ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		Н. И. Чулочникова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 4

СПИСОК ЗАДАЧ И ГАЛЛЕРЕЯ

по курсу:

КРОССПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1 Цель работы

Цель работы: создание мобильных приложений на Kotlin для ОС Android, включающих список задач для мотоциклистов с возможностью редактирования, а также галерею изображений; отображать элементы через RecyclerView.

2 Задание

Работа выполнялась по варианту № 17.

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.

3 Листинг программ

Листинг программного кода приложений представлен в Приложении А.

4 Результаты работы программ

На рисунках 1–4 изображены результаты сборки и запуска программ.

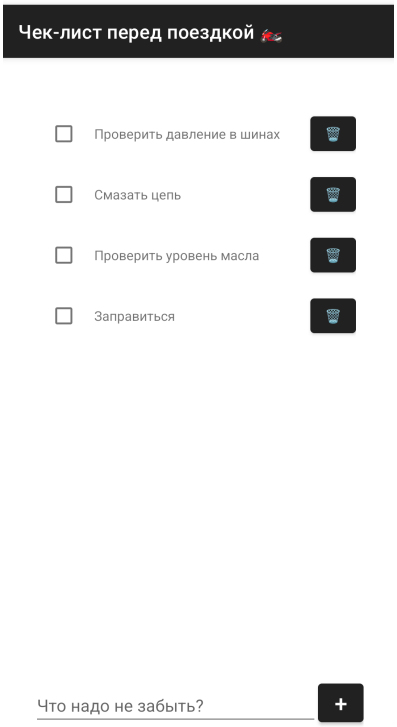

Рисунок 1 — Список задач. Изначальный вид

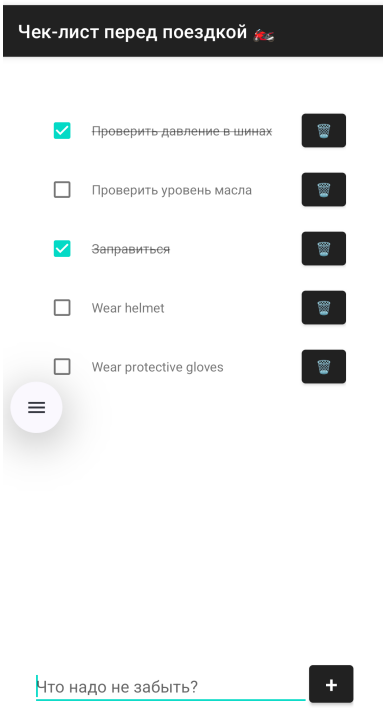

Рисунок 2 — Список задач. Список изменен (добавление, удаление, выполнение задач)

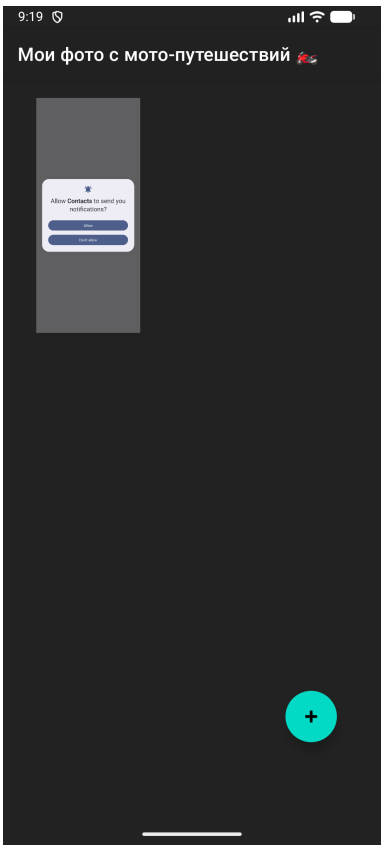

Рисунок 3 — Галерея. Добавлено одно изображение

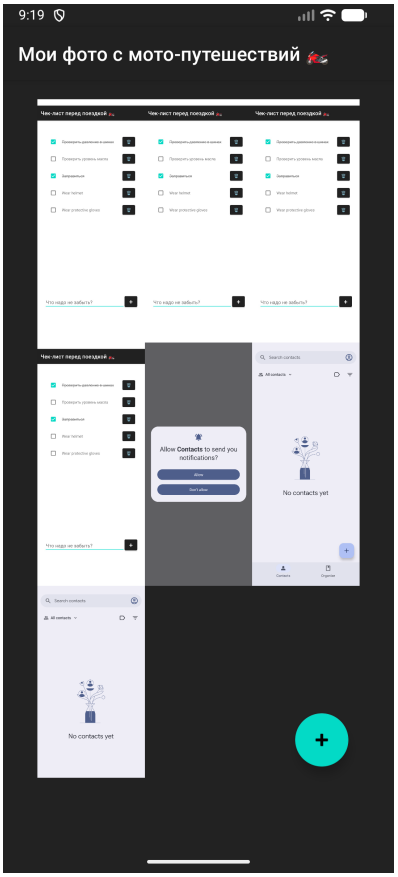

Рисунок 4 — Галерея. Добавлено несколько изображений

5 Выводы

В первой части работы было создано приложение для управления списком задач для мотоциклистов. Список задач отображается динамически, и при добавлении, удалении или отметке задач как выполненных интерфейс обновляется автоматически. Для организации и отображения списка используется подход, при котором элементы данных связываются с элементами интерфейса через адаптер, что обеспечивает эффективное управление и обновление содержимого.

Во второй части работы реализована галерея изображений. Фотографии мотоциклов выводятся в виде сетки, а пользователь может добавлять новые изображения из памяти устройства. Список изображений также управляется через адаптер, что позволяет динамически обновлять интерфейс при добавлении новых элементов и обеспечивать плавное взаимодействие с пользователем.

Обе части работы продемонстрировали использование адаптеров для связывания данных и интерфейса, работу с динамическими списками и коллекциями, а также умение создавать удобные и интерактивные мобильные приложения с возможностью редактирования и обновления содержимого.

ПРИЛОЖЕНИЕ А

Листинг 1 — Задание 1. MainActivity.kt

```
package com.grigorijtomczuk.todoer

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {

    private lateinit var tasksRecyclerView: RecyclerView
    private lateinit var newTaskEditText: EditText
    private lateinit var addTaskButton: Button
    private lateinit var tasksAdapter: TasksAdapter

    // Список задач
    private var tasks = mutableListOf(
        Task(name = "Проверить давление в шинах"),
        Task(name = "Смазать цепь"),
        Task(name = "Проверить уровень масла"),
        Task(name = "Заправиться")
    )

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

        tasksRecyclerView = findViewById(R.id.tasksRecyclerView)
        newTaskEditText = findViewById(R.id.newTaskEditText)
        addTaskButton = findViewById(R.id.addTaskButton)

        tasksAdapter = TasksAdapter(
            onTaskCompleted = { clickedTask ->
                tasks = tasks.map { task ->
                    if (task.id == clickedTask.id) {
                        task.copy(isCompleted = !task.isCompleted)
                    } else {
                        task
                    }
                }.toMutableList()
                // Передаем в адаптер новую, неизменяемую копию списка
                tasksAdapter.submitList(tasks.toList())
            },
            onDeleteTask = { taskToDelete ->
                tasks = tasks.filter { it.id != taskToDelete.id }.toMutableList()
                // Передаем в адаптер новую, неизменяемую копию списка
                tasksAdapter.submitList(tasks.toList())
            }
        )

        tasksRecyclerView.adapter = tasksAdapter
        tasksRecyclerView.layoutManager = LinearLayoutManager(this)

        tasksAdapter.submitList(tasks.toList())

        addTaskButton.setOnClickListener {
```

```
            val taskName = newTaskEditText.text.toString()
            if (taskName.isNotBlank()) {
                // Создаем новый список, добавляя в него новую задачу
                tasks.add(Task(name = taskName))
                // Передаем в адаптер новую, неизменяемую копию списка
                tasksAdapter.submitList(tasks.toList())
                newTaskEditText.text.clear()
            }
        }
    }
}
```

Листинг 2 — Задание 1. Task.kt

```
package com.grigorijtomczuk.todoer

import java.util.UUID

data class Task(
    val id: String = UUID.randomUUID().toString(),
    val name: String,
    val isCompleted: Boolean = false
)
```

Листинг 3 — Задание 1. TaskAdapter.kt

```
package com.grigorijtomczuk.todoer

import android.graphics.Paint
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.CheckBox
import android.widget.TextView
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView

class TasksAdapter(
    private val onTaskCompleted: (Task) -> Unit,
    private val onDeleteTask: (Task) -> Unit
) : ListAdapter<Task, TasksAdapter.TaskViewHolder>(TaskDiffCallback()) {

    // ViewHolder хранит ссылки на View элементы для каждого элемента списка
    class TaskViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val taskNameTextView: TextView = itemView.findViewById(R.id.taskNameTextView)
        val taskCheckBox: CheckBox = itemView.findViewById(R.id.taskCheckBox)
        val deleteTaskButton: Button = itemView.findViewById(R.id.deleteTaskButton)

        fun bind(task: Task, onTaskCompleted: (Task) -> Unit, onDeleteTask: (Task) ->
Unit) {
            taskNameTextView.text = task.name
            taskCheckBox.isChecked = task.isCompleted
            toggleStrikeThrough(taskNameTextView, task.isCompleted)

            taskCheckBox.setOnClickListener {
                onTaskCompleted(task)
            }
            deleteTaskButton.setOnClickListener {
                onDeleteTask(task)
            }
```

```
        }

        private fun toggleStrikeThrough(textView: TextView, isChecked: Boolean) {
            if (isChecked) {
                textView.paintFlags = textView.paintFlags or
Paint.STRIKE_THRU_TEXT_FLAG
            } else {
                textView.paintFlags = textView.paintFlags and
Paint.STRIKE_THRU_TEXT_FLAG.inv()
            }
        }
    }

    // Вызывается, когда RecyclerView требует новый ViewHolder для отображения
элемента
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): TaskViewHolder
{
        val itemView = LayoutInflater.from(parent.context)
            .inflate(R.layout.task_item, parent, false)
        return TaskViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: TaskViewHolder, position: Int) {
        val task = getItem(position)
        holder.bind(task, onTaskCompleted, onDeleteTask)
    }
}

class TaskDiffCallback : DiffUtil.ItemCallback<Task>() {
    override fun areItemsTheSame(oldItem: Task, newItem: Task): Boolean {
        return oldItem.id == newItem.id
    }

    override fun areContentsTheSame(oldItem: Task, newItem: Task): Boolean {
        return oldItem == newItem
    }
}
```

Листинг 4 — Задание 2. MainActivity.kt

```
package com.grigorijtomczuk.mygallery

import android.net.Uri
import android.os.Bundle
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var fab: FloatingActionButton
    private val imageUris = mutableListOf<Uri>()
    private lateinit var imageAdapter: ImageAdapter

    // ActivityResultLauncher для получения изображения из галереи
    private val selectImageLauncher =
        registerForActivityResult(ActivityResultContracts.GetContent()) { uri: Uri? -
>
            // Изображение выбрано
```

```
            uri?.let {
                imageUris.add(it)
                imageAdapter.notifyDataSetChanged()
            }
        }

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

        // Инициализируем RecyclerView и FloatingActionButton из макета
        recyclerView = findViewById(R.id.recyclerView)
        fab = findViewById(R.id.fab)

        // инициализируем адаптер
        imageAdapter = ImageAdapter(imageUris)
        recyclerView.layoutManager =
            GridLayoutManager(this, 3) // GridLayoutManager для отображения в 3
колонки
        recyclerView.adapter = imageAdapter

        fab.setOnClickListener {
            // Запускаем галерею для выбора изображения
            selectImageLauncher.launch("image/*")
        }
    }
}
```

Листинг 5 — Задание 2. ImageAdapter.kt

```
package com.grigorijtomczuk.mygallery

import android.net.Uri
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import androidx.recyclerview.widget.RecyclerView

class ImageAdapter(private val imageUris: MutableList<Uri>) :
    RecyclerView.Adapter<ImageAdapter.ImageViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ImageViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_image,
parent, false)
        return ImageViewHolder(view)
    }

    // связываем URI изображения с ViewHolder
    override fun onBindViewHolder(holder: ImageViewHolder, position: Int) {
        val imageUri = imageUris[position]
        holder.imageView.setImageURI(imageUri)
    }

    override fun getItemCount(): Int = imageUris.size

    class ImageViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val imageView: ImageView = itemView.findViewById(R.id.imageView)
    }
}
```