ANDROID PROGRAMIRANJE

Vežba 8: Priprema za drugi kolokvijum

Kreirati Android aplikaciju za **veterinarsku ambulantu** koja ima sledeće funkcionalnosti:

- **Početni ekran:** Log in i sign up za veterinara. Za log in je potreban username i password, a kada želite da napravite novi nalog treba uneti ime, prezime, titulu (doktor veterine, veterinarski tehničar, administracija, pomoćno osoblje...) i kontakt email. Možete dodati i kontakt telefon u formatu +381 XXX XXXX (bonus 1p).
- **Glavni ekran** koji sadrži poruku pozdrava (npr. u labeli da piše Dobrodošli, dr Nikola!) i CRUD operacije za veterinarske preglede. Možete implementirati logiku takvu da ukoliko je ulogovani korisnik doktor, njemu piše dr ispred imena u poruci dobrodošlice, a ostali da budu oslovljeni samo po imenu (bonus 2p).
- Od Android komponenti potrebni su vam:
    - Textview
    - EditText
    - Button
    - RecycleView
    - DatePicker/TimePicker
    - ImageView
    - Notifikacije ili Toast za uspešno zakazivanje/otkazivanje pregleda.
- Na početnom ekranu aplikacije treba da stoji header slika generisana preko AI, nešto poput ove: https://imgur.com/a/YsAvG8G. Na samom kolokvijumu ćete dobiti gotove slike (planiram da stavim i logotip brenda ili ustanove za koju budemo radili ovakvu sličnu aplikaciju), ali za potrebe ove vežbe slobodno budite kreativni i napravite neki zanimljiv dizajn. Predlog rasporeda komponenti možete takođe pogledati u nastavku. Koristite i odgovarajuću paletu boja sa coolors.co, neka stilski sve bude što lepše sređeno.
- Što se samih operacija tiče, svaki ulogovani veterinar može da:
    - **Izlista sve preglede** koji su zakazani kod njega i klikom na dugme sortira ih prema datumu i vremenu. Svaki pregled ima podatke o ljubimcu, vlasniku (kontakt telefon i ime vlasnika), opisu slučaja, kao i datumu i vremenu kada je zakazan.
    - **Zakaže novi pregled,** tj. doda ga u bazu sa svim podacima i vodi računa da datum i vreme zakazivanja budu u budućnosti i da već nema pregleda za taj dan i to vreme kod tog radnika. Može se zakazati 1h posle zauzetog termina (bonus 2p).
    - **Otkaže postojeći pregled**, tj. izbriše ga iz baze podataka samo ukoliko je zakazan u budućnosti. Ova validacija je bitna, biće negativnih poena ako to nije urađeno.

- **Baza podataka** je svakako **SQLite,** i predlozi za tebele su **Veterinar, Ljubimac i Pregled**. Svaki ljubimac ima svoj naziv, rasu, uzrast i kontakt podatke vlasnika (ime i telefon). Kao što je već rečeno, svaki veterinar sadrži ime, prezime, titulu i kontakt podatke. Za pregled smo već u prethodnom koraku napomenuli šta je potrebno. Vodite računa o vezama. Svaki veterinar može imati više zakazanih pregleda za više različitih ljubimaca. Razmislite koje su vam relacije pogodne za predstavljanje takvih veza. Na kolokvijumu ćete dobiti unose za bazu, ali za potrebu ove vežbe neophodno je generisati bar 5 ljubimaca, bar 5 veterinara (u slučaju ove vežbe, veterinar je univerzalni naziv korišćen za sve radnike ambulante, da vas ne buni taj detalj) i bar 10 pregleda, ili "iz glave", ili preko ChatGPT-a.
- Takođe, treba implementirati i **log out** operaciju. Na korisničkom panelu treba dodati log out dugme koje "gasi" sve otvorene sesije, konekcije i vraća na početni ekran kako bi mogao možda neki drugi veterinar da se uloguje i upravlja svojim pregledima.

Primer baze:

Korisničko ime veterinara je primarni ključ i pri logovanju u aplikaciju prikazaćete sve preglede gde je veterinar_id jednak korisničkom imenu ulogovanog korisnika. Njegovo ime ćete iskoristiti za pozdravnu poruku. Datum i vreme pregleda mogu biti dodati i preko **DATETIME** tipa, gde biste imali ovakav format: ('drnikola', 1, 'Rutinska vakcinacija', '2024-01-15 10:30:00').

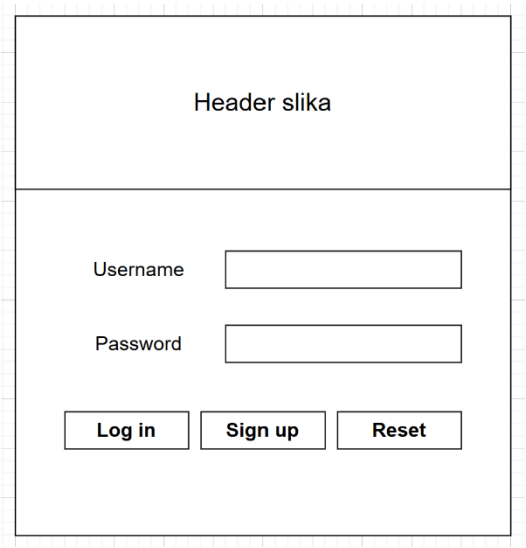
```
CREATE TABLE Veterinar (
    username TEXT NOT NULL PRIMARY KEY,
    password TEXT NOT NULL,
    ime TEXT NOT NULL,
    prezime TEXT NOT NULL,
    titula TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    telefon TEXT
);
```

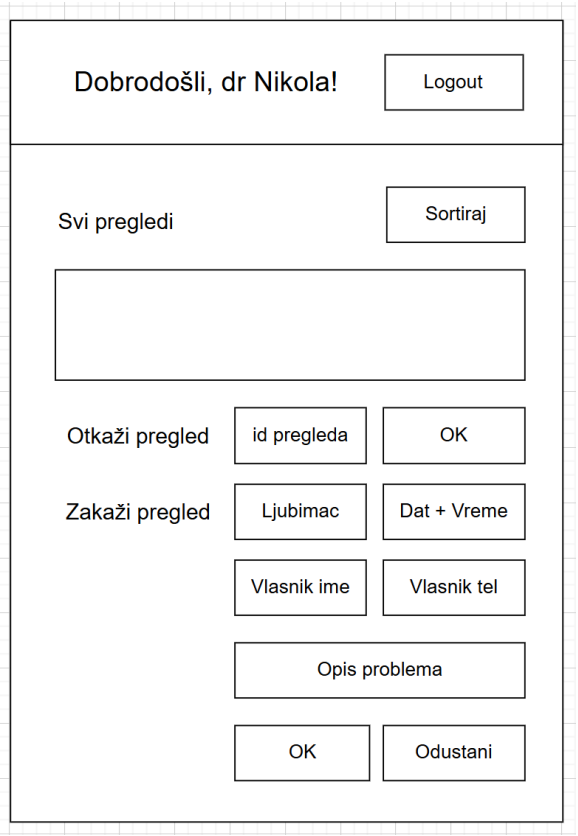
```
CREATE TABLE Ljubimac (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    naziv TEXT NOT NULL,
    rasa TEXT,
    uzrast INTEGER,
    vlasnik_ime TEXT NOT NULL,
    vlasnik_telefon TEXT NOT NULL,
);
```

```
CREATE TABLE Pregled (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    veterinar_id TEXT NOT NULL,
    ljubimac_id INTEGER NOT NULL,
    opis TEXT,
    datum TEXT NOT NULL,
    vreme TEXT NOT NULL,
    FOREIGN KEY(veterinar_id) REFERENCES Veterinar(username),
    FOREIGN KEY(ljubimac_id) REFERENCES Ljubimac(id)
);
```

```
-- Unosi za tabelu Veterinar

INSERT INTO Veterinar (username, password, ime, prezime, titula, email, telefon)
VALUES

('drnikola', 'password123', 'Nikola', 'Jovanović', 'Doktor veterine',
'nikola.jovanovic@vetclinic.rs', '+381 63 1234567'),

('vetana', 'securepass', 'Ana', 'Petrović', 'Veterinarski tehničar',
'ana.petrovic@vetclinic.rs', '+381 64 9876543'),

('adminmilena', 'adminpass', 'Milena', 'Stojanović', 'Administracija',
'milena.stojanovic@vetclinic.rs', '+381 62 5432198'),

('tecdarko', 'darko2024', 'Darko', 'Popović', 'Pomoćno osoblje',
'darko.popovic@vetclinic.rs', NULL),

('drmarija', 'pass4marija', 'Marija', 'Lazarević', 'Doktor veterine',
'marija.lazarevic@vetclinic.rs', '+381 65 4321567');


-- Unosi za tabelu Ljubimac

INSERT INTO Ljubimac (naziv, rasa, uzrast, vlasnik_ime, vlasnik_telefon) VALUES

('Luna', 'Labrador', 3, 'Milan Petrović', '+381 63 1112223'),

('Milo', 'Beagle', 5, 'Ivana Jovanović', '+381 64 2233445'),

('Bella', 'Sibirski haski', 2, 'Jelena Nikolić', '+381 65 3344556'),

('Charlie', 'Persijska mačka', 4, 'Marko Antić', '+381 62 4455667'),

('Max', 'Nemački ovčar', 1, 'Ana Živanović', '+381 60 5566778');


-- Unosi za tabelu Pregled

INSERT INTO Pregled (veterinar_id, ljubimac_id, opis, datum, vreme) VALUES

('drnikola', 1, 'Rutinska vakcinacija', '2025-01-15', '10:30'),

('vetana', 2, 'Kontrola ušiju', '2025-01-16', '11:00'),

('drnikola', 3, 'Pregled nakon povrede', '2025-01-17', '12:45'),

('drmarija', 4, 'Redovni godišnji pregled', '2025-01-18', '09:15'),

('tecdarko', 5, 'Isporuka lekova vlasniku', '2025-01-19', '13:30'),

('drnikola', 1, 'Kontrola težine', '2025-01-20', '14:00'),

('vetana', 3, 'Savetovanje o ishrani', '2025-01-21', '15:30'),

('drmarija', 2, 'Odstranjivanje parazita', '2025-01-22', '16:00'),

('tecdarko', 4, 'Dostava hrane', '2025-01-23', '17:30'),

('drmarija', 5, 'Prva vakcina šteneta', '2025-01-24', '08:00');
```

Ostatak ubacite sami pri samostalnoj izradi ove vežbe. Zbog lakšeg sortiranja i pretraživanja, predlažem da koristite DATETIME format kada pravite bazu. Sledi predlog za dizajn i delovi koda.





Primer koda, Main Activity:

```
package com.example.vetclinic;

import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText usernameInput, passwordInput;
    Button loginButton, signUpButton;
    SQLiteOpenHelper openHelper;
    SQLiteDatabase db;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    usernameInput = findViewById(R.id.usernameInput);
    passwordInput = findViewById(R.id.passwordInput);
    loginButton = findViewById(R.id.loginButton);
    signUpButton = findViewById(R.id.signUpButton);

    openHelper = new DatabaseHelper(this);

    loginButton.setOnClickListener(v -> {
        String username = usernameInput.getText().toString().trim();
        String password = passwordInput.getText().toString().trim();

        db = openHelper.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM Veterinar WHERE username=?
                            AND password=?", new String[]{username, password});
        if (cursor != null && cursor.moveToFirst()) {
            String name = cursor.getString(cursor.getColumnIndex("ime"));
            String title = cursor.getString(cursor.getColumnIndex("titula"));
            cursor.close();

            Intent intent = new Intent(MainActivity.this, HomeActivity.class);
            intent.putExtra("username", username);
            intent.putExtra("name", name);
            intent.putExtra("title", title);
            startActivity(intent);
        } else {
            Toast.makeText(MainActivity.this, "Pogrešno korisničko ime ili
                                    lozinka", Toast.LENGTH_SHORT).show();
        }
    });

    signUpButton.setOnClickListener(v -> {
        Intent intent = new Intent(MainActivity.this, SignUpActivity.class);
        startActivity(intent);
    });  } }
```

Primer koda, Signup Activity:

```
package com.example.vetclinic;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class SignupActivity extends AppCompatActivity {

    private EditText etUser, etPass, etIme, etPrezime, etTitula, etEmail, etTel;
    private Button btnRegister;
    private DatabaseHelper db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);

        etUser = findViewById(R.id.etUsername);
        etPass = findViewById(R.id.etPassword);
        etIme = findViewById(R.id.etIme);
        etPrezime = findViewById(R.id.etPrezime);
        etTitula = findViewById(R.id.etTitula);
        etEmail = findViewById(R.id.etEmail);
        etTel = findViewById(R.id.etTelefon);
        btnRegister = findViewById(R.id.btnRegister);

        db = new DatabaseHelper(this);

        btnRegister.setOnClickListener(v -> {
            String user = etUser.getText().toString();
            String pass = etPass.getText().toString();
            String ime = etIme.getText().toString();
            String prezime = etPrezime.getText().toString();
            String titula = etTitula.getText().toString();
```

```
            String email = etEmail.getText().toString();
            String tel = etTel.getText().toString();

            if (db.registerVeterinar(user, pass, ime, prezime, titula, email, tel))
            {
                Toast.makeText(this, "Uspesna registracija!",
                                        Toast.LENGTH_SHORT).show();
                finish();
            } else {
                Toast.makeText(this, "Greska!", Toast.LENGTH_SHORT).show();
            }
        });
    }}
```

Primer koda, Home Activity, kada je korisnik ulogovan:

```
package com.example.vetclinic;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class HomeActivity extends AppCompatActivity {

    TextView welcomeMessage;
    Button logoutButton, addAppointmentButton, deleteAppointmentButton;
    RecyclerView appointmentsList;
    EditText petIdInput, descriptionInput, dateInput, timeInput, deleteIdInput;
```

```
// Ovo može biti i lista stringova ili da napravite custom klasu Appointment
// sa svim neophodnim atributima, getterima i setterima i onda radite preko nje
ArrayList<Appointment> appointments;
AppointmentsAdapter adapter;
SQLiteDatabase db;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);

    // Povezivanje elemenata interfejsa
    welcomeMessage = findViewById(R.id.welcomeMessage);
    logoutButton = findViewById(R.id.logoutButton);
    addAppointmentButton = findViewById(R.id.addAppointmentButton);
    deleteAppointmentButton = findViewById(R.id.deleteAppointmentButton);
    appointmentsList = findViewById(R.id.appointmentsList);
    petIdInput = findViewById(R.id.petIdInput);
    descriptionInput = findViewById(R.id.descriptionInput);
    dateInput = findViewById(R.id.dateInput);
    timeInput = findViewById(R.id.timeInput);
    deleteIdInput = findViewById(R.id.deleteIdInput);

    Intent intent = getIntent();
    String username = intent.getStringExtra("username");
    String name = intent.getStringExtra("name");
    String title = intent.getStringExtra("title");

    if (title.contains("Doktor")) {
        welcomeMessage.setText("Dobrodošli, dr " + name + "!");
    } else {
        welcomeMessage.setText("Dobrodošli, " + name + "!");
    }

    db = new DatabaseHelper(this).getReadableDatabase();
    appointments = new ArrayList<>();
    loadAppointments(username);
```

```
        appointmentsList.setLayoutManager(new LinearLayoutManager(this));
        adapter = new AppointmentsAdapter(this, appointments, db);
        appointmentsList.setAdapter(adapter);

        // Funkcionalnost za dodavanje pregleda
        addAppointmentButton.setOnClickListener(v -> addAppointment(username));

        // Funkcionalnost za brisanje pregleda
        deleteAppointmentButton.setOnClickListener(v -> deleteAppointment());

        // Funkcionalnost za odjavu
        logoutButton.setOnClickListener(v -> {
            Intent outIntent = new Intent(HomeActivity.this, MainActivity.class);
            startActivity(outIntent);
            finish();
        });
    }

    private void loadAppointments(String username) {
        Cursor cursor = db.rawQuery("SELECT * FROM Pregled WHERE veterinar_id=?
                        ORDER BY datum, vreme", new String[]{username});
        while (cursor.moveToNext()) {
            int id = cursor.getInt(cursor.getColumnIndex("id"));
            String petId = cursor.getString(cursor.getColumnIndex("ljubimac_id"));
            String description = cursor.getString(cursor.getColumnIndex("opis"));
            String date = cursor.getString(cursor.getColumnIndex("datum"));
            String time = cursor.getString(cursor.getColumnIndex("vreme"));
            appointments.add(new Appointment(id, petId, description, date, time));
        }
        cursor.close();
    }

    private void addAppointment(String username) {
        String petId = petIdInput.getText().toString().trim();
        String description = descriptionInput.getText().toString().trim();
         String owner = ownerInput.getText().toString().trim();
        String date = dateInput.getText().toString().trim();
        String time = timeInput.getText().toString().trim();
```

```
// Zbog jednostavosti ovde je vlasnik predstavljen samo sa owner,
// inače vam treba i ime i kontakt vlasnika, dva zasebna polja
 if (petId.isEmpty() || description.isEmpty() || date.isEmpty() ||
     time.isEmpty() || owner.isEmpty()) {
      Toast.makeText(this, "Sva polja moraju biti popunjena!"
       Toast.LENGTH_SHORT).show();
      return;
  }

// Validacija datuma i vremena pregleda
try {
  String dateTimeString = date + " " + time; // Format je "yyyy-MM-dd HH:mm:ss"
  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
  sdf.setLenient(false);
  Date appointmentDate = sdf.parse(dateTimeString);
  Date now = new Date();

  if (appointmentDate.before(now)) {
      Toast.makeText(this, "Ne možete zakazati pregled u prošlosti!",
                      Toast.LENGTH_SHORT).show();
      return;
  }
  ContentValues values = new ContentValues();
  values.put("ljubimac_id", petId);
  values.put("opis", description);
  values.put("datum", date);
  values.put("vreme", time);
   values.put("vlasnik", owner);
  values.put("veterinar_id", username);

  long newRowId = db.insert("Pregled", null, values);
  if (newRowId != -1) {
      appointments.add(new Appointment((int) newRowId, petId, description,
                        owner, date, time));
      adapter.notifyItemInserted(appointments.size() - 1);
      Toast.makeText(this, "Pregled uspešno zakazan!",
                              Toast.LENGTH_SHORT).show();
```

```
    } else {
        Toast.makeText(this, "Greška pri zakazivanju pregleda.",
                       Toast.LENGTH_SHORT).show();
    }
  } catch (Exception e) {
    Toast.makeText(this, "Unesite ispravan format datuma i vremena!",
                  Toast.LENGTH_SHORT).show();
  }
}

private void deleteAppointment() {
    String deleteId = deleteIdInput.getText().toString().trim();
    if (deleteId.isEmpty()) {
        Toast.makeText(this, "Unesite ispravan ID pregleda za brisanje!",
                       Toast.LENGTH_SHORT).show();
        return;
    }
     // Provera da li je pregled u prošlosti
      Cursor cursor = db.rawQuery("SELECT datum, vreme FROM Pregled WHERE id=?",
                  new String[]{deleteId});
   if (cursor.moveToFirst()) {
      String date = cursor.getString(cursor.getColumnIndex("datum"));
      String time = cursor.getString(cursor.getColumnIndex("vreme"));

      try {
        String dateTimeString = date + " " + time;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        sdf.setLenient(false);
        Date appointmentDate = sdf.parse(dateTimeString);
        Date now = new Date();

        if (appointmentDate.before(now)) {
            Toast.makeText(this, "Ne možete otkazati pregled koji je bio u
                           prošlosti!", Toast.LENGTH_SHORT).show();
            cursor.close();
            return;
        }
```

```
            // Otkazivanje pregleda
            int rowsDeleted = db.delete("Pregled", "id=?", new String[]{deleteId});
            if (rowsDeleted > 0) {
                for (int i = 0; i < appointments.size(); i++) {
                    if (appointments.get(i).getId() == Integer.parseInt(deleteId))
                    {
                        appointments.remove(i);
                        adapter.notifyItemRemoved(i);
                        break;
                    }
                }
                Toast.makeText(this, "Pregled uspešno otkazan!",
                               Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Pregled sa unetim ID-jem ne postoji.",
                               Toast.LENGTH_SHORT).show();
            }
        } catch (Exception e) {
            Toast.makeText(this, "Greška pri proveri datuma pregleda.",
                           Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, "Pregled sa unetim ID-jem ne postoji.",
                       Toast.LENGTH_SHORT).show();
    }
    cursor.close();
}
```

Korisni resursi

https://www.youtube.com/watch?v=CWXFWyiuh3s&ab_channel=MiracleGroup

https://www.youtube.com/watch?v=nmwecyCDAi8&ab_channel=FilipVujovic

https://www.youtube.com/playlist?list=PLpnU7bxHjO4WDI22RuoRF4zn1URIxPqMC

https://www.youtube.com/watch?v=kvPrfp28HZg&ab_channel=JaysonMaglalang

https://www.youtube.com/watch?v=9t8VVWebRFM&ab_channel=CodeLabX