ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 6

РАБОТА С БД. JSON

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 мобильного приложения на Kotlin для ОС Android с использованием базы данных и JSON.

2 Задание

Вариант работы: № 17 (мотоциклисты).

1. Задание «БД»
    1. Создать БД в соответствии с вариантом (предметной областью) – для определенного класса;
    2. Реализовать добавление данных в БД с помощью отдельного Activity;
    3. Вывести данные из БД в RecyclerView. Также использовать фрагменты.
    4. Обработка долгого нажатия на элемент списка;
    5. Создание диалогового окна для выбора «Просмотр», «Удаление», «Обновление»;
    6. Обработка нажатия элементов диалогового окна, например, для подтверждения удаления или обновления;
    7. Реализовать обновление данных в БД с помощью отдельного Activity;
    8. Реализовать удаление данных из БД с помощью отдельного Activity;
    9. При работе с БД использовать библиотеку Room.
2. Задание «JSON»
    1. Скачать JSON из интернета (HttpURLConnection/Retrofit);
    2. Распарсить JSON;
    3. Использовать Thread для работы с JSON.

3 Листинг программы

Листинг основного кода приложения представлен в Приложении А.

4 Результаты работы программы

На рисунках 1–8 изображены результаты тестирования программы.

Рисунок 1 — Экран БД «Мотоциклы»

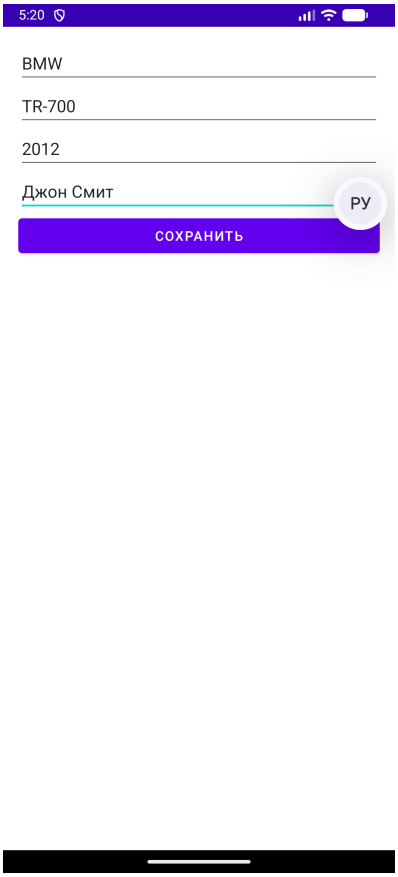

Рисунок 2 — Добавление нового мотоцикла

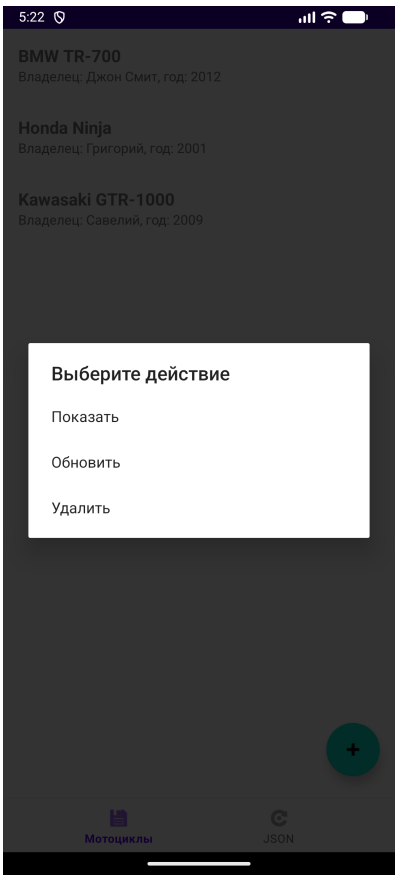

Рисунок 3 — Диалоговое окно при долгом нажатии на запись

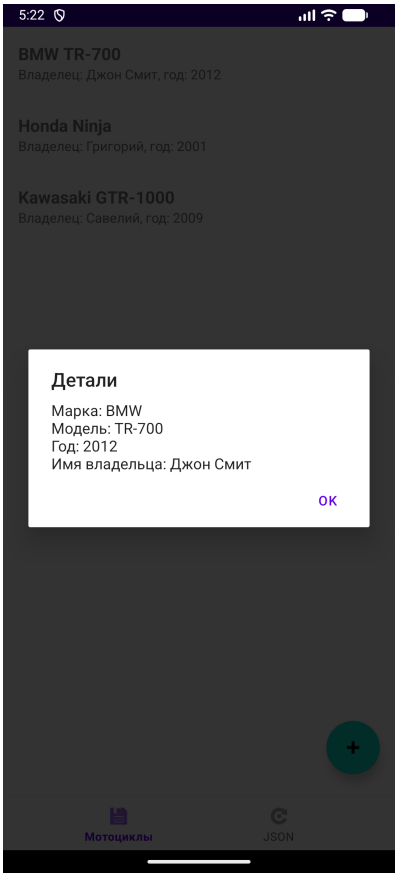

Рисунок 4 — Действие «Показать»

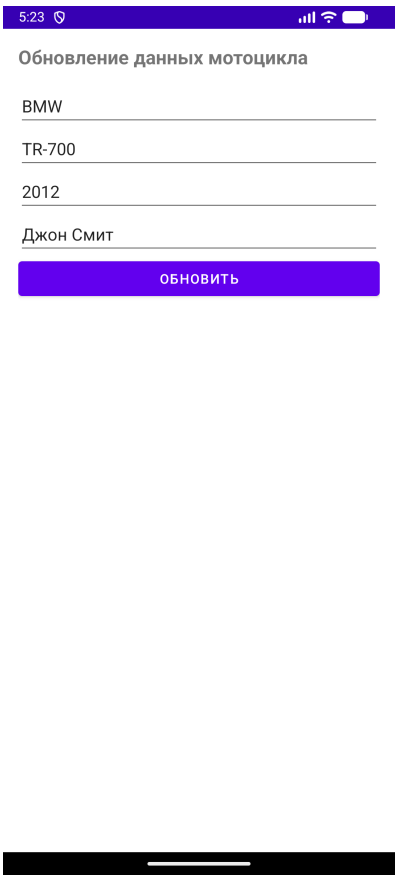

Рисунок 5 — Обновление записи о мотоцикле

Рисунок 6 — Экран подтверждения удаления записи

Рисунок 7 — Экран скачивания JSON-документа

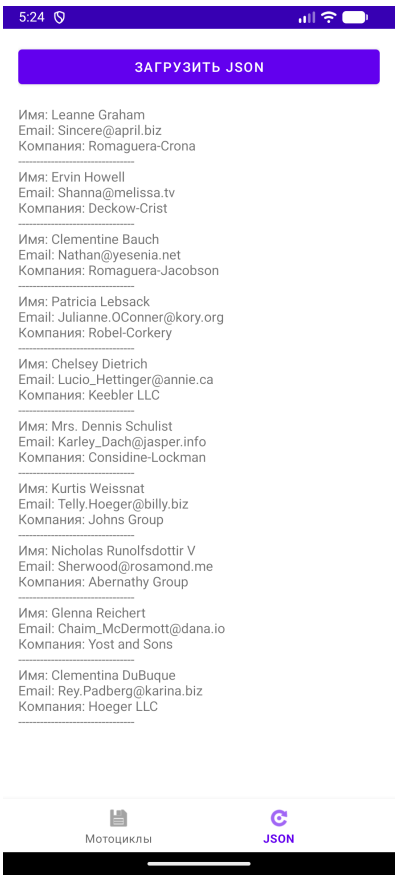

Рисунок 8 — Результат парсинга полученного JSON-объекта

5 Выводы

В ходе выполнения лабораторной работы было разработано Android-приложение, включающее работу с локальной базой данных и обработку данных в формате JSON. В процессе работы была реализована база данных с использованием библиотеки Room, что позволило на практике освоить современный подход к работе с SQLite в Android-приложениях.

В приложении была реализована полноценная CRUD-логика: добавление, просмотр, обновление и удаление данных о мотоциклах. Для взаимодействия с базой данных использовались отдельные Activity, а вывод информации был организован с помощью RecyclerView и фрагментов. Также была реализована обработка долгого нажатия на элементы списка с отображением диалогового окна, что позволило закрепить навыки работы с пользовательским интерфейсом и событиями.

Отдельное внимание было уделено работе с удалёнными данными в формате JSON. В рамках задания был реализован сетевой запрос с использованием HttpURLConnection, выполнено получение и парсинг JSON-данных из внешнего источника. Обработка данных осуществлялась в отдельном потоке с использованием Thread, что позволило избежать блокировки главного потока и корректно отобразить результат в интерфейсе приложения.

В результате выполнения лабораторной работы были закреплены навыки работы с Room Database, RecyclerView, фрагментами, многопоточностью и сетевыми запросами в Android. Разработанное приложение демонстрирует комплексный подход к созданию Android-программ с использованием как локальных, так и удалённых источников данных, а также соответствует требованиям архитектуры и пользовательского взаимодействия.

ПРИЛОЖЕНИЕ А

Листинг 1 — MainActivity.kt

```
package com.grigorijtomczuk.dbuser

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.viewpager2.adapter.FragmentStateAdapter
import androidx.viewpager2.widget.ViewPager2
import com.google.android.material.bottomnavigation.BottomNavigationView
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity() {

    private lateinit var viewPager: ViewPager2
    private lateinit var bottomNavigation: BottomNavigationView
    private lateinit var fab: FloatingActionButton

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

        viewPager = findViewById(R.id.viewPager)
        bottomNavigation = findViewById(R.id.bottomNavigation)
        fab = findViewById(R.id.fab)

        val adapter = ViewPagerAdapter(this)
        viewPager.adapter = adapter

        viewPager.registerOnPageChangeCallback(object :
ViewPager2.OnPageChangeCallback() {
            override fun onPageSelected(position: Int) {
                super.onPageSelected(position)
                when (position) {
                    0 -> {
                        bottomNavigation.selectedItemId = R.id.navigation_db
                        fab.show()
                    }
                    1 -> {
                        bottomNavigation.selectedItemId = R.id.navigation_json
                        fab.hide()
                    }
                }
            }
        })

        bottomNavigation.setOnItemSelectedListener { item ->
            when (item.itemId) {
                R.id.navigation_db -> viewPager.currentItem = 0
                R.id.navigation_json -> viewPager.currentItem = 1
            }
            true
        }

        fab.setOnClickListener {
            val intent = Intent(this, AddActivity::class.java)
            startActivity(intent)
        }
```

```
    }

    private class ViewPagerAdapter(activity: AppCompatActivity) :
FragmentStateAdapter(activity) {
        override fun getItemCount(): Int = 2

        override fun createFragment(position: Int): Fragment {
            return when (position) {
                0 -> DbFragment()
                1 -> JsonFragment()
                else -> DbFragment()
            }
        }
    }
}
```

Листинг 2 — data/AppDatabase.kt

```
package com.grigorijtomczuk.dbuser.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [Motorcyclist::class], version = 1, exportSchema = false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun motorcyclistDao(): MotorcyclistDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "motorcyclist_database"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Листинг 3 — data/Motorcyclist.kt

```
package com.grigorijtomczuk.dbuser.data

import androidx.room.Entity
import androidx.room.PrimaryKey
import java.io.Serializable

@Entity(tableName = "motorcyclists")
data class Motorcyclist(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val brand: String,
    val model: String,
```

```
    val year: Int,
    val ownerName: String
) : Serializable
```

Листинг 4 — data/MotorcyclistDao.kt

```
package com.grigorijtomczuk.dbuser.data

import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query
import androidx.room.Update

@Dao
interface MotorcyclistDao {
    @Query("SELECT * FROM motorcyclists")
    fun getAll(): LiveData<List<Motorcyclist>>

    @Insert
    suspend fun insert(motorcyclist: Motorcyclist)

    @Update
    suspend fun update(motorcyclist: Motorcyclist)

    @Delete
    suspend fun delete(motorcyclist: Motorcyclist)

    @Query("SELECT * FROM motorcyclists WHERE id = :id")
    suspend fun getById(id: Long): Motorcyclist?
}
```

Листинг 5 — data/MotorcyclistRepository.kt

```
package com.grigorijtomczuk.dbuser.data

import androidx.lifecycle.LiveData

class MotorcyclistRepository(private val motorcyclistDao: MotorcyclistDao) {
    val allMotorcyclists: LiveData<List<Motorcyclist>> = motorcyclistDao.getAll()

    suspend fun insert(motorcyclist: Motorcyclist) {
        motorcyclistDao.insert(motorcyclist)
    }

    suspend fun update(motorcyclist: Motorcyclist) {
        motorcyclistDao.update(motorcyclist)
    }

    suspend fun delete(motorcyclist: Motorcyclist) {
        motorcyclistDao.delete(motorcyclist)
    }
}
```

Листинг 6 — MotorcyclistViewModel.kt

```
package com.grigorijtomczuk.dbuser

import android.app.Application
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
```

```
import androidx.lifecycle.viewModelScope
import com.grigorijtomczuk.dbuser.data.AppDatabase
import com.grigorijtomczuk.dbuser.data.Motorcyclist
import com.grigorijtomczuk.dbuser.data.MotorcyclistRepository
import kotlinx.coroutines.launch

class MotorcyclistViewModel(application: Application) : AndroidViewModel(application)
{

    private val repository: MotorcyclistRepository
    val allMotorcyclists: LiveData<List<Motorcyclist>>

    init {
        val motorcyclistDao = AppDatabase.getDatabase(application).motorcyclistDao()
        repository = MotorcyclistRepository(motorcyclistDao)
        allMotorcyclists = repository.allMotorcyclists
    }

    fun insert(motorcyclist: Motorcyclist) = viewModelScope.launch {
        repository.insert(motorcyclist)
    }

    fun update(motorcyclist: Motorcyclist) = viewModelScope.launch {
        repository.update(motorcyclist)
    }

    fun delete(motorcyclist: Motorcyclist) = viewModelScope.launch {
        repository.delete(motorcyclist)
    }
}
```

Листинг 7 — AddActivity.kt

```
package com.grigorijtomczuk.dbuser

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider
import com.grigorijtomczuk.dbuser.data.Motorcyclist

class AddActivity : AppCompatActivity() {

    private lateinit var viewModel: MotorcyclistViewModel
    private lateinit var etBrand: EditText
    private lateinit var etModel: EditText
    private lateinit var etYear: EditText
    private lateinit var etOwner: EditText
    private lateinit var btnSave: Button

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

        viewModel = ViewModelProvider(this)[MotorcyclistViewModel::class.java]

        etBrand = findViewById(R.id.etBrand)
        etModel = findViewById(R.id.etModel)
        etYear = findViewById(R.id.etYear)
        etOwner = findViewById(R.id.etOwner)
```

```
        btnSave = findViewById(R.id.btnSave)

        btnSave.setOnClickListener {
            val brand = etBrand.text.toString()
            val model = etModel.text.toString()
            val yearStr = etYear.text.toString()
            val owner = etOwner.text.toString()

            if (brand.isNotEmpty() && model.isNotEmpty() && yearStr.isNotEmpty() &&
owner.isNotEmpty()) {
                val year = yearStr.toIntOrNull()
                if (year != null) {
                    val motorcyclist =
                        Motorcyclist(brand = brand, model = model, year = year,
ownerName = owner)
                    viewModel.insert(motorcyclist)
                    finish()
                } else {
                    Toast.makeText(this, "Укажите год", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(this, "Укажите все данные", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Листинг 8 — UpdateActivity.kt

```
package com.grigorijtomczuk.dbuser

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider
import com.grigorijtomczuk.dbuser.data.Motorcyclist

class UpdateActivity : AppCompatActivity() {

    private lateinit var viewModel: MotorcyclistViewModel
    private lateinit var etBrand: EditText
    private lateinit var etModel: EditText
    private lateinit var etYear: EditText
    private lateinit var etOwner: EditText
    private lateinit var btnUpdate: Button
    private lateinit var currentMotorcyclist: Motorcyclist

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

        viewModel = ViewModelProvider(this)[MotorcyclistViewModel::class.java]

        etBrand = findViewById(R.id.etBrand)
        etModel = findViewById(R.id.etModel)
        etYear = findViewById(R.id.etYear)
        etOwner = findViewById(R.id.etOwner)
        btnUpdate = findViewById(R.id.btnUpdate)

        @Suppress("DEPRECATION")
```

```
        currentMotorcyclist = intent.getSerializableExtra("motorcyclist") as
Motorcyclist

        etBrand.setText(currentMotorcyclist.brand)
        etModel.setText(currentMotorcyclist.model)
        etYear.setText(currentMotorcyclist.year.toString())
        etOwner.setText(currentMotorcyclist.ownerName)

        btnUpdate.setOnClickListener {
            val brand = etBrand.text.toString()
            val model = etModel.text.toString()
            val yearStr = etYear.text.toString()
            val owner = etOwner.text.toString()

            if (brand.isNotEmpty() && model.isNotEmpty() && yearStr.isNotEmpty() &&
owner.isNotEmpty()) {
                val year = yearStr.toIntOrNull()
                if (year != null) {
                    val updatedMotorcyclist = currentMotorcyclist.copy(
                        brand = brand,
                        model = model,
                        year = year,
                        ownerName = owner
                    )
                    viewModel.update(updatedMotorcyclist)
                    finish()
                } else {
                    Toast.makeText(this, "Укажите год", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(this, "Укажите все данные", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```