Android aplikacija za pomoć pri praćenju vođenja tvrtke

Kuridža, Igor

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

Sveučilišni diplomski studij Računarstvo

ANDROID APLIKACIJA ZA POMOĆ PRI PRAĆENJU VOĐENJA TVRTKE

Diplomski rad

Igor Kuridža

Osijek, 2021.

SADRŽAJ

1. UVOD

U poslovanju svake tvrtke izrazito je važno praćenje poslovnih promjena. Napretkom tehnologije praćenje poslovnih promjena se može voditi pomoću različitih programa i pametnih telefona. Poduzetnicima je vrlo bitno imati uvid u poslovanje tvrtke. Postoje različiti načini kako veliki i mali poduzetnici mogu pratiti poslovne promjene. Veliki poduzetnici mogu koristiti profesionalnu pomoć knjigovodstvenih programa koji se plaćaju te imaju sve funkcionalnosti koje su im potrebne kako bi uspješno vodili tvrtku. Dok mali poduzetnici, kojima je u početku svaki trošak važan, mogu koristiti besplatne programe sa manjim, ali potrebnim brojem funkcionalnosti koje mogu malom poduzetniku pomoći da održava posao. Mali poduzetnici imaju mogućnost korištenja Microsoft Excel alata s kojim mogu organizirati Excel radnu knjigu pomoću koje imaju uvid u ulazne i izlazne račune te iznos poreza na dodanu vrijednost (PDV) koji moraju platiti za određeni mjesec (ako su u sustavu PDV-a). Takav pristup malom poduzetniku ne oduzima mnogo vremena i ono najbitnije ima besplatno rješenje s kojim bi bio u toku s poslovanjem tvrtke.

U praksi svi mali poduzetnici imaju svoj ovlašteni knjigovodstveni servis, ali on ne može u svakom trenutku poduzetniku pružiti informacije koje su mu potrebne za njegovo poslovanje te se poduzetnik mora snaći samostalno kako bi došao do potrebnih informacija. Zbog toga je nastala ideja o Android aplikaciji koja će poduzetniku pružiti pomoć pri praćenju ulaznih i izlaznih računa, zaposlenika, ponuda, plana poslova te izvještaja. Navedena aplikacija će biti opisana u ovom radu.

1.1. Zadatak diplomskog rada

Aplikacija treba omogućiti korisniku dodavanje računa poslovanja, vođenje evidencije o zaposlenicima, vođenje evidencije o ponudama, dodavanje bilješki poput plana rada, dodavanje troškova. U aplikaciju je potrebno ugraditi mogućnost grafičkog načina prikaza troškova i prihoda od poslova na mjesečnoj i godišnjoj razini.

2. POSLOVANJE TVRTKE

Najvažniji dokumenti u poslovanju tvrtke su ulazni i izlazni računi te će oni biti objašnjeni u ovom poglavlju.

2.1. Ulazni računi

Ulazni računi „ulaze“ u pravnu osobu u smislu troška. Ulazni računi (URA) znaju se i zvati ulazne fakture (UFA).

Prema [1] svaki ulazni račun da bi bio ispravan mora sadržavati sljedeće obavezne elemente:

- broj računa
- datum izdavanja
- ime i prezime (naziv), adresa, osobni identifikacijski broj (OIB) ili PDV identifikacijski broj poreznog obveznika koji je isporučio robu ili obavio uslugu
- ime i prezime (naziv), adresa, OIB ili PDV identifikacijski broj poreznog obveznika kome je isporučena roba ili obavljena usluga
- količina i uobičajeni trgovački naziv isporučenih dobara te vrsta, količina ili opseg obavljenih usluga,
- datum isporuke dobara ili obavljenih usluga ili datum primitka predujma u računu za predujam, ako se taj datum može odrediti i razlikuje se od datuma izdavanja računa,
- jedinična cijena bez PDV-a, odnosno iznos naknade za isporučena dobra i obavljene usluge, razvrstane po stopi PDV-a,
- popusti ili rabati ako nisu uključeni u jediničnu cijenu,
- stopa PDV-a,
- iznos PDV-a razvrstan po stopi PDV-a, osim ako se primjenjuje posebni postupak za koji je u smislu ovog Zakon podatak isključen,
- ukupni iznos naknade i PDV-a,
- dodatne napomene (napomena o prijenosu porezne obveze ili napomena o osnovi oslobođenja).

2.2. Izlazni računi

Prema [2] izlazni računi „izlaze“ od tvrtke kako bi ih naplatili. Izlazni računi (IRA) znaju se i zvati izlazne fakture (IFA).

Svaki izlazni račun da bi bio ispravan mora sadržavati sljedeće obavezne elemente:

- broj računa
- datum izdavanja računa
- naziv kupca
- adresa kupca
- mjesto kupca
- OIB ili PDV ID kupca
- vrijednost isporuke (bez PDV-a)
- iznos PDV-a u isporuci
- ukupni iznos računa (vrijednost + PDV)
- datum naplate – kada je račun naplaćen, ako račun nije naplaćen onda bude prazno
- napomena

2.3. Bruto i neto

Bruto je iznos na koji se obračunava mirovinsko, porezi i prirezi.

Neto je iznos koji se dobije kada država uzme „svoj dio“.

2.4. Stope poreza

Prema [3] PDV se obračunava po sniženoj stopi od 13% na isporuke sljedećih dobara i usluga:

- jestiva ulja i masti, biljnog i životinjskog podrijetla
- dječje sjedalice za automobile
- dječje pelene
- dječju hranu
- ulaznice za koncerte
- sadnice i sjemenje
- itd.

Prema [3] PDV se obračunava po sniženoj stopi od 5% na isporuke sljedećih dobara i usluga:

- sve vrste kruha

- medicinsku opremu, pomagala i druge sprave koje se koriste za ublažavanje liječenja invalidnosti isključivo za osobnu uporabu invalida propisane općim aktom o ortopedskim i drugim pomagalima Hrvatskoga zavoda za zdravstveno osiguranje
- kino ulaznice
- znanstvene časopise
- itd.

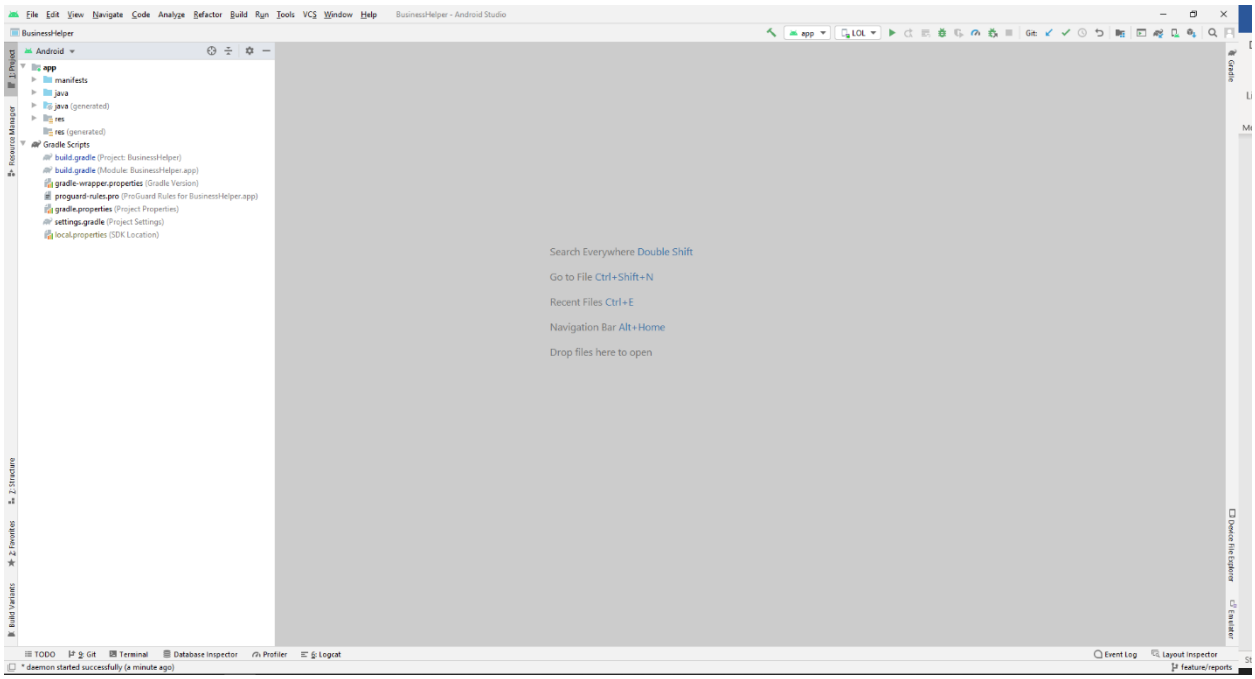
3. OPIS KORIŠTENIH TEHNOLOGIJA I ALATA

U ovom poglavlju biti će navedeni i objašnjeni korišteni alati te tehnologije.

3.1. Android Studio

Android Studio [4] je službeno integrirano razvojno okruženje za Google-ov operacijski sustav Android. Pruža brojne mogućnosti i alate kojima se ubrzava i pomaže u razvoju Android aplikacija, poput:

- brz emulator
- primijeniti promjene u kodu i resursima na pokrenutu aplikaciju bez ponovnog pokretanja aplikacije
- integracija s Git-om
- alati za praćenje performanse, kompatibilnosti verzija i drugih problema
- alati za testiranje



Slika 3.1. Sučelje Android Studio integriranog razvojnog okruženja

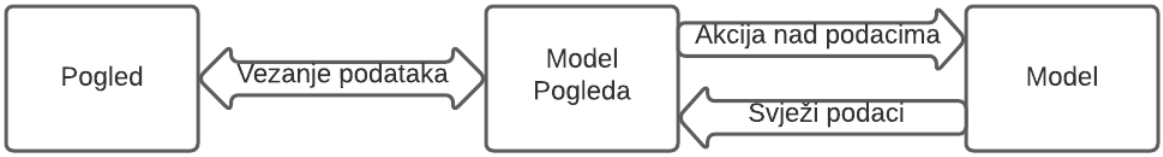
3.2. Kotlin

Programski jezik Kotlin [5] je moderan jezik koji je kompatibilan sa sustavom Android. Jednostavan je i optimizira broj linija koda u odnosu na Javu. Java i Kotlin su u potpunosti interoperabilni - Kotlin kod može se pokretati u Java datotekama i obratno. Prednosti koje

Kotlin donosi su: ništavni tipovi (engl. *nullable types*), funkcije proširenja (engl. *exstension functions*) te funkcije višeg reda (engl. *higher-order functions*). Pomoću ništavih tipova Kotlin smanjuje broj programskih pogrešaka vezanih uz neistancirane objekte (engl. *NullPointerException*). Oko 80% Android aplikacija koriste Kotlin programski jezik.

3.3. Model-Pogled-Model Pogleda arhitekturni obrazac

Model-Pogled-Model Pogleda (engl. *Model-View-ViewModel*, MVVM) arhitekturni obrazac [6] nam omogućuje stvaranje responzivne i dinamičke implementacije koda u Androidu, odvaja poslovnu logiku od sloja *pogleda*, poštujući načelo jedinstvene odgovornosti (engl. *Single Responsibility Priniciple*) [7]. MVVM obrazac je preporučen od strane *Google*-a za implementaciju u Android platformu. Na slici 3.2. prikazana je MVVM struktura i tok podataka.



Slika 3.2. MVVM struktura

Sloj *pogleda* je odgovoran za prikaz svježih podataka primljenih od *modela pogleda* te mu proslijediti interakciju korisnika. U Androidu sloj *pogleda* je obično aktivnost (engl. *activity*) ili fragment.

Model pogleda je centralni dio u MVVM arhitekturi. Sadrži svu poslovnu logiku te izlaže (engl. *expose*) podatke koje sloj pogleda može promatrati (engl. *observe*) te na taj način sloj *pogleda* ima najnovije podatke i može ažurirati korisničko sučelje u realnom vremenu.

Model je komponenta koja ne zna ni za sloj *pogleda* ni za *model pogleda.* Glavna mu je uloga spremanje stanja aplikacije i omogućivanje *modelu pogleda* da dohvati stanje. *Model* je najčešće kreiran pomoću spremišta obrasca (engl. *repository pattern*) [8].

3.4. DataBinding

Vezanje podataka (engl. *data binding*) [9] omogućuje automatsko osvježivanje prikaza na sučelju na promjenu podataka koji su naznačeni unutar *xml* datoteke koja predstavlja kako sučelje izgleda. Naznačeni podaci se povezuju s elementima sučelja unutar *xml* datoteke umjesto da se referenca elementa sučelja dohvaća te se promjena eksplicitno definira u kodu. Na taj način se smanjuje dupliciran kod (engl. *boilerplate code*) te kod postaje pregledniji.

3.5. Room

Room biblioteka (engl. *library*) [10] pruža sloj apstrakcije preko strukturiranog jezika upita (engl. *structured query language*, SQL) kako bi se omogućio pristup lokalnoj bazi podataka. Najčešće se koristi za lokalno spremanje (engl. *cache*) veće količine strukturiranih podataka kojima se može pristupiti i bez internetske veze. *Room* stvara pred memoriju podataka koja služi kao jedinstven izvor podataka aplikacije. *Room* biblioteka je preporučena od strane *Google-a* jer uklanja veliku većinu dupliciranog koda potrebnog za interakcija s *SQLite* i za vrijeme kompiliranja (engl. *compile time*) dodaje provjeru definiranih SQL upita.

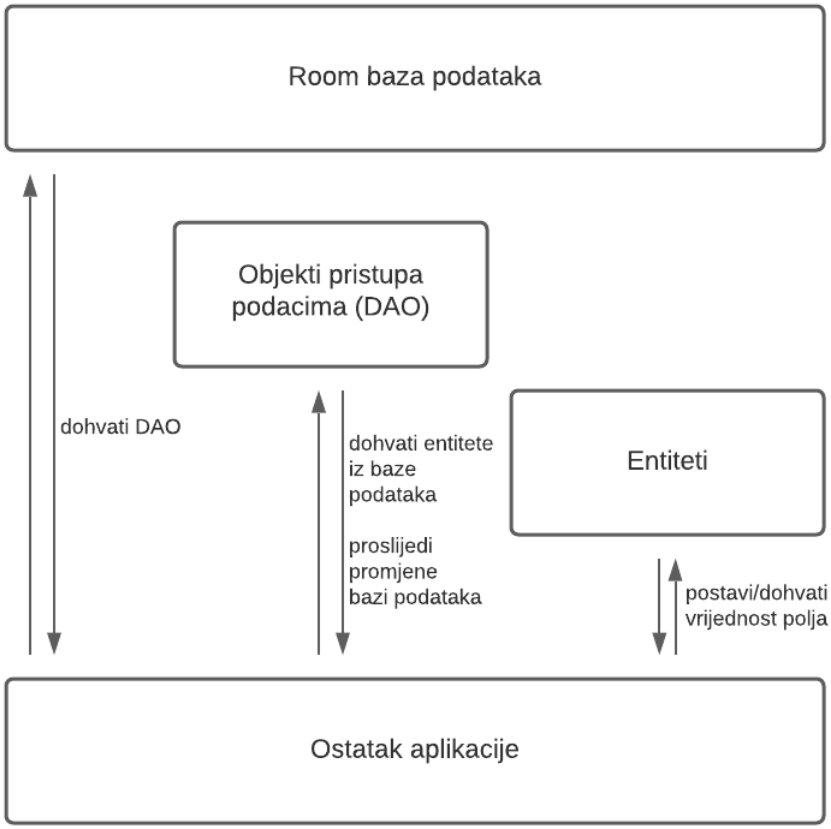
SQL je strukturni upitni jezik koji omogućuje pristup bazi podataka, dohvaćanje podataka iz baze, dodavanje novih podataka, brisanje postojećih podataka, izmjenu postojećih podataka iz baze.

Unutar *Room* biblioteke postoje tri glavne komponente: entitet, objekt pristupa podacima (engl. *data access object*, DAO) i baza podataka (engl. *database*). Na slici 3.3. prikazana je veza između glavnih komponenata *Room* biblioteke.

Entitet predstavlja podatke za jedan redak tablice. Definira se pomoću *@Entity* anotacije.

DAO pruža metode koje koristi aplikacija kako bi pristupala bazi podataka. Definira se pomoću *@Dao* anotacije.

Database predstavlja glavnu pristupnu točku za povezivanje s bazom podataka. Sadrži definirani popis entiteta, verzije baze podataka i slično. Unutar *Database* klase potrebno je definirati apstraktne (engl. *abstract*) metode koje nemaju argumente i vraćaju instancu od DAO klase kako bi *Room*, za vrijeme kompiliranja, automatski iz generirala implementaciju svih DAO koji su definirani. Definira se pomoću *@Database* anotacije i mora biti apstraktna klasa te naslijediti *RoomDatabase* baznu klasu (engl. *base class*).



Slika 3.3. Dijagram arhitekture *Room* biblioteke

3.6. Navigation Component

Navigation Component biblioteka [11] pomaže i olakšava implementaciju navigacije unutar aplikacije te pruža mogućnost uvida u vizualni prikaz navigacije aplikacije.

Navigacijom se smatraju interakcije koje korisnicima omogućuju navigaciju kroz različite dijelove sadržaja unutar aplikacije i nazad.

Prednosti koje *Navigation Component* biblioteka donosi:

- Rukovanje s fragment transakcijama
- Ispravno rukovanje s akcijama prema naprijed i nazad
- Pružanje resursa za animacije i tranzicije
- Implementacija i rukovanje dubokim povezivanjem (engl. *deep linking*)
- Sigurni argumenti (engl. *Safe Args*) [12] - pružaju sigurnost s tipovima podataka prilikom navigacije i prijenosa podataka između odredišta.

Navigation Component biblioteka se sastoji od tri glavne komponente: navigacijski graf (engl. n*avigation graph*), kontejner navigacije (engl. *NavHost*), kontroler navigacije (engl. *NavController*).

Navigacijski graf je komponenta koja sadrži sve informacije vezane uz navigaciju na jednom centraliziranom mjestu. Sadrži sve destinacije i putanje kojima korisnik može navigirati kroz aplikaciju.

NavHost je prazan kontejner koji prikazuje destinacije s navigacijskog grafa.

NavController je objekt koji prati trenutnu poziciju destinacije unutar navigacijskog grafa. Omogućuje zamjenu destinacija dok se korisnik kreće kroz aplikaciju odnosno kroz navigacijski graf.

3.7. Coroutines

Ideja korutina (engl. *coroutines*) [13] je da se više funkcija može izvoditi paralelno bez da jedna utječe na drugu odnosno da ne dolazi do blokiranja. Također korutine pomažu da se neka akcija koja se dugo izvodi (engl. *long running task*) obavi izvan glavne niti (engl. *main thread*) kako ne bi došlo do blokiranja aplikacije.

Najčešće akcije koje se dugo izvode su: mrežni zahtjev (engl. *network request*), interakcija s bazom podataka, čitanje sadržaja datoteke, učitavanje slike s diska i slično. Navedene akcije je potrebno izvoditi izvan glavne niti, a jedan od načina je pomoću povratnih poziva (engl. *callback*). Ali pomoću povratnih poziva se stvara duplicirani kod, npr. jedan mrežni zahtjev ovisi o drugom i tu dolazi do ugnježdivana povratnih poziva te kod postaje nepregledan. Zbog toga je dobra praksa koristiti korutine koje dosta pojednostavljuju kod.

3.8. *Flow*

U korutinama, tok (engl. *flow*) [14] je tip koji može emitirati (engl. *emit*) više vrijednosti sekvencijalno, za razliku od *suspend* funkcija koje vraćaju samo jednu vrijednost. Emitirane vrijednosti moraju biti istog tipa.

Flow može upravljati tokovima (engl. *streams*) vrijednosti i transformirati podatke u složenom više-nitnom (engl. *multi-threaded*) načinu.

U tokove podataka uključena su tri entiteta:

- Proizvođač (engl. *producer*) proizvodi podatke koji se dodaju toku podataka. Zahvaljujući korutinama, *flow* može proizvoditi podatke asinkrono.
- Posrednik (engl. *intermediaries*) može modificirati odnosno izmijeniti svaku emitiranu vrijednost u tok podataka ili izmijeniti sam tok podataka.
- Potrošač (engl. *consumer*) koristi vrijednosti iz toka.

Flow se može koristiti za primanje najsvježijih podataka iz baze podataka.

3.9. *Koin*

Koin [15] je biblioteka koja se koristi za ubrizgavanje ovisnosti (engl. *dependency injection*). *Koin* je napisan u Kotlinu, jednostavan je za učenje i dobro dokumentiran.

Osnove *Koina*

- Deklariranje modula – definira entitete koji će biti ubrizgani u nekom trenutku u aplikaciji.

```
val appModule = module {
    single {
        Gson()
    }
}
```

- Startanje Koin-a – u jednoj liniji pomoću *startKoin* funkcije kojoj se predaje aplikacijski kontekst (engl. *application context*) i lista modula koji će biti dostupni po potrebi.

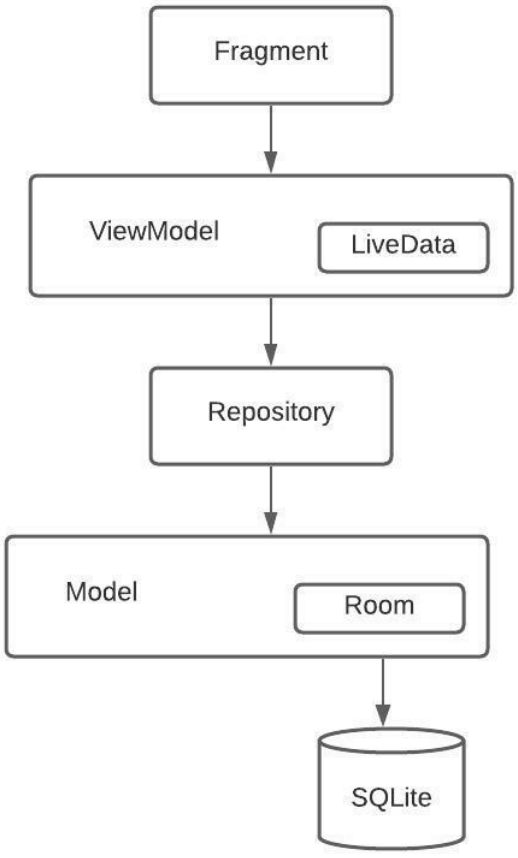
```
startKoin {
    androidContext(this@BusinessHelperApp)
    modules(
 listOf(presentationModule, databaseModule, repositoryModule,   appModule))
}
```

4. ANDROID APLIKACIJA ZA POMOĆ PRI PRAĆENJU VOĐENJA TVRTKE

U ovom poglavlju biti će objašnjena arhitektura aplikacije i baze podataka te struktura projekta.

4.1. Arhitektura aplikacije

U aplikaciji je implementiran MVVM arhitekturni obrazac, *repository pattern* i jedna aktivnost obrazac (engl. *single activity pattern*). Kao što je vidljivo na slici 4.1. svaka komponenta ovisi samo o komponenti koja je jednu razinu ispod. Fragment ovisi samo o ViewModelu. ViewModel može ovisiti o više *repository*-a ovisno o slučaju. U ovom slučaju *repository* ovisi samo o lokalnoj bazi podataka. Ovakav pristup arhitekturi aplikacije daje lakšu mogućnost testiranja izdvojenih komponenti aplikacije i poboljšava strukturu projekta.



Slika 4.1. Arhitektura aplikacije

ViewModel sadrži poslovnu logiku te pruža podatke za komponentu korisničkog sučelja, u ovom slučaju fragmentu. Prednost ViewModel-a je što ne zna ništa o komponentama korisničkog sučelja i preživljava promjenu konfiguracije.

```
class EmployeesViewModel(
    private val employeeRepository: EmployeeRepository
) : BaseViewModel() {

    val viewState: LiveData<ViewState> = getDataList {
        employeeRepository.getEmployees()
    }

    fun deleteEmployeeById(id: Long){
        viewModelScope.launch {
            employeeRepository.deleteEmployeeById(id)
        }
    }
}
```

Slika 4.2. ViewModel zaslona s listom zaposlenika

Repository rukuje s operacijama podataka – dohvaća podatke iz različitih izvora podataka te omogućuje ostatku aplikacije lako dohvaćanje potrebnih podataka. *Repository* se može smatrati kao posrednik između različitih izvora podataka, poput baze podataka i web usluga. Ukoliko je potrebno ažurirati podatke, *repository* zna odakle ih dohvatiti.

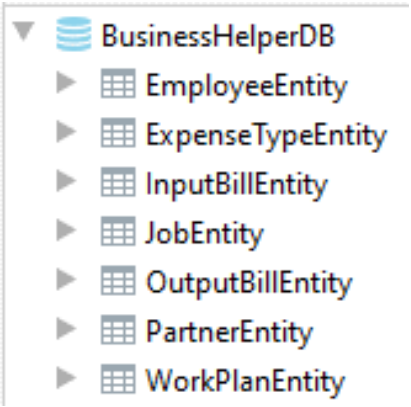
```
class EmployeeRepository(
    private val employeeDao: EmployeeDao
): BaseRepository() {

    suspend fun saveEmployee(employeeEntity: EmployeeEntity) =
        withContext(Dispatchers.IO){
            employeeDao.insertEmployee(employeeEntity)
        }

    fun getEmployees() = fetchListDataFromCache {
        employeeDao.getAllEmployees()
    }

    suspend fun getEmployeeById(id: Long) = fetchItemFromCache {
        employeeDao.getEmployeeById(id)
    }

    suspend fun deleteEmployeeById(id: Long){
        withContext(Dispatchers.IO){
            employeeDao.deleteEmployeeById(id)
        }
    }
}
```

Slika 4.3. *Repository* zaposlenika

4.2. Arhitektura baze podataka

Kao što je vidljivo na slici 4.4. baza podataka se sastoji od sljedećih entiteta:

- zaposlenik (engl. *employee entity*) – sadrži podatke o zaposleniku kao što su id, ime i prezime, datum rođenja, radno mjesto, email, kontakt broj, opis i slika.
- vrsta troška (engl. *expense type entity*) – sadrži podatke o vrstu troška kao što su id, naziv i PDV.
- ulazni račun (engl. *input bill entity*) – sadrži podatke o ulaznom računu kao što su id, broj računa, ime dobavljača, datum računa, datum dospijeća računa, vrijednost, naziv vrste troška i PDV.
- posao (engl. *job entity*) – sadrži podatke o poslu kao što su id, ime, vrijednost, očekivani datum početka i završetka posla, adresa i opis.
- izlazni račun (engl. *output bill entity*) – sadrži sve podatke o izlazno računu kao što su id, broj računa, ime kupca, datum računa, datum dospijeća računa, vrijednost i PDV.
- partner (engl. *partner entity*) – sadrži podatke kao što su id, ime, OIB, adresa, kontakt broj i email.
- plan rada (engl. *work plan entity*) – sadrži podatke kao što su id, ime, očekivani datum početka posla, opis i je li posao završen.



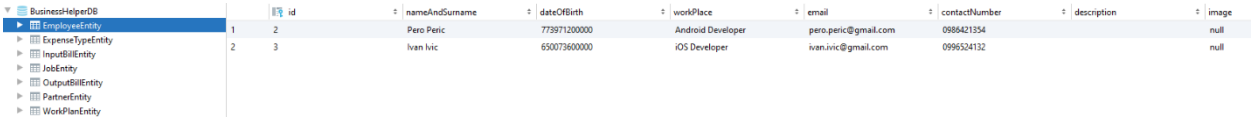
Slika 4.4. Arhitektura baze podataka

Za implementaciju baze podataka korištena je *Room* biblioteka koja se sastoji od tri glavne komponente: entitet, objekt pristupa podacima (DAO) i baza podataka. *Room* biblioteka je detaljnije objašnjena u pod poglavlju 3.5..

Na slici 4.5. prikazana je definicija entiteta zaposlenik u kodu, a na slici 4.6. tablica entiteta zaposlenika.

```kotlin
@Entity
data class EmployeeEntity(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0L,
    val nameAndSurname: String,
    val dateOfBirth: Long,
    val workPlace: String,
    val email: String,
    val contactNumber: String,
    val description: String,
    val image: Uri? = null
)
```

Slika 4.5. Entitet zaposlenik



Slika 4.6. Tablica entiteta zaposlenik

Na slici 4.7. prikazan je primjer DAO zaposlenika koji pruža metode kako bi ostatak aplikacije mogao spremati zaposlenike, dohvaćati sve zaposlenike, dohvaćati i brisati zaposlenike po id-u.

```kotlin
@Dao
interface EmployeeDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertEmployee(employeeEntity: EmployeeEntity)

    @Query("SELECT * FROM EmployeeEntity")
    fun getAllEmployees(): Flow<List<EmployeeEntity>>

    @Query("SELECT * FROM EmployeeEntity WHERE id= :id")
    suspend fun getEmployeeById(id: Long): EmployeeEntity

    @Query("DELETE FROM EmployeeEntity WHERE id= :id")
    suspend fun deleteEmployeeById(id: Long)
}
```

Slika 4.7. DAO zaposlenika

4.3. Struktura projekta

Kao što je vidljivo na slici 4.8. struktura projekta podijeljena je po paketima common, database, di, model, repositories i ui.

Coomon paket sadrži fajlove (engl. *file*) s konstantama i pomoćnim funkcijama, klasama.

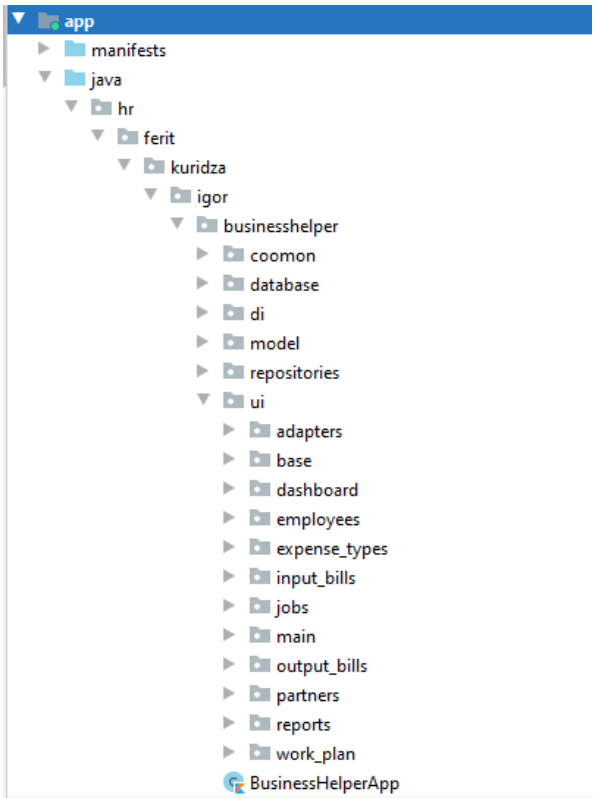
Database paket sadrži klase potrebne za interakciju s bazom podataka.

Di paket sadrži module u kojima se nalaze definicije komponenti koje će biti ubrizgane u nekom trenutku u aplikaciji.

Model paket sadrži modele poslovne logike.

Repositories paket sadrži sve repository klase.

Ui paket sadrži klase koje su vezane za korisničko sučelje – fragmenti, adapteri.. Ui paket sadrži pod pakete čiji naziv govori o kojem zaslonu/slučaju se radi.



Slika 4.8. Struktura projekta

5. IZGLED I KORIŠTENJE APLIKACIJE

U ovom poglavlju će biti prikazan izgled aplikacije bez unesenih podataka, prilikom unosa podataka i s unesenim podacima te navigacija kroz aplikaciju. U aplikaciji je omogućen tamni i svijetli način (engl. *dark and light theme*).

5.1. Bez unesenih podataka

Kao što je vidljivo na slikama 5.1. i 5.2. početni zaslon aplikacije se sastoji od 6 elemenata pomoću kojih se korisnik kreće kroz aplikaciju.

Slika 5.1. Svijetli način - početni zaslon

Slika 5.2. Tamni način - početni zaslon

Klikom na element „izlazni računi“ prikazuje se zaslon s listom izlaznih računa prikazan na slikama 5.3. i 5.4.

Slika 5.3. Svijetli način – izlazni računi

Slika 5.4. Tamni način – izlazni računi

Klikom na element „zaposlenici“ prikazuje se zaslon s listom zaposlenika prikazan na slikama 5.5. i 5.6.

Slika 5.5. Svijetli način – zaposlenici

Slika 5.6. Tamni način – zaposlenici

Klikom na element „ulazni računi“ prikazuje se zaslon s listom ulaznih računa prikazan na slikama 5.7. i 5.8.

Slika 5.7. Svijetli način – ulazni računi

Slika 5.8. Tamni način – ulazni računi

Klikom na element „plan rada“ prikazuje se zaslon s listom novih planova rada prikazan na slikama 5.9. i 5.10. Kliznim pomakom prsta po ekranu u lijevu stranu (engl. *swipe to left*) prikazuje se zaslon s listom završenih planova rada prikazan na slikama 5.11. i 5.12.

Slika 5.9. Svijetli način – novi planovi rada

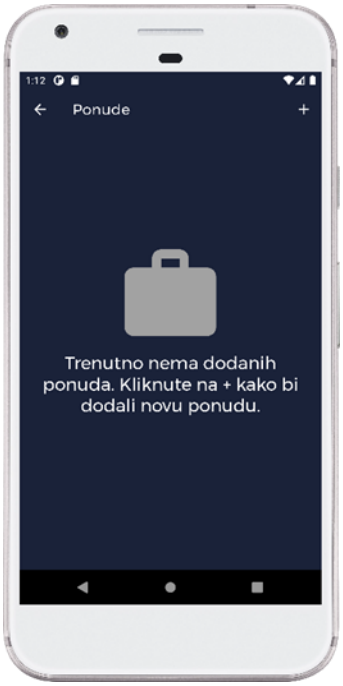
Slika 5.10. Tamni način – novi planovi rada

Slika 5.11. Svijetli način – završeni planovi rada

Slika 5.12. Tamni način – završeni planovi rada

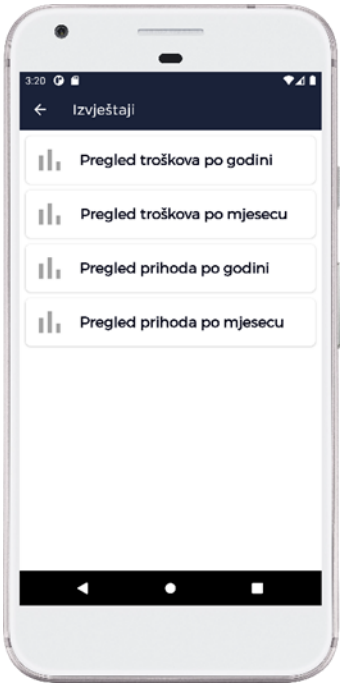
Klikom na element „ponude“ prikazuje se zaslon s listom ponuda prikazan na slikama 5.13. i 5.14.

Slika 5.13. Svijetli način – ponude

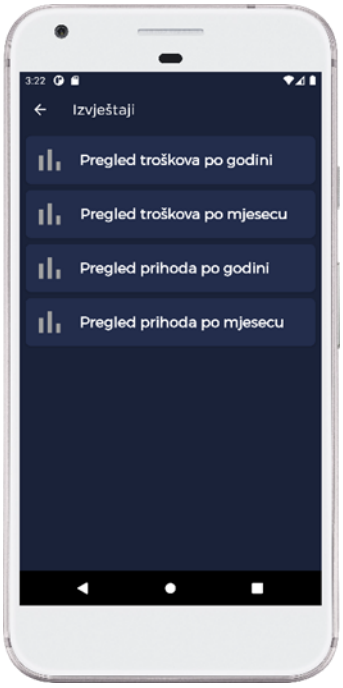


Slika 5.14. Tamni način – ponude

Klikom na element „izvještaji“ prikazuje se zaslon s mogućnosti odabira pregleda grafova troškova i prihoda po mjesecu i godini. Zaslon je prikazan na slikama 5.15. i 5.16.



Slika 5.15. Svijetli način – odabir pregleda

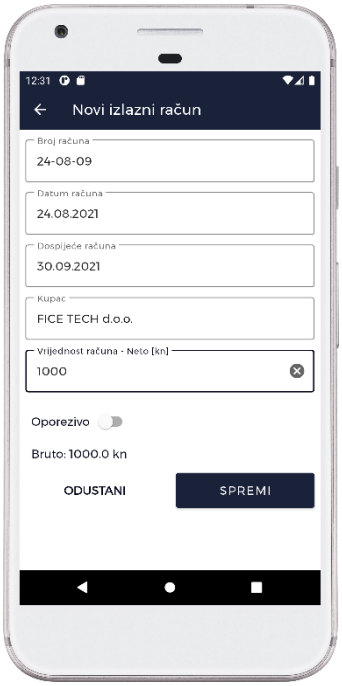


Slika 5.16. Tamni način – odabir pregleda

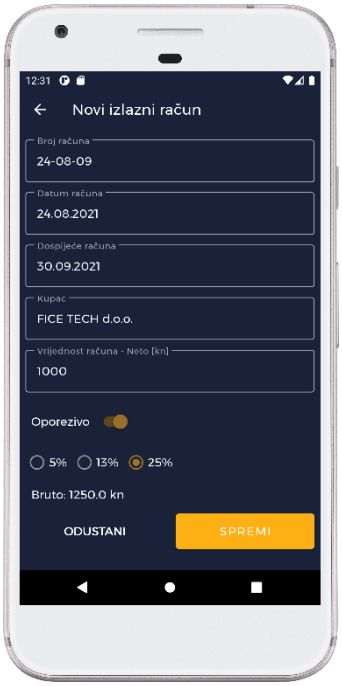
5.2. Unos podataka

Zasloni koji prikazuju listu podataka, klikom na + element otvara se zaslon za unos podataka.

Na slikama 5.17. i 5.18. prikazan je zaslon za unos podataka izlaznog računa. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti račun su: broj računa, datum računa, dospijeće računa, kupac, vrijednost računa (neto). Korisnik ima mogućnost odabira je li račun oporeziv ili ne, ako je oporeziv onda je potrebno odabrati radi li se o stopi od 5%, 13% ili 25%.

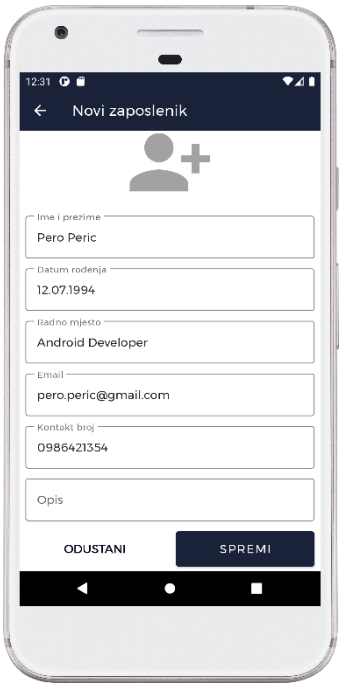


Slika 5.17. Svijetli način – novi izlazni račun

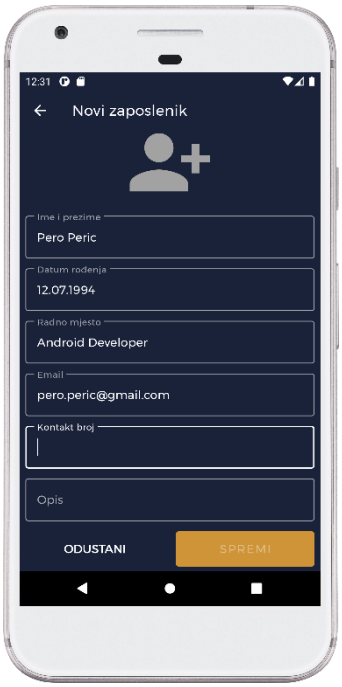


Slika 5.18. Tamni način – novi izlazni račun

Na slikama 5.19. i 5.20. prikazan je zaslon za unos podataka zaposlenika. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti zaposlenika su: ime i prezime, datum rođenja, radno mjesto, email, kontakt broj. Opcionalni podaci su opis zaposlenika i slika.



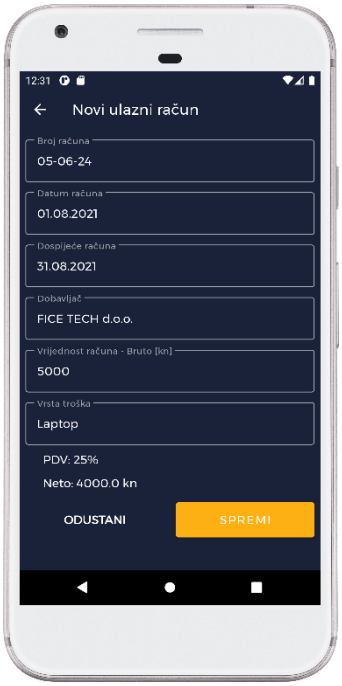
Slika 5.19. Svijetli način – novi zaposlenik



Slika 5.20. Tamni način – novi zaposlenik

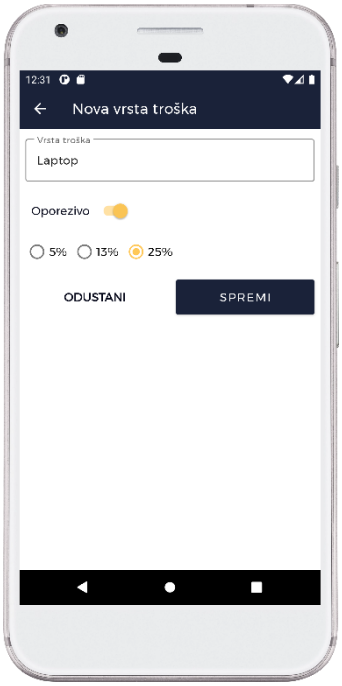
Na slikama 5.21. i 5.22. prikazan je zaslon za unos podataka ulaznog računa. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti račun su: broj računa, datum računa, dospijeće računa, dobavljač, vrijednost računa (bruto) i vrsta troška.

Slika 5.21. Svijetli način – novi ulazni račun

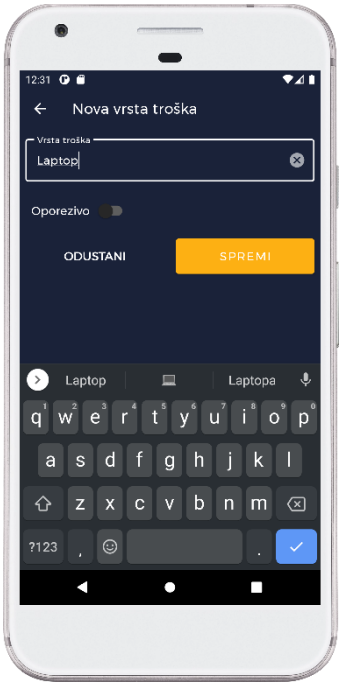


Slika 5.22. Tamni način – novi ulazni račun

Na slikama 5.23. i 5.24. prikazan je zaslon za unos podataka vrste troška. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti vrstu troška su: vrsta troška i ima mogućnost odabira je li račun oporeziv ili ne, ako je oporeziv onda je potrebno odabrati radi li se o stopi od 5%, 13% ili 25%.

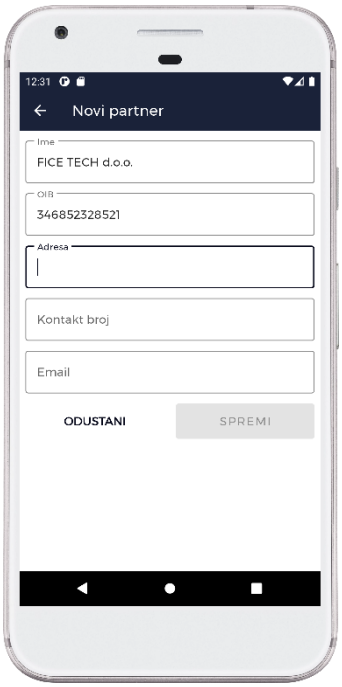


Slika 5.23. Svijetli način – nova vrsta troška

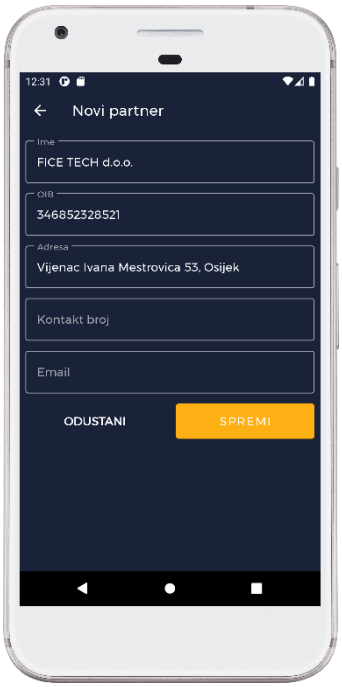


Slika 5.24. Tamni način – nova vrsta troška

Na slikama 5.25. i 5.26. prikazan je zaslon za unos podataka partnera. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti partnera su: ime, oib i adresa. Opcionalni podaci su kontakt broj i email.

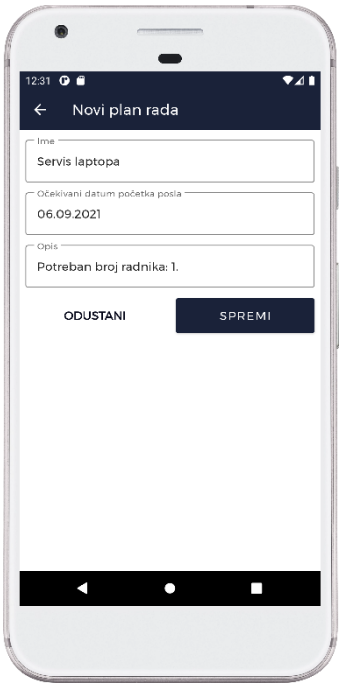


Slika 5.25. Svijetli način – novi partner



Slika 5.26. Tamni način – novi partner

Na slikama 5.27. i 5.28. prikazan je zaslon za unos podataka plana rada. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti plan rada su: ime, očekivani datum početka posla i opis.

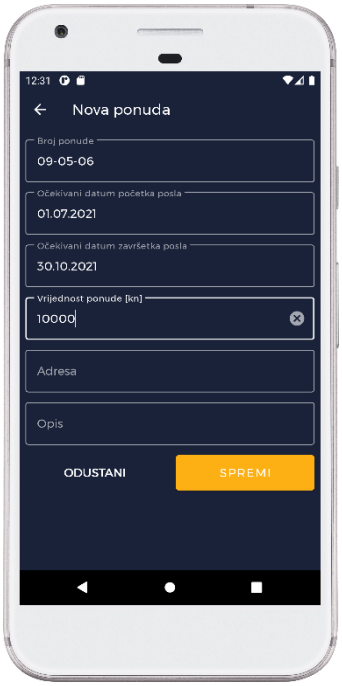


Slika 5.27. Svijetli način – novi plan rada

Slika 5.28. Tamni način – novi plan rada

Na slikama 5.29. i 5.30. prikazan je zaslon za unos podataka ponude. Potrebni podaci koje korisnik mora unijeti kako bi mogao spremiti ponudu su: broj ponude, očekivani datum početka i završetka posla i vrijednost ponude. Opcionalni podaci su opis ponude i adresa.

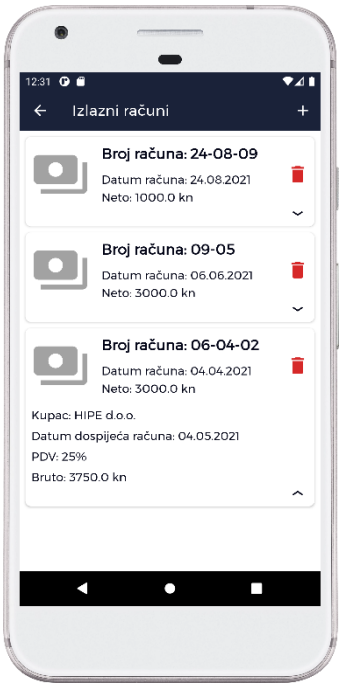
Slika 5.29. Svijetli način – nova ponuda



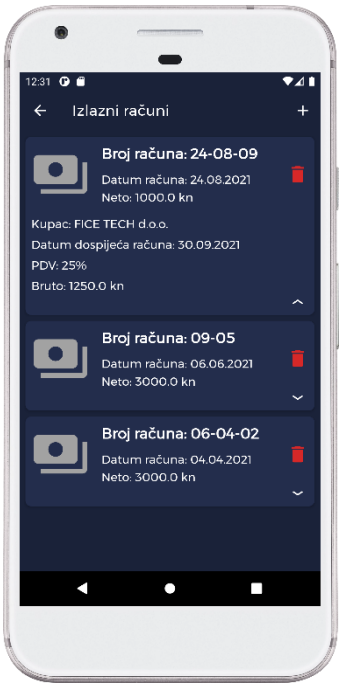
Slika 5.30. Tamni način – nova ponuda

5.3. S unesenim podacima

Na slikama 5.31. i 5.32. prikazan je zaslon s listom izlaznih računa. Korisnik ima mogućnost proširivanja kartice izlaznog računa kako bi dobio dodatne informacije te brisanje izlaznog računa.

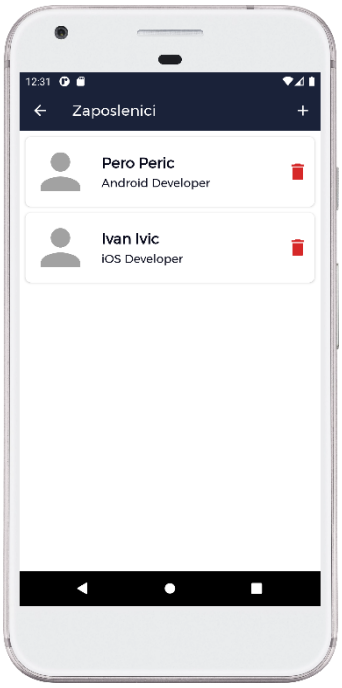


Slika 5.31. Svijetli način – izlazni računi

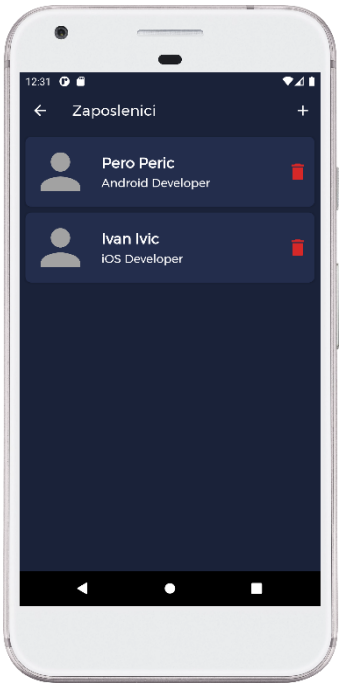


Slika 5.32. Tamni način – izlazni računi

Na slikama 5.33. i 5.34. prikazan je zaslon s listom zaposlenika. Korisnik ima mogućnost brisanja. Klikom na karticu, otvara se zaslon s detaljima zaposlenika.

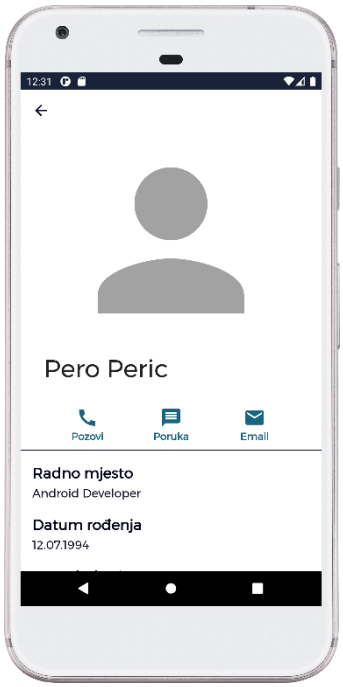


Slika 5.33. Svijetli način – zaposlenici

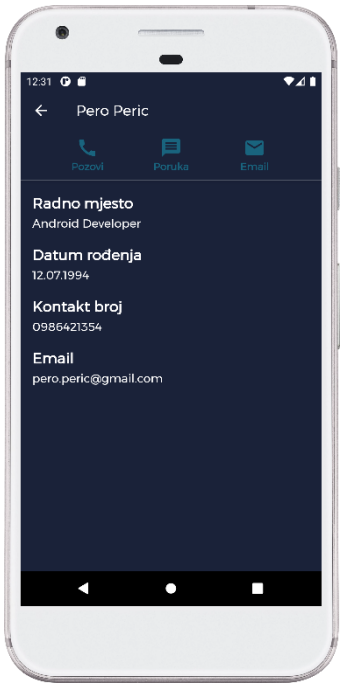


Slika 5.34. Tamni način – zaposlenici

Na slikama 5.35. i 5.36. prikazan je zaslon s detaljima zaposlenika. Klikom na „pozovi“ otvara se aplikacija za pozive s brojem zaposlenika. Klikom na „poruka“ otvara se aplikacija za slanje poruke zaposleniku. Klikom na „email“ otvara se aplikacija za slanje emaila zaposleniku.

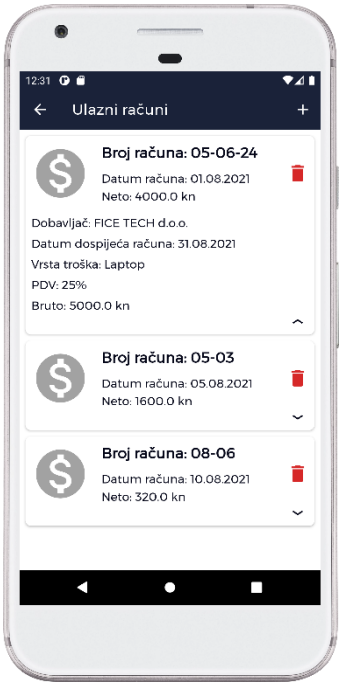


Slika 5.35. Svijetli način – detalji zaposlenika



Slika 5.36. Tamni način – detalji zaposlenika

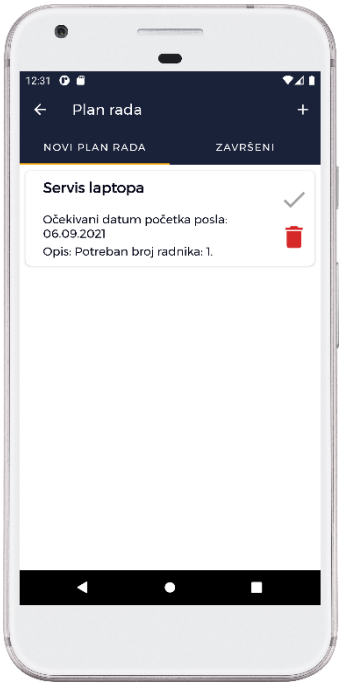
Na slikama 5.37. i 5.38. prikazan je zaslon s listom ulaznih računa. Korisnik ima mogućnost proširivanja kartice ulaznog računa kako bi dobio dodatne informacije te brisanje ulaznog računa.



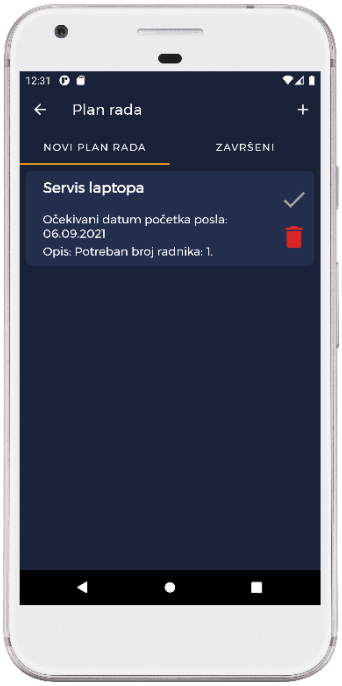
Slika 5.37. Svijetli način – ulazni računi

Slika 5.38. Tamni način – ulazni računi

Na slikama 5.39. i 5.40. prikazan je zaslon s listom novih planova rada. Korisnik ima mogućnost brisanja novog plana rada. Klikom na element kvačicu, plan rada prelazi u završene planove.

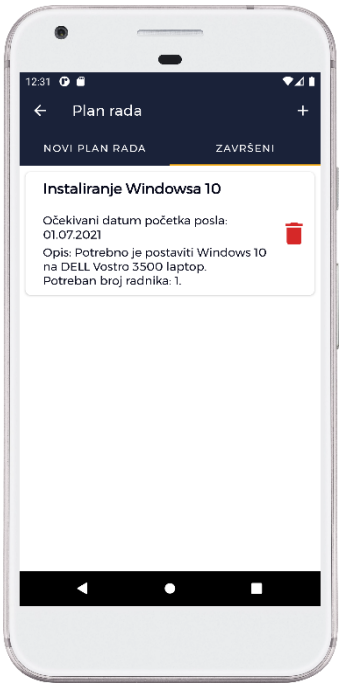


Slika 5.39. Svijetli način – novi planovi rada

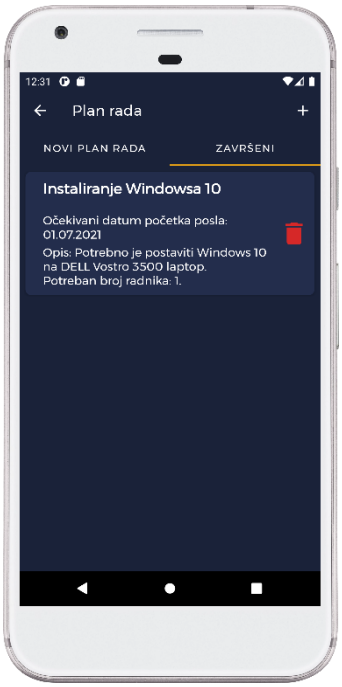


Slika 5.40. Tamni način – novi planovi rada

Na slikama 5.41. i 5.42. prikazan je zaslon s listom završenih planova rada. Korisnik ima mogućnost brisanja završenog plana rada.

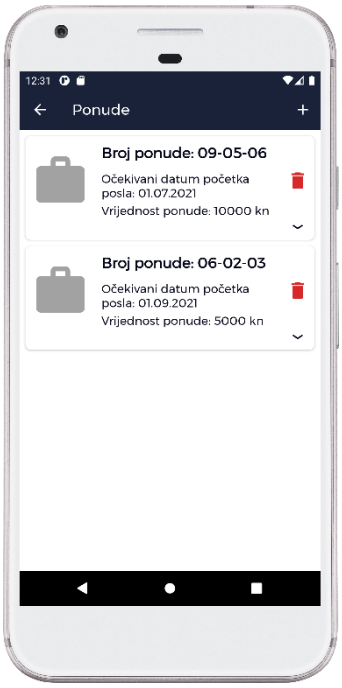


Slika 5.41. Svijetli način – završeni planovi rada

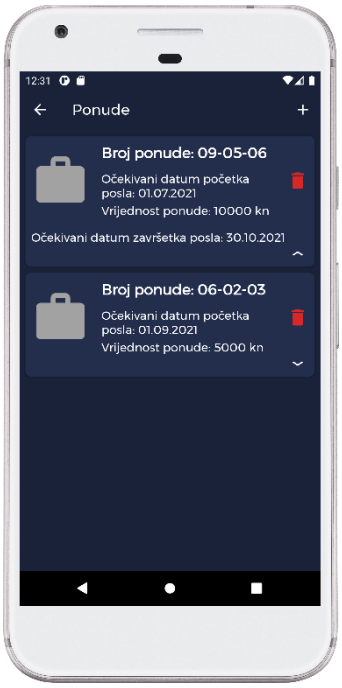


Slika 5.42. Tamni način – završeni planovi rada

Na slikama 5.43. i 5.44. prikazan je zaslon s listom ponuda. Korisnik ima mogućnost proširivanja kartice ponude kako bi dobio dodatne informacije te brisanje ponude.

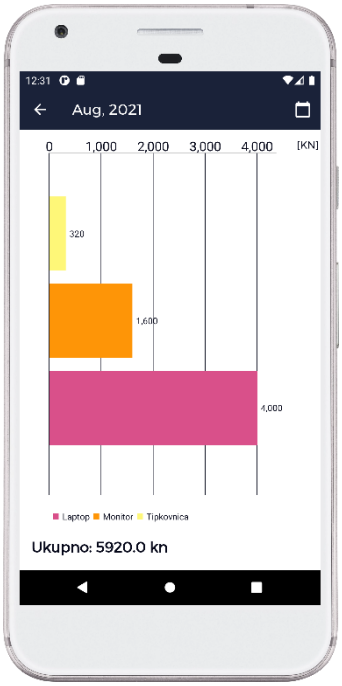


Slika 5.43. Svijetli način – ponude

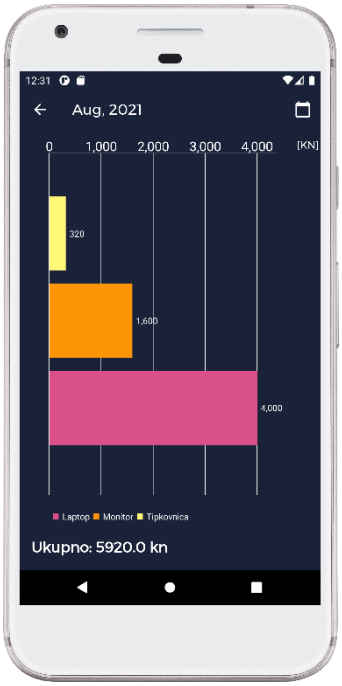


Slika 5.44. Tamni način – ponude

Na slikama 5.45. i 5.46. prikazan je zaslon s pregledom troškova po mjesecu. Troškovi su grupirani po vrsti troška. Korisnik ima mogućnost odabira godine i/ili mjeseca klikom na element kalendara.

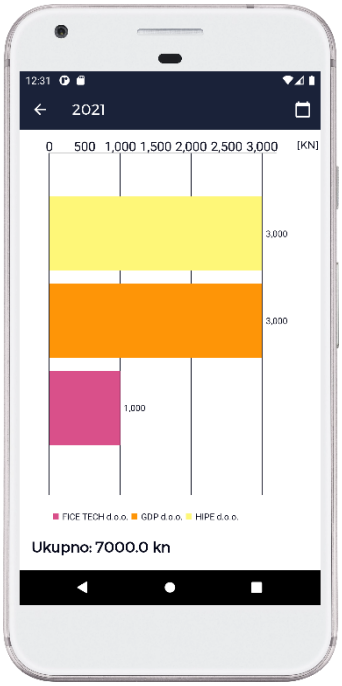


Slika 5.45. Svijetli način – mjesečni troškovi

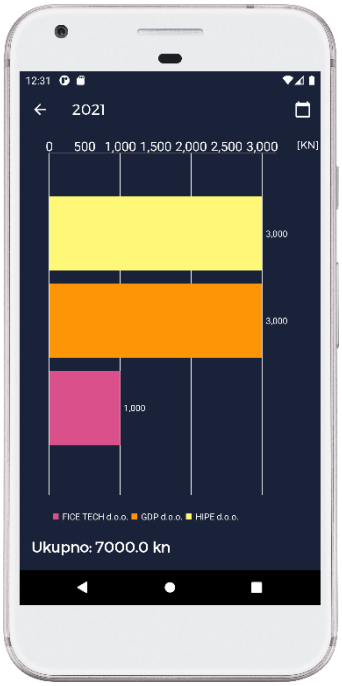


Slika 5.46. Tamni način – mjesečni troškovi

Na slikama 5.47. i 5.48. prikazan je zaslon s pregledom prihoda po godini. Prihodi su grupirani po kupcu. Korisnik ima mogućnost odabira godine i/ili mjeseca klikom na element kalendara.



Slika 5.47. Svijetli način – godišnji prihodi



Slika 5.48. Tamni način –godišnji prihodi

6. ZAKLJUČAK

U ovom diplomskom radu opisana je Android aplikacija za pomoć pri praćenju tvrtke. Objašnjene su korištene tehnologije i alati, arhitektura aplikacije te najvažnije komponente. Prikazan je izgled aplikacije bez unesenih podataka, prilikom unosa podataka i s unesenim podacima. U radu su korištene biblioteke koje se koriste u komercijalnim aplikacijama te su preporučene od Google-a. Uspješno je implementirana jedna od najpopularnijih arhitektura MVVM te *repository pattern*. U aplikaciji je omogućen tamni i svijetli način te se može pokretati na različitim veličinama pametnih telefona. Zadatak diplomskog rada je ispunjen.

Trenutna verzija Android aplikacije nudi mogućnost dodavanja računa poslovanja, vođenja evidencije o zaposlenicima, vođenja evidencije o ponudama, dodavanje bilješki poput plana rada, dodavanje troškova. Ima mogućnost grafičkog prikaza troškova i prihoda od poslova na mjesečnoj i godišnjoj razini.

Android aplikacija je otvorena za dodatna proširenja kako bi mogla konkurirati aplikacijama na tržištu koje pružaju pomoć pri praćenju tvrtke poput Android aplikacije „Moja Tvrtka“.

LITERATURA

[1] https://plaviured.hr/ulazni-racuni/ , pristupljeno 4.8.2021.

[2] https://www.relago.hr/RelaGO/BlogPost?id=141&nameForDisplay=2020-kako-napraviti-racun-fakturu, pristupljeno 4.8.2021.

[3] http://infos.hok.hr/faq/c_porezi_i_carine/c3_pdv/koje_su_stope_poreza_na_dodanu_vrijednost, pristupljeno 4.8.2021.

[4] https://developer.android.com/studio/intro, pristupljeno 5.8.2021.

[5] https://developer.android.com/kotlin/first, pristupljeno 5.8.2021.

[6] https://developer.android.com/jetpack/guide?gclsrc=aw.ds&gclid=Cj0KCQjwsZKJBhC0ARIsAJ96n3XPStRe4RkIrLgKRW2YfD7AlouSCN8mwuIRsXd52lp2kHscQbGH17caAkcaEALw_wcB , pristupljeno 6.8.2021.

[7] Gaurang Shaha, *Single Responsobility Principle,* https://proandroiddev.com/single-responsibility-principle-f806c88f4003, pristupljeno 8.8.2021.

[8] https://developer.android.com/jetpack/guide#recommended-app-arch, pristupljeno 9.8.2021.

[9] https://developer.android.com/topic/libraries/data-binding, pristupljeno 11.8.2021.

[10] https://developer.android.com/training/data-storage/room, pristupljeno 11.8.2021.

[11] https://developer.android.com/guide/navigation, pristupljeno 12.8.2021.

[12] https://developer.android.com/guide/navigation/navigation-pass-data, pristupljeno 12.8.2021.

[13] https://developer.android.com/kotlin/coroutines, pristupljeno 12.8.2021.

[14] https://developer.android.com/kotlin/flow#create, pristupljeno 13.8.2021.

[15] https://insert-koin.io/, pristupljeno 13.8.2021.

SAŽETAK

U ovom diplomskom radu obrađena je tema izrade Android aplikacije za pomoć pri praćenju tvrtke. Aplikacija korisniku omogućuje dodavanje računa poslovanja, vođenje evidencije o zaposlenicima, vođenje evidencije o ponudama, dodavanje bilješki poput plana rada, dodavanje troškova. Ima mogućnost grafičkog prikaza troškova i prihoda od poslova na mjesečnoj i godišnjoj razini. Za izradu diplomskog rada korišten je Kotlin programski jezik.

Ključne riječi: Android, Android Studio, Kotlin, tvrtka

Title: Android business tracking app

ABSTRACT

This master's thesis addresses the topic of creating Android application to track the company's business. The app provides the user to add business bills, keep records of employees, keep records of offers, add notes such as work plan, add expenses. Provides the ability to graphically display expenses and revenues on monthly and annual basis. For this master's thesis Kotlin programming language is used.

Keywords: Android, Android Studio, company, Kotlin

ŽIVOTOPIS

Igor Kuridža rođen 1. srpnja 1997. godine u Sisku, Hrvatska. 2004. godine započinje osnovnoškolsko obrazovanje u OŠ Mate Lovraka, Kutina. 2012. godine upisuje Tehničku Školu Kutina, smjer računarstvo. Nakon završenog srednjoškolskog obrazovanja, 2016. godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. 2019. godine završava Preddiplomski studij Računarstva te upisuje diplomski studij Računarstva, smjer Programsko inženjerstvo. Posjeduje diplomu *Android akademije* čiji su organizatori udruga *Osijek Software City* i tvrtka *COBE.* Tijekom studiranja odradio je stručnu praksu kao Android programer u tvrtki DICE d.o.o. gdje se i nastavio razvijati kao Android programer.