Федеральное государственное бюджетное образовательное учреждение высшего образования

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

ОТЧЕТ

по производственной практике

На материалах АНО «Центр интернет-образования»

Студент _____ Королев А.С.

Группа 31ПИ

Направление 09.03.03 Прикладная информатика

Руководитель практики
от университета _____ А.П. Гордиенко

Руководитель практики
от профильной организации _____ Е. А. Сурова

Оценка защиты: _____

Орел 2025

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Направление подготовки/специальность 09.03.03 Прикладная информатика

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на производственную практику

для обучающегося Королев Антона Сергеевича

2 курса очной формы обучения, группы 31ПИ

Место прохождения практики: ОГУ им.Тургенева

Срок прохождения практики с «16» июня 2025г. по «12» июля 2025 г.

Содержание практики (вопросы, подлежащие изучению):
- изучение предметной области.
- создание схем и диаграмм.
- разработка приложения.

Планируемые результаты практики:
- пояснительная записка с описанием работы приложения
- готовое Android - приложение.

Руководитель практики от университета _____ А.П. Гордиенко

Согласовано:
Руководитель практики от профильной организации _____ Е. А. Сурова

Задание принял: _____ «16» июня 2025 г.
(подпись обучающегося)

СОВМЕСТНЫЙ РАБОЧИЙ ГРАФИК (ПЛАН)
проведения производственной практики

Обучающегося 2 курса очной формы обучения, группы 31ПИ

Направление подготовки 09.03.03 Прикладная информатика

Место прохождения практик АНО «Центр интернет-образования»

Срок прохождения практики с «16» июня 2025 г. по «12» июля 2025 г.

№ п/п	Наименование этапа проведения практики*	Вид работ	Срок прохождения этапа практики	Форма отчетности	Отметка о выполнении
1	Подготовительный (организационный) этап	1.Организационное собрание для разъяснения целей, задач, содержания и порядка прохождения практики. 2. Инструктаж по технике безопасности. 3. Выдача индивидуального задания.	16.06.25	Бланк индивидуальн ого задания	
2	Основной этап	1. Ознакомление с предметной областью. 2. Сбор информации о методах разработки Android - приложений. 3. Проектирование Android - приложение. 4.Разработка Android - приложения.	17.06.25 – 10.07.25	Дневник	
3	Заключительный этап	Составление отчета по практике. Защита отчета по практике с представлением материалов конкретной профильной организации.	11.07.25 – 12.07.25	Отчет	

* Этапы проведения практики и виды работ по ним могут меняться в зависимости от направления подготовки

Руководитель практики от университета _____ А.П. Гордиенко

Руководитель практики от профильной организации _____ Е. А. Сурова

С рабочим графиком (планом) ознакомлен:

Обучающийся _____ А. С. Королев

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

ДНЕВНИК

производственной практики студента

Королев Антон Сергеевич

Курс 2

Группа 31ПИ

Место прохождения практики: АНО «Центр интернет-образования»

Сроки проведения практики: 16.06.2025 – 12.07.2025

Руководитель практики
от университета _____ А.П. Гордиенко

Руководитель практики
от профильной организации _____ Е. А. Сурова

Начало практики
«16» июня 2025 года
Окончание практики
«12» июля 2025 года

Дата	Работа, выполненная студентом	ФИО руководителя (или контролирующего лица)
16.06.25	Прослушано организационное собрание, получено индивидуальное задание.	Сурова Е.А.
17.06.25 — 20.06.25	Изучена предметная область выбранной темы.	Сурова Е.А.
21.06.25 – 24.06.25	Собрана информация о методах разработки Android - приложений.	Сурова Е.А.
25.06.25 — 28.06.25	Проектирование концептуальной схемы базы данных.	Сурова Е.А.
29.06.25 – 02.07.25	Создание диаграмм и схем.	Сурова Е.А.
03.07.25 — 06.07.25	Разработка приложения.	Сурова Е.А.
07.07.25 – 09.07.25	Тестирование приложения.	Сурова Е.А.
10.07.25 – 11.07.25	Составление дневника и отчета учебной практики.	Сурова Е.А.

Студент Королев А. С. (_____)

Руководитель практики
от университета _____ Гордиенко А.П.

Руководитель практики
от профильной организации _____ Сурова Е.А.

Отзыв руководителя практики от профильной организации

Студент Королев Антон Сергеевич

Направление 09.03.03 Прикладная информатика

Направленность (профиль) Интеллектуальная обработка данных

Место прохождения практики: АНО «Центр интернет-образования»

Сроки проведения практики: 16.06.2025 – 12.07.2025

Отражается текст отзыва руководителя практики от профильной организации, в котором отражаются следующие вопросы:

1. В какой степени отчет по практике отвечает требованиям выданного студенту задания, все ли вопросы, поставленные в нем, разрешены и освещены в достаточной степени.
2. Отношение практиканта к выполняемой работе, степень выполнения поручений, качественный уровень и степень подготовленности студента к самостоятельному выполнению отдельных заданий.
3. Грамотность изложения отчета по практике, объем и степень использования отечественной и иностранной литературы и материалов, справочников.
4. Оценка дисциплины студента(ки) во время практики (удовлетворительно/неудовлетворительно). Были ли случаи нарушения трудовой дисциплины и какие меры приняты по ним руководителем практики и руководством предприятия. Наличие отрицательных черт, действий, проявлений, характеризующих студента с негативной стороны в период прохождения практики.
5. Оценка выполнения программы практики и индивидуального задания в целом: отлично, хорошо, удовлетворительно, неудовлетворительно. В случае невыполнения отдельных пунктов программы, указать конкретно, какие пункты не выполнены и причину их невыполнения.

Руководитель практики от АНО «Центр интернет-образования»

Преподаватель Сурова Е.А. «____» _____ 20__г.

Содержание

Введение

В современном мире с его высоким темпом жизни и постоянным потоком информации, эффективное управление задачами становится критически важным навыком. Множество дел, от рабочих проектов до личных поручений, требует систематизации и контроля, чтобы избежать стресса и повысить продуктивность.

Мобильные приложения представляют собой идеальное решение этой проблемы, предоставляя пользователю возможность всегда иметь под рукой персональный планировщик. Данный проект направлен на разработку Android-приложения «To Do List», которое предлагает простой, интуитивно понятный и в то же время функциональный инструмент для учета задач. Ключевыми особенностями приложения являются возможность определения приоритета задач и удобный механизм их удаления, что делает процесс планирования быстрым и эффективным.

Цель проекта — создать стабильное, производительное и удобное для пользователя приложение, которое поможет структурировать ежедневные задачи, фокусироваться на главном и своевременно выполнять поставленные цели.

1. Анализ предметной области

1.1 Описание предметной области

В рамках предметной области рассматривается простая система управления задачами, позволяющая пользователю:

- создавать заметки (задачи),
- указывать название и уровень приоритета,
- сохранять их в локальную базу данных,
- просматривать список всех записей,
- удалять устаревшие или ненужные задачи.

При этом ключевыми элементами предметной области являются:

- Пользователь — инициирует операции с задачами.
- Задача — структурированная единица информации, содержащая текст и приоритет.
- Список задач — совокупность всех заметок, отображаемая на главном экране.
- Хранилище данных — механизм сохранения задач.
- Операции над задачами — добавление, отображение, удаление.

Таким образом, предметная область представляет собой автоматизированную систему персонального управления задачами, ориентированную на удобство, минимализм и быстродействие.

1.2 Постановка задачи

Необходимо разработать мобильное приложение под операционную систему Android, позволяющее пользователю создавать, хранить и управлять личными задачами. Приложение должно обеспечивать простое и интуитивно понятное взаимодействие с данными, а также гарантировать сохранность информации при закрытии и повторном открытии программы.

Приложение должно предоставлять следующие функции:

1. Добавление новой задачи

   - ввод заголовка;
   - выбор уровня приоритета (например: низкий, средний, высокий);
   - сохранение задачи в локальную БД.

2. Просмотр списка задач

   - отображение всех записанных задач;

3. Удаление задачи

   - удаление по свайпу элемента списка;
   - обновление интерфейса после удаления.

4. Хранение задач

   - сохранение данных в базе;
   - доступ к данным после перезапуска приложения.

2. Проектирование приложения

2.1 Сущности предметной области

Сущность предметной области — это абстракция реального объекта, группы однотипных объектов или концептуального понятия предметной области, характеризуемая набором существенных характеристик (данных, атрибутов), связанных с проектируемой программной системой.

В рассматриваемой предметной области на данный момент можно выделить пока только одну следующую сущность:

— Задача

Это основной и самый главный объект, необходимый для реализации проектируемой базы данных. Эта сущность содержит некоторый набор атрибутов. Определим атрибуты для нее.

Задача имеет определенный идентификатор в базе данных, каждый задачу присваивается соответствующий приоритет, а также задача имеет текст.

Основываясь на описаниях предметной области и сущностей построим ER-диаграмму в нотации Чена для визуализации концептуальной модели (Рисунок 1).

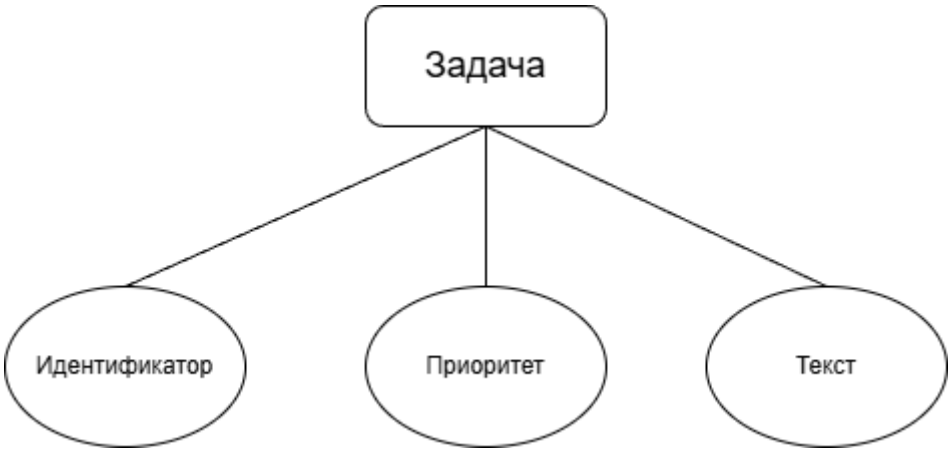

Рисунок 1 – ER-Диаграмма в нотации Чена.

2.2 Создание диаграммы классов

Диаграмма классов показывает структуру системы: классы, их атрибуты, методы и связи. Основные классы: MainActivity, AddNoteActivity, MainViewModel, AddNoteViewModel, NoteDatabase, DataBase, NotesDao, NotesAdapter, Note.

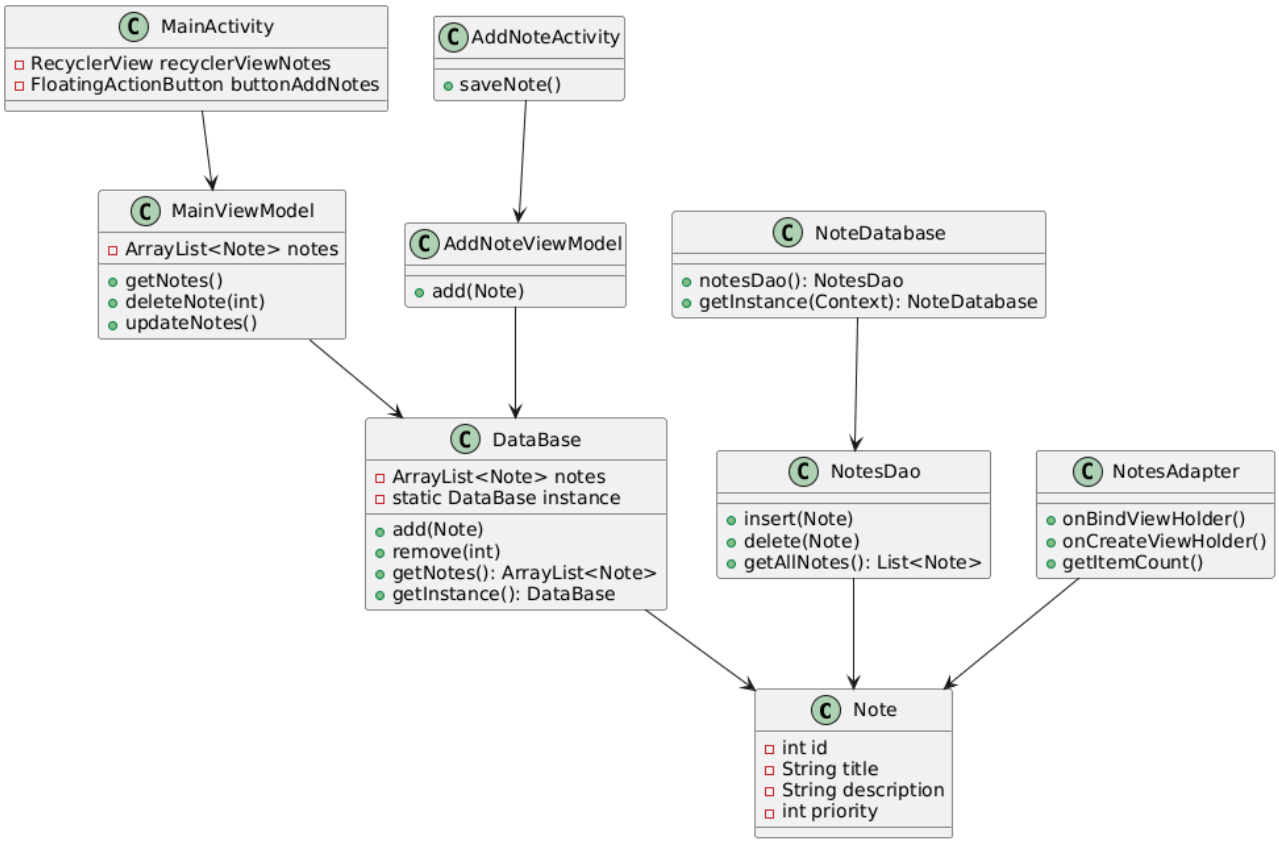

Рисунок 2 – Диаграмма классов (Class diagram)

2.3 Моделирование процессов с помощью сетей Петри

Для описания бизнес-процессов добавления и удаления заметок можно использовать сеть Петри. Основные состояния: «экран заметок», «экран добавления заметок», «заметка добавлена», «новая заметка отображена», «заметка удалена», «отображен новый экран». Переходы отображают события: добавление заметки, заполнение полей добавления заметки, удаление заметки, обновление экрана.

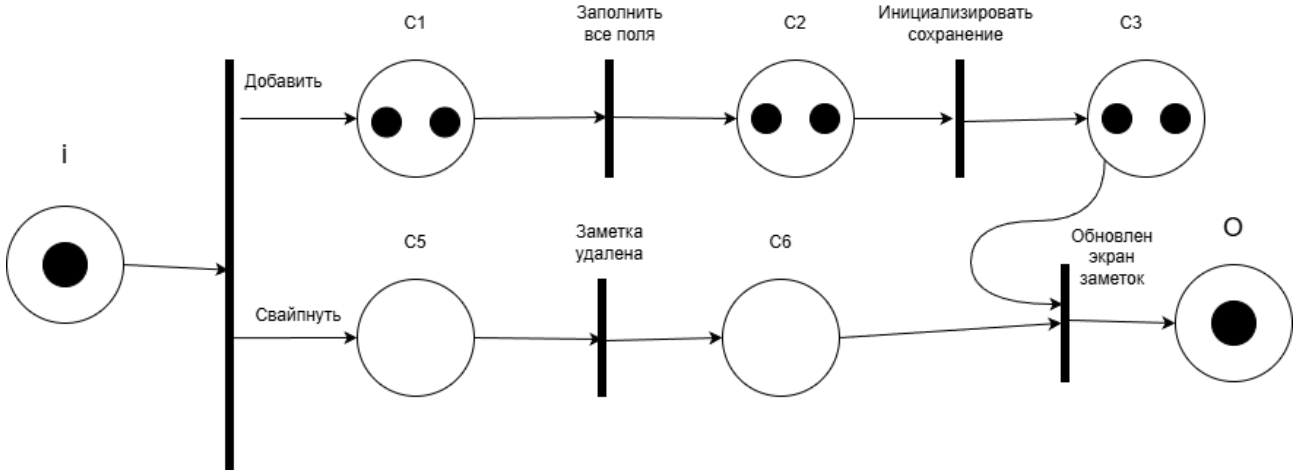

Рисунок 3.1 – Сеть Петри для процесса добавления заметок

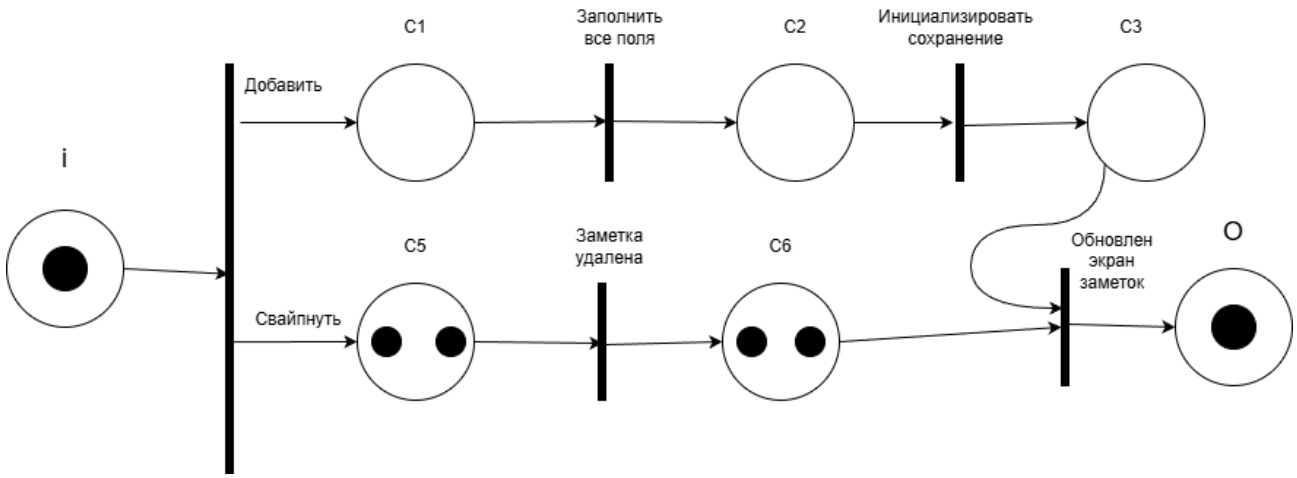

Рисунок 3.2 – Сеть Петри для процесса удаления заметок

3. Реализация и тестирование приложения

3.1 Технические требования

На данном этапе было принято решение реализовать проект, используя Android SDK. Приложение было разработано и реализовано в программе Android Studio используя язык программирования Java и принципы ООП. Приложение обладает архитектурой MVVM (Model – ViewModel – View) - это архитектурный шаблон, который разделяет логику приложения на три части: Model, View и ViewModel. Для хранения данных приложения было принято решение использовать Room Database — это официальная библиотека Google из набора Android Jetpack, предназначенная для удобной работы с локальной базой данных SQLite в Android-приложениях. Приложение также разработано при помощи асинхронное программирования - это способ выполнения кода, при котором операции, занимающие много времени (например, обращение к базе данных, сетевые запросы, задержки), выполняются в фоне, не блокируя основной поток программы. Для этого в приложении использовался инструмент RxJava — это реализация ReactiveX (Reactive Extensions) для Java и Android, которая позволяет работать с асинхронными операциями и событиями в функциональном стиле используя observable sequences.

Полный программный код Android – приложение приведен в приложении А.

3.2 Реализация приложения (принцип работы алгоритмов)

1. Алгоритм запуска приложения

Цель: загрузить данные пользователя и вывести список заметок.

Алгоритм:

1. Инициализировать компонент MainViewModel.
2. Выполнить подписку View на поток данных notesFlow (или LiveData).
3. ViewModel выполняет:
    - запрос в репозиторий → запрос в DAO → чтение всех заметок из Room Database.
4. Если база данных пуста:
    - вывести пустой список.
5. Иначе:
    - передать список заметок в View.
6. RecyclerView выводит элементы списка на экран.

2. Алгоритм добавления новой заметки

Цель: создать новую запись в базе и отобразить её в списке.

Входные данные: текст новой заметки text.

Алгоритм:

1. Пользователь нажимает кнопку **Add**.
2. Открывается экран создания заметки.
3. Пользователь вводит текст и нажимает **Save**.
4. View вызывает метод viewModel.addNote(text).
5. ViewModel выполняет:
    - создать объект Note(id = 0, text = text);
    - вызвать метод repository.insert(note) в фоновом потоке.
6. Репозиторий вызывает noteDao.insert(note).
7. Room сохраняет новую запись и автоматически обновляет поток данных, возвращая новый список заметок.
8. View получает обновлённый список.

9. RecyclerView добавляет элемент и обновляет UI.

3. Алгоритм удаления заметки (свайп-жест)

Цель: удалить выбранный элемент из базы.

Входные данные: noteId.

Алгоритм:

1. Пользователь выполняет свайп по элементу списка.
2. RecyclerView ItemTouchHelper генерирует событие удаления.
3. View вызывает метод viewModel.deleteNote(note).
4. ViewModel передаёт объект в репозиторий:
    - repository.delete(note) (в фоновом потоке).
5. Репозиторий вызывает noteDao.delete(note).
6. Room удаляет запись и обновляет поток данных.
7. View получает обновлённый список.
8. RecyclerView удаляет элемент из UI.

4. Алгоритм обновления пользовательского интерфейса

Цель: обеспечить автоматическую перерисовку экрана при изменении данных.

Алгоритм:

1. View подписывается на поток данных notesFlow или LiveData.
2. Как только данные в Room изменяются:
    - Flow/Livedata отправляет новое значение списка заметок.
3. View получает обновлённый список.
4. Вызывается метод адаптера RecyclerView:
    - submitList(newList)
5. RecyclerView сравнивает старый и новый списки (через DiffUtil).
6. Устанавливаются изменения:
    - добавление;
    - удаление;

- обновление.

7. UI обновляется минимально необходимым способом.

5. Алгоритм взаимодействия между слоями (паттерн MVVM)

Цель: обеспечить формальное разделение ответственности.

Алгоритм:

1. **View** получает событие пользователя (клик, свайп).
2. Передаёт действие во **ViewModel**.
3. ViewModel:
   - выполняет бизнес-логику;
   - обращается в репозиторий.
4. Репозиторий вызывает методы **DAO**.
5. DAO выполняет SQL-операции через **Room**.
6. Room обновляет поток данных (Flow/Livedata).
7. ViewModel получает данные и передаёт их View.
8. View обновляет экран.

6. Алгоритм навигации между экранами

Цель: обработать переходы между MainActivity и NoteActivity.

Алгоритм:

1. Пользователь нажимает кнопку добавления или выбирает заметку.
2. View формирует Intent:
   - intent.putExtra("note_id", id) для редактирования
3. Вызывает startActivity(intent).
4. NoteActivity инициализируется:
   - если передан id → получить заметку из ViewModel
   - иначе → открыть режим создания.
5. После сохранения результат передаётся в базу.
6. Activity закрывается → MainActivity автоматически обновляет список через ViewModel.

3.3 Тестирование приложения

При открытии и загрузки приложения нас встречает специальный загрузочный экран с логотипом приложения. Экран представлен на рисунке ниже.

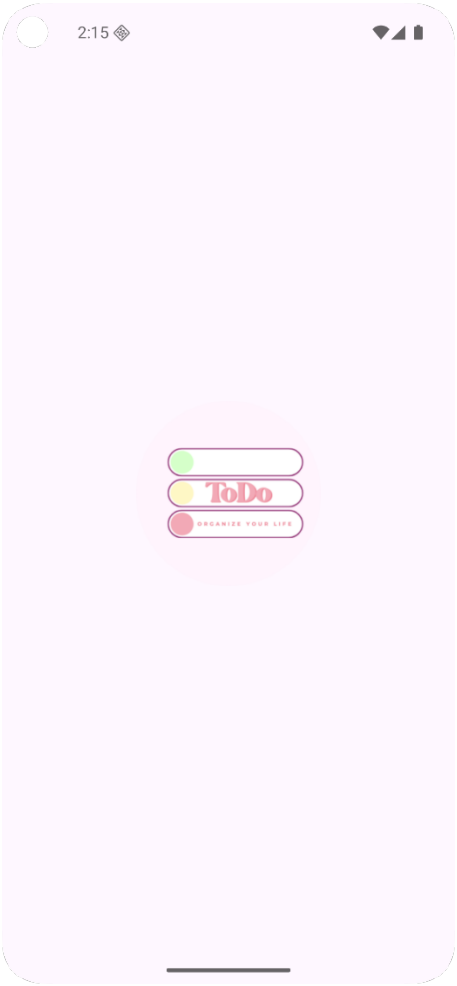

Рисунок 4 - Загрузочный экран приложения

После загрузочного экрана открывается непосредственно главный стартовый экран приложения. Экран представлен на рисунке ниже.

Рисунок 5 – Главный экран приложения

На данном экране пользователь имеет возможность добавить свою заметку, нажав на кнопку плюса в правом нижнем углу экрана. После нажатия открывается отдельный экран с добавлением заметок. Данный экран представлен на рисунке ниже.

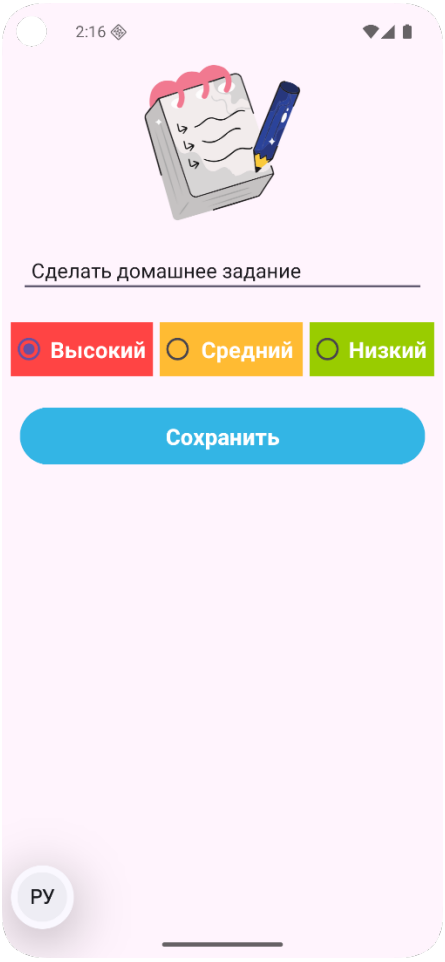

Рисунок 6 –Экран добавления заметок

Здесь пользователь непосредственно добавляет все необходимые ему заметки. Пользователь может ввести необходимый ему текст заметки (отображается подсказка о вводе текста), выбрать приоритет и сохранить ее. После нажатия кнопки «Сохранить» заметка будет добавлена в локальную базу данных приложения и отрисована на главном экране приложения со всеми заметками. Затем будет совершен автоматический переход на экран с заметками. Экран с новой добавленной заметкой предоставлен на рисунке ниже.

Рисунок 7 – Экран с добавленной заметкой

Также пользователь имеет возможность удалить созданные им заметки. Для этого используется жест свайпа (swipe), по которому заметка будет сдвинута влево и удалена с экрана и с локального хранилища приложения (базы данных). Пример работы удаления заметки указан на рисунках ниже.

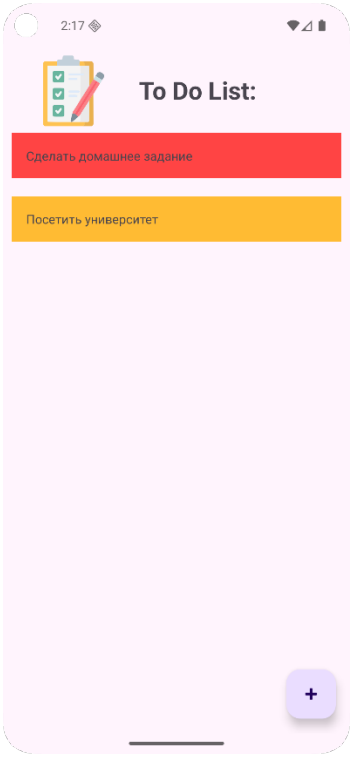

Рисунок 7 – Экран до удаления заметки

Рисунок 7 – Экран после удаления заметки

Заключение

В ходе выполнения курсовой работы была достигнута цель по разработке и реализации мобильного приложения To Do List для операционной системы Android. В процессе работы успешно решены все поставленные задачи, что позволило создать полнофункциональное и удобное приложение для управления задачами.

Основные результаты работы:

1. Проведен анализ предметной области - изучены существующие решения для управления задачами, выявлены их преимущества и недостатки, что позволило сформулировать требования к разрабатываемому приложению.
2. Разработана архитектура приложения - создана диаграмма классов, отражающая логическую структуру проекта, включая модели данных, компоненты пользовательского интерфейса и механизмы взаимодействия между ними.
3. Спроектирована база данных - реализована концептуальная схема хранения данных с использованием Room Database, обеспечивающая надежное сохранение задач и их состояний.
4. Реализовано рабочее приложение - разработан основной функционал, позволяющий создавать и удалять заметки разного приоритета.
5. Протестирована работоспособность - проведено функциональное тестирование всех компонентов приложения, подтвердившее корректность работы реализованного функционала.

Список литературы

1. Android-разработка: базовый курс — основы Java [Электронный ресурс]. Режим доступа: https://stepik.org/course/121507/syllabus (Дата обращения: 17.06.2025)

2. Начните работу с Android [Электронный ресурс]. Режим доступа: https://developer.android.com/get-started/overview?hl=ru (Дата обращения: 18.06.2025)

3. Training courses [Электронный ресурс]. Режим доступа: https://developer.android.com/courses (Дата обращения: 19.06.2025)

4. Developing android apps with java [Электронный ресурс]. Режим доступа: https://www.udacity.com/course/developing-android-apps-with-java--ud851 (Дата обращения: 20.06.2025)

Приложение А

Скрипт создания приложения

(обязательное)

AddNoteActivity.java

```
package com.likabarken.todolist;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;

public class AddNoteActivity extends AppCompatActivity {
    private EditText editTextAddNote;
    private RadioButton radioButtonHigh, radioButtonMedium, radioButtonLow;
    private Button buttonSave;

    private AddNoteViewModel viewModel;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_add_note);

        viewModel = new ViewModelProvider(this).get(AddNoteViewModel.class);
        viewModel.getShouldCloseScreen().observe(this, new Observer<Boolean>() {
            @Override
            public void onChanged(Boolean shouldClose) {
                if (shouldClose) {
                    finish();
                }
            }
        });
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        initViews();

        buttonSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                saveNote();
            }
        });
    }

    private void initViews() {
        editTextAddNote = findViewById(R.id.editTextAddNote);
        radioButtonHigh = findViewById(R.id.radioButtonHigh);
```

```
        radioButtonMedium = findViewById(R.id.radioButtonMedium);
        radioButtonLow = findViewById(R.id.radioButtonLow);
        buttonSave = findViewById(R.id.buttonSave);
    }

    private void saveNote() {
        String text = editTextAddNote.getText().toString().trim();
        int priority = getPriority();
        Note note = new Note(text, priority);
        viewModel.saveNote(note);

    }

    private int getPriority() {
        int priority;
        if (radioButtonHigh.isChecked()) {
            priority = 0;
        } else if (radioButtonMedium.isChecked()) {
            priority = 1;
        } else {
            priority = 2;
        }
        return priority;
    }

    public static Intent newIntent(Context context) {
        return new Intent(context, AddNoteActivity.class);
    }
}
```

AddNoteViewModel.java

```
package com.likabarken.todolist;

import android.app.Application;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import java.util.concurrent.TimeUnit;

import io.reactivex.rxjava3.android.schedulers.AndroidSchedulers;
import io.reactivex.rxjava3.core.Completable;
import io.reactivex.rxjava3.core.Scheduler;
import io.reactivex.rxjava3.disposables.CompositeDisposable;
import io.reactivex.rxjava3.disposables.Disposable;
import io.reactivex.rxjava3.functions.Action;
import io.reactivex.rxjava3.functions.Consumer;
import io.reactivex.rxjava3.schedulers.Schedulers;

public class AddNoteViewModel extends AndroidViewModel {

    private NotesDao notesDao;

    // viewModel и Activity могут общаться при помощи объекта LiveData
    private MutableLiveData<Boolean> shouldCloseScreen = new MutableLiveData<>();
    //Disposable позволяет управлять жизненным циклом подписок
    private CompositeDisposable compositeDisposable = new CompositeDisposable();
    //private Disposable disposable;


    public AddNoteViewModel(@NonNull Application application) {
        super(application);
        notesDao = NoteDatabase.getInstance(application).notesDao();
    }

    public LiveData<Boolean> getShouldCloseScreen() {
```

```
        return shouldCloseScreen;
    }

    public void saveNote(Note note) {
        // Чтобы метод add выполнился - на него надо подписаться
        // Все объекты в RxJava используют механизм callback-ов(н-р, слушатель клика или свайпа)
        Disposable disposable = notesDao.add(note)
                .subscribeOn(Schedulers.io()) // добавление в базу в фоновом потоке, аргумент - поток
                .observeOn(AndroidSchedulers.mainThread()) // переключить поток на главный поток
                .subscribe(new Action() {
                    @Override
                    public void run() throws Throwable {
                        Log.d("AddNoteViewModel", "subscribe");
                        shouldCloseScreen.setValue(true);
                    }
                }, new Consumer<Throwable>() {
                    @Override
                    public void accept(Throwable throwable) throws Throwable {
                        Log.d("AddNoteViewModel", "Error during saving a note.");
                    }
                });

        compositeDisposable.add(disposable);
//        // setValue можно вызывать только на главном потоке
    }

    // В момент уничтожения ViewModel - у нее вызывается метод onCleared()

    @Override
    protected void onCleared() {
        super.onCleared();
        compositeDisposable.dispose();
    }
}
```

DataBase.java

```
package com.likabarken.todolist;

import android.provider.ContactsContract;

import java.util.ArrayList;
import java.util.Random;

public class DataBase {
    private ArrayList<Note> notes = new ArrayList<>();

    private static DataBase instance = null;

    public static DataBase getInstance() {
        if (instance == null) {
            instance = new DataBase();
        }
        return instance;
    }

    private DataBase() {
        Random random = new Random();
        for (int i = 0; i < 20; i++) {
            Note note = new Note(i, "Note " + i, random.nextInt(3));
            notes.add(note);
        }
    }

    public void add(Note note) {
        notes.add(note);
    }

    public void remove(int id) {
        for (int i = 0; i < notes.size(); i++) {
```

```
            Note note = notes.get(i);
            if (note.getId() == id) {
                notes.remove(note);
            }
        }
    }

    public ArrayList<Note> getNotes() {
        return new ArrayList<>(notes);
    }
}
```

MainActivity.java

```
package com.likabarken.todolist;

import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

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

public class MainActivity extends AppCompatActivity {
    private RecyclerView recycleViewNotes;
    private FloatingActionButton buttonAddNote;
    private NotesAdapter notesAdapter;

    private MainViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        // viewModel занимается только отображением данных и взаимодействием с пользователем
        // viewModel живет дольше, чем активити и умеет переживать переворот экрана
        viewModel = new ViewModelProvider(this).get(MainViewModel.class);

//        viewModel.getCount().observe(this, new Observer<Integer>() {
//            @Override
//            public void onChanged(Integer count) {
//                Toast.makeText(
//                        MainActivity.this,
```

```
//                    String.valueOf(count),
//                    Toast.LENGTH_SHORT
//            ).show();
//        }
//    });

        initViews();
        notesAdapter = new NotesAdapter();

//      notesAdapter.setOnNoteClickListener(new NotesAdapter.OnNoteClickListener() {
//          @Override
//          public void onNoteClick(Note note) {
//              viewModel.showCount();
//              //int count  = viewModel.getCount();
//          }
//      });

        recycleViewNotes.setAdapter(notesAdapter);

        // Подписка на все изменения, которые произойдут в базе данных.
        // При наличии изменений, новые данные прилетят в метод onChanged.
        viewModel.getNotes().observe(this, new Observer<List<Note>>() {
            @Override
            public void onChanged(List<Note> notes) {
                notesAdapter.setNotes(notes);
            }
        });

        ItemTouchHelper itemTouchHelper = new ItemTouchHelper(
                new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT) {
                    @Override
                    public boolean onMove(
                            @NonNull RecyclerView recyclerView,
                            @NonNull RecyclerView.ViewHolder viewHolder,
                            @NonNull RecyclerView.ViewHolder target) {
                        return false;
                    }

                    @Override
                    public void onSwiped(
                            @NonNull RecyclerView.ViewHolder viewHolder,
                            int direction) {
                        int position = viewHolder.getAdapterPosition();
                        Note note = notesAdapter.getNotes().get(position);

                        viewModel.remove(note);

                    }
                });

        itemTouchHelper.attachToRecyclerView(recycleViewNotes);

        buttonAddNote.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(AddNoteActivity.newIntent(MainActivity.this));
            }
        });

    }

    private void initViews() {
        recycleViewNotes = findViewById(R.id.recycleViewNotes);
        buttonAddNote = findViewById(R.id.buttonAddNote);
    }
}
```

MainViewModel.java

```
package com.likabarken.todolist;

import android.app.Application;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import java.util.Collections;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.TimeUnit;

import io.reactivex.rxjava3.android.schedulers.AndroidSchedulers;
import io.reactivex.rxjava3.core.Completable;
import io.reactivex.rxjava3.core.Single;
import io.reactivex.rxjava3.disposables.CompositeDisposable;
import io.reactivex.rxjava3.disposables.Disposable;
import io.reactivex.rxjava3.functions.Action;
import io.reactivex.rxjava3.functions.BiConsumer;
import io.reactivex.rxjava3.functions.Consumer;
import io.reactivex.rxjava3.schedulers.Schedulers;

public class MainViewModel extends AndroidViewModel {

    private NoteDatabase noteDatabase;
    private CompositeDisposable compositeDisposable = new CompositeDisposable();

//    private int count = 0;
//    private MutableLiveData<Integer> countLD = new MutableLiveData<>();

    public MainViewModel(@NonNull Application application) {
        super(application);
        noteDatabase = NoteDatabase.getInstance(application);
    }

    public LiveData<List<Note>> getNotes() {
        return noteDatabase.notesDao().getNotes();
    }

    public void remove (Note note) {
        Disposable disposable = noteDatabase.notesDao().remove(note.getId())
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Action() {
                    @Override
                    public void run() throws Throwable {
                        Log.d("MainViewModel", "Removed: " + note.getId());
                    }
                }, new Consumer<Throwable>() {
                    @Override
                    public void accept(Throwable throwable) throws Throwable {
                        Log.d("MainViewModel", "Error during removing a note.");
                    }
                });

        compositeDisposable.add(disposable);
    }

    @Override
    protected void onCleared() {
        super.onCleared();
        compositeDisposable.dispose();
    }
}
```

Note.java

```
package com.likabarken.todolist;

import androidx.room.Entity;
import androidx.room.Ignore;
import androidx.room.PrimaryKey;

@Entity(tableName = "notes")
public class Note {

    @PrimaryKey(autoGenerate = true)
    private int id;
    private String text;
    private int priority;


    public Note(int id, String text, int priority) {
        this.id = id;
        this.text = text;
        this.priority = priority;
    }

    @Ignore
    public Note(String text, int priority) {
        this(0, text, priority);
    }

    public int getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public int getPriority() {
        return priority;
    }
}
```

NoteDatabase.java

```
package com.likabarken.todolist;

import android.app.Application;

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

// Номер версии необходимо поднимать, если произошли какие-то изменения в базе
// Нр, добавилась новая таблица, изменилось поле в существующей таблице
@Database(entities = {Note.class}, version = 1)
public abstract class NoteDatabase extends RoomDatabase {

    private static NoteDatabase instance = null;
    private static final String DB_NAME = "notes.db";

    public static NoteDatabase getInstance(Application application){
        if (instance == null) {
            instance = Room.databaseBuilder(
                    application,
                    NoteDatabase.class,
                    DB_NAME
            ).build();
        }
        return instance;
    }
```

```
    public abstract NotesDao notesDao();
}
```

NotesAdapter.java

```
package com.likabarken.todolist;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.RecyclerView;

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

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

    // Здесь можно в переменной с типом List использовать конструктор с типом ArrayList,
    // поскольку ArrayList реализует интерфейс List
    private List<Note> notes = new ArrayList<>();

    // Из активити в переменную onNoteClickListener будем устанавливать новое значение,
    // будем передавать реализацию интерфейса onNoteClickListener,
    // а в адаптере будем с ней работать
    private OnNoteClickListener onNoteClickListener;

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

    public List<Note> getNotes() {
        // return notes;
        // Если нужно вернуть коллекцию, то возвращать нужно копию коллекции, чтобы снаружи не было
        // возможности изменять этот объект
        return new ArrayList<>(notes);
    }

    public void setNotes(List<Note> notes) {
        this.notes = notes;
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public NotesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // В этом методе показывается, как создается view из макета

        View view = LayoutInflater.from(parent.getContext()).inflate(
                R.layout.note_item,
                parent,
                false
        );
        return new NotesViewHolder(view);
    }

    @Override
    public void onBindViewHolder(NotesViewHolder viewHolder, int position) {
        // Этот метод отвечает за то, какой цвет, какой текст будет установлен во view элементы
        // По номеру позиции position можно узнать, какой элемент нужно установить

        Note note = notes.get(position); // Заметка, которую будем отображать
        viewHolder.textViewNote.setText(note.getText());

        int colorResId;
```

```
        switch (note.getPriority()) {
            case 0:
                colorResId = R.color.red_light;
                break;
            case 1:
                colorResId = R.color.orange_light;
                break;
            default:
                colorResId = R.color.green_light;
        }

        int color = ContextCompat.getColor(viewHolder.itemView.getContext(), colorResId);
        viewHolder.textViewNote.setBackgroundColor(color);

        viewHolder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (onNoteClickListener != null){
                    onNoteClickListener.onNoteClick(note);
                }

            }
        });

    }

    @Override
    public int getItemCount() {
        return notes.size();
    }

    class NotesViewHolder extends RecyclerView.ViewHolder {
        private TextView textViewNote;

        public NotesViewHolder(@NonNull View itemView) {
            super(itemView);
            textViewNote =itemView.findViewById(R.id.textViewNote);
        }
    }

    interface OnNoteClickListener{

        void onNoteClick(Note note);
    }
}
```

NotesDao.java

```
package com.likabarken.todolist;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.OnConflictStrategy;
import androidx.room.Query;

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

import io.reactivex.rxjava3.core.Completable;
import io.reactivex.rxjava3.core.Single;

// DAO - Data Access Objects
@Dao
public interface NotesDao {

    @Query("SELECT * FROM notes")
    LiveData<List<Note>> getNotes();
    // Метод, который возвращает коллекцию данных в Dao должен иметь интерфейсный тип List.
```

```
    // Здесь нельзя указывать конкретную реализацию.

    @Insert(onConflict = OnConflictStrategy.ABORT)
    // Completable - класс из библиотеки RxJava3, на этот объект можно подписываться
    Completable add(Note note);

    @Query("DELETE FROM notes WHERE id = :id")
    Completable remove(int id);

}
```

activity_add_note.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/light_pink"
    tools:context=".AddNoteActivity">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:contentDescription="@string/notebook"
        android:src="@drawable/note_book" />

    <EditText
        android:id="@+id/editTextAddNote"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:autofillHints="Enter note"
        android:hint="@string/edit_text_rus"
        android:inputType="textPersonName"
        android:padding="10dp"
        android:layout_margin="16dp" />

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:gravity="center"
        android:orientation="horizontal">


        <RadioButton
            android:id="@+id/radioButtonHigh"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:padding="3dp"
            android:layout_margin="3dp"
            android:text="@string/high_rus"
            android:checked="true"
            android:textColor="@color/white"
            android:textStyle="bold"
            android:textSize="20sp"
            android:background="@android:color/holo_red_light"
            style="@style/Widget.AppCompat.CompoundButton.RadioButton"/>

        <RadioButton
            android:id="@+id/radioButtonMedium"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:padding="5dp"
```

```
        android:layout_margin="3dp"
        android:text="@string/medium_rus"
        android:textColor="@color/white"
        android:textStyle="bold"
        android:textSize="20sp"
        android:background="@android:color/holo_orange_light"
        style="@style/Widget.AppCompat.CompoundButton.RadioButton"/>

    <RadioButton
        android:id="@+id/radioButtonLow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:padding="3dp"
        android:layout_margin="3dp"
        android:text="@string/low_rus"
        android:textColor="@color/white"
        android:textStyle="bold"
        android:textSize="20sp"
        android:background="@android:color/holo_green_light"
        style="@style/Widget.AppCompat.CompoundButton.RadioButton"/>

  </RadioGroup>

  <Button
    android:id="@+id/buttonSave"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:padding="12dp"
    android:backgroundTint="@android:color/holo_blue_light"
    android:text="@string/save_rus"
    android:textStyle="bold"
    android:textSize="20sp"/>

</LinearLayout>
```

activity_main.xml

```
<?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:id="@+id/main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@color/light_pink"
  tools:context=".MainActivity">

  <ImageView
    android:id="@+id/imageViewChecklist"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_marginTop="10dp"
    android:layout_marginStart="40dp"
    android:contentDescription="@string/to_do_picture"
    android:src="@drawable/checklist_2666505"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

  <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:padding="20dp"
    android:text="@string/to_do_list"
    android:textSize="28sp"
    android:textStyle="bold"
```

```
        app:layout_constraintEnd_toStartOf="@+id/buttonAddNote"
        app:layout_constraintStart_toEndOf="@+id/imageViewChecklist"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycleViewNotes"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintBottom_toBottomOf="parent"/>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/buttonAddNote"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:contentDescription="@string/add_todo_item"
        android:soundEffectsEnabled="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_input_add" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

note_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/textViewNote"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:layout_margin="10dp"
    tools:text="Note"
    tools:background="@color/material_dynamic_neutral70">

</TextView>
```