Elektrotehnički fakultet u Sarajevu

ANDROID I BAZE PODATAKA

SEMINARSKI RAD

Faris Čakarić
Azra Ibrahimović
Eman Jašarević
Ajdin Kahrović
Kerim Rožajac

Sadržaj

0. Uvod

U vrijeme kada fokus razvoja aplikativnog softvera je na razvoju mobilnih aplikacija, neizostavan dio razmatranja u okviru nauke o bazama podataka su baze podataka za mobilne aplikacije. Upravo zbog toga naglasak ovog rada će biti na bazama podataka za mobilne aplikacije na Android operativnom sistemu, kao najpopularnijem operativnom sistemu za mobilne uređaje.

Razvoj baza podataka za Android mobilne aplikacije može ići u smjeru razvoja lokalne baze podataka ili u smjeru razvoja eksterne baze podataka. Lokalna baza podataka se pokreće na istom uređaju na kojem je instalirana aplikacija, dok eksternoj bazi podataka aplikacija pristupa putem mreže i skupine protokola za prijenos podataka. Ovaj rad upravo opisuje način razvoja baze podataka, način povezivanja sa mobilnom aplikacijom i Android bibliotečnu podršku za rad sa bazama podataka, i to na način da prvo razmatra lokalnu, a zatim eskternu bazu podataka. Naravno, oba slučaja su upotpunjena odgovarajućim primjerima koji trebaju ilustrirati teoretski opisane principe i metode.

1. Andorid i lokalna baza podataka

SQLite je *opensource* bibilioteka koja implementira samoodržive, bez serverske, bez konfiguracijske i transakcijske alate SQL baze podataka.
SQLite je ugrađeni alat SQL baze podataka, koji za razliku od drugih SQL baza, nema odvojene serverske procese, već vrši čitanje i pisanje direktno u fajlove na disku. Baza podataka koja uključuje niz tabela, indeksa, trigera i pogleda se nalazi u jednom fajlu. Format u kom je pisana baza je cross-platform, što znači da se može nesmetano kopirati između 32-bitnih i 64-bitnih sistema, te big-endian i little-endian arhitektura.

SQLite je ugrađen u svaki Android uređaj. SQLite bazu podataka na Android uređajima nije potrebno dodatno administrirati, niti podešavati postavke. Potrebno je samo kreirati SQL upite za kreiranje i ažuriranje baze podataka. Ostali posao automatski obavlja Android platforma. Pristup bazi podrazumijeva pristup file sistemu. Kako ovo može biti sporo, preporučeno je asinhrono izvođenje operacija. Kada aplikacija kreira bazu, ona se automatski spašava u DATA/data/APP_NAME/database/FILENAME, gdje je DATA putanja koju vraća Environment.getDataDirectory() metoda, APP_NAME je ime aplikacije, FILENAME je ime baze podataka koje je specificirano unutar aplikacije.

1.1. Baza i model podataka

MySQLHelper klasa je odgovorna za kreiranje baze podataka. Za demonstraciju rada android aplikacije sa SQLite bazom podataka, kreirana je baza vozaci.db i unutar nje tabela TABLE_VOZACI. Tabela sadrži dvije kolone: COLUMN_ID i COLUMN_IME. Metoda onUpgrade() briše postojeće podatke iz baze i zatim kreira tabelu.

```
package com.tim2.baze.seminarski.util;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
```

```
public class MySQLiteHelper extends SQLiteOpenHelper {

      public static final String TABLE_VOZACI = "vozaci";
      public static final String COLUMN_ID = "id";
      public static final String COLUMN_IME = "ime";

      private static final String DATABASE_NAME = "vozaci.db";
      private static final int DATABASE_VERSION = 1;

         // iskaz za kreiranje baze
      private static final String DATABASE_CREATE =
      "create table " + TABLE_VOZACI +
      "(" + COLUMN_ID + " integer primary key autoincrement, " +
      COLUMN_IME + " text not null);";

      public MySQLiteHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
      }

      @Override
      public void onCreate(SQLiteDatabase database) {
            database.execSQL(DATABASE_CREATE);
      }

      @Override
            public void onUpgrade(SQLiteDatabase db, int oldVersion, int
            newVersion) {
            Log.w(MySQLiteHelper.class.getName(), "Upgrading database
            from version " + oldVersion + " to " + newVersion + ", which
            will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_VOZACI);
            onCreate(db);
      }
}
```

Klasa Vozac je model po kome će se kreirati objekti, te sadrži sve potrebne podatke koji će biti pohranjeni u bazu.

```
package com.tim2.baze.seminarski.dto;

public class Vozac {
   private long id;
   private String ime;

   public long getId() {
       return id;
   }

   public void setId(long id) {
       this.id = id;
```

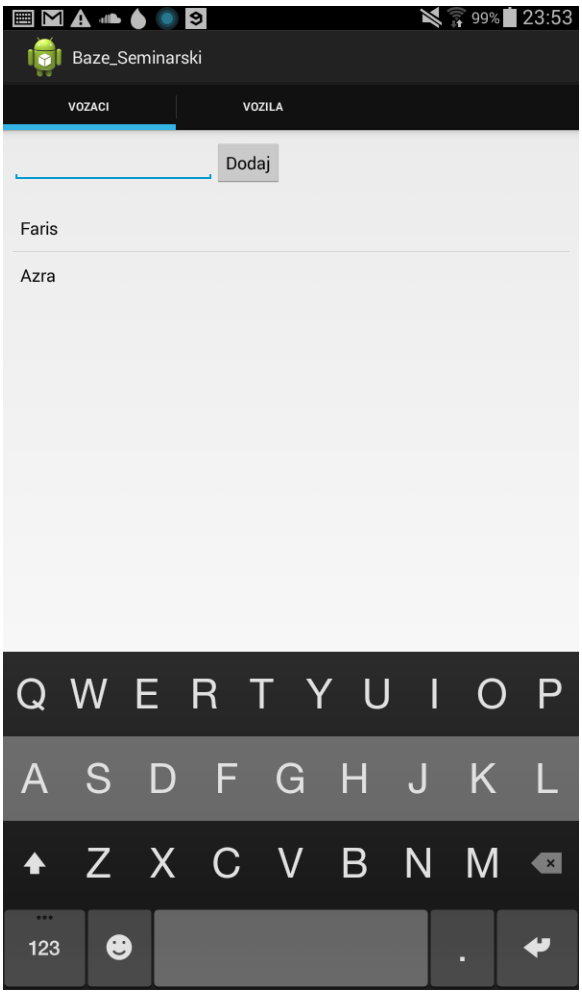
```
    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }
    @Override
    public String toString() {
        return ime;
    }
}
```

Za pristup i upravljanje podacima koristi se DAO klasa (*eng. data access object*). DAO klasa je odgovorna za uspostavu konekcije na bazu, pristup podacima i njihovo modificiranje. Objekte baze podataka pretvara u Java objekte radi lakšeg upravljanja.
Za potrebe testne baze podataka, kreirana je klasa VozacDataSource, koja sadrži konekciju na bazu, dodavanje vozača, izmjenu i njihovo brisanje iz baze, te metodu kojom se vraćaju svi podaci iz tabele.

```
package com.tim2.baze.seminarski.util;

import java.util.ArrayList;
import com.tim2.baze.seminarski.dto.Vozac;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class VozacDataSource {
      private SQLiteDatabase database;
      private MySQLiteHelper dbHelper;
      private   String[]   allColumns   =   {   MySQLiteHelper.COLUMN_ID,
      MySQLiteHelper.COLUMN_IME
};

public VozacDataSource(Context context) {
      dbHelper = new MySQLiteHelper(context);
}

public void open() throws SQLException {
      database = dbHelper.getWritableDatabase();
}

public void close() {
       dbHelper.close();
}

public Vozac createVozac(String v) {
```

```
      ContentValues values = new ContentValues();
      values.put(MySQLiteHelper.COLUMN_IME, v);
      long insertId = database.insert(MySQLiteHelper.TABLE_VOZACI, null,
      values);
      Cursor   cursor   =   database.query(MySQLiteHelper.TABLE_VOZACI,
      allColumns, MySQLiteHelper.COLUMN_ID + " = " + insertId, null,
      null, null, null);
       cursor.moveToFirst();
       Vozac novi = cursorToVozac(cursor);
       cursor.close();
       return novi;
}

public void deleteVozac(Vozac v) {
      long id = v.getId();
      database.delete(MySQLiteHelper.TABLE_VOZACI,
      MySQLiteHelper.COLUMN_ID + " = " + id, null);
}

public boolean updateVozac(long l, String novo) {
      ContentValues args = new ContentValues();
      args.put(MySQLiteHelper.COLUMN_IME, novo);
      return database.update(MySQLiteHelper.TABLE_VOZACI,
      args, MySQLiteHelper.COLUMN_ID + "=" + l, null) > 0;
}

public ArrayList<Vozac> getAll() {
      ArrayList<Vozac> svi = new ArrayList<Vozac>();
      Cursor   cursor   =   database.query(MySQLiteHelper.TABLE_VOZACI,
      allColumns, null, null, null, null, null);

       cursor.moveToFirst();
       while (!cursor.isAfterLast()) {
           Vozac v = cursorToVozac(cursor);
           svi.add(v);
           cursor.moveToNext();
       }
       cursor.close();
       return svi;
}

private Vozac cursorToVozac(Cursor cursor) {
       Vozac v = new Vozac();
       v.setId(cursor.getLong(0));
       v.setIme(cursor.getString(1));
       return v;
}
```

1.2. Izgled aplikacije



Slika 1 – Dodavanje/brisanje/izmjena vozača u SQLite bazi podataka

2. Android i eksterna baza podataka

U drugom dijelu ovog rada opisano je povezivanje Android aplikacije sa eksternom bazom podataka. Za razliku od MySqlite baze podataka, prilikom povezivanja Android aplikacije sa eksternom bazom, potrebno je kreirati bazu podataka (u ovom slučaju MySQL), i koristiti web servise (najčešće napisane u .php programskom jeziku) koji će vršiti konekciju sa bazom i vršiti manipulaciju sa podacima iz baze. Na WAMP serveru je napravljena baza "test", i u njoj tabela "vozila". Tabela vozila se sastoji od kolona: idvozila, vrstavozila, idvozaca. Prilikom pisanja web servisa korišten je REST (Representational State Transfer) standard. U ovom standardu koriste se četiri osnovne HTTP metode (GET, POST, PUT, DELETE).

- Za preuzimanje objekta koristi se metoda GET.
- Za kreiranje novog objekta koristi se metoda POST.

- Za izmjenu postojećeg objekta koristi se metoda PUT.
- Za brisanje objekta koristi se HTTP metoda DELETE.

Napravljen je jedan servis s navedene četiri metode.

Web servis "vozila.php" dobavlja podatke o svim vozilima pohranjenim u bazi, i vraća ih u JSON formatu. Da bi se vršila manipulacija sa dobijenim podacima, potrebno je izvršiti konverziju podataka iz JSON formata u tip pogodan za Android aplikaciju (u ovom radu je korištena lista objekata tipa Vozilo).

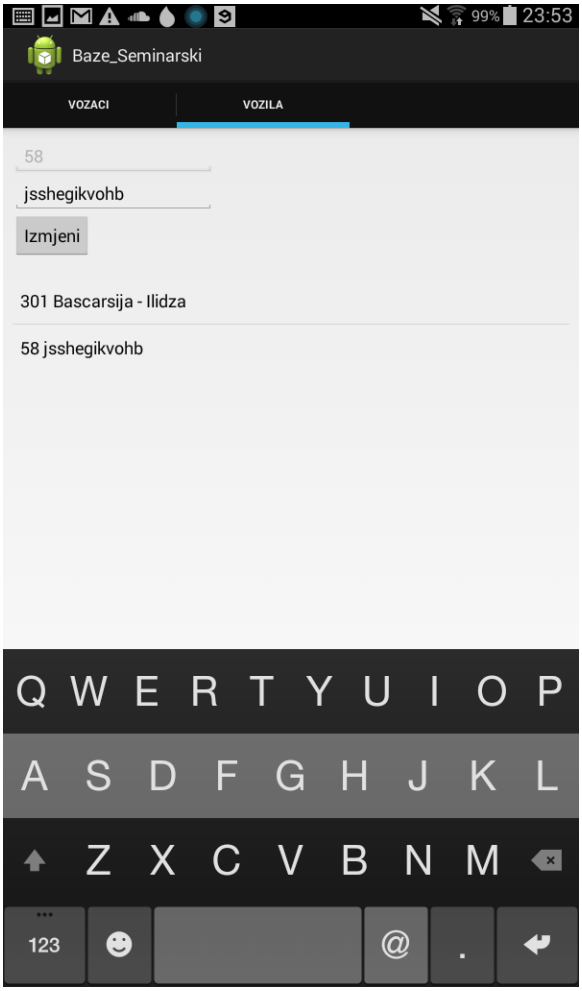
Klasa AsyncTask omogućava pravilnu i jednostavnu upotrebu UI niti. Ona omogućava da se izvrše pozadinske operacije i rezultat tih operacija prikaze na UI niti bez korištenja pravih niti. U našem slučaju ćemo iskoristiti AsyncTask za komunikaciju sa web servisom.
Klasa koja nasljeđuje ovu klasu mora implementirati dvije metode - doInBackground u kojoj je potrebno kontaktirati web servis, i onPostExecute koja se poziva kada se dobije odgovor od servisa.
Za svaku od operacija - get, update, remove, delete potrebno je kreirati po jednu klasu koja nasljeđuje AsyncTask. Tako naprimjer, za dohvat podataka iz baze:

```
class UcitajVozila extends AsyncTask<Void, Void, String> {

@Override
protected String doInBackground(Void... params) {
      HttpGet httpGet = new HttpGet("http://178.77.23.9/vozila.php");
      try {
            HttpClient httpclient = new DefaultHttpClient();
            HttpResponse response = httpclient.execute(httpGet);
            return EntityUtils.toString(response.getEntity(),
            HTTP.UTF_8);
      }
      catch (Exception e) {
            e.printStackTrace();
      }
      return null;
}

@Override
protected void onPostExecute(String response) {
      try {
            final JSONArray jsonObj = new
            JSONObject(response).getJSONArray("vozila");
            vozila.clear();
            for (int i = 0; i < jsonObj.length(); i++) {
                  Vozilo vozilo = new Vozilo();
                  vozilo.setBrojVozila(jsonObj.getJSONObject(i).getInt("
                  broj"));
                  vozilo.setLinija(jsonObj.getJSONObject(i).getString("l
                  inija"));
                  vozila.add(vozilo);
            }
            osvjeziListu();
      } catch (Exception e) {
            e.printStackTrace();
```

```
	}
}
}

class DodajVozilo extends AsyncTask<String, Void, String> {

@Override
protected String doInBackground(String... params) {
	String broj = params[0];
	String linija = params[1];
	HttpClient httpclient = new DefaultHttpClient();
	HttpPost httppost = new
	HttpPost("http://178.77.23.9/vozila.php");
	try {
		List<NameValuePair> nameValuePairs = new
		ArrayList<NameValuePair>(1);
		nameValuePairs.add(new BasicNameValuePair("broj", broj));
		nameValuePairs.add(new BasicNameValuePair("linija",
		linija));
		httppost.setEntity(new
		UrlEncodedFormEntity(nameValuePairs));

		HttpResponse response = httpclient.execute(httppost);
		Log.d("response",
		EntityUtils.toString(response.getEntity(), HTTP.UTF_8));
	} catch (Exception e) {
		e.printStackTrace();
	}
	return null;
}

@Override
protected void onPostExecute(String response) {
	try {
		new UcitajVozila().execute();
	} catch (Exception e) {
		e.printStackTrace();
	}
}
}

class ObrisiVozilo extends AsyncTask<Vozilo, Void, String> {

@Override
protected String doInBackground(Vozilo... params) {
	Vozilo vozilo = params[0];
	HttpClient httpclient = new DefaultHttpClient();
	HttpDelete httpdelete = new
	HttpDelete("http://178.77.23.9/vozila.php?=" +
	String.valueOf(vozilo.getBrojVozila()));
	try {
		HttpResponse response = httpclient.execute(httpdelete);
```

```
      } catch (Exception e) {
            e.printStackTrace();
      }

      return null;
}

@Override
protected void onPostExecute(String response) {
      try {
            new UcitajVozila().execute();
      } catch (Exception e) {
            e.printStackTrace();
      }}
}

class IzmijeniVozilo extends AsyncTask<Vozilo, Void, String> {

@Override
protected String doInBackground(Vozilo... params) {
      Vozilo vozilo = params[0];
      HttpPut httpPut = new HttpPut("http://178.77.23.9/vozila.php");
      try {
            HttpClient httpclient = new DefaultHttpClient();
            List<NameValuePair> nameValuePairs = new
            ArrayList<NameValuePair>(1);
            nameValuePairs.add(new BasicNameValuePair("broj",
            String.valueOf(vozilo.getBrojVozila())));
            nameValuePairs.add(new BasicNameValuePair("linija",
            vozilo.getLinija()));
            httpPut.setEntity(new
            UrlEncodedFormEntity(nameValuePairs));

            HttpResponse response = httpclient.execute(httpPut);
            return EntityUtils.toString(response.getEntity(),
            HTTP.UTF_8);
      } catch (Exception e) {
            e.printStackTrace();
      }
      return null;
}

@Override
protected void onPostExecute(String response) {
      try {
            new UcitajVozila().execute();
      } catch (Exception e) {
            e.printStackTrace();
      }
}
}
```

2.1. Izgled aplikacije



Slika 2 - Dodavanje/brisanje/izmjena vozača u eksternoj bazi podataka

3. Zaključak

Razvoj mobilnih aplikacija za Android operativne sisteme predstavlja najbrže rastući segment tržišta softver inježeringa. Zbog toga je od krucijalne važnosti ovladali svakim aspektom razvoja ovakvih aplikacija, pri čemu zasigurno centralno mjesto zauzima razvoj baza podataka za takve aplikacije. Namjena očitovana kroz cijeli ovaj rad je upravo da čitateljima pruži takvu vrstu znanja. Slijedom toga, ovaj rad opisuje standardne i najkorištenije metode razvoja baza podataka, kako lokalnih, tako i eksternih, a zatim i njihovog povezivanja sa mobilnim aplikacijama. Ovim se tema razvoja baza podataka za Android mobilne aplikacije zaokružuje, te sa jedne strane predstavlja teorijske osnove baza podataka u okviru razvoja mobilnih aplikacija, a sa druge strane predstavlja svojevrstan vodič za sve one koji se amaterski ili profesionalno bave razvojem softvera. Zbog toga ovaj rad, nadamo se, u potpunosti ostvaruje svoju namjenu.

4. Literatura

1. Accessing External Databases from Mobile Applications. Nagesh i Caicedo. Syracuse University.
2. Web tehnologije. Ribić, Samir. Elektrotehnički fakultet u Sarajevu. 2014.