ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

ассистент		И.Д. Свеженин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Нативный пользовательский интерфейс

по курсу: Кроссплатформенное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1. Цель работы:

Выполнить проектирование и разработку мобильного приложения под ОС Android на языке программирования высокого уровня Kotlin.

2. Задание:

1. Создайте приложение для управления списком задач в соответствии с вариантом (предметной областью) – для определенного класса. Пользователь должен иметь возможность добавлять, удалять и отмечать задачи как выполненные. Используйте RecyclerView для отображения списка задач.

2. Разработайте галерею изображений (в соответствии с вариантом (предметной областью) – для определенного класса), где пользователь может просматривать изображения, а также добавлять новые изображения из галереи устройства. Используйте RecyclerView для отображения изображений в виде сетки.

Вариант 24

Предметная область - Попугаи.

3. Краткое описание хода разработки, алгоритма работы программы и назначение используемых технологий

Ход разработки:

1. Разработать главную активность – «список задач»

2. Разработать класс данных для «задачи»

3. Разработать класс-адаптер для «задач» – логика

4. Разработать активность – «галерею» и адаптер к ней

5. Разработать активность – «просмотр изображения из галереи»

Используемые технологии:

- Kotlin – используется для работы написанного кода.
- AppCompat – визуальная часть, «тема» приложения.

Описание структуры работы частей приложения:

1. Основная часть — MainActivity — точка входа приложения

MainActivity является стартовым экраном и обеспечивает управление списком задач.

a. Реакция приложения на событие onCreate()

1. Загрузка основной разметки интерфейса (activity_main.xml).
2. Инициализация элементов UI:
   - поле ввода новой задачи,
   - кнопка «Добавить»,
   - кнопка «Галерея попугаев»,
   - RecyclerView со списком задач.
3. Создание и привязка адаптера TaskAdapter к RecyclerView.
4. Настройка обработчиков событий:
   - добавление новой задачи,
   - отметка задачи как выполненной,
   - удаление задачи,
   - переход к галерее изображений (переход к GalleryActivity).

2. Визуальная часть — интерфейс списка задач

a. Используемые переменные

- список задач (MutableList<Task>),
- адаптер для списка (TaskAdapter),
- счётчик идентификаторов задач.

b. Вертикальный контейнер (LinearLayout) содержит:

i. Поле ввода текста задачи
Позволяет пользователю вписывать название новой задачи.

ii. Кнопку «Добавить»
При нажатии создаётся объект Task и добавляется в RecyclerView.

iii. Кнопку «Галерея попугаев»
Открывает экран GalleryActivity через Intent.

iv. RecyclerView (список задач)
Отображает список в вертикальном формате с помощью LinearLayoutManager.

c. Элемент списка задачи (item_task.xml) содержит:

- чекбокс выполнения,
- текст задачи,
- кнопку удаления.

3. Адаптер списка задач — TaskAdapter

a. Визуальная часть элемента списка

Адаптер связывает данные Task с разметкой item_task.xml.

b. Обработка событий

- изменение состояния чекбокса → обновление статуса выполнения задачи;
- нажатие кнопки удаления → удаление элемента из списка;
- уведомление RecyclerView об изменении данных (notifyItemChanged, notifyItemRemoved).

4. Модуль галереи — GalleryActivity

GalleryActivity реализует работу с коллекцией изображений.

a. Реакция onCreate()

- загрузка разметки activity_gallery.xml;
- инициализация кнопки добавления изображения;
- инициализация RecyclerView с GridLayoutManager (3 столбца);
- создание адаптера GalleryAdapter;
- установка обработчика клика для добавления изображения через системную галерею;
- установка обработчика клика на изображение — переход к PhotoActivity.

b. Элемент галереи (item_image.xml)

- прямоугольное ImageView, отображающее уменьшенную копию изображения.

5. Адаптер галереи — GalleryAdapter

a. Визуальная часть

Отрисовывает изображения в сетке из ImageView.

b. Обработка нажатий

При клике на миниатюру вызывается переданная функция onClick, которая запускает PhotoActivity и передаёт выбранный URI.

6. Экран просмотра изображения — PhotoActivity

a. Реакция onCreate()

- получение URI изображения из Intent;
- загрузка разметки activity_photo.xml;
- отображение изображения в полноэкранном ImageView.

b. Визуальная часть

Полноэкранный ImageView, настроенный на масштабирование без обрезки (scaleType="fitCenter").

7. Логическая часть программы

a. Управление списком задач

- генерация уникальных идентификаторов,
- хранение данных в оперативной памяти,
- динамическое обновление RecyclerView.

b. Работа с изображениями

- получение файлов через системное диалоговое окно,
- работа с URI,
- передача данных между Activity.

8. Функция демонстрации приложения в IDE

- запуск виртуального устройства (Android Emulator);
- развертывание APK в эмуляторе;
- тестирование работы списков, галереи и переходов между Activity;
- просмотр логов в Logcat для анализа ошибок.

4. Скриншоты, иллюстрирующие результаты работы программы:

В качестве демонстрации работы программы приведём несколько скриншотов с разными состояниями каждого экрана (активности).

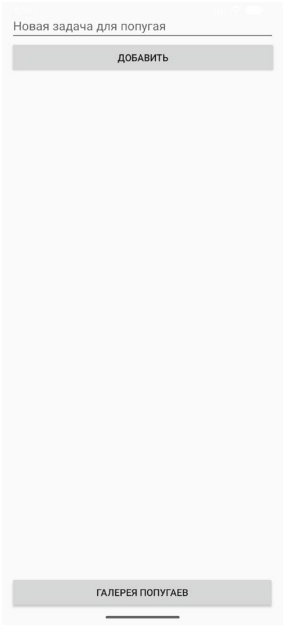

Рисунок 2.1 – Стартовый экран (список задач)

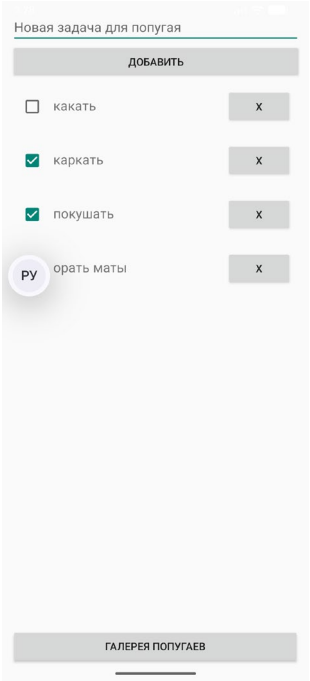

Рисунок 2.2 – Список задач заполненный

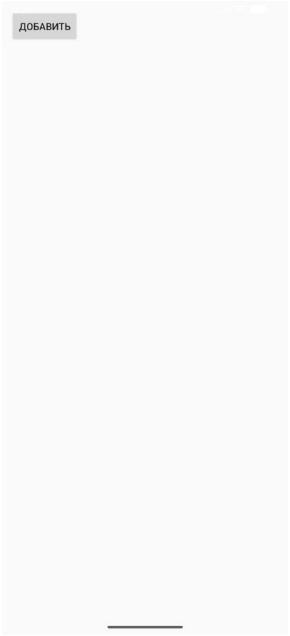

Рисунок 3.1 – Пустая галерея

Рисунок 3.2 – Заполненная галерея

Рисунок 3.3 – Просмотр изображения

5. Вывод:

В ходе выполненной работы были спроектированы и разработаны два взаимосвязанных функциональных модуля мобильного приложения под операционную систему Android на языке высокого уровня Kotlin. Проектирование интерфейса осуществлялось с использованием классической XML-разметки и стандартных компонентов Android SDK.

В результате разработки были созданы визуальная и логическая части приложения, обеспечивающие:

управление списком задач (добавление, удаление и отметка выполнения);

отображение изображений в виде сетки с возможностью добавления новых файлов из галереи устройства;

открытие выбранного изображения на отдельном экране в полноэкранном режиме.

Для эффективного отображения списков и коллекций данных были применены компоненты RecyclerView и соответствующие адаптеры, обеспечивающие связь между моделью данных и пользовательским интерфейсом. Навигация между экранами реализована посредством механизма Intent, что позволило передавать данные между Activity и формировать многомодульную структуру приложения.

Разработанное приложение соответствует поставленным требованиям, демонстрирует принципы модульности, обработки пользовательских событий, работы со списочными структурами и взаимодействия с системными ресурсами Android.

ПРИЛОЖЕНИЕ А

Листинг программы:

MainActivity.kt:

```
package com.example.papoogi

import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.papoogi.ui.theme.PapoogiTheme
import android.widget.EditText
import android.widget.Button
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {

    private val tasks = mutableListOf<Task>()
    private lateinit var adapter: TaskAdapter
    private var nextId = 1


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val editTask = findViewById<EditText>(R.id.editTask)
        val btnAdd = findViewById<Button>(R.id.btnAdd)
        val recycler = findViewById<RecyclerView>(R.id.recyclerTasks)

        val btnGallery = findViewById<Button>(R.id.btnGallery)

        btnGallery.setOnClickListener {
            val intent = Intent(this, GalleryActivity::class.java)
            startActivity(intent)
        }


        adapter = TaskAdapter(
            tasks,
            onDelete = { deleteTask(it) },
            onToggle = { toggleTask(it) }
        )

        recycler.layoutManager = LinearLayoutManager(this)
        recycler.adapter = adapter

        btnAdd.setOnClickListener {
            val text = editTask.text.toString()
            if (text.isNotEmpty()) {
                val t = Task(nextId++, text)
                tasks.add(t)
```

```
                adapter.notifyItemInserted(tasks.size - 1)
                editTask.text.clear()
            }
        }
    }

    private fun deleteTask(t: Task) {
        val index = tasks.indexOf(t)
        tasks.remove(t)
        adapter.notifyItemRemoved(index)
    }

    private fun toggleTask(t: Task) {
        t.isDone = !t.isDone
        adapter.notifyItemChanged(tasks.indexOf(t))
    }

}
```