Programarea aplicațiilor pe telefonul mobil

Student: Ioana Fazacaș

Proiect la Structura Sistemelor de Calcul

Universitatea Tehnica din Cluj-Napoca

2024

Cuprins

I. Introducere

1.1 Motivație & obiective

a) Motivație

Am ales acest proiect deoarece pe parcursul verii m-am interesat de procesul de creare al jocurilor de mobile destinate telefoanelor cu sistemul de operare Android. Am incercat sa creez eu singura un joc in Java care să poate fi jucat de pe Android , dar din nefericire nu am reușit să îl duc la finalizare.

b) Obiective

Obiectivul acestui proiect este crearea unui ghid care explică bazele realizării unui program pentru telefoane mobile cu sistem de operare Android (si facilitatile de programare oferite de acest SO). Pentru o mai ușoară înțelegere a realizării aplicațiilor , ghidul conține implementarea a 2 programe pentru telefoane mobile cu nivele de dificultate diferite. Limbajul de programare de bază folosit este Java.

1.2 Domeniu & context

Aplicatiile au fost initial destinate asistentei in materie de productivitate, cum ar fi e-mail, calendar si baze de date de contact, dar cererea publica pentru aplicatii a determinat extinderea rapida in alte domenii, cum ar fi jocurile mobile, automatizarea productiei, GPS si servicii bazate pe locatie, urmarirea comenzilor si achizitiilor, astfel incat exista acum milioane de aplicatii disponibile.

Tipuri

Aplicatiile mobile pot fi clasificate prin numeroase metode. Acestea pot fi puse în 3 categorii : aplicatiile native, hibride si bazate pe web.

- **Aplicatii native**:
  Toate aplicatiile concepute pentru o anumita platforma mobila sunt cunoscute sub numele de aplicatii native. Prin urmare, o aplicatie destinata dispozitivului Apple nu ruleaza pe dispozitivele Android.
  Scopul principal pentru crearea unor astfel de aplicatii este de a asigura cea mai buna performanta pentru un anumit sistem de operare mobil.

- **Aplicatie hibrida** :
  Conceptul aplicatiei hibride este un amestec de aplicatii native si bazate pe web. Aplicatiile dezvoltate folosind Apache Cordova, Xamarin, React Native, Sencha Touch si alte tehnologii similare se incadreaza in aceasta categorie.
  Acestea sunt create pentru a sprijini tehnologiile web si native, pe mai multe platforme. Mai mult, aceste aplicatii sunt mai usor si mai rapid de dezvoltat. Aceasta implica utilizarea bazei de cod unic care functioneaza pe mai multe sisteme de operare mobile. In ciuda acestor avantaje, aplicatiile hibride prezinta performante mai mici. Adesea, aplicatiile nu au acelasi aspect in diferite sisteme de operare mobile.

- **Aplicatie bazata pe web :**
  O aplicatie bazata pe web este codata in HTML5, CSS sau JavaScript. Accesul la internet este necesar pentru comportamentul adecvat si experienta utilizatorului acestui grup de aplicatii.
  Aceste aplicatii pot necesita un spatiu de memorie minim pe dispozitivele utilizatorului in comparatie cu aplicatiile native si hibride. Deoarece toate bazele de date personale sunt salvate pe serverele de Internet, utilizatorii pot prelua datele dorite de pe orice dispozitiv, prin Internet.

Dezvoltare

Dezvoltarea aplicatiilor pentru dispozitive mobile necesita luarea in considerare a constrangerilor si caracteristicilor acestor dispozitive. Dispozitivele mobile functioneaza cu baterie si au procesoare mai putin puternice decat computerele personale.

Dezvoltatorii trebuie, de asemenea, sa ia in considerare o gama larga de dimensiuni de ecran, specificatii hardware si configuratii, din cauza concurentei intense in software-ul mobil si a modificarilor care apar in cadrul unei anumite platforme.

Dezvoltarea aplicatiilor mobile necesita utilizarea unor medii de dezvoltare integrate specializate. Aplicatiile mobile sunt testate mai intai in mediul de dezvoltare folosind emulatoare si ulterior sunt supuse testarii pe teren. Emulatoarele ofera un mod ieftin de a testa aplicatiile de pe telefoanele mobile la care este posibil ca dezvoltatorii sa nu aiba acces fizic.

Distributie

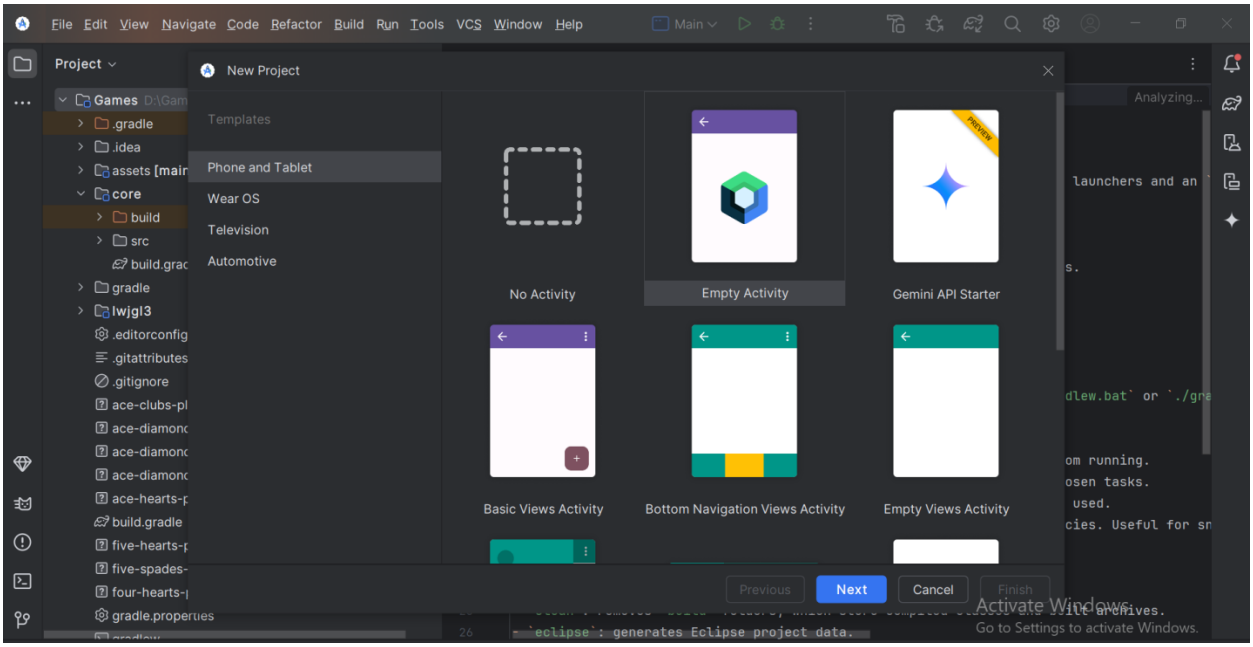
Cele mai mari trei magazine de aplicatii sunt Google Play pentru Android, App Store pentru iOS si Microsoft Store pentru Windows 10, Windows 10 Mobile si Xbox One.

II. Hello, @name!

“Hello, @name” este o aplicatie care te saluta folosind numele tau in formula de salut.

1. Deschidem aplicația și dăm click pe *New Project*

2. Fereastra *New Project* se deschide cu o listă de șabloane oferite de Android Studio.



În Android Studio, un șablon de proiect este un proiect Android care oferă planul pentru un anumit tip de aplicație. Șabloanele creează structura proiectului și fișierele necesare pentru ca Android Studio să construiască proiectul. Șablonul pe care îl alegi oferă codul de început pentru a te ajuta să începi mai rapid.

3. Asigură-te că fila *Phone and Tablet* este selectată.

4. Apasă pe șablonul *Empty Activity* pentru a-l selecta ca șablon pentru proiectul tău. Are un singur ecran și afișează textul „Hello Android!”.

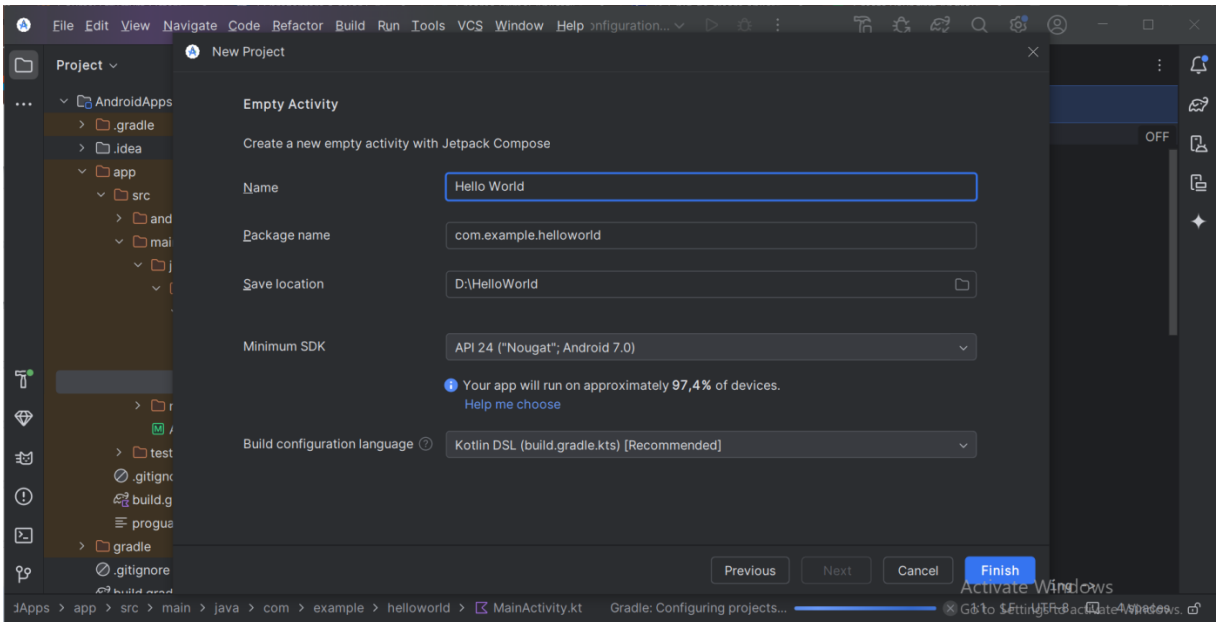
5. Apasă *Next* (Următorul). Dialogul *New Project* se deschide. Acesta are câmpuri pentru configurarea proiectului.

6. Configurează proiectul astfel: Câmpul *Name* (Nume) este folosit pentru a introduce numele proiectului. Pentru acest codelab, scrie „Greeting Card”.

Lasă câmpul *Package name* (Nume pachet) neschimbat. Acesta este modul în care fișierele tale vor fi organizate în structura fișierelor. În acest caz, numele pachetului va fi com.example.greetingcard.

Lasă câmpul *Save location* (Locație de salvare) neschimbat. Acesta conține locația unde sunt salvate toate fișierele legate de proiectul tău. Notează-ți unde se află aceasta pe computer, pentru a-ți găsi fișierele.

Selectează *API 24: Android 7.0 (Nougat)* din meniu pentru câmpul *Minimum SDK*. Minimum SDK indică versiunea minimă de Android pe care aplicația ta poate rula.



7. Apasă *Finish* (Finalizare). Aceasta poate dura un pic - este un moment potrivit pentru a-ți lua o ceașcă de ceai! În timp ce Android Studio îți configurează proiectul, o bară de progres și un mesaj indică stadiul configurării.

8. Este posibil să vezi un panou *What's New* care conține actualizări despre funcționalități noi în Android Studio. Închide-l deocamdată.

Assistant What's New

What's New in Iguana

This panel describes some of the new features and behavior changes included i update.

To open this panel again later, select **Help > What's New in Android Studio** from main menu.

Read in a browser

Version control system integration in App Quality Insights

Use **App Quality Insights** to navigate from a Crashlytics stack trace to the relev code—at the point in time when the crash happened. When you view crash repo choose to navigate to the line of code in your current git checkout or view a diff between the current checkout and the version of your codebase that generated crash.

Learn more

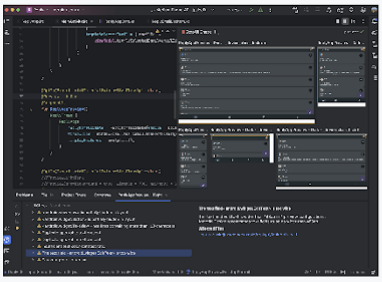
View Crashlytics crash variants in App Quality Insights

To help you analyze the root causes of a crash, use **App Quality Insights** to viev by issue *variants*, or groups of events that share similar stack traces.

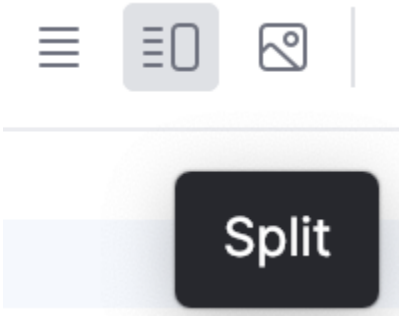
Learn more

UI Check Mode in Compose Preview

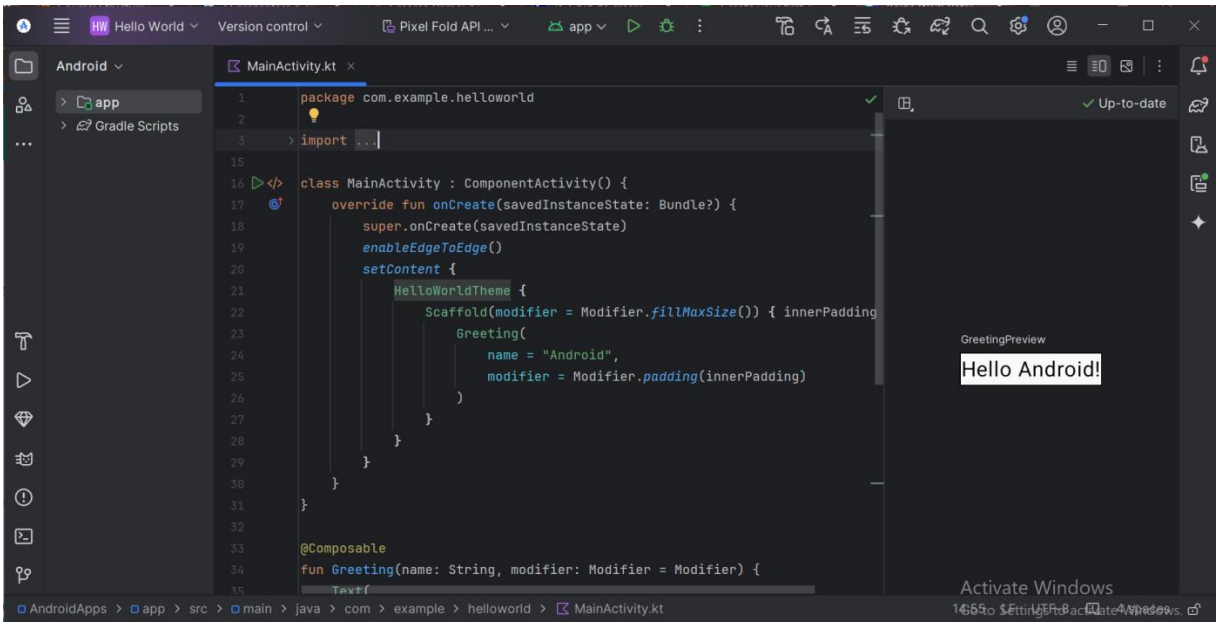
To help developers build more adaptive and accessible UI in Jetpack Compose, Studio Iguana Canary 5 introduced a new UI Check mode in Compose Preview. feature works similar to Visual Linting and Accessibility checks integrations for v Activate Compose UI check mode for Android Studio to automatically audit your Compose UI and check for adaptive and accessibility issues across different scr sizes, such as text stretched on large screens and low color contrast. The mode highlights issues found in different Preview configurations and lists them in the problems panel.



9. Apasă *Split* în colțul din dreapta sus al Android Studio, ceea ce îți permite să vezi atât codul, cât și designul. Poți, de asemenea, să apeși pe *Code* pentru a vizualiza doar codul sau pe *Design* pentru a vizualiza doar designul.

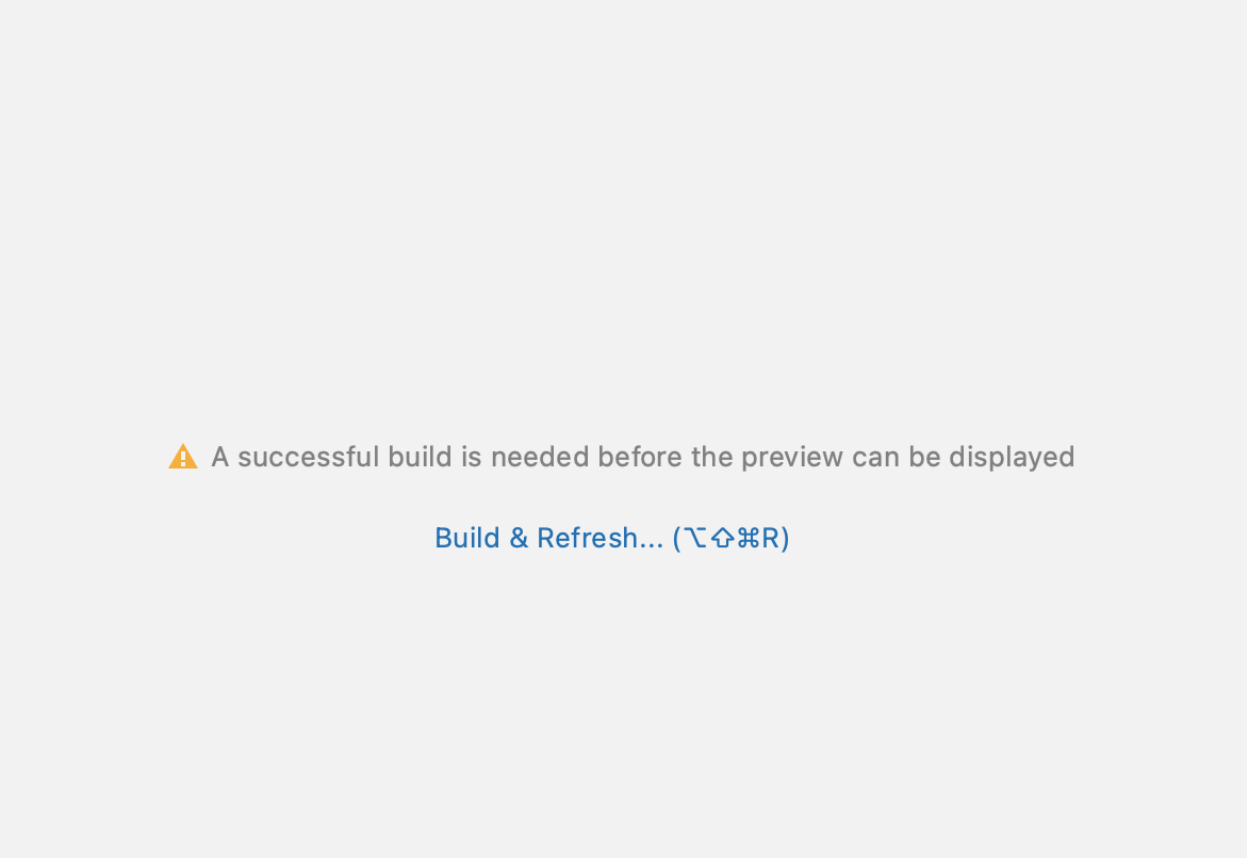


După ce ai apăsat *Split*, ar trebui să vezi trei zone:



- *Project view* (1) afișează fișierele și directoarele proiectului tău
- *Code view* (2) este locul unde editezi codul
- *Design view* (3) este locul unde previzualizezi cum arată aplicația ta

În *Design view*, poți vedea un panou gol cu acest text:

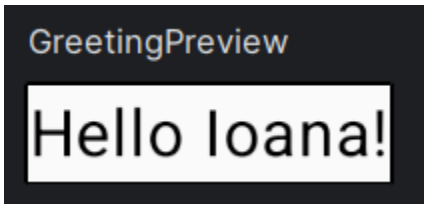


10. Apasă *Build & Refresh*. Este posibil să dureze ceva timp pentru a construi, dar când se finalizează, previzualizarea va afișa o casetă de text care spune „Hello Android!”.

11. Modificarea textului din „Hello Android”! in „Hello <numele meu>”

Trebuie actualizată metoda *GreetingPreview()* cu numele tau propriu. Schimbă parametrul funcției *Greeting(String name)* cu un string ce contine numele tău. Apoi efectueaza din nou rebuild și verifica rezultatul obținut.

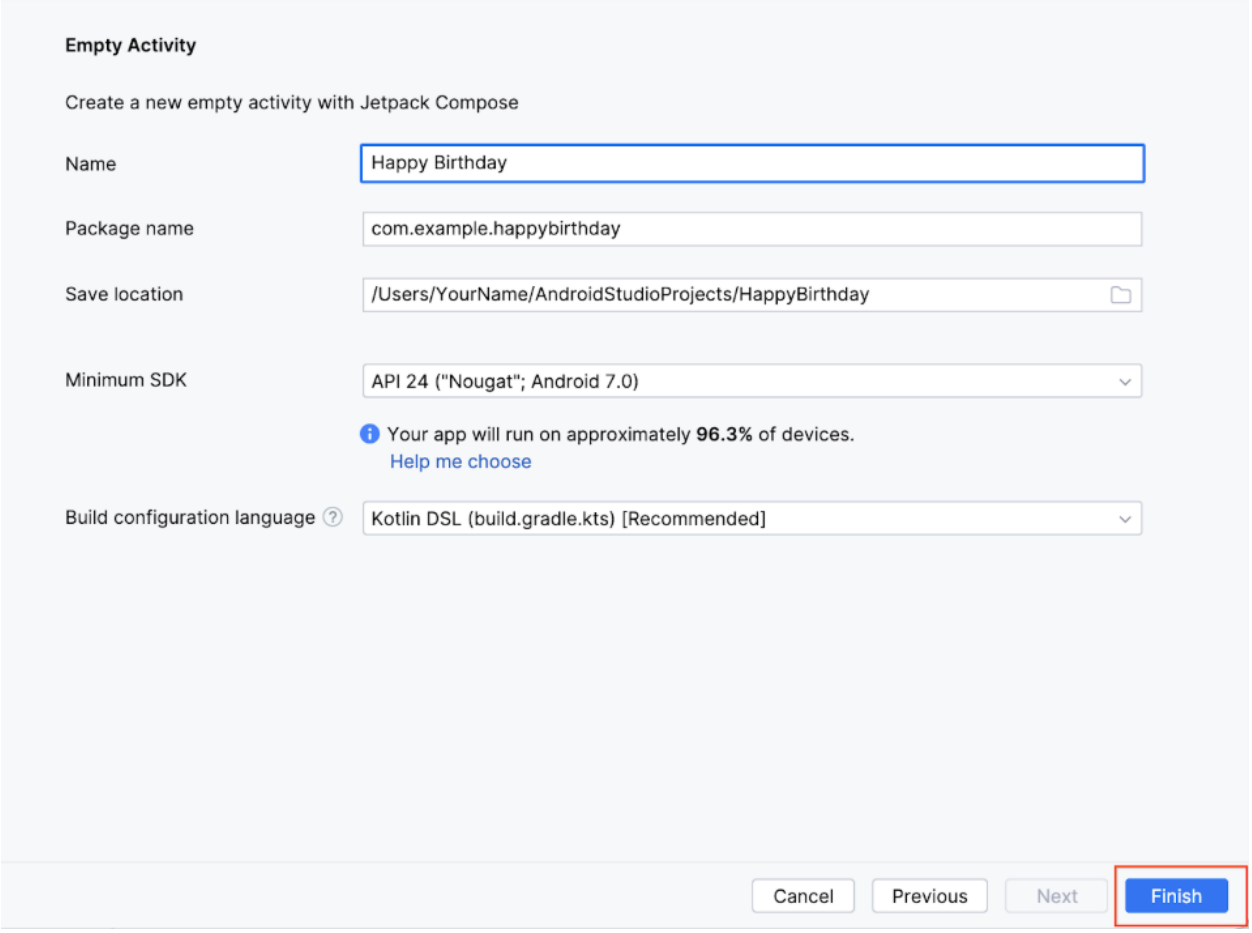
```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    MyApplicationTestTheme {
        Greeting( name: "Ioana")
    }
}
```



III. Happy Birthday Card App

Aceasta apliatie reprezinta o felicitare personalizata de ‘La multi ani, @name !’ cu imagine de fundal si nume personalizat , care are de asemenea atasata un cantec de la multi ani care poate fi redat sau oprit prin intermediul unor butoane.

1. În dialogul **Welcome to Android Studio**, selectați **New Project**.
2. În dialogul **New Project**, selectați **Empty Activity** și apoi faceți clic pe **Next**.
3. În câmpul **Name**, introduceți **Happy Birthday**, apoi selectați un nivel minim de API de **24 (Nougat)** în câmpul **Minimum SDK** și faceți clic pe **Finish**.



4. Așteptați ca Android Studio să creeze fișierele proiectului și să construiască proiectul.
5. Faceți clic pe **Run 'app'.**

In aceasta aplicatie ne vom folosi de Jetpack Compose pentru crearea partii de UI

Jetpack Compose este un toolkit modern pentru crearea interfețelor UI pe Android. **Compose** simplifică și accelerează dezvoltarea UI pe Android prin mai puțin cod, instrumente puternice și capabilități intuitive ale limbajului Kotlin. Cu Compose, puteți construi interfața UI definind un set de funcții, numite **funcții compozabile**, care primesc date și descriu elementele UI.

Funcții compozabile

Funcțiile compozabile sunt blocurile de bază ale unui UI în Compose. O funcție compozabilă:

- Descrie o parte a interfeței UI.
- Nu returnează nimic.
- Primește anumite date ca input și generează ceea ce este afișat pe ecran.

Funcția compozabilă este adnotată cu **@Composable**. Toate funcțiile compozabile trebuie să aibă această adnotare. Aceasta informează compilatorul Compose că funcția este destinată să convertească datele în elemente UI.

```
@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}
```

- **Jetpack Compose** este construit în jurul funcțiilor compozabile. Aceste funcții vă permit să definiți interfața aplicației programatic, descriind cum ar trebui să arate, mai degrabă decât să vă concentrați pe procesul construcției UI. Pentru a crea o funcție compozabilă, adăugați pur și simplu adnotarea **@Composable** la numele funcției.
- Funcțiile compozabile pot accepta argumente, care permit logica aplicației să descrie sau să modifice UI-ul. În acest caz, elementul UI acceptă un **String** pentru a putea saluta utilizatorul după nume.

6. În Android Studio, deschideți fișierul **MainActivity.kt**.
7. Derulați la funcția **GreetingPreview**(). Această funcție compozabilă ajută la previzualizarea funcției **Greeting**(). Ca bună practică, funcțiile ar trebui să fie întotdeauna denumite sau redenumite astfel încât să descrie funcționalitatea lor. Schimbați numele acestei funcții în **BirthdayCardPreview**().

Adnotări

Adnotările sunt mijloace de atașare a informațiilor suplimentare la cod. Aceste informații ajută instrumentele precum compilatorul Jetpack Compose și alți dezvoltatori să înțeleagă codul aplicației.

8. În fișierul **MainActivity.kt**, ștergeți definiția funcției **Greeting()**. Veți adăuga propria funcție pentru a afișa mesajul de salut în cadrul codelab-ului mai târziu.
9. Ștergeți apelul funcției **Greeting()** împreună cu argumentele sale din funcțiile **onCreate()** și **BirthdayCardPreview()**.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //enableEdgeToEdge()
        setContent {
            HappyBirthdayTheme {
                Surface (
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
```

```
                ){

                }
            }
        }
    }
}
```

10. Înainte de funcția **BirthdayCardPreview**(), adăugați o funcție nouă numită **GreetingText**(). Nu uitați să adăugați adnotarea **@Composable** înaintea funcției, deoarece aceasta va fi o funcție compozabilă care descrie un Text compozabil.
11. Este o bună practică ca funcțiile compozabile să accepte un parametru **Modifier** și să îl transmită primului copil. Veți învăța mai multe despre **Modifier** și elementele copil în sarcinile și codelab-urile următoare.

```
@Composable
fun GreetingText(modifier: Modifier = Modifier){

}
```

12. Adăugați un parametru **message** de tip **String** în funcția compozabilă **GreetingText()**.

```
@Composable
fun GreetingText(message: String, modifier: Modifier = Modifier){

}
```

13. În funcția **GreetingText**(), adăugați un Text compozabil, transmițând mesajul text ca argument numit.

```
@Composable
fun GreetingText(message: String, modifier: Modifier = Modifier){
    Text(
        text = message
    )
}
```

Această funcție **GreetingText**() afișează textul în UI. Face acest lucru apelând funcția compozabilă **Text**().

Schimbarea dimensiunii fontului

Pixeli scalabili

Pixeli scalabili (SP) este o unitate de măsură pentru dimensiunea fontului. Elemente UI din aplicațiile Android folosesc două unități diferite de măsură: **density-independent pixels (DP)**, pe care le veți folosi mai târziu pentru layout, și **scalable pixels (SP)**. Implicit, unitatea **SP** are aceeași dimensiune ca unitatea **DP**, dar se redimensionează în funcție de dimensiunea textului preferată de utilizator în setările telefonului.

14. În fișierul **MainActivity**.kt, derulați la compozabilul **Text**() din funcția **GreetingText**().
15. Transmiteți funcției **Text**() un argument numit **fontSize** ca al doilea argument și setați-l la o valoare de **100**.sp.

```
@Composable
fun GreetingText(message: String, modifier: Modifier = Modifier){
```

```
    Text(
        text = message,
        fontSize = 100.sp
    )
}
```

16. Nu uitați să importați **androidx.compose.ui.unit.sp** pentru a utiliza extensia proprietății **.sp**.
17. Actualizați compozabilul **Text** pentru a include **line height**.

```
@Composable
fun GreetingText(message: String, modifier: Modifier = Modifier){
    Text(
        text = message,
        fontSize = 100.sp,
        lineHeight = 116.sp,
    )
}
```

Adaugarea unui noi element text , semnaturarea felicitarii cu numele tau

18. În fișierul **MainActivity**.kt, derulați la funcția **GreetingText**().
19. Adăugați un parametru numit **from** de tip **String** pentru semnătura dumneavoastră.

```
fun GreetingText(message: String, from: String, modifier: Modifier =
Modifier)
```

20. După compozabilul Text pentru mesajul aniversar, adăugați un alt compozabil Text care acceptă un argument text setat la valoarea from.

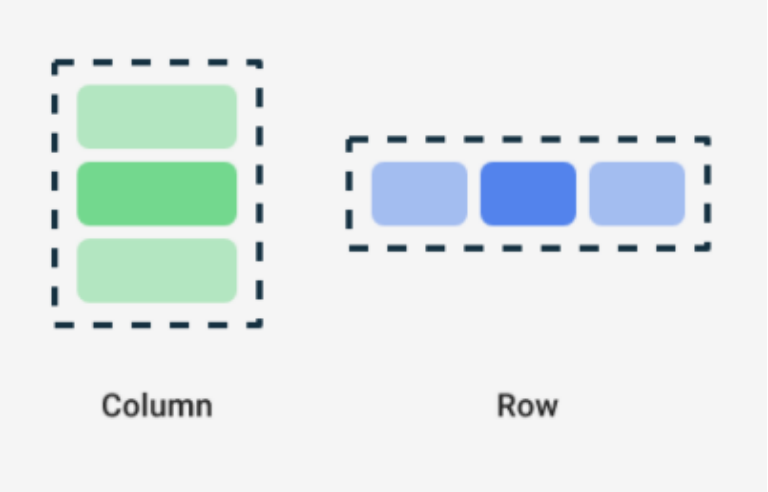
```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier =
Modifier){
    Text(
        text = message,
        fontSize = 100.sp,
        lineHeight = 116.sp,
    )
    Text(
        text = from
    )
}
```

21. Adăugați un argument numit **fontSize**, setat la valoarea **36.sp**.
22. În funcția **BirthdayCardPreview()**, adăugați un alt argument de tip **String** pentru a semna cardul, de exemplu, "From Ioana"

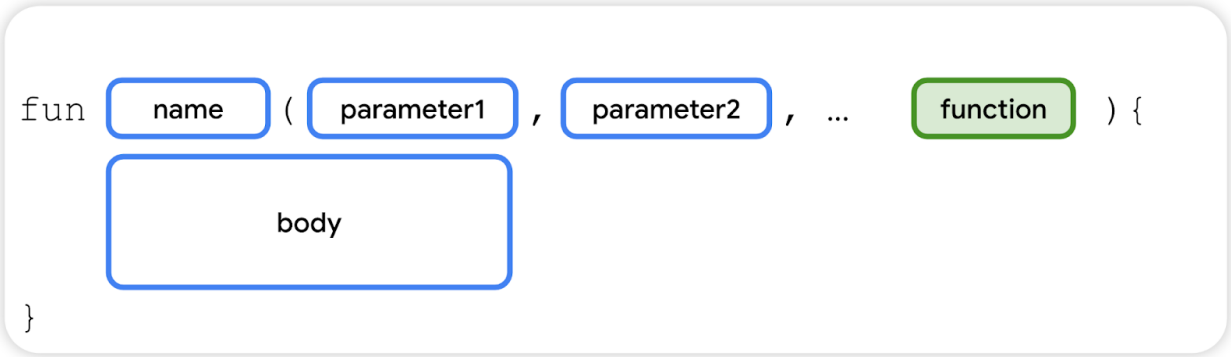
```
GreetingText(message = "Happy Birthday Sam!", from = "From Ioana")
```

Aranjarea elementelor text in randuri si coloane

Column, **Row** și **Box** sunt funcții compozabile care acceptă conținut compozabil ca argumente, astfel încât puteți plasa elemente în interiorul acestor elemente de layout. De exemplu, fiecare element copil dintr-un compozabil **Row** este plasat orizontal unul lângă altul într-un rând.

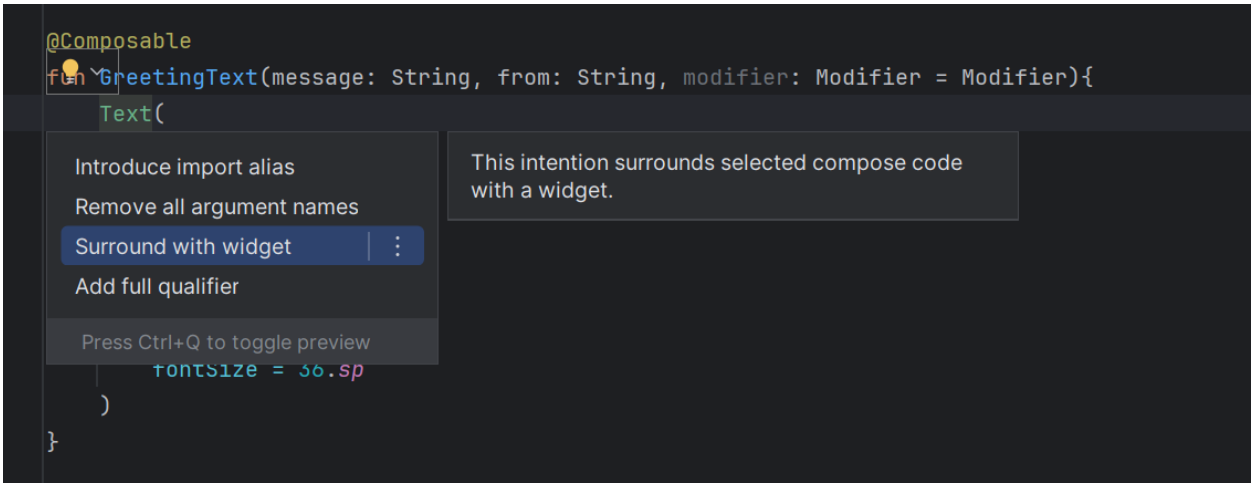


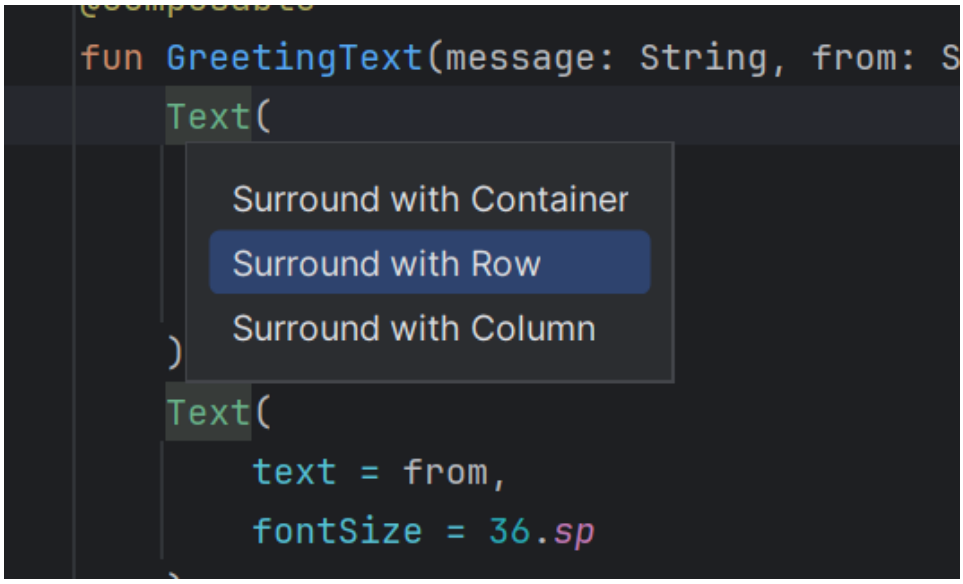
Kotlin oferă o sintaxă specială pentru transmiterea funcțiilor ca parametri ai altor funcții, atunci când ultimul parametru este o funcție.



Când transmiteți o funcție ca parametru, puteți utiliza **trailing lambda syntax**. În loc să puneți funcția între paranteze, o puteți plasa în afara parantezelor între acolade. Aceasta este o practică recomandată și comună în Compose.

23. În funcția **GreetingText()**, adăugați compozabilul **Row** în jurul elementelor text pentru a afișa un rând cu două elemente text. Selectați cele două compozabile **Text**, faceți clic pe becul de lângă ele și selectați **Surround with widget > Surround with Row**.

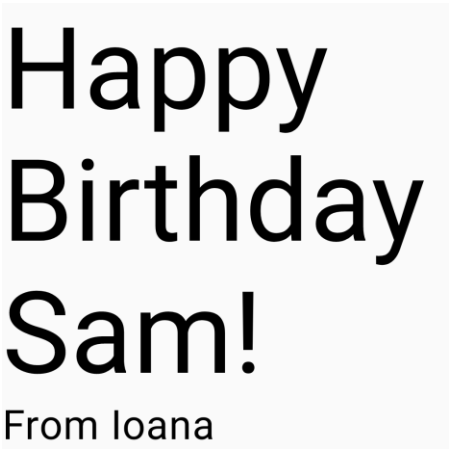





```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier =
Modifier){
    Row {
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp,
        )

        Text(
            text = from,
            fontSize = 36.sp
        )
    }
}
```

24. Poate fi schimbat in format Coloana



25. În funcția **onCreate()**, apelați funcția **GreetingText()** din blocul **Surface** și transmiteți funcției **GreetingText()** mesajul de aniversare și semnătura dumneavoastră.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    //enableEdgeToEdge()
    setContent {
        HappyBirthdayTheme {
            Surface (
                modifier = Modifier.fillMaxSize(),
                color = MaterialTheme.colorScheme.background
            ){
                GreetingText(message = "Happy Birthday Sam!", from = "From
Ioana")
            }
        }
    }
}
```

26. Construiți și rulați aplicația pe emulator.

Alinierea mesajului la centru

27. Pentru a alinia mesajul la centrul ecranului, adăugați un parametru numit **verticalArrangement** și setați-l la valoarea **Arrangement.Center**.

```
fun GreetingText(message: String, from: String, modifier: Modifier =
Modifier){
    Column (
        verticalArrangement = Arrangement.Center,
        modifier = modifier)
```

28. Adăugați o margine de 8.dp în jurul coloanei. Este o bună practică să utilizați valori pentru margini în multipli de 4.dp

```
modifier = modifier.padding(8.dp)
```

29. Pentru a înfrumuseța aplicația, aliniați textul mesajului la centru utilizând **textAlign**.

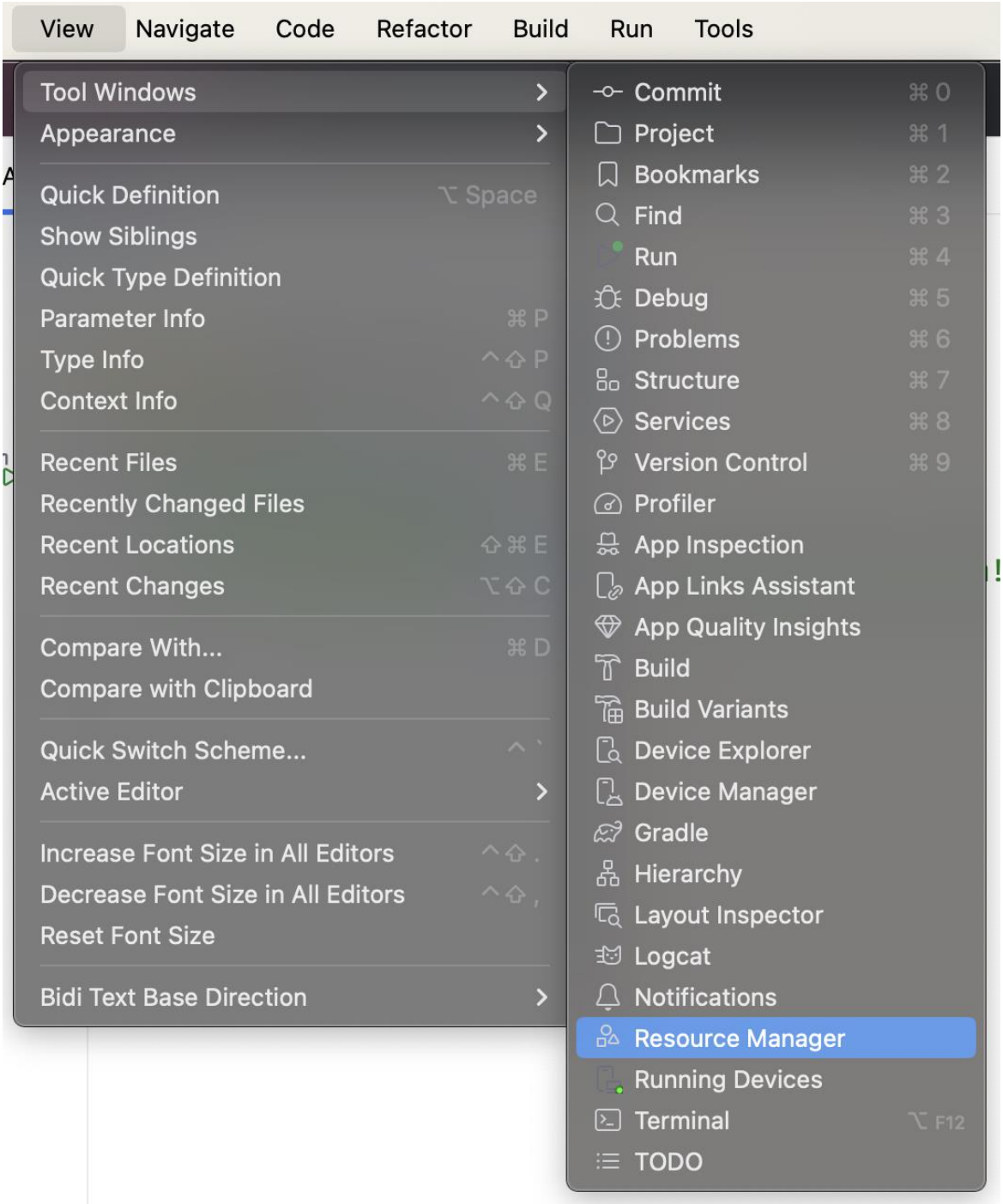
```
Text(
    text = message,
    fontSize = 100.sp,
    lineHeight = 116.sp,
    textAlign = TextAlign.Center
)
```

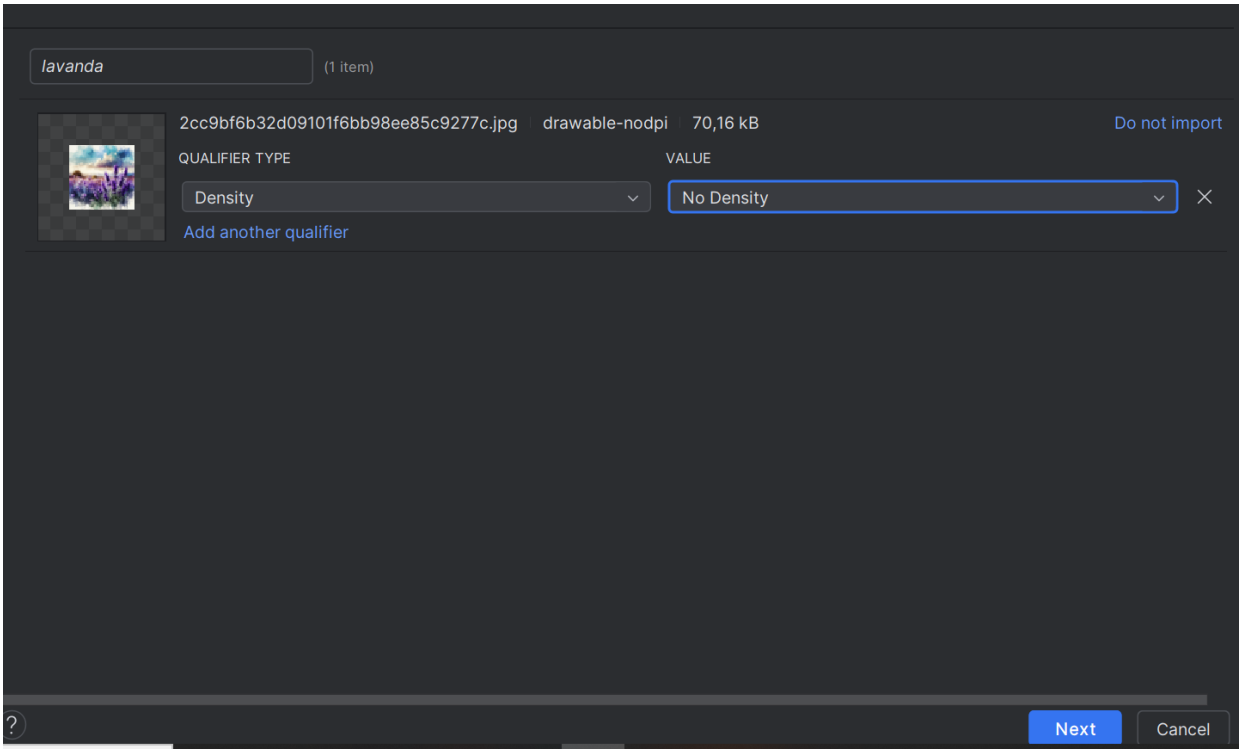
30. Adăugați o margine textului semnăturii și aliniați-l la dreapta.

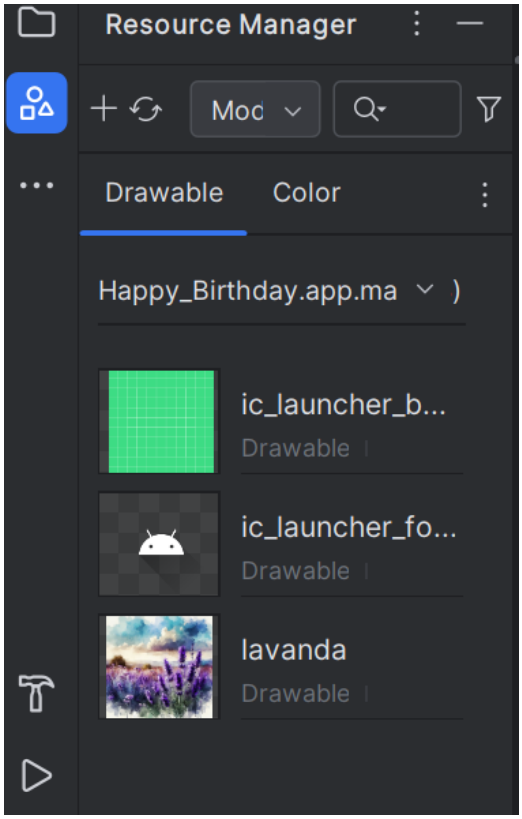
```
Text(
    text = from,
    fontSize = 36.sp,
    modifier = Modifier
        .padding(16.dp)
        .align(alignment = Alignment.End)
)
```

Vom adauga o imagine la felicitarea noastra

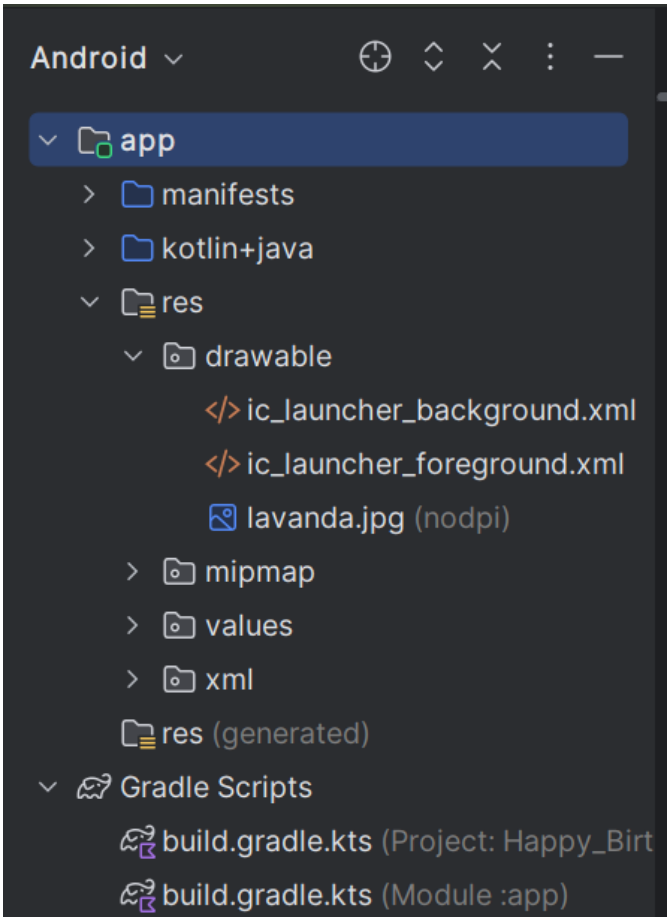
31. În Android Studio, faceți clic pe **View > Tool Windows > Resource Manager** sau pe fila **Resource Manager** de lângă fereastra **Project**.




lavanda
(1 item)
2cc9bf6b32d09101f6bb98ee85c9277c.jpg
drawable-nodpi
70,16 kB
Do not import
QUALIFIER TYPE
VALUE
Density
No Density
Add another qualifier
Next
Cancel


Resource Manager
Mod
Drawable
Color
Happy_Birthday.app.ma
ic_launcher_b...
Drawable
ic_launcher_fo...
Drawable
lavanda
Drawable

32. Faceți clic pe **app > res > drawable** pentru a confirma că imaginea se află în folderul drawable.



Adăugarea unei funcții compozabile pentru imagine

33. În fișierul **MainActivity.kt**, adăugați o funcție compozabilă numită **GreetingImage()** după funcția **GreetingText()**.
34. Transmiteți funcției **GreetingImage()** două parametri de tip **String**: unul numit **message** pentru mesajul de aniversare și altul numit **from** pentru semnătura dvs., împreună cu **Modifier**.

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier){

}
```

35. În funcția **GreetingImage()**, declarați o proprietate **val** și numiți-o **image**.
36. Apelați funcția **painterResource()**, transmițând resursa **lavanda**, și atribuiți valoarea returnată variabilei **image**.

O clasă **R** este o clasă generată automat de Android, care conține ID-urile tuturor resurselor din proiect. În majoritatea cazurilor, ID-ul resursei este același cu numele fișierului.

```
val image = painterResource(R.drawable.lavanda)
```

37. După apelul funcției **painterResource()**, adăugați un compozabil **Image** și transmiteți **image** ca argument pentru **painter**.

Android Studio afișează un nou avertisment care indică: „None of the following functions can be called with the arguments supplied.” Android Studio oferă sugestii și avertismente pentru a vă ajuta să faceți aplicația mai accesibilă. O descriere a conținutului (content description) definește scopul unui element UI, ceea ce face aplicația mai ușor de utilizat cu TalkBack.

Cu toate acestea, imaginea din această aplicație este inclusă doar în scop decorativ. Adăugarea unei descrieri pentru conținutul imaginii ar îngreuna utilizarea aplicației cu TalkBack în acest caz particular. În loc să setați o descriere a conținutului care este anunțată utilizatorului, puteți seta argumentul **contentDescription** al imaginii la null, astfel încât TalkBack să sară peste compozabilul Image.

În compozabilul **Image**, adăugați un alt argument numit contentDescription și setați valoarea acestuia la **null**.

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier =
Modifier){
    val image = painterResource(R.drawable.lavanda)
    Image(
        painter = image,
        contentDescription = null
    )
}
```

Adăugarea unui layout Box

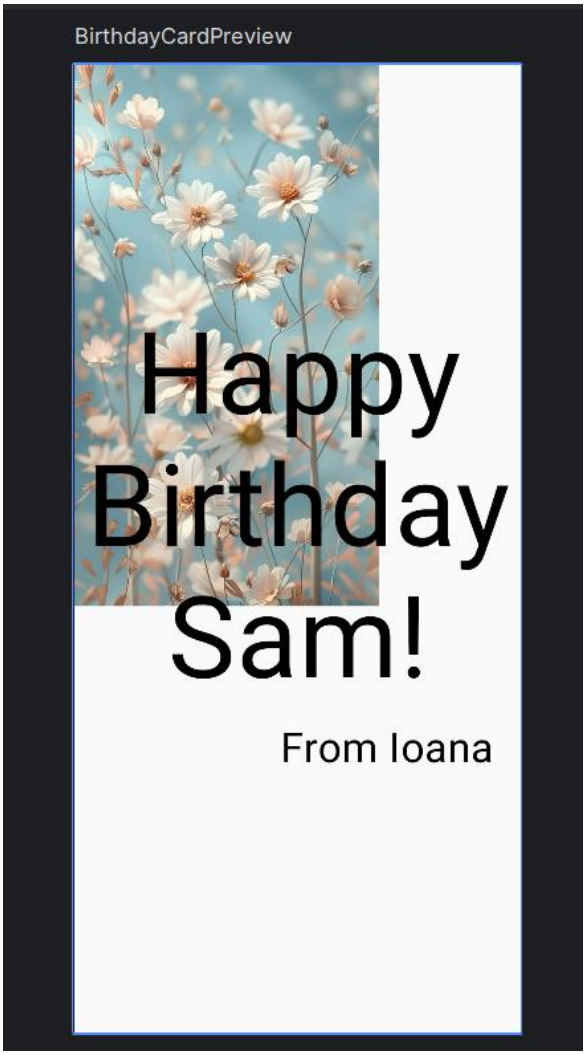
38. **Box** este un layout standard în Compose care permite suprapunerea elementelor. Utilizați **Box** pentru a suprapune elementele și a configura aliniamentul lor specific.
39. În funcția **GreetingImage()**, adăugați un compozabil **Box** în jurul compozabilului **Image**:

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier =
Modifier){
    val image = painterResource(R.drawable.lavanda)
    Box {
        Image(
            painter = image,
            contentDescription = null
        )
    }
}
```

40. Adăugați cod pentru a transmite parametrul **modifier** compozabilului **Box**.
41. La sfârșitul compozabilului **Box**, apelați funcția **GreetingText()** și transmiteți mesajul, semnătura și **modifier**

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier =
Modifier){
    val image = painterResource(R.drawable.lavanda)
    Box(modifier) {
        Image(
            painter = image,
            contentDescription = null
        )
```

```
        GreetingText(
            message = message,
            from = from,
            modifier = Modifier
                .fillMaxSize()
                .padding(8.dp)
        )
    }
}
```



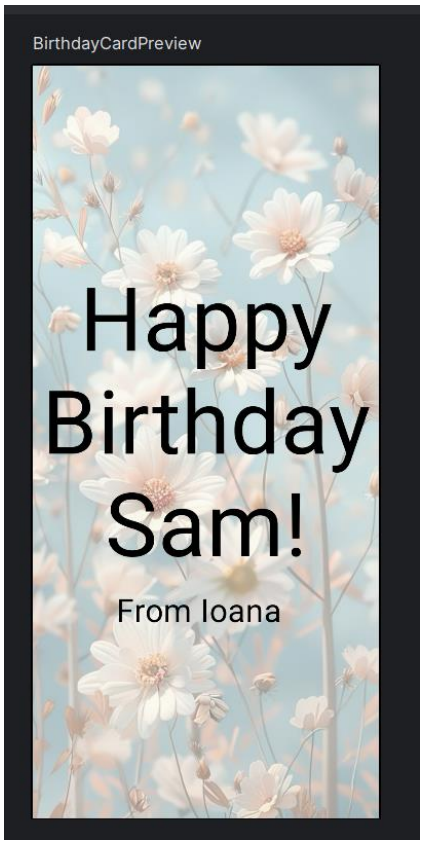
42. În funcția **onCreate()**, înlocuiți apelul funcției **GreetingText()** cu **GreetingImage()** pentru a reflecta modificările în emulator.

```
Surface (
    modifier = Modifier.fillMaxSize(),
    color = MaterialTheme.colorScheme.background
){
    GreetingImage(
        message = "Happy Birthday Sam!",
        from = "From Emma"
    )
}
```

Redimensionarea și scalarea conținutului

43. Adăugați un argument numit **ContentScale** compozabilului **Image**
44. Adăugați un parametru **alpha** la compozabilul **Image** și setați-l la **0.5F** pentru a modifica opacitatea.
45. De asemenea daca imaginea nu umple complet ecranul chiar daca a fost adaugat **ContentScale.Crop**, atunci adaugati si **modifier = Modifier.fillMaxSize**() in interiorul lui **Box**() si **Image**().

```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier =
Modifier){
    val image = painterResource(R.drawable.margarete)
    Box(modifier = Modifier.fillMaxSize()) {
        Image(
            painter = image,
            contentDescription = null,
            contentScale = ContentScale.Crop,
           modifier = Modifier.fillMaxSize(),
            alpha = 0.5F
        )
```



Modificatori pentru Layout

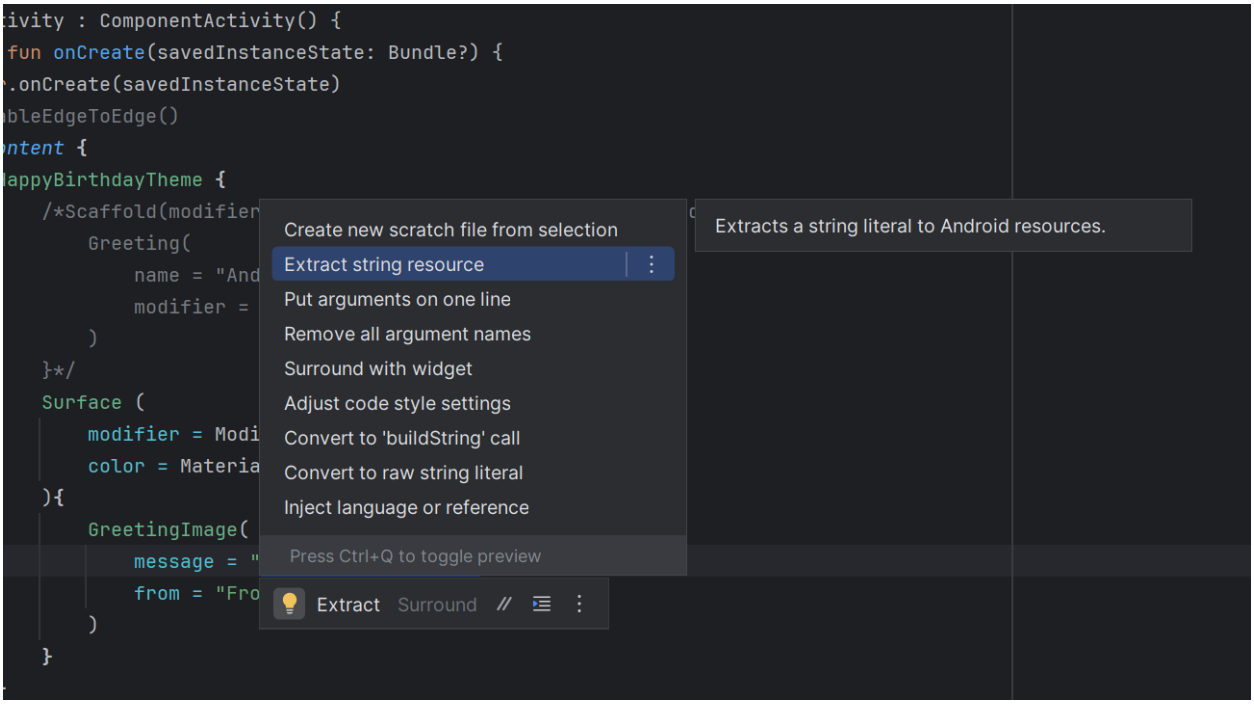
Modificatorii sunt utilizați pentru a decora sau a adăuga comportamente elementelor UI din Jetpack Compose. De exemplu, puteți adăuga fundaluri, spațiere (**padding**) sau comportamente pentru rânduri (**rows**), text sau butoane. Pentru a seta modificatori, un compozabil sau un layout trebuie să accepte un **modifier** ca parametru.

Proprietatea **arrangement** este utilizată pentru a aranja elementele copil atunci când dimensiunea layout-ului este mai mare decât suma dimensiunilor copiilor.

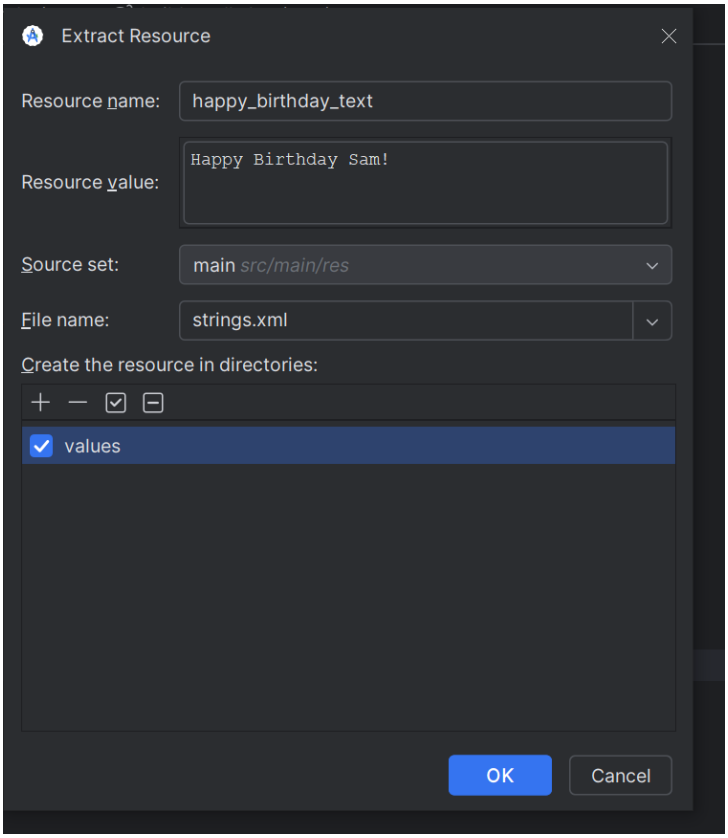
Traducerea

Șirurile hardcodate îngreunează traducerea aplicației în alte limbi și reutilizarea șirurilor în diferite părți ale aplicației. Puteți extrage șirurile într-un fișier de resurse pentru a rezolva aceste probleme. În loc să introduceți direct șiruri în cod, le puneți într-un fișier, le atribuiți un nume de resursă și utilizați numele oriunde aveți nevoie de șiruri. Numele rămâne același, chiar dacă modificați șirul sau îl traduceți într-o altă limbă.

46. În fișierul **MainActivity.kt**, derulați la funcția **onCreate()**. Selectați șirul de caractere **Happy Birthday Sam!**, fără ghilimele.
47. Faceți clic pe becul din stânga ecranului.
48. Selectați **Extract string resource**.



49. În dialogul **Extract Resource**, schimbați numele resursei în **happy_birthday_text**.



50. În panoul **Project**, deschideți fișierul **strings.xml** din calea **app > res > values > strings.xml** și observați că Android Studio a creat o resursă de tip șir numită **happy_birthday_text**.

51. Urmați aceiași pași pentru a extrage textul pentru compozabilul **Text** destinat semnăturii, dar de această dată introduceți **signature_text** în câmpul **Resource name**.

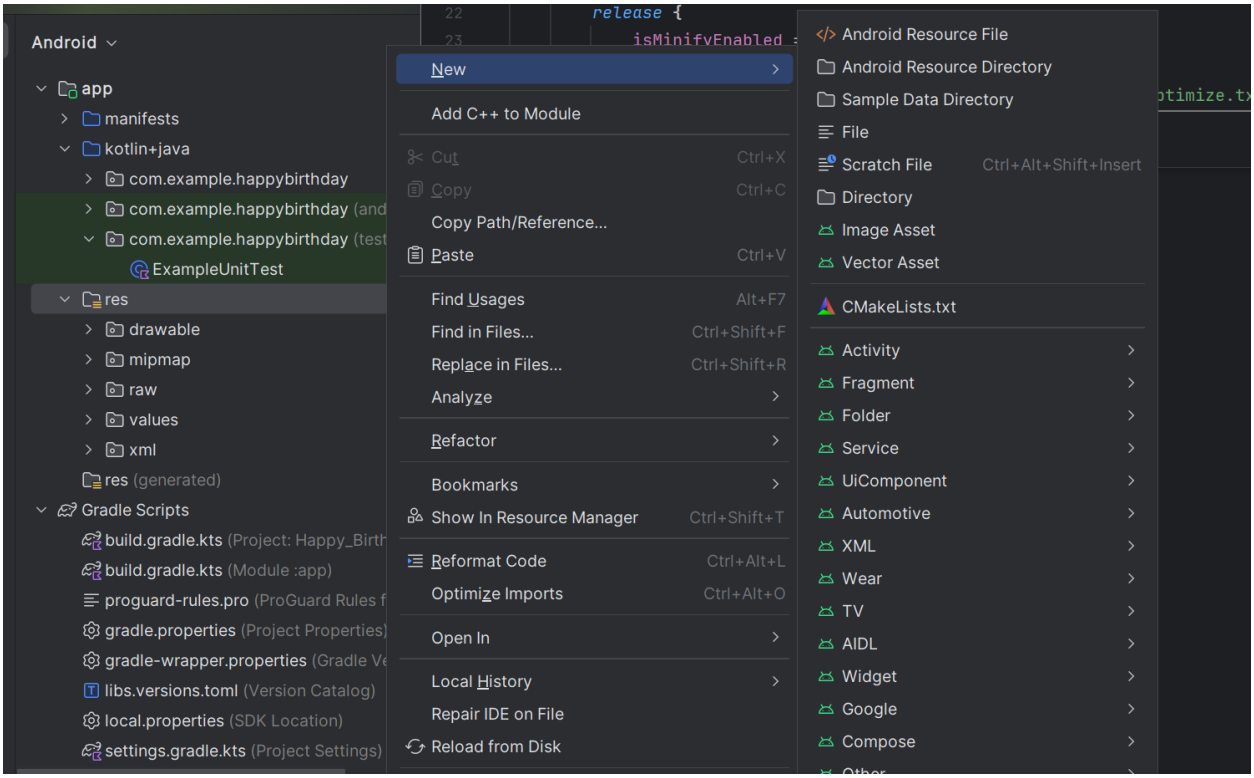
```
<resources>
    <string name="app_name">Happy Birthday</string>
    <string name="happy_birthday_text">Happy Birthday Sam!</string>
    <string name="signature_text">From Ioana</string>
</resources>
```

52. Actualizați funcția **BirthdayCardPreview()** pentru a utiliza **stringResource()** și șirurile extrase.

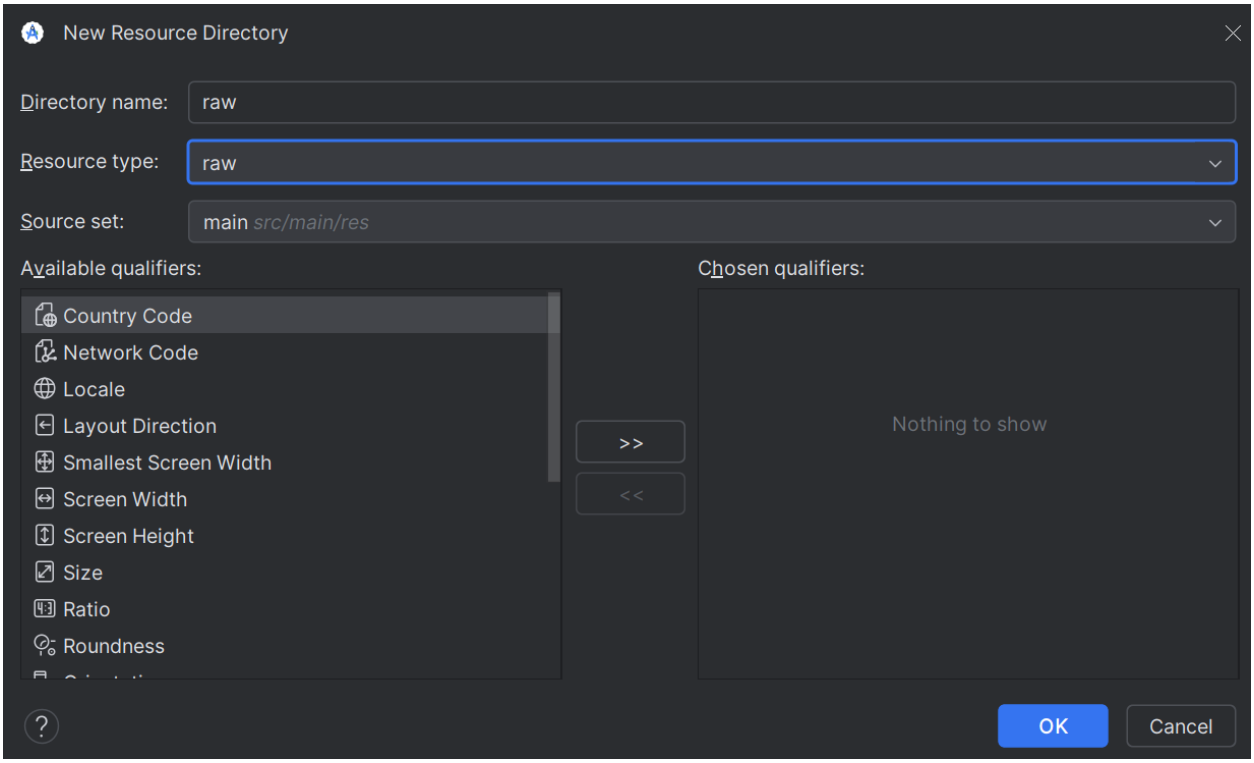
Adaugarea unui cantec de ‘La multi ani’ si 2 butoane pentru pornirea si oprirea muzicii

Prima data vom adauga melodia dorita la proiect. Android suporta urmatoarele formate audio: MP3, MIDI, Opus, Vorbis, PCM/WAVE, FLAC, AMR-WB, AMR-NB, etc.

53. Mergi la **app>res>right-click>New>Android Resource Directory**

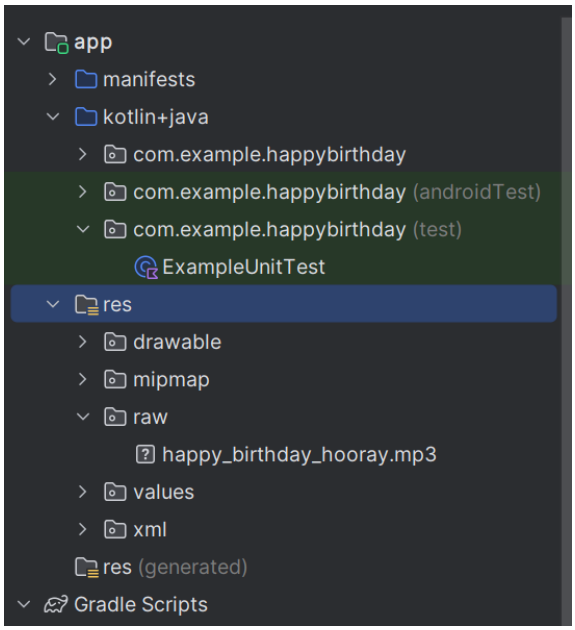


54. La **Resource type** alege **raw** si apasa pe butonul **OK**.



55. Acum poti sa vezi folderul **raw** ca a fost creat , il poti gasi in **app>res>raw.**
56. Cauta in calculatorul tau **fisierul audio** pe care doresti a il adaugi si **copiazal**. Apoi in aplicatie dai click dreapta pe folderul **raw** si dai click pe **Paste**.

57. Redenumesteti fisierul audio prin dand click pe **refactor** si **rename** . Ai grija la redumirea acestuia deoarece Android studio accepta doar litere mici , fara spatii , linii sau cifre , iar nerespectarea acestora iti va crea erori in cod.



Acum vom incepe scrierea codului :

58. In **MainActivity** va fi adaugat un camp music de tipul **MediaPlayer**
59. In functia **onCreate**() adauga in campul **music** melodia care vrei sa fie redata in aplicatie, extragand-o din fisierul **raw**
60. Adaugam si 3 functii private care se ocupa de logica de pornire a muzicii , de oprire a muzicii si de eliberare a resurselor cand este oprita aplicatia .
    Functia **playMusic**() porneste muzica doar daca aceasta nu este deja pornita
    Functia **stopMusic**() opreste muzica si o reseteaza de la inceput
    Functia **onDestroy**() elibereaza resursele cand activitatea este distrusa

```
class MainActivity : ComponentActivity() {
    private lateinit var music : MediaPlayer
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        music = MediaPlayer.create(this, R.raw.happy_birthday_hooray)

        //enableEdgeToEdge()
        setContent {

            HappyBirthdayTheme {
                /*Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding
->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
                    )
                }*/
                Surface (
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
```

```
                ){
                    GreetingImage(
                        message =
stringResource(R.string.happy_birthday_text),
                        from = stringResource(R.string.signature_text)
                    )
                }
            }
        }
    }
    private fun playMusic() {
        if (!music.isPlaying) {
            music.start()
        }
    }

    private fun stopMusic() {
        if (music.isPlaying) {
            music.pause()
            music.seekTo(0) // Reset the music to the beginning
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        music.release() // Release the MediaPlayer resources
    }
}
```

61. Vom crea un **Composable** care afiseaza imaginea de fundal , mesajul de ‘la multi ani’ si butoanele pentru pornirea si oprirea muzicii.

```
@Composable
fun MusicController(
    message: String,
    from: String,
    onPlay: () -> Unit,
    onStop: () -> Unit,
    modifier: Modifier = Modifier
) {
    Box(modifier = Modifier.fillMaxSize()) {
        Image(
            painter = painterResource(id = R.drawable.margarete),
            contentDescription = null,
            contentScale = ContentScale.Crop,
            modifier = Modifier.fillMaxSize(),
            alpha = 0.5F
        )
        GreetingText(
            message = message,
            from = from,
            modifier = Modifier
                .fillMaxSize()
                .padding(8.dp)
        )
        // Buttons for Play and Stop
        Row(
            modifier = Modifier
```

```
                .align(Alignment.BottomCenter)
                .padding(16.dp),
            horizontalArrangement = Arrangement.spacedBy(16.dp)
        ) {
            Button(onClick = onPlay) {
                Text(text = "Play")
            }
            Button(onClick = onStop) {
                Text(text = "Stop")
            }
        }
    }
}
```

62. Modificam referintele la **GreetingImage** la **MusicController** in **MainActivity (in setContent)** si in **BirthdayCardPreview**

```
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        MusicController(
            message = stringResource(R.string.happy_birthday_text),
            from = stringResource(R.string.signature_text),
            onPlay = { /* No-op for preview */ },
            onStop = { /* No-op for preview */ }
        )
    }
}
```

Actiunile butoanelor sunt lăsate goale (No-op) deoarece nu putem reda muzică în modul preview.

```
setContent {

    HappyBirthdayTheme {
        Surface (
            modifier = Modifier.fillMaxSize(),
            color = MaterialTheme.colorScheme.background

        ){
            MusicController(
                message = stringResource(R.string.happy_birthday_text),
                from = stringResource(R.string.signature_text),
                onPlay = { playMusic() },
                onStop = { stopMusic() }
            )
        }
    }
}
```

V. Concluzii

Dezvoltarea aplicatiilor Androi este o aventura interesanta. Ajuta foarte mult sa cunosti limbajul de programare Kotlin si Java. Kotlin este foarte usor de invatat , iar cunostintele de Java au un rol important in intelegerea limbajului. Cei de la Android pun la dispozitie o multime de resurse cum ar fi tutorile si laboratoare detaliate pentru cei care doresc sa dezvolte aplicatii Android. Partea cea mai dificila ar putea fi gasirea unei idei pentru aplicatia ta si modelarea acesteia intr-o arhitectura usor de inteles si care respecta principiile SOLID. Cum lumea se bazeaza din ce in ce mai mult pe tehnologie si digitalizare, acumularea cunostintelor si a experientei in crearea de aplicatii Android poate fi un mare avantaj in viitor.

VI. Resurse

- https://pontifex.ro/aplicatiile-mobile-tipuri-si-dezvoltare/ https://www.roweb.ro/ro/blog/dezvoltare-de-aplicatii-mobile-tendinte-si-statistici/ (cam inutil, nu am extras ceva , sunt statistici si tendinte la aplicatii in ziua de azi)
- https://developer.android.com/get-started/overview (te invata cum sa folosesti android studio)
- https://www.geeksforgeeks.org/how-to-add-audio-files-to-android-app-in-android-studio/
- https://developer.android.com/topic/architecture/ui-layer
- https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-2#codelab-https://developer.android.com/codelabs/basic-android-kotlin-compose-training-add-scrollable-list
- https://developer.android.com/courses/android-basics-compose/course