ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | С.В. Щекин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| Освоение работы с базами данных в мобильном приложении |
| по курсу: Программирование мобильных устройств |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Н.А. Костяков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

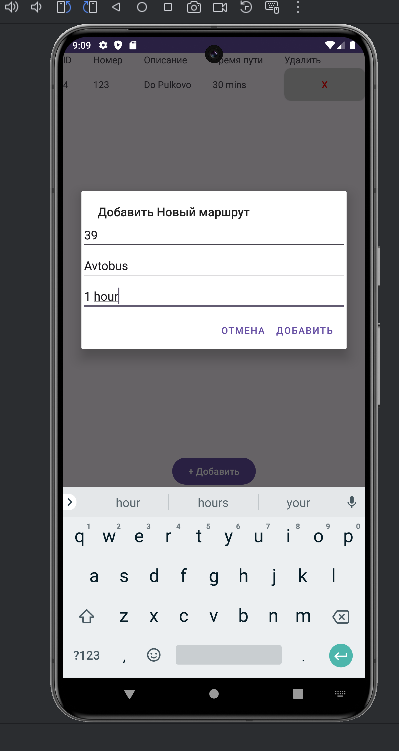
Санкт-Петербург 2024

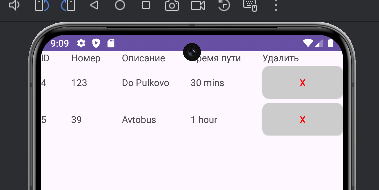
## Задание:

Разработать и отладить мобильное приложение, обеспечивающее подключение и работу с базой данных (например, SQLite для ОС Android). Приложение должно реализовывать создание, чтение, изменение и удаление отдельных записей.

## Ход работы







## Листинг

Api server

package com.example.laba81;  
  
import org.json.JSONArray;  
import retrofit2.Call;  
import retrofit2.http.GET;  
import org.json.JSONObject;  
import retrofit2.http.Body;  
import retrofit2.http.POST;  
import com.example.laba81.User;  
import java.util.List;  
import retrofit2.http.Header;  
import okhttp3.ResponseBody;  
import okhttp3.RequestBody;  
import retrofit2.http.Query;  
  
  
public interface ApiService {  
 @GET("education/read\_users")  
 Call<List<User>> fetchUsers();  
 @POST("/education/create\_user")  
 Call<Void> createUser(@Body RequestBody userData);  
 @GET("/education/delete\_user")  
 Call<ResponseBody> deleteUser(@Query("user\_id") String userId);  
  
 @POST("/education/update\_user")  
 Call<ResponseBody> updateUser(@Query("user\_id") String userId, @Body RequestBody userData);  
  
}

DBHelper

package com.example.laba81;  
  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.content.ContentValues;  
import android.database.Cursor;  
import java.util.ArrayList;  
import java.util.List;  
public class DatabaseHelper extends SQLiteOpenHelper {  
 private static final String *DATABASE\_NAME* = "users.db";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 // Конструктор  
 public DatabaseHelper(Context context) {  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 // Методы создания и обновления базы данных  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 // Создание таблицы пользователей  
 String createTableQuery = "CREATE TABLE users (" +  
 "\_id INTEGER PRIMARY KEY AUTOINCREMENT," +  
 "name TEXT," +  
 "surname TEXT," +  
 "birthday TEXT)";  
 db.execSQL(createTableQuery);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 // Обновление базы данных, если необходимо  
 }  
  
 // Методы работы с таблицей пользователей  
 public void addUser(User user) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put("name", user.getName());  
 values.put("surname", user.getSurname());  
 values.put("birthday", user.getBirthday());  
 db.insert("users", null, values);  
 db.close();  
 }  
  
 public List<User> getAllUsers() {  
 List<User> userList = new ArrayList<>();  
 SQLiteDatabase db = this.getReadableDatabase();  
 Cursor cursor = db.rawQuery("SELECT \* FROM users", null);  
 if (cursor.moveToFirst()) {  
 do {  
 User user = new User();  
 user.set\_id(cursor.getString(cursor.getColumnIndex("\_id")));  
 user.setName(cursor.getString(cursor.getColumnIndex("name")));  
 user.setSurname(cursor.getString(cursor.getColumnIndex("surname")));  
 user.setBirthday(cursor.getString(cursor.getColumnIndex("birthday")));  
 userList.add(user);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 db.close();  
 return userList;  
 }  
  
 public void deleteUser(String userId) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete("users", "\_id = ?", new String[]{userId});  
 db.close();  
 }  
  
 public void updateUser(User user) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put("name", user.getName());  
 values.put("surname", user.getSurname());  
 values.put("birthday", user.getBirthday());  
 db.update("users", values, "\_id = ?", new String[]{user.get\_id()});  
 db.close();  
 }  
}

Main activity

package com.example.laba81;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.util.Log;  
  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TableLayout;  
import android.widget.TableRow;  
import android.widget.TextView;  
import android.widget.Toast;  
import android.widget.EditText;  
import android.graphics.Color;  
  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.util.List;  
  
public class MainActivity extends AppCompatActivity {  
  
 private TableLayout tableLayout;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 tableLayout = findViewById(R.id.*tableLayout*);  
  
 Button addButton = findViewById(R.id.*addButton*);  
 addButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showAddUserDialog();  
 }  
 });  
  
 fetchUsers();  
 }  
  
 private void fetchUsers() {  
 DatabaseHelper databaseHelper = new DatabaseHelper(this);  
 List<User> userList = databaseHelper.getAllUsers();  
 displayUsers(userList);  
 }  
  
 private void displayUsers(List<User> userList) {  
 tableLayout.removeAllViews(); // Очистим таблицу перед добавлением новых пользователей  
  
 // Создаем заголовок таблицы  
 TableRow headerRow = new TableRow(this);  
  
 TextView idHeader = new TextView(this);  
 idHeader.setText("ID");  
 headerRow.addView(idHeader);  
  
 TextView nameHeader = new TextView(this);  
 nameHeader.setText("Имя");  
 headerRow.addView(nameHeader);  
  
 TextView surnameHeader = new TextView(this);  
 surnameHeader.setText("Фамилия");  
 headerRow.addView(surnameHeader);  
  
 TextView birthdayHeader = new TextView(this);  
 birthdayHeader.setText("Дата рождения");  
 headerRow.addView(birthdayHeader);  
  
 TextView deleteHeader = new TextView(this); // Новый заголовок для кнопки удаления  
 deleteHeader.setText("Удалить");  
 headerRow.addView(deleteHeader); // Добавляем заголовок кнопки удаления  
  
 tableLayout.addView(headerRow);  
 int rowMargin = 10;  
  
 // Добавляем пользователей в таблицу  
 for (final User user : userList) {  
 TableRow row = new TableRow(this);  
 row.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.*MATCH\_PARENT*, TableLayout.LayoutParams.*WRAP\_CONTENT*));  
 row.setPadding(0, rowMargin, 0, rowMargin);  
  
 TextView idTextView = new TextView(this);  
 idTextView.setText(user.get\_id());  
 row.addView(idTextView);  
  
 TextView nameTextView = new TextView(this);  
 nameTextView.setText(user.getName());  
 row.addView(nameTextView);  
  
 TextView surnameTextView = new TextView(this);  
 surnameTextView.setText(user.getSurname());  
 row.addView(surnameTextView);  
  
 TextView birthdayTextView = new TextView(this);  
 birthdayTextView.setText(user.getBirthday());  
 row.addView(birthdayTextView);  
  
 // Добавление кнопки "крестик" и обработчика события нажатия  
 Button deleteButton = new Button(this);  
 deleteButton.setText("X");  
  
 // Устанавливаем параметры макета для кнопки  
 TableRow.LayoutParams params = new TableRow.LayoutParams(TableRow.LayoutParams.*WRAP\_CONTENT*, TableRow.LayoutParams.*WRAP\_CONTENT*);  
 deleteButton.setLayoutParams(params);  
  
 deleteButton.setBackgroundResource(R.drawable.*rounded\_button\_background*);  
 deleteButton.setTextColor(Color.*RED*);  
 deleteButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 deleteUser(user.get\_id());  
 }  
 });  
  
 row.addView(deleteButton); // Добавляем кнопку удаления в строку таблицы  
  
 // Добавление обработчика события клика на строку  
 row.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 editUser(user.get\_id(), user.getName(), user.getSurname(), user.getBirthday());  
 }  
 });  
  
 tableLayout.addView(row);  
 }  
 }  
  
  
 private void editUser(final String userId, String name, String surname, String birthday) {  
 AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);  
 LayoutInflater inflater = this.getLayoutInflater();  
 View dialogView = inflater.inflate(R.layout.*edit\_user\_dialog*, null);  
 dialogBuilder.setView(dialogView);  
  
 final EditText nameEditText = dialogView.findViewById(R.id.*nameEditText*);  
 final EditText surnameEditText = dialogView.findViewById(R.id.*surnameEditText*);  
 final EditText birthdayEditText = dialogView.findViewById(R.id.*birthdayEditText*);  
  
 // Заполнение поля данными пользователя  
 nameEditText.setText(name);  
 surnameEditText.setText(surname);  
 birthdayEditText.setText(birthday);  
  
 dialogBuilder.setTitle("Редактировать пользователя");  
 dialogBuilder.setPositiveButton("Отредактировать", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int whichButton) {  
 String name = nameEditText.getText().toString();  
 String surname = surnameEditText.getText().toString();  
 String birthday = birthdayEditText.getText().toString();  
  
 // Обновление пользователя в локальной базе данных  
 updateUser(userId, name, surname, birthday);  
 }  
 });  
 dialogBuilder.setNegativeButton("Отмена", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int whichButton) {  
 // Отмена редактирования пользователя  
 }  
 });  
 AlertDialog alertDialog = dialogBuilder.create();  
 alertDialog.show();  
 }  
  
 private void deleteUser(String userId) {  
 DatabaseHelper databaseHelper = new DatabaseHelper(this);  
 databaseHelper.deleteUser(userId);  
 fetchUsers(); // Обновляем список пользователей после удаления  
 }  
  
 private void updateUser(String userId, String name, String surname, String birthday) {  
 User updatedUser = new User(userId, name, surname, birthday);  
 DatabaseHelper databaseHelper = new DatabaseHelper(this);  
 databaseHelper.updateUser(updatedUser);  
 fetchUsers(); // Обновляем список пользователей после редактирования  
 }  
  
 private void showAddUserDialog() {  
 AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);  
 LayoutInflater inflater = this.getLayoutInflater();  
 View dialogView = inflater.inflate(R.layout.*add\_user\_dialog*, null);  
 dialogBuilder.setView(dialogView);  
  
 final EditText nameEditText = dialogView.findViewById(R.id.*nameEditText*);  
 final EditText surnameEditText = dialogView.findViewById(R.id.*surnameEditText*);  
 final EditText birthdayEditText = dialogView.findViewById(R.id.*birthdayEditText*);  
  
 dialogBuilder.setTitle("Добавить нового пользователя");  
 dialogBuilder.setPositiveButton("Добавить", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int whichButton) {  
 String name = nameEditText.getText().toString();  
 String surname = surnameEditText.getText().toString();  
 String birthday = birthdayEditText.getText().toString();  
  
 // Добавление нового пользователя в локальную базу данных  
 addUserToDatabase(name, surname, birthday);  
 }  
 });  
 dialogBuilder.setNegativeButton("Отмена", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int whichButton) {  
 // Отмена добавления пользователя  
 }  
 });  
 AlertDialog alertDialog = dialogBuilder.create();  
 alertDialog.show();  
 }  
  
 private void addUserToDatabase(String name, String surname, String birthday) {  
 User newUser = new User(); // Создание пользователя с автоматически сгенерированным \_id  
 newUser.setName(name);  
 newUser.setSurname(surname);  
 newUser.setBirthday(birthday);  
  
 // Выводим информацию о новом пользователе в консоль  
 System.*out*.println("Новый пользователь:");  
 System.*out*.println(newUser.toString());  
  
 DatabaseHelper databaseHelper = new DatabaseHelper(this);  
 databaseHelper.addUser(newUser);  
 fetchUsers(); // Обновляем список пользователей после добавления нового пользователя  
 }  
  
}

User

package com.example.laba81;  
import java.util.UUID;  
public class User {  
 private static int *nextId* = 1; // Статическая переменная для следующего доступного id  
 private String \_id;  
 private String name;  
 private String surname;  
 private String birthday;  
  
 // Конструктор без параметров  
 // Пустой конструктор  
 public User() {  
 this.\_id = String.*valueOf*(*nextId*++); // Генерация уникального идентификатора  
 }  
  
 // Конструктор с параметрами (все поля, включая \_id)  
 public User(String \_id, String name, String surname, String birthday) {  
 this.\_id = \_id;  
 this.name = name;  
 this.surname = surname;  
 this.birthday = birthday;  
 }  
  
 // Геттеры и сеттеры  
 public String get\_id() {  
 return \_id;  
 }  
  
 public void set\_id(String \_id) {  
 this.\_id = \_id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(String birthday) {  
 this.birthday = birthday;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "\_id='" + \_id + '\'' +  
 ", name='" + name + '\'' +  
 ", surname='" + surname + '\'' +  
 ", birthday='" + birthday + '\'' +  
 '}';  
 }  
}