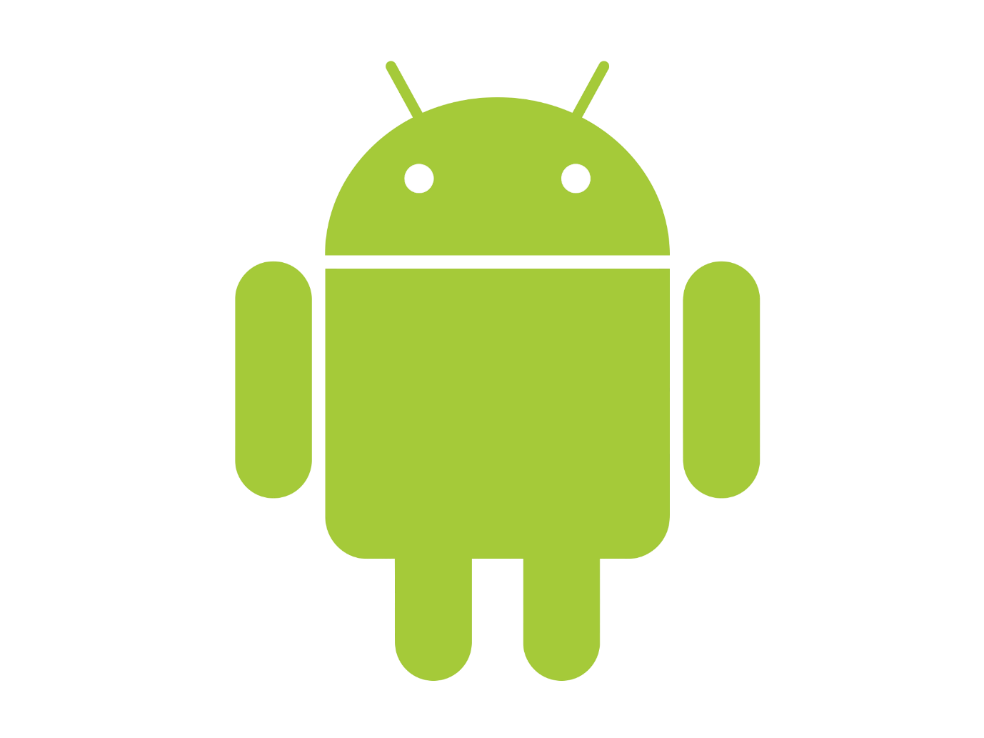
How to Android?

Un ghid conceput de Șerban Sebastian-Mihai

Cuprins

[1. Introducere 3](#_Toc156220494)

[2. Prerequisite 3](#_Toc156220495)

[3. Prima noastră aplicație 7](#_Toc156220496)

[4. Dezvoltarea primei aplicații pentru dispozitivele Android. 11](#_Toc156220497)

[5. Aplicații 19](#_Toc156220498)

[6. Resurse bilbiografice 20](#_Toc156220499)

# Introducere

Android este o platformă software și un sistem de operare conceput pentru dispozivite și telefoane mobile ce se bazează pe kernel-ul Linux. Inițial Android a fost dezvoltat de compania Google, iar ulterior de consorțiul Open Handset Alliance. Android permite developer-ilor să dezvolte aplicații în limbajele de programare Java sau Kotlin, dar și în alte limbaje de programare precum Python sau C++, dar acestea nu sunt recomandate de către Google.

Android este un sistem de operare FOSS – Free and Open Source Software și, la bază, este cunoscut drept AOSP – Android Open Source Project, însă majoritatea dispozitivelor rulează pe versiunea oferită de Google care vine cu anumite servicii Closed Source cum ar fi serviciile GSM – Google Mobile Services.

Android este cel mai utilizat sistem de operare din lume, înregistrând în luna mai a anului 2021 3 miliarde de utilizatori activi. Ultima versiune de Android este Android 14 (4 Octombrie 2023) – Android Upside Down Cake.

# Prerequisite

Pentru a putea dezvolta aplicații pentru dispozitivele ce rulează pe sistemul de operare Android vom avea nevoie de un mediu de dezvoltare în care să dezvoltăm și să testăm aplicațiile. Mediul de dezvoltare recomandat pentru aplicațiile Android este Android Studio, un mediu de dezvoltare ce se află sub egida Google.

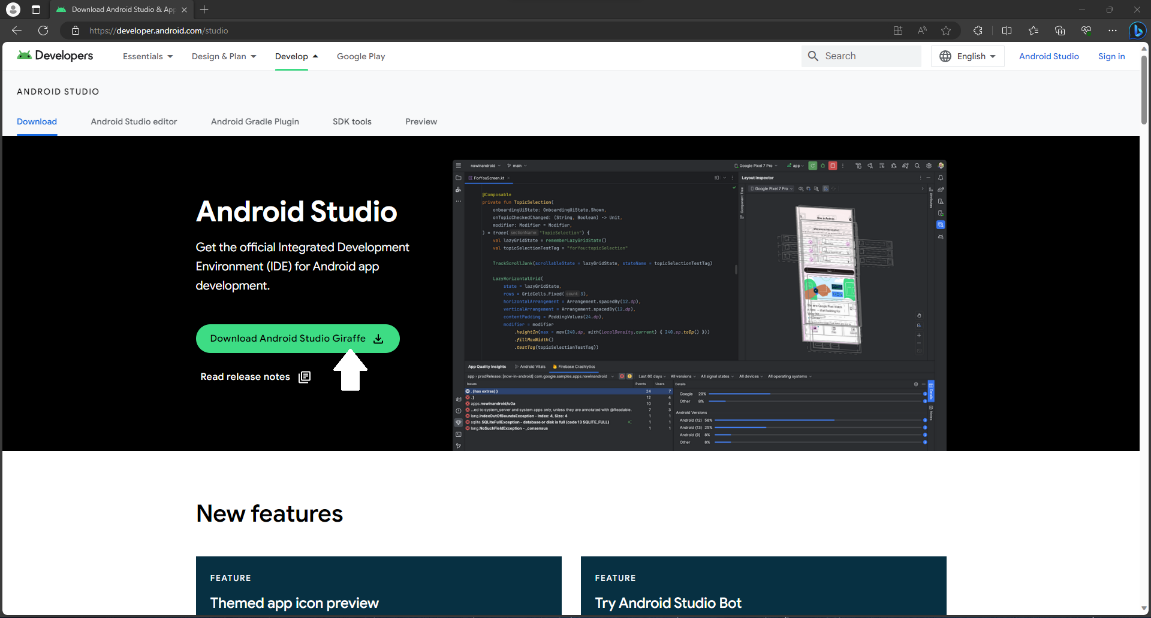
Vom accesa link-ul <https://developer.android.com/studio> de unde vom descărca installer-ul pentru IDE-ul Android Studio. La data la care acest material a fost realizat ultima versiune de Android Studio este Android Studio Giraffe(figura 1). Toți pașii pe care îi vom prezenta în continuare urmează firul instalării pentru dispozitivele ce rulează pe sistemul de operare Windows, însă Android Studio poate fi instalat și pe dispozitive ce rulează pe MacOS, distribuții de Linux sau chiar și ChromeOS.

figura 1

A screenshot of a computer

Description automatically generatedDupă ce installer-ul a fost descărcat, rulăm executabilul pentru a începe instalarea (figura 2).

figura 2

A screenshot of a computer

Description automatically generatedÎnainte de a apăsa pe butonul **Next** ne vom asigura că bifăm căsuța **Android Virtual Device**. Această opțiune va instala și un dispozitiv virtual ce ne va permite să testăm aplicația în emulatorul IDE-ului Android Studio, fără a mai fi nevoie să configurăm noi un dispozitiv virtual sau să conectăm un dispozitiv fizic la computer pentru depanare (figura 3).

figura 3

A screenshot of a computer setup

Description automatically generatedLa următorul pas vom selecta folder-ul în care dorim ca IDE-ul Android Studio să fie instalat. De preferat ar fi ca în calea către fișierul de instalare să nu existe spații pentru a ne asigura că totul decurge bine. Apăsăm butonul **Next** (figura 4).

figura 4

A screenshot of a computer

Description automatically generatedUrmătoare fereastră de dialog ne va da posibilitatea să alegem să nu creem shortcut-uri pentru IDE-ul Android Studio. Aici rămâne la latitudinea fiecărui utilizator să aleagă dacă dorește sau nu shortcut-uri pentru această aplicație. Apăsăm butonul **Install** pentru a începe instalarea propriu-zisă a programului (figura 5).

figura 5

A screenshot of a computer

Description automatically generatedDupă ce apăsăm pe butonul **Install** vom fi întâmpinați de un ecran ce ne va informa despre stadiul instalării pe dispozitiv a IDE-ului Android Studio. După ce instalarea va fi completată vom lăsa activă opțiunea **Start Android Studio** pentru a permite IDE-ului să își finalizeze set-up-ul (figura 6).

figura 6

# Prima noastră aplicație

Acum că IDE-ul Android Studio a fost instalat cu success este momentul să creem primul nostrum proiect în Android Studio și de asemenea și prima noastră aplicație ce va rula pe dispozitive ce rulează pe sistemul de operare Android OS.

A screenshot of a computer

Description automatically generatedPentru a creea un nou proiect deschidem Android Studio și vom fi întâmpinați de următorul chenar de dialog ca cel din figura 7.

figura 7

A screenshot of a computer

Description automatically generatedAlegem opțiunea **New Project** iar aplicația ne va redirecționa către o serie de template-uri care există deja în cadrul IDE-ului. Pentru început vom alege **Empty Activity** (figura 8).

figura 8

A screenshot of a computer

Description automatically generatedÎn următorul chenar de dialog vom alege **numele proiectului**, **calea** la care vrem ca proiectul să fie stocat și de asemenea și **versiunea de Android minimă necesară pentru a rula această aplicație**. După cum bine știm, există dispozitive pe piață ce din diferite considerente rulează versiuni inferioare de Android, iar atunci când construim o aplicație trebuie să luăm în calcul și target-ul device-urilor pe care dorim să le atingem (figura 9).

figura 9

Această opțiune ne oferă posibilitatea de a alege diferite tool-uri care ne pot face dezvoltarea de aplicații pentru Android mai ușoară. Noi vom folosi **Gradle**, un instrument de automatizare al build-ului ce este utilizat pentru a automatiza o gamă largă de sarcini precum testarea, împachetarea, compilarea, etc. Dacă doriți să aflați mai multe despre Gradle vă invit să accesați această pagină web [Gradle | Gradle Build Tool Features](https://gradle.org/features/).

Vom apăsa butonul de **Finish**, iar Android Studio va începe să instaleze toate pachetele, bibliotecile și alte dependințe necesare pentru a putea lansa cu succes primul nostru proiect.

**Urmează să efectuăm un pas facultativ.** În momentul în care vom deschide aplicația Android Studio, ea va veni, în prealabil, cu interfața grafică veche, cea standard pe care și cei de la JetBrains o folosesc. Pe tot parcursul acestui tutorial eu voi folosi noua înterfață grafică pe care o puteți activa urmărind următorii pași: din toolbar-ul din colțul stânga sus alegem opțiunea **File > Settings… > Appearance & Behavior > New UI > Enable new UI**, iar apoi restartăm IDE-ul pentru a se aplica modificările.

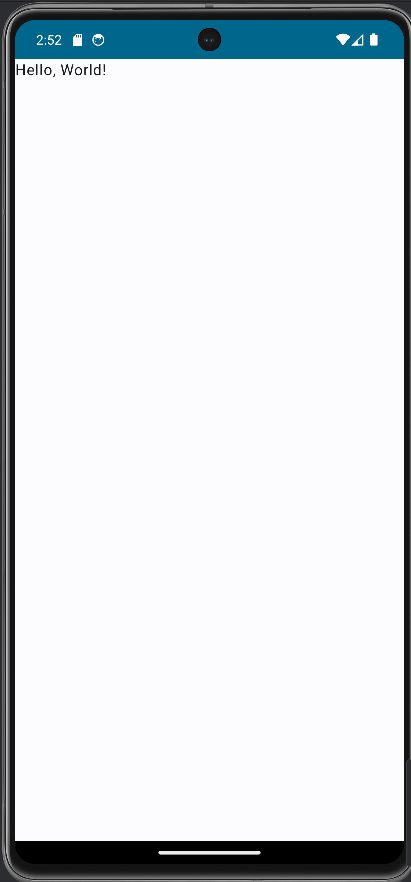
După ce proiectul va finaliza partea de build vom putea rula pe un dispozitiv virtual, sau pe un dispozitiv fizic, o simplă aplicație care ne va întâmpina cu un mesaj de Hello Android precum în figura 10.

figura 10

**Observație:** Pentru a putea încărca pe un dispozitiv fizic este nevoie ca dispozitivul să permită **UBS Debugging** și de asemenea **să fie conectat cu un cablu USB** la dispozitivul care rulează Android Studio.

Pentru a putea începe să dezvoltăm propriu-zis aplicații de Android este nevoie ca mai întâi să înțelegem structura unei aplicații Android.

În structura proiectului pe care tocmai l-am creat putem observa o sumedenie de pachete și fișiere de tipul **.kt**, **.xml** și **.gradle**. O să încercăm în cele ce urmează să explicăm succint importanța fiecărui fișier și pachet din proiect.

Avem următoarea ierarhie de fișiere în proiect, precum în figura 11.

A screenshot of a computer program

Description automatically generated

Observăm că în folderul **app** avem 3 foldere importante, anume **java**, **manifests** și **res**.

* Folderul **java** găzduiește toate clasele ce se ocupă cu funcționalitatea aplicației, de la clasa ce găzduiește Activitatea principală, până la cele ce gestionează fragmente, butoane, recycler-view-uri și alte tool-uri de comunicare cu API-uri sau baze de date.
* Folderul **manifests** găzduiește un fișier foarte important, anume **AndroidManifest.xml**. Prin extensia **.xml** (**Extensible Markup Language**) deja putem intuit pentru ce anume este folosit acest fișier. Acest fișier desenmnează activitatea din care aplicația pornește ca și activitate principală (deoarece, desi nu este recomandat să facem acest lucru, putem avea mai multe activități în cadrul aceleiași aplicații), certificatele pentru conectare la internet, legăturile cu fișierele .xml pentru iconița de start, numele aplicației target API-ul și de asemenea și tema pe care aplicația o utilizează(dacă alegem să definim noi o temă diferită de cea predefintă de către IDE).

figura 11

* Folderul **res** conține toate informațiile legate de butoane, fragmente, iconițe, constante, teme, toate acestea fiind stocate în fișiere .xml. De asemenea, în acest folder va fi găzduit și navgraph-ul (vom afla mai multe detalii despre acest feature în curând).

# Dezvoltarea primei aplicații pentru dispozitivele Android.

Acum că am văzut structura unei aplicații de tipul „Hello, World!”, putem să trecem la dezvoltarea propriu-zisă a unei aplicații. Pentru acest lucru vom încerca să creem o aplicație care va avea următoarele funcționalități: Aplicația ne va întâmpina cu un ecran ce va conține un buton. Aceste buton ne va redirecționa spre o listă cu toate țările lumii, alături de steagul lor și regiunea din care provin.

Pentru a realiza această aplicație va trebui să introducem două noi lucruri, **graful de navigație, recycler-view**-ul și **Retrofit** pentru a face apeluri spre Repo-ul de unde ne vom lua datele.

Majoritatea aplicațiilor de Android din zilele noastre folosesc design pattern-ul Single Activity app. Acest design se bazează pe utilizarea unei singure activități în cadrul întregii aplicații pentru a permite acesteia să ruleze mult mai ușor și mai fluent pe dispozitive. Totuși, avem nevoie ca interfața să fie constant actualizată cu opțiunile pe care utilizatorul le allege. Pentru a face acest lucru ne vom folosi de **fragmente**, ele fiind de fapt un overlay ce ne permite updatarea ecranului fără a schimba activitatea.

Aceste activități au nevoie de o ordine logică de desfășurare pe care aplicația să o poată urmări de la momentul în care a fost lansată până în momentul în care am încheiat execuția aplicației. Această ordine, succesiune a fragmentelor, este definită și utilizată prin intermediul **NavGraph-ului**. Acest tool specific dezvoltării pentru Android funcționează pe ideea unei structuri arborescente.

De asemenea, atunci când dorim să afișăm o listă cu diferite obiecte în ea, by default sistemul de operare Android încarcă întreaga listă și în momentul în care noi o navigăm el creează pentru absolut fiecare obiect un view pe care să îl afișeze. Acest lucru este foarte costisitor cu resursele dispozitivului și de multe ori duce la o aplicație ce face dispozitivul să funcționeze foarte greu.

Pentru a ne veni în ajutor, sistemul de operare Android are o unealtă numită **recycler-view** care ne permite să încărcăm dinamic conținutul unei liste și, cel mai important, creează **un singur view** pe care îl **refolosește pe tot parcursul navigării**, schimbând doar datele din interiorul obiectelor în funcție de preferințe. Acest lucru face ca aplicația să opereze mult mai smooth, iar stresul pus pe componentele hardware ale dispozitivului este minimizat considerabil.

**Retrofit** este o bibliotecă ce ne ajută să gestionăm cererile de tip HTTP către un API. De ce am dori să folosim această bibliotecă? Ei bine, nu ne dorim să hardcode-ăm toate țările lumii și de asemenea țările existente, recunoscute sau nu, se modifică de-a lungul timpului. Cu ajutorul Retrofit vom reuși să apelăm API-ul REST Countries PE care conține o gamă de request-uri http care returnează diferite răspunsuri sub formă de fișiere de tip **.json** .

Pentru început vom creea un proiect nou cu template-ul **Basic Fragment** în care vom descrie funcționalitatea aplicației. Primul lucru pe care trebuie să îl facem în momentul în care proiectul este genera teste acela de a adăuga dependițele în fișierul build:gradle.

A screenshot of a computer program

Description automatically generatedVom adăuga următoarele dependințe:

Aceste dependințe ne asigură că toate bibliotecile pe care le vom importa de-a lungul proiectului vor fi importate de către instrumentul de lucru gradle. După adăugarea acestor dependințe vom fi nevoiți să resincronizăm proiectul, lucru pe care IDE-ul ni-l recomandă de fiecare dată când se modifică vreun parametru în fișierul buil.gradle. Această sincronizare îi va permite gradle-ului să importe toate bibliotecile pentru a le putea folosi.

Acum că build-ul aplicației a reușit, trebuie să facem următorul lucru foarte important, acela de a oferi permisiunea aplicației de a folosi internetul, pentru a putea face request-uri către API-ul care ne va oferi informațiile despre țări. Intrăm în fișierul AndroidManifest.xml și adăugăm urmatoarea linie de cod:

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

După ce am făcut și acest pas, putem să trecem mai departe la a scrie cod în limbajul de programare Kotlin. În continuare, creem un pachet nou cu numele **Model** iar în acest pachet vom creea fișierul **Country.kt** care va conține toate datele de care avem nevoie despre fiecare țară în parte. Desigur, puteți adăuga mai multe proprietăți A screen shot of a computer program

Description automatically generatedale clasei, în funcție de ce funcționalități doriți să implementați.

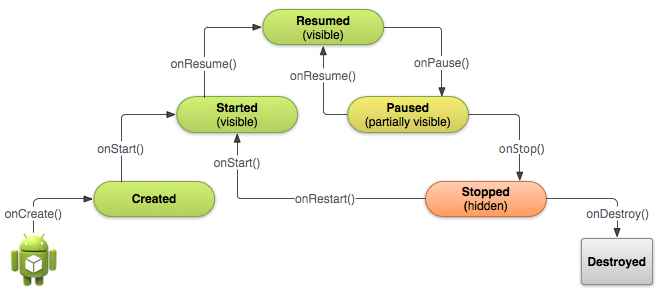
A screen shot of a computer

Description automatically generatedAcum că avem clasa serializabilă, putem să creem pachetul **resource** în care vom creea interfața **CountryResource.kt** care va avea un singur rol, acela de a descrie funcția care va trimite un request de tip GET spre api-ul pe care îl folosim. Request-urile GET sunt folosite atunci când clientul dorește să obțină informații de la server. Acestea nu ar trebui să modifice starea serverului și nu ar trebui să aibă efecte secundare. De exemplu, când accesezi o pagină web într-un browser, browserul face de obicei un request GET pentru a obține conținutul paginii.

Următorul lucru pe care îl vom face este să modificăm clasa **MainActivity.kt**. În ansamblu, această clasă reprezintă activitatea principală a aplicației și utilizează Data

A screen shot of a computer program

Description automatically generatedBinding pentru a gestiona interacțiunea cu elementele de interfață definite în fișierul XML al layout-ului activității.

Următoarea clasă pe care o vom creea este cea care va defini funcționalitatea fragmentului principal care ne va întâmpina. După cum am văzut și în cadrul activității, noi suprascriem anumite funcții care au diferite nume. În clasa anterioară am suprascris metoda **onCreate()**.De ce facem asta? Ei bine, fragmentele și activitățile, principalele blocuri de construcție într-o aplicație Android, au un anumite ciclu de viață. Acest ciclu de viață are câte o metodă alocată și urmărește următoarea schemă:

1. **onCreate()**: Metoda este apelată atunci când activitatea sau fragmentul este creat. Aici, se realizează inițializări generale, cum ar fi setarea layout-ului sau inițializarea datelor.
2. **onStart()**: Activitatea devine vizibilă pentru utilizator. În acest punct, aceasta poate începe să interacționeze cu utilizatorul.
3. **onResume()**: Activitatea devine activă și primește focusul. Este momentul ideal pentru a începe actualizarea interfeței de utilizator sau pentru a iniția conexiuni cu servicii.
4. **onPause()**: Acesta este primul stadiu în care activitatea își pierde focusul. În această etapă, sunt efectuate acțiuni de salvare a datelor sau de eliberare a resurselor.
5. **onStop()**: Activitatea nu mai este vizibilă. În această fază, pot fi efectuate acțiuni pentru economisirea energiei și eliberarea resurselor neutilizate.
6. **onDestroy()**: Acesta este ultimul stadiu, indicând că activitatea sau fragmentul va fi distrus. În această fază, resursele trebuie eliberate și conexiunile trebuie închise.
7. **onSaveInstanceState(Bundle outState)**: Este apelată înainte ca activitatea sau fragmentul să fie distrus, oferind șansa de a salva datele importante pentru a fi restaurate ulterior.
8. **onCreateView()** (doar pentru fragmente): Metoda este responsabilă pentru crearea ierarhiei de vizualizare asociate fragmentului.
9. **onViewCreated()** (doar pentru fragmente): Apelată după ce vizualizarea asociată fragmentului a fost creată, permitând inițializarea interfeței de utilizator.

Acum că ne-am format o idee despre ciclul de viață al unei componente în Android, putem creea primul fragment, cel ce va conține un buton care ne va redirecționa spre lista tuturor țărilor din lume.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generatedPentru că tot am început cu fragmentele, să continuăm în aceeași manieră, creând clasa **AllCountriesFragment.kt** care va avea drept scop să apeleze api-ul, să încarce datele primite într-o listă și să le așeze în pagină astfel încât să putem scroll-a prin listă. În mare parte avem aceleași metode, însă există o metodă pe care aș dori să o discutăm mai pe larg.

Metoda **setupCountries()** ar drept scop să creeze un obiect de tip json care ne va permite să parsăm răspunsul pe care îl vom primi de la API. Valorea **ignoredUnknowKeys** este setată pe **false** pentru a ne permite să omitem informațiile pe care fișierul json le va conține însă noi nu le vom reține în cadrul unui obiect de tip Country. După cum spuneam și mai sus, vom folosi retrofit pentru a face apelul de tip GET către API.

Încă un lucru foarte important este că în momentul în care facem acțiuni de tipul Input/Output este de preferat să paralelizăm acest lucru. Kotlin folosește corutinile pentru a face acest lucru. Nu vom intra în detalii, pentru mai multe informații vă ofer drept material de studiu următorul link [Kotlin coroutines on Android  |  Android Developers](https://developer.android.com/kotlin/coroutines#:~:text=A%20coroutine%20is%20a%20concurrency,established%20concepts%20from%20other%20languages.).

În interiorul corutinei de tip Main nu facem altceva decât să încărcăm totalitatea obiectelor de tip Country pe care le-am deserializat și pe care dorim să le afișăm pe ecranul telefonului.

Observăm că în interiorul corutinei Main folosim clasa **BasicAdapter**. Această clasă nu este predefinită în limbajul Kotlin, deci va trebui să o descriem noi, însă va extinde clasa RecyclerVIew.Adapter.

**RecyclerView.Adapter** este o clasă de bază în Android utilizată pentru a conecta seturile de date la un **RecyclerView**. **RecyclerView** este un component de interfață utilizator care afișează o listă sau un grup de elemente într-un mod eficient și flexibil. **RecyclerView.Adapter** funcționează ca un adaptor între datele tale și **RecyclerView**, gestionând modul în care acestea sunt afișate și interacționează cu utilizatorul.

A screenshot of a computer screen

Description automatically generatedPentru a putea folosi un recycler view și pentru a descrie totalitatea interfețelor de utilizator pe care le vom folosi pe tot parcursul rulării, este nevoie să creem fișiere XML care să descrie exact cu dorim să arate aceste interfețe.

Un exemplu este fișierul recycler\_view.xml care conține elementul de bază al listei ce ne va afișa toate țările cu steagurile și regiunea lor.

Restul fișierelor XML le veți putea descărca din repository-ul de github pe care îl voi atașa la finalul acestei documentații.

Tot ce ne mai rămâne de făcut acum, după ce am descris toate XML-urile și am terminat de implementat clasele, fragmentele și activitățile, este să construim aplicația și să o rulăm fie pe un telefon ce rulează sistemul de operare Android, fie pe emulatorul pe care IDE-ul Android Studio ni-l oferă.

# Aplicații

Acum că am dobândit toate cunoștințele necesare pentru a putea dezvolta o aplicație simplă pentru Android este timpul să încercăm să creem prima aplicație care va avea cu adevărat utilitate pentru noi.

M-am gândit că o aplicație de notițe ar fi utilă pentru a ne exersa cunoștințele și pentru a reuși să aprofundăm tot ceea ce am învățat pe parcursul acestei lucrări.

Aplicația trebuie să îndeplinească următoarele cerințe:

* Trebuie să permită stocarea notițelor într-o bază de date
* Trebuie să permită creearea de noi notițe
* Trebuie să ofere notițelor un titlu, un subtitlu și un conținut
* Trebuie să permită editarea notițelor deja existente
* Trebuie să permită căutarea după titlu a unei anume notițe

Un exemplu de aplicație de notițe se va regăsi tot în repository-ul de pe github, în cazul în care aveți nevoie de inspirație.

Link repository: [SerbanSebastian/AndroidApps (github.com)](https://github.com/SerbanSebastian/AndroidApps)

# Resurse bilbiografice

* [ChatGPT (openai.com)](https://chat.openai.com/)
* [Kotlin and Android  |  Android Developers](https://developer.android.com/kotlin?gclid=CjwKCAiAqY6tBhAtEiwAHeRopSQCLk0FLetR-SlIbXUUa16qevX9ZcdHIoF5zbgTadMtBhtRson1fxoCHHkQAvD_BwE&gclsrc=aw.ds)
* [Android Activity Lifecycle - javatpoint](https://www.javatpoint.com/android-life-cycle-of-activity)
* [GeeksforGeeks | A computer science portal for geeks](https://www.geeksforgeeks.org/)