

**Министерство науки и высшего образования Российской Федерации**  
**Федеральное государственное автономное образовательное**  
**учреждение высшего образования**  
**Национальный исследовательский**  
**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники  
Отделение информационных технологий

Отчет по лабораторной работе №15 по дисциплине

**«Язык Kotlin и основы разработки»**

Диалоги

Выполнил:

Студент группы 1A22



О.К. Кравцов

Проверил:

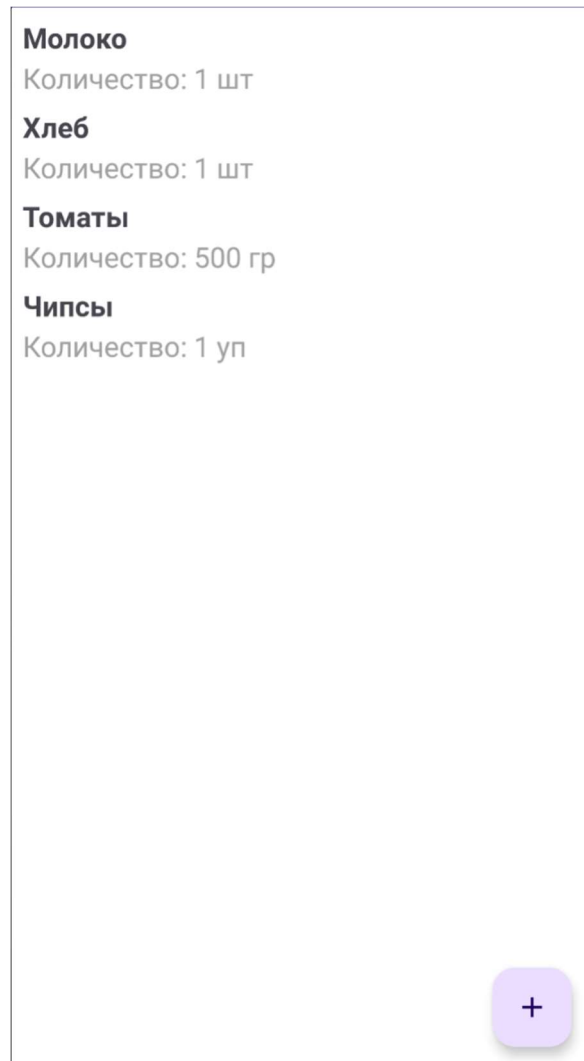
Ст. преп. ОИТ ИШИТР



В.А. Дорофеев

## Задание

Создайте приложение для ведения списка покупок. В основе должен лежать RecyclerView, который отображает список примерно в следующем виде:



**Молоко**  
Количество: 1 шт

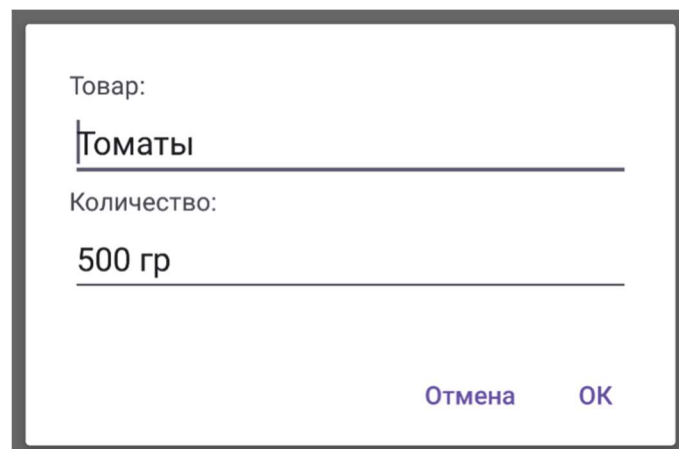
**Хлеб**  
Количество: 1 шт

**Томаты**  
Количество: 500 гр

**Чипсы**  
Количество: 1 уп

+

При нажатии плавающей кнопки должен появляться диалог с пустыми полями, а при его положительном завершении – в список должна добавляться новая покупка:



Товар:

Количество:

Отмена    ОК

При касании записи она должна открываться на редактирование: появляется такой же диалог, но уже с заполненными полями, а при положительном завершении – этот элемент в списке обновляется.

При жесте смахивания на какой-либо записи (проведении по ней пальцем влево или вправо) она должна удаляться из списка.

## Ход работы

1. Создан проект Lab15 на основе Empty Views Activity.
2. Настроены ресурсы приложения:
  - `dimens.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Отступы -->
    <dimen name="margin_small">8dp</dimen>
    <dimen name="margin_medium">16dp</dimen>

    <!-- Размеры текста -->
    <dimen name="text_size_small">14sp</dimen>
    <dimen name="text_size_medium">16sp</dimen>
    <dimen name="text_size_large">18sp</dimen>

    <!-- Радиус скругления -->
    <dimen name="corner_radius">8dp</dimen>

    <!-- Высота тени -->
    <dimen name="elevation">4dp</dimen>
</resources>
```

- `strings.xml`

```
<resources>
    <string name="app_name">Lab15</string>

    <!-- Диалог -->
    <string name="dialog_title_add">Добавить товар</string>
    <string name="dialog_title_edit">Редактировать товар</string>
    <string name="dialog_product_label">Товар:</string>
    <string name="dialog_quantity_label">Количество:</string>
    <string name="dialog_product_hint">Введите название товара</string>
    <string name="dialog_quantity_hint">Например: 1 шт или 500 гр</string>
    <string name="dialog_button_ok">ОК</string>
    <string name="dialog_button_cancel">Отмена</string>

    <!-- Элемент списка -->
    <string name="item_quantity_format">Количество: %s</string>

    <!-- FAB -->
    <string name="fab_content_description">Добавить новый товар</string>
</resources>
```

### 3. Реализована модель данных:

– Purchase.kt

```
package ru.olegkravtsov.lab15

import java.io.Serializable

data class Purchase(
    val id: Long = System.currentTimeMillis(),
    var name: String,
    var quantity: String
) : Serializable
```

### 4. Создана основная активность MainActivity:

```
package ru.olegkravtsov.lab15

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.ItemTouchHelper
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity(),
PurchaseDialogFragment.PurchaseDialogListener {

    private lateinit var recyclerView: RecyclerView
    private lateinit var fabAdd: FloatingActionButton
    private lateinit var adapter: PurchaseAdapter

    private val purchases = mutableListOf<Purchase>()

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

        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
        }

        initView()
        setupRecyclerView()
        setupSwipeToDelete()
    }

    private fun initView() {
        recyclerView = findViewById(R.id.recyclerView)
        fabAdd = findViewById(R.id.fabAdd)
```

```

        fabAdd.setOnClickListener {
            showAddPurchaseDialog()
        }
    }

    private fun setupRecyclerView() {
        adapter = PurchaseAdapter(
            purchases = purchases,
            onItemClick = { purchase ->
                showEditPurchaseDialog(purchase)
            }
        )

        recyclerView.layoutManager = LinearLayoutManager(this)
        recyclerView.adapter = adapter
    }

    private fun setupSwipeToDelete() {
        val swipeCallback = object : ItemTouchHelper.SimpleCallback(
            0,
            ItemTouchHelper.LEFT or ItemTouchHelper.RIGHT
        ) {
            override fun onMove(
                recyclerView: RecyclerView,
                viewHolder: RecyclerView.ViewHolder,
                target: RecyclerView.ViewHolder
            ): Boolean = false

            override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction:
Int) {
                val position = viewHolder.absoluteAdapterPosition
                adapter.removePurchase(position)
            }
        }

        val itemTouchHelper = ItemTouchHelper(swipeCallback)
        itemTouchHelper.attachToRecyclerView(recyclerView)
    }

    private fun showAddPurchaseDialog() {
        val dialog = PurchaseDialogFragment.newInstance()
        dialog.setListener(this)
        dialog.show(supportFragmentManager, "add_purchase")
    }

    private fun showEditPurchaseDialog(purchase: Purchase) {
        val dialog = PurchaseDialogFragment.newInstance(purchase)
        dialog.setListener(this)
        dialog.show(supportFragmentManager, "edit_purchase")
    }

    override fun onPurchaseAdded(purchase: Purchase) {
        adapter.addPurchase(purchase)
    }
}

```

```

        override fun onPurchaseUpdated(purchase: Purchase) {
            val position = adapter.getPurchasePosition(purchase)
            if (position != -1) {
                adapter.updatePurchase(position, purchase)
            }
        }
    }
}

```

## 5. Реализован адаптер PurchaseAdapter для отображения списка покупок:

```

package ru.olegkravtsov.lab15

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class PurchaseAdapter(
    private var purchases: MutableList<Purchase>,
    private val onItemClick: (Purchase) -> Unit
) : RecyclerView.Adapter<PurchaseAdapter.PurchaseViewHolder>() {

    class PurchaseViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val tvName: TextView = itemView.findViewById(R.id.tvName)
        val tvQuantity: TextView = itemView.findViewById(R.id.tvQuantity)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
PurchaseViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_purchase, parent, false)
        return PurchaseViewHolder(view)
    }

    override fun onBindViewHolder(holder: PurchaseViewHolder, position: Int) {
        val purchase = purchases[position]
        holder.tvName.text = purchase.name
        holder.tvQuantity.text =
holder.itemView.context.getString(R.string.item_quantity_format, purchase.quantity)

        holder.itemView.setOnClickListener {
            onItemClick(purchase)
        }
    }

    override fun getItemCount(): Int = purchases.size

    fun addPurchase(purchase: Purchase) {
        purchases.add(purchase)
        notifyItemInserted(purchases.size - 1)
    }

    fun updatePurchase(position: Int, purchase: Purchase) {
        purchases[position] = purchase
        notifyItemChanged(position)
    }
}

```

```

fun removePurchase(position: Int) {
    purchases.removeAt(position)
    notifyItemRemoved(position)
}

fun getPurchasePosition(purchase: Purchase): Int {
    return purchases.indexOfFirst { it.id == purchase.id }
}
}

```

## 6. Создан диалог PurchaseDialogFragment для добавления/редактирования:

```

package ru.olegkravtsov.lab15

import android.app.AlertDialog
import android.app.Dialog
import android.os.Bundle
import android.widget.EditText
import androidx.fragment.app.DialogFragment

class PurchaseDialogFragment : DialogFragment() {

    interface PurchaseDialogListener {
        fun onPurchaseAdded(purchase: Purchase)
        fun onPurchaseUpdated(purchase: Purchase)
    }

    companion object {
        private const val ARG_PURCHASE = "purchase"

        fun newInstance(purchase: Purchase? = null): PurchaseDialogFragment {
            val args = Bundle().apply {
                putSerializable(ARG_PURCHASE, purchase)
            }
            return PurchaseDialogFragment().apply {
                arguments = args
            }
        }
    }

    private var listener: PurchaseDialogListener? = null
    private var existingPurchase: Purchase? = null

    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        existingPurchase = arguments?.getSerializable(ARG_PURCHASE,
Purchase::class.java)

        val builder = AlertDialog.Builder(requireContext())
        val inflater = requireActivity().layoutInflater
        val view = inflater.inflate(R.layout.dialog_purchase, null)

        val etName = view.findViewById<EditText>(R.id.etName)
        val etQuantity = view.findViewById<EditText>(R.id.etQuantity)

        // Если передан существующий товар - заполняем поля
        existingPurchase?.let {

```



```

        etName.setText(it.name)
        etQuantity.setText(it.quantity)
    }

    val title = if (existingPurchase == null) {
        R.string.dialog_title_add
    } else {
        R.string.dialog_title_edit
    }

    builder.setView(view)
        .setTitle(title)
        .setPositiveButton(R.string.dialog_button_ok) { dialog, which ->
            val name = etName.text.toString().trim()
            val quantity = etQuantity.text.toString().trim()

            if (name.isNotEmpty() && quantity.isNotEmpty()) {
                if (existingPurchase == null) {
                    // Добавление нового товара
                    listener?.onPurchaseAdded(Purchase(name = name, quantity =
quantity))
                } else {
                    // Обновление существующего товара
                    val updatedPurchase = existingPurchase!!.copy(
                        name = name,
                        quantity = quantity
                    )
                    listener?.onPurchaseUpdated(updatedPurchase)
                }
            }
        }
        .setNegativeButton(R.string.dialog_button_cancel) { dialog, which ->
            dialog.dismiss()
        }

    return builder.create()
}

fun setListener(listener: PurchaseDialogListener) {
    this.listener = listener
}
}

```

## 7. Настроены макеты:

– activity\_main.xml - основной layout с RecyclerView и FAB

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/margin_medium"
        android:contentDescription="@string/fab_content_description"
        android:src="@android:drawable/ic_input_add" />

</FrameLayout>
```

– item\_purchase.xml - элемент списка на основе CardView

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="@dimen/margin_small"
    app:cardCornerRadius="@dimen/corner_radius"
    app:cardElevation="@dimen/elevation">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="@dimen/margin_medium">

        <TextView
            android:id="@+id/tvName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="@dimen/text_size_large"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tvQuantity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_small"
            android:textSize="@dimen/text_size_small" />

    </LinearLayout>

</androidx.cardview.widget.CardView>
```

– dialog\_purchase.xml - разметка диалогового окна

```
<?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="@dimen/margin_medium">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/dialog_product_label"
        android:textSize="@dimen/text_size_medium" />

    <EditText
        android:id="@+id/etName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="@dimen/margin_medium"
        android:hint="@string/dialog_product_hint"
        android:inputType="textCapWords"
        android:autofillHints="name" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/dialog_quantity_label"
        android:textSize="@dimen/text_size_medium" />

    <EditText
        android:id="@+id/etQuantity"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/dialog_quantity_hint"
        android:inputType="text"
        android:autofillHints="none" />

</LinearLayout>
```

## Результат работы

Приложение успешно запускается и отображает интерфейс списка покупок (рис. 1). При нажатии на плавающую кнопку появляется диалоговое окно для добавления нового товара (рис. 2). После заполнения полей и нажатия кнопки "ОК" товар добавляется в список. При нажатии на существующий товар в списке открывается диалоговое окно с заполненными полями для редактирования (рис. 3). После изменения данных и подтверждения товар обновляется в списке. При смахивании товара влево или вправо он удаляется из списка с анимацией (рис. 4).

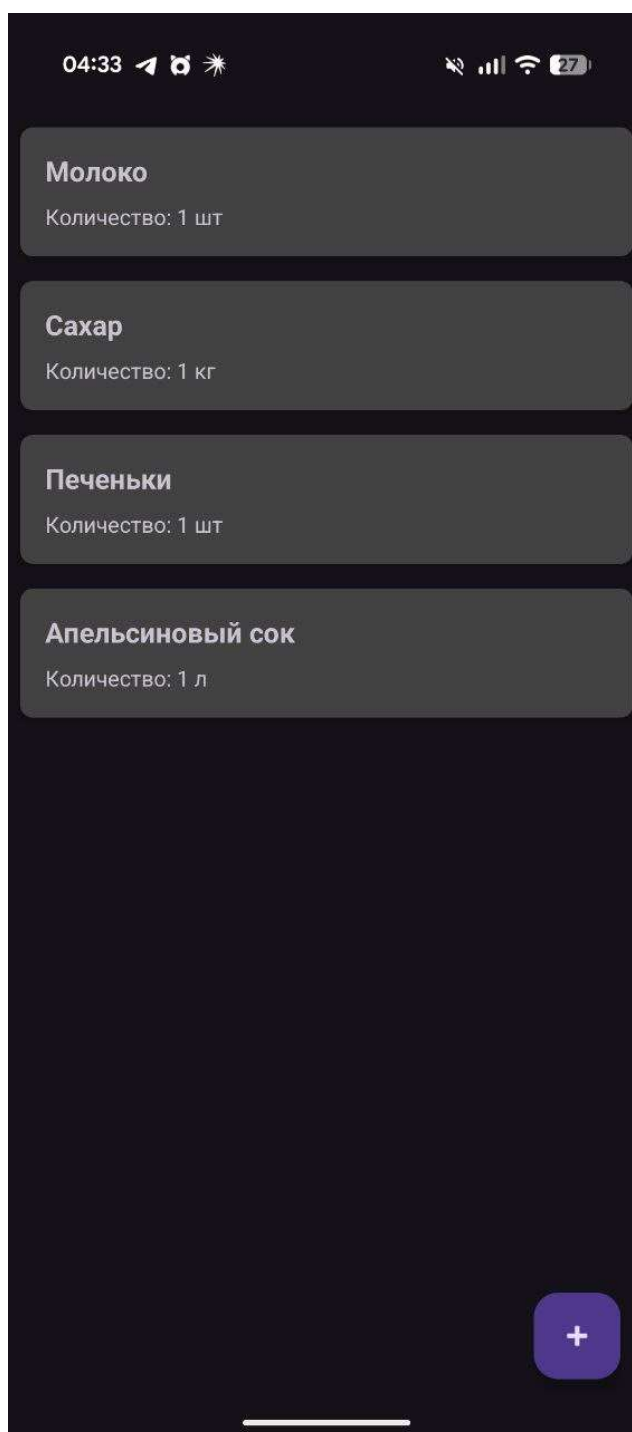


Рисунок 1 – Главный экран приложения с 4 добавленными товарами в список

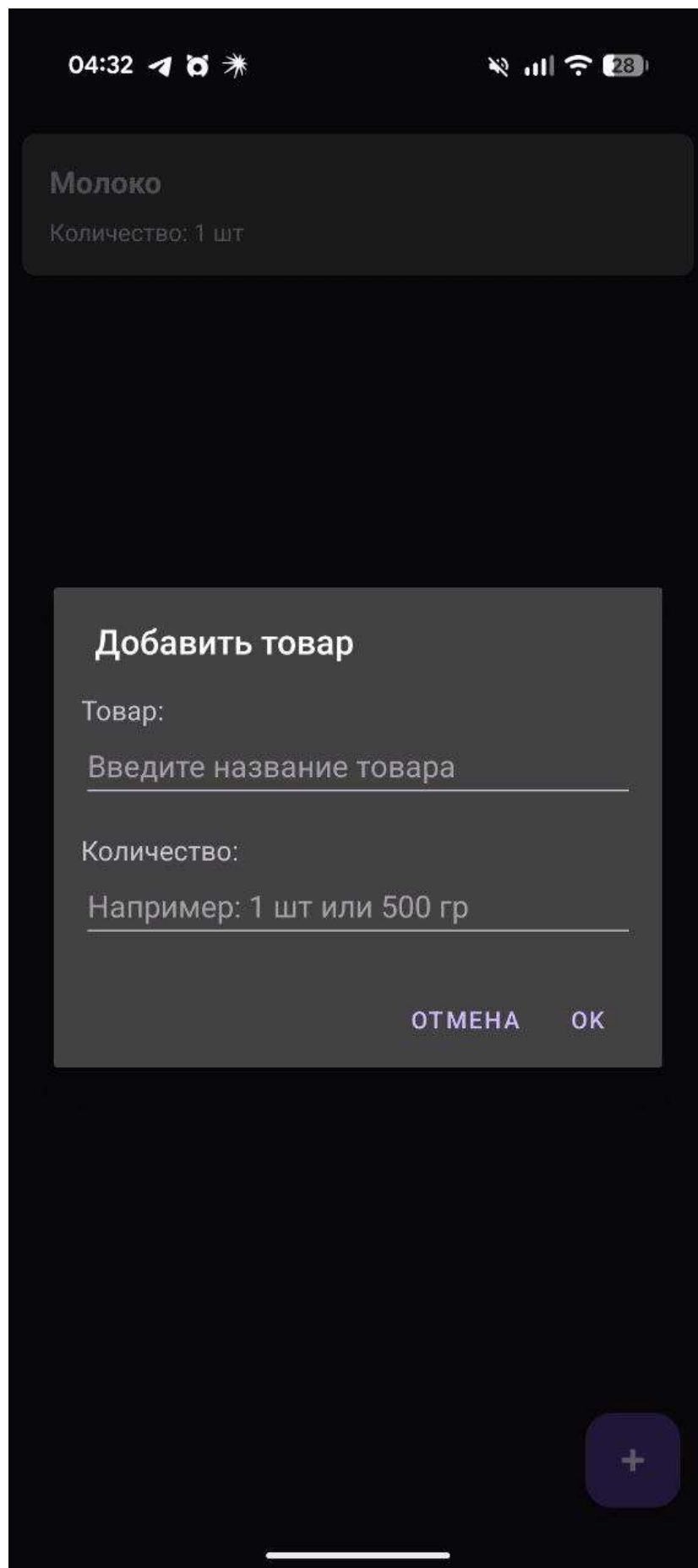


Рисунок 2 – Диалоговое окно добавления нового товара с пустыми полями ввода

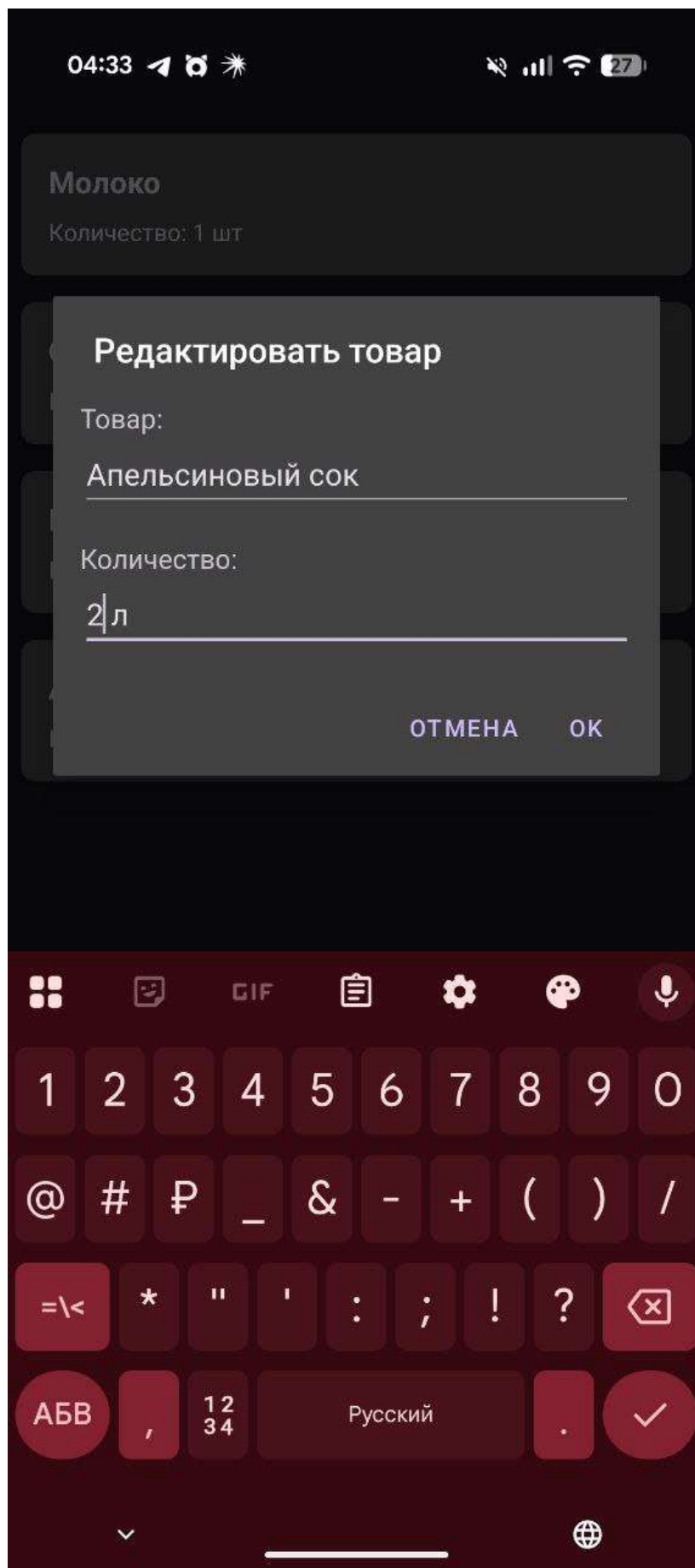


Рисунок 3 – Диалоговое окно редактирования существующего товара из списка



Рисунок 4 – Анимация удаления элемента списка жестом смахивания



## **Выводы**

В ходе выполнения лабораторной работы было разработано приложение для ведения списка покупок, использующее современные компоненты Android. RecyclerView с CardView обеспечивает отображение списка с современным дизайном. DialogFragment используется для показа диалоговых окон добавления и редактирования. Реализовано удаление элементов списка жестом смахивания с помощью ItemTouchHelper. Все строковые ресурсы и размеры вынесены в ресурсы, что облегчает поддержку и локализацию приложения. Приложение демонстрирует правильное использование архитектурных компонентов и соответствует требованиям задания.