Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА №4

«Использование БД а Andoid приложениях»

ДИСЦИПЛИНА: «Разработка мобильного ПО»

Выполнил: студент гр. ИУК4-52Б _____ (_____Губин Е.В._____)
(Подпись) (Ф.И.О.)

Проверил: _____ (_____Прудяк П.Н._____)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2024 г.

Цель: формирование практических навыков разработки приложений с использованием СУБД SQLite, списков и файлов при разработке Android-приложений с несколькими Activity.

Задачи:
1. Научиться работать с СУБД SQLite.
2. Научиться сохранять результаты выполнения запросов к базе данных в списки, файлы и LogCat.
3. Понять особенности реализации Android-приложений с использованием списков и СУБД SQLite

Формулировка задания:

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Программа может быть реализована на языке высокого уровня Kotlin.

9. Ноутбук2: производитель, объем жесткого диска, наличие SSD, объем оперативной памяти, наличие Full HD разрешения экрана, время автономной работы

Листинг:

avg_group.kt:

```
package com.example.lw_4

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.ListView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class avg_group : AppCompatActivity() {
    private lateinit var laptops: MutableList<Laptop>
    private lateinit var prop: String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_avg_group)
```

```
        val prop = intent.getStringExtra("prop")
        if (prop != null) {
            this.prop = prop
        }
        this.laptops = intent.getParcelableArrayListExtra<Laptop>("laptops") as
ArrayList<Laptop>

        this.renderLaptopListViews()

        val mainButton = findViewById<Button>(R.id.mainButton)
        mainButton.setOnClickListener {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }
    }

    private fun renderLaptopListViews() {
        val laptopsListViews = findViewById<ListView>(R.id.listView)

        if (this.prop == "Производитель") {
            val laptopsItems: MutableList<TableAVGSomeItem> = mutableListOf()
            for (i in 0..this.laptops.size - 1) {
                laptopsItems.add(
                    TableAVGSomeItem(
                    "Производитель",
                    this.laptops[i].manufacturerName,
                    this.laptops[i].HDDVolume,
                    this.laptops[i].RAMVolume,
                    this.laptops[i].screenTime
                ))
            }
            val adapter = TableAVGSomeAdapter(this, laptopsItems)
            laptopsListViews.adapter = adapter
        }
        else if (this.prop == "Наличие SSD") {
            val laptopsItems: MutableList<TableAVGSomeItem> = mutableListOf()
            for (i in 0..this.laptops.size - 1) {
                laptopsItems.add(
                    TableAVGSomeItem(
                      "Наличие SSD",
                      this.laptops[i].SSDPresent.toString(),
                      this.laptops[i].HDDVolume,
```

```
                    this.laptops[i].RAMVolume,
                    this.laptops[i].screenTime
                ))
            }
            val adapter = TableAVGSomeAdapter(this, laptopsItems)
            laptopsListViews.adapter = adapter
        }
        else if (this.prop == "Наличие FULL HD") {
            val laptopsItems: MutableList<TableAVGSomeItem> = mutableListOf()
            for (i in 0..this.laptops.size - 1) {
                laptopsItems.add(
                    TableAVGSomeItem(
                        "Наличие FULL HD",
                        this.laptops[i].isFHD.toString(),
                        this.laptops[i].HDDVolume,
                        this.laptops[i].RAMVolume,
                        this.laptops[i].screenTime
                    ))
            }
            val adapter = TableAVGSomeAdapter(this, laptopsItems)
            laptopsListViews.adapter = adapter
        }
        else {
            val laptopsItems: MutableList<TableAVGItem> = mutableListOf()
            for (i in 0..this.laptops.size - 1) {
                laptopsItems.add(
                    TableAVGItem(
                        this.laptops[i].HDDVolume,
                        this.laptops[i].RAMVolume,
                        this.laptops[i].screenTime
                    )
                )
            }
            val adapter = TableAVGAdapter(this, laptopsItems)
            laptopsListViews.adapter = adapter
        }
    }
}
```

DBHelper.kt:

```
package com.example.lw_4

import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.widget.Toast
```

```
class DBHelper(context: Context, factory: SQLiteDatabase.CursorFactory?) :
    SQLiteOpenHelper(context, DATABASE_NAME, factory, DATABASE_VERSION) {

    override fun onCreate(db: SQLiteDatabase) {
        val query = ("CREATE TABLE " + TABLE_NAME + " ("
                + ID_COL + " INTEGER PRIMARY KEY, " +
                MANUFACTURE_NAME_COL + " VARCHAR(30) NOT NULL, " +
                HDD_VOLUME_COL + " INTEGER NOT NULL, " +
                SSD_PRESENT_COL + " BOOLEAN NOT NULL, " +
                RAM_VOLUME_COL + " INTEGER NOT NULL, " +
                IS_FHD_COL + " BOOLEAN NOT NULL, " +
                SCREEN_TIME + " INTEGER NOT NULL);")
        db.execSQL(query)
    }

    override fun onUpgrade(db: SQLiteDatabase, p1: Int, p2: Int) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME)
        onCreate(db)
    }

    fun getSortedList(prop: String, typeSort: String): MutableList<Laptop> {
        var col = ""
        if (prop == "Объём HDD") {
            col = HDD_VOLUME_COL
        }
        else if (prop == "Объём RAM") {
            col = RAM_VOLUME_COL
        }
        else {
            col = SCREEN_TIME
        }

        val db = this.readableDatabase
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME ORDER BY ${col}
$typeSort", null)
        if (!cursor.moveToFirst()) {
            cursor.close()
            return mutableListOf()
        }

        val laptops: MutableList<Laptop> = mutableListOf()

        var ID = 0
        var manufacturerName = ""
        var HDDVolume = 0
        var SSDPresent = 0
        var RAMVolume = 0
        var isFHD = 0
        var screenTime = 0
        do {
```

```
        var columnIndex = cursor.getColumnIndex(DBHelper.ID_COL)
        ID = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.MANUFACTURE_NAME_COL)
        manufacturerName = cursor.getString(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.HDD_VOLUME_COL)
        HDDVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SSD_PRESENT_COL)
        SSDPresent = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.RAM_VOLUME_COL)
        RAMVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.IS_FHD_COL)
        isFHD = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SCREEN_TIME)
        screenTime = cursor.getInt(columnIndex)
        laptops.add(Laptop(
            ID,
            manufacturerName,
            HDDVolume,
            if (SSDPresent == 1) true else false,
            RAMVolume,
            if (isFHD == 1) true else false, screenTime)
        )
    } while (cursor.moveToNext())

    cursor.close()

    return laptops
}

fun addLaptop(
    manufacturerName: String,
    HDDVolume: Int, SSDPresent: Boolean,
    RAMVolume: Int,
    isFHD: Boolean,
    screenTime: Int
){
    val values = ContentValues()
    values.put(MANUFACTURE_NAME_COL, manufacturerName)
    values.put(HDD_VOLUME_COL, HDDVolume)
    values.put(SSD_PRESENT_COL, SSDPresent)
    values.put(RAM_VOLUME_COL, RAMVolume)
    values.put(IS_FHD_COL, isFHD)
    values.put(SCREEN_TIME, screenTime)

    val db = this.writableDatabase

    db.insert(TABLE_NAME, null, values)

    db.close()
}
```

```
fun deleteLaptopById(id: Int) {
    val db = this.writableDatabase
    val whereClause = "id = ?"
    val whereArgs = arrayOf(id.toString())

    db.delete(TABLE_NAME, whereClause, whereArgs)

    db.close()
}

fun getLaptops(): MutableList<Laptop> {
    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT * FROM " + TABLE_NAME, null)
    if (!cursor.moveToFirst()) {
        cursor.close()
        return mutableListOf()
    }

    val laptops: MutableList<Laptop> = mutableListOf()

    var ID = 0
    var manufacturerName = ""
    var HDDVolume = 0
    var SSDPresent = 0
    var RAMVolume = 0
    var isFHD = 0
    var screenTime = 0
    do {
        var columnIndex = cursor.getColumnIndex(DBHelper.ID_COL)
        ID = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.MANUFACTURE_NAME_COL)
        manufacturerName = cursor.getString(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.HDD_VOLUME_COL)
        HDDVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SSD_PRESENT_COL)
        SSDPresent = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.RAM_VOLUME_COL)
        RAMVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.IS_FHD_COL)
        isFHD = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SCREEN_TIME)
        screenTime = cursor.getInt(columnIndex)
        laptops.add(Laptop(
            ID,
            manufacturerName,
            HDDVolume,
            if (SSDPresent == 1) true else false,
            RAMVolume,
            if (isFHD == 1) true else false, screenTime)
```

```
            )
        } while (cursor.moveToNext())

        cursor.close()

        return laptops
    }

    fun getIDs(): List<Int> {
        val db = this.readableDatabase
        val idList = mutableListOf<Int>()
        val cursor = db.rawQuery("SELECT id FROM " + TABLE_NAME, null)
        val index = cursor.getColumnIndex(ID_COL)
        if (cursor.moveToFirst()) {
            do {
                val ID = cursor.getInt(index)
                idList.add(ID)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return idList
    }

    fun getSum(prop: String): Int {
        var col = ""
        if (prop == "Объём HDD") {
            col = HDD_VOLUME_COL
        }
        else if (prop == "Объём RAM") {
            col = RAM_VOLUME_COL
        }
        else {
            col = SCREEN_TIME
        }

        val db = this.readableDatabase
        var total = 0
        val cursor = db.rawQuery("SELECT SUM(${col}) AS total FROM ${TABLE_NAME}",
null)
        if (cursor.moveToFirst()) {
            val index = cursor.getColumnIndex("total")
            total = cursor.getInt(index)
        }
        cursor.close()
        return total
    }

    fun doubleGroup(option1: String, option2: String): MutableList<Laptop> {
        var col1 = ""
```

```
    if (option1 == "Производитель") {
        col1 = MANUFACTURE_NAME_COL
    }
    else if (option1 == "Объём HDD") {
        col1 = HDD_VOLUME_COL
    }
    else if (option1 == "Наличие SSD") {
        col1 = SSD_PRESENT_COL
    }
    else if (option1 == "Объём RAM") {
        col1 = RAM_VOLUME_COL
    }
    else if (option1 == "Наличие FULL HD") {
        col1 = IS_FHD_COL
    }
    else {
        col1 = SCREEN_TIME
    }

    var col2 = ""
    if (option2 == "Производитель") {
        col2 = MANUFACTURE_NAME_COL
    }
    else if (option2 == "Объём HDD") {
        col2 = HDD_VOLUME_COL
    }
    else if (option2 == "Наличие SSD") {
        col2 = SSD_PRESENT_COL
    }
    else if (option2 == "Объём RAM") {
        col2 = RAM_VOLUME_COL
    }
    else if (option2 == "Наличие FULL HD") {
        col2 = IS_FHD_COL
    }
    else {
        col2 = SCREEN_TIME
    }

    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT ${col1}, ${col2}, COUNT(*) as CountLines FROM
${TABLE_NAME} GROUP BY ${col1}, ${col2}", null)

    if (!cursor.moveToFirst()) {
        cursor.close()
        return mutableListOf()
    }

    val laptops: MutableList<Laptop> = mutableListOf()
```

```
var columnIndex = 0
var laptop = Laptop()

do {
    laptop = Laptop()

    columnIndex = cursor.getColumnIndex(col1)
    if (col1 == MANUFACTURE_NAME_COL) {
        laptop.manufacturerName = cursor.getString(columnIndex)
    }
    else if (col1 == HDD_VOLUME_COL) {
        laptop.HDDVolume = cursor.getInt(columnIndex)
    }
    else if (col1 == SSD_PRESENT_COL) {
        val baf = cursor.getInt(columnIndex)
        laptop.SSDPresent = if (baf == 1) true else false
    }
    else if (col1 == RAM_VOLUME_COL) {
        laptop.RAMVolume = cursor.getInt(columnIndex)
    }
    else if (col1 == IS_FHD_COL) {
        val baf = cursor.getInt(columnIndex)
        laptop.isFHD = if (baf == 1) true else false
    }
    else {
        laptop.screenTime = cursor.getInt(columnIndex)
    }

    columnIndex = cursor.getColumnIndex(col2)
    if (col2 == MANUFACTURE_NAME_COL) {
        laptop.manufacturerName = cursor.getString(columnIndex)
    }
    else if (col2 == HDD_VOLUME_COL) {
        laptop.HDDVolume = cursor.getInt(columnIndex)
    }
    else if (col2 == SSD_PRESENT_COL) {
        val baf = cursor.getInt(columnIndex)
        laptop.SSDPresent = if (baf == 1) true else false
    }
    else if (col2 == RAM_VOLUME_COL) {
        laptop.RAMVolume = cursor.getInt(columnIndex)
    }
    else if (col2 == IS_FHD_COL) {
        val baf = cursor.getInt(columnIndex)
        laptop.isFHD = if (baf == 1) true else false
    }
    else {
        laptop.screenTime = cursor.getInt(columnIndex)
    }
```

```
            columnIndex = cursor.getColumnIndex("CountLines")

            laptop.count = cursor.getInt(columnIndex)
            laptops.add(laptop)
        } while (cursor.moveToNext())

        cursor.close()

        return laptops
    }

    fun groupBy(prop: String): MutableList<Laptop> {
        var col = ""
        val cols: MutableList<String> = mutableListOf(
            HDD_VOLUME_COL,
            RAM_VOLUME_COL,
            SCREEN_TIME
        )
        if (prop == "Производитель") {
            col = MANUFACTURE_NAME_COL
        }
        else if (prop == "Объём HDD") {
            col = HDD_VOLUME_COL
        }
        else if (prop == "Наличие SSD") {
            col = SSD_PRESENT_COL
        }
        else if (prop == "Объём RAM") {
            col = RAM_VOLUME_COL
        }
        else if (prop == "Наличие FULL HD") {
            col = IS_FHD_COL
        }
        else {
            col = SCREEN_TIME
        }

        val db = this.readableDatabase

        val cursor: Cursor

        if (cols.contains(col)) {
            cursor = db.rawQuery(
                "SELECT " +
                        "AVG(${cols[0]}) as ${cols[0]}, " +
                        "AVG(${cols[1]}) as ${cols[1]}, " +
                        "AVG(${cols[2]}) as ${cols[2]} " +
                        "FROM $TABLE_NAME " +
                        "GROUP BY $col",
                null)
```

```
    }
    else {
        cursor = db.rawQuery(
            "SELECT " +
                    "$col, " +
                    "AVG(${cols[0]}) as ${cols[0]}, " +
                    "AVG(${cols[1]}) as ${cols[1]}, " +
                    "AVG(${cols[2]}) as ${cols[2]} " +
                    "FROM $TABLE_NAME " +
                    "GROUP BY $col",
            null)
    }


    if (!cursor.moveToFirst()) {
        cursor.close()
        return mutableListOf()
    }

    val laptops: MutableList<Laptop> =  mutableListOf()

    var HDDVolume = 0
    var RAMVolume = 0
    var screenTime = 0

    var laptop: Laptop
    var columnIndex = 0
    if (cols.contains(col)) {
        do {
            columnIndex = cursor.getColumnIndex(HDD_VOLUME_COL)
            HDDVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(RAM_VOLUME_COL)
            RAMVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(SCREEN_TIME)
            screenTime = cursor.getInt(columnIndex)
            laptop = Laptop()
            laptop.HDDVolume = HDDVolume
            laptop.RAMVolume = RAMVolume
            laptop.screenTime = screenTime
            laptops.add(laptop)
        } while (cursor.moveToNext())
    }
    else {
        var someString = ""
        var someBoolean = 0
        var isBoolean = false
        if (col != MANUFACTURE_NAME_COL) {
            isBoolean = true
        }
        do {
```

```
            columnIndex = cursor.getColumnIndex(HDD_VOLUME_COL)
            HDDVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(RAM_VOLUME_COL)
            RAMVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(SCREEN_TIME)
            screenTime = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(col)
            if (isBoolean) {
                someBoolean = cursor.getInt(columnIndex)
            }
            else {
                someString = cursor.getString(columnIndex)
            }
            laptop = Laptop()
            laptop.HDDVolume = HDDVolume
            laptop.RAMVolume = RAMVolume
            laptop.screenTime = screenTime
            if (col == SSD_PRESENT_COL) {
                laptop.SSDPresent = if (someBoolean == 1) true else false
            }
            else if (col == IS_FHD_COL) {
                laptop.isFHD = if (someBoolean == 1) true else false
            }
            else {
                laptop.manufacturerName = someString
            }
            laptops.add(laptop)
        } while (cursor.moveToNext())
    }

    cursor.close()
    return laptops
}

fun laptopsWithMaxValue(prop: String): MutableList<Laptop> {
    var col = ""
    if (prop == "Объём HDD") {
        col = HDD_VOLUME_COL
    }
    else if (prop == "Объём RAM") {
        col = RAM_VOLUME_COL
    }
    else {
        col = SCREEN_TIME
    }

    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT * FROM ${TABLE_NAME} WHERE ${col} =
(SELECT MAX(${col}) FROM ${TABLE_NAME})", null)
    if (!cursor.moveToFirst()) {
```

```
        cursor.close()
        return mutableListOf()
    }

    val laptops: MutableList<Laptop> = mutableListOf()

    var ID = 0
    var manufacturerName = ""
    var HDDVolume = 0
    var SSDPresent = 0
    var RAMVolume = 0
    var isFHD = 0
    var screenTime = 0
    do {
        var columnIndex = cursor.getColumnIndex(DBHelper.ID_COL)
        ID = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.MANUFACTURE_NAME_COL)
        manufacturerName = cursor.getString(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.HDD_VOLUME_COL)
        HDDVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SSD_PRESENT_COL)
        SSDPresent = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.RAM_VOLUME_COL)
        RAMVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.IS_FHD_COL)
        isFHD = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SCREEN_TIME)
        screenTime = cursor.getInt(columnIndex)
        laptops.add(Laptop(
            ID,
            manufacturerName,
            HDDVolume,
            if (SSDPresent == 1) true else false,
            RAMVolume,
            if (isFHD == 1) true else false, screenTime)
        )
    } while (cursor.moveToNext())

    cursor.close()

    return laptops
}

fun laptopsValueGreaterThan(prop: String, value: Int): MutableList<Laptop> {
    var col = ""
    if (prop == "Объём HDD") {
        col = HDD_VOLUME_COL
    }
    else if (prop == "Объём RAM") {
        col = RAM_VOLUME_COL
```

```
        }
        else {
            col = SCREEN_TIME
        }

        val db = this.readableDatabase
        val cursor = db.rawQuery("SELECT * FROM ${TABLE_NAME} WHERE ${col} > $
{value}", null)
        if (!cursor.moveToFirst()) {
            cursor.close()
            return mutableListOf()
        }

        val laptops: MutableList<Laptop> = mutableListOf()

        var ID = 0
        var manufacturerName = ""
        var HDDVolume = 0
        var SSDPresent = 0
        var RAMVolume = 0
        var isFHD = 0
        var screenTime = 0
        do {
            var columnIndex = cursor.getColumnIndex(DBHelper.ID_COL)
            ID = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.MANUFACTURE_NAME_COL)
            manufacturerName = cursor.getString(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.HDD_VOLUME_COL)
            HDDVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.SSD_PRESENT_COL)
            SSDPresent = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.RAM_VOLUME_COL)
            RAMVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.IS_FHD_COL)
            isFHD = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.SCREEN_TIME)
            screenTime = cursor.getInt(columnIndex)
            laptops.add(Laptop(
                ID,
                manufacturerName,
                HDDVolume,
                if (SSDPresent == 1) true else false,
                RAMVolume,
                if (isFHD == 1) true else false, screenTime)
            )
        } while (cursor.moveToNext())

        cursor.close()

        return laptops
```

```
    }

    fun laptopsValueGreaterThanOne(prop: String, value: Int): Laptop {
        var col = ""
        if (prop == "Объём HDD") {
            col = HDD_VOLUME_COL
        }
        else if (prop == "Объём RAM") {
            col = RAM_VOLUME_COL
        }
        else {
            col = SCREEN_TIME
        }

        val db = this.readableDatabase
        val cursor = db.rawQuery("SELECT * FROM ${TABLE_NAME} WHERE ${col} > ${value}", null)
        if (!cursor.moveToFirst()) {
            cursor.close()
            return Laptop()
        }

        var ID = 0
        var manufacturerName = ""
        var HDDVolume = 0
        var SSDPresent = 0
        var RAMVolume = 0
        var isFHD = 0
        var screenTime = 0

        var columnIndex = cursor.getColumnIndex(DBHelper.ID_COL)
        ID = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.MANUFACTURE_NAME_COL)
        manufacturerName = cursor.getString(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.HDD_VOLUME_COL)
        HDDVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SSD_PRESENT_COL)
        SSDPresent = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.RAM_VOLUME_COL)
        RAMVolume = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.IS_FHD_COL)
        isFHD = cursor.getInt(columnIndex)
        columnIndex = cursor.getColumnIndex(DBHelper.SCREEN_TIME)
        screenTime = cursor.getInt(columnIndex)


        cursor.close()
        return Laptop(
            ID,
            manufacturerName,
```

```
            HDDVolume,
            if (SSDPresent == 1) true else false,
            RAMVolume,
            if (isFHD == 1) true else false, screenTime
        )
    }

    fun laptopsValueLowerAVG(prop: String): MutableList<Laptop> {
        var col = ""
        if (prop == "Объём HDD") {
            col = HDD_VOLUME_COL
        }
        else if (prop == "Объём RAM") {
            col = RAM_VOLUME_COL
        }
        else {
            col = SCREEN_TIME
        }

        val db = this.readableDatabase
        val cursor = db.rawQuery("SELECT * FROM ${TABLE_NAME} WHERE ${col} <
(SELECT AVG(${col}) FROM ${TABLE_NAME})", null)
        if (!cursor.moveToFirst()) {
            cursor.close()
            return mutableListOf()
        }

        val laptops: MutableList<Laptop> = mutableListOf()

        var ID = 0
        var manufacturerName = ""
        var HDDVolume = 0
        var SSDPresent = 0
        var RAMVolume = 0
        var isFHD = 0
        var screenTime = 0
        do {
            var columnIndex = cursor.getColumnIndex(DBHelper.ID_COL)
            ID = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.MANUFACTURE_NAME_COL)
            manufacturerName = cursor.getString(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.HDD_VOLUME_COL)
            HDDVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.SSD_PRESENT_COL)
            SSDPresent = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.RAM_VOLUME_COL)
            RAMVolume = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.IS_FHD_COL)
            isFHD = cursor.getInt(columnIndex)
            columnIndex = cursor.getColumnIndex(DBHelper.SCREEN_TIME)
```

```
            screenTime = cursor.getInt(columnIndex)
            laptops.add(Laptop(
                ID,
                manufacturerName,
                HDDVolume,
                if (SSDPresent == 1) true else false,
                RAMVolume,
                if (isFHD == 1) true else false, screenTime)
            )
        } while (cursor.moveToNext())

        cursor.close()

        return laptops
    }

    companion object{
        private val DATABASE_NAME = "dns"

        private val DATABASE_VERSION = 2

        val TABLE_NAME = "laptops"

        val ID_COL = "id"

        val MANUFACTURE_NAME_COL = "manufacturer_name"

        val HDD_VOLUME_COL = "HDD_volume"

        val SSD_PRESENT_COL = "SSD_present"

        val RAM_VOLUME_COL = "RAM_volume"

        val IS_FHD_COL = "is_FHD"

        val SCREEN_TIME = "screen_time"
    }
}
```

double_group.kt:

```
package com.example.lw_4

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.ListView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
```

```
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class double_group : AppCompatActivity() {
    private lateinit var laptops: MutableList<Laptop>
    private lateinit var col1: String
    private lateinit var col2: String

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

        this.laptops = intent.getParcelableArrayListExtra<Laptop>("laptops") as
ArrayList<Laptop>

        var buf: String? = intent.getStringExtra("col1")
        if (buf != null) {
            this.col1 = buf
        }

        buf = intent.getStringExtra("col2")
        if (buf != null) {
            this.col2 = buf
        }

        this.renderLaptopListViews()

        val mainButton = findViewById<Button>(R.id.mainButton)
        mainButton.setOnClickListener {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }
    }

    private fun renderLaptopListViews() {
        val laptopsListViews = findViewById<ListView>(R.id.laptopsListViews)
        val laptopsItems: MutableList<TableGroupItem> = mutableListOf()

        for (i in 0..this.laptops.size - 1) {
            var opt1 = ""
            if (this.col1 == "Производитель") {
```

```
            opt1 = laptops[i].manufacturerName
        }
        else if (col1 == "Объём HDD") {
            opt1 = laptops[i].HDDVolume.toString()
        }
        else if (col1 == "Наличие SSD") {
            opt1 = laptops[i].SSDPresent.toString()
        }
        else if (col1 == "Объём RAM") {
            opt1 = laptops[i].RAMVolume.toString()
        }
        else if (col1 == "Наличие FULL HD") {
            opt1 = laptops[i].isFHD.toString()
        }
        else {
            opt1 = laptops[i].screenTime.toString()
        }

        var opt2 = ""
        if (this.col2 == "Производитель") {
            opt2 = laptops[i].manufacturerName
        }
        else if (col2 == "Объём HDD") {
            opt2 = laptops[i].HDDVolume.toString()
        }
        else if (col2 == "Наличие SSD") {
            opt2 = laptops[i].SSDPresent.toString()
        }
        else if (col2 == "Объём RAM") {
            opt2 = laptops[i].RAMVolume.toString()
        }
        else if (col2 == "Наличие FULL HD") {
            opt2 = laptops[i].isFHD.toString()
        }
        else {
            opt2 = laptops[i].screenTime.toString()
        }


        laptopsItems.add(TableGroupItem(
            this.col1,
            this.col2,
            opt1,
            opt2,
            this.laptops[i].count
        ))
    }
    val adapter = TableGroupAdapter(this, laptopsItems)
    laptopsListViews.adapter = adapter
```

```
    }
}
```

Laptop.kt:

```
package com.example.lw_4

import android.os.Parcel
import android.os.Parcelable

class Laptop: Parcelable {
    var ID: Int = 0
    var manufacturerName: String = ""
    var HDDVolume: Int = 0
    var SSDPresent: Boolean = false
    var RAMVolume: Int = 0
    var isFHD: Boolean = false
    var screenTime: Int = 0
    var count = 0

    constructor(ID: Int, manufacturerName: String, HDDVolume: Int, SSDPresent: Boolean,
RAMVolume: Int, isFHD: Boolean, screenTime: Int) {
        this.ID = ID
        this.manufacturerName = manufacturerName
        this.HDDVolume = HDDVolume
        this.SSDPresent = SSDPresent
        this.RAMVolume = RAMVolume
        this.isFHD =isFHD
        this.screenTime = screenTime
    }

    constructor(ID: Int, manufacturerName: String, HDDVolume: Int, SSDPresent: Boolean,
RAMVolume: Int, isFHD: Boolean, screenTime: Int, count: Int) {
        this.ID = ID
        this.manufacturerName = manufacturerName
        this.HDDVolume = HDDVolume
        this.SSDPresent = SSDPresent
        this.RAMVolume = RAMVolume
        this.isFHD =isFHD
        this.screenTime = screenTime
        this.count = count
    }

    constructor() {}

    constructor(parcel: Parcel) : this(
        parcel.readInt(),
        parcel.readString() ?: "",
        parcel.readInt(),
        parcel.readByte() != 0.toByte(), // для Boolean
```

```
        parcel.readInt(),
        parcel.readByte() != 0.toByte(), // для Boolean
        parcel.readInt(),
        parcel.readInt()
    )

    override fun writeToParcel(parcel: Parcel, flags: Int) {
        parcel.writeInt(ID)
        parcel.writeString(manufacturerName)
        parcel.writeInt(HDDVolume)
        parcel.writeByte(if (SSDPresent) 1 else 0)
        parcel.writeInt(RAMVolume)
        parcel.writeByte(if (isFHD) 1 else 0)
        parcel.writeInt(screenTime)
        parcel.writeInt(count)
    }

    override fun describeContents(): Int {
        return 0
    }

    companion object CREATOR : Parcelable.Creator<Laptop> {
        override fun createFromParcel(parcel: Parcel): Laptop {
            return Laptop(parcel)
        }

        override fun newArray(size: Int): Array<Laptop?> {
            return arrayOfNulls(size)
        }
    }
}
```

lower_than_AVG:

```
package com.example.lw_4

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.ListView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class lower_than_AVG : AppCompatActivity() {
    private lateinit var laptops: MutableList<Laptop>

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

```
        enableEdgeToEdge()
        setContentView(R.layout.activity_lower_than_avg)

        this.laptops = intent.getParcelableArrayListExtra<Laptop>("laptops") as
ArrayList<Laptop>
        this.renderLaptopListViews()

        val mainButton = findViewById<Button>(R.id.mainButton)
        mainButton.setOnClickListener {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }
    }

    private fun renderLaptopListViews() {
        val laptopsListViews = findViewById<ListView>(R.id.laptopsListViews)
        val laptopsItems: MutableList<TableItem> = mutableListOf()

        for (i in 0..this.laptops.size - 1) {
            laptopsItems.add(TableItem(
                this.laptops[i].ID,
                this.laptops[i].manufacturerName,
                this.laptops[i].HDDVolume,
                this.laptops[i].SSDPresent,
                this.laptops[i].RAMVolume,
                this.laptops[i].isFHD,
                this.laptops[i].screenTime
            ))
        }
        val adapter = TableAdapter(this, laptopsItems)
        laptopsListViews.adapter = adapter
    }
}
```

MainActivity.kt:

```
package com.example.lw_4

import android.content.ContentValues
import android.content.Context
import android.content.Intent
import android.net.Uri
```

```
import android.os.Bundle
import android.os.Environment
import android.provider.MediaStore
import android.util.Log
import android.view.LayoutInflater
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.ListView
import android.widget.RadioButton
import android.widget.RadioGroup
import android.widget.Spinner
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import java.io.File
import java.io.OutputStream

class MainActivity : AppCompatActivity() {
    private lateinit var db: DBHelper
    private var laptops: MutableList<Laptop> = mutableListOf()

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

        this.loadData()
        this.renderLaptopListViews()

        val newLaptopButton = findViewById<Button>(R.id.newLaptopButton)
        newLaptopButton.setOnClickListener {
            this.newLaptop()
        }

         val deleteLaptopButton = findViewById<Button>(R.id.deleteLaptopButton)
        deleteLaptopButton.setOnClickListener {
            this.deleteLaptop()
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
           v.setPadding(systemBars.left, systemBars.top, systemBars.right,
```

```
systemBars.bottom)
            insets
        }
    }

    private fun newLaptop() {
        val intent = Intent(this, NewLaptop::class.java)
        startActivity(intent)
    }

    private fun deleteLaptop() {
        val inflater = LayoutInflater.from(this)
        val dialogView = inflater.inflate(R.layout.delete_laptop, null)
        val dialogBuilder = AlertDialog.Builder(this)
            .setTitle("Введите ID удаляемого ноутбука")
            .setView(dialogView)
            .setPositiveButton("OK") { dialog, which ->
                val deletableID = dialogView.findViewById<EditText>(R.id.deletableID)
                val ID = deletableID.text.toString()
                if (!Regex("^[0-9]+$").matches(ID)) {
                    Toast.makeText(this, "Неккоректный ввод", Toast.LENGTH_SHORT).show()
                }
                else {
                    val intID = ID.toInt()
                    if (this.db.getIDs().contains(intID)) {
                        this.db.deleteLaptopById(intID)
                        this.refreshLocalData()
                        this.renderLaptopListViews()
                    }
                    else {
                        Toast.makeText(
                            this,
                            "Такого ноутбука не существует",
                            Toast.LENGTH_SHORT
                        ).show()
                    }
                }
                dialog.dismiss()
            }
            .setNegativeButton("Отмена") { dialog, which ->
                dialog.cancel()
            }
        dialogBuilder.show()
    }

    private fun refreshLocalData() {
        this.laptops = mutableListOf()
        this.loadData()
    }
```

```
    private fun renderLaptopListViews() {
        val laptopsListViews = findViewById<ListView>(R.id.laptopsListViews)
        val laptopsItems: MutableList<TableItem> = mutableListOf()

        for (i in 0..this.laptops.size - 1) {
            laptopsItems.add(TableItem(
                this.laptops[i].ID,
                this.laptops[i].manufacturerName,
                this.laptops[i].HDDVolume,
                this.laptops[i].SSDPresent,
                this.laptops[i].RAMVolume,
                this.laptops[i].isFHD,
                this.laptops[i].screenTime
            ))
        }
        val adapter = TableAdapter(this, laptopsItems)
        laptopsListViews.adapter = adapter
    }

    private fun loadData() {
        this.db = DBHelper(this, null)
        this.laptops = this.db.getLaptops()
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.option_menu, menu)
        return true
    }

    private fun writeSortedLaptopList(prop: String, typeSort: String, sortedList:
MutableList<Laptop>) {
        val filename = "sort.txt"
        val documentsDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENT
S)
        val file = File(documentsDir, filename)
        if (file.exists()) {
            file.delete()
        }

        val values = ContentValues().apply {
             put(MediaStore.MediaColumns.DISPLAY_NAME, filename)
            put(MediaStore.MediaColumns.MIME_TYPE, "text/plain")
            put(MediaStore.MediaColumns.RELATIVE_PATH, "Documents")
        }

         val uri: Uri? = contentResolver.insert(MediaStore.Files.getContentUri("external"),
values)

        uri?.let { uriValue ->
```

```
            val outputStream: OutputStream? = contentResolver.openOutputStream(uriValue)
            outputStream?.use { stream ->
                stream.write("Сортировка (${prop}) по ${if (typeSort == "ASC") "возрастанию"
else "убыванию"} объёма HDD.\n".toByteArray())
                stream.write("-----------------------------------------------\n".toByteArray())
                for (laptop in sortedList) {
                    stream.write("ID:                    \t${laptop.ID}\n".toByteArray())
                    stream.write("Производитель:         \t${laptop.manufacturerName}
\n".toByteArray())
                    stream.write("Объем HDD:             \t${laptop.HDDVolume}
ГБ\n".toByteArray())
                    stream.write("Наличие SSD:           \t${if (laptop.SSDPresent) "Да" else "нет"}
\n".toByteArray())
                    stream.write("Объём оперативной памяти:\t${laptop.RAMVolume}
ГБ\n".toByteArray())
                    stream.write("Наличие FHD:           \t${if (laptop.isFHD) "Да" else "Нет"}
\n".toByteArray())
                    stream.write("Время автономной работы: \t${laptop.screenTime}
часов\n".toByteArray())
                    stream.write("-----------------------------------------------\n".toByteArray())
                }
                stream.flush()
            }
        }
    }

    private fun writeAVGGroupBy(prop: String, laptops: MutableList<Laptop>) {
        val filename = "AVGGroupBy.txt"
        val documentsDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENT
S)
        val file = File(documentsDir, filename)
        if (file.exists()) {
            file.delete()
        }

        val values = ContentValues().apply {
            put(MediaStore.MediaColumns.DISPLAY_NAME, filename)
            put(MediaStore.MediaColumns.MIME_TYPE, "text/plain")
            put(MediaStore.MediaColumns.RELATIVE_PATH, "Documents")
        }

        val uri: Uri? = contentResolver.insert(MediaStore.Files.getContentUri("external"),
values)

        uri?.let { uriValue ->
            val outputStream: OutputStream? = contentResolver.openOutputStream(uriValue)
            outputStream?.use { stream ->
                stream.write("Средние значения (группировка ${prop})\n".toByteArray())
                stream.write("-----------------------------------------------\n".toByteArray())
```

```
            if (prop == "Производитель") {
                for (laptop in laptops) {
                    stream.write("Производитель:          \t${laptop.manufacturerName}
\n".toByteArray())
                    stream.write("Объем HDD:              \t${laptop.HDDVolume}
ГБ\n".toByteArray())
                    stream.write("Объём оперативной памяти:\t${laptop.RAMVolume}
ГБ\n".toByteArray())
                    stream.write("Время автономной работы: \t${laptop.screenTime}
часов\n".toByteArray())
                    stream.write("-------------------------------------------------\n".toByteArray())
                }
            }
            else if (prop == "Наличие SSD") {
                for (laptop in laptops) {
                    stream.write("Наличие SSD:            \t${if (laptop.SSDPresent) "Да" else
"Нет"}\n".toByteArray())
                    stream.write("Объем HDD:              \t${laptop.HDDVolume}
ГБ\n".toByteArray())
                    stream.write("Объём оперативной памяти:\t${laptop.RAMVolume}
ГБ\n".toByteArray())
                    stream.write("Время автономной работы: \t${laptop.screenTime}
часов\n".toByteArray())
                    stream.write("-------------------------------------------------\n".toByteArray())
                }
            }
            else if (prop == "Наличие FULL HD") {
                for (laptop in laptops) {
                    stream.write("Наличие FULL HD:        \t${if (laptop.isFHD) "Да" else "Нет"}
\n".toByteArray())
                    stream.write("Объем HDD:              \t${laptop.HDDVolume}
ГБ\n".toByteArray())
                    stream.write("Объём оперативной памяти:\t${laptop.RAMVolume}
ГБ\n".toByteArray())
                    stream.write("Время автономной работы: \t${laptop.screenTime}
часов\n".toByteArray())
                    stream.write("-------------------------------------------------\n".toByteArray())
                }
            }
            else {
                for (laptop in laptops) {
                    stream.write("Объем HDD:              \t${laptop.HDDVolume}
ГБ\n".toByteArray())
                    stream.write("Объём оперативной памяти:\t${laptop.RAMVolume}
ГБ\n".toByteArray())
                    stream.write("Время автономной работы: \t${laptop.screenTime}
часов\n".toByteArray())
                    stream.write("-------------------------------------------------\n".toByteArray())
                }
            }
```

```
            stream.flush()
        }
    }
}

private fun writeSum(sum: Int, prop: String) {
    val filename = "sum.txt"
    val documentsDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENT
S)
    val file = File(documentsDir, filename)
    if (file.exists()) {
        file.delete()
    }

    val values = ContentValues().apply {
        put(MediaStore.MediaColumns.DISPLAY_NAME, filename)
        put(MediaStore.MediaColumns.MIME_TYPE, "text/plain")
        put(MediaStore.MediaColumns.RELATIVE_PATH, "Documents")
    }

    val uri: Uri? = contentResolver.insert(MediaStore.Files.getContentUri("external"),
values)

    uri?.let { uriValue ->
        val outputStream: OutputStream? = contentResolver.openOutputStream(uriValue)
        outputStream?.use { stream ->
            stream.write("Сумма значений (${prop}) = ${sum}".toByteArray())
            stream.flush()
        }
    }
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.sort -> {
            val dialogView: View = LayoutInflater.from(this).inflate(R.layout.sort_option, null)
            val radioGroup = dialogView.findViewById<RadioGroup>(R.id.sortRadioGroup)

            val spinner = dialogView.findViewById<Spinner>(R.id.prop)
            val props = arrayOf("Объём HDD", "Объём RAM", "Время автономной
работы")
            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
            spinner.adapter = adapter

            val builder = AlertDialog.Builder(this)
            builder.setTitle("Выберите тип сортировки")
                .setView(dialogView)
```

```
                .setPositiveButton("OK") { _, _ ->
                    val selectedId = radioGroup.checkedRadioButtonId
                  if (selectedId == -1) {
                     Toast.makeText(this, "Выберите тип сортировки",
Toast.LENGTH_SHORT).show()
                  }
                  else {
                     val prop = spinner.selectedItem.toString()
                     val typeSort =
dialogView.findViewById<RadioButton>(selectedId).text.toString()

                     val sortedList: MutableList<Laptop> = this.db.getSortedList(prop,
typeSort)
                     this.writeSortedLaptopList(prop, typeSort, sortedList)
                     Log.i("MainActivity", "Список записан в файл sort.txt")
                     Toast.makeText(this, "Список записан в файл sort.txt",
Toast.LENGTH_SHORT).show()
                  }
               }
                .setNegativeButton("Cancel") { dialog, _ -> dialog.cancel() }

            builder.create().show()
            true
         }

         R.id.double_group -> {
            val dialogView: View =
LayoutInflater.from(this).inflate(R.layout.double_group_option, null)
            val spinner1 = dialogView.findViewById<Spinner>(R.id.prop1)
            val spinner2 = dialogView.findViewById<Spinner>(R.id.prop2)
            val props = arrayOf(
               "Производитель",
               "Объём HDD",
               "Наличие SSD",
               "Объём RAM",
               "Наличие FULL HD",
               "Время автономной работы"
            )
            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
            spinner1.adapter = adapter
            spinner2.adapter = adapter

            val builder = AlertDialog.Builder(this)
            builder.setTitle("Выберите тип сортировки")
               .setView(dialogView)
               .setPositiveButton("OK") { _, _ ->
                  val option1 = spinner1.selectedItem.toString()
                  val option2 = spinner2.selectedItem.toString()
```

```
                if (option1 != option2) {
                    val laptopsDoubleGroup = this.db.doubleGroup(option1, option2)
                    val intent = Intent(this, double_group::class.java)
                    intent.putParcelableArrayListExtra(
                        "laptops",
                        ArrayList(laptopsDoubleGroup)
                    )
                    intent.putExtra("col1", option1)
                    intent.putExtra("col2", option2)
                    startActivity(intent)
                }
                else {
                    Toast.makeText(this, "Выберите разные категории",
Toast.LENGTH_SHORT).show()
                }
            }
              .setNegativeButton("Cancel") { dialog, _ -> dialog.cancel() }

            builder.create().show()
            true
        }

        R.id.RAM_sum -> {
            val dialogView = LayoutInflater.from(this).inflate(R.layout.sum_option, null)
            val spinner = dialogView.findViewById<Spinner>(R.id.prop)

            val props = arrayOf("Объём HDD", "Объём RAM", "Время автономной
работы")

            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

            spinner.adapter = adapter

            val dialogBuilder = AlertDialog.Builder(this)
            dialogBuilder.setTitle("Выберите поле")
            dialogBuilder.setView(dialogView)

            dialogBuilder.setPositiveButton("OK") { dialog, _ ->
                val prop = spinner.selectedItem.toString()
                val sum = this.db.getSum(prop)
                this.writeSum(sum, prop)
                Log.i("MainActivity", "Вычеслнена сумма значений ${prop} = ${sum}")
                Toast.makeText(this, "Сумма записана в файл sum.txt",
Toast.LENGTH_SHORT).show()
            }

             dialogBuilder.setNegativeButton("Cancel") { dialog, _ ->
                dialog.cancel()
```

```
            }

             dialogBuilder.create().show()
            true
        }

        R.id.AVG_group_option -> {
            val dialogView = LayoutInflater.from(this).inflate(R.layout.avg_group_option, null)
            val spinner = dialogView.findViewById<Spinner>(R.id.prop)

            val props = arrayOf(
                "Производитель",
                "Объём HDD",
                "Наличие SSD",
                "Объём RAM",
                "Наличие FULL HD",
                "Время автономной работы"
            )

            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

            spinner.adapter = adapter

            val dialogBuilder = AlertDialog.Builder(this)
            dialogBuilder.setTitle("Группировать по")
            dialogBuilder.setView(dialogView)

            dialogBuilder.setPositiveButton("OK") { dialog, _ ->
                 val prop = spinner.selectedItem.toString()
                val groupedList: MutableList<Laptop> = this.db.groupBy(prop)
                this.writeAVGGroupBy(prop, groupedList)
                Toast.makeText(
                    this,
                    "Средние значение сгруппированных по полям выведены в файл
AVGGroupBy.txt",
                    Toast.LENGTH_SHORT).show()
                Log.i("MainActivity", "Средние значение сгруппированных по полям
выведены в файл AVGGroupBy.txt")
                val intent = Intent(this, avg_group::class.java)
                intent.putExtra("prop", prop)
                intent.putParcelableArrayListExtra("laptops", ArrayList(groupedList))
                startActivity(intent)
            }

             dialogBuilder.setNegativeButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }
```

```
            dialogBuilder.create().show()
            true
        }

        R.id.laptop_max_value -> {
            val dialogView =
LayoutInflater.from(this).inflate(R.layout.laptop_max_value_option, null)
            val spinner = dialogView.findViewById<Spinner>(R.id.prop)

            val props = arrayOf(
                "Объём HDD",
                "Объём RAM",
                "Время автономной работы"
            )

            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

            spinner.adapter = adapter

            val dialogBuilder = AlertDialog.Builder(this)
            dialogBuilder.setTitle("Поле")
            dialogBuilder.setView(dialogView)

            dialogBuilder.setPositiveButton("OK") { dialog, _ ->
                val prop = spinner.selectedItem.toString()
                val laptopsWithMaxValues: MutableList<Laptop> =
this.db.laptopsWithMaxValue(prop)
                var out = "-------------------------------------\n"
                for (laptop in laptopsWithMaxValues) {
                    out += "ID: ${laptop.ID}\n" +
                            "HDD volume: ${laptop.HDDVolume}\n" +
                            "SSD present: ${laptop.SSDPresent}\n" +
                            "RAM volume: ${laptop.RAMVolume}\n" +
                            "Is FHD: ${laptop.isFHD}\n" +
                            "Screen time: ${laptop.screenTime}\n" +
                            "-------------------------------------\n"
                }
                Log.i("MainActivity", "Максимальное значение по ${prop}\n" + out)
            }

            dialogBuilder.setNegativeButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }

            dialogBuilder.create().show()
            true
        }
```

```
        R.id.value_greater_than -> {
            val dialogView =
LayoutInflater.from(this).inflate(R.layout.value_greater_than_option, null)
            val spinner = dialogView.findViewById<Spinner>(R.id.prop)
            val valueMax = dialogView.findViewById<EditText>(R.id.valueMax)

            val props = arrayOf(
                "Объём HDD",
                "Объём RAM",
                "Время автономной работы"
            )

            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

            spinner.adapter = adapter

            val dialogBuilder = AlertDialog.Builder(this)
            dialogBuilder.setTitle("Поле и значение")
            dialogBuilder.setView(dialogView)

            dialogBuilder.setPositiveButton("OK") { dialog, _ ->
                val prop = spinner.selectedItem.toString()
                if (Regex("^[0-9]+$").matches(valueMax.text.toString())) {
                    val laptopsValueGreaterThan: MutableList<Laptop> =
this.db.laptopsValueGreaterThan(prop, valueMax.text.toString().toInt())
                    val intent = Intent(this, value_greater_than::class.java)
                    intent.putParcelableArrayListExtra("laptops",
ArrayList(laptopsValueGreaterThan))
                    startActivity(intent)
                    var out = "-----------------------------------\n"
                    for (laptop in laptopsValueGreaterThan) {
                        out += "ID: ${laptop.ID}\n" +
                                "HDD volume: ${laptop.HDDVolume}\n" +
                                "SSD present: ${laptop.SSDPresent}\n" +
                                "RAM volume: ${laptop.RAMVolume}\n" +
                                "Is FHD: ${laptop.isFHD}\n" +
                                "Screen time: ${laptop.screenTime}\n" +
                                "-----------------------------------\n"
                    }
                    Log.i("MainActivity", "Ноутбуки, где ${prop} > ${valueMax.text}\n" + out)
                }
            }

             dialogBuilder.setNegativeButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }

             dialogBuilder.create().show()
```

```
            true
        }

        R.id.lower_than_AVG -> {
            val dialogView =
LayoutInflater.from(this).inflate(R.layout.lower_than_avg_option, null)
            val spinner = dialogView.findViewById<Spinner>(R.id.prop)

            val props = arrayOf(
                "Объём HDD",
                "Объём RAM",
                "Время автономной работы"
            )

            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

            spinner.adapter = adapter

            val dialogBuilder = AlertDialog.Builder(this)
            dialogBuilder.setTitle("Поле и значение")
            dialogBuilder.setView(dialogView)

            dialogBuilder.setPositiveButton("OK") { dialog, _ ->
                val prop = spinner.selectedItem.toString()
                val laptopsValueLowerAVG: MutableList<Laptop> =
this.db.laptopsValueLowerAVG(prop)


                val intent = Intent(this, lower_than_AVG::class.java)
                intent.putParcelableArrayListExtra("laptops",
ArrayList(laptopsValueLowerAVG))
                startActivity(intent)
                var out = "-----------------------------------\n"
                for (laptop in laptopsValueLowerAVG) {
                    out += "ID: ${laptop.ID}\n" +
                            "HDD volume: ${laptop.HDDVolume}\n" +
                            "SSD present: ${laptop.SSDPresent}\n" +
                            "RAM volume: ${laptop.RAMVolume}\n" +
                            "Is FHD: ${laptop.isFHD}\n" +
                            "Screen time: ${laptop.screenTime}\n" +
                            "-----------------------------------\n"
                }
                Log.i("MainActivity", "Ноутбуки, где значение ${prop} < среднего\n" + out)
            }

             dialogBuilder.setNegativeButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }
```

```
            dialogBuilder.create().show()
            true
        }

        R.id.lower_than_AVG_one -> {
            val dialogView =
LayoutInflater.from(this).inflate(R.layout.value_greater_than_option, null)
            val spinner = dialogView.findViewById<Spinner>(R.id.prop)
            val valueMax = dialogView.findViewById<EditText>(R.id.valueMax)

            val props = arrayOf(
                "Объём HDD",
                "Объём RAM",
                "Время автономной работы"
            )

            val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, props)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

            spinner.adapter = adapter

            val dialogBuilder = AlertDialog.Builder(this)
            dialogBuilder.setTitle("Поле и значение")
            dialogBuilder.setView(dialogView)

            dialogBuilder.setPositiveButton("OK") { dialog, _ ->
                val prop = spinner.selectedItem.toString()
                if (Regex("^[0-9]+$").matches(valueMax.text.toString())) {
                    val laptopGreterThan: Laptop = this.db.laptopsValueGreaterThanOne(prop,
valueMax.text.toString().toInt())
                    var out = "ID: ${laptopGreterThan.ID}\n" +
                            "HDD volume: ${laptopGreterThan.HDDVolume}\n" +
                            "SSD present: ${laptopGreterThan.SSDPresent}\n" +
                            "RAM volume: ${laptopGreterThan.RAMVolume}\n" +
                            "Is FHD: ${laptopGreterThan.isFHD}\n" +
                            "Screen time: ${laptopGreterThan.screenTime}\n" +
                            "-----------------------------------\n"
                    Log.i("MainActivity", "Ноутбук (1), где ${prop} > ${valueMax.text}\n" + out)
                }
            }

            dialogBuilder.setNegativeButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }

            dialogBuilder.create().show()
            true
        }
```

```
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

NewLaptop.kt:

```
package com.example.lw_4

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.CheckBox
import android.widget.EditText
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class NewLaptop : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_new_laptop)

        val newLaptopButton = findViewById<Button>(R.id.newLaptopButton)
        newLaptopButton.setOnClickListener {
            val manufacturerName =
findViewById<EditText>(R.id.manufacturerName).text.toString()
            val HDDVolume = findViewById<EditText>(R.id.HDDVolume).text.toString()
            val SSDPresent = findViewById<CheckBox>(R.id.SSDPresent).isChecked
            val RAMVolume = findViewById<EditText>(R.id.RAMVolume).text.toString()
            val isFHD = findViewById<CheckBox>(R.id.isFHD).isChecked
            val screenTime = findViewById<EditText>(R.id.screenTime).text.toString()

            if (manufacturerName == "") {
                Toast.makeText(this, "Введите производителя",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            if (!Regex("^[0-9]+$").matches(HDDVolume)) {
                Toast.makeText(this, "Некорректный ввод объёма жесткого диска",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            if (!Regex("^[0-9]+$").matches(RAMVolume)) {
                Toast.makeText(this, "Некорректный ввод ОП",
Toast.LENGTH_SHORT).show()
```

```
            return@setOnClickListener
        }
        if (!Regex("^[0-9]+$").matches(screenTime)) {
            Toast.makeText(this, "Некорректный ввод времени автономной работы",
Toast.LENGTH_SHORT).show()
            return@setOnClickListener
        }

        val db = DBHelper(this, null)
        db.addLaptop(manufacturerName, HDDVolume.toInt(), SSDPresent,
RAMVolume.toInt(), isFHD, screenTime.toInt())

        val intent = Intent(this, MainActivity::class.java)
        startActivity(intent)
    }

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
        val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
        insets
    }
  }
}
```

TableAdapter.kt:

```
package com.example.lw_4

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.TextView
import android.widget.Toast

class TableAdapter(private val context: Context, private val items: List<TableItem>) :
  ArrayAdapter<TableItem>(context, R.layout.list_item_table, items) {

  override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
    val item = getItem(position)!!
    val view = convertView ?: LayoutInflater.from(context).inflate(R.layout.list_item_table,
parent, false)

    val val1: TextView = view.findViewById(R.id.val1)
    val val2: TextView = view.findViewById(R.id.val2)
    val val3: TextView = view.findViewById(R.id.val3)
    val val4: TextView = view.findViewById(R.id.val4)
```

```
        val val5: TextView = view.findViewById(R.id.val5)
        val val6: TextView = view.findViewById(R.id.val6)
        val val7: TextView = view.findViewById(R.id.val7)

        val1.text = item.ID.toString()
        val2.text = item.manufacturerName
        val3.text = item.HDDVolume.toString()
        val4.text = if (item.SSDPresent) "Да" else "Нет"
        val5.text = item.RAMVolume.toString()
        val6.text = if (item.isFHD) "Да" else "Нет"
        val7.text = item.screenTime.toString()

        return view
    }
}
```

TableAVGAdapter.kt:

```
package com.example.lw_4

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.TextView

class TableAVGAdapter(private val context: Context, private val items:
List<TableAVGItem>) :
    ArrayAdapter<TableAVGItem>(context, R.layout.list_item_avg_table, items) {

    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
        val item = getItem(position)!!
        val view = convertView ?:
LayoutInflater.from(context).inflate(R.layout.list_item_avg_table, parent, false)

        val val1: TextView = view.findViewById(R.id.val1)
        val val2: TextView = view.findViewById(R.id.val2)
        val val3: TextView = view.findViewById(R.id.val3)

        val1.text = item.HDDVolume.toString()
        val2.text = item.RAMVolume.toString()
        val3.text = item.screenTime.toString()

        return view
    }
}
```

TableAVGItem.kt:

```
package com.example.lw_4

data class TableAVGItem(
                val HDDVolume: Int,
                val RAMVolume: Int,
                val screenTime: Int
)
```

TableAVGSomeAdapter.kt:

```
package com.example.lw_4

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.TextView

class TableAVGSomeAdapter(private val context: Context, private val items:
List<TableAVGSomeItem>) :
    ArrayAdapter<TableAVGSomeItem>(context, R.layout.list_item_avg_some_table,
items) {

    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
        val item = getItem(position)!!
        val view = convertView ?:
LayoutInflater.from(context).inflate(R.layout.list_item_avg_some_table, parent, false)

        val val1: TextView = view.findViewById(R.id.val1)
        val val2: TextView = view.findViewById(R.id.val2)
        val val3: TextView = view.findViewById(R.id.val3)
        val val4: TextView = view.findViewById(R.id.val4)
        val prop4: TextView = view.findViewById(R.id.prop4)

        val1.text = item.HDDVolume.toString()
        val2.text = item.RAMVolume.toString()
        val3.text = item.screenTime.toString()

        if (item.prop == "Производитель") {
            prop4.text = "Производитель"
            val4.text = item.value
        }
        else if (item.prop == "Наличие SSD") {
            prop4.text = "Наличие SSD"
            val4.text = if (item.value == "true") "Да" else "Нет"
        }
        else {
```

```
            prop4.text = "HULL HD"
            val4.text = if (item.value == "true") "Да" else "Нет"
        }

        return view
    }
}
```

TableAVGSomeItem.kt:

```
package com.example.lw_4

data class TableAVGSomeItem(
    val prop: String,
    val value: String,
    val HDDVolume: Int,
    val RAMVolume: Int,
    val screenTime: Int
)
```

TableGroupAdapter.kt:

```
package com.example.lw_4

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.TextView

class TableGroupAdapter(private val context: Context, private val items:
List<TableGroupItem>) :
    ArrayAdapter<TableGroupItem>(context, R.layout.list_item_group_table, items) {

    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
        val item = getItem(position)!!
        val view = convertView ?:
LayoutInflater.from(context).inflate(R.layout.list_item_group_table, parent, false)

        val val1: TextView = view.findViewById(R.id.val1)
        val val2: TextView = view.findViewById(R.id.val2)
        val val3: TextView = view.findViewById(R.id.val3)

        val prop1: TextView = view.findViewById(R.id.prop1)
        val prop2: TextView = view.findViewById(R.id.prop2)

        if (item.col1 == "Производитель") {
            prop1.text = "Производитель"
```

```
        val1.text = item.value1
    }
    else if(item.col1 == "Объём HDD") {
        prop1.text = "Объём HDD"
        val1.text = item.value1
    }
    else if(item.col1 == "Наличие SSD") {
        prop1.text = "Наличие SSD"
        val1.text = if (item.value1 == "true") "Да" else "Нет"
    }
    else if(item.col1 == "Объём RAM") {
        prop1.text = "Объём ОП"
        val1.text = item.value1
    }
    else if(item.col1 == "Наличие FUL HD") {
        prop1.text = "Наличие FUL HD"
        val1.text = if (item.value1 == "true") "Да" else "Нет"
    }
    else {
        prop1.text = "Время автономной работы"
        val1.text = item.value1
    }

    if (item.col2 == "Производитель") {
        prop2.text = "Производитель"
        val2.text = item.value2
    }
    else if(item.col2 == "Объём HDD") {
        prop2.text = "Объём HDD"
        val2.text = item.value2
    }
    else if(item.col2 == "Наличие SSD") {
        prop2.text = "Наличие SSD"
        val2.text = if (item.value2 == "true") "Да" else "Нет"
    }
    else if(item.col2 == "Объём RAM") {
        prop2.text = "Объём ОП"
        val2.text = item.value2
    }
    else if(item.col2 == "Наличие FULL HD") {
        prop2.text = "Наличие FHD"
        val2.text = if (item.value2 == "true") "Да" else "Нет"
    }
    else {
        prop2.text = "Время автономной работы"
        val2.text = item.value2
    }

    val3.text = item.count.toString()
```

```
        return view
    }
}
```

TableGroupItem.kt:

```
package com.example.lw_4

data class TableGroupItem(
    val col1: String,
    val col2: String,
    val value1: String,
    val value2: String,
    val count: Int
)
```

TableItem.kt:

```
package com.example.lw_4

data class TableItem(val ID: Int,
                     val manufacturerName: String,
                     val HDDVolume: Int,
                     val SSDPresent: Boolean,
                     val RAMVolume: Int,
                     val isFHD: Boolean,
                     val screenTime: Int,)
```

value_greater_than.kt:

```
package com.example.lw_4

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.ListView
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class value_greater_than : AppCompatActivity() {
    private lateinit var laptops: MutableList<Laptop>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_value_greater_than)
```

```
        this.laptops = intent.getParcelableArrayListExtra<Laptop>("laptops") as
ArrayList<Laptop>
        this.renderLaptopListViews()

        val mainButton = findViewById<Button>(R.id.mainButton)
        mainButton.setOnClickListener {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets
->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }
    }

    private fun renderLaptopListViews() {
        val laptopsListViews = findViewById<ListView>(R.id.laptopsListViews)
        val laptopsItems: MutableList<TableItem> = mutableListOf()

        for (i in 0..this.laptops.size - 1) {
            laptopsItems.add(TableItem(
                this.laptops[i].ID,
                this.laptops[i].manufacturerName,
                this.laptops[i].HDDVolume,
                this.laptops[i].SSDPresent,
                this.laptops[i].RAMVolume,
                this.laptops[i].isFHD,
                this.laptops[i].screenTime
            ))
        }
        val adapter = TableAdapter(this, laptopsItems)
        laptopsListViews.adapter = adapter
    }
}
```

activity_avg_group.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    tools:context=".avg_group">

    <ListView
        android:id="@+id/listView"
        android:layout_width="409dp"
        android:layout_height="500dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/mainButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/listView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_double_group.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".double_group">

    <ListView
        android:id="@+id/laptopsListViews"
        android:layout_width="409dp"
        android:layout_height="500dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/mainButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/laptopsListViews" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_lower_than_avg.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".lower_than_AVG">

    <ListView
        android:id="@+id/laptopsListViews"
        android:layout_width="409dp"
        android:layout_height="500dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/mainButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/laptopsListViews" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/laptopsListViews"
```

```
        android:layout_width="409dp"
        android:layout_height="300dp"
        android:layout_marginTop="80dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/newLaptopButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:text="Добавить ноутбук"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/laptopsListViews" />

    <Button
        android:id="@+id/deleteLaptopButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Удалить ноутбук"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/newLaptopButton" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_new_laptop.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NewLaptop">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:text="Добавление ноутбука"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Производитель"
    android:textSize="18sp"
    android:textStyle="bold|italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Объём жёсткого диска (в ГБ)"
    android:textSize="18sp"
    android:textStyle="bold|italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/manufacturerName" />

<CheckBox
    android:id="@+id/SSDPresent"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Наличие SSD"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/HDDVolume" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Объём оперативной памяти (в ГБ)"
    android:textSize="18sp"
    android:textStyle="bold|italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/SSDPresent" />

<CheckBox
    android:id="@+id/isFHD"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Наличие Full HD разрешения"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/RAMVolume" />

<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Время автономной работы (в часах)"
    android:textSize="18sp"
    android:textStyle="bold|italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/isFHD" />

<EditText
    android:id="@+id/manufacturerName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:ems="10"
    android:inputType="text"
    android:text="Acer"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />

<EditText
    android:id="@+id/HDDVolume"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:ems="10"
    android:inputType="text"
    android:text="512"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.1"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3" />

    <EditText
        android:id="@+id/RAMVolume"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:ems="10"
        android:inputType="text"
        android:text="12"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.1"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />

    <EditText
        android:id="@+id/screenTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:ems="10"
        android:inputType="text"
        android:text="5"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.1"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView5" />

    <Button
        android:id="@+id/newLaptopButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Добавить ноутбук"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/screenTime" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_value_greater_than.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".value_greater_than">

    <ListView
        android:id="@+id/laptopsListViews"
        android:layout_width="409dp"
        android:layout_height="500dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/mainButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/laptopsListViews" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

avg_group_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <Spinner
        android:id="@+id/prop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

delete_laptop.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/deletableID"
```

```
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:hint="Введите ID" />
</LinearLayout>
```

double_group_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <Spinner
        android:id="@+id/prop1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Spinner
        android:id="@+id/prop2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

greater_than_one_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <Spinner
        android:id="@+id/prop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/valueMax"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Введите ID" />
</LinearLayout>
```

laptop_max_value_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:padding="16dp">

   <Spinner
      android:id="@+id/prop"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />
</LinearLayout>
```

list_item_avg_some_table.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:padding="8dp">

   <TableRow>
      <TextView
         android:id="@+id/prop1"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="HDD" />

      <TextView
         android:id="@+id/val1"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Property 1" />
   </TableRow>

   <TableRow>
      <TextView
         android:id="@+id/prop2"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="RAM" />

      <TextView
         android:id="@+id/val2"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
    </TableRow>

    <TableRow>
      <TextView
        android:id="@+id/prop3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Автономная работа" />

      <TextView
        android:id="@+id/val3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 3" />
    </TableRow>

    <TableRow>
      <TextView
        android:id="@+id/prop4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="" />

      <TextView
        android:id="@+id/val4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 4" />
    </TableRow>
</TableLayout>
```

list_item_avg_table.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">

    <TableRow>
      <TextView
```

```
        android:id="@+id/prop1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="HDD" />

    <TextView
        android:id="@+id/val1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
  </TableRow>

  <TableRow>
    <TextView
        android:id="@+id/prop2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="RAM" />

    <TextView
        android:id="@+id/val2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
  </TableRow>

  <TableRow>
    <TextView
        android:id="@+id/prop3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Автономная работа" />

    <TextView
        android:id="@+id/val3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
  </TableRow>
</TableLayout>
```

list_item_group_table.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:padding="8dp">

   <TableRow>
      <TextView
         android:id="@+id/prop1"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="HDD" />

      <TextView
         android:id="@+id/val1"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Property 2" />
   </TableRow>

   <TableRow>
      <TextView
         android:id="@+id/prop2"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="RAM" />

      <TextView
         android:id="@+id/val2"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Property 2" />
   </TableRow>

   <TableRow>
      <TextView
         android:id="@+id/prop3"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Количество" />

      <TextView
```

```
        android:id="@+id/val3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Количество" />
  </TableRow>
</TableLayout>
```

list_item_table.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:padding="8dp">

  <TableRow>
    <TextView
      android:id="@+id/prop1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="ID" />

    <TextView
      android:id="@+id/val1"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="Property 2" />
  </TableRow>

  <TableRow>
    <TextView
      android:id="@+id/prop2"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="Производитель" />

    <TextView
      android:id="@+id/val2"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="Property 2" />
  </TableRow>

  <TableRow>
```

```
    <TextView
        android:id="@+id/prop3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="HDD" />

    <TextView
        android:id="@+id/val3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
</TableRow>

<TableRow>
    <TextView
        android:id="@+id/prop4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Наличие SSD" />

    <TextView
        android:id="@+id/val4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
</TableRow>

<TableRow>
    <TextView
        android:id="@+id/prop5"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="RAM" />

    <TextView
        android:id="@+id/val5"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
</TableRow>

<TableRow>
    <TextView
        android:id="@+id/prop6"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="FULL HD" />

    <TextView
        android:id="@+id/val6"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
  </TableRow>

  <TableRow>
    <TextView
        android:id="@+id/prop7"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Автономная работа" />

    <TextView
        android:id="@+id/val7"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Property 2" />
  </TableRow>
</TableLayout>
```

lower_than_avg_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:padding="16dp">

  <Spinner
    android:id="@+id/prop"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

sort_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <Spinner
        android:id="@+id/prop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <RadioGroup
        android:id="@+id/sortRadioGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <RadioButton
            android:id="@+id/ASC_radio"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="ASC" />

        <RadioButton
            android:id="@+id/DESC_radio"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="DESC" />
    </RadioGroup>
</LinearLayout>
```

sum_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <Spinner
        android:id="@+id/prop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

value_greater_than_option.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <Spinner
        android:id="@+id/prop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/valueMax"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Введите ID" />
</LinearLayout>
```

AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.LW_4"
        tools:targetApi="31">
        <activity
            android:name=".double_group"
            android:exported="false" />
        <activity
            android:name=".lower_than_AVG"
            android:exported="false" />
        <activity
            android:name=".value_greater_than"
```

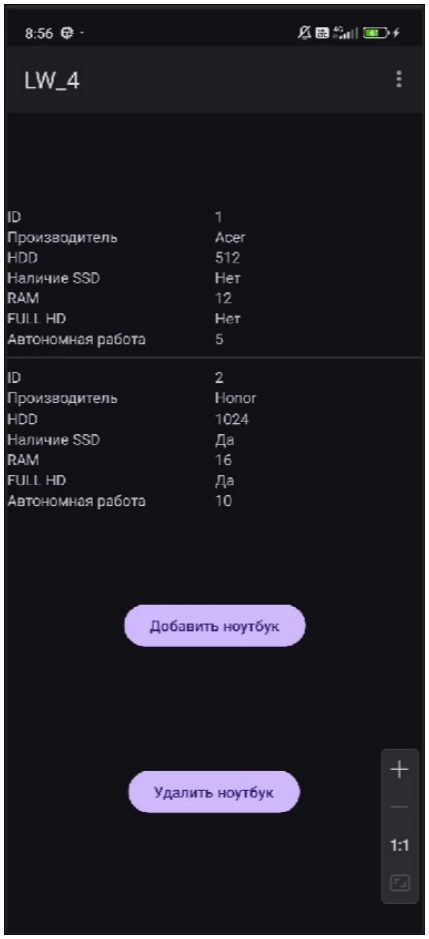
```
        android:exported="false" />
    <activity
        android:name=".avg_group"
        android:exported="false" />
    <activity
        android:name=".NewLaptop"
        android:exported="false" />
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
  </application>

</manifest>
```

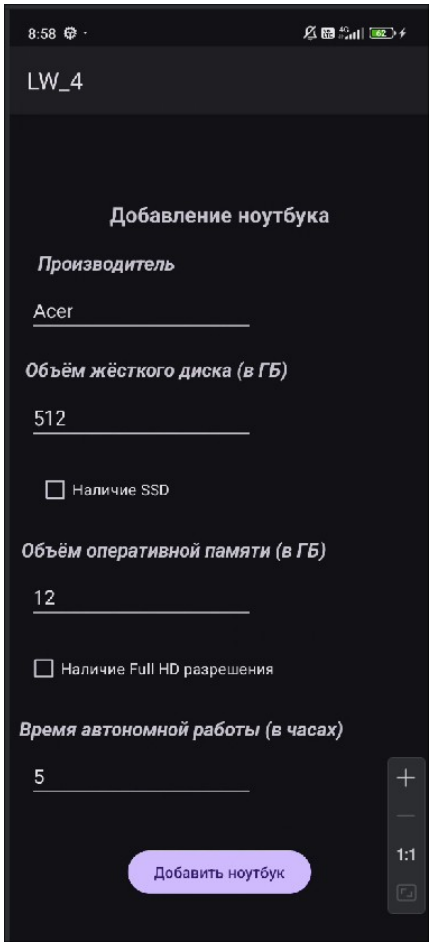
option_menu.xml:

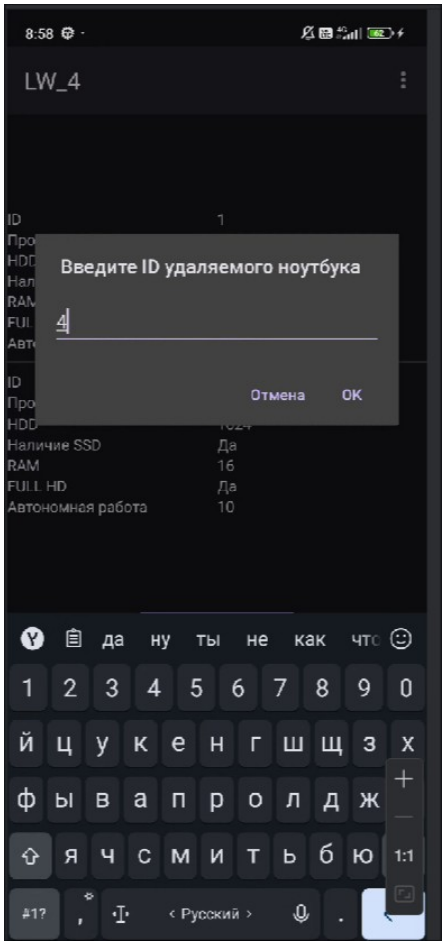
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
    android:id="@+id/sort"
    android:title="1) Сортировка"
    android:orderInCategory="100"
    app:showAsAction="never"/>
  <item
    android:id="@+id/double_group"
    android:title="2) Группировка по 2 параметрам"
    android:orderInCategory="100"
    app:showAsAction="never"/>

  <item
    android:id="@+id/RAM_sum"
    android:title="3) Вычислить сумму ОП"
    android:orderInCategory="100"
    app:showAsAction="never"/>
  <item
    android:id="@+id/AVG_group_option"
    android:title="4) СЗ по сгруппированным поляи"
    android:orderInCategory="100"
    app:showAsAction="never"/>

  <item
    android:id="@+id/laptop_max_value"
    android:title="5) Ноутбук с максимальным значением поля"
```

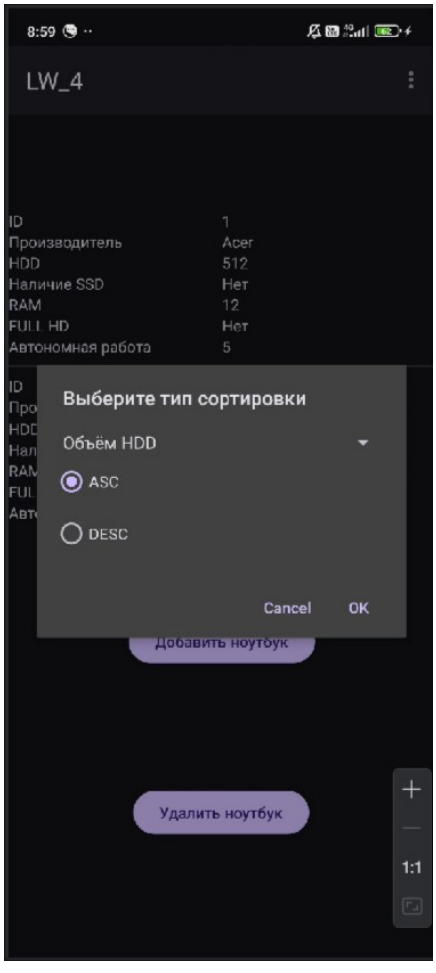
```
        android:orderInCategory="100"
        app:showAsAction="never"/>

    <item
        android:id="@+id/value_greater_than"
        android:title="6) Ноутбуки со значением больше заданного"
        android:orderInCategory="100"
        app:showAsAction="never"/>

    <item
        android:id="@+id/lower_than_AVG"
        android:title="7) Ноутбуки со значением меньше среднего"
        android:orderInCategory="100"
        app:showAsAction="never"/>

    <item
        android:id="@+id/lower_than_AVG_one"
        android:title="8) Ноутбук (1) со значением больше заданного"
        android:orderInCategory="100"
        app:showAsAction="never"/>
</menu>
```

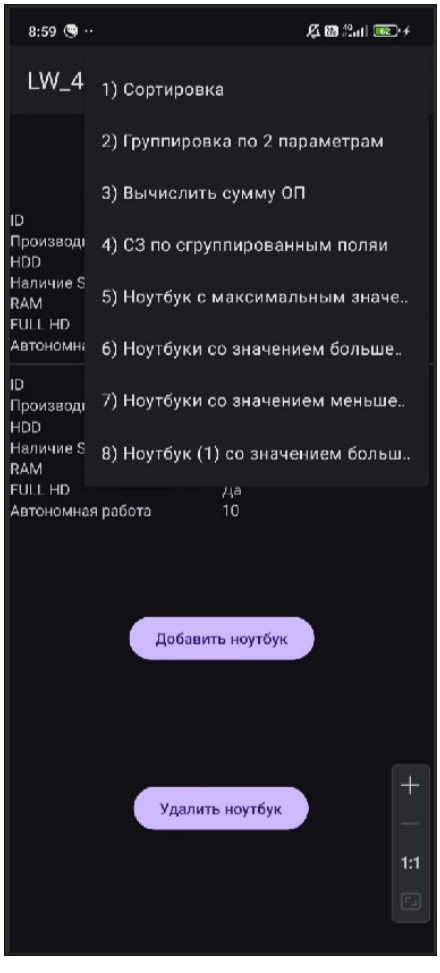
Результаты выполнения работы:

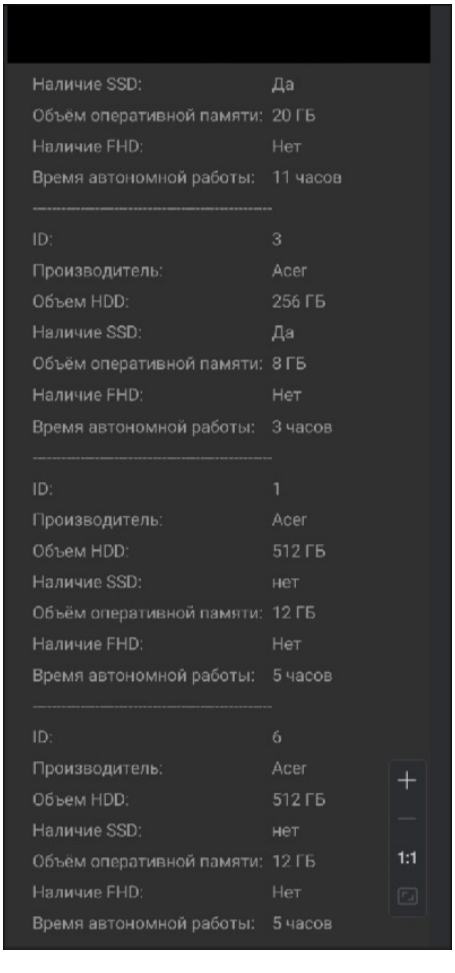


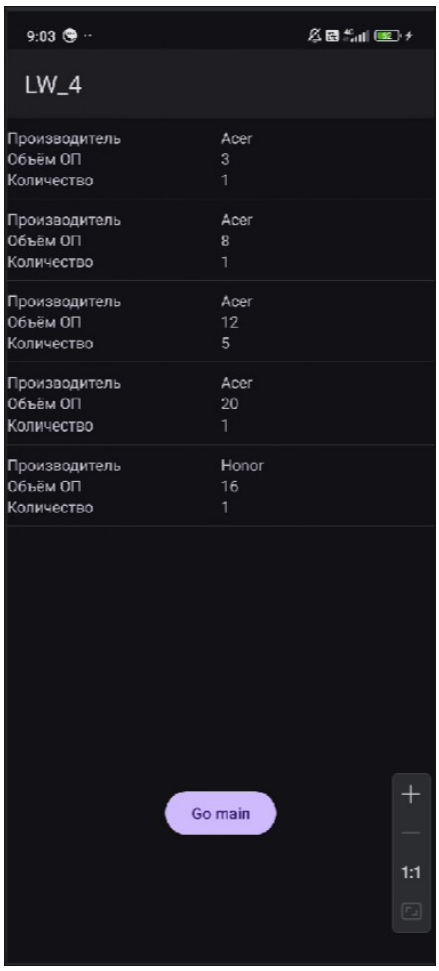


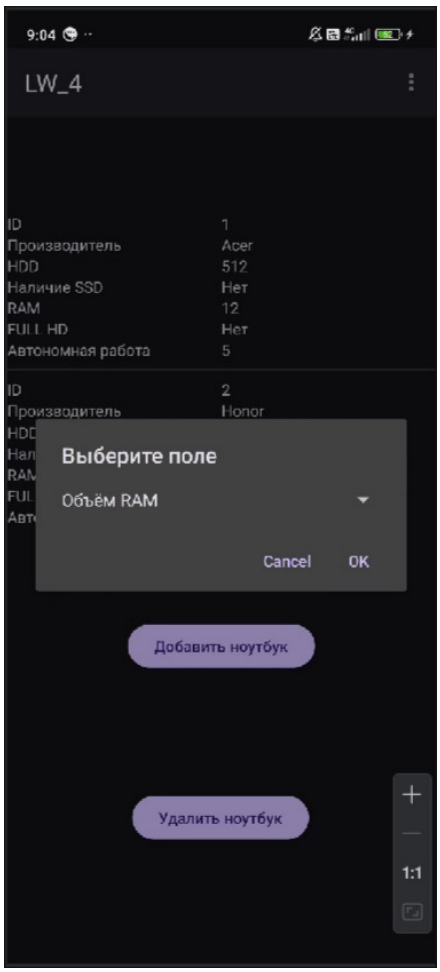




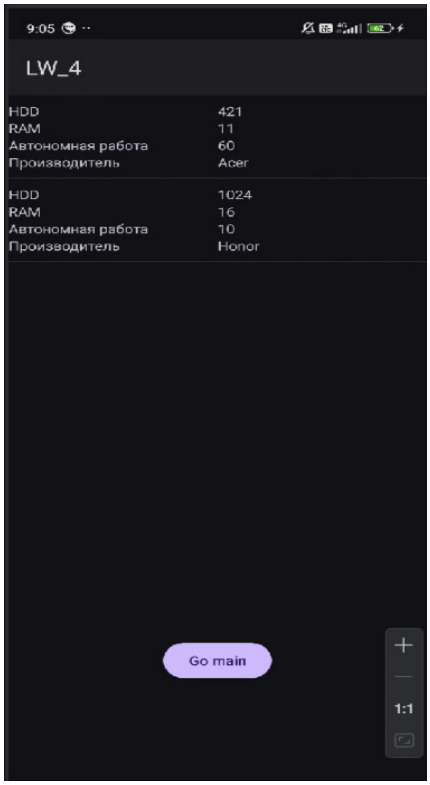


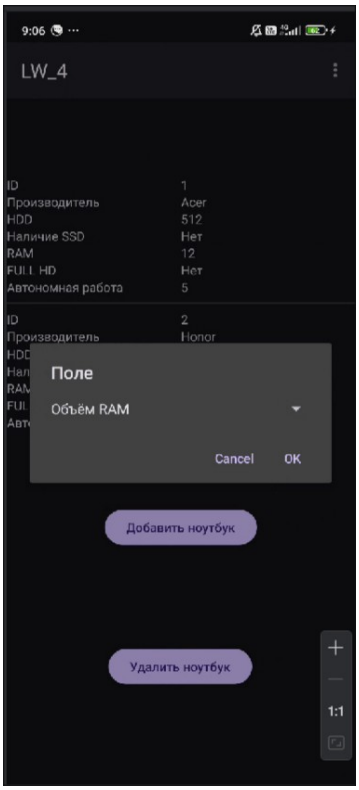


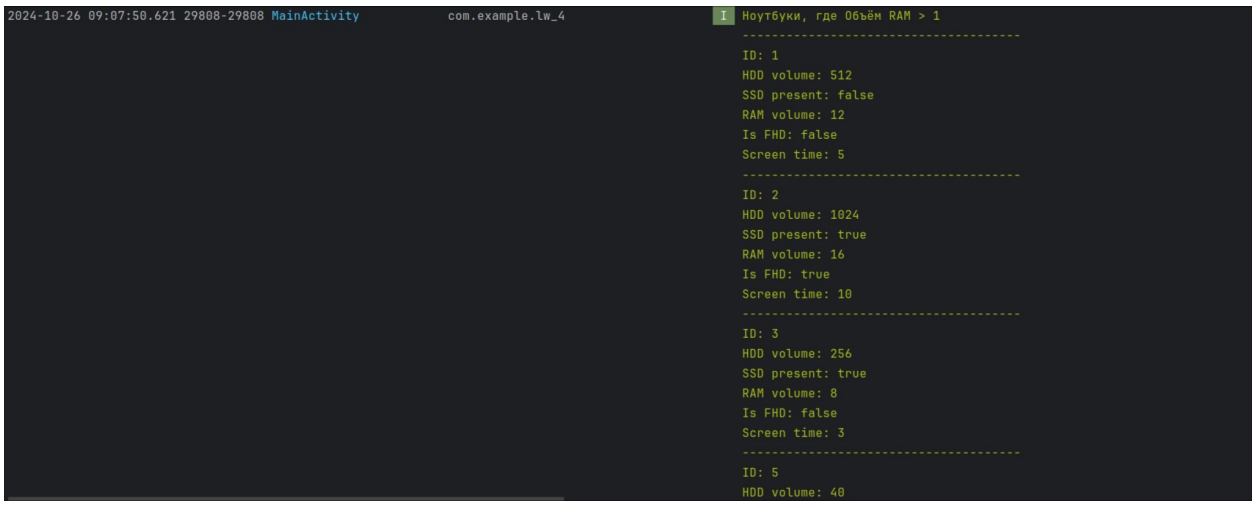

9:03
LW_4
Производитель Acer
Объём ОП 3
Количество 1
Производитель Acer
Объём ОП 8
Количество 1
Производитель Acer
Объём ОП 12
Количество 5
Производитель Acer
Объём ОП 20
Количество 1
Производитель Honor
Объём ОП 16
Количество 1
Go main
1:1

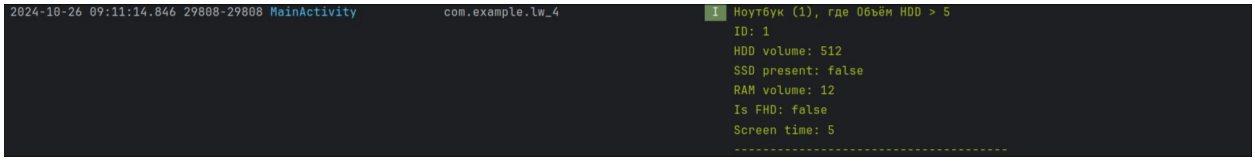

9:04
LW_4
ID 1
Производитель Acer
HDD 512
Наличие SSD Нет
RAM 12
FULL HD Нет
Автономная работа 5
ID 2
Производитель Honor
Выберите поле
Объём RAM
Cancel
OK
Добавить ноутбук
Удалить ноутбук
1:1

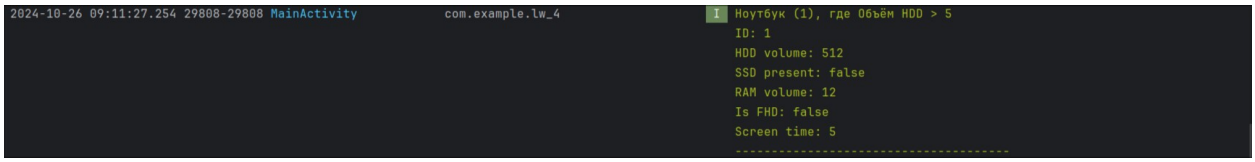

Сумма значений (Объём RAM) = 107
1:1


9:05
LW_4
HDD 421
RAM 11
Автономная работа 60
Производитель Acer
HDD 1024
RAM 16
Автономная работа 10
Производитель Honor
Go main
1:1


9:06
LW_4
ID 1
Производитель Acer
HDD 512
Наличие SSD Нет
RAM 12
FULL HD Нет
Автономная работа 5
ID 2
Производитель Honor
Поле
Объём RAM
Cancel
OK
Добавить ноутбук
Удалить ноутбук
1:1







Вывод: в ходе лабораторной работы было разработано приложение, взаимодействующее с базой данных SQLite.