Programiranje mobilnih uređaja

Implementacija sustava za upravljanje zoološkim vrtom

Petar Lukač 3.G

Elektrotehnička i ekonomska škola

Nova Gradiška

Sadržaj

[**Uvod** 3](#_Toc160655317)

[**Dizajn glavnog sučelja** 4](#_Toc160655318)

[Programski kod glavnog sučelja 5](#_Toc160655319)

[**Programski kod „item\_korisnik“** 8](#_Toc160655320)

[**Dizajn sučelja „Autentikacija“** 13](#_Toc160655321)

[Programski kod „autentikacija“ 14](#_Toc160655322)

[**MainActivity** 16](#_Toc160655323)

[**Data class „korisnik“** 18](#_Toc160655324)

[**Class „korisniks\_adapter“** 18](#_Toc160655325)

[**Class „Authentication“** 19](#_Toc160655326)

# **Uvod**

# **Dizajn glavnog sučelja**

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

Dizajn se sastoji od 4 editTexta, buttona i RecyclerViewa  
U 3 editTexta unosimo podatke: ime, prezime i godine  
Pomoću buttona unosimo podatke u RecyclerView  
Preko RecyclerViewa gledamo unesene podatke

## Programski kod glavnog sučelja

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod buttona „Unesi“***

**id –** označava id našeg gumba, u ovom slučaju to je „commit“ **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; u našem slučaju s gornjim dijelom RV „recycler“  
**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg gumba; u ovom slučaju s desnim dijelom zaslona  
**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg gumba; u ovom slučaju s lijevom stranom ekrana „parent“

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

Opis je automatski generiran Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod editText „ime“, editText „prezime“ i editText “godine“***

**id –** označava id našeg editTexta, u ovom slučaju to je „ime“**,** što „prezime“ i što je „godine“  
**layout\_width –** određuje širinu našeg editTexta; postavljen je na 0dp što znači da će editText zauzeti onoliko mjesta koliko ima znakova  
**layout\_height -** određuje visinu našeg editTexta; 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 prezime“ i on će biti ispisan kada ništa nije upisano u EditTextu  
**textColorHint –** boja teksta za hint  
**textSize –** veličina teksta  
**layout\_constraintBottom\_toTopOf -** određuje nam s čime je poravnat donji dio našeg editTexta; u našem slučaju s gornjim dijelom recyclera „recycler“  
**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 editTexta „ime“  
**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg editTexta; u našem slučaju s gornjim dijelom editTexta „ime“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod recyclerViewa „recycler“***

**id –** označava id našeg recyclera, u ovom slučaju to je „recycler“ **layout\_width –** određuje širinu našeg recyclera; postavljen je na 0dp što znači da će gumb zauzeti onoliko mjesta koliko ima znakova  
**layout\_height -** određuje visinu našeg recyclera; postavljen je na wrap\_content što znači da će gumb zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u recycler  
**layout\_marginStart ­–** početna margina (lijeva)  
**layout\_marginTop ­–** gornja margina   
**layout\_marginEnd ­–** krajnja margina (desna)  
**layout\_marginBottom ­–** donja margina  
**layout\_constraintBottom\_toBottomOf -** određuje nam s čime je poravnat donji dio našeg recyclera; u našem slučaju s donjim dijelom ekrana  
**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg recyclera; u ovom slučaju s desnim dijelom ekrana  
**layout\_constraintStart\_toEndOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg recyclera; u ovom slučaju s lijevom stranom ekrana  
**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg recyclera; u našem slučaju s donjim dijelom editTexta „ime“

# **Programski kod „item\_korisnik“**

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod textViewa „id“***

**id –** označava id našeg editTexta, 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  
**layout\_height -** određuje visinu našeg textViewa; postavljen je na wrap\_content što znači da će textView zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst u textView  
**textSize –** veličina teksta **textColor –** boja teksta  
**layout\_weight –** težina elementa  
**gravity –** poravnanje  
**layout\_margin ­–** sve 4 margine   
**layout\_constraintBottom\_toTopOf -** određuje nam s čime je poravnat donji dio našeg textViewa; u našem slučaju s donjim dijelom linearLayout „ll“  
**layout\_constraintEnd\_toEndOf –** određuje nam s čime je poravnat kraj (desni dio) našeg textView; u ovom slučaju s krajem (desnim dijelom) linearLayout „ll2“  
**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg textViewa; u ovom slučaju s desnom stranom linearLayout „ll2“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod Linear layouta „ll“***

**id –** označava id našeg Linear layouta, u ovom slučaju to je „ll“ **layout\_width –** određuje širinu našeg Linear layouta; postavljen je na match\_parent što znači da će Linear layout zauzeti mjesto preko cijele širine ekrana   
**layout\_height -** određuje visinu našeg Linear layout; postavljen je na wrap\_content što znači da će Linear layout zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst  
**layout\_margin ­–** sve 4 margine   
**orientation –** orijentacija Linear layouta  
**layout\_constraintBottom\_toBottomOf -** određuje nam s čime je poravnat donji dio našeg Linear layouta; u našem slučaju s donjim dijelom linearLayout „ll2“  
**layout\_constraintTop\_toTopOf –** određuje nam s čime je poravnat gornji dio našeg Linear layouta; u ovom slučaju s gornjom stranom zaslona

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran  
***Programski kod TextViewa „ime“ i „prezime“***

**id –** označava id našeg TextViewa, u ovom slučaju to je „ime“ **layout\_width –** određuje širinu našeg TextViewa; postavljen je na 0dp što znači da će TextView zauzeti onoliko mjesta koliko ima znakova  
**layout\_height -** određuje visinu našeg TextViewa; postavljen je na wrap\_content što znači da će TextView zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst  
**layout\_weight –** težina elementa  
**layout\_margin –** veličina sve 4 margine  
**textColor –** boja teksta  
**text –** tekst  
**gravity –** poravnanje  
**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\_toStartOf –** određuje nam s čime je poravnat početak (lijeva strana) našeg TextViewa; u ovom slučaju s lijevom stranom ekrana  
**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg TextViewa; u našem slučaju s gornjim dijelom ekrana „parent“ i gornjim dijelom TextViewa „ime“

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod Linear layout „ll2“***

**id –** označava id našeg Linear layouta, u ovom slučaju to je „ll2“  
**layout\_width –** određuje širinu našeg Linear layouta; postavljen je na match\_parent što znači da će Linear layout zauzeti mjesto preko cijele širine ekrana   
**layout\_height -** određuje visinu našeg Linear layouta; postavljen je na wrap\_content što znači da će Linear layout zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst  
**layout\_margin ­–** sve 4 margine   
**orientation –** orijentacija Linear layouta  
**layout\_constraintBottom\_toBottomOf -** određuje nam s čime je poravnat donji dio našeg Linear layouta; u našem slučaju s donjim dijelom ekrana  
**layout\_constraintStart\_toStartOf –** određuje nam s čime je poravnat lijevi dio našeg Linear layouta; u ovom slučaju s lijevom stranom Linear layouta „ll“  
**layout\_constraintTop\_toTopOf -** određuje nam s čime je poravnat gornji dio našeg Linear layouta; u našem slučaju s gornjim dijelom ekrana

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Opis je automatski generiran  
***Programski kod TextViewa „godine“***

**id –** označava id našeg TextViewa, u ovom slučaju to je „godine“ **layout\_width –** određuje širinu našeg TextViewa; postavljen je na 0dp što znači da će TextView zauzeti onoliko mjesta koliko ima znakova  
**layout\_height -** određuje visinu našeg TextViewa; postavljen je na wrap\_content što znači da će TextView zauzeti onoliko mjesta koliko mu je potrebno da obuhvati tekst  
**layout\_weight –** težina elementa  
**layout\_margin –** veličina sve 4 margine  
**textColor –** boja teksta  
**text –** tekst  
**gravity –** poravnanje  
**textSize –** veličina teksta

# **Dizajn sučelja „Autentikacija“**

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

Opis je automatski generiran  
***Programski kod editTexta „email“ i „password“***

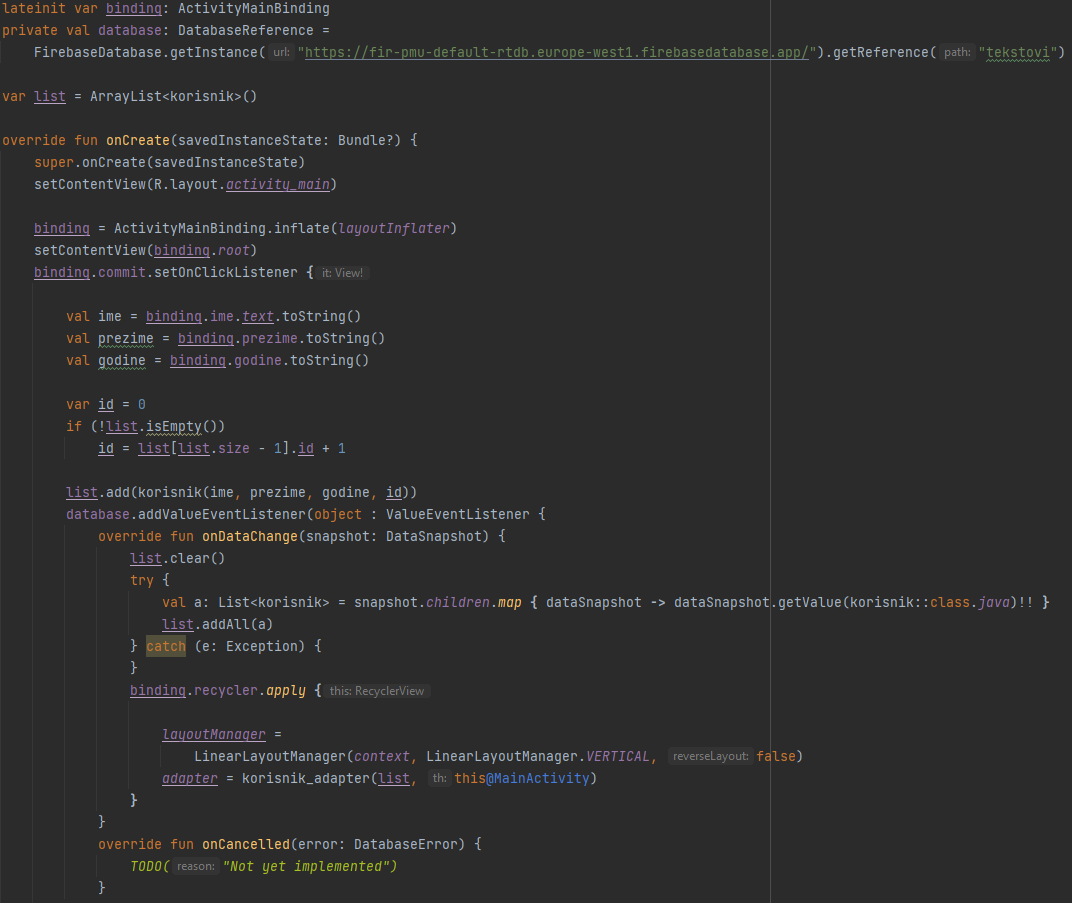
**ndroid:id="@+id/":** Postavlja jedinstveni identifikator za ovaj element, što olakšava pristup iz programskog koda.  
**android:layout\_width="wrap\_content":** Postavlja širinu elementa na veličinu sadržaja, što znači da će se prilagoditi širini teksta ili drugog sadržaja unutar elementa.  
**android:layout\_height="wrap\_content":** Postavlja visinu elementa na veličinu sadržaja, što znači da će se prilagoditi visini teksta ili drugog sadržaja unutar elementa.   
**android:text="@string/email":** Postavlja tekst unutar elementa pomoću resursa niza (string resource). Tekst dolazi iz resursa s identifikatorom "email".   
**android:backgroundTint="@color/secondary":** Postavlja boju pozadine elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".   
**android:textColor="@color/accent":** Postavlja boju teksta unutar elementa na boju definiranu u resursima boja (color resource) pod imenom "accent".   
**app:layout\_constraintBottom\_toTopOf="@+id/login":** Postavlja donju granicu elementa da bude iznad gornje granice elementa s identifikatorom "login", koristeći ConstraintLayout.   
**app:layout\_constraintEnd\_toStartOf="@+id/register":** Postavlja desnu granicu elementa da bude s lijeve strane elementa s identifikatorom "register", koristeći ConstraintLayout.   
**app:layout\_constraintStart\_toEndOf="@+id/login":** Postavlja lijevu granicu elementa da bude s desne strane elementa s identifikatorom "login", koristeći ConstraintLayout.   
**app:layout\_constraintTop\_toTopOf="parent":** Postavlja gornju granicu elementa da bude na vrhu roditeljskog kontejnera, koristeći ConstraintLayout. Ovo će postaviti element na vrh ekrana ili na vrh određenog područja unutar ekrana.

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

Opis je automatski generiran  
***Programski kod buttona „register“ i „login“***

**android:id="@+id/ ":** Postavlja jedinstveni identifikator za ovaj element, što omogućuje jednostavan 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="@string/ ":** Postavlja tekst unutar elementa pomoću resursa niza (string resource). Tekst dolazi iz resursa s identifikatorom "register" ili „login“.  
**android:layout\_marginStart="":** Postavlja lijevi margina elementa na 125 i 100 dip (density-independent pixels). Ovo pomiče element ulijevo, stvarajući prostor između lijeve granice elementa i lijeve granice roditeljskog kontejnera.   
**android:textColor="@color/accent":** Postavlja boju teksta unutar elementa na boju definiranu u resursima boja (color resource) pod imenom "accent".   
**android:backgroundTint="@color/secondary":** Postavlja boju pozadine elementa na boju definiranu u resursima boja (color resource) pod imenom "secondary".  
**app:layout\_constraintBottom\_toBottomOf="@+id/login":** Postavlja donju granicu elementa da bude na donjoj granici elementa s identifikatorom "login", koristeći ConstraintLayout.  
**app:layout\_constraintEnd\_toEndOf="parent": Postavlja** desnu granicu elementa da bude na desnoj granici roditeljskog kontejnera, koristeći ConstraintLayout.   
**app:layout\_constraintStart\_toStartOf="parent":** Postavlja lijevu granicu elementa da bude na lijevoj granici roditeljskog kontejnera, koristeći ConstraintLayout.  
**app:layout\_constraintTop\_toTopOf="@+id/login":** Postavlja gornju granicu elementa da bude na gornjoj granici elementa s identifikatorom "login", koristeći ConstraintLayout.

# **MainActivity**



**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://fir-pmu-default-rtdb.europe-west1.firebasedatabase.app/"), a zatim se dobiva referenca na "zaposlenici" čvor u toj bazi podataka.  
**binding = ActivityMainBinding.inflate(layoutInflater):** Ova linija koda koristi generiranu klasu ActivityMainBinding iz View Binding 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.ime.text.toString():** Ova linija dohvaća tekst unesen u polje s imenom "ime" pomoću View Binding i pretvara ga u string. Vrijednost se pohranjuje u varijablu ime.  
**val prezime = binding.prezime.text.toString():** Slično kao prethodna linija, ova linija dohvaća tekst unesen u polje s imenom "prezime" i pretvara ga u string. Vrijednost se pohranjuje u varijablu prezime.  
**val godine = binding.godine.text.toString():** Ova linija dohvaća tekst unesen u polje s imenom "godine" i pretvara ga u string. Vrijednost se pohranjuje u varijablu godine.  
**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(korisnik(ime, prezime, godine, id)):** Dodavanje novog zaposlenika u listu list. Koristi se konstruktor klase korisniks 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.  
**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<korisniks> = 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 korisniks.   
**list.addAll(a):** Dohvaćeni podaci se dodaju u listu list.   
**binding.recycler.apply { ... }:** Postavljanje RecyclerView pomoću View Binding s ažuriranom listom zaposlenika.   
**adapter = korisniks\_adapter(list, this@MainActivity):** Postavljanje adaptera za RecyclerView. korisniks\_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 „korisnik“**

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

Opis je automatski generiran

**data class korisnik(:** Ova linija označava početak definicije data klase imenom korisnik.  
**var ime: String = "",:** Ovo definira svojstvo (property) ime tipa String unutar data klase korisnik. Zadana vrijednost je postavljena na prazan string.   
**var prezime: String = "",:** Slično kao i prethodno, ova linija definira svojstvo prezime tipa String unutar data klase, s praznim stringom kao zadanim.   
**var godine: String = "",:** Isto kao i prethodne dvije linije, ova linija definira svojstvo godine tipa String.   
**var id: Int = 0:** Ova linija definira svojstvo id tipa Int s zadanim vrijednostima postavljenim na 0.

# **Class „korisniks\_adapter“**

--- errori

# **Class „Authentication“**

Slika na kojoj se prikazuje tekst, snimka zaslona

Opis je automatski generiran

**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, Multimedijski 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. 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, softver, Multimedijski softver

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.