Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №2

«Работа со списками»

Студент Д. А. Снитко

Проверил О. М. Внук

МИНСК 2024

**1 ЦЕЛЬ РАБОТЫ**

Целью работы является:

1. Определить и разработать адаптеры (android.widget.AdapterView), которые будут использоваться в приложении, выбранном в лабораторной работе №1.

2. Реализовать страницу Авторизации в мобильном приложении

**2 ИСХОДНЫЕ ДАННЫЕ**

Среда разработки для Android;

Язык программирования Kotlin;

Источник исходного кода: https://github.com/Luflexia/Currency-Converter

**3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Адаптеры в Android (AdapterView)** — это специальные компоненты, которые связывают данные с виджетами, позволяя отображать информацию в пользовательском интерфейсе. Они используются для работы со списками, таблицами и другими типами представлений, где требуется динамическое отображение данных.

Адаптеры играют роль посредников между источником данных и элементом пользовательского интерфейса, обеспечивая заполнение виджетов данными. Основные задачи адаптеров:

1. **Предоставление данных** — адаптер получает данные из различных источников (например, массивов, баз данных или сетевых запросов).
2. **Создание пользовательских интерфейсов** — адаптер создает элементы интерфейса для каждого элемента данных, обычно с помощью метода getView в классах Adapter.
3. **Оптимизация использования памяти** — за счёт использования механизма переработки (ViewHolder), адаптеры в RecyclerView обеспечивают производительность списка, избегая постоянного создания новых объектов при прокрутке.

Примеры адаптеров в приложениях Android:

* **RecyclerView.Adapter** — самый популярный адаптер для отображения больших списков. Работает с ViewHolder, который повышает производительность.
* **ArrayAdapter** — для простых списков, где каждый элемент представляет одну строку или другую простую структуру данных.

**RecyclerView и ViewHolder**:

* RecyclerView — это более продвинутая версия ListView, которая поддерживает эффективное отображение больших наборов данных. В отличие от ListView, он использует паттерн ViewHolder, который помогает избежать лишнего создания объектов во время прокрутки списка.
* ViewHolder — это класс, который хранит ссылки на виджеты, используемые для отображения элементов списка, что помогает оптимизировать работу приложения.

В данной работе для реализации списков банков и валют были созданы адаптеры BankAdapter и CurrencyAdapter. Эти адаптеры связывают данные (списки банков и валют) с элементами пользовательского интерфейса (списками в RecyclerView), обеспечивая динамическое обновление интерфейса и возможность взаимодействия пользователя с элементами списка.

**4 Код программы**

Файл BankAdapter.kt

package com.example.currencyconverterv2.adapters

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

import com.example.currencyconverterv2.R

import com.example.currencyconverterv2.models.Bank

class BankAdapter(private val banks: List<Bank>, private val onBankSelected: (Bank) -> Unit) :

RecyclerView.Adapter<BankAdapter.BankViewHolder>() {

class BankViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val bankName: TextView = itemView.findViewById(R.id.bankName)

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): BankViewHolder {

val view = LayoutInflater.from(parent.context)

.inflate(R.layout.item\_bank, parent, false)

return BankViewHolder(view)

}

override fun onBindViewHolder(holder: BankViewHolder, position: Int) {

val bank = banks[position]

holder.bankName.text = bank.name

holder.itemView.setOnClickListener {

onBankSelected(bank) // Вызываем лямбда-функцию при выборе банка

}

}

override fun getItemCount(): Int {

return banks.size

}

}

Файл CurrencyAdapter.kt

package com.example.currencyconverterv2.adapters

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView

import com.example.currencyconverterv2.R

import com.example.currencyconverterv2.models.Currency

class CurrencyAdapter(private val currencies: MutableList<Currency>) :

RecyclerView.Adapter<CurrencyAdapter.CurrencyViewHolder>() {

// ViewHolder для каждого элемента списка

class CurrencyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val currencyName: TextView = itemView.findViewById(R.id.currencyName)

val currencyRate: TextView = itemView.findViewById(R.id.currencyRate)

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CurrencyViewHolder {

val view = LayoutInflater.from(parent.context)

.inflate(R.layout.item\_currency, parent, false)

return CurrencyViewHolder(view)

}

override fun onBindViewHolder(holder: CurrencyViewHolder, position: Int) {

val currency = currencies[position]

holder.currencyName.text = currency.name

holder.currencyRate.text = currency.rate.toString()

}

override fun getItemCount(): Int {

return currencies.size

}

// Перемещение валюты

fun moveCurrency(fromPosition: Int, toPosition: Int) {

val fromCurrency = currencies.removeAt(fromPosition)

currencies.add(toPosition, fromCurrency)

notifyItemMoved(fromPosition, toPosition)

}

// Удаление валюты

fun removeCurrency(position: Int) {

currencies.removeAt(position)

notifyItemRemoved(position)

}

}

**5 ДЕМОНСТРАЦИЯ РАБОТЫ ПРИЛОЖЕНИЯ**

Экран авторизации в приложении представлен на рисунке 1

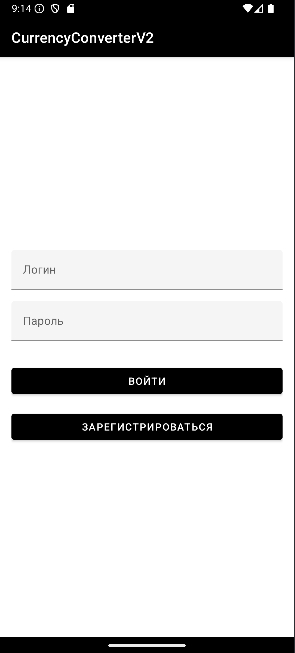


Рисунок 1

**6 Вывод**

В ходе выполнения лабораторной работы был изучен и реализован механизм работы с адаптерами в Android, что позволило создать динамические списки банков и валют с помощью RecyclerView и собственных адаптеров. Был разработан адаптер BankAdapter для отображения банков и адаптер CurrencyAdapter для работы с валютами, включая возможность перемещения и удаления элементов списка.

Ключевым элементом реализации стало использование паттерна ViewHolder для оптимизации работы приложения и повышения производительности при работе с большими списками данных.