

Univerzitet u Kragujevcu
Fakultet inženjerskih nauka



Seminarski rad iz predmeta Programiranje mobilnih aplikacija

Tema:
Android elektronski školski dnevnik

Student:
Jovan Bogdanović 609/2017

Predmetni profesor:
Prof. Dr. Nenad Grujović
Predmetni saradnik:
Vukašin Slavković

Kragujevac 2021.

Sadržaj

1. Uvod	2
2. Opis sistema	3
3. Izgled aplikacije i korišćenje	4
3.1. Administrator interfejs	5
3.2. Nastavnik interfejs	8
3.3. Posetilac interfejs	12
4. Implementacija i opis delova koda	13
4.1. Baza podataka	15
4.1.1. Tabela nastavnik	16
4.1.2. Tabela učenik	18
4.1.3. Tabela predmet	19
4.1.4. Tabela ocene	20
4.2. Java klase	23
4.2.1. MainActivity	23
4.2.2. LoginActivity	23
4.2.3. AdminActivity	27
4.2.4. AdminNastavnikActivity	27
4.2.5. PredmetiActivity	31
4.2.6. NastavnikActivity	33
4.2.7. UceniciActivity	34
4.2.8. OceneActivity	35
4.2.9. PosetilacActivity	37
4.2.10. CustomAdapter	39
5. Zaključak i dalji razvoj aplikacije	40
6. Literatura	41

1. Uvod

U ovom dokumentu biće pojašnjena android aplikacija za elektronski školski dnevnik i njeno korišćenje. Primarna ideja jeste da aplikacija omogući prikaz ocena elektronskim putem učeniku ili njegovim roditeljima, kako ne bi morali da troše vreme na dolazak u školu za potrebe toga.

Android je nastao 2005 godine. Razvila ga je isto imena firma. Android je prvenstveno radio na mobilnim telefonima dok je vremenom počeo da se koristi i na drugim uređajima kao što su tableti, uređaji koji se povezuju na televizore Android TV i igračke konzole.

Aplikacija je izrađena u okruženju Android Studio u programskom jeziku Java. Android Studio je oficijalni softver za Google-ove android operative sisteme, kreiran je na JetBrains-ovom IntelliJ IDEA softveru i dizajniran specijalno za razvoj android aplikacija. Android studio je objavljen 2013. godine i omogućava programiranje u jeziku Kotlin i Java.

U nastavku će biti objašnjene potrebe ovog sistema, opisani delovi programskog koda kao i način na koji je zamišljeno korišćenje aplikacije.

2. Opis sistema

Potreba ovog sistema je da pruži usluge **administratoru**, **nastavniku** i **posetiocu** (učeniku ili roditelju) koje su omogućene adekvatnim interfejsom i bazom podataka o kojoj ćemo reći nešto više kasnije.

Administratoru treba omogućiti upravljanje nad nastavnicima i predmetima, dakle kreiranje ili brisanje određenog nastavnika ili predmeta, pri kreiranju predmeta njemu se dodaje određeni nastavnik koji potom može da daje ocene iz istog.

Nastavniku treba omogućiti upis učenika u bazu podataka ili brisanje iz nje, upis ocene određenom učeniku iz određenog predmeta koji taj nastavnik predaje kao i ažuriranje ocena.

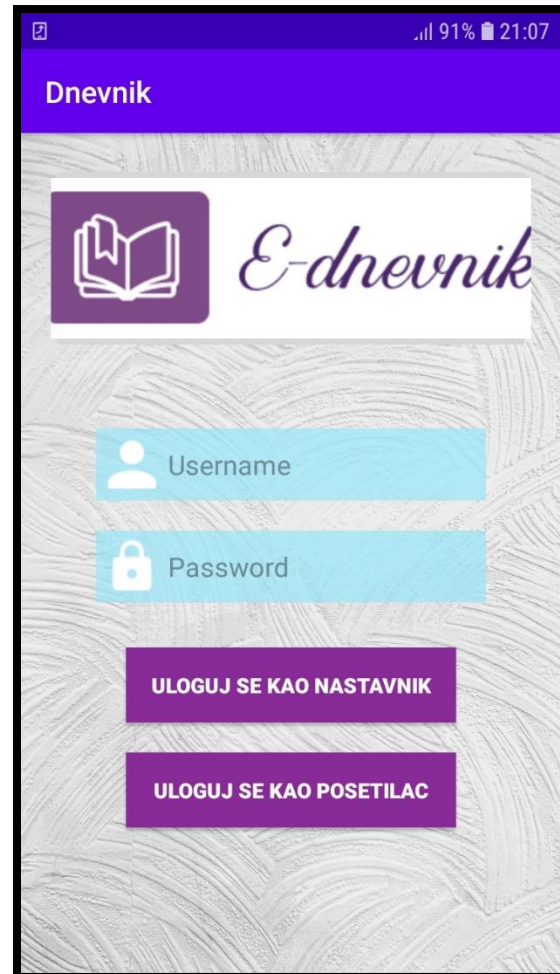
Posetiocu dakle učeniku ili roditelju treba omogućiti prosti prikaz predmeta, ocene iz tih predmeta, ime i prezime nastavnika koji predaje predmet i datum kada je ocena upisana ili ažurirana.

3. Izgled aplikacije i korišćenje

Nakon instaliranja programa na eksterni android uređaj **SAMSUNG Galaxy J7**, urađeni su screenshot-ovi minimalnog načina korišćenja aplikacije.



Slika1: Splash screen layout „activity_main“



Slika2: Login layout „activity_login“

Splash screen (slika1) je prva stvar koju vidimo kada se startuje aplikacija i traje dve sekunde, koristi se samo zbog profesionalnijeg izgleda.

Login layout (slika2) nudi unos korisničkog imena i šifre. Korisničko ime je uvek u formatu ime prezime (npr. Jovan Bogdanović) a šifra je ona koju učenik dobije od nastavnika ili nastavnik dobije od administratora.

3.1. Administrator interfejs



Slika3: Unos i dugme za administratora

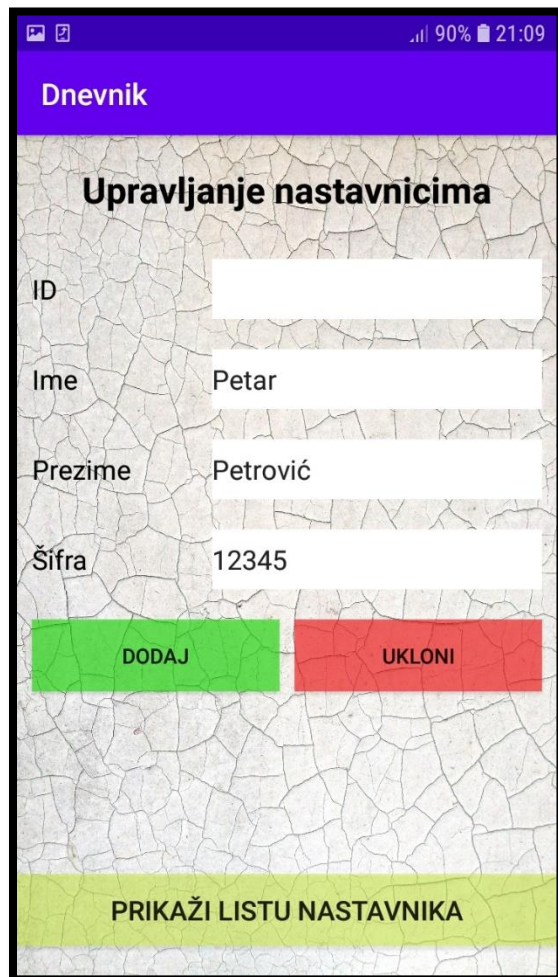


Slika4: Admin layout „activity_admin“

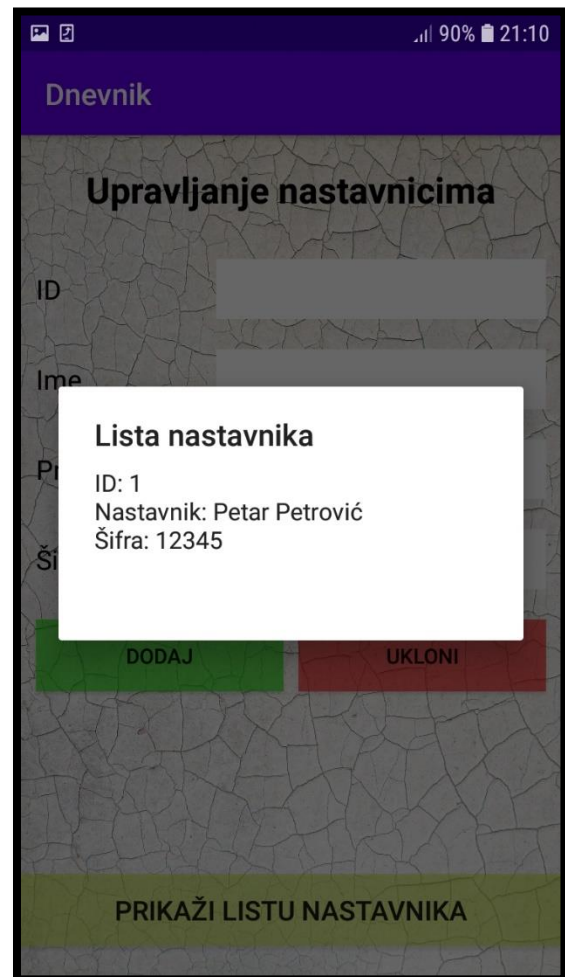
Kada u polja unesemo vrednosti korisničko ime: admin, šifra: admin, i pritisnemo sliku koja funkcioniše kao skriveno dugme i koja je obeležena crvenom strelicom (slika3) ulogovaćemo se kao administrator.

Admin layout (slika4) nam nudi dva dugmeta za upravljanje nastavnicima ili predmetima.

Klikom na dugme Nastavnici u Admin layout-u (slika4) pristupamo upravljanju nastavnicima.



Slika5: Unos „activity_admin_nastavnik“

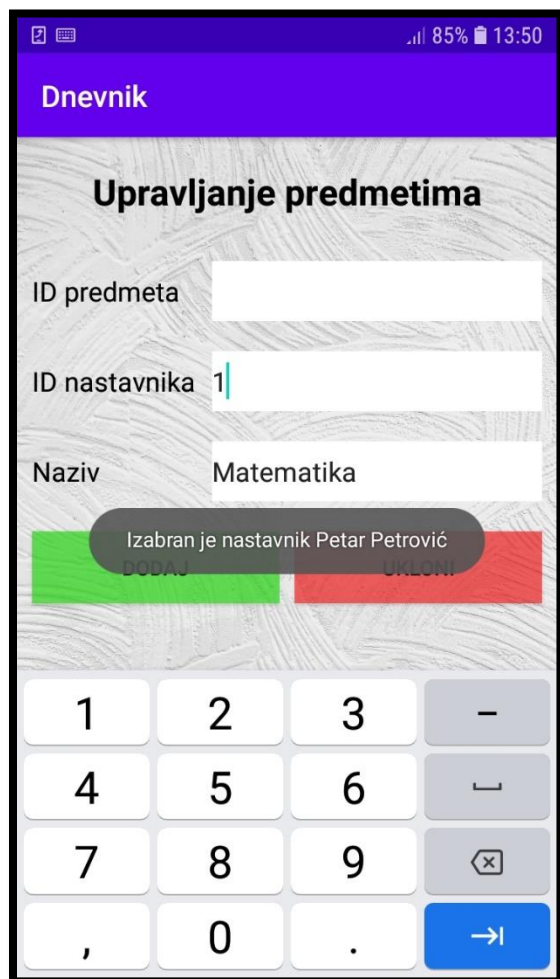


Slika6: Lista nastavnika

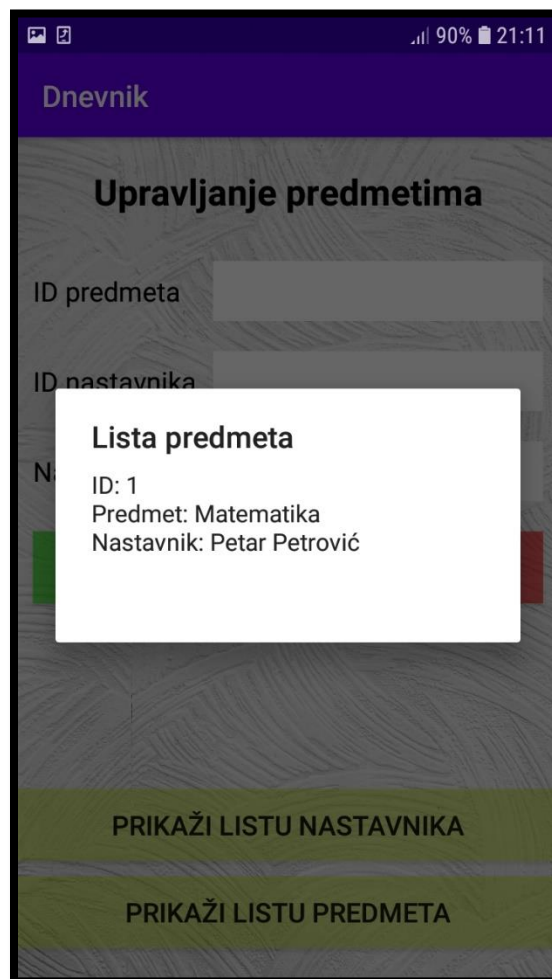
Upravljanje nastavnicima (slika5) nam omogućava da unesemo ime, prezime i šifru nastavnika kojeg želimo da registrujemo klikom na zeleno dugme dodaj ili unošenjem vrednosti u polje ID i klikom na crveno dugme ukloni izbrišemo izabranog nastavnika iz baze podataka.

Lista nastavnika (slika5) nam se prikazuje kada pritisnemo na žuto dugme u dnu (prikaži listu nastavnika) i kreirana je kao Alert dialog u koji se učitavaju vrednosti iz baze podataka.

Klikom na dugme Predmeti u Admin layout-u (slika4) pristupamo upravljanju predmetima.



Slika7: Unos „activity_predmeti“



Slika8: Lista predmeta

Upravljanje predmetima (slika7) nam omogućava da unesemo ID nastavnika kome želimo da dodelimo predmet, unošenjem postojećeg ID-ja pojavljuje se toast koji nam kaže ime i prezime nastavnika koji nosi taj ID, zatim unosimo naziv predmeta i kada pritisnemo zeleno dugme dodaj, predmet se učitava u bazu podataka. Unošenjem vrednosti u polje ID predmeta dobijamo vrednosti u ostalim poljima vezane za taj ID zatim pritiskom na crveno dugme ukloni brišemo predmet iz baze podataka.

Lista predmeta (slika8) funkcioniše kao lista nastavnika (slika6) pritiskom na dugme u dnu (prikaži listu predmeta) izlazi nam Alert dialog u koji se učitavaju vrednosti iz baze podataka.

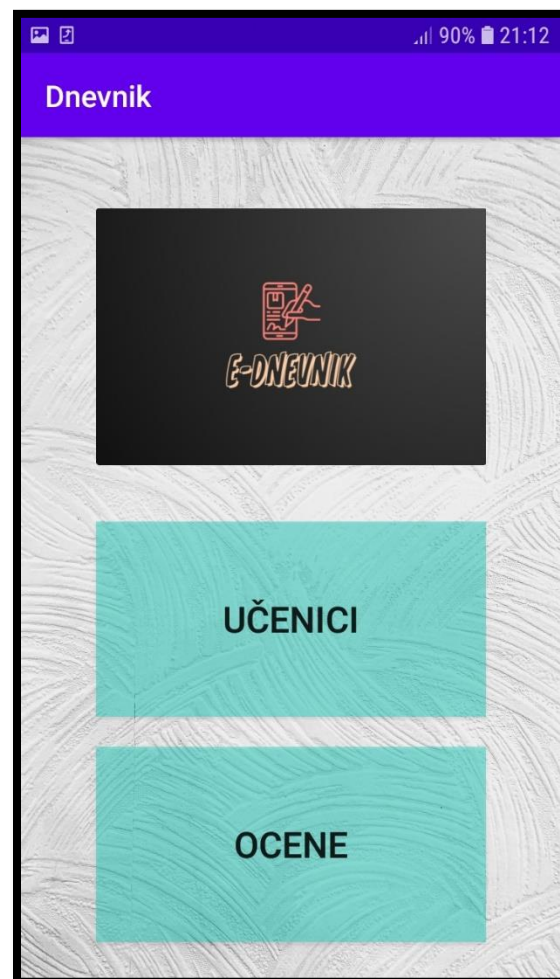
S tim što u aktivnosti upravljanja predmetima imamo i opciju da prikažemo listu nastavnika kako bi nam bilo lakše da pronađemo ID nastavnika kojem dodeljujemo predmet.

3.2. Nastavnik interfejs

Dakle vratimo se na stranicu Login gde unosimo korisničko ime i šifru koju je administrator prosledio nastavniku.



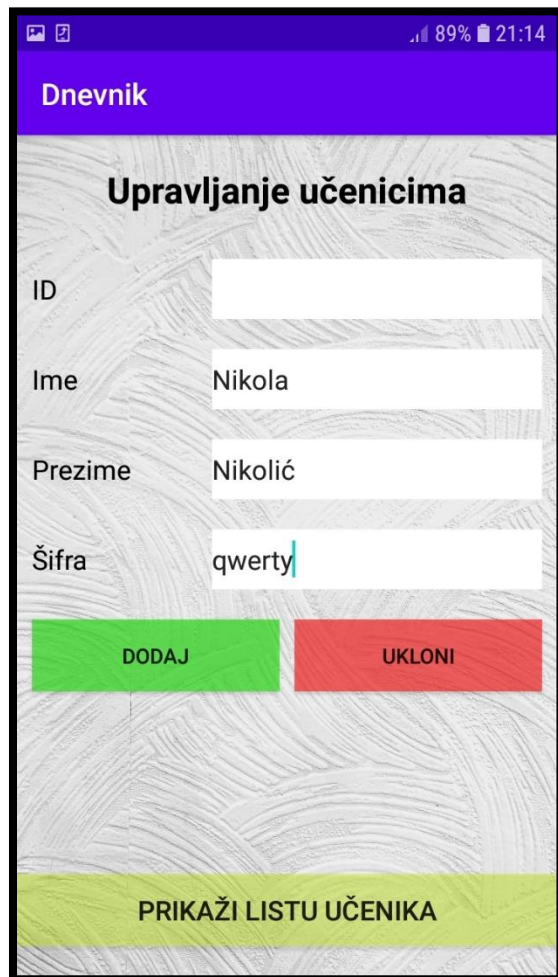
Slika9: Login unos za nastavnika



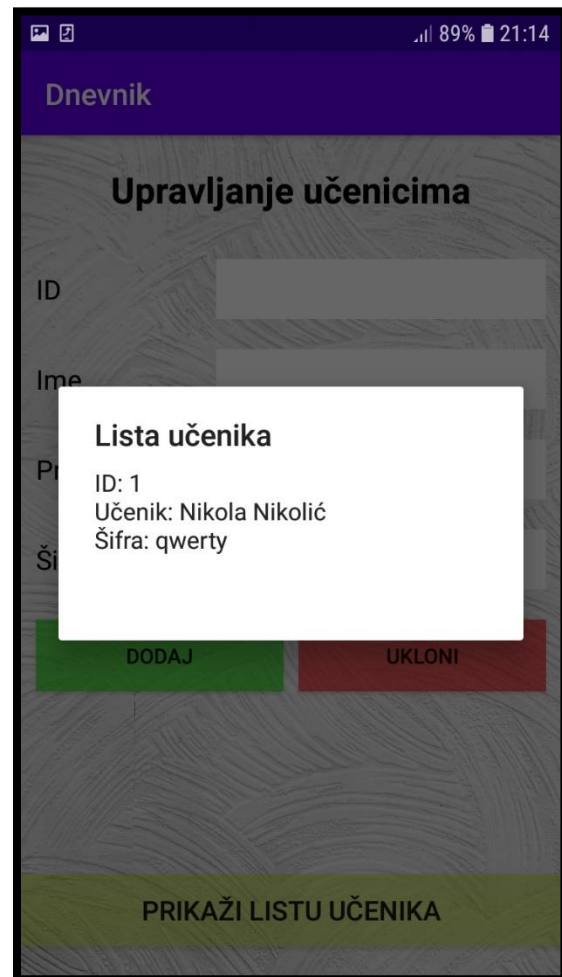
Slika10: Nastavnik layout „activity_nastavnik“

Kao što smo već videli kao administrator smo kreirali nastavnika koji se zove Petar Petrović i dodelili mu šifru 12345, kada unesemo te vrednosti u polja za Login (Slika9) i pritisnemo uloguj se kao nastavnik pristupićemo nastavnikovoj aktivnosti.

Nastavnik prvo ima opciju da izabere da li želi da upravlja učenicima ili ocenama (Slika10).



Slika11: Unos učenika „activity_učenik“



Slika12: Lista učenika

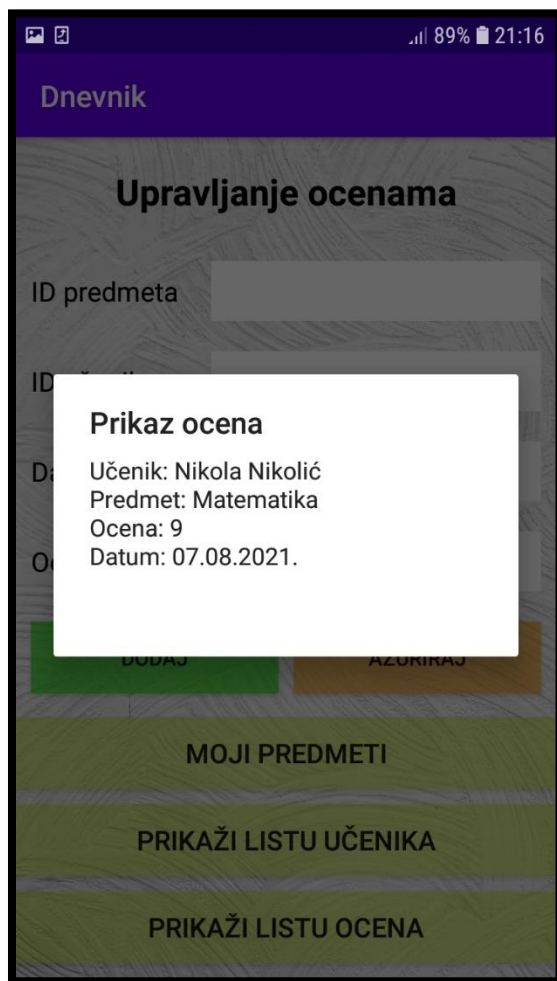
Klikom na dugme učenici u nastavnikovoj aktivnosti (Slika10) pristupamo upravljanju učenicima (Slika11) gde unosimo vrednosti u polja i pritiskom na zeleno dugme dodaj, registrujemo novog učenika u bazu podataka, a kada unesemo

postojeću vrednost u polje ID dobijamo podatke učenika sa tim ID-jem i klikom na crveno dugme ukloni, brišemo izabranog učenika iz baze podataka.

U dnu ponovo imamo dugme koje nam omogućava da vidimo listu registrovanih učenika (Slika12) koja se prikazuje kao Alert dialog u koji se unose vrednosti iz baze podataka.



Slika13: Unos za ocenu „activity_ocene“



Slika14: Lista ocena

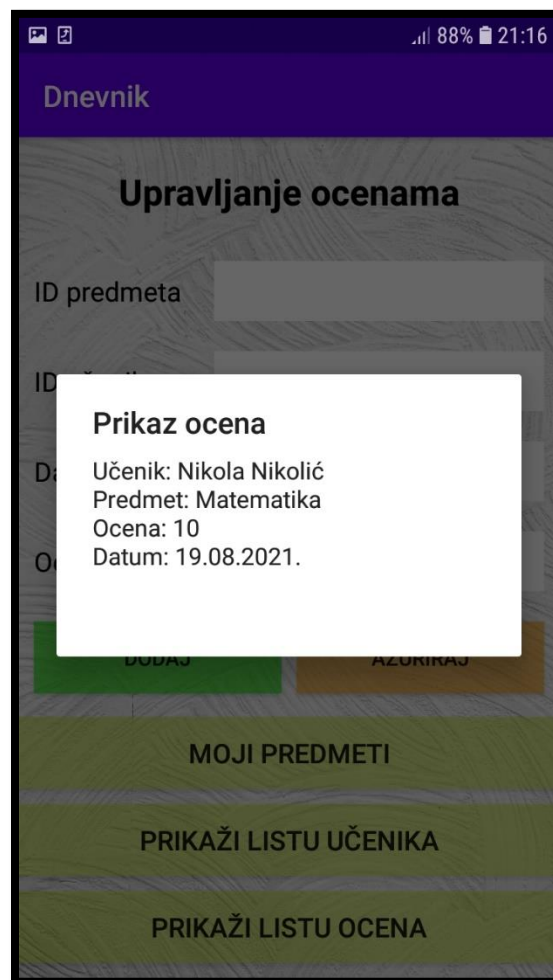
Kada se vratimo na nastavnikovu aktivnost (Slika10) i kliknemo na dugme ocene pristupamo upravljanju ocena (Slika13), gde je potrebno da unesemo ID predmeta iz kog dajemo ocenu što kada uradimo pojavljuje se toast poruka koja nam kaže koji predmet nosi unešeni ID, potom unosimo ID učenika gde se ponovo pojavljuje toast

poruka koja nam kaže koji učenik nosi vrednost izabranog ID-ja, zatim unosimo datum unošenja ocene i konačno unosimo ocenu koju je učenik dobio iz predmeta, kada sve to unesemo klikom na zeleno dugme dodaj, podaci se unose u bazu podataka.

U dnu nastavnik ima tri dugmeta, jedno za prikaz predmeta koji su mu dodeljeni i iz kojih ima pravo da ocenjuje učenike kako bi lakše izabrao ID predmeta koji mu je potreban za unos, zatim sledeće dugme mu omogućava da vidi listu učenika kako bi lakše izabrao ID učenika kojem želi upisati ocenu, i na kraju imamo dugme za prikaz ocena koje je upisao učenicima (Slika14) koja je kreirana kao Alert dialog u koji se unose vrednosti iz baze podataka, unose se samo ocene koje je ulogovani nastavnik upisao.



Slika15: Ažuriranje ocene



Slika16: Prikaz ažurirane ocene

Takođe nastavnik ima opciju da ažurira ocenu ukoliko dođe do promene (Slika15), na taj način što unese ID predmeta i ID učenika koji već ima ocenu iz tog predmeta i onda unese novi datum i novu ocenu i zatim klikne na narandžasto dugme ažuriraj, tada se novi podaci unose u bazu podataka i zamenjuju stare.

Prikaz ažurirane ocene kao primer je prikazan na slici 16.

3.3. Posetilac interfejs

Kada se ponovo vratimo na Login aktivnost i unesemo vrednosti koje nastavnik prosleđuje učeniku možemo se ulogovati i videti njegove ocene. U ovom primeru smo uneli korisničko ime Nikola Nikolić i šifru qwerty.



Slika17: Login unos za učenika

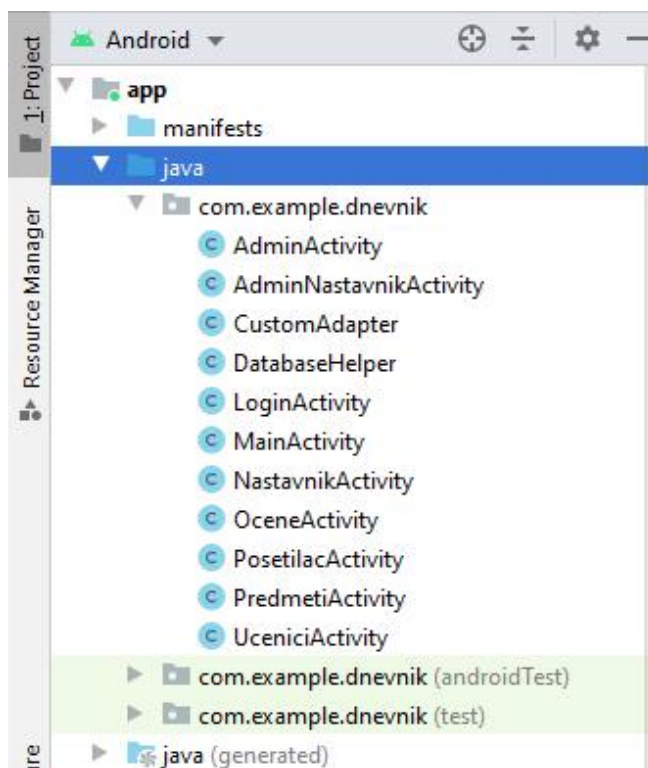


Slika18: Učenik layout „activity_posetilac“

Kada unesemo vrednosti za učenika (Slika17) i pritisnemo dugme uloguj se kao posetilac pristupamo učenikovoj aktivnosti (Slika18) prikazuje nam se samo Ime i prezime učenika na samom vrhu i ispod se ispisuju ocene iz baze podataka, njihov prikaz je omogućen uz pomoć RecyclerView-a i svaka ocena je izdvojena u poseban CardView.

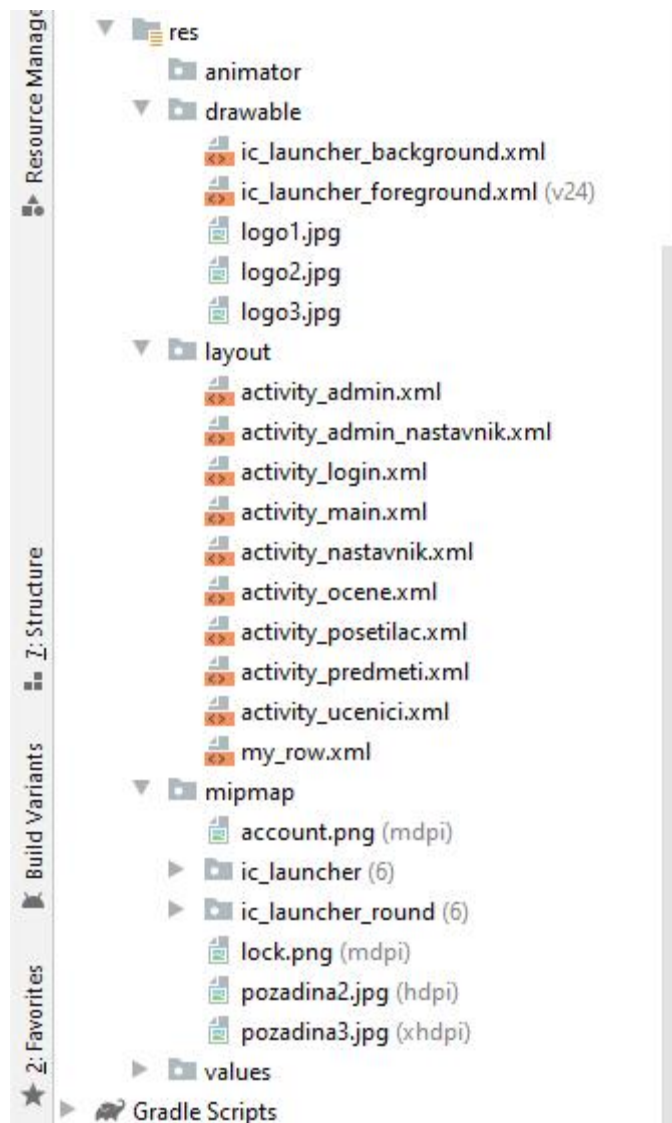
U ovom primeru korišćenja aplikacije kreiran je minimalan sistem od jednog nastavnika, jednog predmeta, jednog učenika i jedne ocene, čisto da bi se pokazala mehanika korišćenja aplikacije.

4. Implementacija i opis delova koda



Slika19: Java klase

Kao što možemo videti na slici 19 u programu postoji 9 Activity klasa koje su zadužene za ponašanje određenih elemenata u svakom layout-u aplikacije, jedna DatabaseHelper klasa za kreiranje, upravljanje bazom podataka i vršenje upita nad njom i jedna CustomAdapter klasa koja služi za potrebe Prikaza ocena posetiocu u RecyclerView-u.



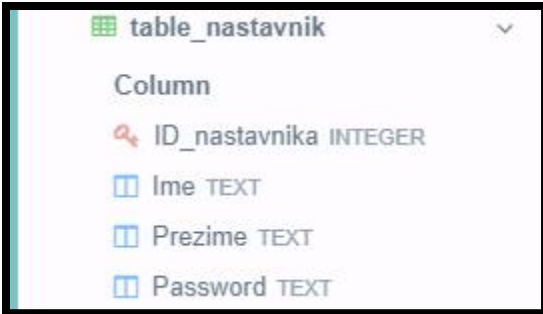
Slika20: xml fajlovi i slike

Na slici 20 vidimo da postoji 9 xml layout fajlova za svaku aktivnost po jedan i jedan my_row fajl za potrebe prikaza ocena u redovima posetiocu. U ovim fajlovima se definiše izgled aplikacije i njenih elemenata.

Takođe imamo i 3 slike koje služe kao logo u folderu drawable, i 4 slike u folderu mipmap one su odvojene tu jer su većeg kvaliteta (HD), od toga imamo 2 slike za potrebe pozadine i 2 slike koje služe kao ikonice u aktivnosti Login.

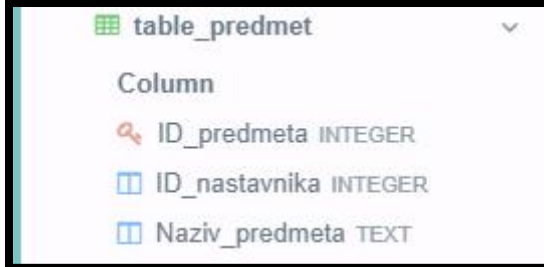
4.1. Baza podataka

Baza podataka se sastoji od četiri tabele, jedna za nastavnike, jedna za predmete, jedna za učenike i jedna za ocene.



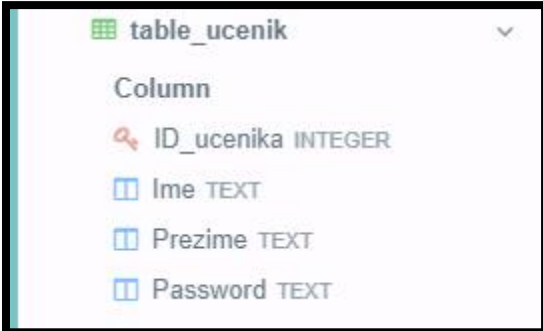
Column
ID_nastavnika INTEGER
Ime TEXT
Prezime TEXT
Password TEXT

Slika21: Izgled tabele nastavnik



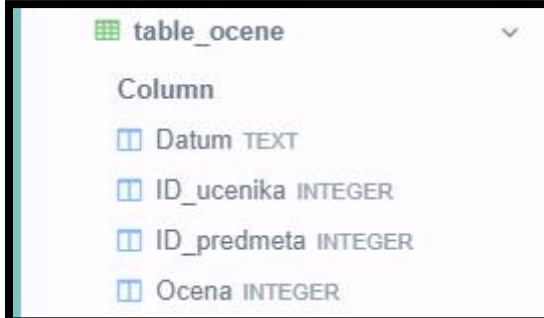
Column
ID_predmeta INTEGER
ID_nastavnika INTEGER
Naziv_predmeta TEXT

Slika22: Izgled tabele predmet



Column
ID_ucenika INTEGER
Ime TEXT
Prezime TEXT
Password TEXT

Slika23: Izgled tabele učenik



Column
Datum TEXT
ID_ucenika INTEGER
ID_predmeta INTEGER
Ocena INTEGER

Slika24: Izgled tabele ocene

U nastavku ćemo prikazati delove koda i pojasniti ih, kao i upite vezane za bazu podata.

4.1.1. Tabela nastavnik

Prvo vršimo inicijalizaciju naziva tabele i kolona.

```
private final static String TABLE_NAME = "table_nastavnik";
private final static String COL1 = "ID_nastavnika";
private final static String COL2 = "Ime";
private final static String COL3 = "Prezime";
private final static String COL4 = "Password";
```

Zatim u metodi onCreate(SQLiteDatabase db) definišemo SQL upit kao String i na kraju ga izvršimo nad bazom, to izgleda ovako.

```
String sql = "CREATE TABLE IF NOT EXISTS "
    + TABLE_NAME + " ( "
    + COL1 + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COL2 + " TEXT not null, "
    + COL3 + " TEXT not null, "
    + COL4 + " TEXT not null)";
db.execSQL(sql);
```

Na ovaj način smo kreirali tabelu, i postavili ID nastavnika za primaran ključ, to znači da je jedinstven za svaki red u tabeli. Autoincrement znači da će se automatski dodeljivati vrednosti u toj koloni i to uvek za +1.

U sledećem delu koda kreiramo metodu koja kad se pozove u Java klasi preuzima sve podatke iz tabele nastavnika.

```
public Cursor getNastavnikData() throws SQLException
{
    db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME;
    return db.rawQuery(sql, null);
}
```

Sledeća metoda koju kreiramo jeste LoginNastavnik gde uz pomoć Cursor-a proveravamo da li kada unete vrednosti u Login layout-u ubacimo u SELECT upit nad tabelom nastavnik dobijamo red u tabeli ili ne, time proveravamo da li je osoba registrovana u bazi.

```

public boolean LoginNastavnik(String username, String password) throws
SQLException
{
    Cursor mCursor = db.rawQuery("SELECT (Ime || ' ' || Prezime) AS username,
password FROM " + TABLE_NAME + " WHERE username=? AND password=?", new
String[]{username,password});
    if (mCursor != null) {
        if(mCursor.getCount() > 0)
        {
            return true;
        }
    }
    return false;
}

```

Uz pomoć „concat-a“ spajamo vrednosti iz kolone ime i kolone prezime sa razmakom između i koristimo kao novu kolonu koja se zove username.

Zatim pravimo metodu za kreiranje novog nastavnika, odnosno unošenje podataka o nastavniku u tabelu.

```

public boolean noviNastavnik(String ime, String prezime, String sifra) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COL2, ime);
    values.put(COL3, prezime);
    values.put(COL4, sifra);

    long result = db.insert(TABLE_NAME, null, values);
    return result != -1;
}

```

Kreiramo i metodu za brisanje nastavnika iz baze tako što preuzmemo uneti ID i uporedimo ga sa kolonom u tabeli nastavnik, ako se pronađe rezultat uklanjamo taj red.

```

public boolean obrisiNastavnika(String id) {
    SQLiteDatabase db = this.getWritableDatabase();
    long result = db.delete(TABLE_NAME, "ID_nastavnika=?", new String[]{id});
    return result != -1;
}

```

Sledeće dve metode su slične, u jednoj za dobijenu vrednost ID-ja selektujemo red u tabeli gde se pronađe taj ID i vratimo taj red, dok u drugoj za dobijene vrednosti ime, prezime i šifra mi pronalazimo red u tabeli i vraćamo ga, ona se koristi da bi pronašli ID nastavnika koji se ulogovao.

```
public Cursor nadjiNastavnika(int id) {
    SQLiteDatabase db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME + " WHERE " + COL1 + " =' " +
id + "'";
    return db.rawQuery(sql, null);
}

public Cursor nadjiIDNastavnika(String ime, String prezime, String pass) {
    SQLiteDatabase db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME + " WHERE " + COL2 + " =? AND
" + COL3 + " = ? AND " + COL4 + "=?";
    return db.rawQuery(sql, new String[]{ime, prezime, pass});
}
```

4.1.2. Tabela učenik

Ponovo inicijalizujemo imena tabele i kolona.

```
private final static String TABLE_NAME2 = "table_ucenik";
private final static String COL1_2 = "ID_ucenika";
private final static String COL2_2 = "Ime";
private final static String COL3_2 = "Prezime";
private final static String COL4_2 = "Password";
```

Kreiramo tabelu i izvršavamo SQL kod u metodi onCreate kao i kod tabele nastavnik. Primarni ključ nam je ID učenika.

```
String sql2 = "CREATE TABLE IF NOT EXISTS "
    + TABLE_NAME2 + " ( "
    + COL1_2 + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COL2_2 + " TEXT not null, "
    + COL3_2 + " TEXT not null, "
    + COL4_2 + " TEXT not null)";
db.execSQL(sql2);
```

Pravimo metode getUcenikData, LoginUcenik, noviUcenik, obrisiUcenika i nadjiUcenika skoro na isti način kao i kod nastavnika, samo vršimo upite nad tabelom učenik.

4.1.3. Tabela predmet

Prva stvar je ponovo inicijalizacija naziva tabele i kolona.

```
private final static String TABLE_NAME3 = "table_predmet";
private final static String COL1_3 = "ID_predmeta";
private final static String COL2_3 = "ID_nastavnika";
private final static String COL3_3 = "Naziv_predmeta";
```

Zatim kreiranje tabele i izvršavanje SQL koda u metodi onCreate.

```
String sql3 = "CREATE TABLE IF NOT EXISTS "
    + TABLE_NAME3 + " ( "
    + COL1_3 + " INTEGER PRIMARY KEY AUTOINCREMENT, "
    + COL2_3 + " INTEGER not null, "
    + COL3_3 + " TEXT not null, "
    + " FOREIGN KEY (" + COL2_3 + ") REFERENCES " + TABLE_NAME + "(" + COL2_3 + "), "
    + " UNIQUE(" + COL3_3 + "));";
db.execSQL(sql3);
```

ID predmeta nam je primarni ključ, sada imamo i strani ključ (Foreign key) i to ID nastavnika koji referenciramo na tabelu nastavnika i na kolonu ID nastavnika. Takođe imamo i jedinstvenu kolonu (Unique), kolona naziv predmeta nam je jedinstvena jer ne želimo da dođe do kreiranja više istih predmeta.

U sledećoj metodi selektujemo tabelu predmet i na nju dodamo tabelu nastavnika koristeći inner join preko kolona ID nastavnika, to radimo kako bismo imali pristup imenu i prezimenu profesora koji predaje predmet. I koristeći order by redove sortiramo rastućim redosledom po koloni ID nastavnika.

```
public Cursor getPredmetiData() throws SQLException
{
    db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME3 + " a INNER JOIN " + TABLE_NAME + "
b ON a." + COL2_3 + " = b." + COL1_3 + " ORDER BY " + COL2_3;
    return db.rawQuery(sql, null);
}
```

Slično, u sledećoj metodi pronalazimo sve predmete za određenu vrednost ID nastavnika, to nam služi kako bi nastavniku ispisali samo njegove predmete.

```

public Cursor getMojPredmetiData(String id) throws SQLException
{
    db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME3 + " a INNER JOIN " + TABLE_NAME4 + " b ON a." + COL2_3 + " = b." + COL1_4 + " WHERE a." + COL2_3 + " =?";
    return db.rawQuery(sql, new String[] {id});
}

```

Metode noviPredmet, obriši predmet i nadjiPredmet su skoro iste kao i kod tabela nastavnik i učenik, samo se upiti vrše nad tabelom predmet.

I za kraj imamo metodu za pronalaženje predmeta uz pomoć njegovog ID-ja i uz pomoć ID nastavnika, ovu metodu koristimo kod rada sa ocenama kako bi pronašli odgovarajući predmet koji nastavnik predaje.

```

public Cursor nadjiMojPredmet(int id, String nastavnikID) {
    SQLiteDatabase db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME3 + " WHERE " + COL1_3 + " = '" + id + "' AND " + COL2_3 + " = '" + nastavnikID + "'";
    return db.rawQuery(sql, null);
}

```

4.1.4. Tabela ocene

Kao i uvek, prvo inicijalizujemo nazive tabele i kolona.

```

private final static String TABLE_NAME4 = "table_ocene";
private final static String COL1_4 = "Datum";
private final static String COL2_4 = "ID_ucenika";
private final static String COL3_4 = "ID_predmeta";
private final static String COL4_4 = "Ocena";

```

Zatim kreiramo tabelu u metodi onCreate.

```

String sql4 = "CREATE TABLE IF NOT EXISTS "
    + TABLE_NAME4 + " ( "
    + COL1_4 + " TEXT , "
    + COL2_4 + " INTEGER not null, "
    + COL3_4 + " INTEGER not null, "
    + COL4_4 + " INTEGER , "
    + " FOREIGN KEY (" + COL2_4 + ") REFERENCES " + TABLE_NAME2 + "(" + COL1_2 + ")",
    FOREIGN KEY (" + COL3_4 + ") REFERENCES " + TABLE_NAME3 + "(" + COL1_3 + ")",

```

```

        + " UNIQUE("+COL2_4+", "+COL3_4+"));";
db.execSQL(sql4);

```

U ovoj tabeli nemamo primarni ključ, imamo dva strana ključa (Foreign key) za ID učenika i ID predmeta koje referenciramo na tabelu učenik i tabelu premet respektivno.

Takođe imamo jedinstvenu (Unique) vrednost za dve kolone (Unique key pair) čime ograničavamo da iz jednog predmeta učenik može da ima jednu ocenu, kako se ne bi pojavljivale duplirane vrednosti u prikazu ocena kod posetioca.

U narednoj metodi spajamo tabele ocene, predmet i učenik uz pomoć odgovarajućih kolona koje su strani ključevi, ovu metodu koristimo kod prikaza ocena za nastavnika zato metoda traži vrednost ID nastavnika koja se posle u SQL naredbi upoređuje sa kolonom ID nastavnika u tabeli predmeti i na taj način ispisuje sve ocene koje je ulogovani nastavnik registrovao u bazu, sortirajući ih rastućim redosledom po koloni ID učenika.

```

public Cursor getOceneData(String id) throws SQLException
{
    db = this.getWritableDatabase();
    String sql = "SELECT * FROM "+ TABLE_NAME4 +" a INNER JOIN
"+TABLE_NAME3+" b ON a."+ COL3_4 +" = b."+ COL1_3 +" INNER JOIN
"+TABLE_NAME2+" c ON a."+ COL2_4 +" = c."+ COL1_2 +" WHERE b."+ COL2_3 +"=?
ORDER BY "+ COL2_4;
    return db.rawQuery(sql, new String[] {id});
}

```

Sledi metoda koja služi za ispis ocena ulogovanom posetiocu.

```

public Cursor ispisiOcene(String ime, String prezime, String pass) throws
SQLException
{
    db = this.getWritableDatabase();
    String sql = "SELECT * FROM "+ TABLE_NAME4 +" a INNER JOIN "+ TABLE_NAME3
+" b ON a."+ COL3_4 +" = b."+ COL1_3
        +" INNER JOIN "+ TABLE_NAME2 +" c ON a."+ COL2_4 +" = c."+ COL1_2
        +" INNER JOIN "+ TABLE_NAME +" d ON b."+ COL2_3 +" = d."+ COL1
        +" WHERE c."+ COL2_2 +" =? AND c."+ COL3_2 +" =? AND c."+ COL4_2
+ " =? ORDER BY d."+ COL1;

```



```

        return db.rawQuery(sql, new String[]{ime, prezime, pass});
    }

```

Kreira se upit i spajaju se sve 4 tabele preko inner join-a i odgovarajućih ključeva, zatim se uzimaju ulazne vrednosti za metodu a to su ime, prezime i šifra učenika i pretražuju se po odgovarajućim kolonama u bazi, rezultat se sortira rastućim redosledom po koloni ID nastavnika.

Metoda novaOcena se kreira skoro isto kao i noviNastavnik, noviUcenik ili noviPredmet. Samo se upit vrši nad tabelom ocene.

```

public void updateOcena(int predmetID, int ucenikID, String datum, String
ocena) {
    SQLiteDatabase db = this.getWritableDatabase();
    String sql = "UPDATE " + TABLE_NAME4 + " SET " + COL1_4 + "='" + datum + "',
" + COL2_4 + "='" + ucenikID + "', "
        + COL3_4 + "='" + predmetID + "', " + COL4_4 + "='" + ocena + "'"
        + " WHERE " + COL1_3 + "='" + predmetID + "' AND " + COL1_2 + "='" +
ucenikID + "'" ;
    db.execSQL(sql);
}

```

Naredbom UPDATE ažuriramo vrednosti u tabeli kao ulaze u metodu imamo ID predmeta i ID učenika, datum i ocena na osnovu ID predmeta i ID učenika pronalazimo red u tabeli ocena koji treba da se ažurira zato što ta dva ID-ja zajedno čine jedinstveni par.

Sledeća metoda nam služi kako bismo proverili da li ocena postoji i ako postoji da je kasnije ažuriramo.

```

public Cursor nadjiOcenu(int predmetID, int ucenikID) {
    SQLiteDatabase db = this.getWritableDatabase();
    String sql = "SELECT * FROM " + TABLE_NAME4 + " WHERE " + COL3_4 + "='"
+ predmetID + "' AND " + COL2_4 + "='" + ucenikID + "'";
    return db.rawQuery(sql, null);
}

```

4.2. Java klase

Kao što smo već naveli u projektu ima 11 java klasa, s tim što smo klasu databaseHelper već objasnili u prethodnom poglavlju (Baza podataka), sada ćemo navesti ostale klase i pojasniti određene delove koda.

4.2.1. MainActivity

Ova klasa je vrlo jednostavna, jedini posao joj je da startuje Login aktivnost ali da pre toga napravi pauzu od 2 sekunde.

```
private final int SPLASH_DISPLAY_LENGTH = 2000;
```

Inicijalizujemo pauzu u milisekundama, zatim uz pomoć funkcije postDelayed omogućimo da se promeni aktivnost uz pomoć intenta nakon pauze.

```
new Handler().postDelayed(new Runnable(){
    @Override
    public void run() {
        Intent mainIntent = new
Intent(MainActivity.this, LoginActivity.class);
        MainActivity.this.startActivity(mainIntent);
        MainActivity.this.finish();
    }
}, SPLASH_DISPLAY_LENGTH);
```

4.2.2. LoginActivity

Prvo povežemo preko ID vrednosti objekata sa odgovarajućeg layout fajla (activity_login) sa inicijalizovanim vrednostima u klasi.

I povežemo se sa databaseHelper klasom.

```
DatabaseHelper db = new DatabaseHelper(this);
```

U nastavku ćemo pogledati kod koji služi za logovanje administratora unošenjem vrednosti u polja i pritiskom na ImageButton „ib“ kao na slici 3.

```

ib.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = txtUsername.getText().toString();
        String password = txtPassword.getText().toString();
        if(username.equals("admin") && password.equals("admin")){
            Toast.makeText(LoginActivity.this, "Dobro došao administratore!",
            Toast.LENGTH_LONG).show();
            Intent intent = new Intent(LoginActivity.this,
            AdminActivity.class);
            startActivity(intent);
        }
    }
});

```

Postavili smo osluškivač na dugme (sliku) u kojem se dešava sledeće, preuzimamo vrednosti koje su unete u polja username i password i ubacujemo ih u if uslov gde proveravamo da li su unete vrednosti admin i admin u oba polja, ako jesu, vrši se promena Intenta, startuje se aktivnost AdminActivity i dočekuje nas toast poruka „Dobro došao administratore!“ ako su unete pogrešne vrednosti, neće se ništa desiti kako bi dugme ostalo skriveno.

Zatim postavljamo na osluškivač na dugme „Uloguj se kao nastavnik“ gde takođe vršimo proveru unešenih podataka tako što ih potražimo u bazi podataka.

```

nastavnik.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String username = txtUsername.getText().toString();
        String password = txtPassword.getText().toString();

        try{
            if(username.length() > 0 && password.length() >0)
            {
                DatabaseHelper dbUser = new
                DatabaseHelper(LoginActivity.this);
                db.getNastavnikData();

                if(db.LoginNastavnik(username, password))
                {
                    Toast.makeText(LoginActivity.this, "Uspešno ste se
                    ulogovali", Toast.LENGTH_LONG).show();
                    Intent intent = new Intent(LoginActivity.this,
                    NastavnikActivity.class);

```

```

        intent.putExtra("usr", username);
        intent.putExtra("pass", password);
        startActivity(intent);
    }else{
        Toast.makeText(LoginActivity.this,"Pogrešan Username ili
Password", Toast.LENGTH_LONG).show();
    }
    dbUser.close();
}
else{
    Toast.makeText(LoginActivity.this,"Polja ne smeju biti
prazna", Toast.LENGTH_LONG).show();
}

}catch(Exception e)
{
    Toast.makeText(LoginActivity.this,e.getMessage(),
Toast.LENGTH_LONG).show();
}
}

});

```

Dakle, ponovo preuzmemo vrednosti iz polja username i password sa aktivnosti Login, i stavljamo ih u prvu if naredbu, gde proveravamo da li su polja prazna, u slučaju da jesu ispisujemo toast „Polja ne smeju biti prazna“, zatim se povezujemo sa bazom i preuzimamo podatke o nastavniku delom koda db.getNastavnikData(); Potom imamo if naredbu u kojoj koristimo metodu db.LoginNastavnik(username, password) gde šaljemo kao ulazne parametre username i password i na taj način ih proveravamo da li su ispravni, u slučaju da nisu, prikazujemo toast poruku „Pogrešan Username ili Password“, a ako jesu ispisujemo toast „Uspešno ste se ulogovali“, promenimo intent, startujemo aktivnost NastavnikActivity i šaljemo u novi intent vrednosti username i password koje će nam kasnije trebati, to radimo pomoću intent.putExtra(...);

Sada istu proveru vršimo u slučaju da je pritisnuto dugme „Uloguj se kao posetilac“.

```

posetilac.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String username = txtUsername.getText().toString();
        String password = txtPassword.getText().toString();

        try{
            if(username.length() > 0 && password.length() >0)
            {
                DatabaseHelper dbUser = new
DatabaseHelper(LoginActivity.this);
                db.getUserData();

                if(db.LoginUcenik(username, password))
                {
                    Toast.makeText(LoginActivity.this,"Uspešno ste se
ulogovali", Toast.LENGTH_LONG).show();
                    Intent intent = new Intent(LoginActivity.this,
PosetilacActivity.class);
                    intent.putExtra("usr", username);
                    intent.putExtra("pass", password);
                    startActivity(intent);
                }else{
                    Toast.makeText(LoginActivity.this,"Pogrešan Username ili
Password", Toast.LENGTH_LONG).show();
                }
                dbUser.close();
            }
            else{
                Toast.makeText(LoginActivity.this,"Polja ne smeju biti
prazna", Toast.LENGTH_LONG).show();
            }

        }catch(Exception e)
        {
            Toast.makeText(LoginActivity.this,e.getMessage(),
Toast.LENGTH_LONG).show();
        }

    }
});

```

Ponovo preuzimamo vrednosti username i password iz „activity_login“ polja, proveravamo da li su polja prazna koristeći .length()>0, ako jesu ispisujemo poruku da polja ne smeju biti prazna, ako nisu vršimo sledeću proveru pretraživanjem vrednosti u bazi podataka metodom db.LoginUčenik(username, password), ukoliko nema takvih vrednosti, ispisujemo poruku da je pogrešan username ili password, a

ako su vrednosti ispravne, menjamo intent, startujemo aktivnost PosetilacActivity, i šaljemo vrednosti username i password u novi intent ponovo kako bi ih kasnije koristili.

4.2.3. AdminActivity

Izgled ove aktivnosti možemo videti na slici 4.

Imamo samo dva dugmeta koji administratoru omogućavaju da upravlja nastavnicima ili predmetima.

Postavimo setOnClickListener-e na oba dugmeta i samo promenimo intent, i startujemo odgovarajuće aktivnosti za dugme nastavnici startujemo AdminNastavnikActivity, a za dugme premeti startujemo PredmetiActivity.

4.2.4. AdminNastavnikActivity

Izgled ove aktivnosti možemo videti na slici 5.

Prvo povežemo sve objekte iz fajla „activity_admin_nastavnik.xml“ sa Java klasom preko odgovarajućih ID-ja uz pomoć findViewById(R.id.vrednost).

Zatim postavimo prislušivač na dugme dodaj.

```
dodaj.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(provera()){
            boolean oo =
databaseHelper.noviNastavnik(txtIme.getText().toString(),
txtPrezime.getText().toString(), txtSifra.getText().toString());
            if(oo) {
                Toast.makeText(AdminNastavnikActivity.this, "Uspešno
registrovan nastavnik", Toast.LENGTH_LONG).show();
                clear();
            } else {
                Toast.makeText(AdminNastavnikActivity.this, "Doslo je do
greske!", Toast.LENGTH_LONG).show();
            }
        }
    }
});
```

Prva if naredba vrši proveru da li su bitna polja popunjena ili ne, ovako izgleda ta funkcija.

```
private boolean provera() {
    if(txtIme.getText().toString().equals("")) {
        txtIme.setError("Ovo polje je obavezno");
        return false;
    }
    if(txtPrezime.getText().toString().equals("")) {
        txtPrezime.setError("Ovo polje je obavezno");
        return false;
    }
    if(txtSifra.getText().toString().equals("")) {
        txtSifra.setError("Ovo polje je obavezno");
        return false;
    }
    return true;
}
```

Ukoliko su bitna polja ispunjena, inicijalizujemo bool vrednost i dodeljujemo joj vrednost metode noviNastavnik(...) ulazi su nam ime, prezime i šifra, proveravamo da li je bool vrednost true odnosno da li je uspešno registrovan novi nastavnik, ako jeste ispisujemo to u toast poruci, ako nije ispisujemo da je došlo do greške.

Sledi posao koji treba da obavi dugme ukloni.

```
obrisi.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!txtID.getText().toString().equals("")){
            Cursor data =
            databaseHelper.nadjiNastavnika(Integer.parseInt(txtID.getText().toString()));
            if(data.getCount() > 0) {
                boolean oo =
                databaseHelper.obrisiNastavnika(txtID.getText().toString());
                if (oo) {
                    Toast.makeText(AdminNastavnikActivity.this, "Uspesno
                    obrisan nastavnik!", Toast.LENGTH_LONG).show();
                    clear2();
                } else {
                    Toast.makeText(AdminNastavnikActivity.this, "Doslo je do
                    greske!", Toast.LENGTH_LONG).show();
                }
            } else {
                Toast.makeText(AdminNastavnikActivity.this, "Ne postoji
```



```
nastavnik sa tim ID", Toast.LENGTH_LONG).show();
    }
}
});
```

Prvo proveravamo da li je polje ID nastavnika prazno, ako nije, radimo sledeće. Inicijalizujemo cursor kojim ćemo pretražiti bazu podataka metodom `nadjiNastavnika(id)`, u slučaju da cursor ne pronađe vrednosti za uneti ID ispisujemo toast da ne postoji nastavnik sa tim ID-jem, a ako se pronađe uneti ID onda kreiramo bool vrednost i vršimo brisanje iz baze metodom `obrisiNastavnika`, proveravamo da li je bool true ukoliko jeste ispisujemo toast poruku da je nastavnik uspešno obrisan, ako je bool false ispisujemo poruku da je došlo do greške.

Na polje za unos ID-ja postavljamo osluškivač ukoliko dođe do promene teksta sledećom linijom koda.

```
txtID.addTextChangedListener(new TextWatcher(){...}
```

Zatim unutar metode `onTextChanged` vršimo pretragu nastavnika ponovo na osnovu ID-ja ukoliko postoji takav nastavnik u bazi, uzimamo vrednosti u cursor i ispisujemo jednu po jednu u ostala polja (ime, prezime, šifra).

```
if(s.length() > 0) {
    Cursor data =
    databaseHelper.nadjiNastavnika(Integer.parseInt(txtID.getText().toString()));
    if(data.getCount() > 0) {
        Toast.makeText(AdminNastavnikActivity.this, "Pronadjen je nastavnik",
        Toast.LENGTH_LONG).show();
        while(data.moveToNext()){
            txtIme.setText(data.getString(1));
            txtPrezime.setText(data.getString(2));
            txtSifra.setText(data.getString(3));
        }
    } else {
        Toast.makeText(AdminNastavnikActivity.this, "Ne postoji nastavnik sa
        tim ID", Toast.LENGTH_LONG).show();
        clear();
    }
}
```

`moveToNext` metodom preuzimamo vrednosti iz svih kolona i metodom `setText` upisujemo vrednosti iz kolona tabele u polja na `AdminNastavnikActivity`.

Na dugme prikaži listu nastavnika postavljamo osluškivač kako bi ispisali vrednosti iz baze u AlertDialog.

```
prikazi.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor cursor = databaseHelper.getNastavnikData();
        if (cursor.getCount() == 0){
            Toast.makeText(AdminNastavnikActivity.this, "Nema podataka o
nastavnicima", Toast.LENGTH_SHORT).show();
        }
        StringBuilder builder = new StringBuilder();
        while (cursor.moveToNext()){
            builder.append("ID: ").append(cursor.getString(0)).append("\n");
            builder.append("Nastavnik:
").append(cursor.getString(1)).append(" ");
            builder.append(cursor.getString(2)).append("\n");
            builder.append("Šifra:
").append(cursor.getString(3)).append("\n\n");
        }
        showMessage("Spisak nastavnika", builder.toString());
    }
});
```

U cursor unosimo vrednosti iz tabele nastavnik metodom getNastavnikData(), if naredbom proveravamo da li ima podataka u tabeli, ako nema ispisujemo to, ako ih ima, ispisujemo ih u AlertDialog koji smo kreirali sledećim kodom.

```
private void showMessage(String title, String message){
    AlertDialog.Builder dialog = new
AlertDialog.Builder(AdminNastavnikActivity.this);
    dialog.setTitle("Lista nastavnika");
    dialog.setMessage(message);
    dialog.show();
}
```

Zatim koristeći StringBuilder upisujemo vrednosti iz cursor-a tako što dodajemo na builder-a tekst koristeći .append() i na kraju pozovemo metodu showMessage kako bi upisali tekst iz builder-a u AlertDialog.

4.2.5. PredmetiActivity

Izgled ove aktivnosti možemo videti na slici 7.

Prvo povežemo sve objekte iz fajla „activity_predmeti.xml“ sa Java klasom preko odgovarajućih ID-ja uz pomoć findViewById(R.id.vrednost).

Zatim postavimo prisluškivač na dugme dodaj, skoro isto kao kod aktivnosti AdminNastavnikActivity samo sada prvo proverimo da li postoji nastavnik kojem treba biti dodeljen predmet metodom nadjiNastavnika(id), a onda dodamo predmet metodom noviPredmet(idNastavnika, nazivPredmeta).

Onda postavljamo osluškivač na dugme ukloni, ponovo gorovo isto kao i kod aktivnosti AdminNastavnikActivity, jedina razlika su metode nadjiPredmet umesto nadjiNastavnika i obrisiPredmet umesto obrisiNastavnika.

Na polje za unos ID predmeta postavljamo osluškivač ukoliko dođe do promene teksta sledećom linijom koda.

```
txtID.addTextChangedListener(new TextWatcher(){...}
```

Zatim unutar metode onTextChanged vršimo pretragu predmeta na osnovu ID-ja ukoliko postoji takav predmet u bazi, uzimamo vrednosti u cursor i ispisujemo jednu po jednu u ostala polja (ID nastavnika i naziv predmeta).

```
if(s.length() > 0) {
    Cursor data =
    databaseHelper.nadjiPredmet(Integer.parseInt(txtID.getText().toString().trim(
    )));
    if(data.getCount() > 0) {
        Toast.makeText(PredmetiActivity.this,"Pronadjeno je predmet",
        Toast.LENGTH_LONG).show();
        while(data.moveToNext()){
            txtNastavnikID.setText(data.getString(1));
            txtNaziv.setText(data.getString(2));
        }
    } else {
        Toast.makeText(PredmetiActivity.this,"Ne postoji predmet sa tim ID",
        Toast.LENGTH_LONG).show();
        clear();
    }
}
```

Na polje za unos ID nastavnika postavljamo osluškivač ukoliko dođe do promene teksta sledećom linijom koda.

```
txtNastavnikID.addTextChangedListener(new TextWatcher(){...}
```

Zatim unutar metode onTextChanged vršimo pretragu nastavnika na osnovu ID-ja ukoliko postoji takav nastavnik u bazi, uzimamo vrednosti za ime i prezime i upisujemo ih u toast poruku da bi obavestili korisnika kojeg nastavnika je izabrao kada unese vrednost ID-ja.

```
if(s.length() > 0) {
    Cursor data =
databaseHelper.nadjiNastavnika(Integer.parseInt(txtNastavnikID.getText().toString().trim()));
    if(data.getCount() > 0) {
        while(data.moveToNext()) {
            String ime = data.getString(1);
            String prezime = data.getString(2);
            Toast.makeText(PredmetiActivity.this, "Izabran je nastavnik
"+ime+" "+prezime, Toast.LENGTH_LONG).show();
        }
    } else {
        Toast.makeText(PredmetiActivity.this, "Ne postoji nastavnik sa tim
ID", Toast.LENGTH_LONG).show();
    }
}
```

Ponovo kao i u aktivnosti AdminNastavnikActivity imamo dugmad za prikaz liste podataka iz baze koristeći AlertDialog, za dugme „prikaži listu nastavnika“ koristimo metodu getNastavnikData a za dugme „prikaži listu predmeta“ koristimo metodu getPredmetiData.

4.2.6. NastavnikActivity

Izgled ove aktivnosti možemo videti na slici 10.

Ova klasa se koristi kada se korisnik uloguje kao nastavnik. Sledećim delom koda preuzmемо vrednosti za username i password iz aktivnosti LoginActivity.

```
Intent intent = getIntent();  
usr = intent.getStringExtra("usr");  
pass = intent.getStringExtra("pass");
```

Zatim podelimo string iz vrednosti usr na dva dela, koristeći .split() u koji postavimo vrednost \\s+ koja navodi da string treba podeliti kada se naiđe na beli znak odnosno razmak. Time dobijamo vrednost za ime i vrednost za prezime nastavnika, i već imamo šifru.

```
String[] split = usr.split("\\s+");  
ime = split[0];  
prezime = split[1];
```

Zatim imamo dva dugmeta, za dugme učenici samo promenimo intent i startujemo aktivnost UceniciActivity, a za dugme ocene takođe promenimo intent i startujemo aktivnost OceneActivity, s tim što u taj intent pošaljemo vrednosti ime, prezime i šifru.

```
intent.putExtra("ime", ime);  
intent.putExtra("prezime", prezime);  
intent.putExtra("pass", pass);
```

Na ovaj način smo preneli dalje login vrednosti u aktivnost OceneActivity.

4.2.7. UceniciActivity

Izgled ove aktivnosti možemo videti na slici 11.

Prvo povežemo sve objekte iz fajla „activity_ucenici.xml“ sa Java klasom preko odgovarajućih ID-ja uz pomoć findViewById(R.id.vrednost).

Na dugme dodaj postavljamo osluškivač, skoro isto kao kod aktivnosti AdminNastavnikActivity i PredmetiActivity samo koristimo metodu noviUcenik(ime, prezime, šifra).

Na dugme ukloni takođe postavljamo osluškivač skoro isto kao kod aktivnosti AdminNastavnikActivity i PredmetiActivity razlika je u metodama nadjiUcenika i obrisiUcenika.

Na polje za unos ID učenika postavljamo osluškivač ukoliko dođe do promene teksta sledećom linijom koda.

```
txtID.addTextChangedListener(new TextWatcher(){...}
```

Zatim unutar metode onTextChanged vršimo pretragu učenika na osnovu ID-ja ukoliko postoji takav učenik u bazi, uzimamo vrednosti u cursor i ispisujemo jednu po jednu u ostala polja (ime, prezime i šifra).

```
if(s.length() > 0) {
    Cursor data =
    databaseHelper.nadjiUcenika(Integer.parseInt(txtID.getText().toString().trim(
    )));
    if(data.getCount() > 0) {
        Toast.makeText(UceniciActivity.this, "Pronadjen je učenik",
        Toast.LENGTH_LONG).show();
        while(data.moveToNext()){
            txtIme.setText(data.getString(1));
            txtPrezime.setText(data.getString(2));
            txtSifra.setText(data.getString(3));
        }
    } else {
        Toast.makeText(UceniciActivity.this, "Ne postoji učenik sa tim ID",
        Toast.LENGTH_LONG).show();
        clear();
    }
}
```

Ponovo kao i kod aktivnosti AdminNastavnikActivity i PredmetiActivity imamo dugme za prikaz liste podataka iz baze koristeći AlertDialog, za dugme „prikaži listu učenika“ koristimo metodu getUcenikData.

4.2.8. OceneActivity

Izgled ove aktivnosti možemo videti na slici 13.

Prvo povežemo sve objekte iz fajla „activity_ocene.xml“ sa Java klasom preko odgovarajućih ID-ja uz pomoć findViewById(R.id.vrednost).

Zatim preuzmемо iz aktivnosti NastavnikActivity preko intentа vrednosti za ulogovanог nastavnika (ime, prezime i šifru).

```
Intent intent = getIntent();  
ime = intent.getStringExtra("ime");  
prezime = intent.getStringExtra("prezime");  
pass = intent.getStringExtra("pass");
```

Zatim pozovemo metodu nadjiID(); kako bi pronašli ID nastavnika preko informacija koje su nam date sa Login strane. Funkcija nadjiID() izgleda ovako.

```
private void nadjiID() {  
    Cursor nadjiID = databaseHelper.nadjiIDNastavnika(ime, prezime, pass);  
    while (nadjiID.moveToNext()){  
        id = nadjiID.getString(0);  
    }  
}
```

Koristimo metodu nadjiIDNastavnika iz DatabaseHelper klase u koju kao ulaz unosimo vrednosti ime, prezime i šifru, zatim preko cursor-a pronađemo vrednost iz kolone ID i upišemo je u string id.

Na dugme dodaj, postavljamo osluškivač i onda if naredbom vršimo proveru da li su unete vrednosti u sva polja. Zatim pravimo dva cursor-a da pretražimo bazu podataka metodom nadjiMojPredmet i nadjiUcenika obe metode preko unetih ID vrednosti u polja kao na slici 13. U if naredbu ubacujemo proveru da li su cursor-i našli vrednosti, ako nisu, ispisujemo poruku da su uneti pogrešni ID-jevi, a ako jesu,

vršimo upis nove ocene u bazu podataka metodom novaOcena(id predmeta, id učenika, datum, ocena) sledećim kodom.

```
boolean oo =
databaseHelper.novaOcena(txtPredmetID.getText().toString().trim(),
txtUcenikID.getText().toString().trim(),
txtDatum.getText().toString().trim(), txtOcena.getText().toString().trim());
if (oo) {
    Toast.makeText(OceneActivity.this, "Uspešno registrovana ocena",
Toast.LENGTH_LONG).show();
    clear();
} else {
    Toast.makeText(OceneActivity.this, "Učenik već ima ocenu iz izabranog
predmeta! Možete je ažurirati", Toast.LENGTH_LONG).show();
}
```

Na dugme ažuriraj takođe postavljamo osluškivač, ponovo proveravamo da li su unete potrebene vrednosti u sva polja, i onda kreiramo tri cursor-a jedan da nadje predmet, drugi da nađe učenika i treći da nađe ocenu, porveravamo da li su svi cursor-i našli vrednosti u bazi odnosno da li učenik već ima ocenu iz izabranog predmeta ukoliko je ispunjen uslov ispisujemo poruku da su uneti pogrešni ID-jevi, a ako jeuslov ispunjen vršimo ažuriranje ocene u bazu podataka sledećim kodom.

```
databaseHelper.updateOcena(Integer.parseInt(txtPredmetID.getText().toString()
.trim()), Integer.parseInt(txtUcenikID.getText().toString().trim()),
txtDatum.getText().toString().trim(),
txtOcena.getText().toString().trim());
Toast.makeText(OceneActivity.this, "Uspešno ažurirana ocena",
Toast.LENGTH_LONG).show();
```

updateOcena je metoda koja je definisana u klasi DatabaseHelper.

Za dugme prikaži listu učenika koristimo isti kod kao i u aktivnosti UceniciActivity.

Za dugme moji predmeti koristimo isti kod kao i u aktivnosti PredmetiActivity samo zamenimo metodu getPredmetiData sa getMojiPredmetiData i u nju unesemo id vrednost nastavnika.

Za dugme prikaži listu ocena takođe je skoro isti kod samo se koristi metoda getOceneData();

Na polje ID predmeta stavljamo osluškivač na promenu teksta, gde pokupimo vrednosti na isti način kao što smo već objasnili prethodno, na osnovu ID-ja predmeta nađemo njegov naziv i zatim ispišemo u toast poruci koji je predmet izabran u trenutku unošenja vrednosti u polje.

Istu stvar radimo i za polje ID učenika, na osnovu ID-ja pokupimo vrednosti iz baze i potom ispišemo ime i prezime učenika čiji je ID unešen u polje.

4.2.9. PosetilacActivity

Izgled ove aktivnosti možemo videti na slici 18.

Ova klasa koristi RecyclerView kako bi prikazala ocene i ostale podatke iz baze podataka posetiocu.

```
Intent intent = getIntent();  
usr = intent.getStringExtra("usr");  
pass = intent.getStringExtra("pass");
```

Prvo pokupimo vrednosti username i password iz intent-a sa Login stranice, zatim razdvojimo username na ime i prezime.

```
String[] split = usr.split("\\s+");  
ime = split[0];  
prezime = split[1];
```

Kreiramo četiri ArrayList-e za naziv predmeta, nastavnika, datum i ocenu.

```
datum = new ArrayList<>();  
nastavnik = new ArrayList<>();  
predmet = new ArrayList<>();  
ocena = new ArrayList<>();
```

Pozovemo metodu upisiVrednosti(); koja izgleda ovako.

```
void upisiVrednosti(){
    Cursor cursor = databaseHelper.ispisiOcene(ime, prezime, pass);
    if(cursor.getCount()==0){
        Toast.makeText(PosetilacActivity.this, "Nema ocena za prikaz",
        Toast.LENGTH_LONG).show();
    } else{
        while (cursor.moveToNext()){
            datum.add(cursor.getString(0));
            nastavnik.add(cursor.getString(12)+" "+cursor.getString(13));
            predmet.add(cursor.getString(6));
            ocena.add(cursor.getString(3));
        }
    }
}
```

Dakle u naše ArrayList-e dodajemo vrednosti koristeći .add() koje cursor pokupi iz baze podataka metodom ispisiOcene(ime, prezime, pass).

I povezujemo se sa klasom CustomAdapter.

```
customAdapter = new CustomAdapter(PosetilacActivity.this, datum, nastavnik,
predmet, ocena);
recyclerView.setAdapter(customAdapter);
recyclerView.setLayoutManager(new
LinearLayoutManager(PosetilacActivity.this));
```

4.2.10. CustomAdapter

Klasa CustomAdapter nasleđuje klasu RecyclerView.Adapter klasu.

```
public class CustomAdapter extends  
RecyclerView.Adapter<CustomAdapter.MyViewHolder>
```

Inicijalizujemo context i ArrayList-e.

```
private Context context;  
private ArrayList datum, nastavnik, predmet, ocena;
```

Zatim kreiramo konstruktor.

```
CustomAdapter(Context context, ArrayList datum, ArrayList nastavnik,  
ArrayList predmet, ArrayList ocena){  
    this.context = context;  
    this.datum = datum;  
    this.nastavnik = nastavnik;  
    this.predmet = predmet;  
    this.ocena = ocena;  
}
```

Sledi metoda onCreateViewHolder u kojem kreiramo LayoutInflater koji dodajemo u View a layout koji dodajemo je „my_row.xml“ u tom fajlu nam se nalazi izgled jednog reda u RecyclerView-u koji je realizovan kao CardView.

```
public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int i) {  
    LayoutInflater inflater = LayoutInflater.from(context);  
    View view = inflater.inflate(R.layout.my_row, parent, false);  
  
    return new MyViewHolder(view);  
}
```

Klasa MyViewHolder nasleđuje klasu RecyclerView.ViewHolder, i u njoj povezujemo preko ID vrednosti TextView-ove iz fajla „my_row.xml“ sa našim ArrayList-ama.

```

public class MyViewHolder extends RecyclerView.ViewHolder {

    TextView txtDatum, txtNastavnik, txtPredmet, txtOcena;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        txtDatum = itemView.findViewById(R.id.txtDatum);
        txtNastavnik = itemView.findViewById(R.id.txtNastavnik);
        txtPredmet = itemView.findViewById(R.id.txtPredmet);
        txtOcena = itemView.findViewById(R.id.txtOcena);
    }
}

```

U metodi onBindViewHolder popunjavamo podatke iz naših ArrayList-a u TextView-ove koji se nalaze u CardView-u fajla „my_row.xml“.

```

public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
    holder.txtDatum.setText(String.valueOf(datum.get(position)));
    holder.txtNastavnik.setText(String.valueOf(nastavnik.get(position)));
    holder.txtPredmet.setText(String.valueOf(predmet.get(position)));
    holder.txtOcena.setText(String.valueOf(ocena.get(position)));
}

```

5. Zaključak i dalji razvoj aplikacije

Ovaj projekat je kreiran za potrebe ispita iz predmeta Programiranje mobilnih aplikacija, aplikacija pruža osnovne mogućnosti korisnicima ali da bi se koristila u realnim uslovima potrebno ju je unaprediti.

Ideje za dalji razvoj su:

- Kreiranje više razreda (godina) kojima bi određeni predmeti i učenici pripadali,
- Dodavanje mogućnosti da se upišu opravdani i neopravdani izostanci,
- Računanje prosečne ocene za svakog učenika i na osnovu toga mu dodeliti uspeh,
- Mogućnost prepiske sa nastavnikom (slanje poruka),
- Mogućnost pisanja beleški koje bi nastavniku pomogle da vodi evidenciju o učenicima...

6. Literatura

- 1) FIN moodle portal, Računarska tehnika i softversko inženjerstvo, VII semestar, Programiranje mobilnih aplikacija: <http://moodle.fink.rs/course/view.php?id=982>
10.08.2021. (17:23)
- 2) GeeksForGeeks, kreiranje splash screen-a u android studiju: <https://www.geeksforgeeks.org/android-creating-a-splash-screen/>
14.08.2021. (13:48)