УДК 004

Создание приложения RoomNotes для Android

Звайгзне Алексей Юрьевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается процесс разработки приложения для создания заметок с использованием СУБД SQLite, данное приложение позволит сохранять заметки и сохранять их в памяти устройства. Приложение разрабатывается в Android Studio с использованием библиотеки Room для работы с СУБД.
Ключевые слова: СУБД, SQLite, Room, Android Studio, Android

Creating the RoomNotes app for Android

Zvaigzne Alexey Yurievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the process of developing an application for creating notes using the SQLite DBMS, this application will allow you to save notes and save them to the device's memory. The application is being developed in Android Studio using the Room library for working with DBMS.
Keywords: DBMS, SQLite, Room, Android Studio, Android

1 Введение

1.1 Актуальность

Андройд платформа является одной из самых востребованных на текущий момент. Из-за доступности в исполнении программного обеспечения на различных языках программирования и простоты установки готового продукта на конечные устройства пользователей, данное направление является одним из самых востребованных на данный момент. Множество компаний делает свои продукты под мобильные устройства под управлением OS Android, а развитие самой системы добавляет новые функции и упрощает работу по установке для конечного пользователя.

В большинстве случаев, для создания приложения на Android используется среда разработки, самой популярной из них является Android Studio, а бесплатность среды и большое количество различных обучающих материалов сети позволяют быстро погрузиться в сферу разработки. Android Studio из коробки позволяет работа сразу с несколькими видами интегрированных сред разработки Kotlin, Java, C++.

Обновления, которые происходят на платформе Android, достаточно быстро становятся общедоступными и также быстро интегрируются в Android Studio, поэтому разработчикам не приходится переписывать свои приложения с нуля, а лишь исправить код в своей работе и заново скомпилировать приложение.

Большинство современных программ имеет встроенную систему управления базами данных (СУБД), позволяющую легко хранить и обрабатывать имеющиеся данные и использовать их в дальнейшем. Такие приложения необходимы в любом крупном проекте, создаваемом на любой платформе.

Очень часто в качестве модели базы данных используют SQLite, Android также может управлять данным видом базы данных. Основной особенностью выступает интерфейс управления базами – Room, который автоматизирует процесс создания базы данных, для работы с ним необходимо лишь передать название базы, имена и тип полей, а дальше библиотека сделает всё сама. При компиляции конечного приложения создается интерфейс со всеми переменными и готовая база данных. Данная библиотека доступна как на Java, так и на Kotlin.

1.2 Обзор исследований

Последняя версия дистрибутива платформы для разработчика Android Studio была скачена и установлена с официального сайта, так же можно изучить сопутствующую литературу в приложенных на сайте ссылках [1]. А.В. Шматко в своей статье обозревает и анализирует возможности инструментов для разработки приложений на устройства под управлением Android [2]. Ж. Ж. Елюбаева в статье описывает краткую историю развития платформы, а также разрабатывает приложение с использованием языка программирования Java [3]. Ф.К. Коновалов в статье использует Java для разработки Android приложение оптимизации поиска решения возможных трудностей при поиске пути между аудиториями ФГБОУ ВО «МГТУ «СТАНКИН»» [4]. И.А. Николаев в научной медицинской статье разрабатывается «Медицинский справочник» на Android с целью быстрого получения необходимой информации о лекарственных препаратах [5]. Д. А. Простоквашин в своей статье разрабатывал «Химический справочник» для Android, упрощающий процесс обучения химии [6]. Е. И. Николаев в своей работе разработал пособие содержащие теоретические аспекты разработки приложений для высокопроизводительных вычислительных систем [7]. В статье Е.Н. Поварницына описываются основы технологии баз данных на языке SQL, дающие вводные знания в данном направлении [8]. М.Ю. Грибанова-Подкина в своей работе использует подключение базы данных из JSP-страниц и сервлетов для разработки веб-приложения основанного на языке программирования Java [9]. А. Д. Нестерова использует библиотеку Room для реализации базы данных в Android приложений [10].

1.3 Цель исследования

Разработка Android приложения RoomNotes с использованием СУБД, реализуемых с помощью библиотеки Room. Конечное приложение сможет добавлять новые записи, удалять имеющиеся с помощью свайпа в сторону.

2 Материалы и методы

Приложение будет разрабатываться на Android Studio, для работы с СУБД SQLite используется библиотека Room и её ответвления.

3 Результаты и обсуждения

Большинство современных программ имеет встроенную систему управления базами данных (СУБД), позволяющую легко хранить и обрабатывать имеющиеся данные и использовать их в дальнейшем. Такие приложения необходимы в любом крупном проекте, создаваемом на любой платформе.

Очень часто в качестве модели базы данных используют SQLite, Android также может управлять данным видом базы данных. Основной особенностью выступает интерфейс управления базами – Room, который автоматизирует процесс создания базы данных, для работы с ним необходимо лишь передать название базы, имена и тип полей, а дальше библиотека сделает всё сама. При компиляции конечного приложения создается интерфейс со всеми переменными и готовая база данных. Данная библиотека доступна как на Java, так и на Kotlin.

Для начала работы необходимо скачать и установить Android Studio с официального сайта (рис. 1).

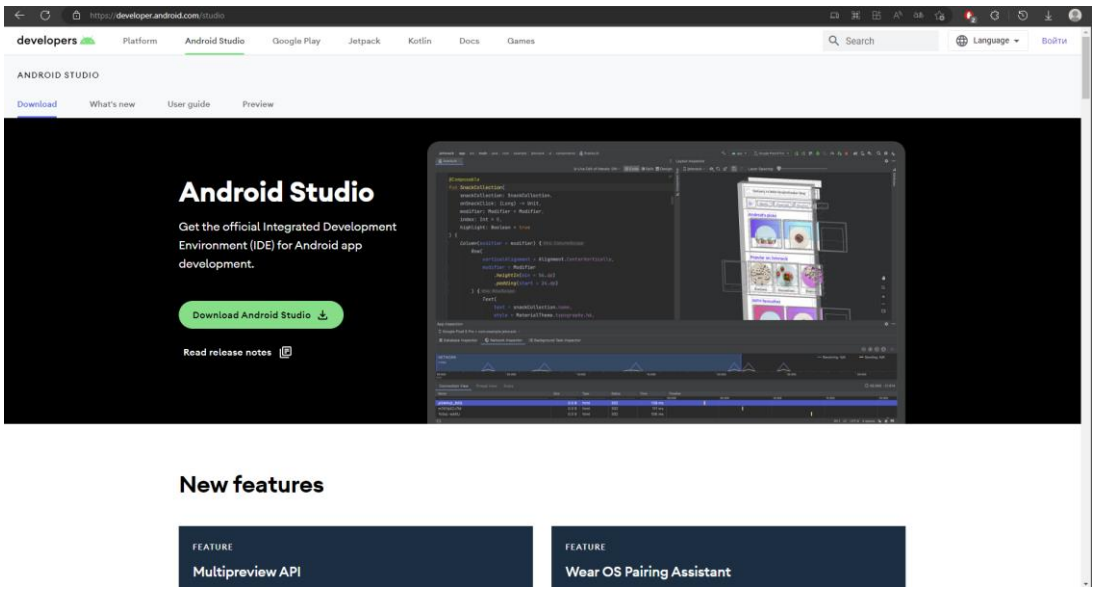


Рисунок 1. Страница загрузки Android Studio

После этого необходимо пройти простую процедуру установки Android Studio (рис. 2).

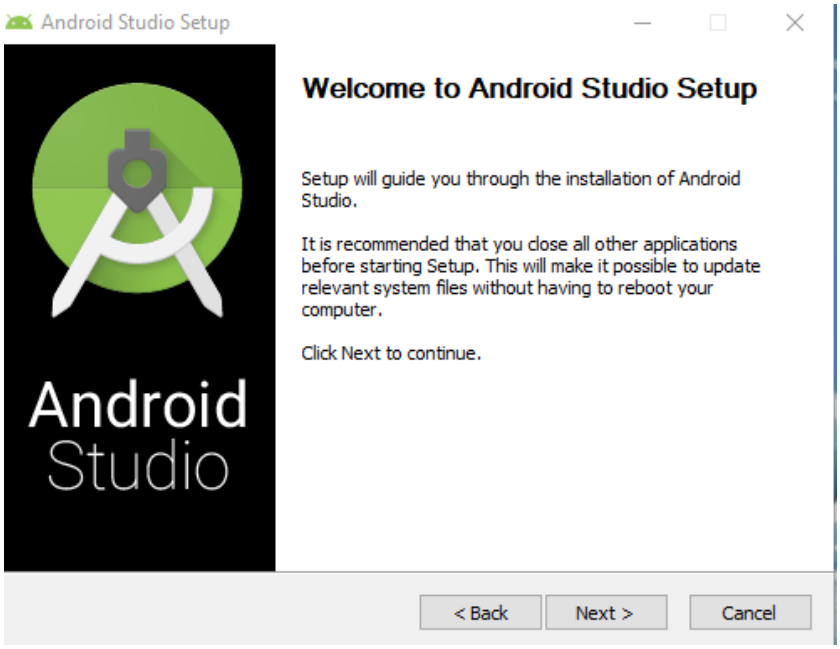


Рисунок 2. Процедура установки Android Studio

После этого необходимо запустить Android Studio, если программа не устанавливалась ранее, то пропускается процесс импорта настроек и приступить к скачиванию набора инструментов для разработки SDK, выбрать путь и установить.

После установки всех необходимых компонентов можно приступать к созданию приложения, при запуске Android Studio он автоматически предложит это сделать или открыть уже имеющийся проект (рис. 3).

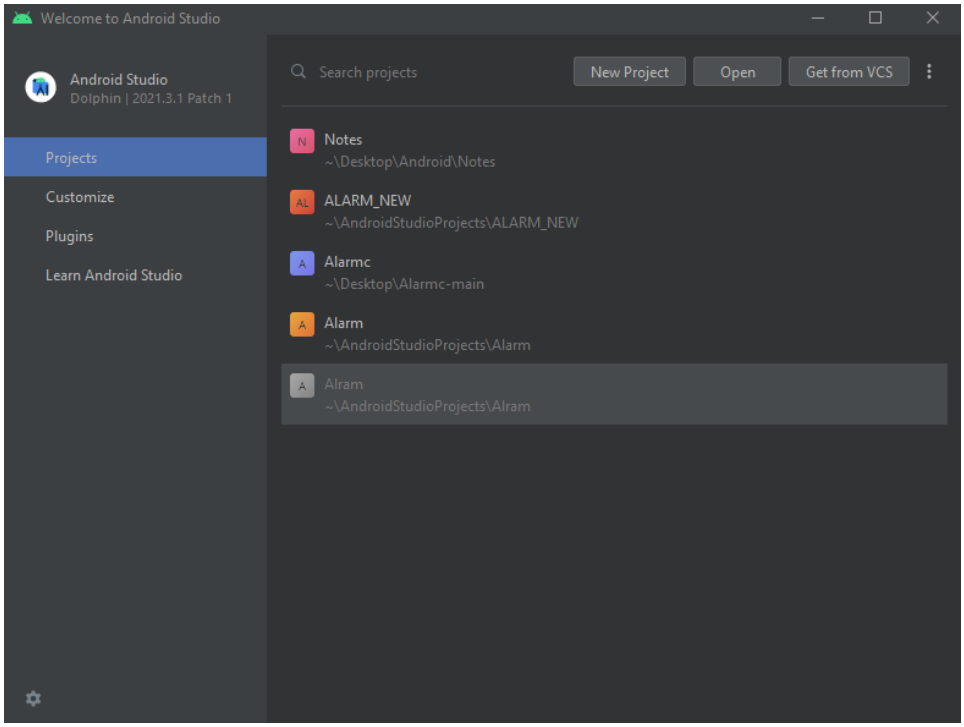


Рисунок 3. Вид окна «Welcome to Android Studio»

Чтобы создать новый проект в данном окне необходимо выбрать «New Project» и выбрать Empty Activity (пустая активность) (рис. 4).

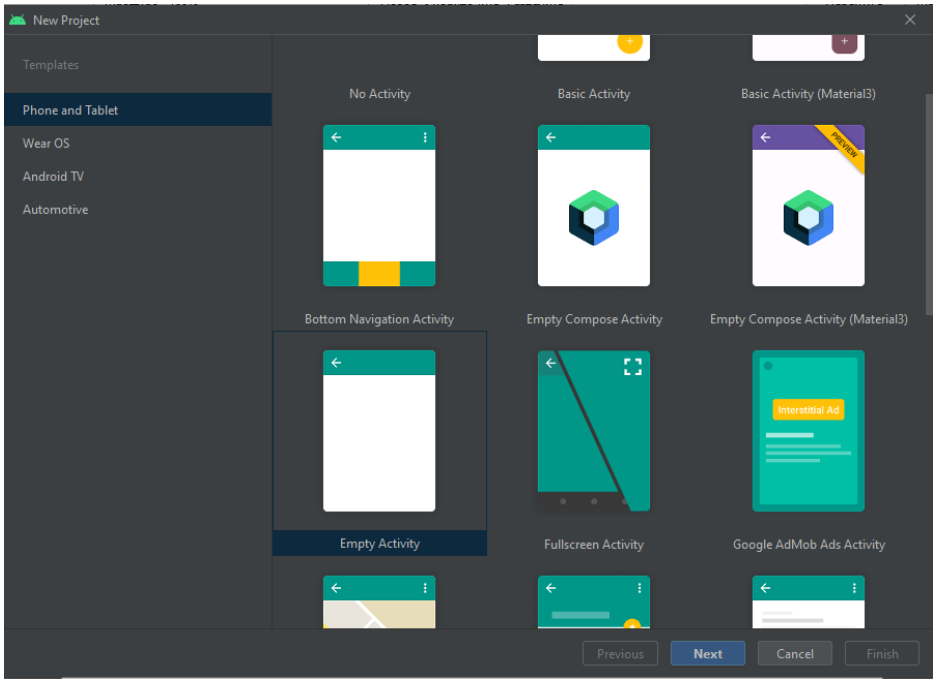


Рисунок 4. Вид окна создания нового проекта

В следующем окне необходимо дать имя приложению, выбрать язык программирования, в данном случае Java и версию Android SDK - 19 API или версия Android'a, а после нажать готово и дождаться компиляции базовых скриптов для работы с проектом (рис. 5-6).

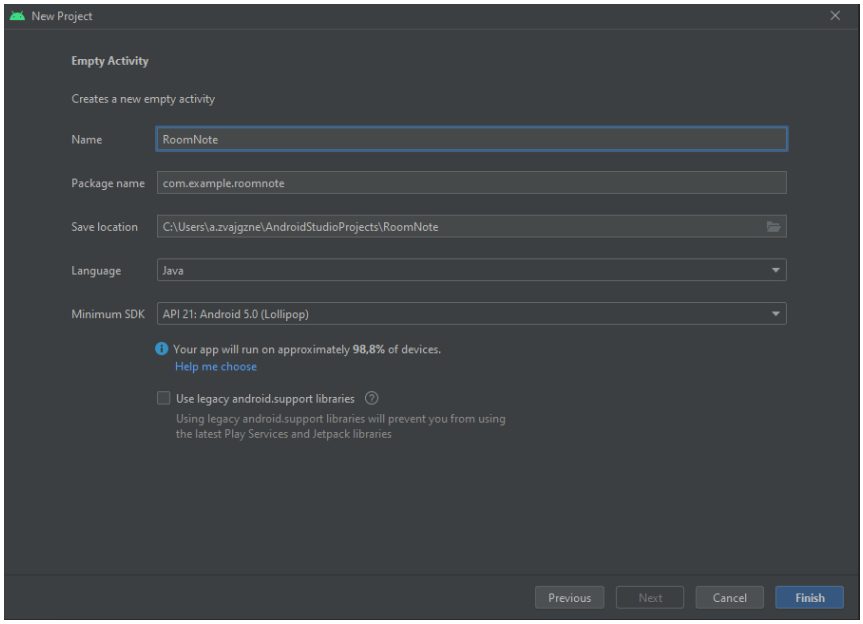


Рисунок 5. Окно создания новой активности в проекте

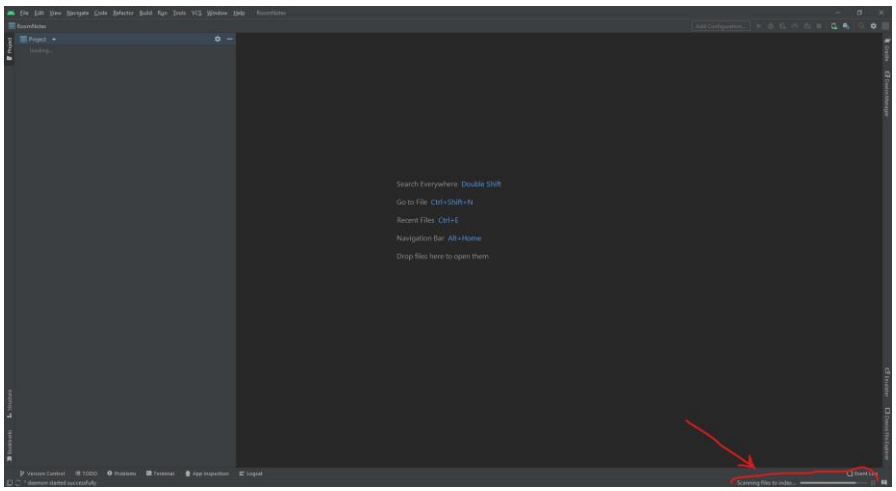


Рисунок 6. Вид окна разработки в момент компиляции и применении настроек проекта

Для начала работы с Room его библиотеку необходимо добавить в компилятор Gradle. Для этого в левой части окна, где располагаются основные файлы проекта, пройти по пути Gradle Scripts и выбрать файл Build.Gradle и открыть его (рис. 7).

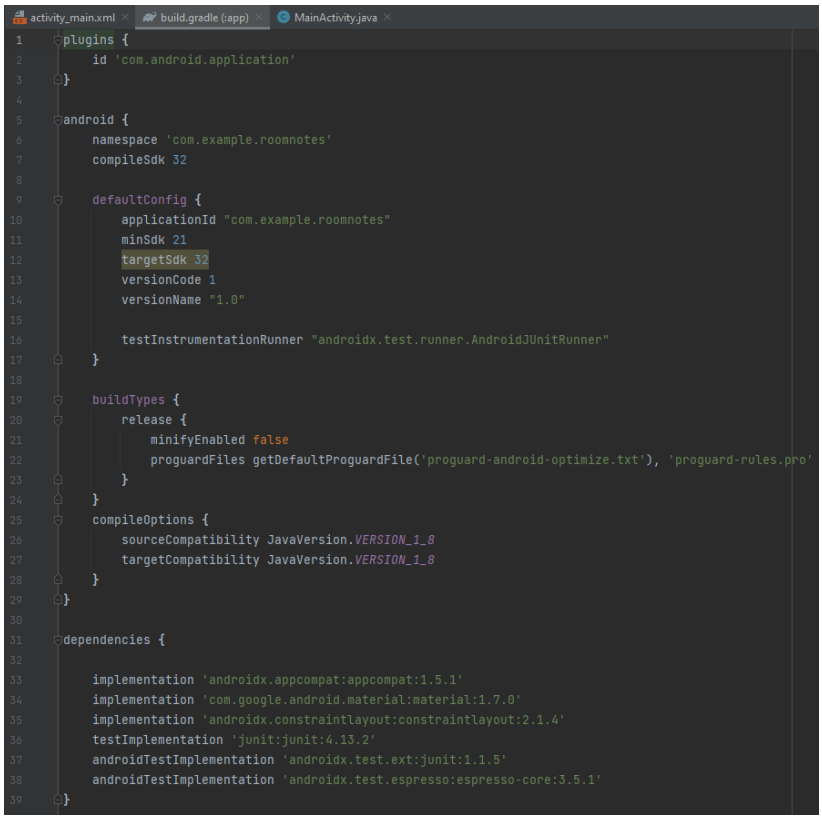


Рисунок 7. Содержимое файла Build

Сюда необходимо в зависимости (dependencies) добавить две реализации и процессор аннотаций (рис. 8):

```
implementation 'androidx.appcompat:appcompat:1.5.1'
implementation 'com.google.android.material:material:1.7.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
implementation 'androidx.room:room-common:2.4.3'
implementation 'androidx.room:room-runtime:2.4.3'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.5'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
annotationProcessor 'androidx.room:room-compiler:2.4.3'
```

Рисунок 8. Добавление новых библиотек

После этого синхронизируются скрипты, нажатием кнопки в верхней части окна «sync now» (рис. 9).

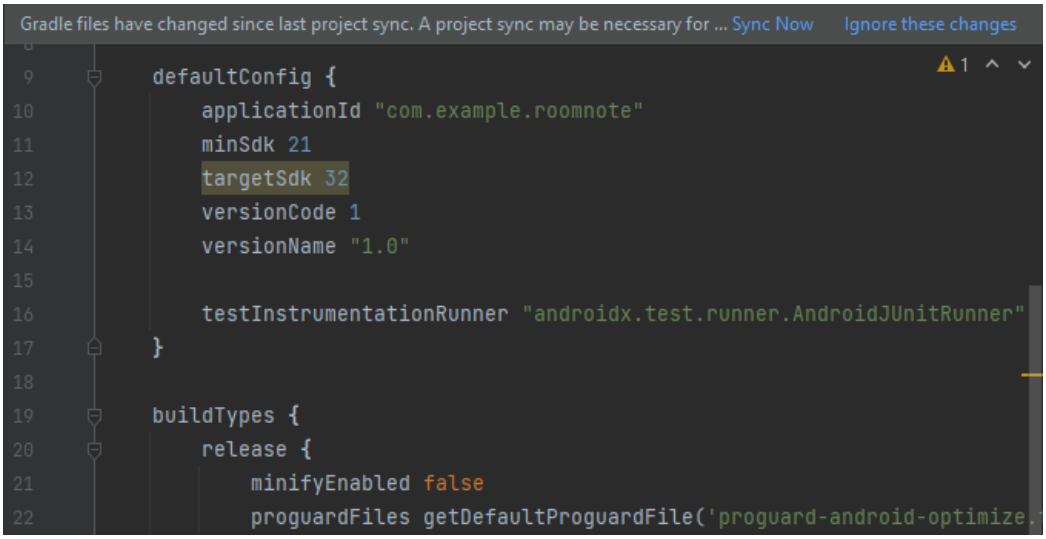


Рисунок 9. Синхронизация скриптов

После синхронизации всех скриптов можно приступать к работе. В этой работе будет предоставлен лишь код и состав приложения с кратким пояснением обозначений файлов.

Для удобства работы и нейтрализации большинства ошибок рекомендуется создавать классы, интерфейсы и рабочие слои отдельно от проекта, сразу в рабочие папки, так как иначе могут возникнуть проблемы дублирования связей, что приведёт к невозможности скомпилировать итоговое приложение или ошибкам внутри него.

Первым рекомендуется заполнить MainActivity.class (рис. 10) и activity_main.xml (рис. 11).

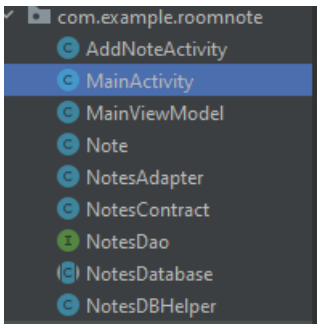


Рисунок 10. MainActivity.class в окне просмотра проектов

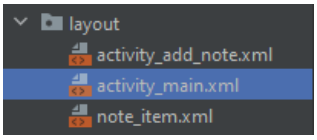


Рисунок 11. activity_main в папке слоев

Файл activity_main.xml как и все файлы, содержащиеся в слоях, описание содержимого окна в данном случае только recyclerview компонент-виджет, содержащий вложенные в него элементы, в этом проекте он будет содержать список созданных заметок, а также на слой будет помещена кнопка создания заметки.

Для того чтобы кнопке дать изображение его необходимо положить в папку drawable, в коде за это отвечает следующая строчка:

app:srcCompat="@drawable/plus"

Значок добавления можно использовать любой, это не принципиально, но желательно использовать графику без фона в формате PNG (рис. 12).

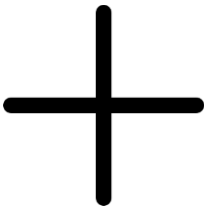


Рисунок 12. Значок добавления записи

Созданный слой должен выглядеть следующим образом (рис. 13).

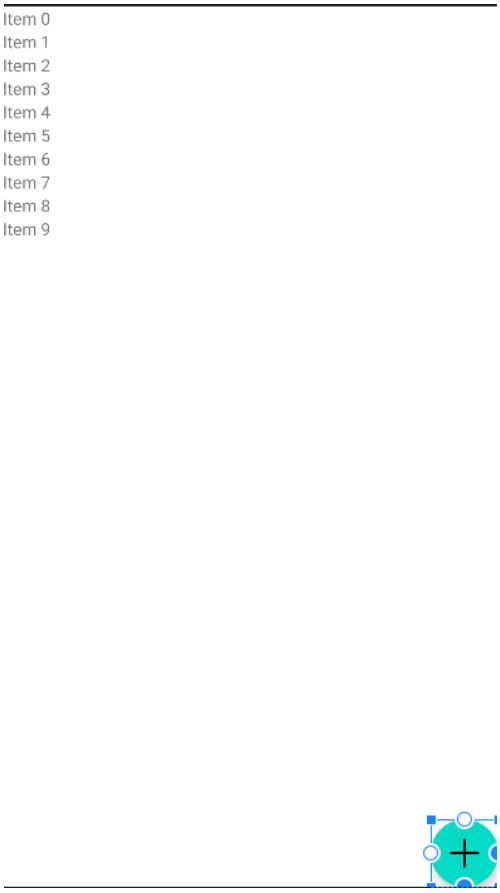
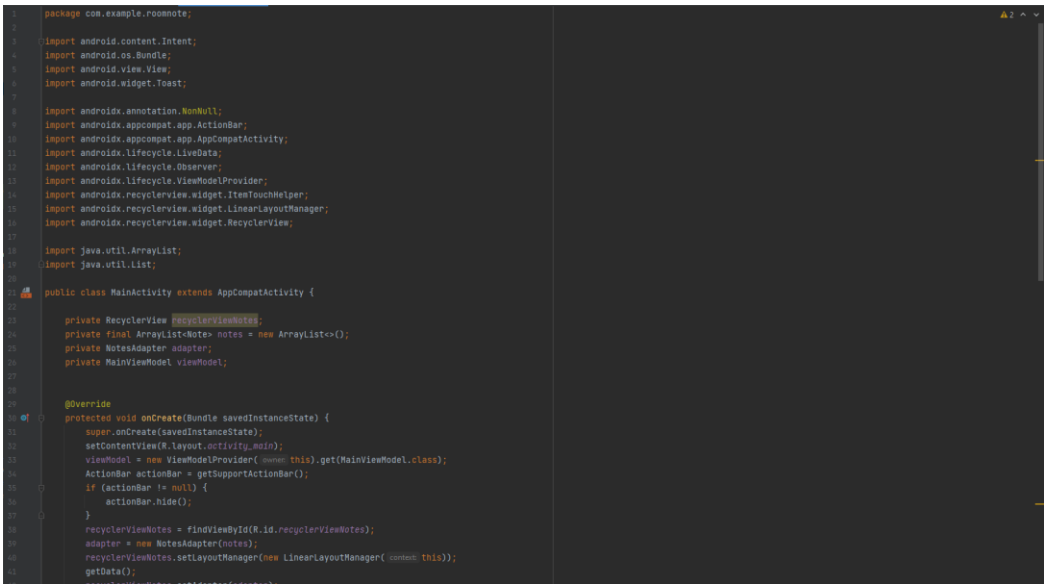


Рисунок 12. Вид слоя главного экрана приложения

После этого можно заполнить уже имеющийся класс MainActivity.Class (рис.13):



```
package com.example.roomnote;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerViewNotes;
    private final ArrayList<Note> notes = new ArrayList<>();
    private NotesAdapter adapter;
    private MainViewModel viewModel;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        viewModel = new ViewModelProvider( owner: this).get(MainViewModel.class);
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.hide();
        }
        recyclerViewNotes = findViewById(R.id.recyclerViewNotes);
        adapter = new NotesAdapter(notes);
        recyclerViewNotes.setLayoutManager(new LinearLayoutManager( context: this));
        getData();
```

Рисунок 13. Вид заполненной активности MainActivity

При копировании вид будет отличаться от изображенного на предыдущем скриншоте, так как остальные активности ещё не созданы, можно создавать дальше активности и слои, ошибки постепенно исчезнут.

В главной активности происходят основные действия, создается массив значений для recyclerview, значения берутся из БД, которая генерируется и помещаются в слой, который создается для отображения значений.

ViewModelProvider отвечает за хранение и отображение переменных из БД.

Adapter – отвечает за передачу и генерацию ячеек в БД, а такжеполучает и передает данные из БД.

ActionBar – это часть интерфейса каждом окне, использовался в старых версиях Android для отображения названия окна активности в вверху приложения. Данный код связанный с ActionBar – прячет ActionBar давая приложению больше пространства для отображения контента.

Метод ItemTouchHelper позволяет управлять вложенным объектом, в данном проекте можно свайпом влево-вправо удалять записи.

Новые слои можно создать нажатием правой кнопкой по соответствующей папке (рис. 14).

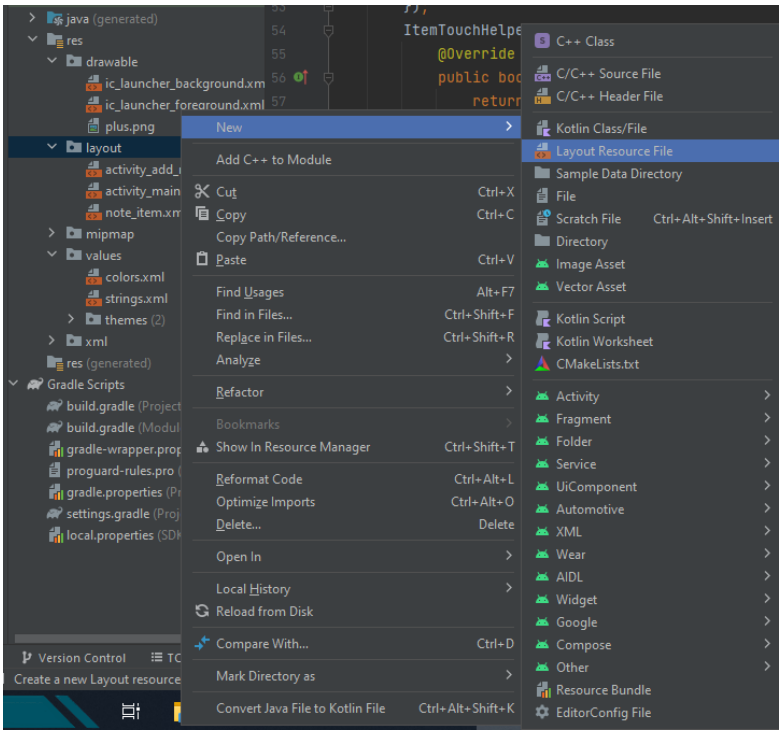


Рисунок 14. Создание нового слоя

Следующий слой, который необходимо создать это note_item.xml (рис. 15) содержащий следующий код (рис. 16):

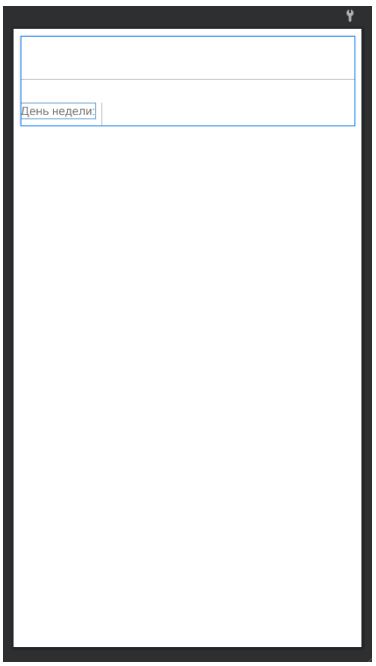


Рисунок 15. Вид слоя, содержащий запись

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/textViewTitle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:padding="16dp"
            android:textColor="@color/white"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/textViewDescription"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textViewTitle" />

        <TextView
            android:id="@+id/textViewDayOfWeek"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:paddingBottom="8dp"
            app:layout_constraintStart_toEndOf="@+id/textViewLabelDayOfWeek"
            app:layout_constraintTop_toBottomOf="@+id/textViewDescription" />
```

Рисунок 16. Код слоя с отображением ячейки

Сразу можно создать ещё один слой, отвечающий за создание новой записи activity_add_note.xml (рис. 17) содержащий следующий код (рис. 18):

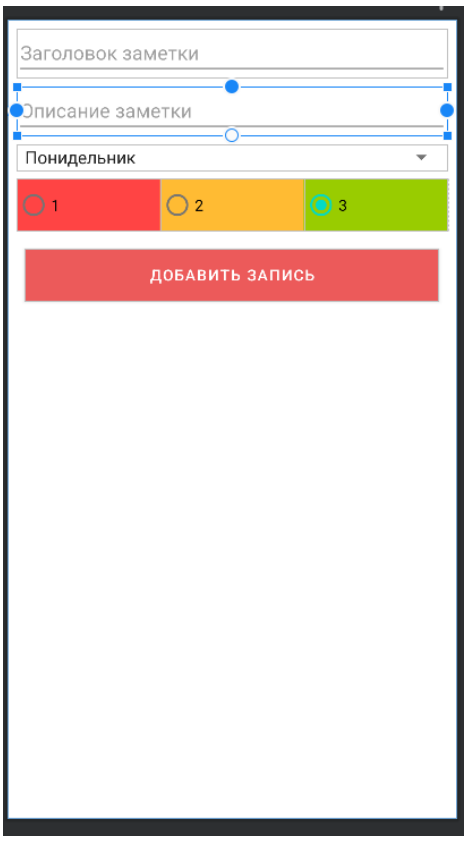


Рисунок 17. Вид окна создания записи

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddNoteActivity">

    <EditText
        android:id="@+id/editTextTitle"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Заголовок заметки"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editTextDescription"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:ems="10"
        android:gravity="start|top"
        android:inputType="textMultiLine"
        android:hint="Описание заметки"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextTitle" />

    <Spinner
        android:id="@+id/spinnerDaysOfWeek"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:entries="@array/days_of_week"
```

Рисунок 18. Вид кода слоя создания записи

Данное окно содержит два поля для ввода названия заметки и её описания, выпадающий список для выбора дня недели и выбора приоритета.

Дальше можно приступать к созданию активностей и интерфейсов: AddNoteActivity, MainViewModel, Note, NotesAdapter, NotesContacrt, NotesDao, NotesDatabase.

Для создания классов необходимо правой кнопкой нажать по папке проекта и выбрать соответствующий пункт, дать название классу и нажать Enter (рис. 19).

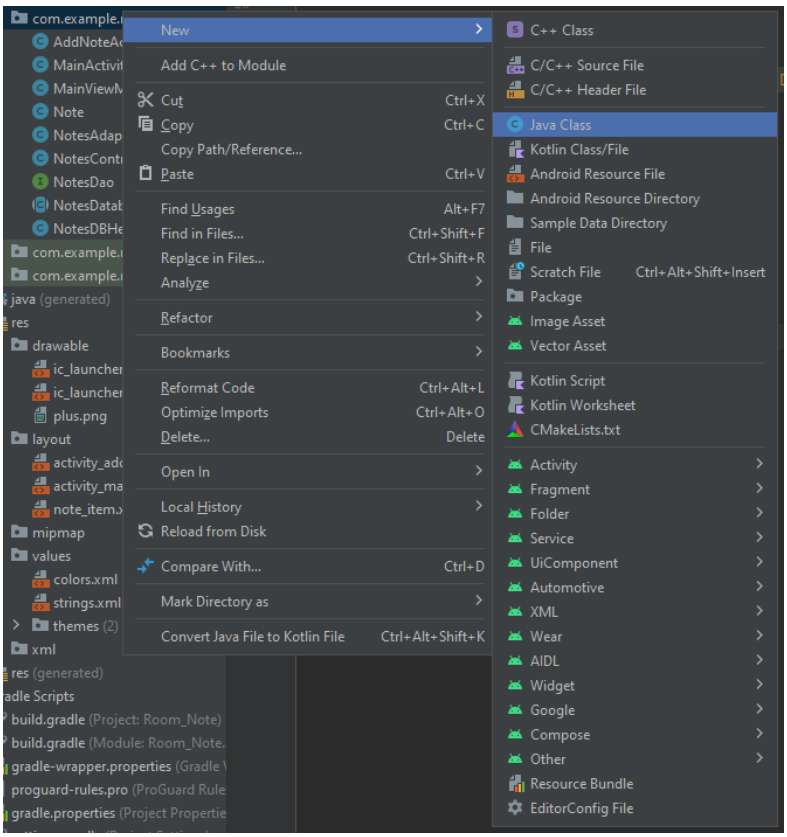


Рисунок 19. Создание нового класса

Класс AddNoteActivity содержит следующий код (рис. 20):

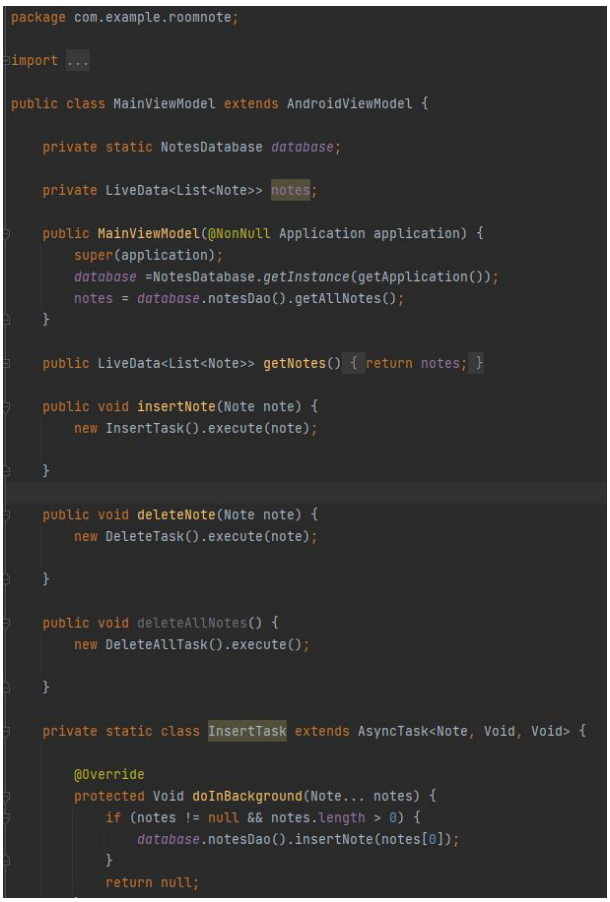
```java
package com.example.roomnote;

import ...

public class AddNoteActivity extends AppCompatActivity {

    private EditText editTextTitle;
    private EditText editTextDescription;
    private Spinner spinnerDaysOfWeek;
    private RadioGroup radioGroupPriority;

    private MainViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_note);
        viewModel = new ViewModelProvider( owner: this).get(MainViewModel.class);
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.hide();
        }
        editTextDescription =findViewById(R.id.editTextDescription);
        editTextTitle = findViewById(R.id.editTextTitle);
        spinnerDaysOfWeek = findViewById(R.id.spinnerDaysOfWeek);
        radioGroupPriority = findViewById(R.id.radioGroupPriority);

    }

    public void OnClickSaveNote(View view) {
        String title = editTextTitle.getText().toString().trim();
        String description = editTextDescription.getText().toString().trim();
        int dayOfWeek = spinnerDaysOfWeek.getSelectedItemPosition();
        int radioButtonId = radioGroupPriority.getCheckedRadioButtonId();
        RadioButton radioButton = findViewById(radioButtonId);
        int priority = Integer.parseInt(radioButton.getText().toString());
        if (isFilled(title, description)) {
            Note note = new Note(title, description, dayOfWeek, priority);
            viewModel.insertNote(note);
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
```

Рисунок 20. Вид кода активности, отвечающей добавление записи

Данная активность помещает данные в ViewModel и через провайдер передает в главную активность. Также данная активность содержи проверку заполненности полей «Название» и «Описание», если они пустые, то запись не сохраняется, а пользователю отображается сообщение о том есть не заполненные поля.

Следующим классом будет MainViewModel, отвечающий за удержание модели в памяти приложения и только после передает её в другой обработчик.

Она содержит следующий код (рис. 21):



```
package com.example.roomnote;

import ...

public class MainViewModel extends AndroidViewModel {

    private static NotesDatabase database;

    private LiveData<List<Note>> notes;

    public MainViewModel(@NonNull Application application) {
        super(application);
        database =NotesDatabase.getInstance(getApplication());
        notes = database.notesDao().getAllNotes();
    }

    public LiveData<List<Note>> getNotes() { return notes; }

    public void insertNote(Note note) {
        new InsertTask().execute(note);

    }

    public void deleteNote(Note note) {
        new DeleteTask().execute(note);

    }

    public void deleteAllNotes() {
        new DeleteAllTask().execute();

    }

    private static class InsertTask extends AsyncTask<Note, Void, Void> {

        @Override
        protected Void doInBackground(Note... notes) {
            if (notes != null && notes.length > 0) {
                database.notesDao().insertNote(notes[0]);
            }
            return null;
```

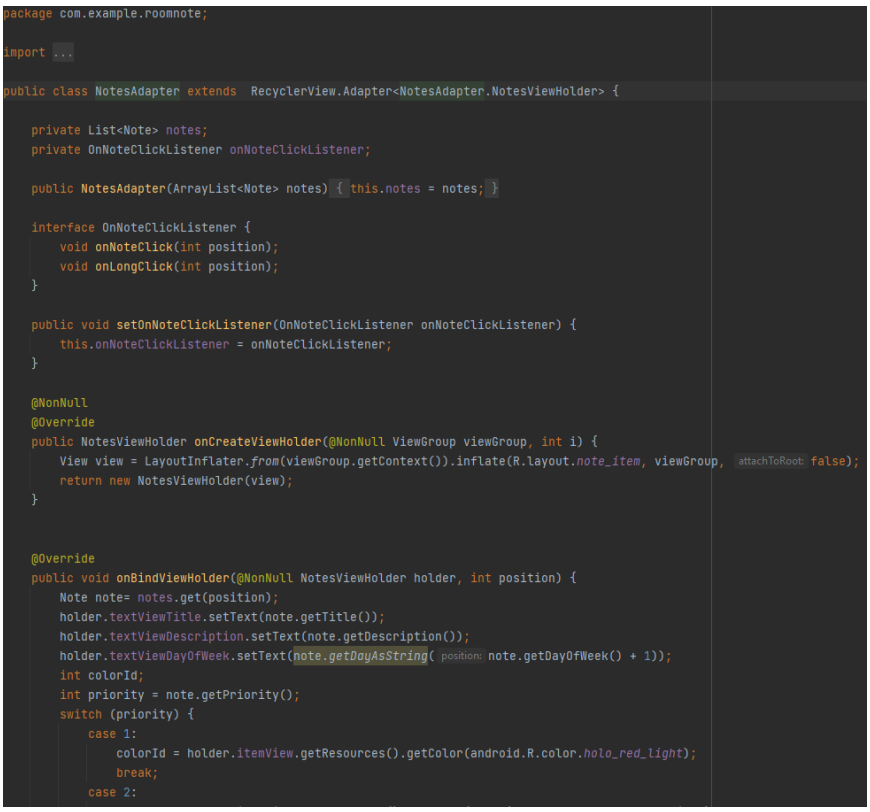
Рисунок 21. Код, отвечающий за хранение модели записи

Дальше необходимо добавить класс, отвечающий за саму запись имеющий следующий код (рис. 22) в нем расписаны записи, типы полей и их содержимое, например, «Дни недели»:

```java
package com.example.roomnote;

import ...

@Entity(tableName = "notes")
public class Note {
    @PrimaryKey(autoGenerate = true)
    private int id;
    private String title;
    private String description;
    private int dayOfWeek;
    private int priority;

    public Note(int id, String title, String description, int dayOfWeek, int priority) {
        this.id = id;
        this.title = title;
        this.description = description;
        this.dayOfWeek = dayOfWeek;
        this.priority = priority;
    }
    @Ignore
    public Note(String title, String description, int dayOfWeek, int priority) {
        this.title = title;
        this.description = description;
        this.dayOfWeek = dayOfWeek;
        this.priority = priority;
    }

    public int getId() { return id; }

    public String getTitle() { return title; }

    public String getDescription() { return description; }

    public int getDayOfWeek() { return dayOfWeek; }

    public int getPriority() { return priority; }

    public void setId(int id) { this.id = id; }

    public void setTitle(String title) { this.title = title; }
```

Рисунок 22. Содержимое класса Notes

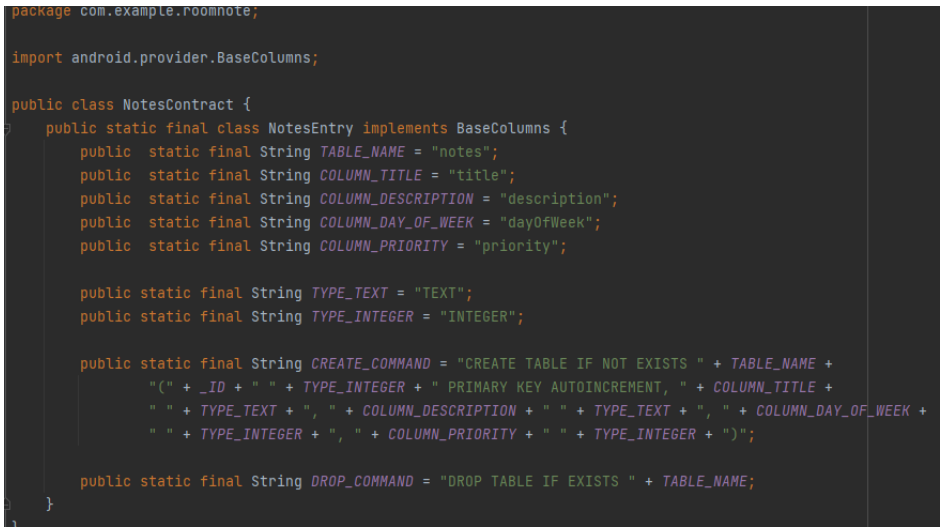
Класс NotesAdapter получает данные из полей и передает их ViewModel и отвечает за отображение в главной активности, а так же следит за обновлением данных в конечной таблице, за это отвечает метод notifyDataSetChanged() (рис. 23):



```
package com.example.roomnote;

import ...

public class NotesAdapter extends  RecyclerView.Adapter<NotesAdapter.NotesViewHolder> {

    private List<Note> notes;
    private OnNoteClickListener onNoteClickListener;

    public NotesAdapter(ArrayList<Note> notes) { this.notes = notes; }

    interface OnNoteClickListener {
        void onNoteClick(int position);
        void onLongClick(int position);
    }

    public void setOnNoteClickListener(OnNoteClickListener onNoteClickListener) {
        this.onNoteClickListener = onNoteClickListener;
    }

    @NonNull
    @Override
    public NotesViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.note_item, viewGroup,  attachToRoot: false);
        return new NotesViewHolder(view);
    }


    @Override
    public void onBindViewHolder(@NonNull NotesViewHolder holder, int position) {
        Note note= notes.get(position);
        holder.textViewTitle.setText(note.getTitle());
        holder.textViewDescription.setText(note.getDescription());
        holder.textViewDayOfWeek.setText(note.getDayAsString( position: note.getDayOfWeek() + 1));
        int colorId;
        int priority = note.getPriority();
        switch (priority) {
            case 1:
                colorId = holder.itemView.getResources().getColor(android.R.color.holo_red_light);
                break;
            case 2:
```

Рисунок 23. Код адаптера записей

Класс NotesContract отвечает за вид базы данных, название таблицы, описание строк, столбцов для SQLite имеющий следующий код (рис. 24):



```
package com.example.roomnote;

import android.provider.BaseColumns;

public class NotesContract {
    public static final class NotesEntry implements BaseColumns {
        public  static final String TABLE_NAME = "notes";
        public  static final String COLUMN_TITLE = "title";
        public  static final String COLUMN_DESCRIPTION = "description";
        public  static final String COLUMN_DAY_OF_WEEK = "dayOfWeek";
        public  static final String COLUMN_PRIORITY = "priority";

        public static final String TYPE_TEXT = "TEXT";
        public static final String TYPE_INTEGER = "INTEGER";

        public static final String CREATE_COMMAND = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME +
                "(" + _ID + " " + TYPE_INTEGER + " PRIMARY KEY AUTOINCREMENT, " + COLUMN_TITLE +
                " " + TYPE_TEXT + ", " + COLUMN_DESCRIPTION + " " + TYPE_TEXT + ", " + COLUMN_DAY_OF_WEEK +
                " " + TYPE_INTEGER + ", " + COLUMN_PRIORITY + " " + TYPE_INTEGER + ")";

        public static final String DROP_COMMAND = "DROP TABLE IF EXISTS " + TABLE_NAME;
    }
```

Рисунок 24. Код NotesContract

Дальше необходимо создать интерфейс для БД, чтобы создать интерфейс и после выбора имени указывается тип класса, Interface (рис. 25).

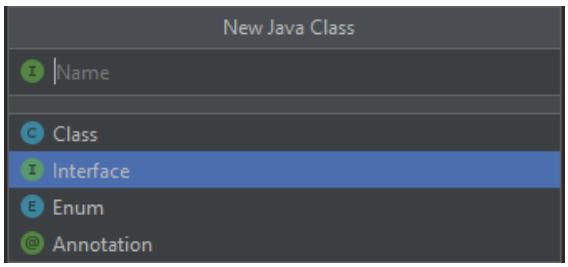


Рисунок 25. Вид окна создания интерфейса

содержащий три команды, добавление записи, удаление записи, удаление всех записей, за работу интерфейса отвечает библиотека room и её вложения, а также сортирует записи по столбцу «ДниНедели» по убыванию содержащий следующий код (рис. 26):

```
package com.example.roomnote;

import ...

@Dao
public interface NotesDao {
    @Query("SELECT * FROM notes ORDER BY dayOfWeek DESC")
    LiveData<List<Note>> getAllNotes();

    @Insert
    void insertNote(Note note);
    @Delete
    void deleteNote(Note note);
    @Query("DELETE FROM notes")
    void deleteAllNotes();
}
```

Рисунок 26. Код работающий с записями

Следующий класс NotesDatabase содержит тело самой базы данных и имеет лишь один метод, не дающий другим потокам перехватывать активность над базой synchronized (LOCK) (рис. 27):

```
package com.example.roomnote;

import android.content.Context;

import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;

@Database(entities = {Note.class}, version = 1, exportSchema = false)
public abstract class NotesDatabase extends RoomDatabase {
    private static NotesDatabase database;
    private static final String DB_NAME = "notes2.db";
    private static final Object LOCK = new Object();

    public static NotesDatabase getInstance(Context context) {
        synchronized (LOCK) {
            if (database == null) {
                database = Room.databaseBuilder(context, NotesDatabase.class, DB_NAME)
                        .build();
            }
        }
        return database;
    }

    public abstract NotesDao notesDao();
}
```

Рисунок 27. Вид кода тела базы данных

Дальше необходимо обновить содержимое файла strings.xml расположенного в директории res – values – strings.xml (рис. 28):

```xml
<resources>
    <string name="app_name">Notes</string>
    <string name="label_day_of_week">День недели:</string>
    <string name="header_note">Заголовок заметки</string>
    <string name="note_description">Описание заметки</string>
    <string name="priority_1">1</string>
    <string name="priority_2">2</string>
    <string name="priority_3">3</string>
    <string name="button_add_note">Добавить запись</string>
    <string name="warning_fill_fields">Все поля должны быть заполнены</string>

    <string-array name="days_of_week">
        <item>Понидельник</item>
        <item>Вторник</item>
        <item>Среда</item>
        <item>Четверг</item>
        <item>Пятница</item>
        <item>Суббота</item>
        <item>Воскресенье</item>
    </string-array>
</resources>
```

Рисунок 28. Содержимое файла, отвечающего за строковые ресурсы

Осталось отредактировать AndroidManifest.xml лежащий в корне проекта и файлы темы приложения (рис. 29-30):

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Notes"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.RoomNote"
        tools:targetApi="31">
        <activity
            android:name=".AddNoteActivity"
            android:exported="false">
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

Рисунок 29. Вид манифеста

Данный файл хранит глобальные переменные, активности, правила и прочие значения.

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.RoomNote" parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="21">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Рисунок 30. Вид файла темы

Здесь необходимо поменять только DarkActionBar на NoActionBar.

По итогу можно скомпилировать и проверить созданное приложение. Можно добавить запись, с помощью свайпа удалять записи. При перезапуске приложения данные загружаются из памяти телефона (рис 31).

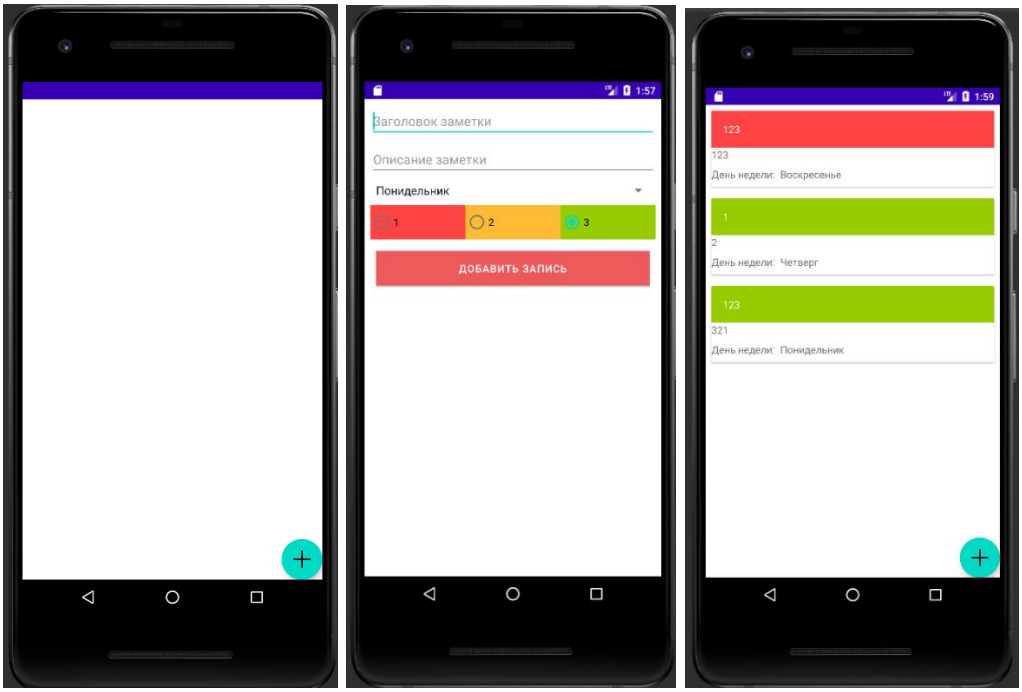


Рисунок 31. Результат работы приложения

Выводы

Создание приложения с интегрированной базой данных гораздо более трудозатратный процесс, но имеет ряд преимуществ против обычных программ, позволяет в длительном промежутке времени хранить и обрабатывать данные, удалять, добавлять и изменять их.

Данная программа может быть усовершенствована путем добавления в неё возможности создавать уведомление или будильник, позволяющие отслеживать состояние указанной заметки.

Библиографический список

1. Сайт загрузки Android Studio с ссылками на форум URL: https://developer.android.com/studio
2. Шматко А. В., Федорченко В. Н. Обзор и анализ инструментов разработки мобильных приложений для ОС Android//Инновации в науке. 2016. №. 5-1 (54). С. 59-73.
3. Елюбаева Ж. Ж. Разработка приложения для Android на Java//Международный научно-исследовательский журнал. 2015. №. 5-2 (36). С. 60-61.
4. Коновалов Ф. К., Ефромеева Е. В. Использование языка Java для разработки Android-приложения//Техническое творчество молодежи. 2021. №. 3. С. 45-48.
5. Николаев И. А., Березовская Е. М. Разработка мобильного приложения «Медицинский справочник» на платформе Android с использованием языка Java//Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. 2018. С. 275-276.
6. Простокишин Д. А. Разработка приложения "Химический справочник" для ОС Android // Молодежь XXI века: шаг в будущее. 2019. С. 214-215.
7. Николаев Е. И. Базы данных в высокопроизводительных информационных системах. 2016.
8. Поварницын Е. Н. Введение в базу данных на языке SQL//Форум молодых ученых. 2019. №. 8. С. 241-257.
9. Грибанова-Подкина М. Ю. Технологии подключения к базе данных из JSP-страниц и сервлетов веб-приложений Java // Кибернетика и программирование. 2019. №. 2. С. 73-85.
10. Нестерова А. Д., Шегушев Н. Б., Кумратова А. М. Инструментальные средства реализации баз данных для ОС Android на примере библиотеки «Room» // Цифровизация экономики: направления, методы, инструменты. 2021. С. 317-321.