ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 5

# РАБОТА С HTTP API. FRAGMENTS

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 открытого API и с реализацией навигации.

## 2 Задание

Работа выполнялась по варианту № 17 (мотоциклисты).

1. Создайте приложение, которое получает данные с открытого API (например, данные о погоде) и отображает их пользователю в удобном формате. Обеспечьте обработку ошибок при отсутствии интернет-соединения.
2. Задание «Фрагменты»
    1. Создание Navigation Drawer или Bottom Navigation в соответствии с вариантом (предметной областью) – для определенного класса.
    2. Реализовать навигацию по пунктам меню.
    3. Создать фрагмент.
    4. Обработка переходов между фрагментами.
    5. Создать список с использованием RecycleView.

## 3 Листинг программы

Листинг программного кода приложения представлен в Приложении А.

## 4 Результаты работы программы

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

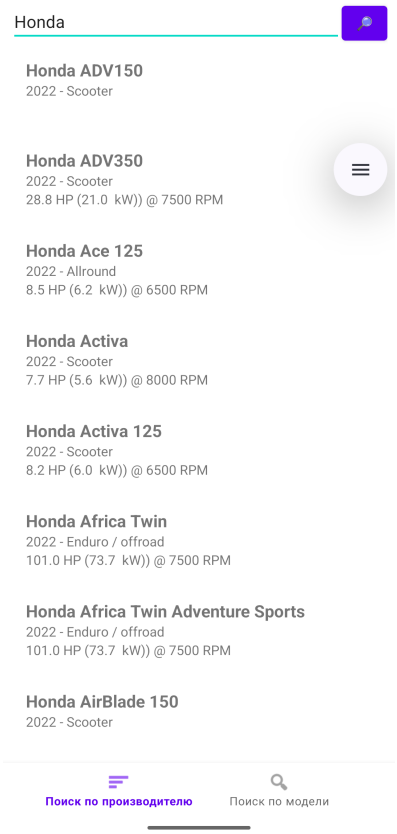


Рисунок 1 — Меню поиска мотоцикла по производителю

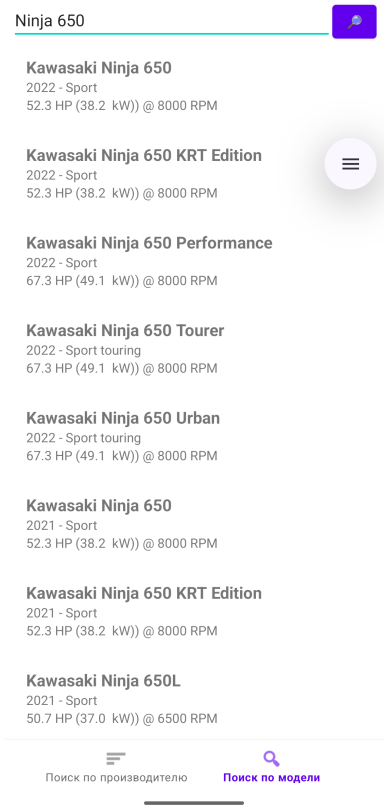


Рисунок 2 — Меню поиска мотоцикла по модели

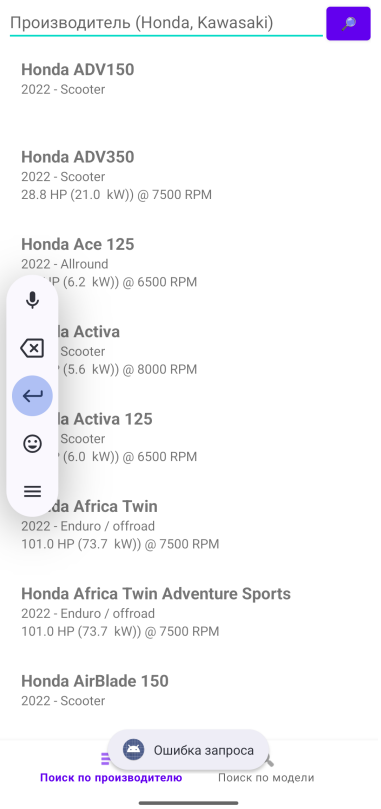


Рисунок 3 — Ошибка при пустом запросе

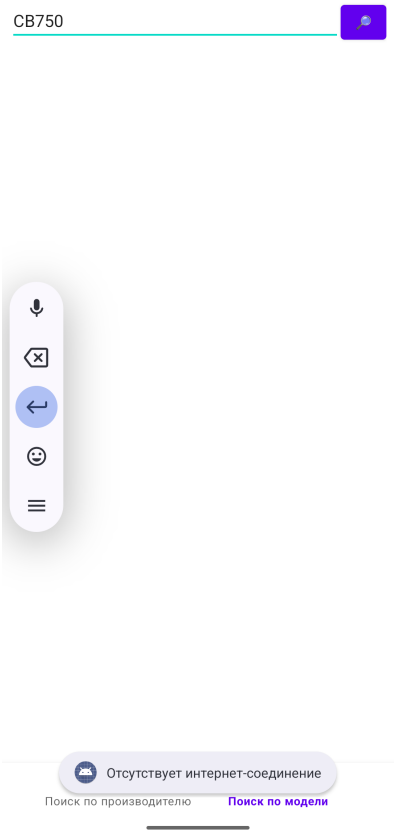


Рисунок 4 — Ошибка при отсутствии интернета

## 5 Выводы

В ходе выполнения лабораторной работы было разработано мобильное приложение для операционной системы Android на языке программирования Kotlin с использованием открытого API. В качестве предметной области были выбраны данные о мотоциклах (вариант № 17). Приложение получает информацию из «API Ninjas» Motorcycles API и отображает её пользователю в удобном и наглядном виде.

В процессе работы была реализована навигация между экранами приложения с использованием компонентов навигации Android. Приложение содержит два основных пункта меню: поиск мотоциклов по производителю и поиск по модели. Для каждого пункта меню создан отдельный фрагмент, обеспечивающий ввод поискового запроса через элемент EditText и отображение результатов в виде списка.

Для вывода данных был реализован список на основе RecyclerView, позволяющий эффективно отображать результаты запросов, полученных от API. Также была настроена обработка сетевых запросов и предусмотрена обработка ошибок, возникающих при отсутствии интернет-соединения или недоступности сервиса.

В результате выполнения работы были освоены основы работы с открытыми API, организация навигации с использованием фрагментов, создание пользовательского интерфейса с несколькими экранами и списками, а также принципы взаимодействия с сетевыми данными в Android-приложениях.

## ПРИЛОЖЕНИЕ А

Листинг 1 — MainActivity.kt

```
package com.grigorijtomczuk.apiuser

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.findNavController
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.setupWithNavController
import com.google.android.material.bottomnavigation.BottomNavigationView
import com.grigorijtomczuk.apiuser.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val navView: BottomNavigationView = binding.navView

        val navController = findNavController(R.id.nav_host_fragment_activity_main)
        AppBarConfiguration(
            setOf(
                R.id.navigation_motorcycles, R.id.navigation_models
            )
        )
        navView.setupWithNavController(navController)
    }
}
```

Листинг 2 — Motorcycle.kt

```
package com.grigorijtomczuk.apiuser.api

import com.google.gson.annotations.SerializedName

data class Motorcycle(
    @SerializedName("make") val make: String,
    @SerializedName("model") val model: String,
    @SerializedName("year") val year: Int,
    @SerializedName("type") val type: String,
    @SerializedName("power") val power: String
)
```

Листинг 3 — ApiService.kt

```
package com.grigorijtomczuk.apiuser.api

import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Header
import retrofit2.http.Query

interface ApiService {
    @GET("motorcycles")
    suspend fun getMotorcycles(
```

```
        @Header("X-Api-Key") apiKey: String =
"4hmnPRfAo9Cmst+a/XuQLA==jFZfmwEbiPVbBv1u",
        @Query("make") make: String?,
        @Query("model") model: String?,
//        @Query("year") year: Int?,
    ): Response<List<Motorcycle>>
}
```

Листинг 4 — ApiClient.kt

```
package com.grigorijtomczuk.apiuser.api

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object ApiClient {
    private const val BASE_URL = "https://api.api-ninjas.com/v1/"

    val instance: ApiService by lazy {
        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        retrofit.create(ApiService::class.java)
    }
}
```

Листинг 5 — MotorcyclesFragment.kt

```
package com.grigorijtomczuk.apiuser.ui.motorcycles

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.LinearLayoutManager
import com.grigorijtomczuk.apiuser.databinding.FragmentMotorcyclesBinding
import com.grigorijtomczuk.apiuser.viewmodel.MotorcyclesViewModel

class MotorcyclesFragment : Fragment() {

    private var _binding: FragmentMotorcyclesBinding? = null
    private val binding get() = _binding!!

    private lateinit var viewModel: MotorcyclesViewModel
    private lateinit var motorcyclesAdapter: MotorcyclesAdapter

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentMotorcyclesBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
```

```
        viewModel = ViewModelProvider(this).get(MotorcyclesViewModel::class.java)

        setupRecyclerView()

        binding.searchButton.setOnClickListener {
            val make = binding.makeEditText.text.toString().trim()
            viewModel.fetchMotorcycles(make = make, model = "")
        }

        viewModel.motorcycles.observe(viewLifecycleOwner) {
            motorcyclesAdapter.updateData(it)
        }

        viewModel.isLoading.observe(viewLifecycleOwner) { isLoading ->
            binding.progressBar.visibility = if (isLoading) View.VISIBLE else
View.GONE
        }

        viewModel.error.observe(viewLifecycleOwner) { error ->
            Toast.makeText(context, error, Toast.LENGTH_LONG).show()
        }
    }

    private fun setupRecyclerView() {
        motorcyclesAdapter = MotorcyclesAdapter(emptyList())
        binding.motorcyclesRecyclerView.apply {
            adapter = motorcyclesAdapter
            layoutManager = LinearLayoutManager(context)
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

Листинг 6 — MotorcyclesAdapter.kt

```
package com.grigorijtomczuk.apiuser.ui.motorcycles

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.grigorijtomczuk.apiuser.api.Motorcycle
import com.grigorijtomczuk.apiuser.databinding.MotorcycleItemBinding

class MotorcyclesAdapter(private var motorcycles: List<Motorcycle>) :
    RecyclerView.Adapter<MotorcyclesAdapter.MotorcycleViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MotorcycleViewHolder {
        val binding =
            MotorcycleItemBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return MotorcycleViewHolder(binding)
    }

    override fun onBindViewHolder(holder: MotorcycleViewHolder, position: Int) {
        val motorcycle = motorcycles[position]
        holder.bind(motorcycle)
    }
```

```
    override fun getItemCount() = motorcycles.size

    fun updateData(motorcycles: List<Motorcycle>) {
        this.motorcycles = motorcycles
        notifyDataSetChanged()
    }

    class MotorcycleViewHolder(private val binding: MotorcycleItemBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun bind(motorcycle: Motorcycle) {
            binding.motorcycleMakeModel.text = "${motorcycle.make}
${motorcycle.model}"
            binding.motorcycleYearType.text = "${motorcycle.year} -
${motorcycle.type}"
            binding.motorcyclePower.text = motorcycle.power
        }
    }
}
```

Листинг 7 — MotorcyclesViewModel.kt

```
package com.grigorijtomczuk.apiuser.viewmodel

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.grigorijtomczuk.apiuser.api.ApiClient
import com.grigorijtomczuk.apiuser.api.Motorcycle
import kotlinx.coroutines.launch

class MotorcyclesViewModel : ViewModel() {

    private val _motorcycles = MutableLiveData<List<Motorcycle>>()
    val motorcycles: LiveData<List<Motorcycle>> = _motorcycles

    private val _isLoading = MutableLiveData<Boolean>()
    val isLoading: LiveData<Boolean> = _isLoading

    private val _error = MutableLiveData<String>()
    val error: LiveData<String> = _error

    fun fetchMotorcycles(make: String, model: String) {
        viewModelScope.launch {
            _isLoading.value = true
            try {
                val response = ApiClient.instance.getMotorcycles(make = make, model =
model)
                if (response.isSuccessful && response.body() != null) {
                    _motorcycles.value = response.body()!!
                } else {
                    _error.value = "Ошибка запроса"
                }
            } catch (e: Exception) {
                if (e is java.net.SocketTimeoutException || e is
java.net.UnknownHostException)
                    _error.value = "Отсутствует интернет-соединение"
                else
                    _error.value = "Возникла ошибка: ${e.message}"
            }
            _isLoading.value = false
```

```
            }
        }
    }
```