Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

**Название работы** – “Разработка Android приложения”

Отчет по лабораторной работе “Лабораторная работа №7”

по дисциплине Объектно-ориентированное программирование

Вариант 24

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Д. Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Маланова

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2021 г.

Содержание

[1 Постановка задач 3](#_Toc68939177)

[2 Проектирование графического интерфейса 4](#_Toc68939178)

[3 Проектирование базы данных 14](#_Toc68939179)

[4 Проектирование классов 17](#_Toc68939180)

[5 Таблица тестов 19](#_Toc68939181)

[6 Результаты тестов 20](#_Toc68939182)

[7 Листинг исходного кода 25](#_Toc68939183)

[8 Заключение 59](#_Toc68939184)

1 Постановка задач

**Цель работы:**

1. Получение общего представления об объектном подходе к организации доступа к базе данных (БД), выполнению запросов и обработке результатов.

2. Получение практического опыта организации работы с БД средствами платформы Android Studio.

**Задача:**

Предметная область: Библиотека

Имеются книги (регистрационный номер, название, количество страниц, год издания, раздел - учебник, художественная общественно-политическая и т.д. ) и читатели (ФИО, домашний адрес, паспортные данные).

Необходимо регистрировать дату, когда какой - либо читатель берет или возвращает книгу.

Выходные документы:

1. список читателей, которые держат на руках книги более месяца, с перечислением названия книги и даты выдачи, упорядоченный по датам выдачи, с указанием количества книг, которые должен сдать каждый читатель;

2. для заданного читателя выдать список прочитанных им книг, сортируя по датам получения.

**Содержание задания:**

Для выполнения задания необходимо разработать java-приложение с графическим пользовательским интерфейсом позволяющее просматривать и редактировать информацию из базы данных. Работу предваряет организация и наполнение базы данных (БД) в соответствии с предметной областью из индивидуального задания.

Графический пользовательский интерфейс должен предоставлять возможности работы с записями основных таблиц БД. Состав используемых компонентов определяется самостоятельно, но для их размещения должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах.

Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий.

2 Проектирование графического интерфейса

Интерфейс программы спроектирован с помощью библиотеки средств среды разработки Android Studio. Управление программой разбито на четыре активности, связанных с таблицами. В первом, основном окне, расположена таблица читателей и пункт меню, позволяющий менять активность.

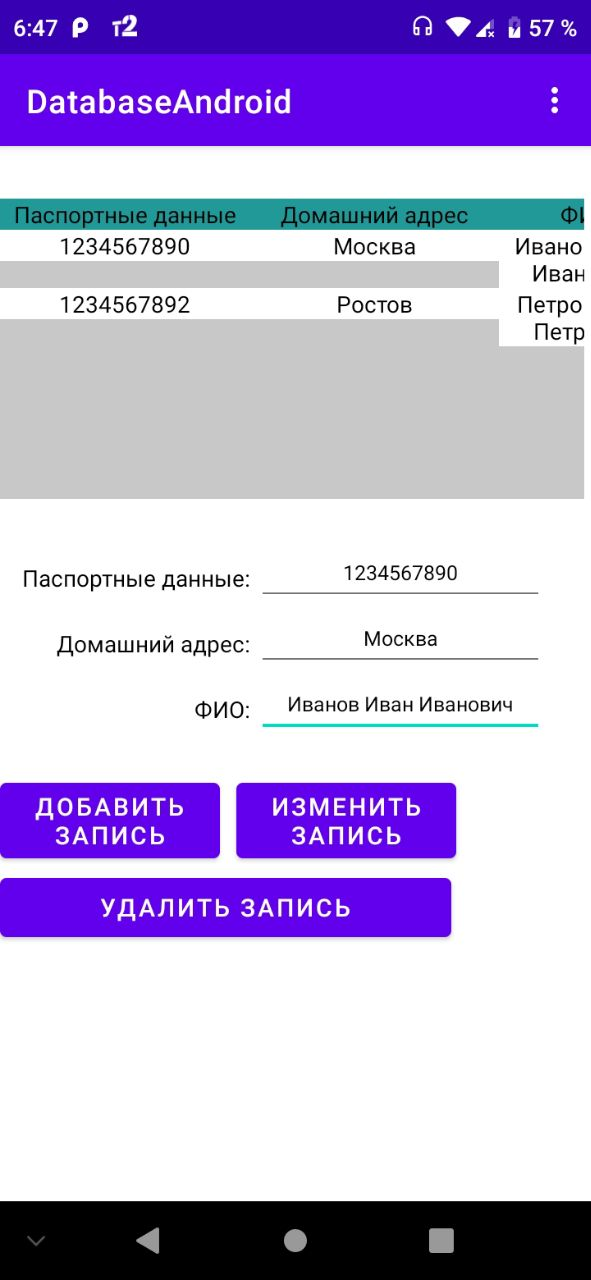


Рисунок 1 – Первая активность, которая позволяет взаимодействовать со списком читателей

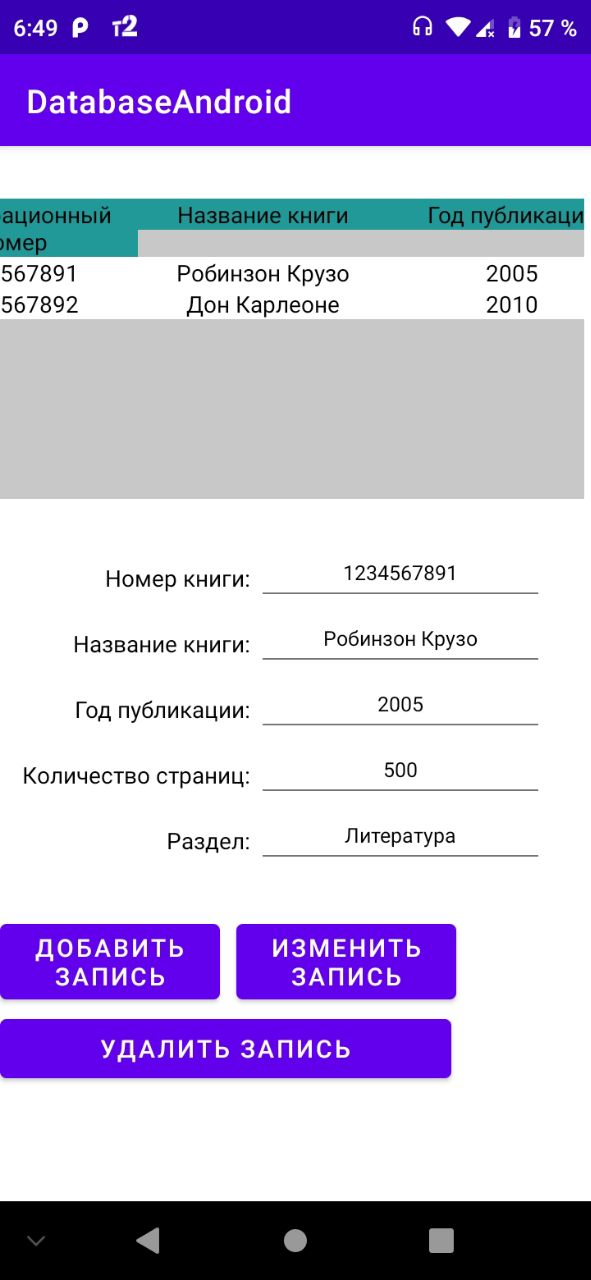


Рисунок 2 – Вторая активность, которая позволяет взаимодействовать со списком книг



Рисунок 3 – Третья активность, которая позволяет взаимодействовать со списком регистрации



Рисунок 4 – Чётвертая активность, предоставляющая интерфейс для решения задач индивидуального варианта

Переход между активностями осуществляется с помощью вложенного меню, которое расположено в правом верхнем углу главной активности. После нажатия на определённую кнопку, текущая активность приостанавливается и открывается активность, выбранна пользователем.

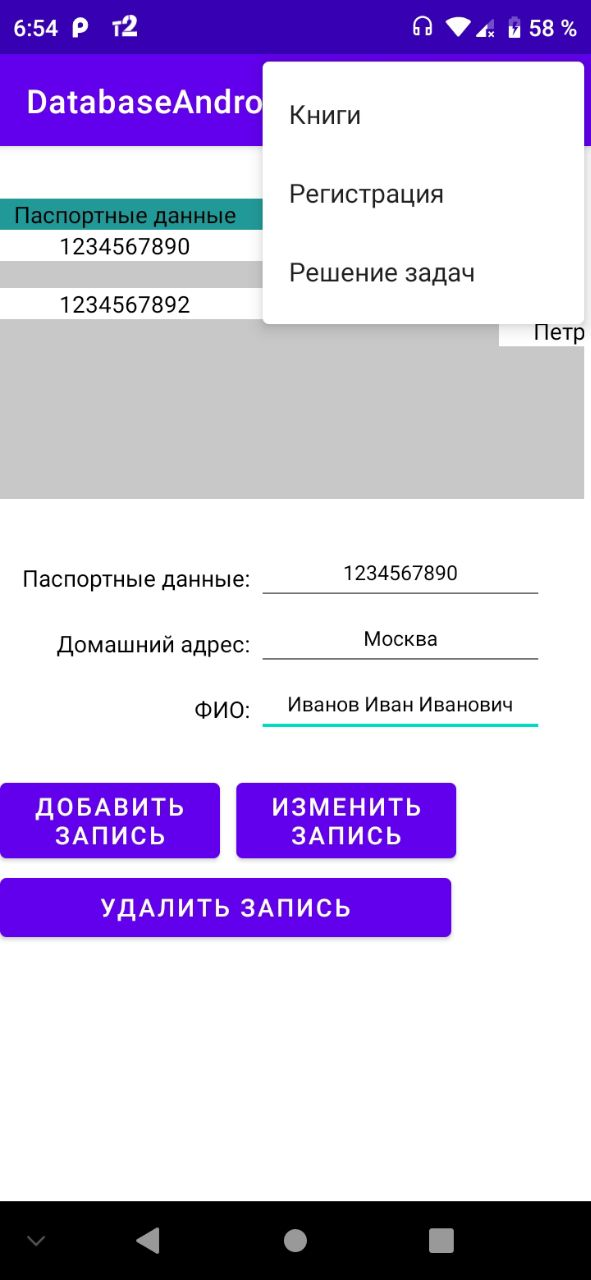


Рисунок 5 – Пункты меню, с помощью которых можно осуществлять переход между активностями

Алгоритм добавления записей в таблицу, удаления и изменения их в каждой активности одинаков, по этой причине будет разобрано взаимодействие с главной активностью (см. рис. 1).

Для добавления записи о читателе, необходимо ввести его паспортные данные, домашний адрес и ФИО в текстовые блоки (EditText), расположенные напротив соответствующих им по смыслу текстовых полей (TextView). После внесение данных в текстовые блоки можно добавить данную запись в таблицу, нажав на кнопку “Добавить запись”.

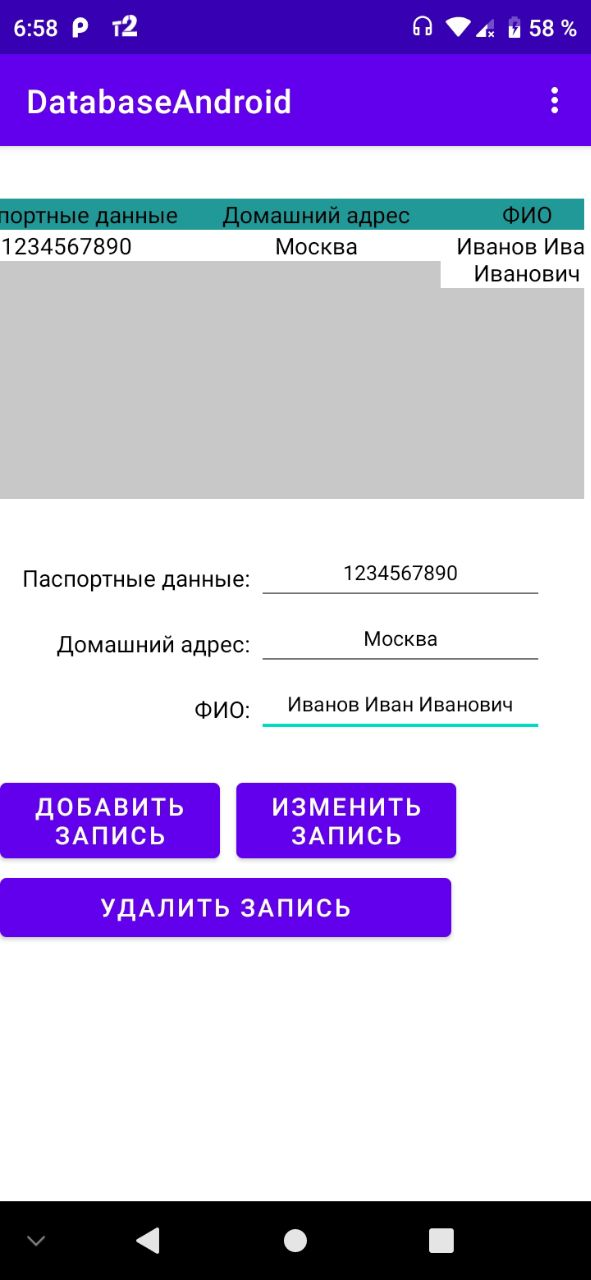


Рисунок 6 – Добавление записи в таблицу

Любую запись в таблице можно изменить. Для этого нужно нажать на любую строку таблицы (для выделения), ввести данные в текстовые блоки и нажать на кнопку “Изменить запись”.

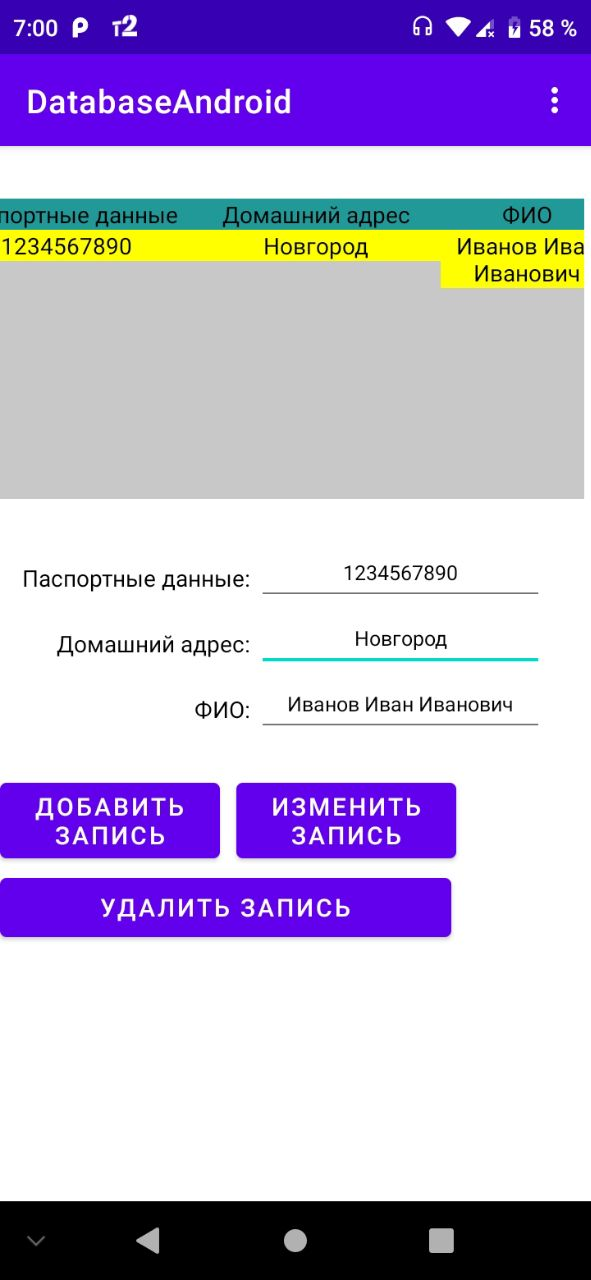


Рисунок 7 – Изменение записи в таблице (сравните с рис. 6)

Также, любую запись в таблице можно удалить. Для этого достаточно щелкнуть левой кнопкой мыши на строку, которую необходимо удалить (для её выделения), и нажать на кнопку “Удалить”.

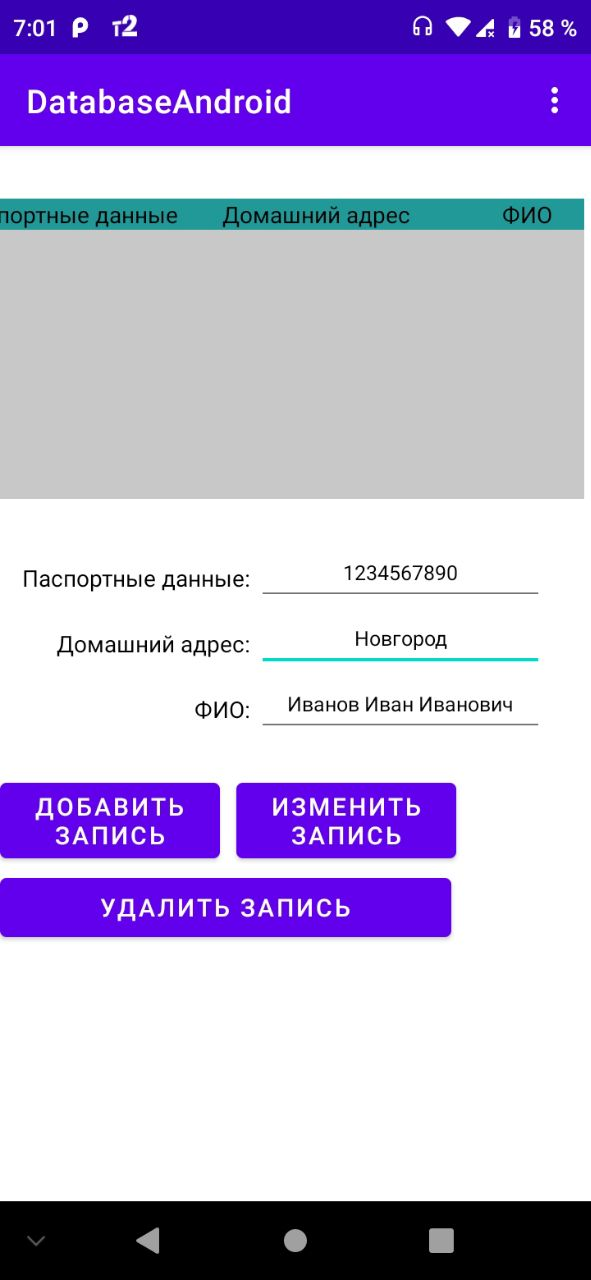


Рисунок 8 – Удаление строки в таблице

При добавлении данных в таблицу в SQLite автоматически добавляется соответствующая запись. То есть, при любом изменении значения таблиц, эти изменения отражаются на содержимом базы данных в целом. При запуске приложения для любой открытой активности данные считываются из базы данных в таблицу, тем самым пользователь имеет актуальные данные в настоящее время пользования приложением.

Для решения практических задач, поставленных в данной лабораторной работе, необходимо предварительное наполнение базы данных.



Рисунок 9 – Наполненная таблица “Читатели”



Рисунок 10 – Наполненная таблица “Книги”



Рисунок 11 – Наполненная таблица “Регистрация”

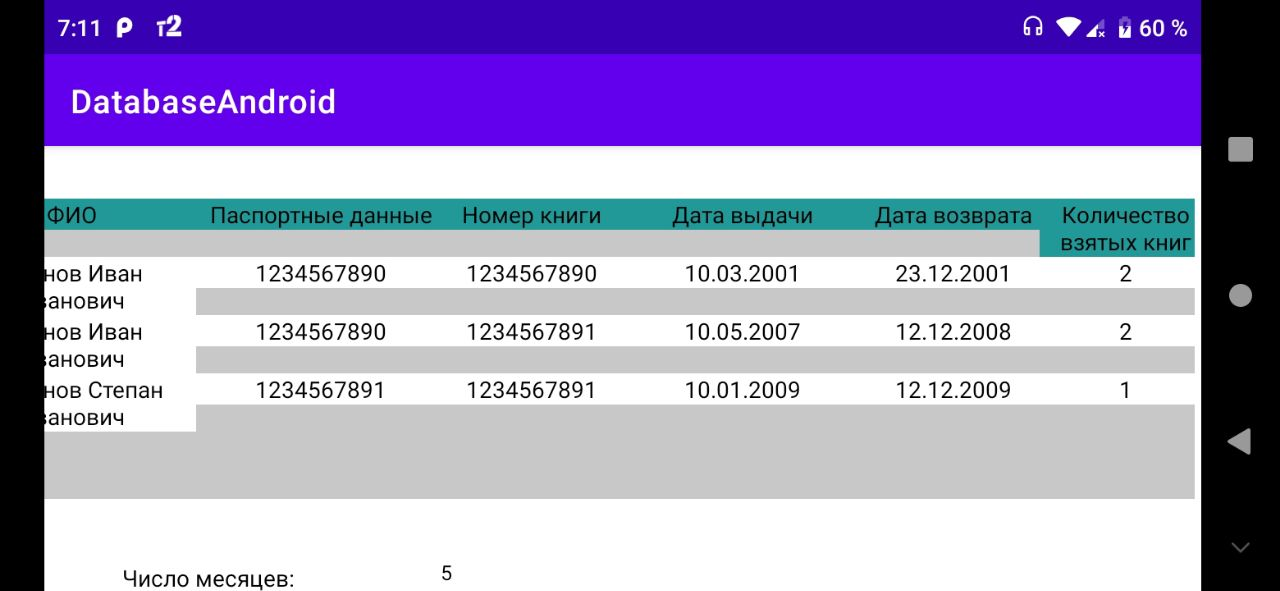


Рисунок 12 – Решение задания №1 индивидуального варианта

При решении практических задач программа руководствуется фактическим наполнением базы данных SQLite, то есть при решении данные из БД считываются.



Рисунок 13 – Решение задания №2 индивидуального варианта

3 Проектирование базы данных

Степень связи N:M. Необходимо регистрировать дату получения книги читателем и дату возврата книги обратно. Данные о книгах содержаться в отдельном экземпляре объекта, как и данные о читателях. Экземпляры объектов “Читатель” и “Книга” не связаны внешними ключами указывающими друг на друга и хранят информацию не зависимо. Экземпляр объекта “Книга” не зависит от экземпляра объекта “Читатель”. Между ними в связи существует посредник – экземпляр объекта “Регистрация записи”, который должен содержать внешние не уникальные ключи на экземпляры объектов, поскольку в таблице. “Регистрация записи” может регистрироваться ситуация, когда один и тот же читатель с одними и теми же паспортными данным получает и возвращает одну и ту же книгу несколько раз с одними и теми же регистрационными данными.

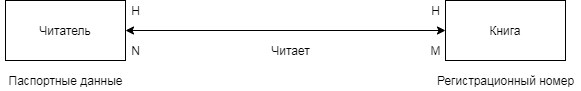


Рисунок 14 – ER-Диаграмма

Если степень связи M:N, то необходимо построение трех отношений: по одному на каждый объект (первичные ключи отношений совпадают с ключами объектов) и связующего отношения, первичным ключом которого будет комбинация ключей объектов.

Для ER-диаграммы получаем по 3 отношения, добавляя связующее звено (экземпляр объекта “Регистрация записи”) первичным ключом, которого является комбинация ключей объектов “Читатель” и “Книга”.

Читатель (Паспортные данные)

Книга (Регистрационный номер, Паспортные данные)

Регистрация записи (Паспортные данные, Регистрационный номер)

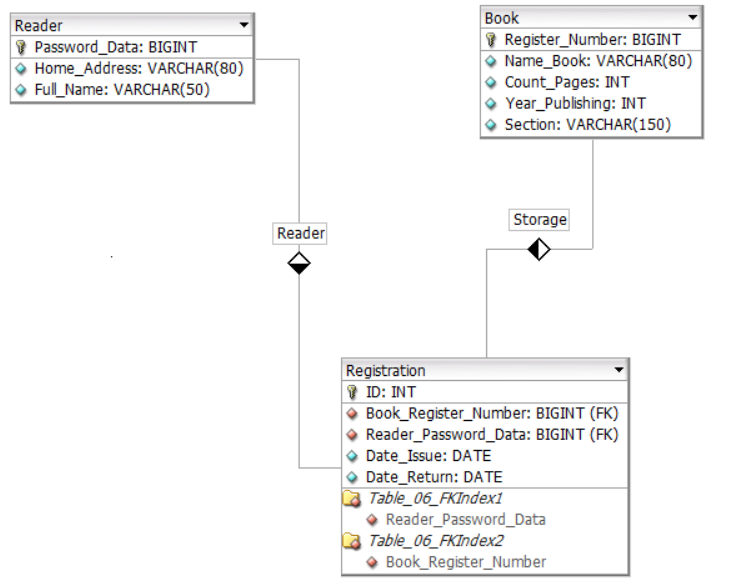


Рисунок 15 – Логическая модель базы данных

Для практической реализации базы данных за основу взята база данных SQLite и язык, для взаимодействия с базой данных – SQL.

В силу ограничений SQLite, логическая модель базы не совподает с физической реализацией, т.к. типы данных будут использоваться разные и функционал, предоставляемый SQLite, достаточно немногочислен.

SQL-Скрипт создания таблицы для хранения книг:

CREATE TABLE Book (

Register\_Number INTEGER PRIMARY KEY,

Name\_Book TEXT,

Count\_Pages INTEGER,

Year\_Publishing INTEGER,

Section TEXT

);

SQL-Скрипт создания таблицы для хранения читателей:

CREATE TABLE Reader (

Password\_Data INTEGER PRIMARY KEY,

Home\_Address TEXT,

Full\_Name TEXT,

);

SQL-Скрипт создания таблицы регистрации:

CREATE TABLE Register(

ID INTEGER PRIMARY KEY,

Register\_Number TEXT,

Password\_Data TEXT,

Date\_Issue TEXT,

Date\_Return TEXT,

FOREIGN KEY (Register\_Number) REFERENCES Book (Register\_Number),

FOREIGN KEY (Password\_Data) REFERENCES Reader (Password\_Data));

4 Проектирование классов

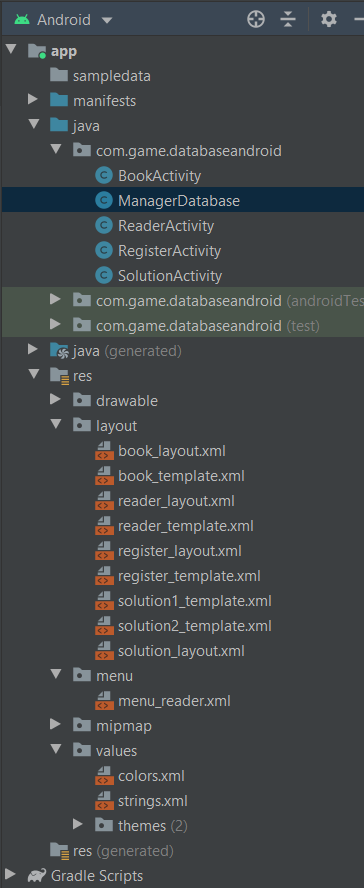


Рисунок 16 – Структура проекта

В проекте присутствуют 5 пользовательских классов, а также более 10 файлов с расширением \*.xml, для разметки пользовательского интерфейса.

Класс BookActivity предоставляет интерфейс для взаимодействия с таблицей “Книги”.

Класс ReaderActivity является основной активностью, которая запускает все остальные и также предоставляет интерфейс для взаимодействия с таблицей “Читатели”.

Класс SolutionActivity предоставляет интерфейс для решения задач индивидуального варианта.

Класс RegisterActivity предоставляет интерфейс для взаимодействия с таблицей “Регистрация”.

Класс ManagerDatabase предоставляет интерфейс для взаимодействия с базой данных SQLite.

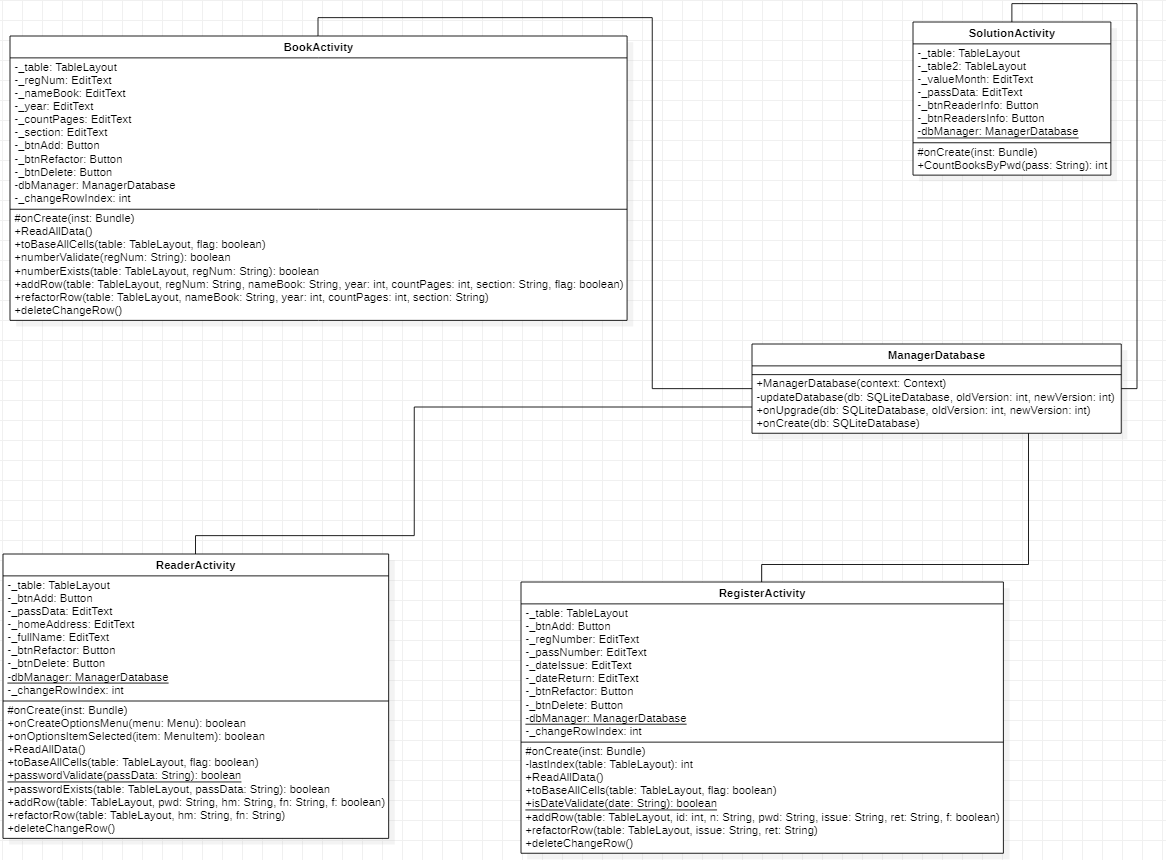


Рисунок 17 – UML-Диаграмма классов

5 Таблица тестов

Таблица 1 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Попытка добавления не корректных данных в таблицу читетелей | Паспортные данные: 125126  Домашний адрес:  ФИО: Иван | (см. рис. 18) |
| 2 | Попытка удаление данных, зарегистрированных в таблице регистрации | - | (см. рис. 19) |
| 3 | Добавление данных в таблицу книг, с номером книги, который уже имеется в таблице | - | (см. рис. 20) |
| 4 | Попытка изменить дату публикации книги таким образом, чтобы все зарегистрированные записи были не корректны по времени | - | (см. рис. 21) |
| 5 | Попытка изменить дату выдачи книги таким образом, чтобы она была раньше чем год публикации книги | - | (см. рис. 22) |

6 Результаты тестов

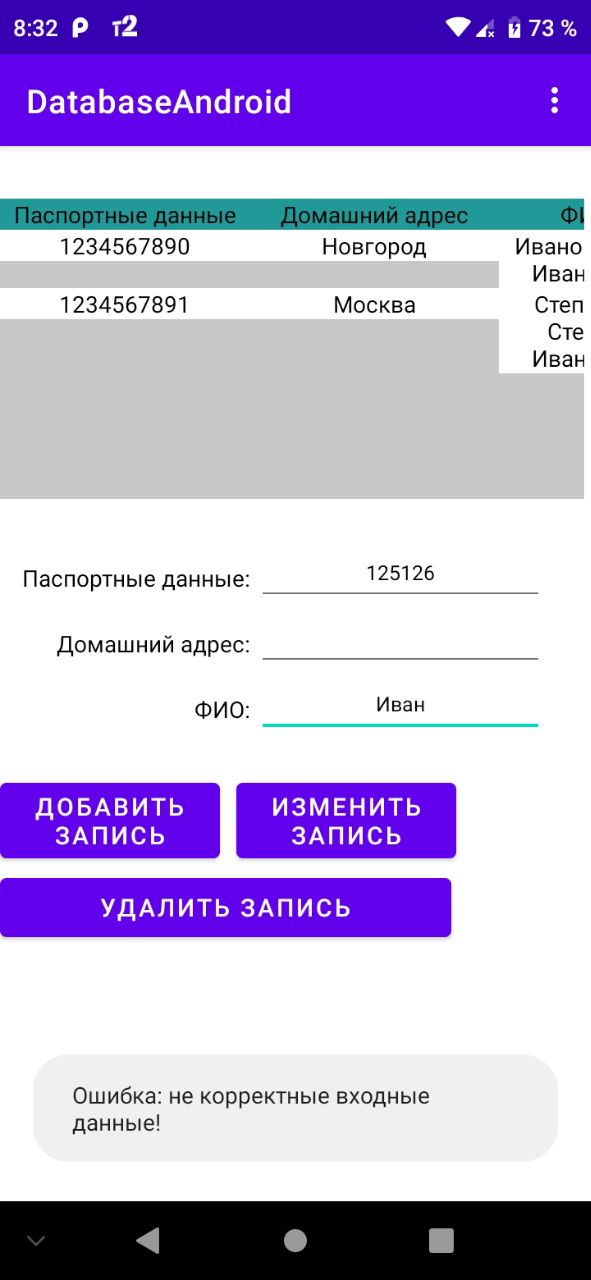


Рисунок 18 – Результат теста №1 из таблицы тестов №1

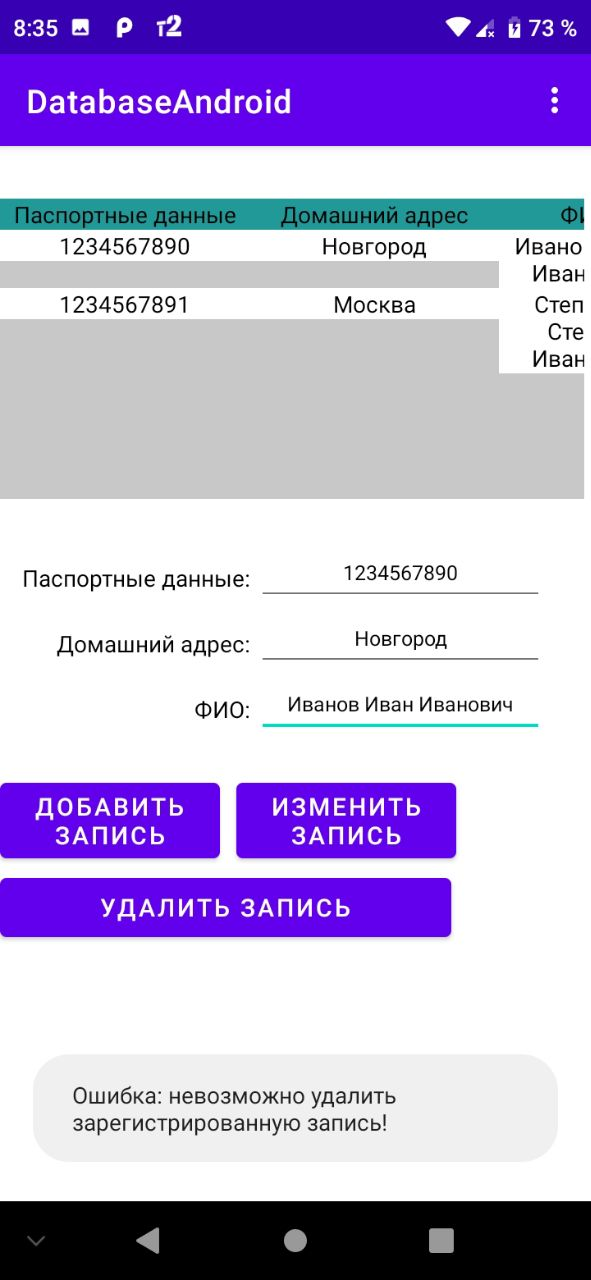


Рисунок 19 – Результат теста №2 из таблицы тестов №1



Рисунок 20 – Результат теста №3 из таблицы тестов №1

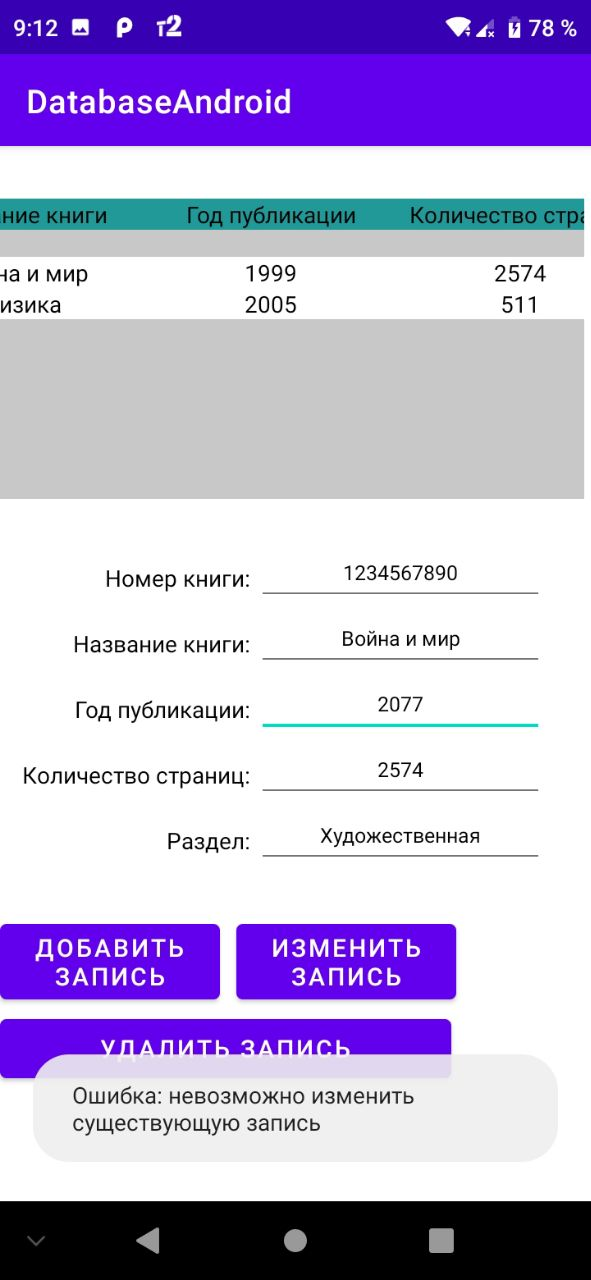


Рисунок 21 – Результат теста №4 из таблицы тестов №1



Рисунок 22 – Результат теста №5 из таблицы тестов №1

7 Листинг исходного кода

**ReaderActivity:**

package com.game.databaseandroid;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Color;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TableLayout;

import android.widget.TableRow;

import android.widget.TextView;

import android.widget.Toast;

public class ReaderActivity extends AppCompatActivity {

private TableLayout \_table = null;

private Button \_btnAdd = null;

private EditText \_passData = null;

private EditText \_homeAddress = null;

private EditText \_fullName = null;

private Button \_btnRefactor = null;

private Button \_btnDelete = null;

private static ManagerDatabase dbManager;

public static final int STD\_SIZE\_PASS = 10;

private int \_changeRowIndex = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.reader\_layout);

\_table = (TableLayout) findViewById(R.id.tableReader);

\_btnAdd = (Button) findViewById(R.id.btnAdd);

\_passData = (EditText) findViewById(R.id.pass\_data);

\_homeAddress = (EditText) findViewById(R.id.home\_address);

\_fullName = (EditText) findViewById(R.id.full\_name);

\_btnRefactor = (Button) findViewById(R.id.btnRefactor);

\_btnDelete = (Button) findViewById(R.id.btnDelete);

\_btnAdd.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

toBaseAllCells(\_table, true);

addRow(\_table, \_passData.getText().toString().trim(), \_homeAddress.getText().toString().trim(),

\_fullName.getText().toString().trim(), true);

}catch (Exception e){

Toast.makeText(ReaderActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

\_btnRefactor.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

refactorRow(\_table, \_homeAddress.getText().toString().trim(),

\_fullName.getText().toString().trim());

toBaseAllCells(\_table, true);

}catch (Exception e){

Toast.makeText(ReaderActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

\_btnDelete.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

deleteChangeRow();

toBaseAllCells(\_table, true);

}catch (Exception e){

Toast.makeText(ReaderActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

dbManager = new ManagerDatabase(ReaderActivity.this);

try {

ReadAllData();

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.menu\_reader, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

switch(id){

case R.id.goToBook :

Intent intent1 = new Intent(ReaderActivity.this, BookActivity.class);

startActivity(intent1);

return true;

case R.id.goToReader:

Intent intent2 = new Intent(ReaderActivity.this, RegisterActivity.class);

startActivity(intent2);

return true;

case R.id.goToSolution:

Intent intent3 = new Intent(ReaderActivity.this, SolutionActivity.class);

startActivity(intent3);

return true;

}

return super.onOptionsItemSelected(item);

}

private void ReadAllData() throws Exception {

SQLiteDatabase database = dbManager.getWritableDatabase();

Cursor cursor = database.query(ManagerDatabase.TABLE\_READER, null, null,

null, null, null, null);

while(cursor.moveToNext()){

int passIndex= cursor.getColumnIndex(ManagerDatabase.PASSWORD),

homeIndex = cursor.getColumnIndex(ManagerDatabase.HOME),

fullIndex = cursor.getColumnIndex(ManagerDatabase.FULLNAME);

addRow(\_table, cursor.getString(passIndex).trim(),

cursor.getString(homeIndex).trim(),

cursor.getString(fullIndex).trim(), false);

}

cursor.close();

database.close();

}

public void toBaseAllCells(TableLayout table, boolean flag){

if((table == null) || (\_changeRowIndex <= 0) || (\_changeRowIndex >= table.getChildCount())){

\_changeRowIndex = 0;

return;

}

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol1)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol2)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol3)).setBackgroundColor(Color.WHITE);

if(flag)

\_changeRowIndex = 0;

}

public static boolean passwordValidate(String passData){

if((passData == null) || (passData.length() != STD\_SIZE\_PASS))

return false;

for(int i = 0; i < passData.length(); i++)

if(!Character.isDigit(passData.charAt(i)))

return false;

return true;

}

public boolean passwordExists(TableLayout table, String passData){

if((!passwordValidate(passData)) || (table == null))

return false;

for(int i = 1; i < table.getChildCount(); i++){

if(((TextView) (table.getChildAt(i)).findViewById(R.id.readerCol1)).getText().toString().equals(passData))

return true;

}

return false;

}

public void addRow(TableLayout table, String passData, String homeAddress, String fullName, boolean flag) throws Exception{

if((table == null) || (homeAddress == null)

|| (fullName == null) || (homeAddress.length() == 0)

|| (fullName.length() == 0)

|| (!passwordValidate(passData)))

throw new Exception("Ошибка: не корректные входные данные!");

if(passwordExists(table, passData))

throw new Exception("Ошибка: данные паспортные данные уже есть в базе данных!");

LayoutInflater inflater = LayoutInflater.from(ReaderActivity.this);

TableRow row = (TableRow) inflater.inflate(R.layout.reader\_template, null);

((TextView) row.findViewById(R.id.readerCol1)).setText(passData);

((TextView) row.findViewById(R.id.readerCol2)).setText(homeAddress);

((TextView) row.findViewById(R.id.readerCol3)).setText(fullName);

row.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

toBaseAllCells(table, false);

for(int i = 1; i < table.getChildCount(); i++){

if(

(table.getChildAt(i) != null) &&

(table.getChildAt(i).findViewById(R.id.readerCol1) != null) &&

((TextView)table.getChildAt(i).findViewById(R.id.readerCol1)).getText().toString()

.equals(((TextView) row.findViewById(R.id.readerCol1)).getText().toString())){

if(\_changeRowIndex == i){

\_changeRowIndex = 0;

return;

}

\_changeRowIndex = i;

break;

}

}

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= table.getChildCount())){

toBaseAllCells(table, true);

return;

}

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol1)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol2)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol3)).setBackgroundColor(Color.YELLOW);

\_passData.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol1)).getText().toString());

\_homeAddress.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol2)).getText().toString());

\_fullName.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol3)).getText().toString());

}

});

table.addView(row);

if(flag){

SQLiteDatabase database = dbManager.getWritableDatabase();

ContentValues contentValues = new ContentValues();

contentValues.put(ManagerDatabase.PASSWORD, passData);

contentValues.put(ManagerDatabase.HOME, homeAddress);

contentValues.put(ManagerDatabase.FULLNAME, fullName);

database.insert(ManagerDatabase.TABLE\_READER, null, contentValues);

database.close();

}

}

public void refactorRow(TableLayout table, String homeAddress, String fullName) throws Exception{

if((table == null) || (homeAddress == null)

|| (fullName == null) || (homeAddress.length() == 0)

|| (fullName.length() == 0))

throw new Exception("Ошибка: не корректные входные данные!");

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= \_table.getChildCount()))

throw new Exception("Ошибка: не выделена строка для изменения!");

SQLiteDatabase database = dbManager.getWritableDatabase();

try{

ContentValues cv = new ContentValues();

cv.put(ManagerDatabase.HOME, homeAddress);

cv.put(ManagerDatabase.FULLNAME, fullName);

database.update(ManagerDatabase.TABLE\_READER, cv, "Number\_Password = ?",

new String[]{

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol1)).getText().toString()

});

database.close();

}catch(Exception e){

Toast.makeText(ReaderActivity.this, "Ошибка: невозможно изменить существующую запись", Toast.LENGTH\_LONG).show();

toBaseAllCells(table, true);

if(database != null)

database.close();

return;

}

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol2)).setText(homeAddress);

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol3)).setText(fullName);

}

public void deleteChangeRow() throws Exception{

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= \_table.getChildCount()))

throw new Exception("Ошибка: не выделена строка для удаления!");

SQLiteDatabase database = null;

try{

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_PASSWORD));

if(data.equals(((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol1)).getText().toString())){

exist = true;

break;

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(exist)

throw new Exception();

}

database = dbManager.getWritableDatabase();

database.delete(ManagerDatabase.TABLE\_READER, "Number\_Password = ?",

new String[]{

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.readerCol1)).getText().toString()

});

database.close();

}catch (Exception e){

Toast.makeText(ReaderActivity.this, "Ошибка: невозможно удалить зарегистрированную запись!", Toast.LENGTH\_LONG).show();

toBaseAllCells(\_table, true);

if(database != null)

database.close();

return;

}

\_table.removeViewAt(\_changeRowIndex);

}

}

**BookActivity:**

package com.game.databaseandroid;

import android.content.ContentValues;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Color;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TableLayout;

import android.widget.TableRow;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class BookActivity extends AppCompatActivity {

private TableLayout \_table = null;

private EditText \_regNum = null;

private EditText \_nameBook = null;

private EditText \_year = null;

private EditText \_countPages = null;

private EditText \_section = null;

private Button \_btnAdd = null;

private Button \_btnRefactor = null;

private Button \_btnDelete = null;

private static ManagerDatabase dbManager;

public static final int STD\_SIZE\_REGNUM = 10;

private int \_changeRowIndex = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.book\_layout);

\_table = (TableLayout) findViewById(R.id.tableReader);

\_regNum = (EditText) findViewById(R.id.reg\_data);

\_nameBook = (EditText) findViewById(R.id.name\_book);

\_year = (EditText) findViewById(R.id.year);

\_countPages = (EditText) findViewById(R.id.count\_pages);

\_section = (EditText) findViewById(R.id.section);

\_btnAdd = (Button) findViewById(R.id.btnAdd);

\_btnRefactor = (Button) findViewById(R.id.btnRefactor);

\_btnDelete = (Button) findViewById(R.id.btnDelete);

\_btnAdd.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

toBaseAllCells(\_table, true);

addRow(\_table, \_regNum.getText().toString().trim(), \_nameBook.getText().toString().trim(),

Integer.valueOf(\_year.getText().toString().trim()),

Integer.valueOf(\_countPages.getText().toString().trim()),

\_section.getText().toString().trim(), true);

}catch (Exception e){

Toast.makeText(BookActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

\_btnRefactor.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

refactorRow(\_table,

\_nameBook.getText().toString().trim(),

Integer.valueOf(\_year.getText().toString().trim()),

Integer.valueOf(\_countPages.getText().toString().trim()),

\_section.getText().toString().trim());

toBaseAllCells(\_table, true);

}catch (Exception e){

Toast.makeText(BookActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

\_btnDelete.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

deleteChangeRow();

toBaseAllCells(\_table, true);

}catch (Exception e){

Toast.makeText(BookActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

dbManager = new ManagerDatabase(BookActivity.this);

try {

ReadAllData();

} catch (Exception e) {

e.printStackTrace();

}

}

private void ReadAllData() throws Exception {

SQLiteDatabase database = dbManager.getWritableDatabase();

Cursor cursor = database.query(ManagerDatabase.TABLE\_BOOK, null, null,

null, null, null, null);

while(cursor.moveToNext()){

int regIndex= cursor.getColumnIndex(ManagerDatabase.REGISTER),

nameIndex = cursor.getColumnIndex(ManagerDatabase.NAME),

yearIndex = cursor.getColumnIndex(ManagerDatabase.YEAR),

pagesIndex = cursor.getColumnIndex(ManagerDatabase.PAGES),

sectionIndex = cursor.getColumnIndex(ManagerDatabase.SECTION);

addRow(\_table, cursor.getString(regIndex).trim(),

cursor.getString(nameIndex).trim(),

cursor.getInt(yearIndex),

cursor.getInt(pagesIndex),

cursor.getString(sectionIndex).trim(), false);

}

cursor.close();

database.close();

}

public void toBaseAllCells(TableLayout table, boolean flag){

if((table == null) || (\_changeRowIndex <= 0) || (\_changeRowIndex >= table.getChildCount())){

\_changeRowIndex = 0;

return;

}

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol2)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol3)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol4)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol5)).setBackgroundColor(Color.WHITE);

if(flag)

\_changeRowIndex = 0;

}

public static boolean numberValidate(String regNum){

if((regNum == null) || (regNum.length() != STD\_SIZE\_REGNUM))

return false;

for(int i = 0; i < regNum.length(); i++)

if(!Character.isDigit(regNum.charAt(i)))

return false;

return true;

}

public boolean numberExists(TableLayout table, String regNum){

if((!numberValidate(regNum)) || (table == null))

return false;

for(int i = 1; i < table.getChildCount(); i++){

if(((TextView) (table.getChildAt(i)).findViewById(R.id.bookCol1)).getText().toString().equals(regNum))

return true;

}

return false;

}

public void addRow(TableLayout table, String regNum, String nameBook, int year, int countPages, String section, boolean flag) throws Exception{

if((table == null) || (regNum == null) || (section == null)

|| (year < 0) || (countPages <= 0)

|| (nameBook == null) || (nameBook.length() == 0)

|| (section.length() == 0)

|| (section.length() == 0)

|| (!numberValidate(regNum)))

throw new Exception("Ошибка: не корректные входные данные!");

if(numberExists(table, regNum))

throw new Exception("Ошибка: данный регистрационный номер уже есть в базе данных!");

LayoutInflater inflater = LayoutInflater.from(BookActivity.this);

TableRow row = (TableRow) inflater.inflate(R.layout.book\_template, null);

((TextView) row.findViewById(R.id.bookCol1)).setText(regNum);

((TextView) row.findViewById(R.id.bookCol2)).setText(nameBook);

((TextView) row.findViewById(R.id.bookCol3)).setText(String.valueOf(year));

((TextView) row.findViewById(R.id.bookCol4)).setText(String.valueOf(countPages));

((TextView) row.findViewById(R.id.bookCol5)).setText(section);

row.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

toBaseAllCells(table, false);

for(int i = 1; i < table.getChildCount(); i++){

if(

(table.getChildAt(i) != null) && (table.getChildAt(i).findViewById(R.id.bookCol1) != null) &&

((TextView)table.getChildAt(i).findViewById(R.id.bookCol1)).getText().toString()

.equals(((TextView) row.findViewById(R.id.bookCol1)).getText().toString())){

if(\_changeRowIndex == i){

\_changeRowIndex = 0;

return;

}

\_changeRowIndex = i;

break;

}

}

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= table.getChildCount())){

toBaseAllCells(table, true);

return;

}

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol2)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol3)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol4)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol5)).setBackgroundColor(Color.YELLOW);

\_regNum.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).getText().toString());

\_nameBook.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol2)).getText().toString());

\_year.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol3)).getText().toString());

\_countPages.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol4)).getText().toString());

\_section.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol5)).getText().toString());

}

});

table.addView(row);

if(flag){

SQLiteDatabase database = dbManager.getWritableDatabase();

ContentValues contentValues = new ContentValues();

contentValues.put(ManagerDatabase.REGISTER, regNum);

contentValues.put(ManagerDatabase.NAME, nameBook);

contentValues.put(ManagerDatabase.YEAR, year);

contentValues.put(ManagerDatabase.PAGES, countPages);

contentValues.put(ManagerDatabase.SECTION, section);

database.insert(ManagerDatabase.TABLE\_BOOK, null, contentValues);

database.close();

}

}

public void refactorRow(TableLayout table, String nameBook, int year, int countPages, String section) throws Exception{

if((table == null) || (section == null)

|| (year < 0) || (countPages <= 0)

|| (nameBook == null) || (nameBook.length() == 0)

|| (section.length() == 0)

|| (section.length() == 0))

throw new Exception("Ошибка: не корректные входные данные!");

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= \_table.getChildCount()))

throw new Exception("Ошибка: не выделена строка для изменения!");

SQLiteDatabase database = null;

try{

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_REGNUM));

if(data.equals(((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).getText().toString())){

int yearReg = Integer.valueOf(cursor.getString(cursor.getColumnIndex(ManagerDatabase.DATA\_ISSUE)).split("\\.")[2]);

if(yearReg < Integer.valueOf(year)){

exist = true;

break;

}

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(exist)

throw new Exception();

}

database = dbManager.getWritableDatabase();

ContentValues cv = new ContentValues();

cv.put(ManagerDatabase.NAME, nameBook);

cv.put(ManagerDatabase.YEAR, year);

cv.put(ManagerDatabase.PAGES, countPages);

cv.put(ManagerDatabase.SECTION, section);

database.update(ManagerDatabase.TABLE\_BOOK, cv, "Number\_Register = ?",

new String[]{

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).getText().toString()

});

database.close();

}catch(Exception e){

Toast.makeText(BookActivity.this, "Ошибка: невозможно изменить существующую запись", Toast.LENGTH\_LONG).show();

toBaseAllCells(table, true);

if(database != null)

database.close();

return;

}

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol2)).setText(nameBook);

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol3)).setText(String.valueOf(year));

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol4)).setText(String.valueOf(countPages));

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol5)).setText(section);

}

public void deleteChangeRow() throws Exception{

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= \_table.getChildCount()))

throw new Exception("Ошибка: не выделена строка для удаления!");

SQLiteDatabase database = null;

try{

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_REGNUM));

if(data.equals(((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).getText().toString())){

exist = true;

break;

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(exist)

throw new Exception();

}

database = dbManager.getWritableDatabase();

database.delete(ManagerDatabase.TABLE\_BOOK, "Number\_Register = ?",

new String[]{

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.bookCol1)).getText().toString()

});

database.close();

}catch (Exception e){

Toast.makeText(BookActivity.this, "Ошибка: невозможно удалить зарегистрированную запись!", Toast.LENGTH\_LONG).show();

toBaseAllCells(\_table, true);

if(database != null)

database.close();

return;

}

\_table.removeViewAt(\_changeRowIndex);

}

}

**RegisterActivity:**

package com.game.databaseandroid;

import android.content.ContentValues;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteConstraintException;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Color;

import android.os.Bundle;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TableLayout;

import android.widget.TableRow;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

public class RegisterActivity extends AppCompatActivity {

private TableLayout \_table = null;

private Button \_btnAdd = null;

private EditText \_regNumber = null;

private EditText \_passNumber = null;

private EditText \_dateIssue = null;

private EditText \_dateReturn = null;

private Button \_btnRefactor = null;

private Button \_btnDelete = null;

private static ManagerDatabase dbManager;

public static final int STD\_SIZE\_PASS = 10;

private int \_changeRowIndex = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.register\_layout);

\_table = (TableLayout) findViewById(R.id.tableReader);

\_regNumber = (EditText) findViewById(R.id.dataNumber);

\_passNumber = (EditText) findViewById(R.id.dataPassword);

\_dateIssue = (EditText) findViewById(R.id.dataDateIssue);

\_dateReturn = (EditText) findViewById(R.id.dataDateReturn);

\_btnAdd = (Button) findViewById(R.id.btnAdd);

\_btnRefactor = (Button) findViewById(R.id.btnRefactor);

\_btnDelete = (Button) findViewById(R.id.btnDelete);

\_btnAdd.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

toBaseAllCells(\_table, true);

addRow(\_table, lastIndex(\_table), \_regNumber.getText().toString().trim(), \_passNumber.getText().toString().trim(),

\_dateIssue.getText().toString().trim(), \_dateReturn.getText().toString().trim(), true);

}catch (Exception e){

Toast.makeText(RegisterActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

\_btnRefactor.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

refactorRow(\_table, \_dateIssue.getText().toString().trim(),

\_dateReturn.getText().toString().trim());

toBaseAllCells(\_table, true);

}catch (Exception e){

Toast.makeText(RegisterActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

\_btnDelete.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try{

deleteChangeRow();

toBaseAllCells(\_table, true);

}catch (Exception e){

Toast.makeText(RegisterActivity.this, e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

});

dbManager = new ManagerDatabase(RegisterActivity.this);

try {

ReadAllData();

} catch (Exception e) {

e.printStackTrace();

}

}

private int lastIndex(TableLayout table){

if((table == null) || (table.getChildCount() <= 1))

return 1;

int max = 1;

for(int i = 1; i < table.getChildCount(); i++){

int value = Integer.valueOf(((TextView)table.getChildAt(i).findViewById(R.id.registerCol1)).getText().toString());

if(max < value)

max = value;

}

return (max + 1);

}

private void ReadAllData() throws Exception {

SQLiteDatabase database = dbManager.getWritableDatabase();

Cursor cursor = database.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

while(cursor.moveToNext()){

int index = cursor.getColumnIndex(ManagerDatabase.ID);

int numIndex= cursor.getColumnIndex(ManagerDatabase.REGISTER\_REGNUM),

passIndex = cursor.getColumnIndex(ManagerDatabase.REGISTER\_PASSWORD),

issueIndex = cursor.getColumnIndex(ManagerDatabase.DATA\_ISSUE),

returnIndex = cursor.getColumnIndex(ManagerDatabase.DATA\_RETURN);

addRow(\_table, cursor.getInt(index),

cursor.getString(numIndex).trim(),

cursor.getString(passIndex).trim(),

cursor.getString(issueIndex).trim(),

cursor.getString(returnIndex).trim(), false);

}

cursor.close();

database.close();

}

public void toBaseAllCells(TableLayout table, boolean flag){

if((table == null) || (\_changeRowIndex <= 0) || (\_changeRowIndex >= table.getChildCount())){

\_changeRowIndex = 0;

return;

}

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol1)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol2)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol3)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol4)).setBackgroundColor(Color.WHITE);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol5)).setBackgroundColor(Color.WHITE);

if(flag)

\_changeRowIndex = 0;

}

public static boolean isDateValidate(String date){

if((date == null) || (date.length() == 0))

return false;

int count = 0;

for(int i = 0; i < date.length(); i++){

if(date.charAt(i) == '.'){

count++;

}else if(!Character.isDigit(date.charAt(i))){

return false;

}

}

if(count != 2)

return false;

String[] dateSplit = date.split("\\.");

if((dateSplit.length != 3) || (dateSplit[2].length() != 4)

|| (dateSplit[0].length() != 2) || (dateSplit[1].length() != 2))

return false;

int value = Integer.valueOf(dateSplit[0]);

if((value <= 0) || (value > 31))

return false;

value = Integer.valueOf(dateSplit[1]);

if((value <= 0) || (value > 12))

return false;

return true;

}

public void addRow(TableLayout table, int id, String numData, String passData, String issueData, String returnData, boolean flag) throws Exception{

if((table == null)

|| (numData == null) || (passData == null) || (issueData == null) || (returnData == null)

|| (numData.length() == 0) || (passData.length() == 0) || (issueData.length() != 10) || (returnData.length() != 10)

|| (!ReaderActivity.passwordValidate(passData)) || (!BookActivity.numberValidate(numData))

|| (!isDateValidate(issueData)) || (!isDateValidate(returnData)))

throw new Exception("Ошибка: не корректные входные данные!");

DateFormat format = new SimpleDateFormat("dd.mm.yyyy");

if(format.parse(issueData).after(

format.parse(returnData)

)){

throw new Exception("Ошибка: дата выдачи не может быть позже даты возврата!");

}

LayoutInflater inflater = LayoutInflater.from(RegisterActivity.this);

TableRow row = (TableRow) inflater.inflate(R.layout.register\_template, null);

((TextView) row.findViewById(R.id.registerCol1)).setText(String.valueOf(id));

((TextView) row.findViewById(R.id.registerCol2)).setText(numData);

((TextView) row.findViewById(R.id.registerCol3)).setText(passData);

((TextView) row.findViewById(R.id.registerCol4)).setText(issueData);

((TextView) row.findViewById(R.id.registerCol5)).setText(returnData);

row.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

toBaseAllCells(table, false);

for(int i = 1; i < table.getChildCount(); i++){

if(

(table.getChildAt(i) != null) && (table.getChildAt(i).findViewById(R.id.registerCol1) != null) &&

((TextView)table.getChildAt(i).findViewById(R.id.registerCol1)).getText().toString()

.equals(((TextView) row.findViewById(R.id.registerCol1)).getText().toString())){

if(\_changeRowIndex == i){

\_changeRowIndex = 0;

return;

}

\_changeRowIndex = i;

break;

}

}

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol1)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol2)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol3)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol4)).setBackgroundColor(Color.YELLOW);

((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol5)).setBackgroundColor(Color.YELLOW);

\_regNumber.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol2)).getText().toString());

\_passNumber.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol3)).getText().toString());

\_dateIssue.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol4)).getText().toString());

\_dateReturn.setText(((TextView)table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol5)).getText().toString());

}

});

if(flag){

SQLiteDatabase database = null;

try{

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_READER, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.PASSWORD));

if(data.equals(passData)){

exist = true;

break;

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(!exist){

throw new Exception();

}

}

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_BOOK, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER));

if(data.equals(numData)){

exist = true;

break;

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(!exist){

throw new Exception();

}

}

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_BOOK, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER));

if(data.equals(numData)){

int yearBook = cursor.getInt(cursor.getColumnIndex(ManagerDatabase.YEAR));

if(yearBook > Integer.valueOf(issueData.split("\\.")[2])){

exist = true;

break;

}

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(exist){

throw new Exception();

}

}

database = dbManager.getWritableDatabase();

database.setForeignKeyConstraintsEnabled(true);

ContentValues contentValues = new ContentValues();

contentValues.put(ManagerDatabase.ID, id);

contentValues.put(ManagerDatabase.REGISTER\_REGNUM, numData);

contentValues.put(ManagerDatabase.REGISTER\_PASSWORD, passData);

contentValues.put(ManagerDatabase.DATA\_ISSUE, issueData);

contentValues.put(ManagerDatabase.DATA\_RETURN, returnData);

database.insert(ManagerDatabase.TABLE\_REGISTER, null, contentValues);

database.close();

} catch (Exception e){

Toast.makeText(RegisterActivity.this, "Ошибка: невозможно добавить запись в базу данных! В данной записи содержиться не существующие паспортные" +

" данные, регистрационный номер или данные содержат некорректную дату!", Toast.LENGTH\_LONG).show();

toBaseAllCells(table, true);

if(database != null)

database.close();

return;

}

}

table.addView(row);

}

public void refactorRow(TableLayout table, String issueData, String returnData) throws Exception{

if((table == null)

|| (issueData == null) || (returnData == null)

|| (issueData.length() == 0) || (returnData.length() == 0)

|| (!isDateValidate(issueData)) || (!isDateValidate(returnData)))

throw new Exception("Ошибка: не корректные входные данные!");

DateFormat format = new SimpleDateFormat("dd.mm.yyyy");

if(format.parse(issueData).after(

format.parse(returnData)

)){

throw new Exception("Ошибка: дата выдачи не может быть позже даты возврата!");

}

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= \_table.getChildCount()))

throw new Exception("Ошибка: не выделена строка для изменения!");

SQLiteDatabase database = null;

try{

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_BOOK, null, null,

null, null, null, null);

boolean exist = false;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER));

if(data.equals(((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol2)).getText().toString())){

int yearBook = cursor.getInt(cursor.getColumnIndex(ManagerDatabase.YEAR));

if(yearBook > Integer.valueOf(issueData.split("\\.")[2])){

exist = true;

break;

}

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

if(exist)

throw new Exception();

}

database = dbManager.getWritableDatabase();

ContentValues cv = new ContentValues();

cv.put(ManagerDatabase.DATA\_ISSUE, issueData);

cv.put(ManagerDatabase.DATA\_RETURN, returnData);

database.update(ManagerDatabase.TABLE\_REGISTER, cv, "id = ?",

new String[]{

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol1)).getText().toString()

});

database.close();

}catch(Exception e){

Toast.makeText(RegisterActivity.this, "Ошибка: невозможно изменить существующую запись", Toast.LENGTH\_LONG).show();

toBaseAllCells(table, true);

if(database != null)

database.close();

return;

}

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol4)).setText(issueData);

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol5)).setText(returnData);

}

public void deleteChangeRow() throws Exception{

if((\_changeRowIndex <= 0) || (\_changeRowIndex >= \_table.getChildCount()))

throw new Exception("Ошибка: не выделена строка для удаления!");

SQLiteDatabase database = dbManager.getWritableDatabase();

try{

database.delete(ManagerDatabase.TABLE\_REGISTER, "id = ?",

new String[]{

((TextView) \_table.getChildAt(\_changeRowIndex).findViewById(R.id.registerCol1)).getText().toString()

});

database.close();

}catch (Exception e){

Toast.makeText(RegisterActivity.this, "Ошибка: невозможно удалить зарегистрированную запись!", Toast.LENGTH\_LONG).show();

toBaseAllCells(\_table, true);

if(database != null)

database.close();

return;

}

\_table.removeViewAt(\_changeRowIndex);

}

}

**SolutionActivity:**

package com.game.databaseandroid;

import android.content.ContentValues;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Color;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TableLayout;

import android.widget.TableRow;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class SolutionActivity extends AppCompatActivity {

private TableLayout \_table = null;

private TableLayout \_table2 = null;

private EditText \_valueMonth = null;

private EditText \_passData = null;

private Button \_btnReaderInfo = null;

private Button \_btnReadersInfo = null;

private static ManagerDatabase dbManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.solution\_layout);

\_table = (TableLayout) findViewById(R.id.tableSolution2);

\_table2 = (TableLayout) findViewById(R.id.tableSolution1);

\_valueMonth = (EditText) findViewById(R.id.monthValue);

\_passData = (EditText) findViewById(R.id.passDataD);

\_btnReaderInfo = (Button) findViewById(R.id.btnReaderInfo);

\_btnReadersInfo = (Button) findViewById(R.id.btnReadersInfo);

dbManager = new ManagerDatabase(SolutionActivity.this);

\_btnReaderInfo.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if((\_passData.getText() == null) || (\_passData.getText().toString().length() == 0))

return;

if(\_table.getChildCount() != 1){

while(\_table.getChildCount() != 1)

\_table.removeViewAt(1);

}

LayoutInflater inflater = LayoutInflater.from(SolutionActivity.this);

String pwd = \_passData.getText().toString().trim();

String fullName = "";

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_READER, null, null,

null, null, null, null);

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.PASSWORD));

if(data.equals(pwd)){

fullName = cursor.getString(cursor.getColumnIndex(ManagerDatabase.FULLNAME));

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

}

ArrayList<String> numbersBook = new ArrayList<>();

ArrayList<String> dataIssue = new ArrayList<>();

ArrayList<String> namesBook = new ArrayList<>();

ArrayList<Integer> years = new ArrayList<>();

ArrayList<Integer> pages = new ArrayList<>();

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_PASSWORD));

if(data.equals(pwd)){

numbersBook.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_REGNUM)));

dataIssue.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.DATA\_ISSUE)));

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

}

for(int i = 0; i < numbersBook.size(); i++){

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_BOOK, null, null,

null, null, null, null);

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER));

if(data.equals(numbersBook.get(i))){

namesBook.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.NAME)));

years.add(cursor.getInt(cursor.getColumnIndex(ManagerDatabase.YEAR)));

pages.add(cursor.getInt(cursor.getColumnIndex(ManagerDatabase.PAGES)));

break;

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

}

for(int i = 0; i < numbersBook.size(); i++){

TableRow row = (TableRow) inflater.inflate(R.layout.solution2\_template, null);

((TextView) row.findViewById(R.id.solution2Col1)).setText(fullName);

((TextView) row.findViewById(R.id.solution2Col2)).setText(namesBook.get(i));

((TextView) row.findViewById(R.id.solution2Col3)).setText(dataIssue.get(i));

((TextView) row.findViewById(R.id.solution2Col4)).setText(pages.get(i).toString());

((TextView) row.findViewById(R.id.solution2Col5)).setText(years.get(i).toString());

\_table.addView(row);

}

}

});

\_btnReadersInfo.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if((\_valueMonth.getText() == null) || (\_valueMonth.getText().toString().length() == 0))

return;

if(\_table2.getChildCount() != 1){

while(\_table2.getChildCount() != 1)

\_table2.removeViewAt(1);

}

LayoutInflater inflater = LayoutInflater.from(SolutionActivity.this);

int month = Integer.valueOf(\_valueMonth.getText().toString().trim());

ArrayList<String> fullnames = new ArrayList<>();

ArrayList<String> pwds = new ArrayList<>();

ArrayList<String> numbersBook = new ArrayList<>();

ArrayList<String> datasIssue = new ArrayList<>();

ArrayList<String> datasReturn = new ArrayList<>();

{

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String issue = cursor.getString(cursor.getColumnIndex(ManagerDatabase.DATA\_ISSUE)),

ret = cursor.getString(cursor.getColumnIndex(ManagerDatabase.DATA\_RETURN));

int yearI = Integer.valueOf(issue.split("\\.")[2]),

yearR = Integer.valueOf(ret.split("\\.")[2]),

monthI = Integer.valueOf(issue.split("\\.")[1]),

monthR = Integer.valueOf(ret.split("\\.")[1]);

if(((monthR - monthI) + 12 \* (yearR - yearI)) > month){

datasIssue.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.DATA\_ISSUE)));

datasReturn.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.DATA\_RETURN)));

numbersBook.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_REGNUM)));

pwds.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_PASSWORD)));

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

}

for(int i = 0; i < pwds.size(); i++){

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_READER, null, null,

null, null, null, null);

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.PASSWORD));

if(data.equals(pwds.get(i))){

fullnames.add(cursor.getString(cursor.getColumnIndex(ManagerDatabase.FULLNAME)));

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

}

for(int i = 0; i < pwds.size(); i++){

TableRow row = (TableRow) inflater.inflate(R.layout.solution1\_template, null);

((TextView) row.findViewById(R.id.solution1Col1)).setText(fullnames.get(i));

((TextView) row.findViewById(R.id.solution1Col2)).setText(pwds.get(i));

((TextView) row.findViewById(R.id.solution1Col3)).setText(numbersBook.get(i));

((TextView) row.findViewById(R.id.solution1Col4)).setText(datasIssue.get(i));

((TextView) row.findViewById(R.id.solution1Col5)).setText(datasReturn.get(i));

try {

((TextView) row.findViewById(R.id.solution1Col6)).setText(String.valueOf(CountBooksByPwd(pwds.get(i))));

} catch (Exception e) {

e.printStackTrace();

}

\_table2.addView(row);

}

}

});

}

private int CountBooksByPwd(String pass) throws Exception {

if((pass == null) || (pass.length() != ReaderActivity.STD\_SIZE\_PASS))

throw new Exception("Ошибка: введены некорректные данные!");

SQLiteDatabase dbs = dbManager.getWritableDatabase();

Cursor cursor = dbs.query(ManagerDatabase.TABLE\_REGISTER, null, null,

null, null, null, null);

int count = 0;

if(cursor.moveToFirst()) {

while (!cursor.isAfterLast()) {

String data = cursor.getString(cursor.getColumnIndex(ManagerDatabase.REGISTER\_PASSWORD));

if(data.equals(pass)){

count++;

}

cursor.moveToNext();

}

}

cursor.close();

dbs.close();

return count;

}

}

**ManagerDatabase:**

package com.game.databaseandroid;

import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class ManagerDatabase extends SQLiteOpenHelper {

public static final String DATABASE\_NAME = "DataBaseSQL";

//Таблица читателей

public static final String TABLE\_READER = "Reader";

public static final String PASSWORD = "Number\_Password";

public static final String HOME = "Home\_Address";

public static final String FULLNAME = "Full\_Name";

//Таблица книг

public static final String TABLE\_BOOK = "Book";

public static final String REGISTER = "Number\_Register";

public static final String NAME = "Name\_Book";

public static final String YEAR = "Year\_Publishing";

public static final String PAGES = "Count\_Pages";

public static final String SECTION = "Section";

//Таблица регистрации

public static final String TABLE\_REGISTER = "RecordRegisterT";

public static final String ID = "id";

public static final String REGISTER\_REGNUM = "Register";

public static final String REGISTER\_PASSWORD = "Password";

public static final String DATA\_ISSUE = "Data\_Issue";

public static final String DATA\_RETURN = "Data\_Return";

private static final int DATABASE\_VERSION = 9;

public ManagerDatabase(@Nullable Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

private void updateDatabase(SQLiteDatabase db, int oldVersion, int newVersion) {

/\*if(newVersion == 1){

try{

db.execSQL("CREATE TABLE " + TABLE\_REGISTER + " ("

+ ID + " INTEGER PRIMARY KEY, " +

REGISTER\_REGNUM + " TEXT, " +

REGISTER\_PASSWORD + " TEXT, " +

DATA\_ISSUE + " TEXT, " +

DATA\_RETURN + " TEXT, " +

"FOREIGN KEY (" + REGISTER\_PASSWORD + ") REFERENCES " + TABLE\_READER + "(" + PASSWORD + "), " +

"FOREIGN KEY (" + REGISTER\_REGNUM + ") REFERENCES " + TABLE\_BOOK + "(" + REGISTER + "));"

);

}catch(Exception e){}

}else if(newVersion == 2){

try{

db.execSQL("CREATE TABLE " + TABLE\_READER +

" (" + PASSWORD + " TEXT PRIMARY KEY, "

+ HOME + " TEXT, " + FULLNAME + " TEXT);"

);

}catch(Exception e){}

}else if(newVersion == 3){

try{

db.execSQL("CREATE TABLE " + TABLE\_BOOK + " (" +

REGISTER + " TEXT PRIMARY KEY, " +

NAME + " TEXT, " +

YEAR + " INTEGER, " +

PAGES+ " INTEGER, " +

SECTION + " TEXT);");

}catch (Exception e){}

}\*/

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

{

updateDatabase(db, oldVersion, newVersion);

}

@Override

public void onCreate(SQLiteDatabase db) {

updateDatabase(db, 0, DATABASE\_VERSION);

}

}

8 Заключение

Были получены общие представления об объектном подходе к организации доступа к базе данных (БД), выполнению запросов и обработке результатов. Был получен практический опыт организации работы с БД средствами платформы java.

Было разработано мобильное Android-приложение с графическим пользовательским инетфейсом позволяющее просматривать и редактировать информацию из базы данных. Была организована и наполнена база данных в соответствии с предметной областью “Библиотека”.