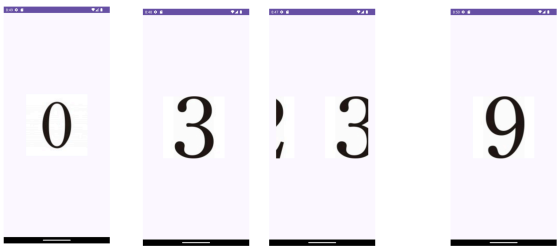
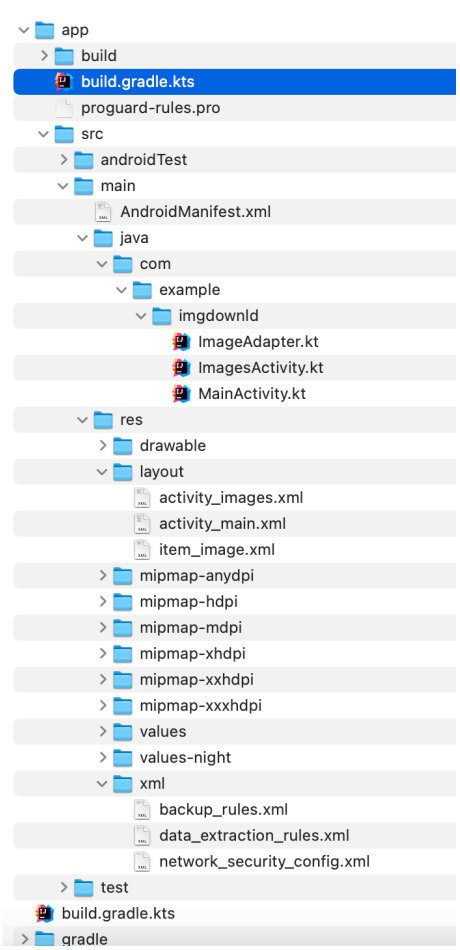
Kotlin co-routines - download more images

La adresa http://cti.ubm.ro/cmo/digits/ exista imaginile img0.jp, img1.jpg, .. img9.jpg

Construiți o aplicație Android folosind co-rutine Kotlin pentru download-ul acestor imagini și afișarea acestora intr-o listă derulantă (scroll orizontal). Aplicația va porni cu o activitate conținând un buton de start, care va lansa o a doua activitate pentru afișarea imaginilor - activitate în care se va determina și timpul total de preluare a imaginilor din rețea.



```
..

..
    recyclerView.adapter = imageAdapter

    loadImages()
}

private fun loadImages() {
    coroutineScope.launch {
        val loadingTime = measureTimeMillis {
            val images = imageUrls.map { url ->
                async { downloadImage(url) }
            }.awaitAll()

            withContext(Dispatchers.Main) {
                imageAdapter.submitList(images)
            }
        }
        withContext(Dispatchers.Main) {
            Log.d(TAG, "Timp de încărcare: $loadingTime ms")

        }
    }
}

private suspend fun downloadImage(url: String): Bitmap? {
    return try {
        val connection = URL(url).openConnection() as HttpURLConnection
        connection.doInput = true
        connection.connect()
        val inputStream = connection.inputStream
        BitmapFactory.decodeStream(inputStream)
    } catch (e: Exception) {
        e.printStackTrace()
        null
    }
}
```




```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
}

android {
    namespace = "com.example.imgdownld"
    compileSdk = 35

    viewBinding {
        enable = true
    }

    defaultConfig {
        applicationId = "com.example.imgdownld"
        minSdk = 33
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

edit files:

build.gradle.kts,

AndroidManifest.xml,

MainActivity.kt,

ImagesActivity.kt,

ImageAdapter.kt,

activity_main.xml, activity_images.xml, item_image.xml,

network_security_config.xml, ..

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.INTERNET" />

    <application
        android:networkSecurityConfig="@xml/network_security_config"
        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.ImgDownld"
        tools:targetApi="31">
        <activity
            android:name=".ImagesActivity"
            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>
```

MainActivity.kt

```
package com.example.imgdownld

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

import com.example.imgdownld.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.loadImagesButton.setOnClickListener {
            val intent = Intent(this, ImagesActivity::class.java)
            startActivity(intent)
        }


    }
}
```

ImagesActivity.kt

```
package com.example.imgdownld

import android.content.ContentValues.TAG
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

import kotlinx.coroutines.*
import java.net.HttpURLConnection
import java.net.URL
import kotlin.system.measureTimeMillis

class ImagesActivity : AppCompatActivity() {

    private val imageUrls = List(10) { "http://cti.ubm.ro/cmo/digits/img${it}.jpg" }
    private val imageAdapter = ImageAdapter()
    private val coroutineScope = CoroutineScope(Dispatchers.IO)

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

        val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)

        // Setează layout-ul pe orizontală
        recyclerView.layoutManager = LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL,
false)

        recyclerView.adapter = imageAdapter

        loadImages()
    }

    private fun loadImages() {
        coroutineScope.launch {
            val loadingTime = measureTimeMillis {
                val images = imageUrls.map { url ->
                    async { downloadImage(url) } // descarcă imaginea pe un alt fir
                }.awaitAll() // așteaptă toate imaginile să fie descărcate

                withContext(Dispatchers.Main) {
                    imageAdapter.submitList(images)
                }
            }
            withContext(Dispatchers.Main) {
                //Toast.makeText(this@ImagesActivity, "Timp încărcare: $loadingTime ms",
Toast.LENGTH_LONG).show()
                Log.d(TAG, "Timp de încărcare: $loadingTime ms")

            }
        }
    }

    private suspend fun downloadImage(url: String): Bitmap? {
        return try {
            val connection = URL(url).openConnection() as HttpURLConnection
            connection.doInput = true
            connection.connect()
            val inputStream = connection.inputStream
            BitmapFactory.decodeStream(inputStream)
        } catch (e: Exception) {
            e.printStackTrace()
            null
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        coroutineScope.cancel() // curățăm toate co-rutinele la distrugerea activității
    }
}
```

ImageAdapter.kt

```
package com.example.imgdownld

import android.graphics.Bitmap
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView


class ImageAdapter : ListAdapter<Bitmap, ImageAdapter.ImageViewHolder>(DiffCallback()) {

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

    override fun onBindViewHolder(holder: ImageViewHolder, position: Int) {
        holder.bind(getItem(position))
    }

    class ImageViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        private val imageView: ImageView = itemView.findViewById(R.id.imageView)

        fun bind(bitmap: Bitmap?) {
            imageView.setImageBitmap(bitmap)
        }
    }

    class DiffCallback : DiffUtil.ItemCallback<Bitmap>() {
        override fun areItemsTheSame(oldItem: Bitmap, newItem: Bitmap) = oldItem == newItem
        override fun areContentsTheSame(oldItem: Bitmap, newItem: Bitmap) = oldItem.sameAs(newItem)
    }
}
```

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">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/kotlin_co_routines_demo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/loadImagesButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="166dp"
        android:layout_marginTop="18dp"
        android:layout_marginEnd="154dp"
        android:layout_marginBottom="286dp"
        android:text="@string/button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

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

activity_images.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=".ImagesActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="358dp"
        android:layout_height="638dp"
        android:layout_marginStart="1dp"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

item_image.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"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:scaleType="fitCenter"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

res/xml/network_security_config.xml,

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">cti.ubm.ro</domain>
    </domain-config>
</network-security-config>
```

res/values/strings.xml,

```
<resources>
    <string name="app_name">ImgDownld</string>
    <string name="kotlin_co_routines_demo">Kotlin co-routines demo(1)</string>
    <string name="button">Start</string>
</resources>
```

Incepând cu Android 9 (API level 28), accesul la traficul HTTP necriptat este dezactivat în mod implicit din motive de securitate.

Dacă domeniul accesat este “de încredere” (în acest caz : `cti.ubm.ro`) se poate configura permisiunea de a accesa doar acel domeniu cu `http` și nu cu `https` (situația implicita) prin adăugarea unei excepții în configurația de rețea a aplicației.

În `AndroidManifest.xml`, se specifică fișierul de configurare pentru securitatea rețelei în secțiunea `<application>:`

```
<application
    android:networkSecurityConfig="@xml/network_security_config"
    ...>
    ...
</application>
```

Iar fișierul de configurare a securității rețelei în folderul `res/xml:`

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">cti.ubm.ro</domain>
    </domain-config>
</network-security-config>
```

`measureTimeMillis` este o funcție din biblioteca standard Kotlin care măsoară timpul de execuție al unui bloc de cod și returnează valoarea acestuia în milisecunde.

Sintaxa este (bloc de cod definit ca o lambda):

```
val time = measureTimeMillis {
    // codul de măsurat
}
```

`View Binding` este o caracteristică introdusă de Android pentru a simplifica accesul la elementele din layout-uri (`views`) în codul Kotlin sau Java, oferind o metodă mai sigură și mai eficientă decât `findViewById`.

Cu `View Binding`, pentru fiecare layout XML din proiect este generată automat o clasă de legătură (binding class), permițându-ți să accesezi elementele de interfață direct și în siguranță, fără casting manual sau riscul de `NullPointerException`.

Avantajele View Binding

1. **Siguranța la compilare**: Accesarea `views` devine mai sigură, deoarece clasele de binding sunt generate automat pentru fiecare layout. Dacă încerci să accesezi un element care nu există în layout-ul respectiv, Android Studio va genera o eroare de compilare.

2. **Fără `findViewById`**: Cu `View Binding`, nu mai este nevoie să folosești `findViewById` pentru a asocia un element din layout cu codul Kotlin/Java. Clasele generate automat includ referințele tuturor elementelor cu `id` din layout.

3. **Reducerea erorilor**: Codul devine mai concis și mai ușor de citit, iar riscul de erori (`NullPointerException` sau `ClassCastException`) este redus semnificativ.

4. **Performanță optimizată**: `View Binding` este mai performant decât metode alternative, cum ar fi `Data Binding` (în cazul în care nu ai nevoie de funcționalități complexe de binding, cum ar fi expresii și legături cu date din XML).

Atunci când activezi `View Binding`, Android Studio generează automat o clasă de binding pentru fiecare layout XML. Numele clasei de binding este format din numele fișierului XML în `PascalCase`, urmat de sufixul `Binding`. De exemplu, pentru un layout numit `activity_main.xml`, va fi generată o clasă `ActivityMainBinding`.

```
// Exemplu de utilizare a View Binding într-o activitate
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

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

        // Inflația layout-ului utilizând binding
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Accesarea elementelor din layout direct prin binding
        binding.textView.text = "Hello, World!"
    }
}
```

Pentru a folosi `View Binding` într-un proiect, trebuie să activezi această opțiune în fișierul `build.gradle` al modulului (`app/build.gradle`). Activarea binding-ului indică Android Studio să genereze automat clasele de binding pentru layout-urile din proiect.

```
android {
    ...
    viewBinding {
        enable = true
    }
}
```

- După ce ai făcut această modificare, sincronizează proiectul (`Sync Now`), iar Android Studio va începe să genereze clasele de binding pentru layout-urile din proiect.