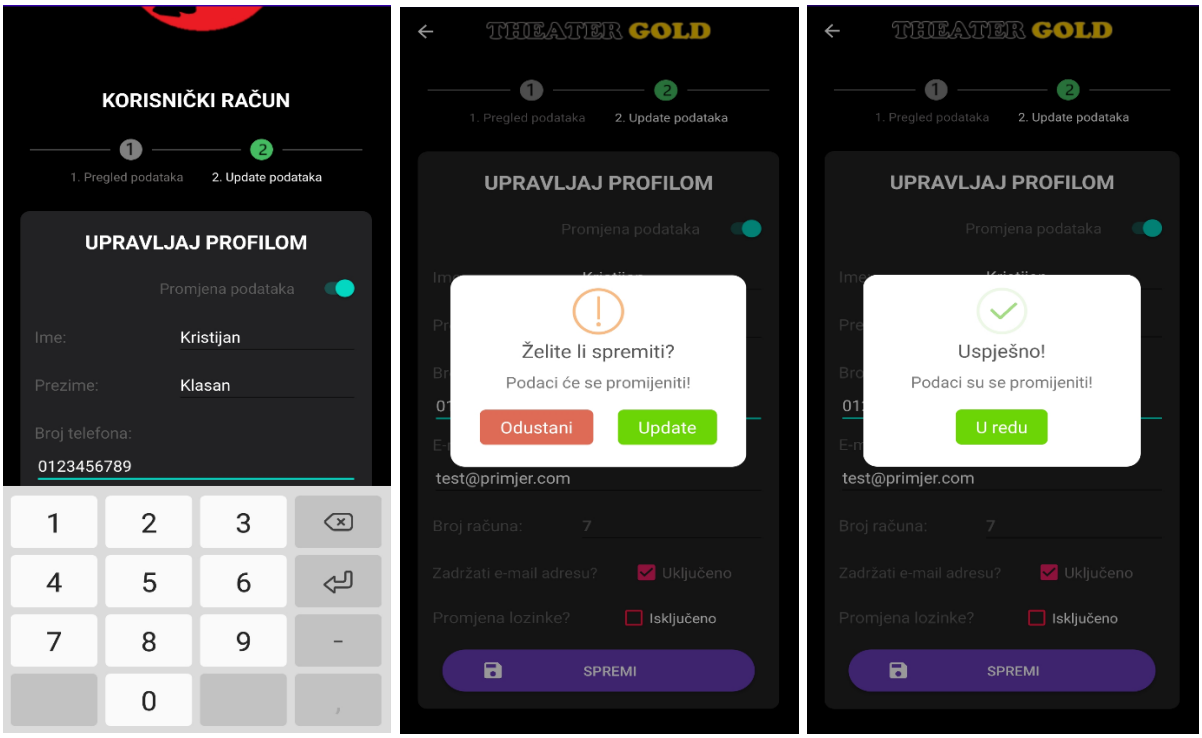
Daje prikaz i ažurira osobne podatke korisnika. Ovom stavkom se mogu prekontrolirati svi korisnički podaci na jednom mjestu. Ukoliko korisnik želi ažurirati podatke potrebno je uključiti opciju klikom na prekidač. Unutar statusne trake (ispuna s zelenom bojom) vidi se status uključene opcije.



Slika 64. Prikaz podataka o korisniku i uključivanje opcije ažuriranja

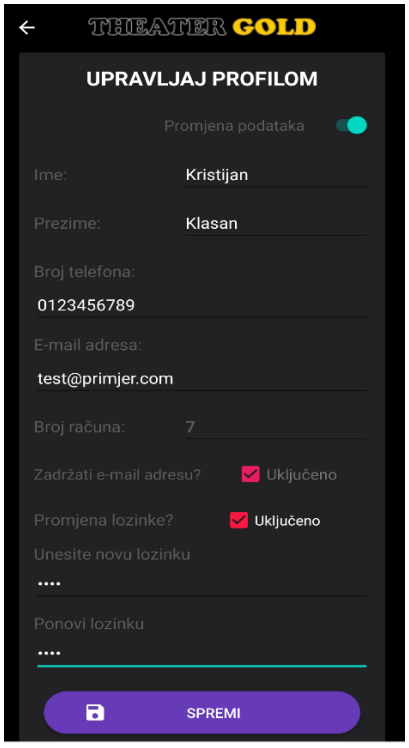
Uključivanjem ažuriranja, gumb za spremanje promjena postaje vidljiv. Korisnik mijenja podake za koje smatra da je potrebna izmijena, a po potrebi i lozinku.

Klikom na gumb „Spremi“ pokreće se proces spremanja svih podataka osim lozinke. Tijekom procesa se izvršava provjera utvrđivanja postojanje e-mail adrese u bazi podataka, te ispunjenosti podataka unutar polja. Zbog toga je vrlo važno naznačiti u aplikaciji želi li se promijeniti ili zadržati adresa. Zadržanjem jednake adrese se očekuje njeno postojanje u bazi i program neće vratiti pogrešku. Ako korisnik odbije zadržati staru adresu i želi unijeti novu, program prvo provjerava njeno postojanje. U slučaju postojanja, a nije naznačena da se želi zadržati, program javlja grešku koju korisnik mora ispraviti u obliku unosa nove e-mail adrese. Nakon svih korekcija, ponavlja se proces provjere i ispravljeni podaci se spremaju.

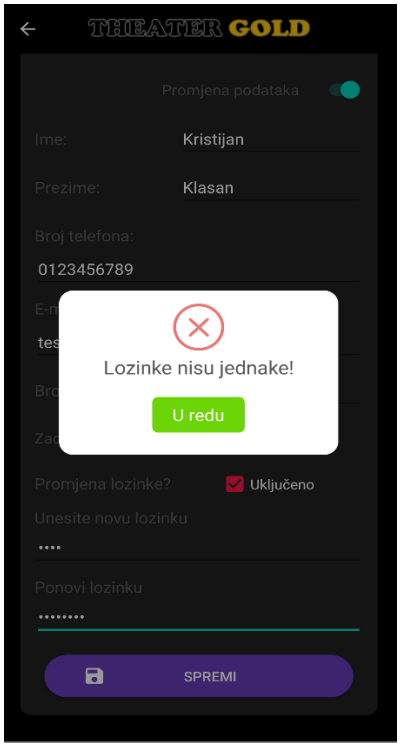


Slika 65. Prikaz ažuriranja podataka i spremanja promjena

Uključivanje opcije za promijenu lozinke se postiže postavljanjem kvačice na gumb. Rezultat ishoda je prikaz polja unutar kojih se unose dvije identične lozinke koje se planiraju u budućnosti koristiti u svrhu prijave u aplikaciju. Klikom na gumb „Spremi“ dohvaćaju se podaci iz preostalih polja, te se zajedno s lozinkama pohranjuju u bazu.



Slika 66. Prikaz promjene lozinke



Slika 67. Prikaz pogreške prilikom unosa

7. Pregled najčešćih pitanja

Ovaj dio prikazuje pregled pitanja i odgovora vezanih za opće korištenje aplikacije. Obuhvaća standardna pitanja poput načina rezervacije ulaznica i njihovo otkazivanje. Korisnik na jednom mjestu može saznati sva pravila otvaranjem pitanja i čitanjem pripadnog odgovora. Klikom na naslov pitanja otvaraju se detalji unutar kojih se nalazi odgovor na postavljeno pitanje.

Slika 68. Prikaz svih pitanja i detalja odabranog pitanja

8. Ostale korisničke funkcionalnosti

Pregled cijena daje prikaz najnovijih cijena rezervacija kazališnih ulaznica. Cijene su definirane po: dobnim kategorijama, naknadama za rezerviranje na dan izvođenja i premijeri predstave. Unutar svake je uključen PDV. Sve navedene cijene uključuju najnovije podatke s obzirom na datum unosa kojeg je moguće pronaći na začelju zaslona, definire od strane administratora. Stare cijene se neće prikazivati bez obzira na njihovo postojanje u bazi. Konačni iznos cijene rezervacije se obračunava temeljem stavki kategorija cijena, cijena o premijeri i cijena na dan kupnje.

Pregled informacija o aplikaciji izlistava sve tehničke podatke aplikacije, kao što su: SDK verzija, verzija operacijskog sustava, verzija aplikacije, model uređaja na kojem se aplikacija izvršava i način na koji je aplikacija spojena s mrežom (Wi-Fi ili mobilni podaci). Korisnik u ovom dijelu može čitati prikazane podatke o detaljima.

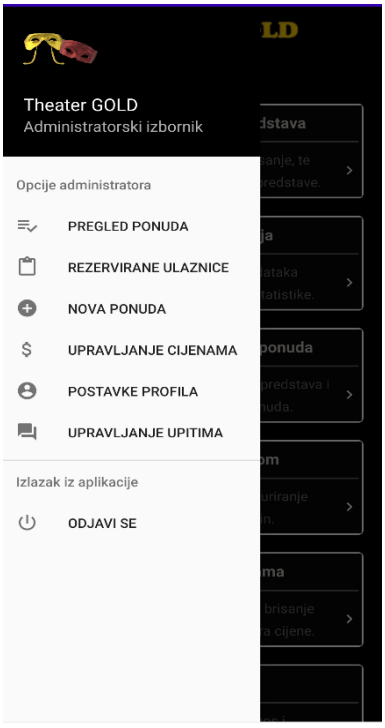
Slika 69. Pregled cijena – korisnik

Slika 70. Pregled informacija o aplikaciji

9. Administratorski izbornik

Nakon prijave administratora u aplikaciju, otvara se zaslon s izbornikom glavnih opcija.

Slika 71. Početni zaslon administratora

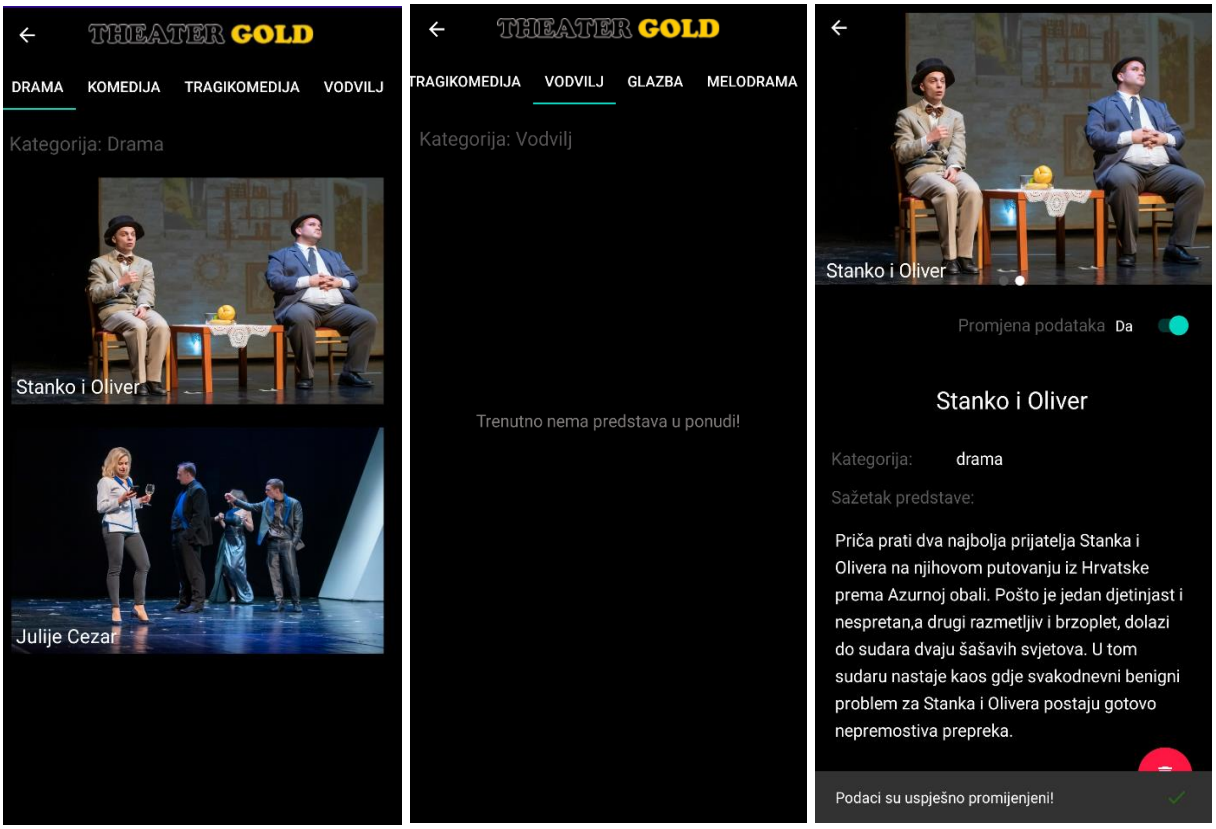


Slika 72. Prikaz administratorskog izbornika

10. Pretraživanje predstava – administrator

Odabirom opcije „Pregled predstava“ otvara se novi zaslon koji pruža administratoru prikaz svih predstava s obzirom na kategorije. Popis listajućih kategorija se može pronaći u alatnoj traci. Sveukupno, u aplikaciji postoji ugrađeno devet kategorija predstava. Klikom na kategoriju, dohvaćaju se sve predstave koje su spremljene pod njom. Administrator potom bira predstavu čije podatke želi pregledati ili izmijeniti. Klikom na predstavu, otvara se zaslon koji prikazuje detalje te predstave. Ako nema predstave pod kategorijom, aplikacija javlja porukom.

Predstave pod kategorijom „Drama“ su Julije Cezar, Stanko i Oliver. [32] [33]

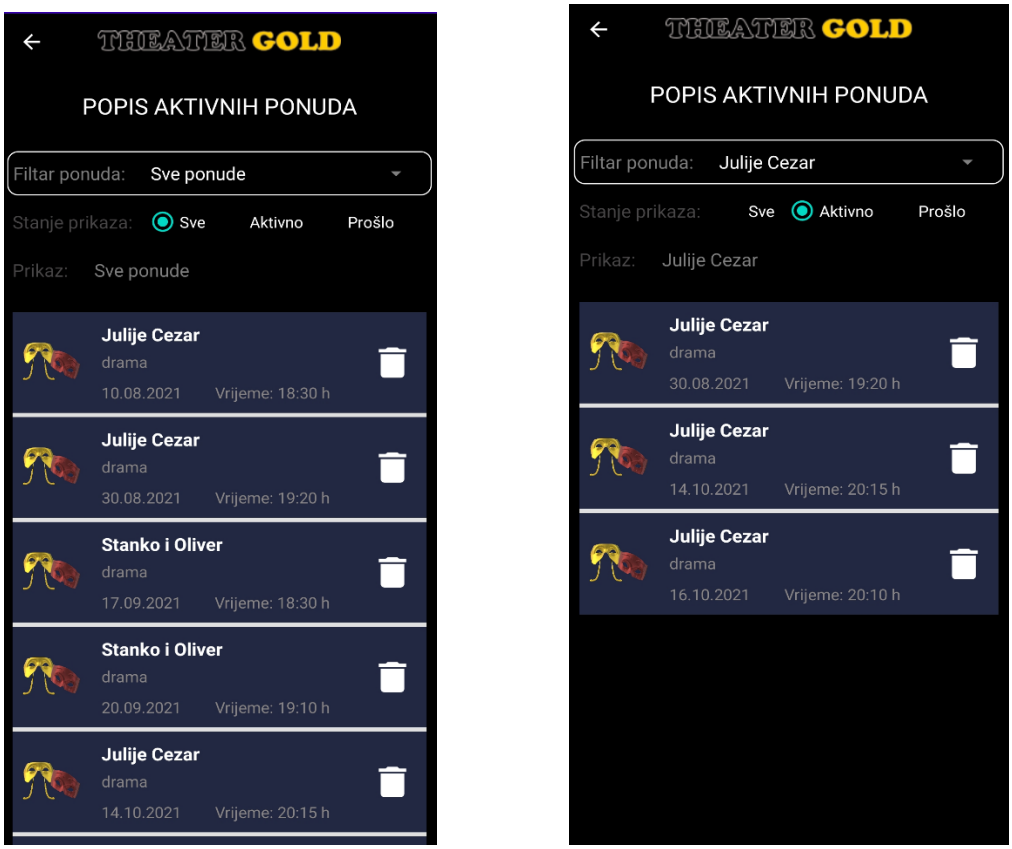


Slika 73. Prikaz procesa pretraživanja, odabira, te ažuriranja podataka

Kako bi se omogućilo ažuriranje podataka, potrebno je uključiti prekidač. Pritiskom na prekidač, omogućuje se izmjena teksta, postavlja se padajući izbornik za izmjenu kategorije predstave, te se prikazuje gumb za spremanje promjena. Administrator tada mijenja podatke za koje smatra da je potrebna izmjena. Dodatno se može promijeniti slika otvaranjem galerije i odabirom postojeće ili nove slike. Klikom na gumb „Spremi promjene“ pokreće se proces spremanja.

11. Pregled rezervacija

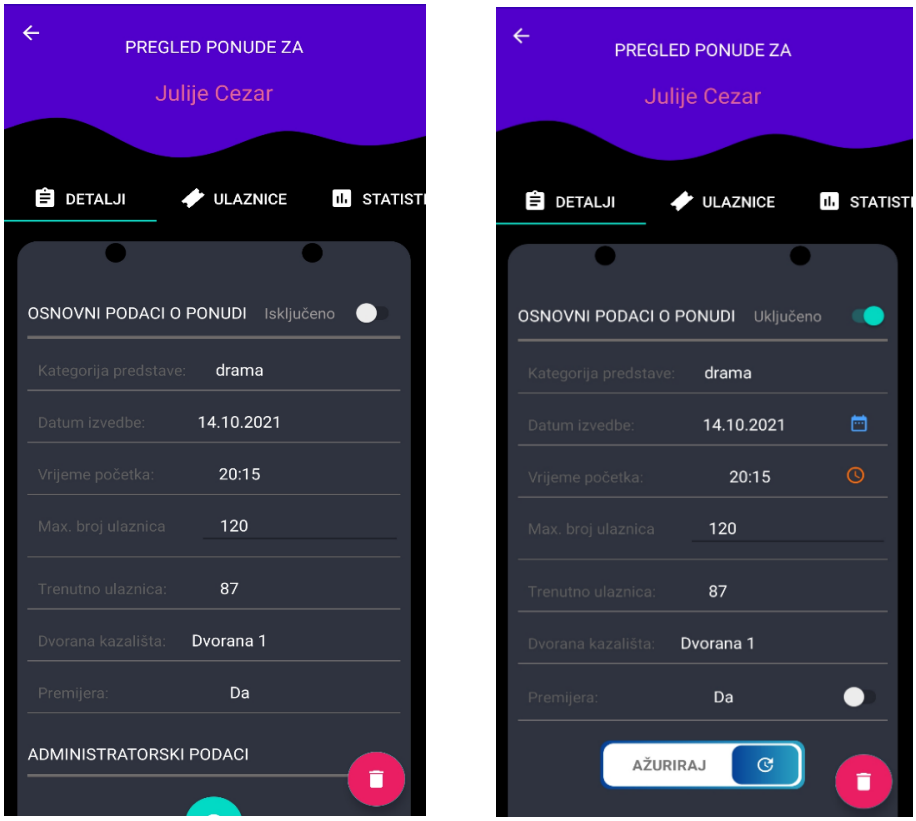
Omogućava prikaz svih ponuda predstava koje je prethodno administrator unio u sustav. Aplikacija povlači podatke o ponudama predstava koje su sortirane prema datumu i vremenu prikazivanja. Radi lakšeg prikaza rezultata, postoji komponenta padajućeg izbornika koja sadrži popis ponuđenih predstava. Dodatna opcija daje prikaze ponuda kroz tri stanja: sve, aktivno i prošlo, Opcija *„Sve“* služi prikazu svih ponuda predstava (aktivne + prošle ponude) koje postoje u bazi bez obzira na datum prikazivanja predstave. Filtar „Aktivno“ predstavlja sve predstave koje imaju nadolazeći datum. Zadnja opcija *„Prošlo“* prikazuje ponude koje posjeduju datum prikazivanja koji je manji od trenutnog datuma. Ponuda se briše klikom na ikonu za brisanje.



Slika 74. Prikaz svih ponuda predstava i filtrirani prikaz

Primjer: adminstrator bira željenu predstavu za koju želi pregledati ponude. Odabirom stanje „Sve“, prikazati će se ponude koje se odnose na tu predstavu (aktivne + prošle ponude). Za prikaz samo aktivnih ponuda potrebno je odabrati opciju „Aktivne“. Odabirom jedne od ponuđenih ponuda otvara se novi prozor koji prikazuje detalje. Otvaranjem se nudi mogućnost pregleda i ažuriranje ponuda, pregled rezerviranih ulaznica, te prikaz statistike.

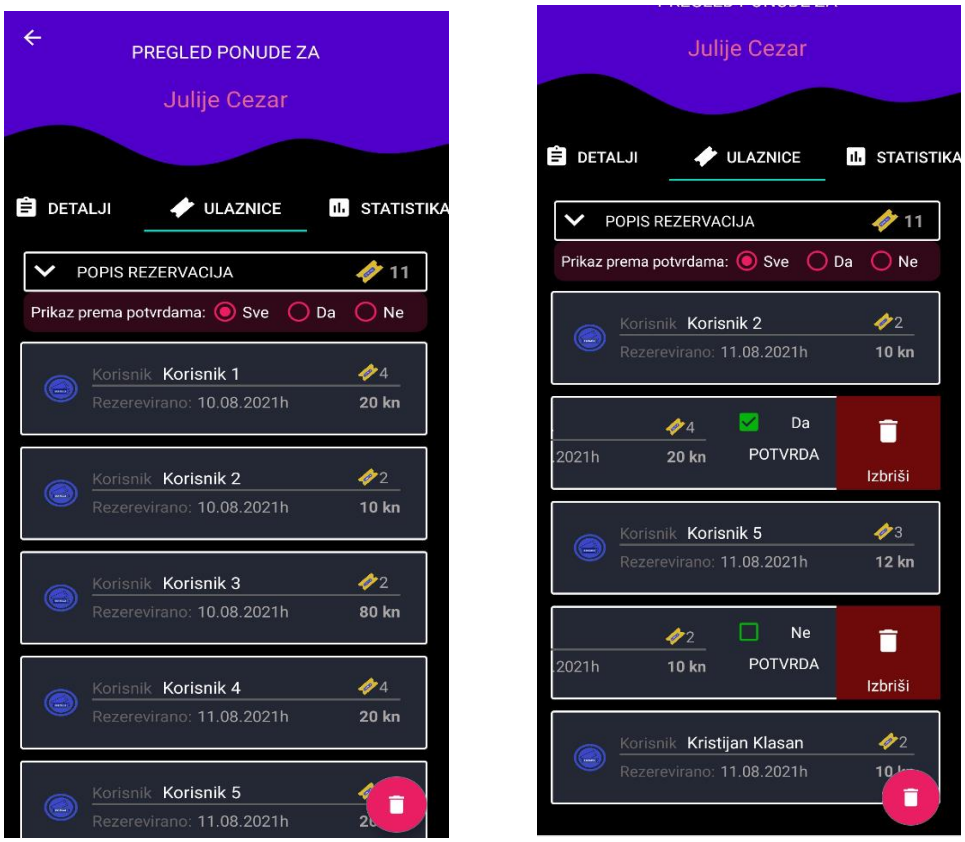
Pregled ponuda služi prikazu svih detalja ponude predstave. Podaci sadrže: naslov predstave, datum i vrijeme prikazivanja, premijeru, broj ulaznica, kategoriju predstave. Administratorski podaci prikazuju: ime i prezime, šifru, te e-mail administratora. Klikom na gumb (oznaka +) otvaraju se detalji administratora koji ukazuju tko je unio ponudu. Aplikacija omogućuje administratoru uređivanje podataka o ponudi. Potvrdnim odgovorom na elementu prekidača, otvaraju se polja za uređivanje podataka. Nakon uređivanja, dolazi do spremanja gdje se ponovno provjerava dostupnost dvorane. Ukoliko sve odgovara, promjene se spremaju.



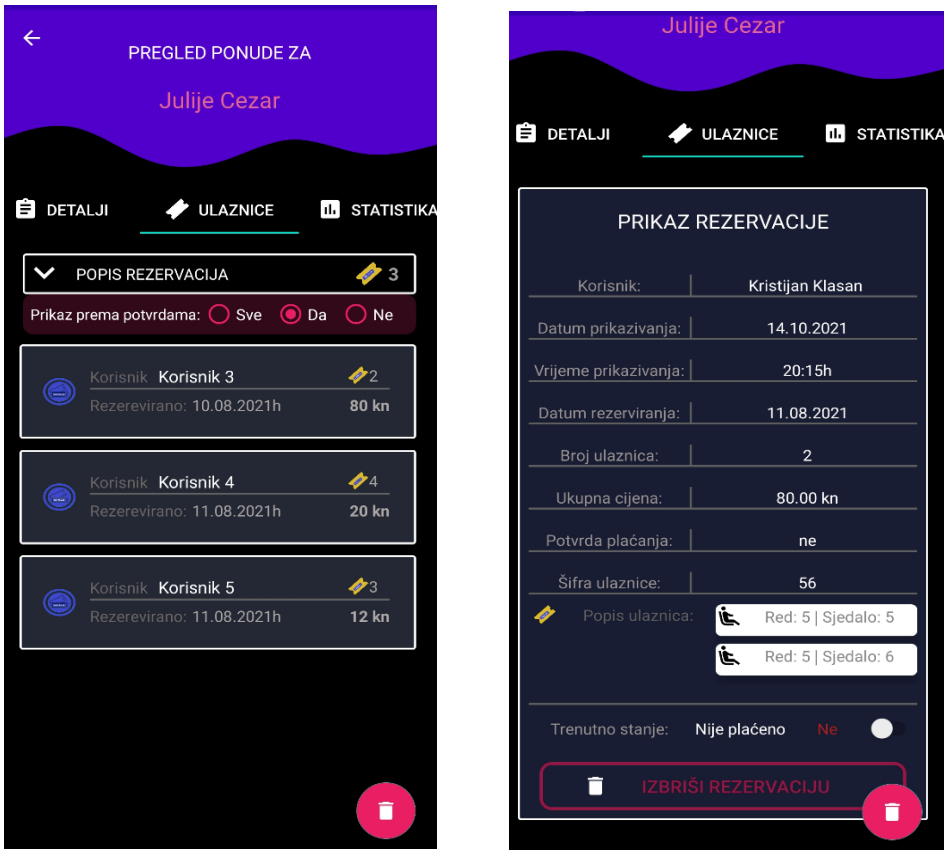
Slika 75. Pregled podataka i ažuriranje ponude predstave

Dodatna opcija je brisanje ponude iz baze podataka. Klikom na gumb za brisanje otvara se okvir koji provjerava želi li administrator izbrisati ponudu. Potvrdnim odgovorom, ponuda se briše, a time i podaci o rezervacijama.

Ulaznice pružaju uvid u rezervirane ulaznice odabrane ponude. One daju nekoliko važnih opcija, kao što su: pregled, otkazivanje i postavljanje statusa plaćanja. Prikaz podataka se prilagođava koristeći filtar prikaz. On obuhvaća filtriranje rezervacija s obzirom na status plaćanja. Unutar njega postoje tri opcije koje filtriraju: sve rezervacije, samo one koje su podignute i plaćene, te one koje nisu plaćene. Odabirom filtra se prikazuju rezervacije.



Slika 76, Prikaz rezervacija i dodatnih opcija upravljanja

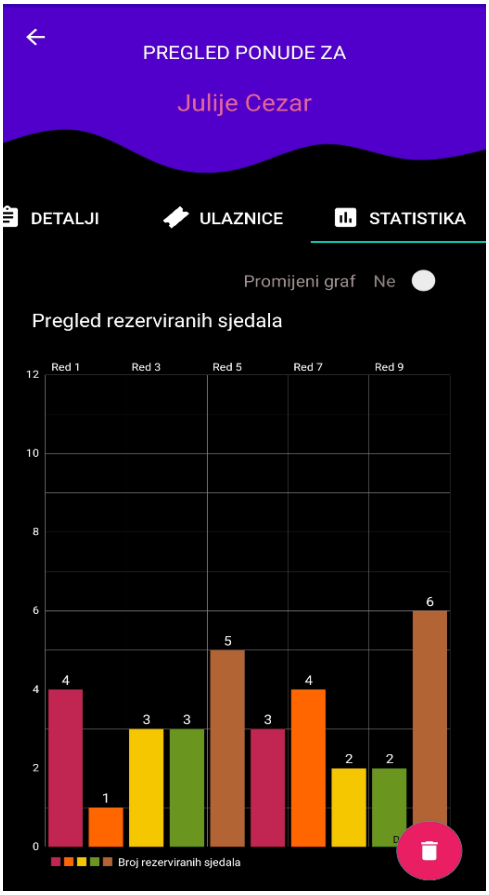


Slika 77. Filtriranje i prikaz detalja rezervacije

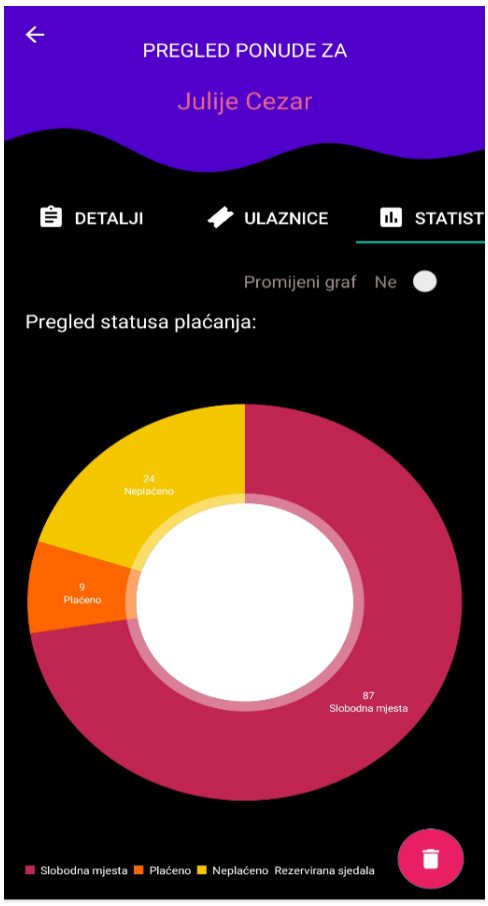
Svaka rezervacija se sastoji od od sljedećih podataka: ime i prezime korisnika, datuma rezervacije, ukupne cijene, te količine rezerviranih ulaznica. Povlačenjem rezervacije u desno, otkrivaju se dodatne funkcije. One služe potvrdi statusa plaćanja i brisanju rezervacije. Potvrda plaćanja je napravljena radi lakšeg bilježenja u trenutku plaćanja i zbog toga nije nužno potrebno otvarati detalje. Brisanje pojedine rezervacije kao rezultat uklanja rezervaciju s popisa, te umanjuje ukupni broj rezerviranih ulaznica.

Klikom na određenu rezervaciju, otvaraju se detalji rezervacije. Detalji pružaju jednake funkcije pregleda, brisanja i ažuriranja statusa plaćanja. Postojanje ove opcije je realizirano radi jednostavnosti pregleda svih podataka.

Statistika pruža prikaz podataka uporabom dinamičnih grafikona. Svaki dio grafikona je predstavljen zasebnom bojom radi isticanja različitosti podataka. Ispod njega se nalazi legenda koja pomaže u boljem razumijevanju. Prvi je stupčasti grafikon (eng. Bar Chart) koji se odnosi na pregled rezerviranih sjedala po redovima unutar dvorane određene predstave. Drugi je tortni grafikon (eng. Pie Chart) koji prikazuje količinu slobodnih mjesta i broj plaćenih odnosno neplaćenih rezervacija.



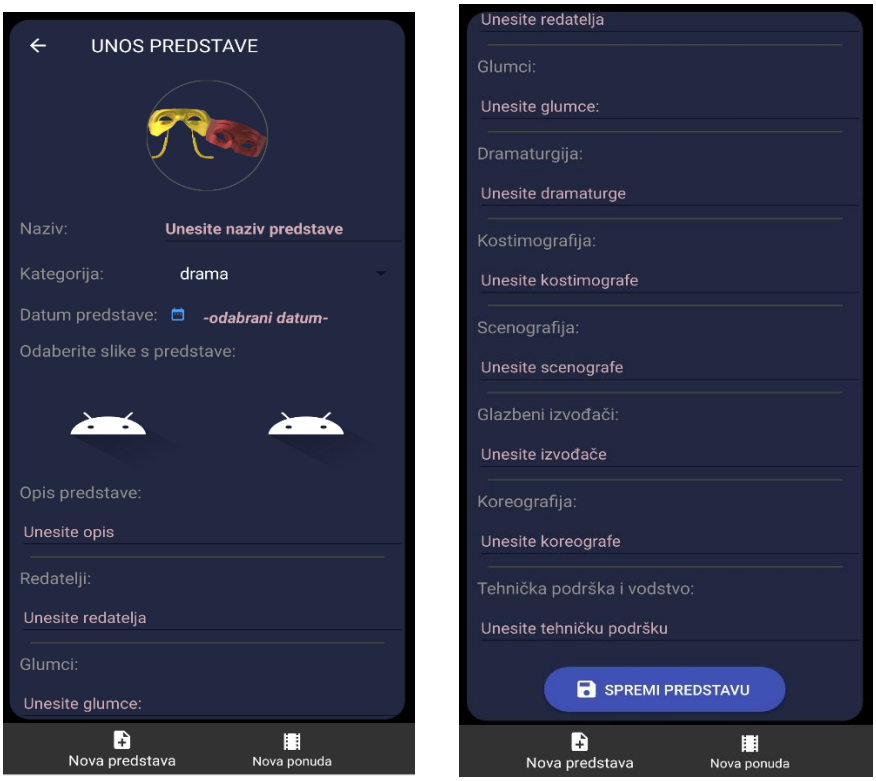
Slika 78. Prikaz stupčastog grafikona



Slika 79. Prikaz tortnog grafikona

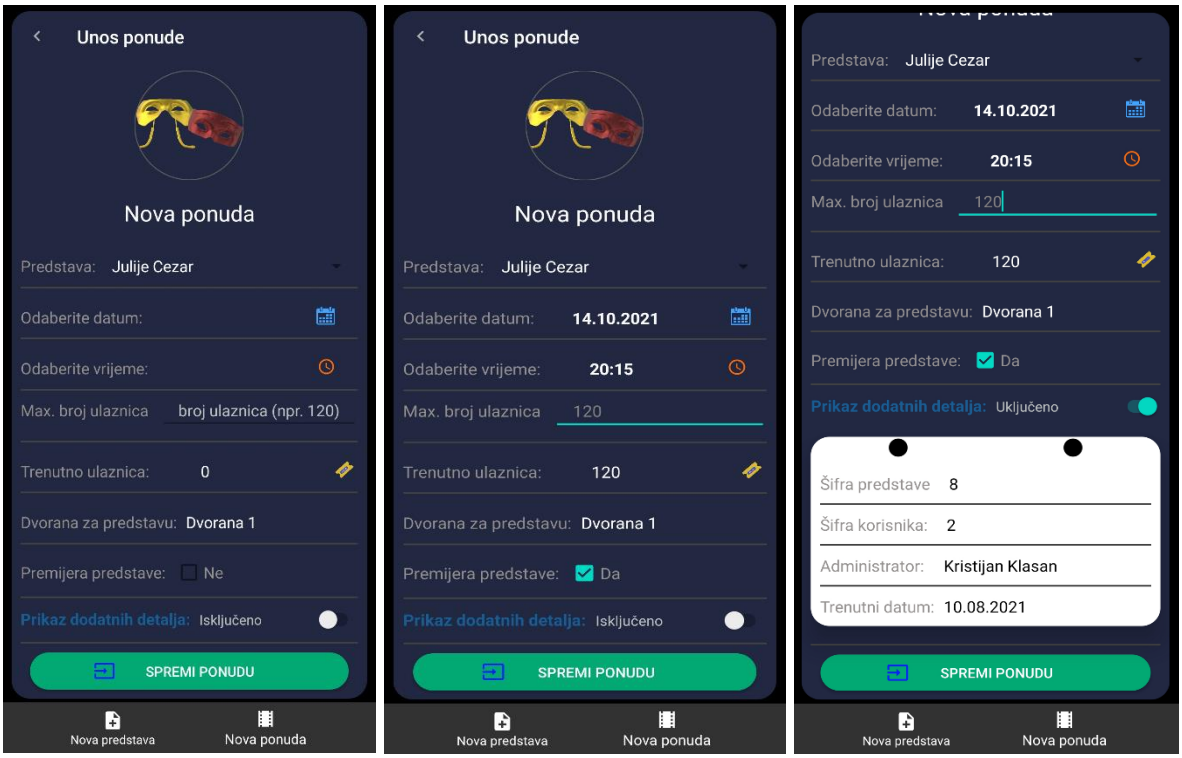
10. Unos predstava i ponuda

Administrator slijedno ispunjava polja s podacima u skladu s unaprijed definiranim planom i programom kazališta. Slike s predstave se odabiru iz galerije u koju se mogu dodatno unositi nove slike. Dolazak do galerije je moguć klikom na jedan od prozora slike. Unutar polja *glumci* je potrebno unijeti popis glumaca, svaki mora biti ispravno gramatički unesen i odvojen zarezom. Na taj način se postiže pravilan pregled podataka o glumcima jedan ispod drugog. Klikom na gumb „Spremi predstavu“ vrši se provjera ispunjenosti podataka, ako su svi podaci uneseni aplikacija sprema predstavu u bazu, dok u suprotnome traži unos određenog podatka.



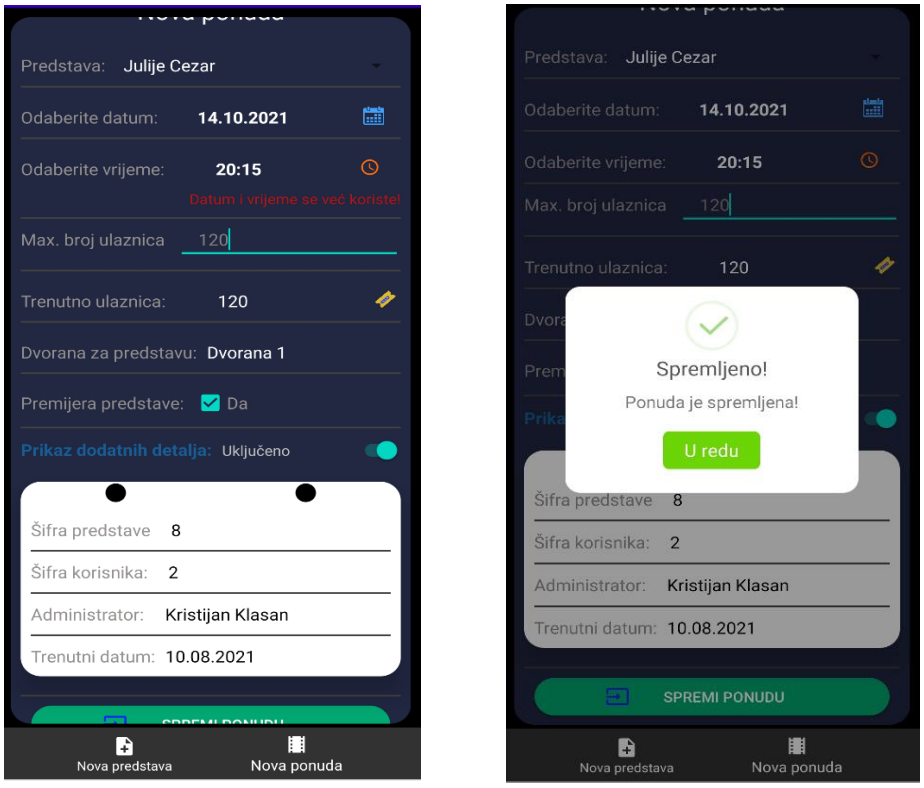
Slika 80. Prikaz zaslona za unos podataka o predstavi

Administrator u prvom koraku bira predstavu za koju želi stvoriti ponudu. Preduvjet je postojanje predstave u sustavu kako bi se za nju mogla izraditi ponuda. Potom slijedi odabir datuma i vremena izvođenja predstave. Maksimalni broj ulaznica podrazumijeva ukupan broj sjedala koja se mogu rezervirati od strane korisnika. Trenutni kapacitet dvorane obuhvaća 120 sjedećih mjesta. U većini slučajeva korisnik rezervira jednu ulaznicu. Podaci o dvorani i nazivu predstave govore unutar koje će se dvorane izvoditi predstava, te da li se predstava izvodi prvi put. Administrator nakon ispune podataka sprema ponudu klikom na gumb *„Spremi ponudu“.*



Slika 81. Prikaz procesa spremanja nove ponude predstave

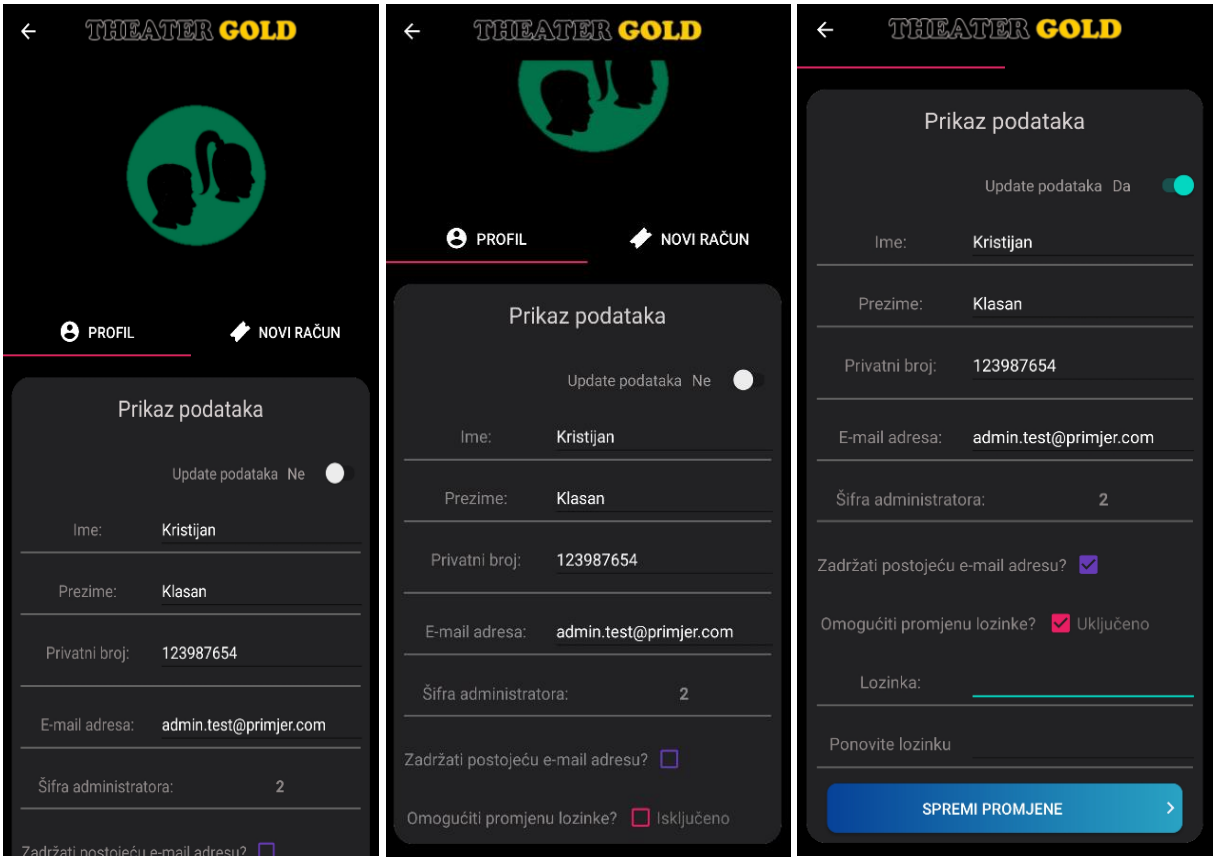
U slučaju slobodne dvorane u odabranom terminu, ponuda će biti spremljena u bazu, dok će u suprotnome javiti o potrebi promjene termina nakon ćega se ponovno provjerava dostupnost.



Slika 82. Poruka o zauzetosti dvorane i spremanja promjenom termina

11. Upravljanje profilom

Upravljanje profilom osigurava unos novog administratora, pregled i izmjenu osobnih podataka. Opcije su podijeljene u dvije odvojene cjeline, profil i novi račun. Samo ažuriranje podataka je podijeljeno na izmjenu podataka s lozinkom i bez nje. Postupak je sličan opciji ažuriranja iz korisničkog dijela aplikacije. Ažuriranje povlači podatke: ime, prezime, broj telefona, e-mail adresu i lozinku. Klikom na gumb „Spremi promjene“, pohranjuju se svi podaci u slučaju ispravnog unosa. U suprotnom, potrebno je izmijeniti podatke, te ponoviti proces spremanja podataka.

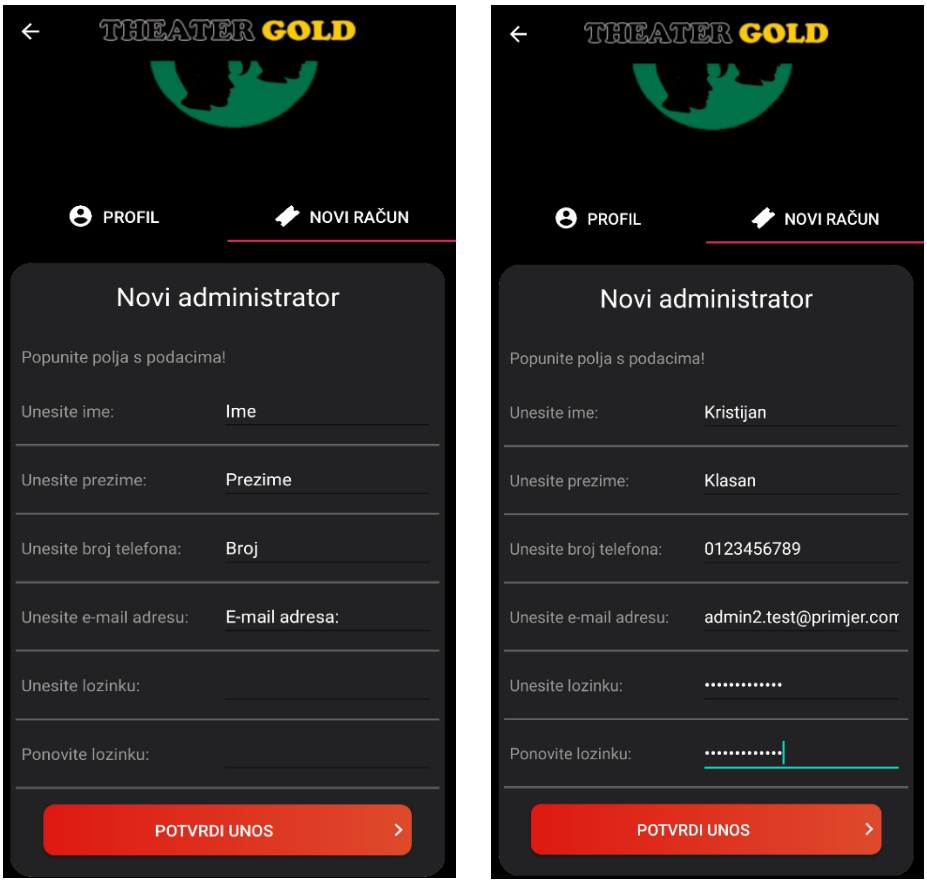


Slika 83. Prikaz zaslona upravljanja profilom

2. Klikom na opciju iz mini izbornika „Novi račun“ otvara se prozor za unos podataka o novom administratoru. Dodavanje obuhvaća unos osobnih podataka novog administratora. U prvom koraku postojeći administrator unosi podatke o budućem. Ispunom podataka, provjerava se dostupnost e-mail adrese budućeg administratora.

U slučaju da administrator ne popuni sve podatke, aplikacija ispisuje grešku te obaviještava o unosu traženih podataka. Posebno obratiti pozornost prilikom unosa

jednakih lozinki. Upisom različitih lozinki, neće biti spremljen novi administrator. U tom slučaju je potrebno ponovno unijeti lozinku i ponoviti postupak spremanja. Identičan scenarij se može evenutualno ponavljati dok se ne ispune uvjeti ispravnog unosa podataka.



Slika 84. Prikaz zaslona i unosa podataka administratora

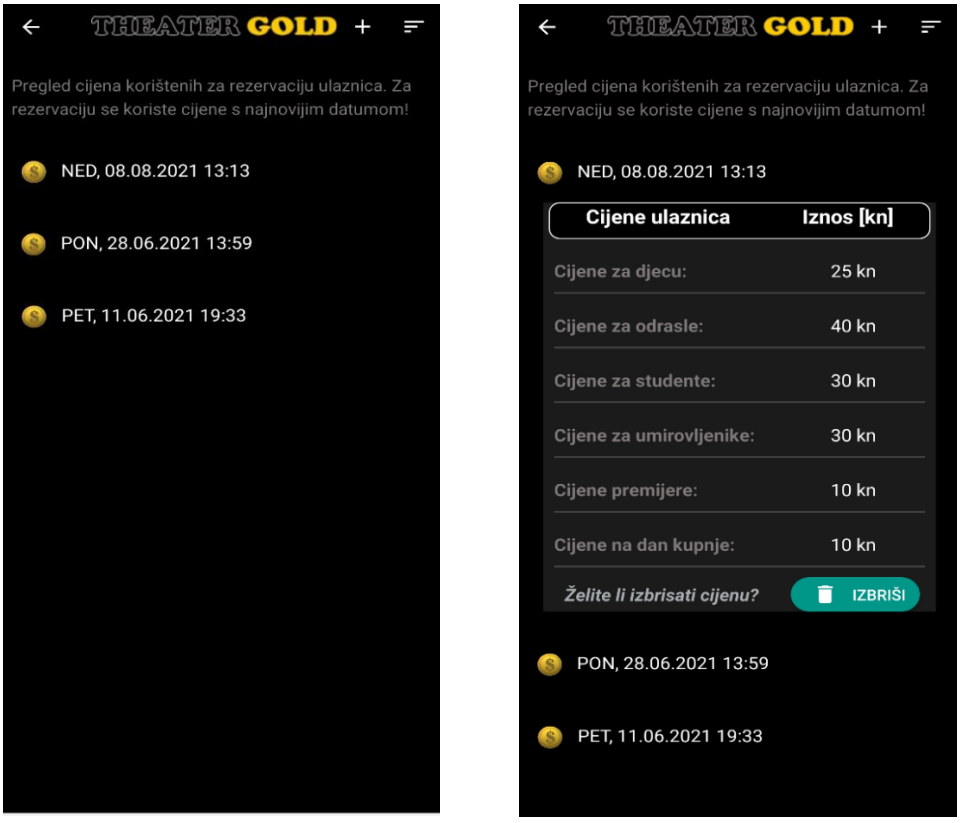
Klikom na gumb „Potvrdi unos“ i potvrdnim odgovorom u dijaloškom okviru, pokreće se proces provjere postojanja e-mail adrese i spremanja podataka. Uspješnim unosom dodjeljuju se sva prava na aplikaciju i njenu uporabu, te upravljanje podacima.

U teoriji, prije ispune obrasca unosa, potrebno je dobiti dozvolu kazališta za zapošljavanje novog administratora. Na taj način bi se zadržala kvaliteta i usluga aplikacije, te bi se spriječio potencijalni postupak konstantnog dodavanja novih administratora u sustav.

Napomena: za potrebe prikaza rada aplikcije su se koristili nepostojeći podaci koji nemaju veze sa stvarnim podacima.

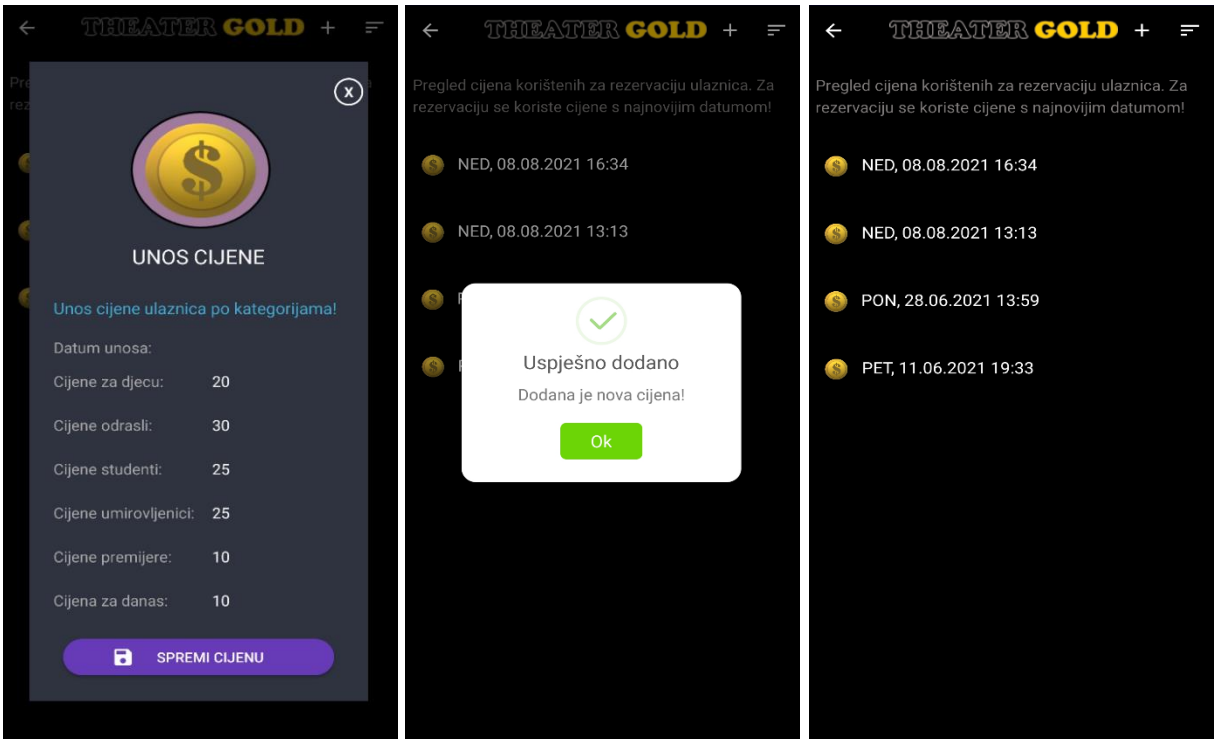
12. Upravljanje cijenama

Cijene su raspoređene prema dobnim kategorijama: studenti, djeca, odrasli i umirovljenici. Unutar svake cijene je uračunat PDV i naknade za dan rezervacije odnosno premijeru predstave. Cijena ulaznice na dan prikazivanja predstavlja cijenu koja se pridodaje standardnim cijenama. Primjer: korisnik želi kupiti studentsku ulaznicu za određenu predstavu. Cijena ulaznice iznosi cca. 30kn. Ukoliko student kupuje ulaznicu na dan predstave tada se osnovnoj cijeni pridodaje i iznos za cijenu na dan rezervacije. Studentska ulaznica je skuplja za 10 kn i ukupno iznosi 40 kn. Identičan postupak se primjenjuje i za naknadu premijerne predstave. Rezervacija koristi cijene s najnovijim datumom unosa. Administratoru je stavljeno na raspolaganje: unos nove cijene, brisanje postojećih cijena i sortiranje prema datumu.



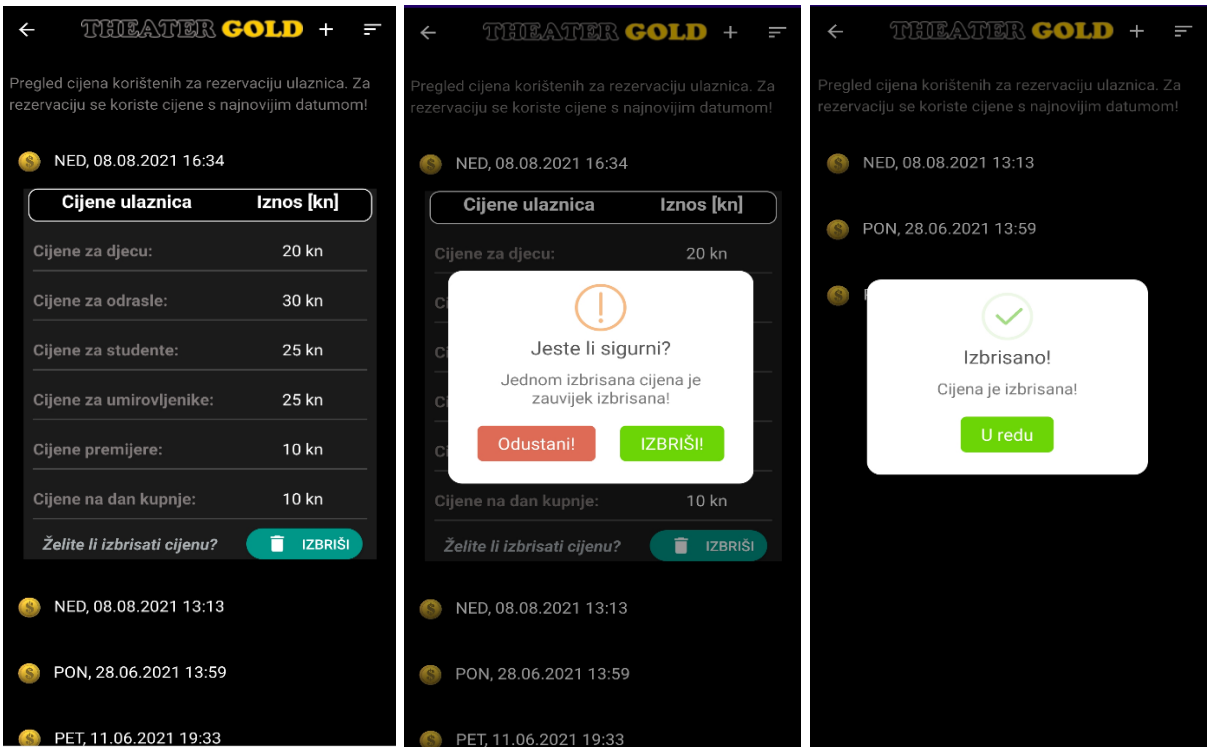
Slika 85. Popis svih cijena i pregled detalja odabrane cijene

1. Unos nove cijene se postiže administratorskim odabirom ikone dodaj (oznaka +) iz izbornika. Otvara se skočni prozor u koji se unose podaci o cijenama izraženim u kunama. Spremanje se postiže klikom na gumb „Spremi cijenu“. Upisana cijena je automatski aktivna jer sadrži najnoviji datum i vrijeme. U slučaju administratorske greške, praznog polja, program upozorava o unosu podataka radi njegova spremanja.



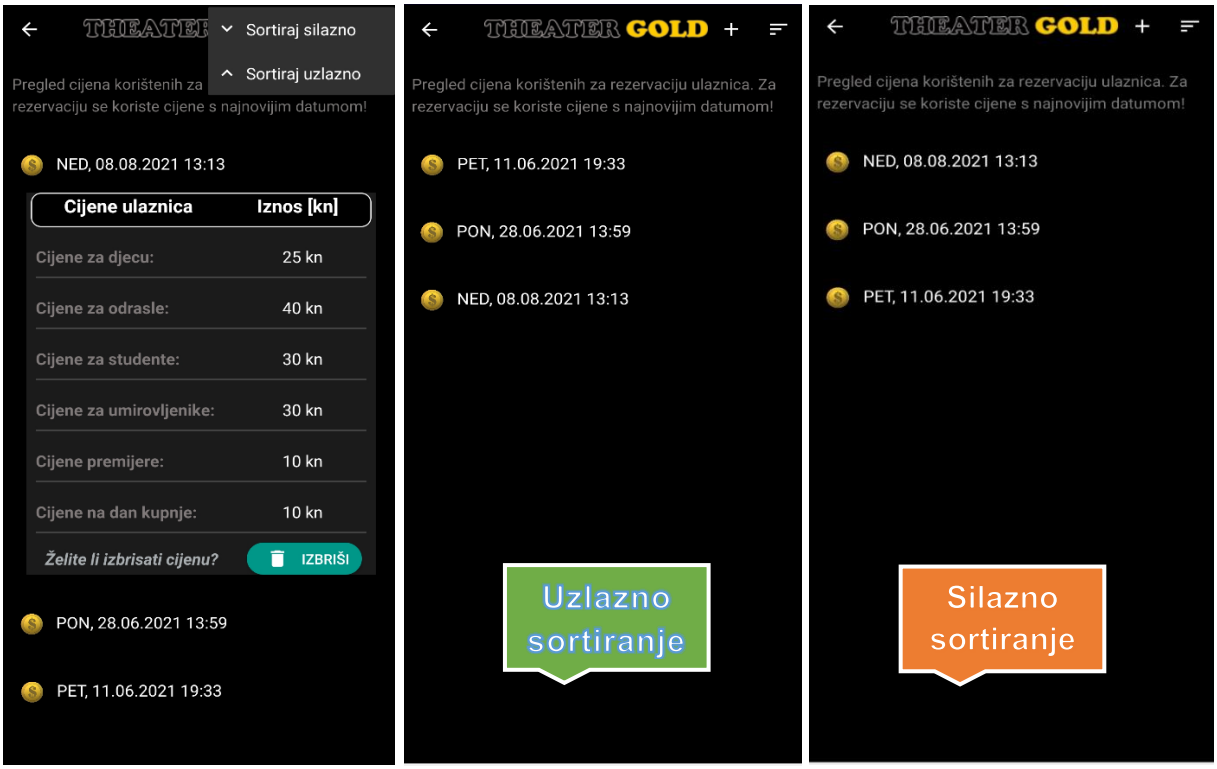
Slika 86. Prikaz procesa unosa cijene u aplikaciju

2. Brisanje cijene se postiže odabirom jedne cijene za koju se prvo prikazuju detalji. Klikom na datum, proširuje se okvir u kojem se između ostalih podataka nalazi gumb za brisanje trenutne cijene. Klikom na gumb „Izbriši“ otvara se dijaloški okvir u kojem se provjerava želi li se izbrisati cijena. Potvrdnim odgovorom briše se trenutni zapis.



Slika 87. Prikaz procesa brisanja cijene

3. Sortiranje cijena je moguće izvršiti na dva načina: silazno (eng. Descending) i uzlazno (eng. Ascending). Silazno sortiranje omogućuje prikaz podataka o cijenama od najnovijeg aktualnog datuma prema najstarijem. Uzlazno sortiranje obuhvaća suprotni prikaz podataka.

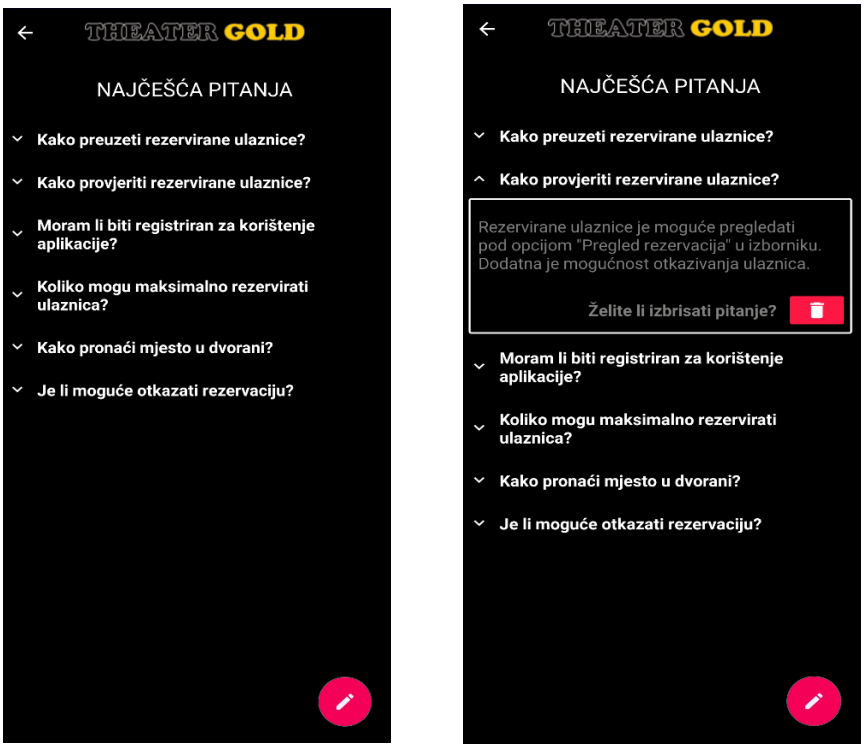


Slika 88. Opcija sortiranja, te prikaz silaznog i uzlaznog sortiranja cijena

13. Pregled pitanja

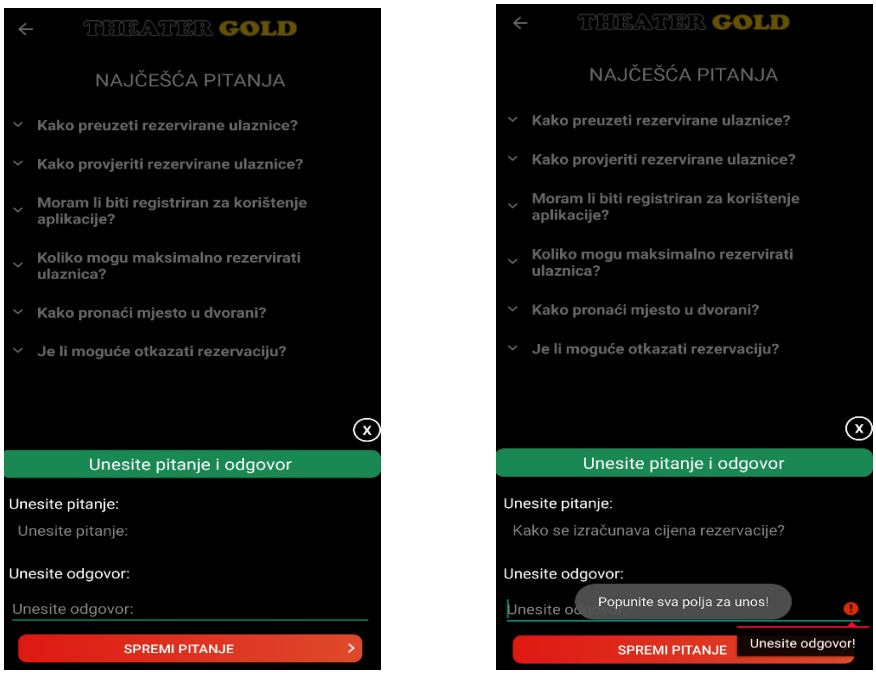
Aplikacija izlistava popis najčešćih pitanja koje su administratori prethodno unijeli u sustav. Pitanja i njihovi odgovori nastoje pomoći korisnicima u lakšem snalaženju i razumijevanju načina rada aplikacije. Strelica u izborniku služi za povratak na početni zaslon. Svaki administrator unutar ove opcije može pregledavati, unositi i brisati pitanja. Također, omogućeno je po želji otvaranje i zatvaranje pitanja. Otvaranjem pitanja se prikazuje naziv pitanja, te odgovor na postavljeno pitanje. U donjem dijelu otvorenog pitanja postoji gumb za brisanje.

U slučaju brisanja, klikom na gumb se otvara novi dijaloški okvir unutar kojega se provjerava želi li administrator uistinu izbrisati pitanje. Potvrdnim odgovorom na postavljeni upit, aplikacija briše pitanje i informira administratora o uspješnosti izvršene operacije. Jednom izbrisano pitanje nije moguće vratiti.



Slika 89. Prikaz popisa pitanja i detalja odabranog pitanja

Unos pitanja se realizira gumbom koji se nalazi pri dnu zaslona. Klikom na njega se otvara skočni prozor koji sadrži polja za unos pitanja i odgovora, te gumba za spremanje. Ako administrator zaboravi unijeti određeni podatak, aplikacija informira o potrebi popunjenosti polja. Spremanje se izvršava klikom na gumb „Spremi pitanje“. Aplikacija nakon spremanja prazni polja radi unosa novog pitanja. Prozor se zatvara klikom na gumb X.



Slika 90. Prozor za unos pitanja i pokušaj unosa bez podataka

8. Zaključak

Izrađena mobilna aplikacija rješava osnovne probleme s kojima se kazališta svakodnevno susreću. Glavni segment korištenja aplikacije je povezanost s pozadinskim sustavom koji komunicira s bazom podataka. Prilikom testiranja potrebno je osigurati više fizičkih uređaja nad kojima će se provjeravati ispravnost aplikacije. Dobra praksa je korištenje fizičkih uređaja radi pružanja stvarnog izgleda aplikacije. Prema dobivenoj statistici i grafičkim prikazima zaključuje se kako Android operacijski sustav ima više registriranih korisnika i samim time posjeduje veće tržište aplikacijama nego iOS. Tabličnim prikazom su evidentirane glavne stavke proučavanih operacijskih sustava. Postojeće aplikacije daju uvid u osnovne funkcionalnosti i princip rada, ali nude mogućnost korištenja samo krajnjim korisnicima koji žele kupiti ulaznicu, dok su administratori uskraćeni za njihovu uporabu.

Temeljem dijagrama budući korisnici mogu shvatiti način rada aplikacije, te vidjeti povezanost među tablicama. Povezanost je realizirana korištenjem ključeva. Uporaba relacijske MySQL baze podataka je osiguralo idealno rješenje za spremanje podataka unutar različitih tablica. Napredno poznavanje SQL-a je bilo od krucijalnog značaja jer su svi upiti nad bazom podataka realizirani s njima. Nad bazom podataka su primjenjene sve CRUD operacije. Izgradnja pozadinskog sustava iziskuje korištenje Node.js okvira radi postizanja suvremenijeg načina razvoja i kvalitete. Za online uporabu aplikacije, potrebno je *backend* postaviti na neki od besplatnih servisa, dok je za lokalno potrebno paziti da se svi uređaji nalaze na istoj IP adrese mreže. Mobilna aplikacija radi isključivo na Android operacijskom sustavu. Buduće poboljšanje se može ogledati u razvoju iste za iOS. Aplikacijom s korisničke strane je realizirano učinkovito pregledavanje i pretraživanje predstava, te pregledom i rezervacijom ulaznica definiranih termina izvođenja predstava. Administrator je u posjedu funkcionalnosti aplikacije i može rukovoditi cjelokupnim sustavom. Glavne funkcionalnosti se dotiču upravljanja: rezervacijama, unosom i ažuriranjem podataka o predstavi odnosno njihovim ponudama. Ovakva vrsta aplikacije pruža odgovor i rješenja za navedene probleme i nedostatke.Troškovi održavanja bi mogli predstavljati jedini nedostatak jer zahtjeva neprestano ulaganje. Na kraju se može zaključiti kako bi uvođenje ovakve aplikacije u čim više kulturnih ustanova rezultiralo pozitivnijem radu kazališta i jednostavnijem pristupu ulaznicama. Budući razvoj aplikacije ostavlja dovoljan prostor za poboljšanje i nadogradnju.

9. Literatura

[1] Phillips B., Stewart C., Marsicano K., *Android Programming: the big nerd ranch guide (3rd edition),* Big Nerd Ranch, 2017

[2] BITWARE (veljača 16, 2020), Kerempuh (2.0.5), Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.bitware.kerempuh&hl=en_US&gl=US, pristupano: 17.7.2021

[3] BITWARE (rujan 27, 2020), Zagrebačko Kazalište Mladih (1.0.21), Google Play, dostupno:https://play.google.com/store/apps/details?id=com.bitware.zkm&hl=en_US&gl=US, pristupano: 17.7.2021

[4] IfM University of Cambridge, *SWOT (Strengths, Weakness, Opportunities, Threats)*, dostupno na: https://www.ifm.eng.cam.ac.uk/research/dstools/swot/, pristupano: 18.7.2021

[5] MacQueen S., *What is a Mobile App Prototype?*, Software Design & UX, studeni 2018, dostupno na: https://discoverbigfish.com/blog/what-is-a-mobile-app-prototype.html , pristupano: 24.6.2021

[6] Statistički ljetopis grada Zagreba 2019, Poglavlje 21: Kultura i umjetnost, dostupno:http://www1.zagreb.hr/zgstat/documents/Ljetopis_2019/2019_21_Kultura_Umjetnost.pdf , pristupano 5.7.2021

[7] TechTarget, *Relational database*, Home/DBMS/Data center management, dostupno:https://searchdatamanagement.techtarget.com/definition/relational-database , pristupano 6.7.2021

[8] Codecademy, *What is a Relational Database Management System ?*, dostupno: https://www.codecademy.com/articles/what-is-rdbms-sql , pristupano: 8.7.2021

[9] Gallop H., *Creating and Using Resource Files in Android Studio*, Medium, veljača 2018, dostupno na: https://medium.com/@doyouseeitmyway/creating-and-using-resource-files-in-android-studio-8b4208c08862 , pristupano: 9.7.2021

[10] Vogel L., *Building dynamic user interfaces in Android with fragments*, 2016, dostupno na: https://www.vogella.com/tutorials/AndroidFragments/article.html, pristupano: 11.7.2021

[11] JavaTpoint, *Android Intent Tutorial*, dostupno na Internet stranici: https://www.javatpoint.com/android-intent-tutorial, pristupano: 9.7.2021

[12] Tutorialspoint simply easy learning, *Android - Services*, dostupno na: https://www.tutorialspoint.com/android/android_services.htm, pristupano dana: 11.7.2021

[13] TechTarget, *Android Studio*, Home/Applications and Infrastructure/Developer, dostupno na: https://searchmobilecomputing.techtarget.com/definition/Android-Studio, pristupano 11.7.2021

[14] Cervantes E.D., *What is Android?,* lipanj 2021, Android Authority, dostupno na: https://www.androidauthority.com/what-is-android-328076/,pristupano: 11.7.2021

[15] JavaTpoint, *Android Versions*, dostupno na: https://www.javatpoint.com/android-versions, pristupano 12.7.2021

[16] Wikipedia, *Android (operating system)*, srpanj 2021, dostupno na: https://en.wikipedia.org/wiki/Android_(operating_system), pristupano: 13.7.2021

[17] Programiz: Java Programming, Tutorials/Java Resources, dostupno na: https://www.programiz.com/java-programming/guide, pristupano: 15.7.2021

[18] Kenton W., *Apple iOS*, Investopedia, srpanj 2020, dostupno na: https://www.investopedia.com/terms/a/apple-ios.asp, pristupano: 15.7.2021

[19] Statcounter GlobalStats, *Mobile OS Market Share Worldwide*, dostupno na: https://gs.statcounter.com/os-market-share/mobile/, pristupano: 15.7.2021

[20] App Store, *Pronađi omiljene aplikacije. I one koje će to tek postati*, dostupno na: https://www.apple.com/hr/ios/app-store/, pristupano: 16.7.2021

[21] Wei-Meng L., *Android 4: Razvoj aplikacija*, Wiley Publishing Inc., Beograd, 2012

[22] Node.js, *Introduction to Node.js*, dostupno na Internet stranici: https://nodejs.dev/learn/introduction-to-nodejs, pristupano 30.7.2021

[23] Sharma A., *Full-Stack Web Development with Vue.js and Node*, Packt Publishing, May 2018

[24] Advanced REST Client (10.0.12, 14. veljače 2019.), Chrome web-trgovina, English, dostupno na: https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo?hl=hr, pristupano 30.7.2021

[25] Lucidchart, dostupno: https://www.lucidchart.com/pages/, pristupano: 10.6.2021

[26] Android Studio, dostupno na: https://developer.android.com/studio, 10.3.2021

[27] Mishura A., Backlin G., *Razvoj aplikacija za iPhone i iPad,* Dobar Plan, 2012

[28] Our Code World, How to use Sweet Alert Dialogs in Android, dostupno na: https://ourcodeworld.com/articles/read/928/how-to-use-sweet-alert-dialogs-in-android, pristupano: 20.4.2021

[29] Microsoft Corporation (veljača 17, 2016), Bojanje 3D, Microsoft trgovina, dostupno na Internet stranici: https://www.microsoft.com/hr-hr/p/bojanje-3d/9nblggh5fv99#activetab=pivot:overviewtab, pristupano: 20.5.2021

[30] TodayTix (srpanj 27,2021), TodayTix – Theater Tickets (2.9.6), Google Play, dostupno:https://play.google.com/store/apps/details?id=com.todaytix.TodayTix&hl=hr&gl=US, pristupano 15.4.2021

[31] Karaga M., Stojanović M., *Programiranje aplikacija za Android*, 0. izdanje, Element, Zagreb, 2018

[32] Gradsko kazalište Zorin dom, *Predstava: Julije Cezar*, dostupno na: http://www.zorin-dom.hr/predstave?item=2001, pristupano: 10.8.2021

[33] Gradsko kazalište Zorin dom, *Predstava: Stanko i Oliver*, dostupno na: http://www.zorin-dom.hr/predstave?item=1995 , pristupano: 10.8.2021

[34] Istarsko narodno kazalište, dostupno: https://www.ink.hr, pristupano: 12.8.2021

10. Popis slika

Slika 1. Prikaz dosadašnjih verzija Android sustava 2
Slika 2. Prikaz Android arhitekture 3
Slika 3. Sučelje razvojnog okruženja Android Studio 5
Slika 4. Prikaz strukture mape java 6
Slika 5. Prikaz strukture mape res 6
Slika 6.Tržište mobilnih OS – svijet 9
Slika 7. Tržište mobilnih OS – Hrvatska 9
Slika 8. Prikaz trgovine App Store 10
Slika 9. Prikaz SWOT analize 14
Slika 10. Popis i prikaz detalja predstava 16
Slika 11. Izbornik i sadržaj predstave 17
Slika 12. Početni zaslon i pojedinosti predstave 18
Slika 13. Dijagram slučajeva korištenja – Korisnik 19
Slika 14. Dijagram slučajeva korištenja – Administrator 21
Slika 15. Sekvencijski dijagram rezervacije ulaznica 23
Slika 16. Sekvencijski dijagram pregleda rezervacija i favorita 24
Slika 17. Sekvencijski dijagram upravljanja rezervacijama 25
Slika 18. Sekvencijski dijagram obrade cijena 26
Slika 19. Sekvencijski dijagram upravljanja računom 27
Slika 20. Sekvencijski dijagram obrade najčešćih pitanja 28
Slika 21. Sekvencijski dijagram unosa predstave i ponude 29
Slika 22. Prikaz klasnog dijagrama 30
Slika 23. Prototip prijave korisnika 32
Slika 24. Prototip detalja predstave 32
Slika 25. Prototip odabir podataka 33
Slika 26. Prototip rezervacija ulaznica 33
Slika 27. Prototip prikaza rezervacija 34
Slika 28. Prototip upravljanja računom 34
Slika 29. Prototip unosa ponude 34
Slika 30. Prototip prikaza ponuda 34
Slika 31. Prototip prikaza detalja rezervacije 35
Slika 32. Prototip unosa cijene 35

Slika 33. Prikaz tablice „Administrator“ 37
Slika 34. Prikaz tablice „Korisnik“ 37
Slika 35. Prikaz tablice „Predstava“ 38
Slika 36. Prikaz tablice „Raspored_predstava“ 38
Slika 37. Prikaz tablice „Moj izbor“ 39
Slika 38. Prikaz tablice „Cijene“ 39
Slika 39. Prikaz tablice „Rezervacija“ 39
Slika 40. Prikaz tablice „Karta“ 40
Slika 41. Prikaz tablice „Najcesca_pitanja“ 40
Slika 42. Prikaz tablice „Slika“ 40
Slika 43. Prikaz popisa tablica iz baze podataka 41
Slika 44. Struktura tablice predstava u bazi podataka 41
Slika 45. Primjer SQL upita dohvaćanja rasporeda prema nazivu predstava 41
Slika 46. Primjer korištenja alata „Advanced REST client“ 43
Slika 47. Prikaz dobivenog odgovora u JSON formatu 47
Slika 48. Prikaz navigacije za prebacivanje fragmenata 59
Slika 49. Usporedba prikaza aktivnosti odabrane funkcionalnosti 61
Slika 50. Početni zaslon aplikacije 69
Slika 51. Nema internetske veze 69
Slika 52. Prikaz zaslona za prijavu korisnika i administratora 70
Slika 53. Prikaz zaslona registracije, početnog zaslona i izbornika aplikacije 70
Slika 54. Pretraživanje po kategorijama i naslovu 71
Slika 55. Pregled detalja i rasporeda predstave 72
Slika 56. Dodavanje i uklanjanje predstave iz favorita 72
Slika 57. Prikaz postupka rezerviranja ulaznice 73
Slika 58. Vertikalni prikaz odabira sjedala 74
Slika 59. Horizontalni prikaz zaslona odabira sjedala u dvorani 74
Slika 60. Pregled detalja i završetak rezervacije 75
Slika 61. Prikaz rezerviranih ulaznica i isključivanja opcije filtriranja 76
Slika 62. Filtriranje rezervacija prema nazivu te prikaz detalja 77
Slika 63. Prikaz favorita i filtriranje favorita prema datumu 77
Slika 64. Prikaz podataka o korisniku i uključivanje opcije ažuriranja 78
Slika 65. Prikaz ažuriranja podataka i spremanja promjena 79
Slika 66. Prikaz promjene lozinke 79

Slika 67. Prikaz pogreške prilikom unosa 79
Slika 68. Prikaz svih pitanja i detalja odabranog pitanja 80
Slika 69. Pregled cijena – korisnik 81
Slika 70. Pregled informacija o aplikaciji 81
Slika 71. Početni zaslon administratora 81
Slika 72. Prikaz administratorskog izbornika 81
Slika 73. Prikaz procesa pretraživanja, odabira, te ažuriranja podataka 82
Slika 74. Prikaz svih ponuda predstava i filtrirani prikaz 83
Slika 75. Pregled podataka i ažuriranje ponude predstave 84
Slika 76, Prikaz rezervacija i dodatnih opcija upravljanja 85
Slika 77. Filtriranje i prikaz detalja rezervacije 85
Slika 78. Prikaz stupčastog grafikona 86
Slika 79. Prikaz tortnog grafikona 86
Slika 80. Prikaz zaslona za unos podataka o predstavi 87
Slika 81. Prikaz procesa spremanja nove ponude predstave 88
Slika 82. Poruka o zauzetosti dvorane i spremanja promjenom termina 88
Slika 83. Prikaz zaslona upravljanja profilom 89
Slika 84. Prikaz zaslona i unosa podataka administratora 90
Slika 85. Popis svih cijena i pregled detalja odabrane cijene 91
Slika 86. Prikaz procesa unosa cijene u aplikaciju 92
Slika 87. Prikaz procesa brisanja cijene 92
Slika 88. Opcija sortiranja, te prikaz silaznog i uzlaznog sortiranja cijena 93
Slika 89. Prikaz popisa pitanja i detalja odabranog pitanja 94
Slika 90. Prozor za unos pitanja i pokušaj unosa bez podataka 94

11. Popis programskog koda

Programski kod 1. Prikaz dinamičke zamjene fragmenata 8
Programski kod 2. Pokretanje eksplicitne namjere nove aktivnosti 8
Programski kod 3. Prikaz dopuštenja unutar datoteke manifest 44
Programski kod 4. Prikaz referenci na Retrofit i GSON biblioteke 44
Programski kod 5. Primjer kreiranja instance Retrofit objekta GSON biblioteke 45
Programski kod 6. Provjera ispunjenosti polja e-mail adrese i lozinke 45
Programski kod 7. Implementacija poziva web servisa 46
Programski kod 8. Metoda za poziv web servisa 47
Programski kod 9. Definiranje ruta za prijavu korisnika koristeći Node.js 47
Programski kod 10. Primjer modela klase „UserModel“ 48
Programski kod 11. Prikaz registracije korisnika koristeći Retrofit 49
Programski kod 12.Slanje podataka na server pomoću rute 49
Programski kod 13. Pomoćna funkcija koja provjerava postojanje korisnika 50
Programski kod 14. Definiranje rute registracije korisnika koristeći Node.js 50
Programski kod 15. Primjer pokretanja nove aktivnosti koristeći namjere 51
Programski kod 16. Prikaz isječka XML koda za pregled kategorija predstava 52
Programski kod 17. Dodavanje predstave u favorite 53
Programski kod 18. Programski kod dijeljenja sadržaja predstave 54
Programski kod 19. Prikaz punjenja Spinnera i odabira stavke predstave 55
Programski kod 20. Postavljanje slušača radi mogućnosti zakretanja zaslona 56
Programski kod 21. Provjera zauzetosti sjedala u dvorani 57
Programski kod 22. Prikaz XML koda navigacije 58
Programski kod 23. Prikaz dodavanja izbornika u donji navigacijski prikaz 58
Programski kod 24. Prikaz odabira fragmenata koristeći navigaciju 59
Programski kod 25. Prosljeđivanje podataka sljedećem fragmentu 62
Programski kod 26. Prikaz ažuriranja podataka 63
Programski kod 27. Promjene grafikona i slanje identifikacijskog ključa 65
Programski kod 28. Prikaz izrade tortnog grafikona 65
Programski kod 29. Prikaz ažuriranja administratorskih podataka 66

12. Popis tablica

Tablica 1. Tablični prikaz razlika operacijskih sustava Android i iOS 11
Tablica 2. Prikaz podataka kazališta u razdoblju od 2010. do 2018. godine 12
Tablica 3. Pregled i opis funkcionalnosti korisnika u aplikaciji 20
Tablica 4. Pregled funkcionalnosti administratora u aplikaciji 22
Tablica 5. Pregled popisa tablica u bazi podataka 37
Tablica 6. Tablični prikaz osnovnih korištenih HTTP metoda 42

SAŽETAK

Korisnici se svakodnevno suočavaju s problemima rasprodanih ulaznica, dugim redovima tijekom kupovine, malim brojem mjesta za parkiranje vozila, kašnjenjima u ažuriranju sadržaja predstava i ponuda. Navedeni problemi rješavaju se izradom mobilne aplikacije za korisnika i administratora. Korisnički dio omogućuje: rezervaciju kazališnih ulaznica, pregled i upravljanje rezerviranih ulaznica, upravljanje korisničkim računom, te upute za korištenje. Administratorski dio obuhvaća unos kazališnih predstava, upravljanje ponudama i rezervacijama, održavanje računa, te unos pitanja i cijena. Rad se zasniva na teorijskom i praktičnom dijelu. U teorijskom dijelu se opisuje načelo rada Android operacijskog sustava i osnovne razlike u odnosu na operacijski sustav iOS. Praktični dio obuhvaća opis korištenih tehnologija i razvijenog sustava koji sadrži: dijagrame, prototip sučelja, te implementaciju programskih rješenja.

Ključne riječi: mobilna aplikacija, Android, Java, Node.js, kazalište, relacijska baza podataka, rezervacija ulaznica, informatika

SUMMARY

Users face daily problems with sold-out tickets, long queues during purchases, small number of parking spaces, delays in updating the content of performances and offers. These problems are solved by creating a mobile application for users and administrators. The user part enables: reservation of theater tickets, review and management of reserved tickets, management of the user account, and instructions for use. The administrative part includes entering theater performances, managing offers and reservations, maintaining accounts, entering questions and prices. The paper is based on the theoretical and practical part. The theoretical part describes the principle of operation of the Android operating system and the basic differences in relation to the operating system iOS. The practical part includes a description of the technologies used and developed system, which contains: diagrams, prototype interfaces, and the implementation of software solutions.

Keywords: mobile application, Android, Java, Node.js, theater, relational database, ticket reservation, informatics