**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**імені ТАРАСА ШЕВЧЕНКА**

****

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №8**

**з курсу**

**«Архітектура мобільних операційних систем і пристроїв»**

*Студентки 4 курсу*

*групи ПП-42*

*спеціальності 122 «Комп'ютерні науки»*

*ОП «Прикладне програмування»*

Матвіїв А.Ю.

*Викладач:*

Краснощок В.М.

**Київ – 2024**

***Тема:*** Робота із контент-провайдерами

***Мета:*** Отримати практичні навички створення власних контент-провайдерів.

***Хід роботи:***

Створимо проект під назвою «ContactsSample»:

MainActivity.kt:

package com.example.contactssample

import android.Manifest

import android.content.pm.PackageManager

import android.os.Bundle

import android.util.Log

import android.view.ContextMenu

import android.view.MenuItem

import android.view.View

import android.widget.AdapterView

import android.widget.ArrayAdapter

import android.widget.Button

import android.widget.EditText

import android.widget.ListView

import android.widget.SimpleAdapter

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.core.app.ActivityCompat

import androidx.core.content.ContextCompat

import androidx.lifecycle.*lifecycleScope*

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.launch

import kotlinx.coroutines.withContext

import org.w3c.dom.\*

import java.io.IOException

import java.net.HttpURLConnection

import java.net.URL

import javax.xml.parsers.DocumentBuilderFactory

import org.xml.sax.SAXException

import java.io.InputStream

import java.net.MalformedURLException

import javax.xml.parsers.DocumentBuilder

import javax.xml.parsers.ParserConfigurationException

class MainActivity : AppCompatActivity() {

   private lateinit var contactNameEditText: EditText

   private lateinit var saveContactButton: Button

   private lateinit var contactsListView: ListView

   private lateinit var contactsHandler: ContactsHandler

   private lateinit var contactsAdapter: ArrayAdapter<String>

   private lateinit var selectedContact: String

   override fun onCreate(savedInstanceState: Bundle?) {

       super.onCreate(savedInstanceState)

       setContentView(R.layout.*main\_layout*)

       requestContactPermissions()

       registerForContextMenu(contactsListView)

   }

   override fun onRestart() {

       super.onRestart()

       loadContacts()

   }

   private fun requestContactPermissions() {

       if (ContextCompat.checkSelfPermission(this, Manifest.permission.*READ\_CONTACTS*)

           != PackageManager.*PERMISSION\_GRANTED* ||

           ContextCompat.checkSelfPermission(this, Manifest.permission.*WRITE\_CONTACTS*)

           != PackageManager.*PERMISSION\_GRANTED*) {

           ActivityCompat.requestPermissions(

               this,

*arrayOf*(Manifest.permission.*READ\_CONTACTS*, Manifest.permission.*WRITE\_CONTACTS*),

               1

           )

       } else {

           initializeApp()

       }

   }

   override fun onCreateContextMenu(menu: ContextMenu?, v: View?, menuInfo: ContextMenu.ContextMenuInfo?) {

       super.onCreateContextMenu(menu, v, menuInfo)

*menuInflater*.inflate(R.menu.*menu\_delete*, menu)

   }

   override fun onContextItemSelected(item: MenuItem): Boolean {

       val info = item.*menuInfo* as AdapterView.AdapterContextMenuInfo

       val position = info.position

       val selectedContact = contactsAdapter.getItem(position)

       return when (item.*itemId*) {

           R.id.*delete* -> {

               if (selectedContact != null) {

                   contactsHandler.deleteContact(selectedContact)

                   loadContacts()

                   Toast.makeText(this, "$selectedContact deleted", Toast.*LENGTH\_SHORT*).show()

               }

               true

           }

           else -> super.onContextItemSelected(item)

       }

   }

   private fun deleteSelectedContact() {

       if (this::selectedContact.*isInitialized* && selectedContact.*isNotEmpty*()) {

           contactsHandler.deleteContact(selectedContact) // Delete the selected contact

           loadContacts() // Reload the contacts after deletion

           selectedContact = "" // Clear the selected contact

       } else {

           Toast.makeText(this, "Please select a contact to delete", Toast.*LENGTH\_SHORT*).show()

       }

   }

   private fun initializeApp() {

       contactNameEditText = findViewById(R.id.*contactNameEditText*)

       saveContactButton = findViewById(R.id.*saveContactButton*)

       contactsListView = findViewById(R.id.*contactsListView*)

       contactsHandler = ContactsHandler(this)

       contactsAdapter = ArrayAdapter(this, android.R.layout.*simple\_list\_item\_1*, *mutableListOf*())

       contactsListView.*adapter* = contactsAdapter

       loadContacts()

       saveContactButton.setOnClickListener **{**

saveContact()

**}**

contactsListView.setOnItemClickListener **{** \_, \_, position, \_ **->**

selectContact(position)

**}**

}

   private fun saveContact() {

       val contactName = contactNameEditText.*text*.toString()

       if (contactName.*isNotEmpty*()) {

           contactsHandler.insertContact(contactName)

           contactNameEditText.*text*.clear()

           loadContacts()

       } else {

           Toast.makeText(this, "Please enter a contact name", Toast.*LENGTH\_SHORT*).show()

       }

   }

   private fun selectContact(position: Int) {

       selectedContact = contactsAdapter.getItem(position) ?: ""

       Toast.makeText(this, "Selected: $selectedContact", Toast.*LENGTH\_SHORT*).show()

   }

   private fun loadContacts() {

       val contacts = contactsHandler.readContacts()

       val contactNames = contacts.*map* **{ it**.name **}**

contactsAdapter.clear()

       contactsAdapter.addAll(contactNames)

       contactsAdapter.notifyDataSetChanged()

   }

}

Contact

data class Contact(

   var name: String = ""

)

ContactsHelper.kt

package com.example.contactssample

import Contact

import android.content.ContentValues

import android.content.Context

import android.provider.ContactsContract

import android.widget.Toast

class ContactsHandler(var context: Context) {

   // Вставка нового контакту

   fun insertContact(name: String) {

       val contentResolver = context.*contentResolver*

val rawContactValues = ContentValues().*apply* **{**

put(ContactsContract.RawContacts.*ACCOUNT\_TYPE*, null as String?)

           put(ContactsContract.RawContacts.*ACCOUNT\_NAME*, null as String?)

**}**

val rawContactUri = contentResolver.insert(ContactsContract.RawContacts.*CONTENT\_URI*, rawContactValues)

       val rawContactId = rawContactUri?.*lastPathSegment*?.*toLongOrNull*() ?: -1L

       if (rawContactId == -1L) {

           Toast.makeText(context, "Error creating contact", Toast.*LENGTH\_SHORT*).show()

           return

       }

       val nameValues = ContentValues().*apply* **{**

put(ContactsContract.Data.*RAW\_CONTACT\_ID*, rawContactId)

           put(ContactsContract.Data.*MIMETYPE*, ContactsContract.CommonDataKinds.StructuredName.*CONTENT\_ITEM\_TYPE*)

           put(ContactsContract.CommonDataKinds.StructuredName.*DISPLAY\_NAME*, name)

**}**

contentResolver.insert(ContactsContract.Data.*CONTENT\_URI*, nameValues)

   }

   // Читання всіх контактів

   fun readContacts(): MutableList<Contact> {

       val contactsList: MutableList<Contact> = *mutableListOf*()

       val contentResolver = context.*contentResolver*

val cursor = contentResolver.query(

           ContactsContract.Data.*CONTENT\_URI*,

*arrayOf*(ContactsContract.CommonDataKinds.StructuredName.*DISPLAY\_NAME*),

           "${ContactsContract.Data.*MIMETYPE*} = ?",

*arrayOf*(ContactsContract.CommonDataKinds.StructuredName.*CONTENT\_ITEM\_TYPE*),

           null

       )

       cursor?.*use* **{**

if (**it**.moveToFirst()) {

               do {

                   val name = **it**.getString(**it**.getColumnIndexOrThrow(ContactsContract.CommonDataKinds.StructuredName.*DISPLAY\_NAME*))

                   contactsList.add(Contact(name))

               } while (**it**.moveToNext())

           }

**}**

return contactsList

   }

   // Видалення контакту за ім'ям

   fun deleteContact(name: String): Int {

       val contentResolver = context.*contentResolver*

// Знайдемо RAW\_CONTACT\_ID для контакту

       val cursor = contentResolver.query(

           ContactsContract.Data.*CONTENT\_URI*,

*arrayOf*(ContactsContract.Data.*RAW\_CONTACT\_ID*),

           "${ContactsContract.Data.*MIMETYPE*} = ? AND ${ContactsContract.CommonDataKinds.StructuredName.*DISPLAY\_NAME*} = ?",

*arrayOf*(ContactsContract.CommonDataKinds.StructuredName.*CONTENT\_ITEM\_TYPE*, name),

           null

       )

       var rawContactId: Long? = null

       cursor?.*use* **{**

if (**it**.moveToFirst()) {

               rawContactId = **it**.getLong(**it**.getColumnIndexOrThrow(ContactsContract.Data.*RAW\_CONTACT\_ID*))

           }

**}**

// Видалення контакту за знайденим ID

       return if (rawContactId != null) {

           val where = "${ContactsContract.RawContacts.*\_ID*} = ?"

           val whereArgs = *arrayOf*(rawContactId.*toString*())

           val result = contentResolver.delete(ContactsContract.RawContacts.*CONTENT\_URI*, where, whereArgs)

           if (result > 0) {

               Toast.makeText(context, "Contact deleted successfully", Toast.*LENGTH\_SHORT*).show()

           } else {

               Toast.makeText(context, "Error deleting contact", Toast.*LENGTH\_SHORT*).show()

           }

           result

       } else {

           Toast.makeText(context, "Contact not found", Toast.*LENGTH\_SHORT*).show()

           0

       }

   }

}

activity\_main.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="match\_parent"

   android:orientation="vertical">

   <ListView

       android:id="@+id/contact\_list\_view"

       android:layout\_width="match\_parent"

       android:layout\_height="match\_parent" />

</LinearLayout>

main\_layout.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="match\_parent"

   android:orientation="vertical"

   android:padding="16dp">

   <EditText

       android:id="@+id/contactNameEditText"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:hint="Введіть ім'я"

       android:inputType="textPersonName"/>

   <Button

       android:id="@+id/saveContactButton"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:text="Зберегти контакт"/>

   <ListView

       android:id="@+id/contactsListView"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:dividerHeight="1dp"/>

</LinearLayout>

menu\_delete.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:title="Delete"

       android:id="@+id/delete"></item>

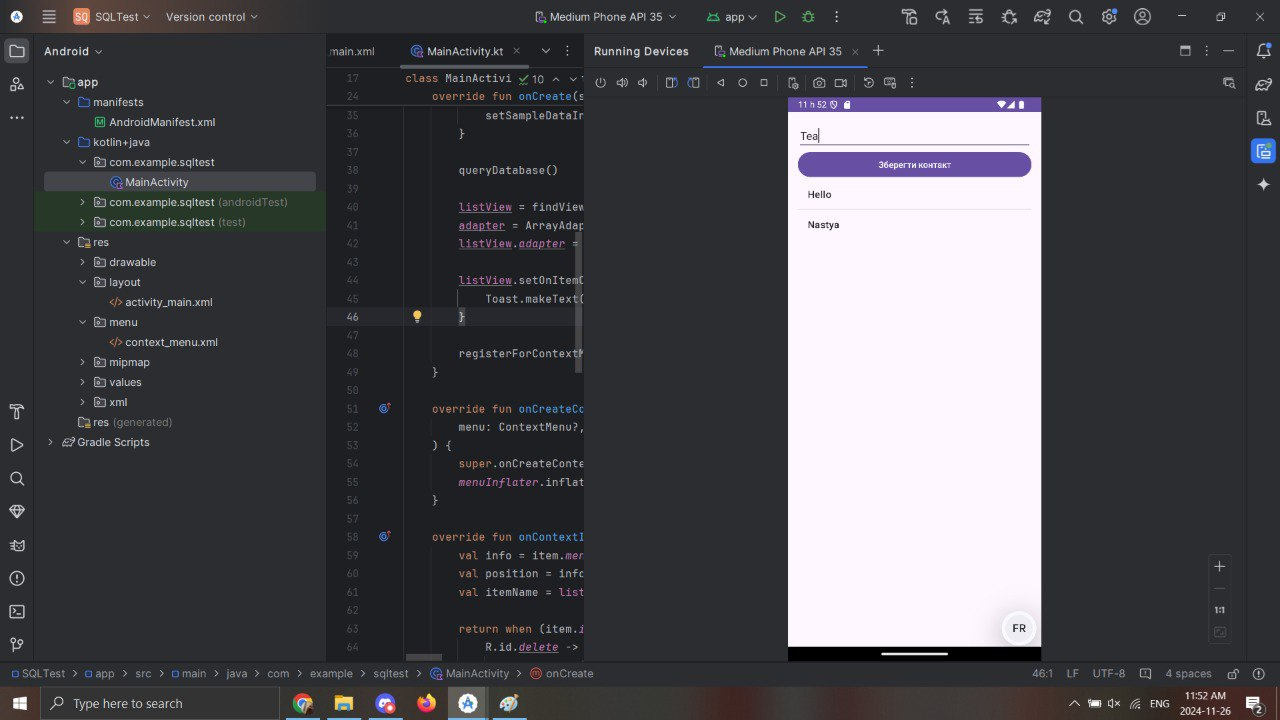
</menu>

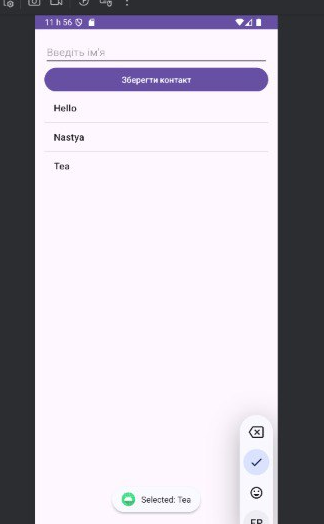
В AndroidManifest.xml добавимо рядки:

<uses-permission android:name="android.permission.READ\_CONTACTS" />

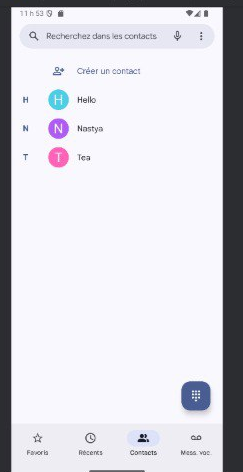
<uses-permission android:name="android.permission.WRITE\_CONTACTS" />

Переглянемо результат роботи:





Перейдемо в контакти на емуляторі:



Зробимо новий проект для перегляду курсу. Назвемо «CurrencyRates»:

package com.example.currencyrates

import android.os.Bundle

import android.util.Log

import android.widget.ListView

import android.widget.SimpleAdapter

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.lifecycle.*lifecycleScope*

import com.example.currencyrates.R

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.launch

import kotlinx.coroutines.withContext

import org.w3c.dom.\*

import java.io.IOException

import java.net.HttpURLConnection

import java.net.URL

import javax.xml.parsers.DocumentBuilderFactory

import org.xml.sax.SAXException

import java.io.InputStream

import java.net.MalformedURLException

import javax.xml.parsers.DocumentBuilder

import javax.xml.parsers.ParserConfigurationException

class MainActivity : AppCompatActivity() {

   companion object {

       private const val KEY\_CHAR\_CODE = "targetCurrency"

       private const val KEY\_VALUE = "exchangeRate"

       private const val KEY\_NAME = "targetName"

   }

   private lateinit var data: ArrayList<Map<String, String>>

   override fun onCreate(savedInstanceState: Bundle?) {

       super.onCreate(savedInstanceState)

       setContentView(R.layout.*activity\_currency\_rates*) // Встановлюємо правильний layout

       populate()

   }

   private fun populate() {

       val thread = Thread **{**

try {

               data = getData()

           } catch (e: Exception) {

               e.printStackTrace()

           }

**}**

thread.start()

       try {

           thread.join()

       } catch (e: InterruptedException) {

           Log.e("CurrencyRates", "Thread interrupted", e)

       }

       // Оновлюємо UI на головному потоці

       runOnUiThread **{**

val from = *arrayOf*(KEY\_CHAR\_CODE, KEY\_VALUE, KEY\_NAME)

           val to = *intArrayOf*(R.id.*charCodeView*, R.id.*valueView*, R.id.*nameView*)

           val sa = SimpleAdapter(*applicationContext*, data, R.layout.*activity\_main*, from, to)

           // Встановлюємо адаптер для ListView

           findViewById<ListView>(R.id.*listView*).*adapter* = sa

**}**

}

   private fun getData(): ArrayList<Map<String, String>> {

       val list = ArrayList<Map<String, String>>()

       try {

           val url = URL(getString(R.string.*ratesurl*))

           val httpConnection = url.openConnection() as HttpURLConnection

           val responseCode = httpConnection.*responseCode*

if (responseCode == HttpURLConnection.*HTTP\_OK*) {

               val inputStream: InputStream = httpConnection.*inputStream*

val dbf = DocumentBuilderFactory.newInstance()

               val db: DocumentBuilder = dbf.newDocumentBuilder()

               val dom: Document = db.parse(inputStream)

               val docElement: Element = dom.*documentElement*

val lastBuildDateNodeList = docElement.getElementsByTagName("lastBuildDate")

               val lastBuildDateElement = lastBuildDateNodeList.item(0) as Element

               val date = lastBuildDateElement.*textContent*

*title* = "$*title* на $date"

               val nodeList: NodeList = docElement.getElementsByTagName("item")

               val count = nodeList.*length*

if (count > 0) {

                   for (i in 0 *until* count) {

                       val entry = nodeList.item(i) as Element

                       val m = HashMap<String, String>()

                       val charCode = entry.getElementsByTagName(KEY\_CHAR\_CODE).item(0).*textContent*

val value = entry.getElementsByTagName(KEY\_VALUE).item(0).*textContent*

val name = entry.getElementsByTagName(KEY\_NAME).item(0).*textContent*

m[KEY\_CHAR\_CODE] = charCode

                       m[KEY\_VALUE] = value

                       m[KEY\_NAME] = name

                       list.add(m)

                   }

               }

           }

       } catch (e: MalformedURLException) {

           e.printStackTrace()

       } catch (e: IOException) {

           e.printStackTrace()

       } catch (e: ParserConfigurationException) {

           e.printStackTrace()

       } catch (e: SAXException) {

           e.printStackTrace()

       }

       return list

   }

}

activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout\_width="fill\_parent"

   android:layout\_height="fill\_parent"

   android:orientation="horizontal">

   <TextView

       android:id="@+id/charCodeView"

       android:layout\_width="wrap\_content"

       android:layout\_height="wrap\_content"

       android:background="#FF8"

       android:minWidth="45sp"

       android:padding="4dp"

       android:textColor="#00F"

       android:textStyle="bold"

       android:gravity="center"

       android:shadowDx="8"

       android:shadowDy="8"

       android:shadowColor="#000"

       android:shadowRadius="8" />

   <TextView

       android:id="@+id/valueView"

       android:layout\_width="wrap\_content"

       android:layout\_height="wrap\_content"

       android:textColor="#008"

       android:background="#FFE"

       android:minEms="3"

       android:padding="3dp" />

   <TextView

       android:id="@+id/nominalView"

       android:layout\_width="wrap\_content"

       android:layout\_height="wrap\_content"

       android:padding="3dp" />

   <TextView

       android:id="@+id/nameView"

       android:layout\_width="wrap\_content"

       android:layout\_height="wrap\_content"

       android:ellipsize="marquee"

       android:singleLine="true" />

</LinearLayout>

activivty\_currency\_rates.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="match\_parent"

   android:orientation="vertical">

   <ListView

       android:id="@+id/listView"

       android:layout\_width="match\_parent"

       android:layout\_height="match\_parent" />

</LinearLayout>

string.xml

<?xml version="1.0" encoding="utf-8"?>

<resources>

   <string name="app\_name"> Курс гривні НБУ</string>

   <string name="ratesurl">https://www.floatrates.com/daily/uah.xml</string>

</resources>

В AndroidManifest.xml добавимо рядки:

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

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

Переглянемо результат роботи:



***Висновок:*** На даній лабораторній роботі я вивчила принципи створення та використання контент-провайдерів в Android. Навчилася управляти даними, забезпечувати до них доступ та розумію їхню роль в архітектурі додатків.

***Контрольні запитання:***

1. ***Для чого потрібен Content Provider?***

Content Provider – це як прошарок між додатком і його даними. Він дозволяє різним додаткам на пристрої можна надавати доступ до даних одного додатка. Можна контролювати, які саме дані і які саме додатки можуть їх бачити та змінювати. Також він надає єдиний спосіб для всіх додатків отримувати та змінювати дані.

1. ***Чи кожен Content Provider надає відкритий URI?***

Ні, не кожен. Якщо ти не хочеш, щоб інші додатки мали доступ до твоїх даних, то можеш не оголошувати URI у маніфесті

1. ***Як задекларувати Content Provider у файлі маніфесту додатку?***

Використовується елемент <provider>, наприклад:

<provider

android:name=".MyContentProvider"

android:authorities="com.example.myapp.provider" />

1. ***Які параметри необхідні для того, щоб зробити запити до Content Provider?***

* URI: Унікальний ідентифікатор для конкретного набору даних.
* Projection: Список колонок, які ти хочеш отримати.
* Selection: Умова для фільтрації даних.
* SelectionArgs: Аргументи для умови.
* SortOrder: Порядок сортування результатів.