АКТ (ф) СПб ГУТ

Отчет

По лабораторным работам №1-30

Разработка мобильных приложений

Студент ИСПП-21 Капитанская Д.А

Преподаватель Садовский Р.В

Архангельск 2025

# Лабораторная работа №1

## Установка инструментария и настройка среды для разработки мобильных приложений

1. Цель работы
   1. Изучить процесс установки среды для разработки мобильных приложений.
2. Ответы на контрольные вопросы
   1. Какие среды разработки поддерживают создание мобильных приложений?

Android Studio — официальная среда для разработки Android-приложений

Xcode — среда разработки для iOS-приложений (macOS)

Visual Studio с Xamarin — кроссплатформенная разработка на C#

Flutter (через Android Studio, VS Code и др.) — кроссплатформенная разработка на Dart

React Native (через VS Code и др.) — кроссплатформенная разработка на JavaScript

Unity — для разработки мобильных игр и приложений на C#

* 1. На каких языках программирования можно создавать мобильные приложения?

Java (Android)

Kotlin (Android)

Swift (iOS)

Objective-C (iOS)

C# (Xamarin, Unity)

Dart (Flutter)

JavaScript/TypeScript (React Native)

C++ (через NDK для Android, а также для кроссплатформенных движков)

* 1. Какие языки программирования доступны в Android Studio?

Kotlin (официальный язык Android-разработки)

Java

C/C++ (через Android NDK)

Также можно использовать Dart (через плагины Flutter)

* 1. Какой язык программирования является основным для разработки приложений под ОС Android?

Kotlin — официальный и рекомендуемый язык разработки Android-приложений. Ранее основным был Java.

* 1. Что дополнительно нужно установить для работы Android Studio?

Java Development Kit (JDK) — для компиляции Java/Kotlin-кода

Android SDK — набор инструментов и библиотек для разработки Android-приложений

Android Emulator (эмулятор Android-устройств)

Опционально: Android NDK — для разработки с использованием C/C++

* 1. Что такое Java Development Kit и что входит в его состав?

Java Development Kit (JDK) — это комплект инструментов для разработки на языке Java.

В его состав входят:

Java Compiler (javac) — компилятор исходного кода Java в байт-код

Java Runtime Environment (JRE) — среда выполнения Java-приложений, включая JVM

Стандартные библиотеки Java (API)

Инструменты для отладки, профилирования и управления приложениями (javadoc, jar, jdb и др.)

# Лабораторная работа №2

## Установка среды разработки мобильных приложений с применением виртуальной машины

1. Цель работы
   1. Изучить процесс установки среды для разработки мобильных приложений с применением виртуальной машины.
2. Ответы на контрольные вопросы
   1. Что такое «виртуальная машина»?

Виртуальная машина (Virtual Machine, VM) — это программная среда, которая эмулирует аппаратное обеспечение компьютера и позволяет запускать программы в изолированном и переносимом виде. Например, Java Virtual Machine (JVM) выполняет байт-код Java, обеспечивая независимость приложений от конкретной платформы.

* 1. Какие IDE позволяют создавать приложения для Android?

Android Studio — официальная IDE для Android-разработки

IntelliJ IDEA — на базе которой построена Android Studio, поддерживает Android-проекты

Eclipse с плагином ADT (устаревшая, но ранее популярная)

Visual Studio с Xamarin для кроссплатформенной разработки

VS Code с расширениями для Flutter или React Native

* 1. Какие языки программирования доступны в Intellij IDEA?

Java

Kotlin

Groovy

Scala

Python (с плагинами)

JavaScript и TypeScript (с плагинами)

SQL и другие языки через плагины

* 1. Какие языки программирования доступны в Android Studio?

Kotlin (официальный язык Android)

Java

C/C++ (через Android NDK)

Dart (через плагин Flutter)

* 1. На каких языках программирования можно разрабатывать нативные приложения для Android?

Kotlin (рекомендуемый официальный язык)

Java

C и C++ (через Android NDK для части кода с высокой производительностью)

# Лабораторная работа №3

## Разработка линейных алгоритмов

1. Цель работы
   1. Изучить процесс разработки линейных алгоритмов в приложениях на языке Kotlin.
   2. 1.2 Изучить процесс ввода и вывода данных в приложениях на языке Kotlin.
2. Ответы на контрольные вопросы
   1. Как объявить переменные на Kotlin?

В Kotlin переменные объявляются с помощью ключевых слов var и val:

var mutableVariable: Int = 10 // изменяемая переменная

val immutableVariable: String = "Hello" // неизменяемая (константа)

* 1. Как выполнить ввод данных на Kotlin?

Для чтения строки с консоли используется функция readLine():

val input = readLine() // возвращает String? (nullable String)

Чтобы получить не null значение, можно использовать оператор безопасного вызова или проверку:

val input = readLine() ?: ""

* 1. Как выполнить вывод данных на Kotlin?

Для вывода на консоль используется функция println() (с переводом строки) или print() (без перевода):

println("Hello, world!")

print("Введите число: ")

* 1. Как преобразовать значение из строкового в числовой?

Как преобразовать значение из строкового в числовой?

Для преобразования строки в число используются методы типа:

val number = input.toInt() // преобразовать в Int

val doubleNumber = input.toDouble() // преобразовать в Double

Если строка может быть некорректной, лучше использовать безопасные версии:

val number = input.toIntOrNull() // вернёт null, если преобразование невозможно

* 1. Как округлить данные на Kotlin?

Для округления чисел можно использовать функции из пакета kotlin.math:

import kotlin.math.\*

val x = 3.14159

val rounded = x.roundToInt() // округление до Int

val floorValue = floor(x) // округление вниз

val ceilValue = ceil(x) // округление вверх

val roundedDouble = round(x \* 100) / 100.0 // округление до 2 знаков после запятой

* 1. Как сгенерировать случайное число на Kotlin?

Используется класс kotlin.random.Random:

import kotlin.random.Random

val randomInt = Random.nextInt() // любое Int

val randomIntInRange = Random.nextInt(0, 100) // от 0 до 99

val randomDouble = Random.nextDouble() // от 0.0 до 1.0

* 1. В чем отличие между ключевыми словами var и val в Kotlin?

var — объявляет изменяемую переменную, значение которой можно менять после инициализации.

val — объявляет неизменяемую переменную (константу), значение которой нельзя изменить после присвоения.

Использование val рекомендуется, если значение не предполагается менять, что повышает безопасность и читаемость кода.

# Лабораторная работа №4

## Разработка разветвляющихся алгоритмов

1. Цель работы
   1. Изучить процесс разработки разветвляющихся алгоритмов в приложениях на Kotlin.
2. Ответы на контрольные вопросы
   1. Каков синтаксис условного оператора на Kotlin?

В Kotlin условный оператор реализуется с помощью ключевого слова if:

if (условие) {

// блок кода, если условие истинно

} else {

// блок кода, если условие ложно

}

* 1. Каков синтаксис оператора множественного выбора на Kotlin?

В Kotlin используется оператор when, аналогичный switch в других языках:

when (выражение) {

значение1 -> {

// действия для значение1

}

значение2 -> {

// действия для значение2

}

else -> {

// действия, если ни одно значение не подошло

}

}

* 1. Каков синтаксис тернарного оператора на Kotlin?

В Kotlin нет тернарного оператора ?:, как в Java или C. Вместо него используется выражение if как выражение:

val max = if (a > b) a else b

То есть if в Kotlin возвращает значение и может использоваться в выражениях.

# Лабораторная работа №5

## Разработка циклических алгоритмов

1. Цель работы
   1. Изучить процесс разработки циклических алгоритмов в приложениях на Kotlin.
2. Ответы на контрольные вопросы
   1. Каков синтаксис оператор for на Kotlin?

В Kotlin цикл for используется для перебора элементов коллекций или диапазонов:

for (item in collection) {

// тело цикла

}

* 1. Каков синтаксис оператора while на Kotlin?

Каков синтаксис оператора while на Kotlin?

Цикл while выполняет тело, пока условие истинно:

while (условие) {

// тело цикла

}

* 1. Каков синтаксис оператора do-while на Kotlin?

Цикл do-while сначала выполняет тело, затем проверяет условие:

do {

// тело цикла

} while (условие)

* 1. Какие операторы досрочного выхода из цикла применяются в Kotlin?

break — прерывает выполнение ближайшего внешнего цикла полностью.

continue — пропускает остаток текущей итерации и переходит к следующей итерации цикла.

# Лабораторная работа №6

## Разработка и вызов функций

1. Цель работы
   1. Изучить процесс разработки функций в приложениях на Kotlin.
2. Ответы на контрольные вопросы
   1. Каков синтаксис создания функций на Kotlin?

Функции объявляются с помощью ключевого слова fun:

fun имяФункции(параметры): ТипВозвращаемогоЗначения {

// тело функции

return значение

}

* 1. Как указать для параметров методов значения по умолчанию на Kotlin?

Для параметров можно задавать значения по умолчанию прямо в объявлении:

fun greet(name: String = "Гость") {

println("Привет, $name!")

}

При вызове функции можно опускать такие параметры, и будет использоваться значение по умолчанию:

greet() // Выведет: Привет, Гость!

greet("Иван") // Выведет: Привет, Иван!

* 1. Как указать для параметров переменное количество аргументов на Kotlin?

Используется модификатор vararg:

fun printAll(vararg messages: String) {

for (m in messages) println(m)

}

* 1. Что такое функции высокого порядка в Kotlin?

Функции высокого порядка — это функции, которые принимают другие функции в качестве параметров или возвращают функции. Например:

fun operateOnNumbers(a: Int, b: Int, operation: (Int, Int) -> Int): Int {

return operation(a, b)

}

Вызов с лямбдой:

val sum = operateOnNumbers(3, 4) { x, y -> x + y } // sum = 7

# Лабораторная работа №7

## Разработка классов

1. Цель работы
   1. Изучить процесс разработки и применения классов на языке Kotlin.
2. Ответы на контрольные вопросы
   1. Как объявить класс на Kotlin?

Класс объявляется с помощью ключевого слова class:

class ИмяКласса {

// тело класса

}

Пример:

class Person {

// свойства и методы

}

* 1. Как объявить свойства в Kotlin?

Свойства объявляются внутри класса с помощью val (только для чтения) или var (для чтения и записи):

class Person {

var name: String = "Без имени"

val birthYear: Int = 1990

}

* 1. Каков синтаксис объявления геттера и сеттера свойства в Kotlin?

Для свойства можно определить пользовательские геттер и сеттер:

var age: Int = 0

get() = field // поле field содержит текущее значение

set(value) {

if (value >= 0) field = value

}

* 1. Как объявить конструктор в Kotlin?

В Kotlin есть первичный и вторичные конструкторы.

Первичный конструктор указывается в объявлении класса:

class Person(val name: String, var age: Int)

* 1. Как создать объект в Kotlin?

Объект создаётся с помощью оператора new не требуется, просто вызывается конструктор:

val person = Person("Иван", 25)

# Лабораторная работа №8

## Наследование классов

1. Цель работы
   1. Изучить процесс разработки дочерних классов в приложениях на Kotlin.
2. Ответы на контрольные вопросы
   1. Что такое «наследование»?

Наследование — это механизм объектно-ориентированного программирования, позволяющий создавать новый класс (дочерний), который расширяет или переопределяет функциональность существующего класса (родительского). Дочерний класс наследует свойства и методы родителя, что способствует повторному использованию кода и организации иерархий.

* 1. Сколько родительских классов может быть у класса в Kotlin?

В Kotlin класс может наследоваться только от одного родительского класса (одинарное наследование).

* 1. Сколько интерфейсов может реализовывать класс в Kotlin?

Класс может реализовывать любое количество интерфейсов.

* 1. Как указать родительский класс на Kotlin?

Родительский класс указывается после имени класса через двоеточие:

open class Parent

class Child : Parent()

Обратите внимание, что родительский класс вызывается с круглыми скобками (конструктор).

* 1. Как переопределить реализацию метода в дочернем классе на Kotlin?

В родительском классе метод должен быть помечен ключевым словом open, а в дочернем — override:

open class Parent {

open fun greet() {

println("Привет из Parent")

}

}

class Child : Parent() {

override fun greet() {

println("Привет из Child")

}

}

* 1. Как вызвать реализацию родительского метода в дочернем классе на Kotlin?

Используется ключевое слово super:

override fun greet() {

super.greet() // вызов метода родителя

println("Дополнение в Child")

}

* 1. Как указать, что класс может быть родительским?

По умолчанию классы в Kotlin final (не наследуемы). Чтобы класс можно было наследовать, его нужно объявить с модификатором open:

open class Parent

* 1. Как указать, что у класса нельзя создавать дочерние классы?

По умолчанию все классы в Kotlin являются final, то есть наследование запрещено. Чтобы явно запретить наследование, ничего дополнительно делать не нужно. Если класс был объявлен с open, а вы хотите запретить наследование, уберите open.

Также можно использовать модификатор final явно:

final class SomeClass

# Лабораторная работа №9

## Обработка коллекций

1. Цель работы
   1. Изучить процесс разработки дочерних классов в приложениях на Kotlin.
2. Ответы на контрольные вопросы
   1. Какие стандартные типы данных коллекций имеются в Kotlin?

В Kotlin есть три основных типа коллекций:

List — упорядоченный список элементов. Может быть неизменяемым (List) или изменяемым (MutableList).

Set — множество уникальных элементов. Тоже бывает неизменяемым (Set) и изменяемым (MutableSet).

Map — отображение (словарь) ключ-значение. Есть неизменяемый (Map) и изменяемый (MutableMap).

* 1. Как объявить коллекцию в Kotlin?

Примеры объявления:

// Неизменяемый список

val list: List<Int> = listOf(1, 2, 3)

// Изменяемый список

val mutableList: MutableList<String> = mutableListOf("a", "b")

// Неизменяемое множество

val set: Set<String> = setOf("apple", "banana")

// Изменяемое множество

val mutableSet: MutableSet<Int> = mutableSetOf(1, 2, 3)

// Неизменяемый словарь

val map: Map<String, Int> = mapOf("one" to 1, "two" to 2)

// Изменяемый словарь

val mutableMap: MutableMap<String, String> = mutableMapOf("key" to "value")

* 1. Как добавить элемент в список в Kotlin?

Если список изменяемый (MutableList), используйте метод add:

val list = mutableListOf(1, 2, 3)

list.add(4) // теперь list = [1, 2, 3, 4]

В неизменяемом списке (List) добавить элемент нельзя, нужно создавать новый список.

* 1. Как добавить элемент в словарь в Kotlin?

Для изменяемого словаря (MutableMap) используйте индексирование или метод put:

val map = mutableMapOf("a" to 1)

map["b"] = 2

// или

map.put("c", 3)

В неизменяемом словаре (Map) добавить элемент нельзя.

* 1. Как перебрать элементы коллекции в Kotlin?

Для списков и множеств:

for (item in collection) {

println(item)

}

Для словарей (Map):

for ((key, value) in map) {

println("$key -> $value")

}

# Лабораторная работа №10

## Использование корутин

1. Цель работы
   1. Изучить процесс создания и использования сопрограмм в приложениях на Kotlin.
2. Ответы на контрольные вопросы
   1. Что такое «корутина»?

Корутина — это легковесная, приостанавливаемая и возобновляемая единица выполнения в Kotlin, которая позволяет писать асинхронный и неблокирующий код. Корутины упрощают работу с параллелизмом и асинхронностью, позволяя писать последовательный код, который при этом эффективно использует ресурсы.

* 1. Для чего используется launch?

launch — это функция из библиотеки Kotlin Coroutines, которая запускает новую корутину в заданном контексте и возвращает объект Job. Корутина, запущенная через launch, выполняется асинхронно и не блокирует вызывающий поток. Используется для запуска фоновых задач, где не требуется возвращать результат.

Пример:

GlobalScope.launch {

// код корутины

}

* 1. Какие особенности использования у suspend-функций?

suspend — модификатор, обозначающий функцию, которую можно приостанавливать и возобновлять без блокировки потока.

Такие функции можно вызывать только из других suspend-функций или внутри корутин.

suspend-функции позволяют выполнять асинхронные операции последовательно и читаемо.

* 1. Как приостановить выполнение корутины?

Выполнение корутины можно приостановить, вызвав другую suspend-функцию, которая выполняет асинхронную операцию (например, delay()). При этом поток не блокируется, а корутина ожидает возобновления. Пример:

suspend fun example() {

delay(1000L) // приостанавливает корутину на 1 секунду

}

* 1. Как отменить выполнение корутины?

Для отмены корутины используется метод cancel() у объекта Job, который возвращает launch или другие строители корутин. При отмене корутина прекращает выполнение, если она корректно обрабатывает отмену (например, проверяет isActive или использует отменяемые функции).

* 1. Как подключить библиотеку для работы с корутинами?

В build.gradle (Gradle) добавьте зависимость:

implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.7.3'

# Лабораторная работа №11

## Создание эмуляторов и подключение устройств

1. Цель работы
   1. Изучить процесс создания эмуляторов и подключения устройств для мобильной разработки.
2. Ответы на контрольные вопросы
   1. Что такое AVD?

AVD (Android Virtual Device) — это конфигурация виртуального устройства, используемого в Android Studio для эмуляции работы реального Android-устройства. AVD задаёт параметры, такие как модель устройства, версия Android, размер экрана и другие характеристики.

* 1. Что такое эмулятор?

Эмулятор — это программное средство, которое имитирует работу реального устройства (в данном случае Android-устройства) на компьютере. Он позволяет запускать и тестировать приложения без необходимости использовать физический девайс.

* 1. Что определяет профиль устройства?

Профиль устройства определяет аппаратные и программные характеристики виртуального устройства, такие как: модель (например, Pixel 5), размер и разрешение экрана, объём памяти, версия Android, наличие сенсоров и других компонентов.

* 1. Как создать новый эмулятор?

В Android Studio:

Откройте AVD Manager (через меню Tools > AVD Manager).

Нажмите «Create Virtual Device».

Выберите профиль устройства из списка.

Выберите системный образ (версию Android).

Настройте дополнительные параметры и нажмите «Finish».

* 1. Как указать расширенные настройки эмулятора?

При создании или редактировании AVD в Android Studio доступны расширенные настройки, такие как:

объём оперативной памяти и хранилища,

тип процессора (x86, ARM),

параметры графики (GPU),

ориентация экрана,

эмуляция сенсоров,

настройки сети и др.

Их можно задать в соответствующем окне конфигурации AVD.

* 1. Какие эмуляторы могут использоваться вместо эмуляторов Android Studio?

Альтернативные эмуляторы:

Genymotion — быстрый и функциональный эмулятор на базе VirtualBox.

BlueStacks — ориентирован на запуск Android-приложений на ПК.

NoxPlayer, MEmu — популярные эмуляторы для игр и приложений.

Android Emulator от Google — встроенный в Android Studio, но можно запускать и отдельно.

Эти эмуляторы могут иметь свои особенности и преимущества в скорости, совместимости и функциональности.

# Лабораторная работа №12

## Настройка режима терминала

1. Цель работы
   1. Изучить процесс использования терминала при разработке мобильных приложений Android.
2. Ответы на контрольные вопросы
   1. Что такое adb?

ADB (Android Debug Bridge) — это командная утилита для взаимодействия с Android-устройствами и эмуляторами. Она позволяет выполнять отладку, управление устройством, установку приложений, передачу файлов и другие операции с Android-устройствами из командной строки ПК.

* 1. Как узнать путь к папке с adb?

ADB входит в состав Android SDK. Путь к adb обычно находится в каталоге:

<путь\_к\_Android\_SDK>/platform-tools/

В Android Studio путь к SDK можно посмотреть в настройках (File > Settings > Appearance & Behavior > System Settings > Android SDK).

* 1. Как запустить утилиту adb?

Откройте терминал или командную строку, перейдите в папку с adb (platform-tools) или добавьте её в переменную PATH, затем выполните команду:

adb

для просмотра основных опций или любую другую команду adb, например:

adb devices

* 1. Как узнать список доступных команд adb?

В терминале выполните:

adb help

или просто

adb

— это выведет список основных команд и их описание.

* 1. Как подключиться к shell из adb?

Для доступа к командной оболочке Android-устройства используйте:

adb shell

Это откроет интерактивный терминал на устройстве.

* 1. Как сохранить скриншот и видеозапись экрана из adb?

Скриншот:

adb shell screencap /sdcard/screenshot.png

adb pull /sdcard/screenshot.png <путь\_на\_ПК>

Видеозапись экрана (начать запись):

adb shell screenrecord /sdcard/demo.mp4

Для остановки записи — нажмите Ctrl+C. Затем скопируйте файл:

adb pull /sdcard/demo.mp4 <путь\_на\_ПК>

* 1. Как выполнить обмен файлами между ПК и устройством на Android из adb?

Скопировать файл с ПК на устройство:

adb push <путь\_на\_ПК> <путь\_на\_устройстве>

Скопировать файл с устройства на ПК:

adb pull <путь\_на\_устройстве> <путь\_на\_ПК>

# Лабораторная работа №13

## Создание нового проекта

1. Цель работы
   1. Изучить процесс создания приложения в Android Studio.
2. Ответы на контрольные вопросы
   1. Что такое Jetpack Compose?

Jetpack Compose — это современный декларативный фреймворк от Google для создания пользовательских интерфейсов в Android-приложениях. Он позволяет описывать UI с помощью Kotlin-функций, упрощая разработку и делая код более читаемым и удобным для поддержки.

* 1. Какой тип проекта нужно выбрать, чтобы создать проект с использованием Jetpack Compose?

В Android Studio при создании нового проекта следует выбрать шаблон, поддерживающий Jetpack Compose, например:

«Empty Compose Activity» (Пустая Compose-активность)

Этот шаблон сразу настроен для работы с Compose.

* 1. Что такое Activity?

Activity — это один из основных компонентов Android-приложения, представляющий экран с пользовательским интерфейсом. Activity управляет жизненным циклом экрана и взаимодействием пользователя с приложением.

* 1. Для чего используется MainActivity?

MainActivity — это главная (стартовая) активность приложения, которая обычно запускается первой при старте приложения и отображает основной интерфейс.

* 1. Для чего используется функция onCreate?

onCreate — это метод жизненного цикла Activity, вызываемый при создании активности. В нём обычно выполняется инициализация компонентов, установка интерфейса и начальная настройка.

* 1. Что делает функция setContent?

setContent — функция, используемая в Jetpack Compose внутри onCreate, которая задаёт содержимое UI активности, принимая в качестве параметра композицию (Composable-функции), описывающую интерфейс.

* 1. Для чего используется функция @Composable?

Аннотация @Composable обозначает функции, которые описывают элементы пользовательского интерфейса в Jetpack Compose. Такие функции могут создавать и комбинировать UI-компоненты декларативно.

* 1. Для чего указывается @Preview?

Аннотация @Preview используется для отображения предварительного просмотра Composable-функции прямо в редакторе Android Studio без запуска приложения на устройстве или эмуляторе. Это ускоряет разработку интерфейса.

# Лабораторная работа №14

## Изучение и комментирование кода

1. Цель работы
   1. Изучить процесс документирования и комментирования приложения Android.
2. Ответы на контрольные вопросы
   1. Как оформляются комментарии в силе KDoc?

Комментарии в стиле KDoc оформляются с помощью многострочных комментариев, начинающихся с /\*\* и заканчивающихся \*/. Внутри комментария можно использовать специальные теги для описания кода.

* 1. Какая общая форма комментария в стиле KDoc?

Общая форма KDoc-комментария:

/\*\*

\* Краткое описание.

\*

\* Подробное описание (необязательно).

\*

\* @param имяПараметра описание параметра

\* @return описание возвращаемого значения

\* @property имяСвойства описание свойства

\*/

* 1. Как в комментарии KDoc описать свойства класса?

Для описания свойств класса используется тег @property, например:

/\*\*

\* Класс описывает пользователя.

\*

\* @property name имя пользователя

\* @property age возраст пользователя

\*/

class User(val name: String, val age: Int)

* 1. Как в комментарии KDoc описать параметры функции?

Для параметров функции используется тег @param, указывающий имя параметра и его описание:

/\*\*

\* Функция складывает два числа.

\*

\* @param a первое число

\* @param b второе число

\* @return сумма чисел

\*/

fun sum(a: Int, b: Int): Int = a + b

* 1. Как в комментарии KDoc описать возвращаемое значение функции?

Для описания возвращаемого значения используется тег @return, например:

/\*\*

\* Возвращает сумму двух чисел.

\*

\* @param a первое число

\* @param b второе число

\* @return сумма чисел

\*/

fun sum(a: Int, b: Int): Int = a + b

# Лабораторная работа №15

## Изменение элементов дизайна

1. Цель работы
   1. Изучить процесс изменения элементов дизайна приложения в Android Studio.
2. Ответы на контрольные вопросы
   1. Какой синтаксис у применения модификаторов?

В Jetpack Compose модификаторы применяются через параметр modifier, используя цепочку вызовов функций Modifier, например:

Modifier

.background(Color.Red)

.padding(16.dp)

.size(100.dp)

* 1. Как указать фон элемента?

Для задания фона используется модификатор .background(), принимающий цвет или Brush:

modifier = Modifier.background(Color.Blue)

* 1. Как указать размер элемента?

Для задания размера применяют модификаторы .size(), .width(), .height():

modifier = Modifier.size(100.dp) // ширина и высота 100 dp

modifier = Modifier.width(150.dp) // только ширина

modifier = Modifier.height(50.dp) // только высота

* 1. Как указать отступы элемента?

Для отступов используется модификатор .padding():

modifier = Modifier.padding(16.dp) // одинаковый отступ со всех сторон

modifier = Modifier.padding(start = 8.dp, top = 4.dp, end = 8.dp, bottom = 4.dp) // разные отступы

* 1. Как указать настройки текста элемента?

Для настройки текста в Compose используется параметр style у компонента Text, например:

Text(

text = "Пример",

style = TextStyle(

fontSize = 20.sp,

fontWeight = FontWeight.Bold,

color = Color.Black,

fontFamily = FontFamily.Serif

)

)

Или через готовые стили из MaterialTheme.typography.

* 1. Как добавить и использовать изображение из ресурсов?

Поместите изображение (например, PNG) в папку res/drawable.

Используйте компонент Image с функцией painterResource для загрузки:

import androidx.compose.ui.res.painterResource

Image(

painter = painterResource(id = R.drawable.my\_image),

contentDescription = "Описание изображения"

)

# Лабораторная работа №16

## Использование ресурсов

1. Цель работы
   1. Изучить процесс добавления и применения ресурсов при разработке мобильных приложений Android.
2. Ответы на контрольные вопросы
   1. Что хранится в папке res?

Папка res (resources) содержит все ресурсы приложения — неисполняемые файлы, используемые в приложении, такие как изображения, строки, стили, макеты, цвета и другие.

* 1. Что хранится в папке values?

В папке res/values хранятся XML-файлы с определениями ресурсов, таких как строки (strings.xml), цвета (colors.xml), стили (styles.xml), размеры (dimens.xml), массивы (arrays.xml) и другие константы.

* 1. Что хранится в папке drawable?

В папке res/drawable хранятся графические ресурсы — изображения в форматах PNG, JPG, XML (векторные изображения, селекторы и т.п.), которые используются для отображения в интерфейсе.

* 1. Для чего используется Resource Manager?

Resource Manager — это инструмент в Android Studio для удобного просмотра, управления и вставки ресурсов проекта (изображений, строк, стилей и пр.) прямо из IDE.

* 1. Как использовать строковые ресурсы?

Строковые ресурсы определяются в res/values/strings.xml и используются в коде через идентификатор:

В XML: @string/string\_name

В Kotlin (Compose):

val text = stringResource(id = R.string.string\_name)

Text(text = text)

* 1. Как использовать ресурсы изображения?

Изображения из res/drawable подключаются в Compose через painterResource:

Image(painter = painterResource(id = R.drawable.image\_name), contentDescription = "Описание")

* 1. Как использовать массивы из ресурсов?

Массивы определяются в res/values/arrays.xml и загружаются в коде через resources.getStringArray() или специальные функции Compose:

val items = stringArrayResource(id = R.array.array\_name)

# Лабораторная работа №17

## Настройка элементов управления для ввода данных

1. Цель работы
   1. Изучить процесс настройки и использования элементов управления для ввода данных приложения Android.
2. Ответы на контрольные вопросы
   1. Какие элементы позволяют ввести текстовые значения?

В Jetpack Compose для ввода текста используются:

TextField — однострочное текстовое поле

OutlinedTextField — текстовое поле с обводкой

BasicTextField — базовое текстовое поле без оформления

* 1. Как настроить тип клавиатуры?

Тип клавиатуры задаётся параметром keyboardOptions в TextField, например:

TextField(

value = text,

onValueChange = { text = it },

keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number)

)

Основные типы: KeyboardType.Text, Number, Email, Phone и др.

* 1. Как связать значение из элемента управления с переменной?

Используется состояние (state) — переменная с mutableStateOf и связывается с параметрами value и onValueChange:

var text by remember { mutableStateOf("") }

TextField(

value = text,

onValueChange = { text = it }

)

* 1. Для чего используется combobox, какие его основные настройки?

В Compose нет стандартного ComboBox, но аналог — выпадающий список (ExposedDropdownMenuBox), позволяющий выбрать одно значение из списка. Основные настройки: список элементов, выбранное значение, раскрытие меню, обработчики выбора.

* 1. Для чего используется radiobutton, какие его основные настройки?

RadioButton используется для выбора одного варианта из группы. Основные настройки:

selected — выбран ли элемент

onClick — обработчик нажатия

Группируют несколько RadioButton вместе, управляя состоянием выбора.

* 1. Для чего используется slider, какие его основные настройки?

Slider — ползунок для выбора значения из диапазона. Основные параметры:

value — текущее значение

onValueChange — обработчик изменения

valueRange — диапазон значений (по умолчанию 0f..1f)

steps — количество шагов между значениями

* 1. Для чего используется switch, какие его основные настройки?

Switch — переключатель включено/выключено. Основные параметры:

checked — текущее состояние (true/false)

onCheckedChange — обработчик переключения

Используется для двоичных настроек.

# Лабораторная работа №18

## Настройка кнопок

1. Цель работы
   1. Изучить процесс настройки и использования элементов управления кнопок в приложениях Android.
2. Ответы на контрольные вопросы
   1. Как объявить button?

Для создания кнопки используется компонент Button, где указывается действие при нажатии и содержимое (обычно текст):

Button(onClick = { /\* действие при нажатии \*/ }) {

Text("Нажми меня")

}

* 1. Что такое «IconButton»?

IconButton — это кнопка, которая содержит только иконку без текста. Используется для компактных кнопок, например, в тулбарах:

IconButton(onClick = { /\* действие \*/ }) {

Icon(Icons.Default.Favorite, contentDescription = "Избранное")

}

* 1. Что такое «IconToggleButton»?

IconToggleButton — кнопка-переключатель с иконкой, которая меняет своё состояние между включённым и выключенным:

var checked by remember { mutableStateOf(false) }

IconToggleButton(checked = checked, onCheckedChange = { checked = it }) {

Icon(Icons.Default.Favorite, contentDescription = "Избранное")

}

* 1. Что такое «FloatingActionButton»?

FloatingActionButton (FAB) — плавающая круглая кнопка для основных действий на экране, обычно с иконкой:

FloatingActionButton(onClick = { /\* действие \*/ }) {

Icon(Icons.Default.Add, contentDescription = "Добавить")

}

* 1. Что такое «ExtendedFloatingActionButton»?

ExtendedFloatingActionButton — расширенная версия FAB, которая содержит иконку и текст, позволяя более явно обозначить действие:

ExtendedFloatingActionButton(

text = { Text("Добавить") },

icon = { Icon(Icons.Default.Add, contentDescription = null) },

onClick = { /\* действие \*/ }

)

* 1. Какие параметры нужно указать при создании объекта Icon?

При создании Icon указываются:

imageVector или painter — источник иконки (например, Icons.Default.\*)

contentDescription — описание для доступности (важно для экранных читалок)

опционально tint — цвет иконки

# Лабораторная работа №19

## Отображение списков и таблиц

1. Цель работы
   1. Изучить процесс настройки и использования элементов управления для отображения списков в приложении Android.
2. Ответы на контрольные вопросы
   1. Что такое «LazyColumn»?

LazyColumn — это вертикальный список в Jetpack Compose, который эффективно отображает большое количество элементов, создавая только видимые на экране компоненты. Используется для прокручиваемых списков с вертикальной ориентацией:

LazyColumn {

items(itemsList) { item ->

Text(text = item)

}

}

* 1. Что такое «LazyRow»?

LazyRow — это горизонтальный прокручиваемый список, аналогичный LazyColumn, но с горизонтальной ориентацией. Позволяет отображать элементы в строку с возможностью прокрутки:

LazyRow {

items(itemsList) { item ->

Text(text = item)

}

}

* 1. Что такое «LazyVerticalGrid»?

LazyVerticalGrid — это вертикальная сетка с ленивой загрузкой элементов, где элементы располагаются в несколько колонок и прокручиваются вертикально. Позволяет создавать адаптивные сеточные интерфейсы:

LazyVerticalGrid(

cells = GridCells.Fixed(2) // 2 колонки

) {

items(itemsList) { item ->

Text(text = item)

}

}

* 1. Что такое «LazyHorizontalGrid»?

LazyHorizontalGrid — горизонтальная сетка с ленивой загрузкой, где элементы располагаются в несколько строк и прокручиваются по горизонтали. Используется для создания сеток с горизонтальной прокруткой:

LazyHorizontalGrid(

rows = GridCells.Fixed(2) // 2 строки

) {

items(itemsList) { item ->

Text(text = item)

}

}

# Лабораторная работа №20

## Обработка событий: переключение между экранами

1. Цель работы
   1. Изучить процесс настройки и реализации перехода между экранами в приложении Android
2. Ответы на контрольные вопросы
   1. Какую зависимость нужно добавить в Gradle для поддержки навигации?

Для использования навигации в Jetpack Compose добавьте в файл build.gradle (модульный) следующую зависимость:

implementation "androidx.navigation:navigation-compose:<версия>"

Где <версия> — актуальная версия библиотеки, например, "2.5.3" или новее.

Также убедитесь, что в проекте подключен Compose BOM или соответствующие Compose зависимости.

* 1. Для чего предназначен NavHost?

NavHost — это контейнер, который отображает текущий экран (композабл) в соответствии с состоянием навигации. Он связывает NavController с графом навигации и управляет переходами между экранами.

* 1. Для чего предназначен NavController?

NavController — объект, управляющий навигацией: он отвечает за переходы между экранами, управление стеком навигации, передачу параметров и возврат назад.

* 1. Что указывается в графе навигации?

В графе навигации (NavGraph) описываются все маршруты (destinations) приложения — экраны, их ключи (routes), параметры, а также возможные переходы между ними.

* 1. В каких случаях для организации маршрутов используется sealed class?

sealed class используется для типобезопасного и расширяемого описания маршрутов, когда нужно задать набор экранов с параметрами и логикой, обеспечивая проверку на этапе компиляции и удобную работу с навигацией в коде.

* 1. В каких случаях для организации маршрутов используется enum?

enum class может применяться для простого фиксированного набора экранов без параметров или с минимальной логикой, когда важна простота и ограниченный набор значений.

* 1. Как считать параметры, переданные при навигации?

Параметры, переданные в маршрут, считываются внутри экрана через NavBackStackEntry или с помощью вспомогательных функций, например:

val navBackStackEntry = navController.currentBackStackEntryAsState().value

val param = navBackStackEntry?.arguments?.getString("paramName")

Или при объявлении composable в NavHost:

composable("screen/{param}") { backStackEntry ->

val param = backStackEntry.arguments?.getString("param")

// использовать param

}

# Лабораторная работа №21

## Обработка событий: подсказки

1. Цель работы
   1. Изучить процесс настройки и использования элементов управления для отображения уведомлений и подсказок.
2. Ответы на контрольные вопросы
   1. Какой класс позволяет описать диалоговое окно?

В Jetpack Compose для создания диалогового окна используется функция-компонент AlertDialog.

* 1. Какая общая форма описания диалогового окна?

Общая форма AlertDialog включает параметры для заголовка, текста, кнопок и обработчика закрытия:

AlertDialog(

onDismissRequest = { /\* действие при закрытии диалога \*/ },

title = { Text("Заголовок") },

text = { Text("Текст диалога") },

confirmButton = {

Button(onClick = { /\* действие подтверждения \*/ }) {

Text("ОК")

}

},

dismissButton = {

Button(onClick = { /\* действие отмены \*/ }) {

Text("Отмена")

}

}

)

* 1. Какой класс позволяет описать настройки всплывающего сообщения?

Для настройки и отображения всплывающих сообщений используется класс Snackbar вместе с SnackbarHostState и SnackbarHost.

* 1. Для чего в диалоговых окнах используется функция onDismissRequest?

onDismissRequest — это лямбда, вызываемая при попытке закрыть диалог (например, при нажатии вне диалога или кнопкой назад). Ее используют для обработки логики закрытия и изменения состояния видимости диалога.

* 1. Для чего применяется объект SnackbarHost?

SnackbarHost — это контейнер, который отображает Snackbar на экране, управляя их показом и скрытием в соответствии с состоянием SnackbarHostState. Обычно он используется для централизованного показа всплывающих сообщений в приложении.

# Лабораторная работа №22

## Обработка событий: индикация

1. Цель работы
   1. Изучить процесс изменения интерфейса для информирования пользователей в приложениях Android.
2. Ответы на контрольные вопросы
   1. Какие классы индикаторов прогресса используются в приложениях Android?

В Android (в том числе в Jetpack Compose) используются следующие основные классы индикаторов прогресса:

В классическом Android View: ProgressBar (с возможностью установки стиля для кругового или линейного индикатора).

В Jetpack Compose:

CircularProgressIndicator — круговой индикатор прогресса.

LinearProgressIndicator — линейный индикатор прогресса.

* 1. Как указать шаг в индикаторах прогресса?

В Compose для индикаторов прогресса шаг (progress) указывается параметром progress как значение типа Float в диапазоне от 0.0 до 1.0. Например:

LinearProgressIndicator(progress = 0.5f)

* 1. Как указать задержку в индикаторах прогресса?

В стандартных компонентах индикаторов прогресса нет встроенного параметра задержки. Задержку можно реализовать программно, например, используя LaunchedEffect и delay из корутин, чтобы отложить показ индикатора.

* 1. Как настроить цвет в индикаторах прогресса?

Цвет индикатора задаётся параметрами:

Для CircularProgressIndicator и LinearProgressIndicator — параметр color.

Для вторичного цвета (фонового) — параметр trackColor.

Пример:

CircularProgressIndicator(

progress = 0.7f,

color = Color.Red,

trackColor = Color.LightGray

)

* 1. Для чего используется BadgedBox?

BadgedBox — это контейнер в Compose, который позволяет добавить бейдж (маленькую метку, например, с числом уведомлений) поверх другого компонента (например, иконки). Используется для отображения дополнительной информации, например, количества новых сообщений на иконке.

# Лабораторная работа №23

## Настройка тем

1. Цель работы
   1. Изучить процесс изменения интерфейса для информирования пользователей в приложениях Android.
2. Ответы на контрольные вопросы
   1. Что такое MaterialTheme?

MaterialTheme — это базовый компонент в Jetpack Compose, обеспечивающий единый стиль приложения согласно принципам Material Design. Он предоставляет доступ к цветовой палитре, типографике и формам (shapes), которые используются во всех элементах интерфейса для согласованного оформления.

* 1. Как настроить палитру?

Палитру настраивают, создавая объект Colors с нужными цветами (primary, secondary, background и др.) и передавая его в параметр colors функции MaterialTheme. Обычно это делается в отдельном файле темы, например:

private val LightColors = lightColors(

primary = Color(0xFF6200EE),

secondary = Color(0xFF03DAC6),

background = Color.White,

// другие цвета

)

MaterialTheme(

colors = LightColors,

typography = Typography,

shapes = Shapes,

) {

// UI

}

* 1. Как настроить типографику?

Типографику настраивают через объект Typography, где задаются стили текста (шрифты, размеры, веса, межстрочные интервалы и т.д.). Его передают в параметр typography MaterialTheme:

val Typography = Typography(

h1 = TextStyle(

fontWeight = FontWeight.Bold,

fontSize = 30.sp,

letterSpacing = 0.sp

),

body1 = TextStyle(

fontWeight = FontWeight.Normal,

fontSize = 16.sp

)

// другие стили

)

* 1. Как сменить тему на светлую/темную?

Для переключения между светлой и темной темами создают две палитры цветов — для светлой (lightColors) и темной (darkColors). В функции темы выбирают палитру в зависимости от параметра (например, isSystemInDarkTheme() или пользовательского флага):

@Composable

fun MyAppTheme(

darkTheme: Boolean = isSystemInDarkTheme(),

content: @Composable () -> Unit

) {

val colors = if (darkTheme) darkColors(

primary = Color(0xFFBB86FC),

background = Color.Black,

// ...

) else lightColors(

primary = Color(0xFF6200EE),

background = Color.White,

// ...

)

MaterialTheme(

colors = colors,

typography = Typography,

shapes = Shapes,

content = content

)

}

# Лабораторная работа №24

## Настройка анимации

1. Цель работы
   1. Изучить процесс настройки анимации интерфейса приложений Android.
2. Ответы на контрольные вопросы
   1. Для чего используется анимация в приложениях?

Анимация используется для улучшения пользовательского опыта, делая интерфейс более живым, понятным и отзывчивым. Она помогает визуально связывать действия пользователя с изменениями интерфейса, акцентировать внимание, облегчать восприятие переходов и состояний.

* 1. На какие виды делится анимация в Jetpack Compose?

В Jetpack Compose анимации делятся на:

Одноразовые анимации — анимации, которые запускаются один раз при изменении состояния (например, animate\*AsState).

Переходы состояний (Transition) — анимации, которые управляют плавными переходами между несколькими состояниями (updateTransition).

Бесконечные анимации (InfiniteTransition) — циклические анимации, которые повторяются бесконечно (rememberInfiniteTransition).

Анимации жестов и физические анимации — для взаимодействия с пользователем (например, анимация перетаскивания).

* 1. Для чего используется анимация Animate\*AsState?

animate\*AsState используется для анимации изменения отдельного значения (например, цвета, размера, прозрачности) при изменении связанного состояния. Она автоматически плавно интерполирует значение от текущего к новому.

* 1. Для чего используется анимация UpdateTransition?

updateTransition применяется для создания анимаций между несколькими состояниями интерфейса, позволяя одновременно анимировать несколько параметров, синхронизируя их переходы.

* 1. Для чего используется анимация InfiniteTransition?

rememberInfiniteTransition используется для создания бесконечных циклических анимаций, например, для индикаторов загрузки или пульсирующих эффектов, которые повторяются без остановки.

# Лабораторная работа №25

## Работа с файловой системой

1. Цель работы
   1. Изучить процесс работы с файлами в приложениях Android.
2. Ответы на контрольные вопросы
   1. Как добавить папку assets в проект?

В Android-проекте папка assets создаётся вручную:

В структуре проекта в app/src/main/ создайте папку assets (если её нет).

Поместите туда необходимые файлы (например, текстовые, JSON, шрифты и т.п.).

Gradle автоматически распознает эту папку и включит её в APK.

* 1. Как открыть файл из assets?

Для чтения файла из assets используйте AssetManager, доступный через Context:

val assetManager = context.assets

val inputStream = assetManager.open("filename.ext")

* 1. Как получить путь к папке приложения?

Путь к внутреннему каталогу приложения можно получить так:

val filesDir = context.filesDir.absolutePath

Для кэша:

val cacheDir = context.cacheDir.absolutePath

* 1. Как считать данные из файла?

Пример чтения текста из файла (например, из assets):

val inputStream = context.assets.open("filename.txt")

val text = inputStream.bufferedReader().use { it.readText() }

* 1. Как записать данные в файл?

Для записи в файл во внутреннее хранилище:

val fileOutputStream = context.openFileOutput("filename.txt", Context.MODE\_PRIVATE)

fileOutputStream.use {

it.write(data.toByteArray())

}

* 1. Как вызвать диалог выбора файла?

Для выбора файла обычно используют Intent с действием ACTION\_OPEN\_DOCUMENT:

val intent = Intent(Intent.ACTION\_OPEN\_DOCUMENT).apply {

addCategory(Intent.CATEGORY\_OPENABLE)

type = "\*/\*" // или нужный MIME тип

}

startActivityForResult(intent, REQUEST\_CODE)

В AndroidX Activity Result API — через registerForActivityResult.

* 1. Как вызвать диалог сохранения файла?

В Android нет стандартного диалога "Сохранить файл" как на десктопе. Для сохранения файла можно использовать Intent с ACTION\_CREATE\_DOCUMENT:

val intent = Intent(Intent.ACTION\_CREATE\_DOCUMENT).apply {

addCategory(Intent.CATEGORY\_OPENABLE)

type = "application/octet-stream" // нужный MIME тип

putExtra(Intent.EXTRA\_TITLE, "filename.txt")

}

startActivityForResult(intent, REQUEST\_CODE\_SAVE)

Пользователь выберет место и имя файла, затем можно получить Uri для записи.

# Лабораторная работа №26

## Подготовка стандартных модулей

1. Цель работы
   1. Изучить процесс создания стандартных модулей в приложении Android.
2. Ответы на контрольные вопросы
   1. Для чего используется модульный подход?

Модульный подход используется для разделения большого проекта на независимые, логически обособленные части (модули). Это улучшает читаемость, повторное использование кода, ускоряет сборку, облегчает тестирование и поддержку приложения.

* 1. Что такое data class и какие у него особенности?

data class в Kotlin — это класс, предназначенный для хранения данных. Особенности:

Автоматически генерируются методы equals(), hashCode(), toString(), copy() и componentN() функции.

Основная цель — удобное и лаконичное представление моделей данных.

Все свойства в первичном конструкторе автоматически становятся полями класса.

* 1. Что такое MVVM?

MVVM (Model-View-ViewModel) — архитектурный паттерн для разделения логики приложения:

Model — слой данных и бизнес-логики.

View — пользовательский интерфейс.

ViewModel — посредник между Model и View, содержит логику представления и состояние UI, обеспечивает реактивное обновление View при изменениях данных.

MVVM облегчает тестирование и поддержку кода, улучшает разделение ответственности.

# Лабораторная работа №27

## Передача данных между модулями

1. Цель работы
   1. Изучить процесс передачи данных между модулями в приложении Android.
2. Ответы на контрольные вопросы
   1. Как передать параметры между модулями?

Для передачи параметров между модулями в Android обычно используют интерфейсы, общие модели данных или навигационные компоненты, которые поддерживают аргументы. В случае модульной архитектуры можно:

Определить общие модели (DTO) в отдельном общем модуле.

Использовать навигацию с передачей параметров через Bundle или безопасные аргументы (Safe Args).

Передавать параметры через вызовы функций или через ViewModel, если модули связаны через DI (например, Hilt).

* 1. Как передать значение простого типа с использованием навигации?

При использовании Jetpack Navigation для передачи простых типов (Int, String и т.п.) в аргументах фрагментов или экранов:

В navigation graph указываете аргумент:

<argument

android:name="userId"

app:argType="integer" />

При навигации передаёте параметр:

val action = CurrentFragmentDirections.actionToNextFragment(userId = 123)

findNavController().navigate(action)

В целевом фрагменте получаете аргумент:

val args: NextFragmentArgs by navArgs()

val userId = args.userId

* 1. Как передать объект с использованием навигации?

Для передачи объекта через навигацию:

Объект должен быть Parcelable или Serializable.

В navigation graph указываете аргумент с типом:

<argument

android:name="user"

app:argType="com.example.User" />

Передаёте объект:

val action = CurrentFragmentDirections.actionToNextFragment(user = userObject)

findNavController().navigate(action)

Получаете в целевом фрагменте:

val args: NextFragmentArgs by navArgs()

val user = args.user

Если объект слишком большой или сложный, рекомендуется передавать только идентификатор и загружать данные в целевом модуле.

# Лабораторная работа №28

## Работа с БД

1. Цель работы
   1. Изучить процесс работы с БД в приложении Android.
2. Ответы на контрольные вопросы
   1. Как создать БД SQLite в приложении Android?

Для создания SQLite базы данных в Android обычно создают класс, наследующий SQLiteOpenHelper:

class MyDatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

override fun onCreate(db: SQLiteDatabase) {

val createTable = """

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

age INTEGER

)

""".trimIndent()

db.execSQL(createTable)

}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {

db.execSQL("DROP TABLE IF EXISTS users")

onCreate(db)

}

}

Затем создаёте экземпляр и вызываете методы для работы с БД.

* 1. Как получить список строк таблицы БД SQLite в приложении Android?

Для чтения данных используйте метод query или rawQuery:

val db = dbHelper.readableDatabase

val cursor = db.query("users", arrayOf("id", "name", "age"), null, null, null, null, null)

val users = mutableListOf<User>()

while (cursor.moveToNext()) {

val id = cursor.getInt(cursor.getColumnIndexOrThrow("id"))

val name = cursor.getString(cursor.getColumnIndexOrThrow("name"))

val age = cursor.getInt(cursor.getColumnIndexOrThrow("age"))

users.add(User(id, name, age))

}

cursor.close()

* 1. Как добавить новую строку в таблицу БД SQLite в приложении Android?

Для вставки используйте insert:

val db = dbHelper.writableDatabase

val values = ContentValues().apply {

put("name", "John")

put("age", 30)

}

val newRowId = db.insert("users", null, values)

* 1. Как изменить существующую строку в таблице БД SQLite в приложении Android?

Для обновления используйте update:

val db = dbHelper.writableDatabase

val values = ContentValues().apply {

put("age", 31)

}

val selection = "id = ?"

val selectionArgs = arrayOf(userId.toString())

val count = db.update("users", values, selection, selectionArgs)

* 1. Как добавить удалить строку из таблицы БД SQLite в приложении Android?

Для удаления используйте delete:

val db = dbHelper.writableDatabase

val selection = "id = ?"

val selectionArgs = arrayOf(userId.toString())

val deletedRows = db.delete("users", selection, selectionArgs)

# Лабораторная работа №29

## Работа с API

1. Цель работы
   1. Изучить процесс работы с API в приложениях Android.
2. Ответы на контрольные вопросы
   1. Что такое REST?

REST (Representational State Transfer) — архитектурный стиль для создания веб-сервисов, основанный на использовании стандартных HTTP-методов для взаимодействия с ресурсами, представленными в виде URL. RESTful API обеспечивает простоту, масштабируемость и независимость клиент-сервера.

* 1. Какие методы являются стандартными для REST?

Основные HTTP-методы в REST:

GET — получить ресурс или список ресурсов

POST — создать новый ресурс

PUT — обновить существующий ресурс полностью

PATCH — частично обновить ресурс

DELETE — удалить ресурс

* 1. Какой формат может быть у данных, возвращаемых при использовании REST?

Чаще всего данные возвращаются в формате:

JSON (JavaScript Object Notation) — наиболее популярный, легковесный и читаемый формат

XML — более громоздкий, но иногда используется

Также возможны другие форматы, например, YAML, CSV, но реже.

* 1. Для чего используется библиотека retrofit2?

Retrofit2 — это популярная библиотека для Android и Java, которая упрощает работу с REST API:

Позволяет легко описывать HTTP-запросы через интерфейсы и аннотации

Автоматически преобразует JSON/XML ответы в объекты Kotlin/Java с помощью конвертеров (например, Gson, Moshi)

Обрабатывает асинхронные вызовы и ошибки

Упрощает интеграцию с сетевыми сервисами

* 1. Как в манифесте Android разрешить использование интернет?

Для доступа к интернету нужно добавить разрешение в файл AndroidManifest.xml:

<uses-permission android:name="android.permission.INTERNET" />

# Лабораторная работа №30

## Тестирование и оптимизация мобильного приложения

1. Цель работы
   1. Изучить процесс тестирования и оптимизации приложений Android.
2. Ответы на контрольные вопросы
   1. Что такое модульное тестирование?

Модульное тестирование — это процесс проверки отдельных частей (модулей) кода, например функций или классов, чтобы убедиться, что они работают корректно независимо от остальной системы. Цель — выявить ошибки на раннем этапе и повысить качество кода.

* 1. В какой папке проекта хранятся модульные тесты?

В Android-проектах модульные тесты обычно располагаются в папке:

src/test/java/

(внутри соответствующего модуля), где test — папка для локальных unit-тестов, выполняемых на JVM.

* 1. Как добавить точку останова в приложение?

В Android Studio:

Откройте нужный файл с кодом.

Кликните слева от номера строки (в серой области), чтобы поставить красную точку — точку останова (breakpoint).

При запуске приложения в режиме отладки выполнение остановится на этой строке.

* 1. Как запустить приложение в режиме отладки?

В Android Studio:

Выберите устройство или эмулятор для запуска.

Нажмите кнопку «Debug» (значок жука) или используйте сочетание клавиш (обычно Shift + F9).

Приложение запустится с возможностью пошагового выполнения, просмотра переменных и работы с точками останова.