Programiranje mobilnih uređaja

Implementacija sustava za online rezervaciju hotela

Martin Grgić 3.G

Elektrotehnička i ekonomska škola

Nova Gradiška

Sadržaj

[**Uvod** 3](#_Toc161774780)

[**Dizajn glavnog sučelja** 4](#_Toc161774781)

[Programski kod glavnog sučelja 5](#_Toc161774782)

[**Programski kod „soba\_item“** 11](#_Toc161774783)

[**Dizajn sučelja „autentikacija“** 16](#_Toc161774784)

[Programski kod „autentikacija“ 17](#_Toc161774785)

[**MainActivity** 21](#_Toc161774786)

[**Data class „Soba“** 24](#_Toc161774787)

[**Class „TextAdapter“** 25](#_Toc161774788)

[**Class „LoginActivity“** 27](#_Toc161774789)

# **Uvod**

U Android Studiu potrebno je bilo implementirati sustav za online rezervaciju hotela koji sadrži klase: Soba, Gost u Sobi i Rezervacija upotrebom klasa, firebase-a i RecyclerView-a. Uz to je potrebno napraviti autentikaciju korisnika.

# **Dizajn glavnog sučelja**

Slika na kojoj se prikazuje tekst, snimka zaslona, softver

Opis je automatski generiran

***Dizajn glavnog sučelja***

Dizajn se sastoji od 4 editTexta, buttona i RecyclerView-a

U 4 editTexta unosimo podatke: broj sobe, tip sobe, cijenu noci i velicinu sobe

Pomoću buttona unosimo podatke u RecyclerView

Preko RecyclerView-a gledamo unesene podatke

## Programski kod glavnog sučelja

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod button-a „Unesi“***

**id –** označava id našeg gumba, u ovom slučaju to je „Unesi“

**layout\_width –** određuje širinu našeg gumba; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u gumbu

**layout\_height -** određuje visinu našeg gumba; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u gumbu

**layout\_constraintBottom\_toTopOf –** određuje nam s čime je poravnat donji dio našeg gumba;

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg gumba;

**layout\_constraintTop\_toBottomOf -** određuje nam s čime je poravnat gornji dio našeg gumba;

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod editTexta „broj sobe“***

**id –** označava id našeg editText-a, u ovom slučaju to je „BrojSobe“

**layout\_width –** određuje širinu našeg editText-a; postavljen je na 0dp što znači da će editText zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg editText-a; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u editText

**hint –** iz resursa string.xml uzima tekst koji smo dodijelili stringu brzina, u našem slučaju to je tekst „Unesi broj sobe“ i on će biti ispisan kada ništa nije upisano u EditText-u

**textSize –** veličina teksta

**layout\_constraintEnd\_toStartOf –** određuje nam s čime je poravnat kraj (desni dio) našeg editText; u ovom slučaju s početkom (lijevim dijelom) gumba „commit“

**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg editTexta; u ovom slučaju s lijevom stranom recycler-a „recycler“

**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg editText-a; u našem slučaju s gornjim dijelom ekrana „parent“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod editText „TipSobe“***

**id –** označava id našeg editText-a, u ovom slučaju to je „TipSobe“

**layout\_width –** određuje širinu našeg editText-a; postavljen je na 0dp što znači da će editText zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg editText-a; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u editText

**hint –** iz resursa string.xml uzima tekst koji smo dodijelili stringu brzina, u našem slučaju to je tekst „Unesi tip sobe“ i on će biti ispisan kada ništa nije upisano u EditText-u

**textSize –** veličina teksta

**layout\_constraintBottom\_toTopOf -** određuje nam s čime je poravnat donji dio našeg editText-a;

**layout\_constraintEnd\_toStartOf –** određuje nam s čime je poravnat kraj (desni dio) našeg editText;

**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg editTexta;

**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg editText-a;

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod editText „CijenaNoci“***

**id –** označava id našeg editText-a, u ovom slučaju to je „CijenaNoci“

**layout\_width –** određuje širinu našeg editText-a; postavljen je na 0dp što znači da će editText zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg editText-a; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u editText

**hint –** iz resursa string.xml uzima tekst koji smo dodijelili stringu brzina, u našem slučaju to je tekst „Unesi cijenu nocenja“ i on će biti ispisan kada ništa nije upisano u EditText-u

**textSize –** veličina teksta

**layout\_marginLeft ­–** početna margina (lijeva)

**layout\_constraintEnd\_toStartOf –** određuje nam s čime je poravnat kraj (desni dio) našeg editText;

**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg editTexta;

**layout\_constraintTop\_toBottomOf -** određuje nam s čime je poravnat gornji dio našeg editText-a;

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod editText „VelicinaSobe“***

**id –** označava id našeg editText-a, u ovom slučaju to je „velicinaSobe“

**layout\_width –** određuje širinu našeg editText-a; postavljen je na 0dp što znači da će editText zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg editText-a; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u editText

**hint –** iz resursa string.xml uzima tekst koji smo dodijelili stringu brzina, u našem slučaju to je tekst „Unesi velicinu sobe“ i on će biti ispisan kada ništa nije upisano u EditText-u

**textSize –** veličina teksta

**layout\_marginLeft ­–** početna margina (lijeva)

**layout\_constraintEnd\_toStartOf –** određuje nam s čime je poravnat kraj (desni dio) našeg editText;

**layout\_constraintTop\_toBottomOf -** određuje nam s čime je poravnat gornji dio našeg editText-a;

**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg editTexta;

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod recyclerView-a „RV“***

**id –** označava id našeg recycler-a, u ovom slučaju to je „recycler“

**layout\_width –** određuje širinu našeg recycler-a; postavljen je na 0dp što znači da će gumb zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg recycler-a; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u recycler

**layout\_margin .** margina

**layout\_constraintBottom\_toBottomOf -** određuje nam s čime je poravnat donji dio našeg recycler-a; u našem slučaju s donjim dijelom ekrana

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg recycler-a; u ovom slučaju s desnim dijelom ekrana

**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg recycler-a; u ovom slučaju s lijevom stranom ekrana

**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg recycler-a;

# **Programski kod „soba\_item“**

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod textView-a „id“***

**id –** označava id našeg editText-a, u ovom slučaju to je „id“

**layout\_width –** određuje širinu našeg textView; postavljen je na wrap\_content što znači da će textView zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u textView-a

**layout\_height -** određuje visinu našeg textView -a; postavljen je na match\_parent što znači da će textView zauzeti cijelu visinu do parenta

**textSize –** veličina teksta

**layout\_margin ­–** sve 4 margine

**layout\_constraintBottom\_toTopOf -** određuje nam s čime je poravnat donji dio našeg textView -a;

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg textView;

**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg textView-a;

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, softver

Opis je automatski generiran

***Programski kod TextView-a „BrojSobe“***

**id –** označava id našeg TextView -a, u ovom slučaju to je „BrojSobe“

**layout\_width –** određuje širinu našeg TextView -a; postavljen je na 0dp što znači da će TextView zauzeti onoliko mjesta koliko ima znakova

određuje visinu našeg textView -a; postavljen je na match\_parent što znači da će textView zauzeti cijelu visinu do parenta

**layout\_margin –** veličina sve 4 margine

**text –** tekst

**textSize –** veličina teksta

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg TextView; u ovom slučaju s desnom stranom ekrana

**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg TextView-a; u ovom slučaju s id-em

**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg TextView -a; u našem slučaju s gornjim dijelom ekrana „parent“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod TextView-a „TipSobe“***

**id –** označava id našeg TextView -a, u ovom slučaju to je „TipSobe“

**layout\_width –** određuje širinu našeg TextView -a; postavljen je na 0dp što znači da će TextView zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg textView -a; postavljen je na match\_parent što znači da će textView zauzeti cijelu visinu do parenta

**layout\_margin –** veličina sve 4 margine

**text –** tekst

**textSize –** veličina teksta

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg TextView;

**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg TextView-a;

**layout\_constraintTop\_toBottomOf -** određuje nam s čime je poravnat gornji dio našeg TextView -a;

***Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran***

***Programski kod TextView-a „CijenaNoci“***

**id –** označava id našeg TextView -a, u ovom slučaju to je „CijenaNoci“

**layout\_width –** određuje širinu našeg TextView -a; postavljen je na 0dp što znači da će TextView zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg textView -a; postavljen je na match\_parent što znači da će textView zauzeti cijelu visinu do parenta

**layout\_margin –** veličina sve 4 margine

**text –** tekst

**textSize –** veličina teksta

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg TextView;

**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg TextView-a;

**layout\_constraintTop\_toBottomOf -** određuje nam s čime je poravnat gornji dio našeg TextView -a;

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod TextView-a „VelicinaSobe“***

**id –** označava id našeg TextView -a, u ovom slučaju to je „VelicinaSobe“

**layout\_width –** određuje širinu našeg TextView -a; postavljen je na 0dp što znači da će TextView zauzeti onoliko mjesta koliko ima znakova

**layout\_height -** određuje visinu našeg textView -a; postavljen je na match\_parent što znači da će textView zauzeti cijelu visinu do parenta

**layout\_margin –** veličina sve 4 margine

**text –** tekst

**textSize –** veličina teksta

**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg TextView;

**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg TextView-a;

**layout\_constraintTop\_toBottomOf -** određuje nam s čime je poravnat gornji dio našeg TextView -a;

# **Dizajn sučelja „autentikacija“**

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, dizajn

Opis je automatski generiran

Dizajn se sastoji od 2 editTexta i 2 buttona.

U 2 editTexta unosimo podatke: E-mail i lozinku.

Pomoću buttona login se logiramo u našu bazu podataka ako već imamo napravljen račun, a pomoću Register button pravimo račun prvi puta.

## Programski kod „autentikacija“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

***Programski kod editTexta „email“***

**android:id="@+id/eMail":** Postavlja jedinstveni identifikator za ovaj element, što olakšava pristup iz programskog koda.

**android:layout\_width="match?parent":** Postavlja širinu elementa kao širinu zaslona odnosno maksimalnu širinu.

**android:layout\_height="wrap\_content":** Postavlja visinu elementa na veličinu sadržaja, što znači da će se prilagoditi visini teksta ili drugog sadržaja unutar elementa.

**android:hint="E-mail /** [**-@gmail.com**](mailto:-@gmail.com)**"**  Dodaje hint u editText

**android:backgroundTint="@color/secondary":** Postavlja boju pozadine elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".

**android:textColor="@color/secondary":** Postavlja boju teksta unutar elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".

EditText smo constraintali za parenta odnosno za rubove našeg zaslona.

Postavili smo margine lijevo, desno i gore na 60dp kako bi smo ograničili veličinu našeg EditTexta

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

**android:id="@+id/eMail":** Postavlja jedinstveni identifikator za ovaj element, što olakšava pristup iz programskog koda.

**android:layout\_width="match?parent":** Postavlja širinu elementa kao širinu zaslona odnosno maksimalnu širinu.

**android:layout\_height="wrap\_content":** Postavlja visinu elementa na veličinu sadržaja, što znači da će se prilagoditi visini teksta ili drugog sadržaja unutar elementa.

**android:hint=“Lozinka"**  Dodaje hint u editText

**android:backgroundTint="@color/secondary":** Postavlja boju pozadine elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".

**android:textColor="@color/secondary":** Postavlja boju teksta unutar elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".

Postavili smo margine lijevo, desno i gore na 60dp kako bi smo ograničili veličinu našeg EditTexta

**android:inputType="textPassword"** :Postavili smo inputType na teksutalnu lozinku što nam zapravo sakriva naš uneseni tekst.

EditText „lozinka“ smo constraintali na prethodni EditText „eMail“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran

**android:id="@+id/prijava":** Postavlja jedinstveni identifikator za ovaj element, olakšavajući pristup iz programskog koda.

**android:layout\_width="wrap\_content":** Postavlja širinu elementa na veličinu sadržaja, što znači da će se prilagoditi širini teksta ili drugog sadržaja unutar elementa.

**android:layout\_height="wrap\_content":** Postavlja visinu elementa na veličinu sadržaja, što znači da će se prilagoditi visini teksta ili drugog sadržaja unutar elementa.

**android:text="prijava":** Postavlja tekst unutar elementa na „prijava“

**android:backgroundTint="@color/accent":** Postavlja boju pozadine elementa na boju definiranu u resursima boja (color resource) pod imenom "accent".

**android:textColor="@color/secondary":** Postavlja boju teksta unutar elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".

**android:layout\_marginEnd="170dp":** Postavlja desni margina elementa na 170 dp. Ovo pomiče element udesno, stvarajući prostor između desne granice elementa i desne granice roditeljskog kontejnera.

**app:layout\_constraintBottom\_toBottomOf="parent":** Postavlja donju granicu elementa da bude na donjoj granici roditeljskog kontejnera, koristeći ConstraintLayout.

**app:layout\_constraintEnd\_toStartOf="@+id/lozinka":** Postavlja desnu granicu elementa da bude s lijeve strane elementa s identifikatorom "lozinka", koristeći ConstraintLayout.

**app:layout\_constraintStart\_toStartOf="@+id/lozinka ":** Postavlja lijevu granicu elementa da bude na lijevoj granici elementa „lozinka“ , koristeći ConstraintLayout.

**app:layout\_constraintTop\_toBottomOf="parent":** Postavlja gornju granicu elementa da bude na dnu elementa „lozinka“, koristeći ConstraintLayout.

**Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran**

**android:id="@+id/registracija":** Postavlja jedinstveni identifikator za ovaj element, olakšavajući pristup iz programskog koda.

**android:layout\_width="wrap\_content":** Postavlja širinu elementa na veličinu sadržaja, što znači da će se prilagoditi širini teksta ili drugog sadržaja unutar elementa.

**android:layout\_height="wrap\_content":** Postavlja visinu elementa na veličinu sadržaja, što znači da će se prilagoditi visini teksta ili drugog sadržaja unutar elementa.

**android:text="prijava":** Postavlja tekst unutar elementa na „prijava“

**android:backgroundTint="@color/accent":** Postavlja boju pozadine elementa na boju definiranu u resursima boja (color resource) pod imenom "accent".

**android:textColor="@color/secondary":** Postavlja boju teksta unutar elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".

**android:layout\_marginEnd="170dp":** Postavlja desni margina elementa na 170 dp. Ovo pomiče element udesno, stvarajući prostor između desne granice elementa i desne granice roditeljskog kontejnera.

**app:layout\_constraintBottom\_toBottomOf="@+id/prijava“:** Postavlja donju granicu elementa da bude na donjoj elementa „prijava“, koristeći ConstraintLayout.

**app:layout\_constraintEnd\_toStartOf="@+id/lozinka":** Postavlja desnu granicu elementa da bude s lijeve strane elementa s identifikatorom "lozinka", koristeći ConstraintLayout.

**app:layout\_constraintStart\_toStartOf="@+id/lozinka ":** Postavlja lijevu granicu elementa da bude na lijevoj granici elementa „lozinka“ , koristeći ConstraintLayout.

**app:layout\_constraintTop\_toBottomOf=="@+id/prijava":** Postavlja gornju granicu elementa da bude na dnu elementa „prijava“, koristeći ConstraintLayout.

# **MainActivity**

Slika na kojoj se prikazuje tekst, snimka zaslona, softver, Multimedijski softver

Opis je automatski generiran

**lateinit var binding: ActivityMainBinding:** Ova linija deklarira svojstvo binding tipa ActivityMainBinding s odgodnim inicijaliziranjem (lateinit)

**private val database: DatabaseReference:** Ova linija deklarira privatno svojstvo database tipa DatabaseReference. DatabaseReference je klasa koja dolazi s Firebase Realtime Database bibliotekom. Ovdje se stvara instanca FirebaseDatabase s određenim URL-om za Firebase Realtime Database ("https://pmuprojekt-default-rtdb.europe-west1.firebasedatabase.app/"), a zatim se dobiva referenca na "tekst" čvor u toj bazi podataka.

**binding = ActivityMainBinding.inflate(layoutInflater):** Ova linija koda koristi generiranu klasu ActivityMainBinding iz View Binding-a kako bi se inflatirao (stvorio) objekt binding.

**setContentView(binding.root):** Ova linija postavlja sadržaj aktivnosti na korijenski (root) pogled iz objekta binding. Metoda setContentView() se koristi u Android aktivnosti kako bi se postavio prikaz sadržaja aktivnosti.

**val ime = binding.BrojSobe.text.toString():** Ova linija dohvaća tekst unesen u polje s imenom "BrojSobe" pomoću View Binding-a i pretvara ga u string. Vrijednost se pohranjuje u varijablu BrojSobe.

**val prezime = binding.TipSobe.text.toString():** Slično kao prethodna linija, ova linija dohvaća tekst unesen u polje s imenom "TipSobe" i pretvara ga u string. Vrijednost se pohranjuje u varijablu TipSobe.

**val godine = binding.CijenaNoci.text.toString():** Ova linija dohvaća tekst unesen u polje s imenom "CijenaNoci" i pretvara ga u string. Vrijednost se pohranjuje u varijablu CijenaNoci.

**val radniPolozaj = binding.VelicinaSobe.text.toString():** Ova linija dohvaća tekst unesen u polje s imenom "VelicinaSobe" i pretvara ga u string. Vrijednost se pohranjuje u varijablu VelicinaSobe.

**var id = 0:** Inicijalizacija varijable id na 0. Ova varijabla će kasnije sadržavati jedinstveni identifikator za novog zaposlenika.

**if (!list.isEmpty()) id = list[list.size - 1].id + 1:** Ova linija provjerava je li lista list prazna. Ako nije prazna, dodjeljuje se vrijednost id zadnjeg zaposlenika u listi povećana za 1, čime se generira jedinstveni identifikator za novog zaposlenika.

**list.add(Soba(ID, BrojSobe, TipSobe, CijenaNoci, VelicinaSobe)):** Dodavanje nove sobe u listu list. Koristi se konstruktor klase soba s podacima koje smo prikupili ranije.

**database.setValue(list**): Postavljanje vrijednosti u Firebase Realtime Database. Cijela lista list, koja sadrži podatke o zaposlenicima, postavlja se kao nova vrijednost u bazi podataka. Ova linija se koristi za ažuriranje podataka u stvarnom vremenu na Firebaseu.

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

**list.clear():** Prije nego što se ažurira s novim podacima, stari podaci u listi list se brišu.

**try { ... }:** Pokušaj izvršiti sljedeći blok koda. U slučaju neuspjeha (iznimka), program će preskočiti blok i nastaviti s izvođenjem ostatka koda.

**val a: List<Soba> = snapshot.children.map { ... }:** Pomoću snapshot.children prolazi kroz svu djecu (podatke) u trenutnom stanju baze podataka. Metoda map konvertira te podatke u listu objekata tipa Soba.

**list.addAll(a):** Dohvaćeni podaci se dodaju u listu list.

**binding.RV.apply { ... }:** Postavljanje RecyclerView-a pomoću View Binding-a s ažuriranom listom zaposlenika.

**adapter = employees\_adapter(list, this@MainActivity):** Postavljanje adaptera za RecyclerView. employees\_adapter je adapter koji se koristi za povezivanje podataka iz liste s prikazom u RecyclerView.

**onCancelled(error: DatabaseError):** Ova metoda se poziva ako čitanje iz baze podataka ne uspije, ali u navedenom kodu nije implementirana.

# **Data class „Soba“**

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Opis je automatski generiran

***Programski kod klase „employees“***

**data class Soba(:** Ova linija označava početak definicije data klase imenom Soba.

**var BrojSobe: String = "",:** Ovo definira svojstvo (property) BrojSobe tipa String unutar data klase soba. Zadana vrijednost je postavljena na prazan string.

**var TipSobe: String = "",:** Slično kao i prethodno, ova linija definira svojstvo TipSobe tipa String unutar data klase, s praznim stringom kao zadanim.

**var CijenaNoci: String = "",:** Isto kao i prethodne dvije linije, ova linija definira svojstvo CijenaNoci tipa String.

**var VelicinaSobe: String = "",:** Ova linija definira svojstvo VelicinaSobe tipa String.

**var id: Int = 0:** Ova linija definira svojstvo id tipa Int s zadanim vrijednostima postavljenim na 0.

# **Class „TextAdapter“**

Slika na kojoj se prikazuje tekst, snimka zaslona, zaslon, softver

Opis je automatski generiran

***Programski kod klase „TextAdapter“***

**val list: ArrayList<Soba>,:** Ovo definira svojstvo list kao ArrayList objekata tipa soba. To je lista soba koju će adapter prikazivati.

**val th: Context:** Ovo definira svojstvo th kao objekt tipa Context. Context se obično koristi za pristup resursima, pozivanje aktivnosti i slično.

**): RecyclerView.Adapter<employees\_adapter.ViewHolder>() {:** Ova linija zatvara definiciju klase i označava da employees\_adapter nasljeđuje od RecyclerView.Adapter klase. Navodi se i da koristi ViewHolder unutar RecyclerView.Adapter.

**override fun onCreateViewHolder(:** Ova funkcija se koristi za stvaranje novog ViewHolder objekta.

**val v = SobaItemBinding.inflate(LayoutInflater.from(th), parent, false): Ovdje** se inflatira (stvara) objekt SobaItemBinding iz layouta.

**return ViewHolder(v):** Vraća se novi ViewHolder stvoren sa SobaItemBinding objektom.

**override fun onBindViewHolder(:** Ova funkcija se koristi za postavljanje podataka iz list na određenu poziciju position u ViewHolder-u.

**holder.bindItem(list[position], th):** Poziva se funkcija bindItem na ViewHolder objektu, a kao argumenti se predaju podaci (zaposlenik) na toj poziciji i Context objekt.

**override fun getItemCount(): Int {:** Ova funkcija vraća ukupan broj stavki u listi. To je važno za RecyclerView kako bi znao koliko stavki treba prikazati.

**class ViewHolder(:** Ovo označava početak definicije unutarnje klase ViewHolder. ViewHolder je odgovoran za držanje referenci na elemente sučelja koje je potrebno ažurirati.

**private var itemBinding: SobaItemBinding:** Ovo definira svojstvo itemBinding tipa SobaItemBinding, koje drži referencu na View Binding objekt za svaki element retka.

**RecyclerView.ViewHolder(itemBinding.root):** Poziva konstruktor roditeljske klase i prosljeđuje mu root View iz ItemEmployeeBinding.

**fun bindItem(tekst:soba, th:Context){:** Ova funkcija se koristi za postavljanje podataka (soba) na odgovarajuće elemente sučelja.

**itemBinding.BrojSobe.text = tekst.BrojSobe:** Postavljanje teksta za BrojSobe na odgovarajući element sučelja.

**itemBinding.TipSobe.text = tekst.TipSobe:** Postavljanje teksta za TipSobe na odgovarajući element sučelja.

**itemBinding.CijenaNoci.text = tekst.CijenaNoci:** Postavljanje teksta za CijenuNoci sobe na odgovarajući element sučelja.

**itemBinding.VelicinaSobe.text = tekst.VelicinaSobe:** Postavljanje teksta za VelicinuSobe na odgovarajući element sučelja.

**itemBinding.id.text = tekst.id.toString():** Postavljanje teksta za identifikator sobe na odgovarajući element sučelja, pretvarajući id u string.

# **Class „LoginActivity“**

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

***Programski kod klase „loginActivity“***

**private lateinit var auth: FirebaseAuth:** Deklaracija privatne varijable auth tipa FirebaseAuth, koja će se koristiti za upravljanje autentikacijom korisnika.

**private lateinit var binding: AutentikacijaBinding**: Deklaracija privatne varijable binding koja se koristi za povezivanje s XML sučeljem definiranim u AutentikacijaBinding. Korištenje lateinit znači da će se varijabla inicijalizirati kasnije, prije nego što joj se pristupi.

**private val database2: DatabaseReference = ...:** Deklaracija i inicijalizacija privatnih varijabli database2 i database3 tipa DatabaseReference koje predstavljaju referencu na određenu lokaciju u Firebase Realtime Database. database2 se odnosi na "User", a database3 na "UsersLeaderboard".

**private var userId: String = "":** Deklaracija privatne varijable userId tipa String koja će se koristiti za pohranu identifikatora trenutnog korisnika.

**binding = AutentikacijaBinding.inflate(layoutInflater):** Inicijalizacija varijable binding pomoću generiranog objekta AutentikacijaBinding, što omogućuje pristup elementima sučelja.

**setContentView(binding.root):** Postavljanje sadržaja aktivnosti na korijenski element sučelja definiran u binding.

**auth = FirebaseAuth.getInstance():** Inicijalizacija varijable auth pomoću singleton instance FirebaseAuth objekta.

**val currentUser = auth.currentUser:** Dohvaćanje trenutno prijavljenog korisnika, ako postoji.

**if(currentUser != null){...}:** Provjera je li trenutni korisnik već prijavljen. Ako je, postavlja se userId, stvara se Intent za prelazak na Menu aktivnost, i pokreće se ta aktivnost uz prosljeđivanje trenutnog korisnika.

Slika na kojoj se prikazuje tekst, snimka zaslona, softver

Opis je automatski generiran

**binding.login.setOnClickListener{...}:** Postavljanje klikera (click listener) na gumb za prijavu.

**if (binding.email.text.toString().isEmpty()):** Provjera je li polje za unos e-pošte prazno. Toast.makeText(applicationContext, getString(R.string.unesi\_ime), Toast.LENGTH\_SHORT).show(): Ako je polje za unos e-pošte prazno, prikazuje se kratka obavijest korisniku da unese svoje ime.

**else if (binding.password.text.toString().isEmpty**()): Provjera je li polje za unos lozinke prazno. Toast.makeText(applicationContext, getString(R.string.unesi\_lozinku), Toast.LENGTH\_SHORT).show(): Ako je polje za unos lozinke prazno, prikazuje se kratka obavijest korisniku da unese lozinku.

**else if (binding.password.text.toString().length < 8):** Provjera je li unesena lozinka kraća od 8 znakova. Toast.makeText(applicationContext, getString(R.string.pass\_kratak), Toast.LENGTH\_SHORT).show(): Ako je unesena lozinka kraća od 8 znakova, prikazuje se kratka obavijest korisniku da unese dužu lozinku.

**else{...}:** Ako niti jedan od prethodnih uvjeta nije ispunjen, izvršava se blok koda unutar else-a. val str = binding.email.text.toString().filter { !it.isWhitespace() }: Uklanja bijele znakove iz unesenog teksta e-pošte.

**val email = "$str@gmail.com":** Stvara se e-pošta dodajući "@gmail.com" na obrađeni tekst e-pošte.

**val password = binding.password.text.toString():** Dohvaća se lozinka iz unesenog teksta lozinke. auth.signInWithEmailAndPassword(email, password).addOnCompleteListener(this) { task ->...}: Pokušava izvršiti prijavu korisnika s unesenom e-poštom i lozinkom. Dodaje se slušatelj na taj zadatak (task).

**if (task.isSuccessful){...}:** Ako je prijava uspješna, prikazuje se obavijest o uspješnoj prijavi, a zatim se korisnika preusmjerava na Menu aktivnost.

**else {...}:** Ako prijava nije uspjela, prikazuje se obavijest o neuspješnoj prijavi.

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

**if (binding.email.text.toString().isEmpty()):** Provjerava je li polje za unos e-maila prazno. Ako jest, prikazuje poruku korisniku da unese e-mail.

**else if (binding.password.text.toString().isEmpty()):** Ako polje za unos lozinke prazno, prikazuje poruku korisniku da unese lozinku.

**else if (binding.password.text.toString().length < 8):** Ako lozinka ima manje od 8 znakova, prikazuje poruku korisniku da unese dulju lozinku.

**val str = binding.email.text.toString().filter { !it.isWhitespace() }:** Filtrira praznine iz unesenog e-maila.

**val email = "$str@gmail.com**": Stvara konačan oblik e-maila, dodajući "@gmail.com" na filtrirani e-mail.

**val password = binding.password.text.toString():** Dohvaća unesenu lozinku.

**auth.createUserWithEmailAndPassword(email, password):** Koristi Firebase Auth da pokuša kreirati korisnički račun s unesenim e-mailom i lozinkom.

**.addOnCompleteListener(this) { task ->:** Postavlja OnCompleteListener za obradu rezultata kreiranja korisničkog računa.

**if (task.isSuccessful):** Provjerava je li kreiranje korisničkog računa bilo uspješno. Toast.makeText(baseContext, getString(R.string.uspjesna\_registracija), Toast.LENGTH\_SHORT).show(): Ako jest, prikazuje poruku korisniku o uspješnoj registraciji.

**val user** = auth.currentUser: Dohvaća trenutno registriranog korisnika.

**userId = user!!.uid:** Postavlja userId na UID trenutno registriranog korisnika.

**val currentUserDb = database2.child(userId):** Postavlja referencu u Firebase Realtime Database na podatke tog korisnika.

**else:** Ako kreiranje korisničkog računa nije uspjelo. Toast.makeText(baseContext, getString(R.string.neuspjesna\_registracija), Toast.LENGTH\_SHORT).show(): Prikazuje poruku korisniku o neuspješnoj registraciji.