Android aplikacija za bilješke

Ojvan, Ivo Stjepan

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

Sveučilišni studij

ANDROID APLIKACIJA ZA BILJEŠKE

Završni rad

Ivo Stjepan Ojvan

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 08.07.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Ivo Stjepan Ojvan
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4545, 27.07.2020.
OIB Pristupnika:	04713843916
Mentor:	prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Android aplikacija za bilješke
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Opisati postupak izrade aplikacije za Android platformu pomoću koje registrirani korisnici mogu voditi bilješke. Opisati dizajn baze podataka koja će biti korištena uz aplikaciju. Aplikacija treba imati mogućnost pisanja novih bilješki, uređivanja postojećih i brisanja bilješki. Potrebno je omogućiti korisniku postavljanje notifikacije za bilješko (ako je potrebno) za neko određeno vrijeme. Tema rezervirana: Ivo
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	08.07.2023.
Datum potvrde ocjene od strane Odbora:	12.07.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 12.07.2023.

Ime i prezime studenta:	Ivo Stjepan Ojvan
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4545, 27.07.2020.
Turnitin podudaranje [%]:	15

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za bilješke**

izrađen pod vodstvom mentora prof. dr. sc. Krešimir Nenadić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Aplikacija za zapisivanje bilješki za Android operacijski sustav je aplikacija koja koristi tehnologije razvojnog okruženja Android Studio kao što su *IntelliJ IDEA, Android Emulator, Layout Editor, Debugger, Profiler* i drugi, te besplatnu verziju lokalne baze za pohranu podataka SQLite. Nudi usluge unosa novih bilješki, omogućuje nove izmjene na već unesenim bilješkama, njeno čitanje te brisanje same bilješke. Uz ostale mogućnosti omogućuje stavljanje bilješke na aplikaciju Google Calendar.

Ova Android aplikacija od drugih se razlikuje po mogućnosti spremanja bilješke na Google Calendar aplikaciju kao događaj, dok druge aplikacije nude mogućnosti rada više korisnika na istoj bilješci ili slanja same bilješke drugim korisnicima preko raznih servisa.

U drugom poglavlju opisane su već postojeće aplikacije za rad s bilješkama. U trećem poglavlju opisane su tehnologije korištene za razvoj aplikacije kao razvojno okruženje i programski jezik. U četvrtom poglavlju opisan je postupak izrade same aplikacije.

1.1. Tekst zadatka rada

Opisati postupak izrade aplikacije za Android platformu pomoću koje registrirani korisnici mogu voditi bilješke. Opisati dizajn baze podataka koja će biti korištena uz aplikaciju. Aplikacija treba imati mogućnost pisanja novih bilješki, uređivanja postojećih i brisanja bilješki. Potrebno je omogućiti korisniku postavljanje notifikacije za bilješke (ako je potrebno) za neko određeno vrijeme.

2. PREGLED POSTOJEĆIH PROGRAMSKIH RJEŠENJA

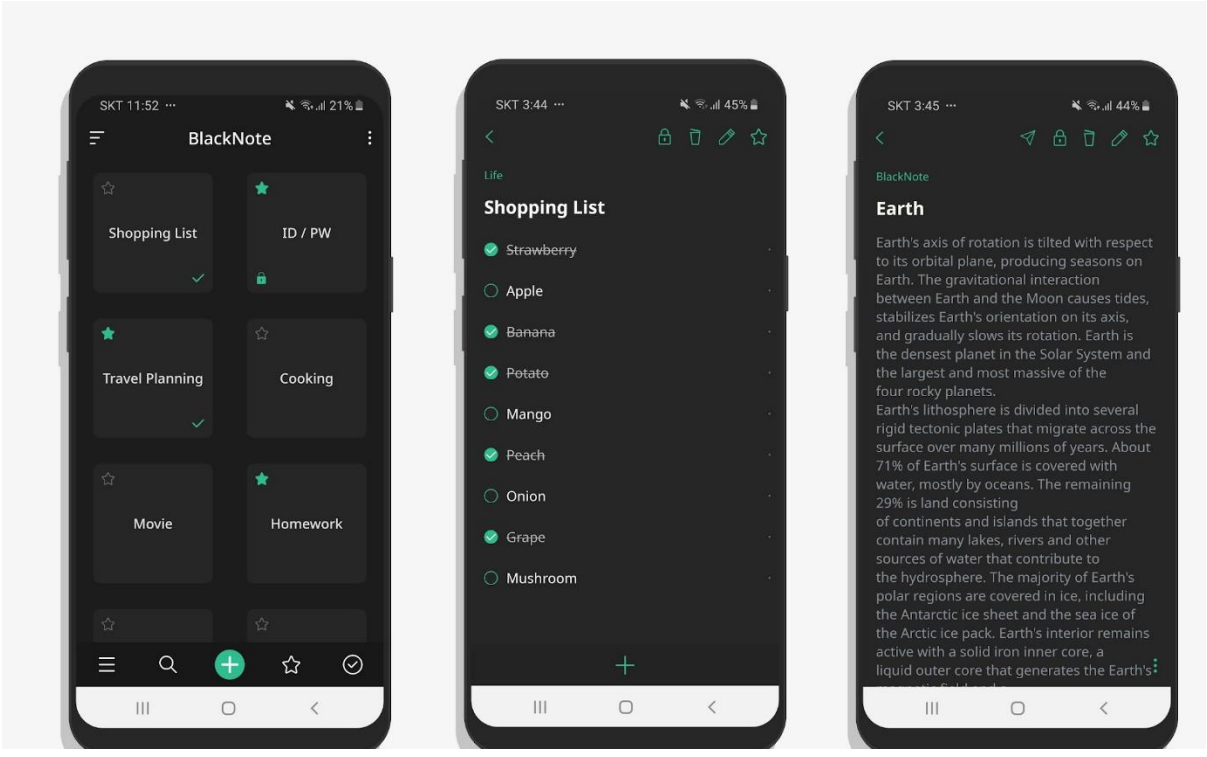
U ovo poglavlju navedene su i opisane druge Android aplikacije koje se bave problematikom pohranjivanja bilješki za kasniju uporabu.

2.1. BlackNote

BlackNote je besplatna Android aplikacija s preko sto tisuća recenzija i više od pet milijuna preuzimanja. Aplikacija omogućuje unos bilješki i spiskova, omogućuje pretraživanje putem naslova bilješke ili spiska, sortiranje po naslovu, datumu unosa, datumu izmjene ili prema važnosti korisniku. Nudi kreiranje kategorija te dijeljenje bilješki i spiskova s drugim osobama.

Prednosti ove aplikacije su uz mogućnost pohranjivanja bilješki i mogućnost popisa za kupnju, sortiranje bilješki i popisa prema raznim filterima te mogućnost zaključavanja bilješke ili popisa. Neodostatak bi bio mogućnost spremanja bilješke na kalendar.

Na slici 2.1. prikazano je korisničko sučelje aplikacije BlackNote.

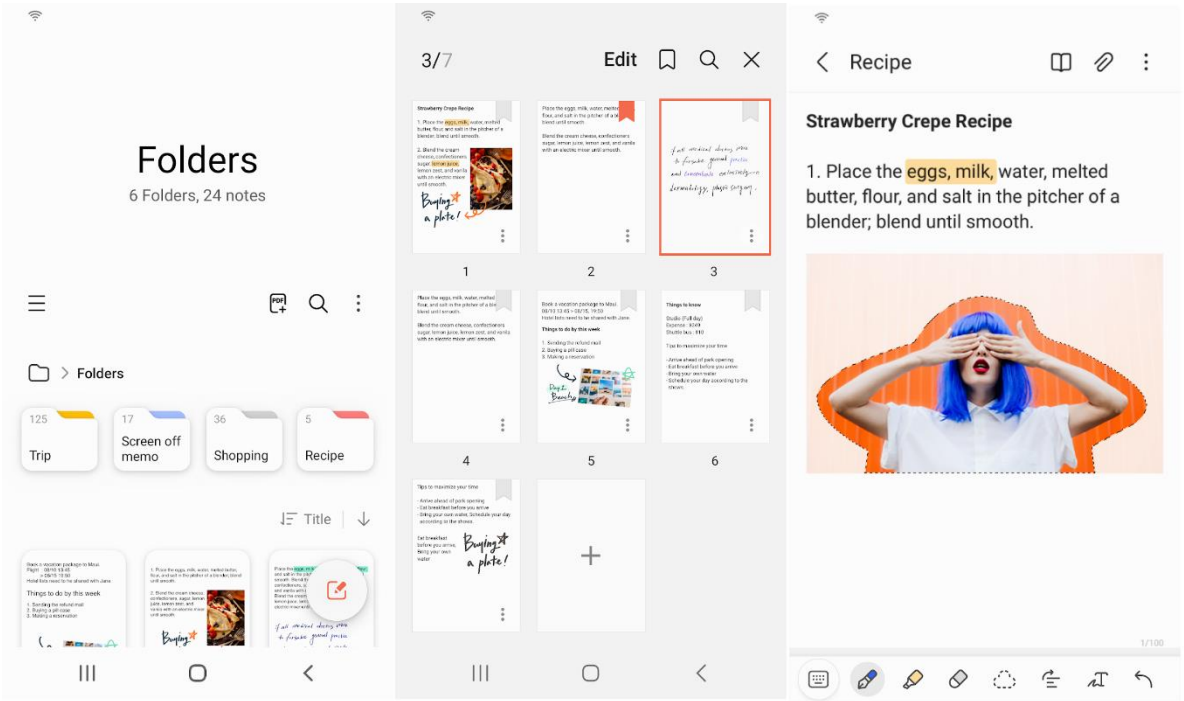


Slika 2.1. Sučelje aplikacije BlackNote
(aplikacija dostupna na: [1])

2.2. Samsung Notes

Samsung Notes je Android aplikacija koju je razvila tvrtka Samsung Electronics Co., Ltd. Aplikacija omogućuje kreiranje novih bilježaka, spremanje PDF (engl. *Portable Document Format*) datoteka kao pozadinu po kojoj se mogu zapisivati vlastita zapažanja. Omogućuje suradnju više korisnika na istoj bilješci, slanje bilješke na printanje ili spremanje same bilješke kao datoteku. Slika 2.2. prikazuje korisničko sučelje aplikacije i mogućnosti aplikacije Samsung Notes.

Prednost ove aplikacije je mogućnost spremanja bilješki u različite mape što omogućava bolji pregled, mogućnost pisanja vlastitih zapažanja po PDF datoteci. Neki od nedostataka ove aplikacije su ograničena integracija s drugim aplikacijama, nedostatak dostupnosti na drugim platformama.



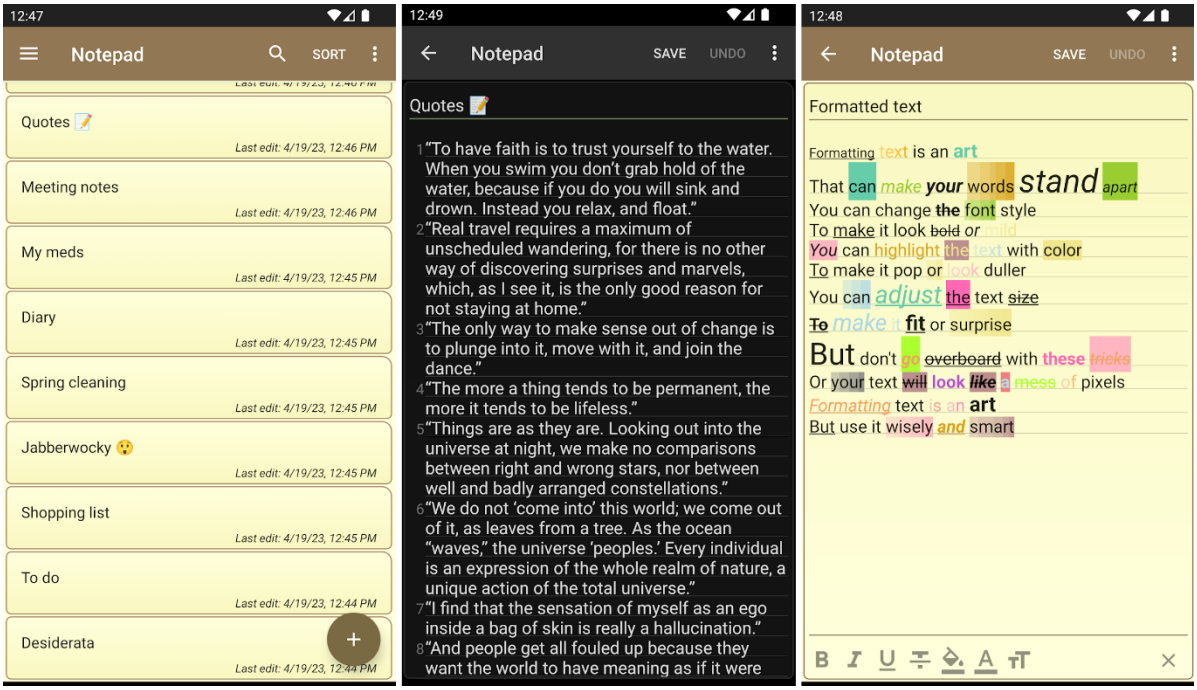
Slika 2.2. Sučelje aplikacije Samsung Notes
(aplikacija dostupna na: [2])

2.3. Notepad - simple notes

Notepad – simple notes aplikacija za Android operacijski sustav koju je izradio razvojni programer atomczak. Ovo je mala i brza aplikacija za pisanje bilješki. Ima jednostavno sučelje orijentirano lakšem korisničkom korištenju, omogućuje opciju uvoza bilješki iz tekstualnih datoteka te

spremanje bilješki kao tekstualnu datoteku i omogućuje dijeljenje bilješki s drugim aplikacijama. Na slici 2.3. prikazano je korisničko sučelje Notepad - simple notes aplikacije.

Nedostatak ove aplikacije je dizajn korisničkog sučelje koje nije ugodno oku, ali prednost ove aplikacije je mogućnost naprednog formatiranja teksta same bilješke.

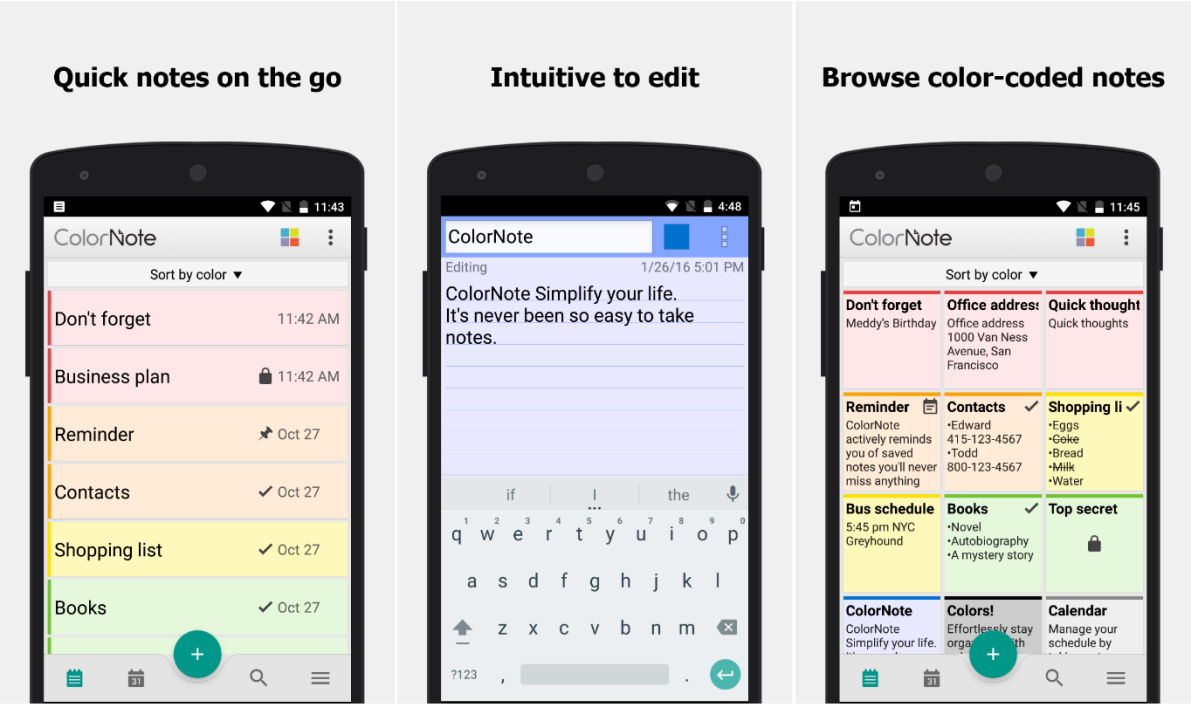


Slika 2.3. Sučelje aplikacije Notepad – simple notes
(aplikacija dostupna na: [3])

2.4. ColorNote Notepad Notes

ColorNote Notepad Notes je Android aplikacija koja omogućuje unos bilješki, spiskova te popisa za kupovinu. Omogućuje organiziranje bilješki po boji, organiziranje bilježaka prema kalendaru, izradu sigurnosnih kopija u unutarnju pohranu uređaja, podržava mrežno sigurnosno kopiranje i sinkronizaciju te dijeljenje bilješki putem SMS-a, e-pošte ili Twitter društvene mreže. Slika 2.4. prikazuje korisničko sučelje aplikacije ColorNote Notepad Notes.

Prednost ove aplikacije mogućnost promjene rasporeda prikaza bilješki te mogućnost označavanja bilješaka različitim bojama. Nedostatak je konstantno prikazivanje *Toast* poruka prilikom spremanja nove bilješke ili promjene na nekoj bilješci.

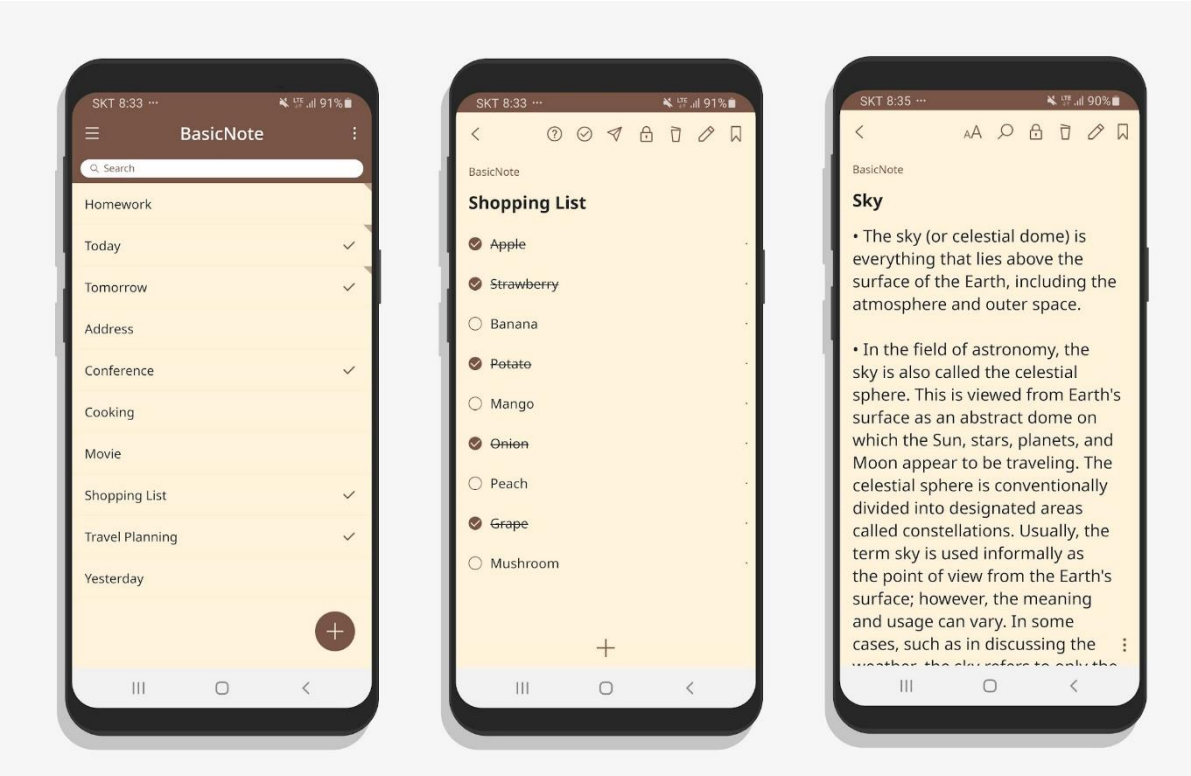


Slika 2.4. Sučelje aplikacije ColorNote Notepad Notes
(aplikacija dostupna na: [4])

2.5. BasicNote – Notes, Notepad

BasicNote aplikacija za spremanje bilješki s preko sedamdeset tisuća recenzija i preko pet milijuna preuzimanja, omogućuje unos bilješki i popisa za kupovinu te njihovo uređivanje, zaključavanje zaporkom ili otiskom prsta te dijeljenje s drugim osobama.

Nedostatak je nemogućnost spremanja bilješke u kalendar ili postavljanje bilo kakvog podsjetnika. Prednosti su mogućnost mijenjanja između svijetle i tamna teme aplikacije, spremanje bilješki i popisa za kupovinu i njeno jednostavno korisničko sučelje. Na slici 2.5. prikazano je korisničko sučelje aplikacije BasicNote - Notes, Notepad



Slika 2.5. Sučelje aplikacije BasicNote – Notes, Notepad
(aplikacija dostupna na: [5])

3. KORIŠTENE TEHNOLOGIJE I ZNAČAJKE APLIKACIJE

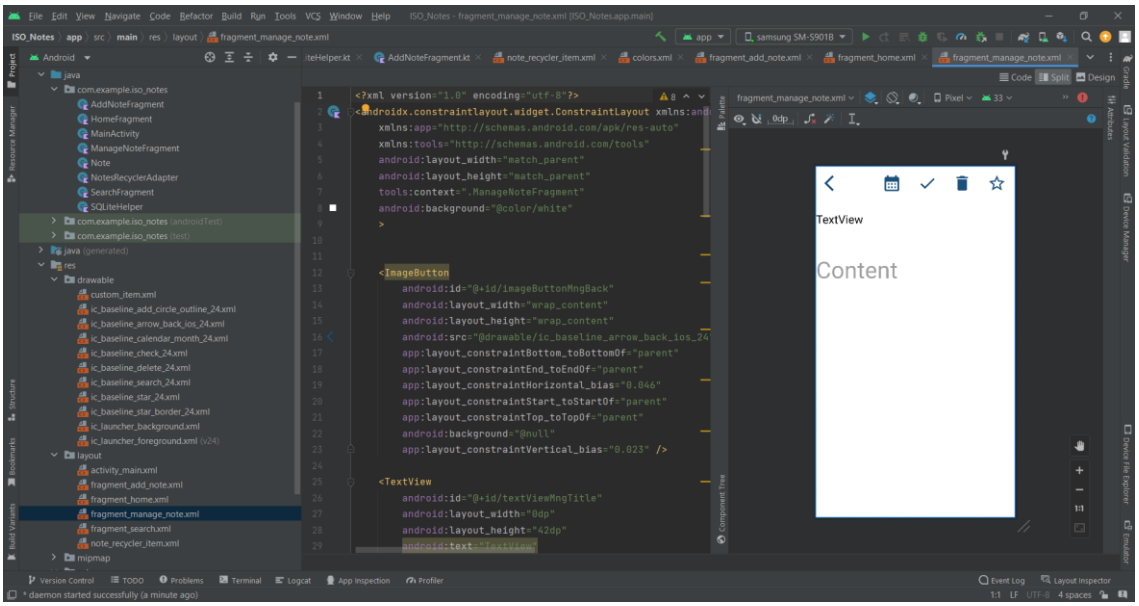
U ovom poglavlju opisane su tehnologije korištene u razvoju aplikacije kao razvojno okruženje i programski jezik. Također je opisan proces razvoja same aplikacije.

3.1. Opis korištenih tehnologija

Za izradu Android aplikacije koristilo se Android Studio razvojno okruženje, programski jezik Kotlin i jezik za označavanje podataka XML (engl. *eXtensible Markup Language*). Za omogućavanje pohrane podataka na lokalnoj bazi podataka koristila se SQLite baza podataka koja omogućava pohranu različitih tipova podataka u obliku zapisa u tablici.

3.1.1. Android Studio

Integrirano razvojno okruženje Android Studio, izgrađen je na JetBrains IntelliJ IDEA softveru i prema [6] služi za razvoj Android aplikacija. Neke od podržanih značajka su pametno uređivanje, napredno refaktoriranje koda te opcija automatskog nadopunjavanja. Dostupan je za operacijske sustave: Windows, macOS, Linux i Chrome OS. Umjesto programskog jezika Java, u svibnju 2019. godine, programski jezik Kotlin je postao preporučeni programski jezik za razvijanje Android aplikacija. Slika 3.1. prikazuje korisničko sučelje Android Studio razvojnog okruženja.



Slika 3.1. Korisničko sučelje Android Studio razvojnog okruženja

3.1.2. Kotlin programski jezik

Kotlin je višeplatformski programski jezik visoke razine opće namjene kako je navedeno u [7]. Dizajniran je za potpunu međuoperativnost s programskim jezikom Java. Izvorna namjena programskog jezika Kotlin je poboljšanje programskog jezika Java i često je korišten u kombinaciji s Java programskih jezikom. Među glavnim primjenama je razvoj Android aplikacija.

3.1.3. XML opisni jezik

XML opisni jezik je standardni jezik za označavanje podataka koji kao navedeno u [8] ima definirani set pravila za opisivanje, tj. kodiranje dokumenata u formatu čitljivom ljudima i računalu. Njegove glavne karakteristike su jednostavnost, generalizacija i upotrebljivost preko interneta. XML se sintaksno jako sliči sintaksi HTML-a (engl. *HyperText Markup Language*) iako se razlikuju u namjeni. Slika 3.2. prikazuje primjer XML sintakse.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeFragment"
    android:background="@color/white">

    <ImageButton
        android:id="@+id/imageButtonAddNote"
        android:layout_width="71dp"
        android:layout_height="60dp"
        android:background="@null"
        android:src="@drawable/ic_baseline_add_circle_outline_24"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.976" />
```

Slika 3.2. Primjer XML sintakse

3.1.4. SQLite

SQLite je softverska biblioteka koja pruža sustav za upravljanje relacijskim bazama podataka. Omogućava stvaranje, izmjenu i upravljanje bazama podataka kroz SQL (eng. *Structured Query Language*) upite. Prema [9] SQLite je najčešće korištena baza podataka na svijetu te je ugrađena

u sve mobilne telefone i većinu računala i dolazi u paketu s velikim brojem aplikacija. SQLite je *serverless* što znači da ne zahtjeva odvojeni server proces za rad već se direktno implementira u aplikaciju.

3.2. Razvoj aplikacije

3.2.1. Gradle

Prema [10] Gradle je napredni alat za izgradnju kojim se automatizira i upravlja proces izgradnje te dopušta definiranje prilagođenih konfiguracija izgradnje. Unutar gradle dijela definira se ciljana verzija Android operacijskog sustava te razne ovisnosti koje zahtijevaju različite biblioteke. Dio Gradle skripte prikazan je na slici 3.3..

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.iso_notes'
    compileSdk 33

    defaultConfig {...}

    buildTypes {...}
    compileOptions {...}
    kotlinOptions {jvmTarget = '1.8'}
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Slika 3.3. Prikaz gradle skripte

3.2.2. Android Manifest

Prema [11] datoteka *AndroidManifest.xml* sastavni je dio svakog Android projekta. Zadaća ove datoteke je deklarirati komponenete kao aktivnosti, usluge i pružatelje sadržaja, deklarirati

dopuštenja za pristup određenim resursima, deklariranje informacija o aplikaciji te deklariranje pokretačke aktivnosti. Ovoj aplikaciji potrebna je dozvola pristupa internetu zbog mogućnosti spremanje zabilješke u Google Calendar. Dio *AndroidManifest.xml* datoteke prikazan je na slici 3.4..

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="ISO_Notes"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ISO_Notes"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:windowSoftInputMode="adjustPan"
            android:screenOrientation="portrait"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Slika 3.4. Prikaz dijela AndroidManifest.xml datoteke

3.2.3. Aktivnost MainActivity

Prema [12] aktivnost je ključna komponenta aplikacije. Pri pokretanju aplikacije ova aktivnost se prva pokreće te zamjenjuje sadržaj FrameLayout komponente s početnim zaslonom aplikacije sadržanim unutar *Home Fragmenta*. Prikaz koda nalazi se na slici 3.5. u kojem se vidi kako se preko *supportFragmentManager* objekta obavlja transakcija zamjene sadržaja.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
            replace(R.id.fragmentContainer, HomeFragment())
            commit()
        }
    }
}
```

Slika 3.5. Prikaz koda koji zamjenjuje sadržaj

3.2.4. Implementacija unosa bilješki

Kako bi se unijela nova bilješka na početnom zaslonu aplikacije potrebno je pritisnuti dugme *Save* čime se prikazuje fragment preko kojeg se unosi naslov i sadržaj bilješke. Uvjeti koji moraju biti zadovoljeni pri unosu bilješke je da polja u koja se unose naslov i sadržaj ne smiju biti prazna, odnosno ne smije biti prazna bilješka. Pritiskom dugmeta *Spremi* provodi se proces provjere praznih polja te se u slučaju ispravnog korisničkog unosa nova bilješka sprema u bazu podataka. Slika 3.6. prikazuje kod kojim se bilješka sprema u bazu podataka.

```
btnSave.setOnClickListener{ it: View!
    val title = etTitle.text.toString()
    val text = etText.text.toString()

    if(title.isNotEmpty() && text.isNotEmpty()){
        val note = Note(id = sqLiteHelper.getNumberOfIstances()+1,title = title, text = text)
        val status : Long = sqLiteHelper.insertNote(note)

        if(status > -1){
            Toast.makeText(this.context, text: "Note added", Toast.LENGTH_LONG).show()
        }else{
            Toast.makeText(this.context, text: "Note not added", Toast.LENGTH_LONG).show()
        }
        changeFragment(HomeFragment())
    }else{
        Toast.makeText(context, text: "Empty fields", Toast.LENGTH_LONG).show()
    }
}
```

Slika 3.6. Prikaz koda za unos bilješke u bazu podataka

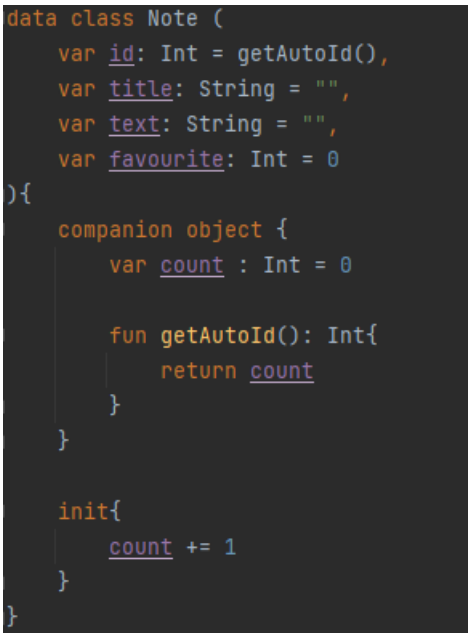
Ako provjera praznih polja utvrdi da obavezna polja nisu prazna poziva se *insertNote* metoda iz *SQLiteHelper* klase. Unutar te metoda krira se objekt tipa *SQLiteDatabase* koji predstavlja bazu s mogućnosti pisanja. U objekt tipa *ContentValues* pohranjuju se vrijednosti atributa prosljeđenog parametra tipa *Note* te se poziva metoda *insert* koja prima parametre: ime tablice, parametar koji provjera je li bar jedan stupac imenovan i objekt tipa *ContentValues* u kojem su pohranjene vrijednosti za unos u tablicu. Proces spremanja je prikazan na slici 3.7..

```
fun insertNote(note:Note) : Long{
    val db : SQLiteDatabase = this.writableDatabase

    val contentValues = ContentValues()
    contentValues.put(ID, note.id)
    contentValues.put(TITLE, note.title)
    contentValues.put(TEXT, note.text)
    contentValues.put(FAVOURITE, note.favourite)

    val success = db.insert(TBL_NOTES,  nullColumnHack: null, contentValues)
    db.close()

    return success
}
```

Slika 3.7. Prikaz koda metode za spremanje bilješke

3.2.5. Implementacija pregleda svih bilješki

Unutar *HomeFragment* fragmenta koji se prvi pokreće prilikom paljenja aplikacije nalazi se *RecyclerView* komponenta u kojoj se svaka bilješka prikazuje kao zasebni objekt. Pritiskom na neku od bilješki aktivira se fragment u kojemu je moguće daljnje uređivanje bilješke. Implementacija je ostvarena preko *NotesRecyclerAdapter* klase koji prima popis objekata *Note* tipa te definira kako će sadržaj biti prikazan unutar zasebne *note_recycler_item.xml* datoteke u kojoj je XML-om opisan izgled za prikaz pojedine bilješke. Slika 3.8. prikazuje kod podatkovne Note klase.



Slika 3.8. Prikaz Note podatkovne klase

Kod kojim *NotesRecyclerAdapter* odrađuje zadatak popunjavanja odgovarajućih komponenta odgovarajućim vrijednostima Note objekata prikazan je na slici 3.9..

```
class NotesRecyclerAdapter(val items : ArrayList<Note>) : RecyclerView.Adapter<RecyclerView.ViewHolder>() {

    private var onClickItem: ((Note) -> Unit)? = null

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
        return NoteViewHolder(
            LayoutInflater.from(parent.context).inflate(R.layout.note_recycler_item, parent, attachToRoot: false)
        )
    }

    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
        when(holder){
            is NoteViewHolder ->{
                holder.bind(position, items[position])
                holder.itemView.setOnClickListener{onClickItem?.invoke(items[position])}
            }
        }
    }

    override fun getItemCount(): Int {
        return items.size
    }

    fun setOnClickItem(callback: (Note)->Unit){
        this.onClickItem = callback
    }

    class NoteViewHolder(val view: View): RecyclerView.ViewHolder(view){
        private val title = view.findViewById<TextView>(R.id.textViewNoteTitle)
        private val imgBtnFavourite = view.findViewById<ImageButton>(R.id.imageButtonItemFav)

        fun bind(index: Int, note:Note){
            title.text = note.title
            if(note.favourite == 1){
                imgBtnFavourite.setImageResource(R.drawable.ic_baseline_star_24)
            }else{
                imgBtnFavourite.setImageResource(R.drawable.ic_baseline_star_border_24)
            }
        }
    }
}
```

Slika 3.9. Prikaz koda NotesRecyclerAdapter klase

Pri pokretanju aplikacije prilikom inicijalizacije *HomeFragment* fragmenta stvara se instanca klase koja komunicira s bazom podataka te se *RecyclerView* komponenti pridružuju adapter koji govori kako će se popunjavati komponenete pojedinog elementa i upravitelj rasporeda koji određuje orijentaciju prikazanih bilješki. Slika 3.9. prikazuje kod kojim se popunjava *RecyclerView* komponenta.

```
sqLiteHelper = SQLiteHelper(requireContext())

recyclerAdapter = NotesRecyclerAdapter(sqLiteHelper.getAllNotes())
notesRecycler.apply {  this: RecyclerView!
    layoutManager = GridLayoutManager(context,  spanCount: 2)
    adapter = recyclerAdapter
}
```

Slika 3.9. Prikaz koda koji popunjava RecylcerView komponentu

Dohvaćanje svih bilješki iz baze podataka ostvareno je getAllNotes metodom koja vraća objekt tipa *ArrayList<Note>* koji sadrži sve zapisane bilješke unutar tablice pomoću SQL upita. Slika 3.10. programsko rješenje ove funkcionalnosti.

```
fun getAllNotes():ArrayList<Note>{
    val noteList : ArrayList<Note> = ArrayList()
    val selectQuery = "SELECT * FROM $TBL_NOTES"
    val db : SQLiteDatabase = this.readableDatabase

    val cursor : Cursor?

    try{
        cursor = db.rawQuery(selectQuery,  selectionArgs: null)
    }catch (e:java.lang.Exception){
        e.printStackTrace()
        db.execSQL(selectQuery)
        return ArrayList()
    }

    var id: Int
    var title: String
    var text: String
    var favourite: Int

    if(cursor.moveToFirst()){
        do{
            id = cursor.getInt(cursor.getColumnIndex( columnName: "id"))
            title = cursor.getString(cursor.getColumnIndex( columnName: "title"))
            text = cursor.getString(cursor.getColumnIndex( columnName: "text"))
            favourite = cursor.getInt(cursor.getColumnIndex( columnName: "favourite"))

            val note = Note(id, title, text, favourite)
            noteList.add(note)
        }while(cursor.moveToNext())
    }

    return noteList
}
```

Slika 3.10. Prikaz koda *getAllNotes* metode

3.2.6. Pregled pojedine bilješke

Za pregled pojedine bilješke potrebno je pritisnuti jednu od bilješki prikazanih na početnom zaslonu aplikacije čime se podaci odabrane bilješke spremaju u *Bundle* objekt te se prosljeđuju *ManageNoteFragmentu* te se aktivira *ManageNoteFragment* fragment u kojem se proslijeđene vrijednosti spremaju u *Note* objekt te se komponente fragmenta popunjavaju odgovarajućim vrijednostima i na taj način postaju dostupne mogućnosti brisanja bilješke, izmjena sadržaja te dodavanje aplikacije na Google Calendar. Slika 3.11. prikazuje kod kojim se pristupa odabranoj bilješci.

```
recyclerAdapter.setOnClickItem {  it: Note
    val mangeNoteFragment = ManageNoteFragment()
    val bundle = Bundle()
    bundle.putInt("id", it.id)
    bundle.putString("title", it.title)
    bundle.putString("text", it.text)
    bundle.putInt("favourite", it.favourite)

    mangeNoteFragment.arguments = bundle
    val fragmentTransaction: FragmentTransaction? = activity?.supportFragmentManager?.beginTransaction()
    fragmentTransaction?.replace(R.id.fragmentContainer, mangeNoteFragment)
    fragmentTransaction?.commit()
}
```

Slika 3.11. Prikaz koda za pristup pojedinoj bilješci

Na slici 3.12. prikazan je kod koji popunjava komponente unutar *ManageNoteFragment* komponente odgovarajućim vrijednostima bilješke.

```
val id = arguments?.getInt( key: "id")
val title = arguments?.getString( key: "title")
val text = arguments?.getString( key: "text")
val favourite = arguments?.getInt( key: "favourite")

val note = Note()
if (id != null) {
    note.id = id
}
note.title = title.toString()
note.text = text.toString()
note.favourite = favourite!!.toInt()

tvTitle.text = note.title
etText.setText(note.text)

if(note.favourite == 1){
    imgBtnFav.setImageResource(R.drawable.ic_baseline_star_24)
}else{
    imgBtnFav.setImageResource(R.drawable.ic_baseline_star_border_24)
}
```

Slika 3.12. Prikaz koda koji popunjava komponente odgovarajućim vrijednostima

3.2.7. Implementacija izmjene sadržaja bilješke

Prilikom pritiska dugmeta s ikonom kvačice unutar *ManageNoteFragment* fragmenta provjerava se je li došlo do promjene sadržaja prije komuniciranja s bazom i spremanja novog sadržaja te povratak na početni zaslon aplikacije. Slika 3.13. prikazuje kod kojim se provjerava je li došlo do promjene prije spremanja.

```
imgBtnSave.setOnClickListener {  it: View!
    val newText = etText.text.toString()
    if(newText != note.text){
        note.text = etText.text.toString()

        sqLiteHelper.updateNote(note)
        changeFragment(HomeFragment())
    }
}
```

Slika 3.13. Prikaz koda za provjeru promjene sadržaja i njegovog spremanja

Ukoliko je došlo do promjene sadržaja poziva se *updateNote* metoda koja prima jedan parametar tipa *Note* te se unutar te metode kreira objekt koji predstavlja bazu podataka s mogućnosti pisanja. Stvara se novi objekt tipa *ContentValues* u koji se pohranjuju vrijednosti atributa parametra te se

poziva *update* metoda klase SQLiteDatabase koja prima parametre: ime tablice, objekt tipa *ContentValues* koji sadrži vrijednosti atributa ulaznog parametra i String parametar koji govori vrijednosti ID vrijednosti retka koji se ažurira. Kod metode *updateNote* prikazan je na slici 3.14..

```
fun updateNote(note:Note): Int{
    val db = this.writableDatabase

    val contentValues = ContentValues()
    contentValues.put(ID, note.id)
    contentValues.put(TITLE, note.title)
    contentValues.put(TEXT, note.text)
    contentValues.put(FAVOURITE, note.favourite)

    val success = db.update(TBL_NOTES, contentValues, whereClause: "id=${note.id}", whereArgs: null)
    db.close()
    return success
}
```

Slika 3.14. Prikaz koda *updateNote* metode

3.2.8. Implementacija brisanja bilješke

Za brisanje odabrane bilješke potrebno je pritisnuti *ImageButton* s ikonom koša za smeće unutar *ManageNoteFragment* fragmenta te će se izvršiti proces brisanja bilješke te povratak na početni zaslon aplikacije. Slika 3.15. prikazuje kod kojim se briše zabilješka.

```
imgBtnDelete.setOnClickListener { it: View!
    sqLiteHelper.deleteNoteById(note.id)
    changeFragment(HomeFragment())
}
```

Slika 3.15. Prikaz koda za brisanje bilješke

Brisanje ciljane bilješke ostvareno je pomoću *deleteNoteById* metode koja pomoću prosljeđenog parametra koji predstavlja ID vrijednost bilješke kreira objekt *SQLiteDatabase* s mogućnosti pisanja i objekt tipa *ContentValues* u koji će piti pohranjena vrijednost ID atributa bilješke te se poziva *delete* metoda klase SQLiteDatabase koja prima parametre: ime tablice i vrijednost ID retka bilješke koja se želi obrisati. Slika 3.16. prikazuje proces brisanja bilješke iz baze podataka.

```
fun deleteNoteById(id:Int): Int{
    val db = this.writableDatabase

    val contentValues = ContentValues()
    contentValues.put(ID, id)

    val success = db.delete(TBL_NOTES,  whereClause: "id=$id",  whereArgs: null)
    db.close()
    return success
}
```

Slika 3.16. Prikaz koda za brisanje bilješke iz baze podataka

3.2.9. Implementacija oznake važnosti bilješci

Za dodavanje oznake važnosti bilješci potrebno je pritisnuti gumb s ikonom zvijezde s bijelom ispunom te za uklanjanje oznake važnosti potrebno je pritisnuti gumb s ikonom plave zvijezde čime se odvija provjera je li bilješka već označena kao važna ili nije označena te se mijenja njeno stanje unutar baze podataka i ikona gumba sa zvijezdom. Slika 3.17. prikazuje kod kojim se dodaje ili uklanja oznaka važnosti bilješke.

```
imgBtnFav.setOnClickListener {  it: View!
    if(note.favourite == 0){
        note.favourite = 1
        imgBtnFav.setImageResource(R.drawable.ic_baseline_star_24)
    }
    else{
        note.favourite = 0
        imgBtnFav.setImageResource(R.drawable.ic_baseline_star_border_24)
    }
    sqLiteHelper.updateNoteFav(note)
}
```

Slika 3.17. Prikaz koda za dodavanje oznake važnosti

3.2.10. Implementacija dodavanja bilješke na Google Calendar

Za dodavanje bilješke na Google Calendar potrebno je pritisnuti gumb s ikonom kalendara unutar *MangeNoteFragment* fragmenta. Budući da je potrebno odabrati datum na koji se sprema bilješka potrebno je izračunati razliku od trenutnog do izabranog datuma u milisekundama jer takav format atributa koji se koristi za prosljeđivanje odabranog datuma unutar objekta tipa Intent. Računanja razlike između dva datuma ostvareno je ugrađenom metodom *getTimeInMillis* koja vraća vrijednost razlike u milisekundama. Nakon toga kreira se novi objekt tipa Intent u koji se

pohranjuju naslov i opis bilješke unutar atributa *TITLE* i *DESCRIPTION* objekta *CalendarContract* tipa metodom *setExtra* klase Intent, a odabrani datum se pohranjuje u atribute *EXTRA_EVENT_BEGIN_TIME* i *EXTRA_EVENT_END_TIME*. Za odabir datuma koristi se izbornik datuma te se pritiskom na OK gumb pokrene aplikacija Google Calendar s novim događajem koji će sadržavati naslov bilješke, njen sadržaj te datum odabran na izborniku za datum te je potrebno pritisnuti gumb spremi u aplikaciji Google Calendar čime će se spremiti novi događaj. Slika 3.18. prikazuje kod kojim se dodaje bilješka na Google Calendar u obliku događaja.

```
imgBtnCalendar.setOnClickListener {  it: View!
    val c = Calendar.getInstance()
    val year = c.get(Calendar.YEAR)
    val month = c.get(Calendar.MONTH)
    val day = c.get(Calendar.DAY_OF_MONTH)

    val dpd = DatePickerDialog(requireContext(), DatePickerDialog.OnDateSetListener{ view, mYear:Int, mMonth:Int, mDay:Int->
        Toast.makeText(context,  text: "Date: " + mDay.toString(), Toast.LENGTH_LONG).show()

        // Save on select date
        val beginTime = Calendar.getInstance()
        beginTime.set(mYear, mMonth, mDay,  hourOfDay: 12,  minute: 0)
        val startMillis = beginTime.getTimeInMillis()

        val endTime = Calendar.getInstance()
        endTime.set(mYear, mMonth, mDay,  hourOfDay: 12,  minute: 30)
        val endMillis = endTime.getTimeInMillis()

        val intent = Intent(Intent.ACTION_INSERT)
        intent.setData(CalendarContract.Events.CONTENT_URI)
        intent.putExtra(CalendarContract.Events.TITLE, note.title)
        intent.putExtra(CalendarContract.Events.DESCRIPTION, note.text)

        intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, startMillis)
        intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, endMillis)

        intent.putExtra(CalendarContract.Events.ALL_DAY,  value: true)

        startActivity(intent)

    }, year, month, day)
    dpd.show()
}
```

Slika 3.18. Prikaz koda za dodavanje bilješke na Google Calendar

3.2.11. Implementacija prikaza filtriranih bilješki

Prikaz filtriranih bilješki ostvaren je pritiskom dugmeta s ikonom zvijezde na početnom zaslonu aplikacije čime će se prikazivati samo bilješke sa statusom važnosti ili one bez ovisno o ikoni zvijezde na početnom zaslonu. Slika 3.19. prikazuje kod kojim se biraju zatim prikazuju samo bilješke označene kao važne.

```
imgBtnFavourites.setOnClickListener {  it: View!
    if(!fav){
        recyclerAdapter = NotesRecyclerAdapter(sqLiteHelper.getFavouriteNotes())
        notesRecycler.apply {  this: RecyclerView!
            layoutManager = GridLayoutManager(context,  spanCount: 2)
            adapter = recyclerAdapter
        }
        imgBtnFavourites.setImageResource(R.drawable.ic_baseline_star_24)
        recyclerAdapter.setOnClickItem {...}
    }else{
        recyclerAdapter = NotesRecyclerAdapter(sqLiteHelper.getAllNotes())
        notesRecycler.apply {  this: RecyclerView!
            layoutManager = GridLayoutManager(context,  spanCount: 2)
            adapter = recyclerAdapter
        }
        imgBtnFavourites.setImageResource(R.drawable.ic_baseline_star_border_24)
        recyclerAdapter.setOnClickItem {...}
    }
    fav = !fav
}
```

Slika 3.19. Prikaz koda za prikaz filtriranih bilješki

Dohvaćanje samo bilješki označenih kao važne ostvareno je pomoću *getFavouriteNotes* metode koja iz baze podataka dohvaća samo redove kojima je vrijednost stupca Favourite jednaka 1. Slika 3.20. prikazuje kod metode.

```
fun getFavouriteNotes(): ArrayList<Note>{
    val notes = getAllNotes()
    val favouriteNotes : ArrayList<Note> = ArrayList()

    for (i in notes.size-1 ≥ downTo  ≥ 0){
        if(notes[i].favourite == 1){
            favouriteNotes.add(notes[i])
        }
    }

    return favouriteNotes
}
```

Slika 3.20. Prikaz koda *getFavouriteNotes* metode

3.2.12. Implementacija pretraživanje bilješki prema naslovu

Pritiskom dugmeta s ikonom povećala na početnom zaslonu aplikacije pokreće se *SearchFragment* u kojem se unosom teksta naslova pri promjeni teksta prikazuju bilješke čiji naslov počinje s unesenim tekstom. Slika 3.21. prikazuje kod kojim se ostvaruje mogućnost pretraživanja spremljenih bilješki preko naslova.

```
etSearchTitle.addTextChangedListener(object : TextWatcher{
    override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {...}

    override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
        val title = etSearchTitle.text.toString()

        val notes = sqLiteHelper.getAllNotes()
        var targetNotes : ArrayList<Note> = ArrayList()

        for (i in notes.size-1 ≥ downTo ≥ 0){
            if(notes[i].title.startsWith(title)){
                targetNotes.add(notes[i])
            }
        }

        if(title == ""){
            targetNotes = ArrayList()
        }

        recyclerAdapter = NotesRecyclerAdapter(targetNotes)
        searchRecycleView.apply { this: RecyclerView!
            layoutManager = GridLayoutManager(context, spanCount: 2)
            adapter = recyclerAdapter
        }

        recyclerAdapter.setOnClickItem {...}
    }
    override fun afterTextChanged(s: Editable?) {...}
})
```

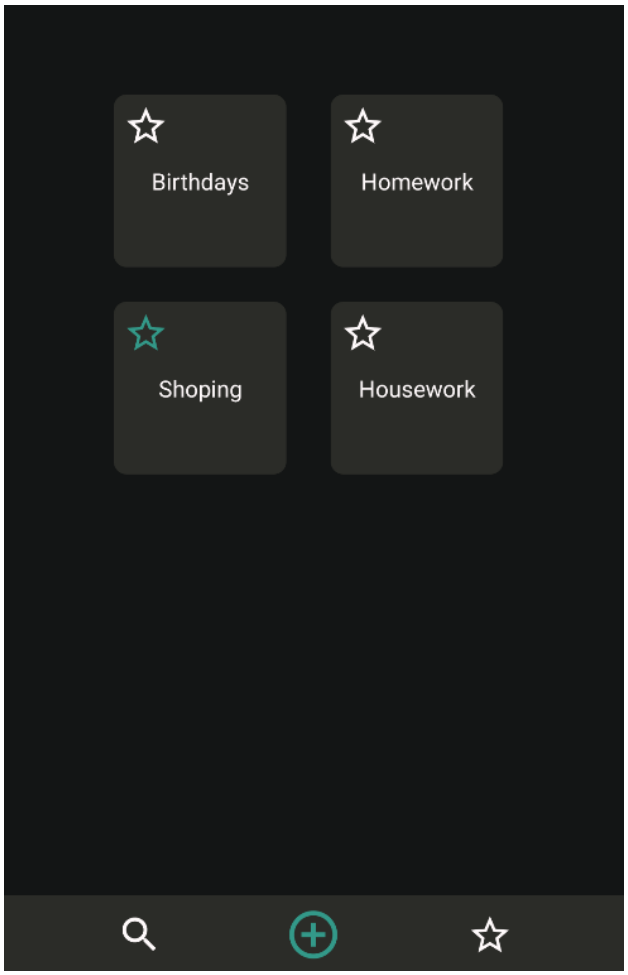
Slika 3.21. Prikaz koda za pretraživanje bilješki

4. ZASLONI I FUNKCIONALNOSTI APLIKACIJE

U ovom poglavlju opisano je i prikazano korisničko sučelje Android aplikacije za spremanje bilješki.

4.1. Početni zaslon

Početni zaslon sastoji se od popisa spremljenih bilješki i tri dugmeta koji omogućuju dodavanje novih bilježaka, pretraživanje bilješki putem naslova te prikaz bilješki s oznakom važong statusa. Bilješke koje sadržavaju zelenu zvijezdu su bilješke koje su označene kao važne bilješke. Slika 4.1 prikazuje početni zaslon aplikacije za bilješke.



Slika 4.1. Prikaz početnog zaslona aplikacije

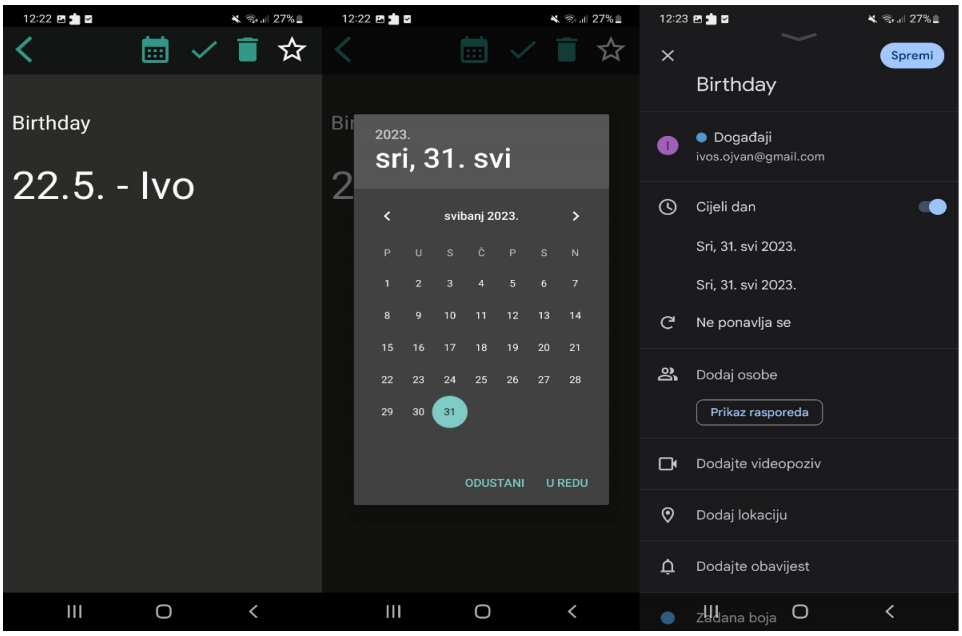
4.2. Zaslon za dodavanje nove bilješke

Ovaj zaslon sadrži dva zasebna područja u koje se može unositi tekst za naslov bilješke, a drugi za sadržaj te dugme koje omogućava spremanje nove bilješke te povrat na početni zaslon aplikacije. Slika 4.2. prikazuje zaslon preko kojeg se unosi nova bilješka.

Slika 4.2. Prikaz zaslona za dodavanje bilješke

4.3. Zaslon za uređivanje bilješke

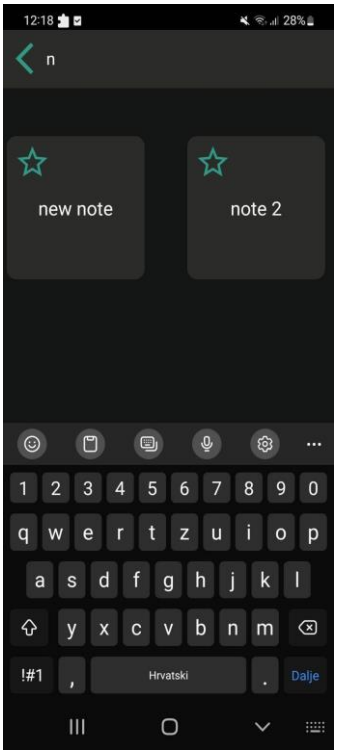
Zaslon prikazan na slici 4.3. prikazuje naslov i sadržaj bilješke, te omogućava opcije brisanja bilješke, uređivanje sadržaja, dodavanje oznake važnosti te dodavanje bilješke na Google Calendar.



Slika 4.3. Prikaz zaslona za uređivanje bilješke

4.4. Zaslon za pretraživanje bilježaka

Slika 4.4. sadrži komponentu za unos naslova te se automatski pri unosu teksta prikazuju sve bilješke čiji naslov započinje unesenim tekstom.



Slika 4.4. Prikaz zaslona za pretraživanje bilježaka

5. ZAKLJUČAK

Razvojem tehnologija i njihovom pristupačnosti više nije potrebno zapamtiti veliku količinu informacija napamet. Mogućnost zapisivanja informacija koje se mogu bilo kada ponovno pročitati i izmijeniti je razlog za kreiranja ovakve aplikacije koja svim uzrastima pruža te usluge.

Cilj ovog završnog rada je razvoj Android aplikacije koja će omogućiti lako i jednostavno spremanje informacija u tekstualnom obliku, a razvijena je pomoću Android Studio integriranog razvojnog okruženja. Aplikacija je napisanu u Kotlin objektno orijentiranom programskom jeziku, a dizajn je ostvaren pomoću XML jezika za opis podataka. Mogućnost spremanja podataka je ostvarena uz pomoć SQLite baze podataka. Funkcionalnosti zadane u tekstu zadatka rada su realizirane i potpuno funkcionalne uz jednostavno i intuitivno korisničko sučelje.

Aplikacija ima mogućnost daljnjeg razvoja ugradnjom novih funkcionalnosti kao na primjer slanje bilješki drugim osobama, dodavanjem mogućnosti spremanja bilježaka u mape po kategoriji i druge korisne nadogradnje.

LITERATURA

[1] BlackNote aplikacija, dostupna na:

https://play.google.com/store/apps/details?id=notepad.note.notas.notes.notizen

[pristupano svibanj 2023.]

[2] Samsung Notes aplikacija, dostupna na:

https://play.google.com/store/apps/details?id=com.samsung.android.app.notes

[pristupano svibanj 2023.]

[3] Notepad – Simple notes aplikacija, dostupna na:

https://play.google.com/store/apps/details?id=com.atomczak.notepat

[pristupano svibanj 2023.]

[4] ColorNote Notepad Notes aplikacija, dostupna na:

https://play.google.com/store/apps/details?id=com.socialnmobile.dictapps.notepad.color.note

[pristupano svibanj 2023.]

[5] BasicNote – Notes, Notepad aplikacija, dostupna na:

https://play.google.com/store/apps/details?id=notizen.basic.notes.notas.note.notepad

[pristupano svibanj 2023.]

[6] X. Ducrohet, T. Norbye, K. Chou, Android Studio: An IDE built for Android, dostupno na:

https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html

[pristupano svibanj 2023.]

[7] Kotlin, dokumentacija je dostupna na:

https://kotlinlang.org/docs/android-overview.html [pristupano svibanj 2023.]

[8] XML, dokumentacija je dostupna na:

https://www.w3.org/TR/REC-xml/ [pristupano svibanj 2023.]

[9] SQLite, dokumentacija je dostupna na:

https://www.sqlite.org/index.html [pristupano svibanj 2023.]

[10] Gradle, dokumentacija je dostupna na:

https://developer.android.com/build/releases/gradle-plugin [pristupano lipanj 2023.]

[11] Android Manifest, dokumentacija je dostupna na:

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

[pristupano lipanj 2023.]

[12] Activity, dokumentacija je dostupna na:

https://developer.android.com/guide/components/activities/intro-activities

[pristupano lipanj 2023.]

SAŽETAK

Ovaj završni rad prikazuje razvoj Android aplikacije za bilješke razvijene u Android Studio razvojnom okruženju. Aplikacija je pisana u Kotlin programskom jeziku visoke razine, a dizajn je ostvaren upotrebon XML jezika za opis podataka. Aplikacija omogućuje lako i jednostavno unošenje bilježaka u tekstualnom formatu pomoću SQLite baze podataka, te omogućuje njeno mijenjanje i dodavanje na Google Calendar u obliku događaja. U drugom poglavlju opisane su još neke aplikacije koje omogućuju slične mogućnosti kao ova aplikacija, te su kroz ovaj rad opisane glavne funkcionalnosti aplikacije te je prikazan kod koji to omogućuje.

Ključne riječi: Android, aplikacija za bilješke, Kotlin, SQLite

ABSTRACT

Android App for Notes

This paper shows the development of the Android application for notes developed in the Android Studio development environment. The application is developed in Kotlin high-level programming language and the design is achieved by using XML language for data description. This application allows you to easily enter notes in text format using the SQLite database, and allows you to change and add them to Google Calendar in the form of an event. The second chapter describes some more applications that allow similar features to this application, and through this paper the main functionalities of the application are described and a code that allows it is displayed.

Keywords: Android, Kotlin, note application, SQLite

ŽIVOTOPIS

Ivo Stjepan Ojvan rođen je 22.05.2001. godine u gradu Osijeku, Hrvatska. U gradu Osijeku pohađao je i završio Osnovnu Školu Ljudevita Gaja. Nakon završene osnovne škole, 2016. godine upisuje smjer tehničar za računalstvo Elektrotehničke i prometne škole Osijek. Godine 2020. završava svoje srednjoškolsko obrazovanje te upisuje preddiplomski sveučilišni studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.