ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

Намерения, меню и работа с данными

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, 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.

Вариант 24

Предметная область - Попугаи.

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

Ход разработки:

1. Разработать главную активность
2. Разработать фрагменты: HomeFragment, ParrotsFragment, SettingsFragment
3. Разработать класс-адаптер ParrotAdapter для отображения списка попугаев в RecyclerView
4. Создать файлы разметки для фрагментов и элементов списка
5. Реализовать навигацию между фрагментами через BottomNavigationView
6. Реализовать получение данных о погоде с открытого API (например, OpenWeatherMap) с обработкой ошибок
7. Обеспечить обработку отсутствия интернет-соединения.

Используемые технологии:

- Kotlin – используется для работы написанного кода.
- AppCompat – визуальная часть, «тема» приложения.
- Retrofit – для работы с сетевыми запросами к API.
- Gson – для преобразования JSON-ответов в Kotlin-объекты.
- RecyclerView – для отображения списка попугаев.
- BottomNavigationView – для навигации между фрагментами.
- Material Design – для современного вида интерфейса.

Описание структуры работы частей приложения:

1. Основная часть — MainActivity — точка входа приложения

MainActivity является стартовым экраном и обеспечивает управление списком задач.

а. Реакция приложения на событие onCreate()

1. Загрузка основной разметки интерфейса (activity_main.xml).
2. Инициализация элементов UI:
   - BottomNavigationView — для навигации,
   - FrameLayout — контейнер для фрагментов,
   - TextView — для отображения погоды.
3. Запуск API-запроса для получения погоды.
4. Настройка обработчика нажатий на BottomNavigationView.
5. Загрузка начального фрагмента (HomeFragment).

2. Визуальная часть — интерфейс главной активности

1. Используемые переменные
   - BottomNavigationView — для переключения между фрагментами.
   - FrameLayout — отображает текущий фрагмент.
      a. Контейнер (LinearLayout) содержит:
         i. FrameLayout — отображает фрагменты.
         ii. BottomNavigationView — содержит кнопки навигации.

3. Фрагменты

1) HomeFragment — отображает приветственное сообщение.
2) ParrotsFragment — содержит RecyclerView с адаптером для отображения списка попугаев.
3) SettingsFragment — отображает информацию о настройках.

4. Адаптер списка — ParrotAdapter

1) Визуальная часть элемента списка – адаптер связывает список попугаев с разметкой item_parrot.xml.
2) Обработка данных
   a. Создаёт ViewHolder для каждого элемента списка.
   b. Устанавливает текст имени попугая в TextView.

5. Навигация между фрагментами

1) При нажатии на кнопку в BottomNavigationView вызывается replaceFragment(...).
2) Метод replaceFragment заменяет текущий фрагмент в FrameLayout на новый, используя FragmentTransaction.

6. Получение данных с API

1) В MainActivity создаётся экземпляр Retrofit для работы с API.
2) Выполняется асинхронный запрос к API.
3) При успешном ответе данные отображаются в TextView.
4) При ошибке (например, нет интернета) выводится сообщение об ошибке.

7. Обработка ошибок

1) Проверка подключения к интернету перед выполнением запроса.
2) Отображение пользовательского сообщения при отсутствии подключения или ошибке сети.

4. Скриншоты, иллюстрирующие результаты работы программы:

В качестве демонстрации работы программы приведём несколько скриншотов каждого фрагмента.

Рисунок 2.1 – Стартовый экран

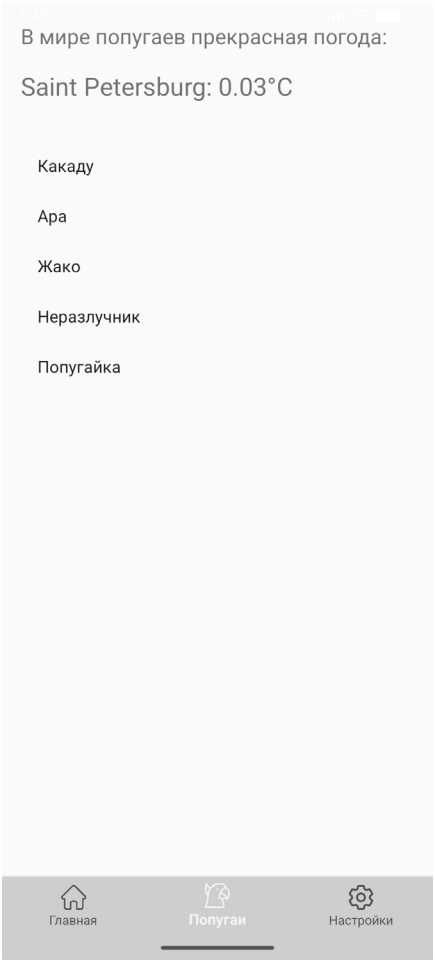

Рисунок 2.2 – Список попугаев

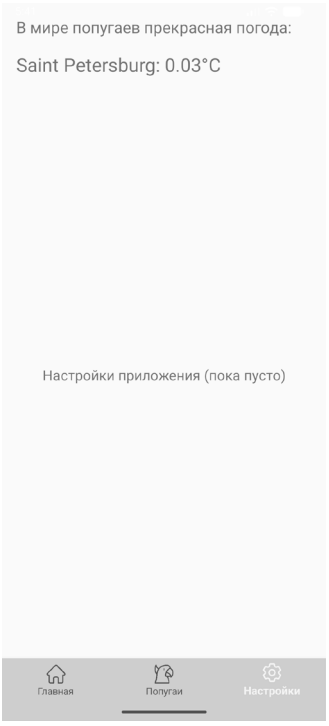

Рисунок 2.3 – Пустая вкладка настроек

Рисунок 2.4 – Результат работы API, при отсутствии интернета

5. Вывод:

В ходе выполнения работы было разработано Android-приложение на языке Kotlin.

Приложение реализует функциональность получения данных с открытого API, отображения информации пользователю, а также обеспечивает навигацию между экранами с помощью фрагментов и компонента Bottom Navigation.

В рамках работы реализованы ключевые аспекты разработки: интеграция сетевых запросов с использованием библиотеки Retrofit, обработка ошибок, включая отсутствие интернет-соединения, и построение списка элементов с помощью RecyclerView.

Предметная область – попугаи – отражена в содержимом списка.

Разработанное приложение демонстрирует понимание основ взаимодействия компонентов Android-приложения, а также навыки работы с API, фрагментами и адаптерами.

ПРИЛОЖЕНИЕ А

Листинг программы:

MainActivity.kt:

```
package com.example.papoogi
package com.example.a6lr
import android.Manifest
import android.content.Context
import android.content.pm.PackageManager
import android.net.ConnectivityManager
import android.os.Build
import android.os.Bundle
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import com.google.android.material.bottomnavigation.BottomNavigationView
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class MainActivity : AppCompatActivity() {

    private val REQUEST_CODE_PERMISSIONS = 100

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

        // Проверяем разрешения
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.INTERNET) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.INTERNET), REQUEST_CODE_PERMISSIONS)
        }

        // Запускаем API-запрос
        fetchWeatherData()

        // Устанавливаем навигацию
        setupNavigation()
    }

    private fun fetchWeatherData() {
        val retrofit = Retrofit.Builder()
            .baseUrl("https://api.openweathermap.org/data/2.5/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()

        val service = retrofit.create(WeatherApiService::class.java)

        val call = service.getCurrentWeather("Saint Petersburg",
"0afc736cf6ba687d3e6d20ed62574b95", "metric")

        call.enqueue(object : Callback<WeatherResponse> {
            override fun onResponse(call: Call<WeatherResponse>, response:
Response<WeatherResponse>) {
                if (response.isSuccessful) {
```

```
                    val weather = response.body()
                    findViewById<TextView>(R.id.textView).text =
"${weather?.name}: ${weather?.main?.temp}°C"
                } else {
                    Toast.makeText(this@MainActivity, "Ошибка:
${response.code()}", Toast.LENGTH_SHORT).show()
                }
            }

            override fun onFailure(call: Call<WeatherResponse>, t: Throwable)
{
                if (!isNetworkAvailable()) {
                    Toast.makeText(this@MainActivity, "Нет подключения к
интернету", Toast.LENGTH_LONG).show()
                } else {
                    Toast.makeText(this@MainActivity, "Ошибка сети:
${t.message}", Toast.LENGTH_SHORT).show()
                }
            }
        })
    }

    private fun isNetworkAvailable(): Boolean {
        val connectivityManager =
getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager

        return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            val network = connectivityManager.activeNetwork
            val capabilities =
connectivityManager.getNetworkCapabilities(network)

capabilities?.hasCapability(android.net.NetworkCapabilities.NET_CAPABILITY_IN
TERNET) == true
        } else {
            @Suppress("DEPRECATION")
            val networkInfo = connectivityManager.activeNetworkInfo
            networkInfo?.isConnected == true
        }
    }

    private fun setupNavigation() {
        val bottomNav =
findViewById<BottomNavigationView>(R.id.bottom_navigation)

        bottomNav.setOnItemSelectedListener { item ->
            when (item.itemId) {
                R.id.nav_home -> {
                    replaceFragment(HomeFragment())
                    true
                }
                R.id.nav_parrots -> {
                    replaceFragment(ParrotsFragment())
                    true
                }
                R.id.nav_settings -> {
                    replaceFragment(SettingsFragment())
                    true
                }
                else -> false
            }
        }

        // Загрузка начального фрагмента
```

```
        replaceFragment(HomeFragment())
    }

    private fun replaceFragment(fragment: Fragment) {
        supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, fragment)
            .commit()
    }
}
```