aplikacje mobilne
Android Studio Jetpack Compose

tutorial version 2.10

Telefonia komórkowa [edytuj]

48 języków

Artykuł Dyskusja | Czytaj Edytuj Edytuj kod źródłowy Wyświetl historię Narzędzia

Ten artykuł należy dopracować:

od 2013-08 → usunąć zwodnicze wyrażenia,

od 2013-01 → zaktualizować na podstawie najświeższych informacji,

od 2013-08 → dodać przypisy do treści niemających odnośników do źródeł.

Dokładniejsze informacje o tym, co należy poprawić, być może znajdują się w dyskusji tego artykułu.

Po wyeliminowaniu niedoskonałości należy usunąć szablon {{Dopracować}} z tego artykułu.

Telefonia komórkowa – infrastruktura telekomunikacyjna (oraz procesy związane z jej budową i eksploatacją), umożliwiająca abonentom bezprzewodowe połączenia na obszarze złożonym z tzw. komórek (ang. *cells*), obszarów kontrolowanych przez poszczególne anteny stacji bazowych. Charakterystyczną cechą tego typu telefonii jest zapewnienie użytkownikowi telefonu komórkowego mobilności, może on zestawiać połączenia (oraz połączenia mogą być zestawione do niego) na terenie pokrytym zasięgiem radiowym związanym ze wszystkimi stacjami bazowymi w danej sieci.

Wieża przekaźnikowa w Zieleńcu

Systemem telefonii komórkowej o największym zasięgu na świecie jest GSM[1] (około 80% rynku telefonii mobilnej[2]). Należy on do tzw. telefonii komórkowej drugiej generacji. W 2001 roku uruchomiono pierwszą komercyjną sieć telefonii trzeciej generacji[3]. Wśród wdrażanych na świecie systemów 3G najwięcej sieci (73%) zbudowanych jest na bazie standardu UMTS (ang. *Universal Mobile Telecommunications System*)[4].

Standardy telefonii komórkowej [edytuj | edytuj kod]

https://pl.wikipedia.org/wiki/Telefonia_kom%C3%B3rkowa#Telefonia_kom%C3%B3rkowa_pierwszej_generacji

3GPP

The **3rd Generation Partnership Project** (**3GPP**) is an umbrella term for a number of standards organizations which develop protocols for mobile telecommunications. Its best known work is the development and maintenance of:[1]

- GSM and related 2G and 2.5G standards, including GPRS and EDGE
- UMTS and related 3G standards, including HSPA and HSPA+
- LTE and related 4G standards, including LTE Advanced and LTE Advanced Pro
- 5G NR and related 5G standards, including 5G-Advanced
- An evolved IP Multimedia Subsystem (IMS) developed in an access independent manner

3rd Generation Partnership Project

Abbreviation	3GPP
Formation	1998; 26 years ago
Type	Standards organization
Region served	Worldwide
Website	www.3gpp.org

https://en.wikipedia.org/wiki/3GPP

3GPP

https://vimeo.com/790081425

3GPP

About 3GPP

3GPP specifications cover cellular telecommunications technologies, including radio access, core network and service capabilities, which provide a complete system description for mobile telecommunications.

Our mission continues to be the creation of the Mobile Broadband Standard, with an increasing emphasis towards connecting the internet of things – whether the need is for ultra-reliable low latency communications at one end of the scale or for energy efficient low-cost, low-power sensors and devices at the other.

https://www.3gpp.org/

3GPP - LTE

3GPP Specification series

Go to spec numbering scheme page

Click on spec number for details

spec number	title	notes
TS 36.101	Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception	
TS 36.102	Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception for satellite access	
TS 36.104	Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception	
TS 36.106	Evolved Universal Terrestrial Radio Access (E-UTRA); FDD repeater radio transmission and reception	
TS 36.108	Evolved Universal Terrestrial Radio Access (E-UTRA); Satellite Access Node radio transmission and reception	
TS 36.111	Location Measurement Unit (LMU) performance specification; Network based positioning systems in Evolved Universal Terrestrial Radio Access Network (E-UTRAN)	

https://www.3gpp.org/dynareport?code=36-series.htm

3GPP – LTE Architecture description

3GPP Portal

General | Versions | Responsibility | Related

Specification #: 36.401

Release 18(Spec is UCC for this Release) **Latest Remark:**

Meetings	Version	Upload date	Comment
RAN#104	18.1.0	2024-07-03	
SA#103	18.0.0	2024-04-03	Automatic upgrade from previo...

Release 17(Spec is UCC for this Release) **Latest Remark:**

Meetings	Version	Upload date	Comment
RAN#96	17.1.0	2022-06-23	
RAN#95-e	17.0.0	2022-04-06	

Release 16(Spec is UCC for this Release) **Latest Remark:**

https://www.3gpp.org/ftp/Specs/archive/36_series/36.401/36401-i10.zip

Android

Android – to system operacyjny z jądrem bazującym na Linuksie dla urządzeń mobilnych takich jak telefony komórkowe, smartfony, tablety (tablety PC) i netbooki. W 2013 roku był najpopularniejszym systemem mobilnym na świecie.

https://pl.wikipedia.org/wiki/Android_(system_operacyjny)

https://www.android.com/intl/pl_pl/why-android/

Android - wersje

Od kwietnia 2009 każda wersja Androida zostaje opracowana alfabetycznie pod nazwą nawiązującą do jakiegoś deseru bądź innego słodkiego produktu, każde kolejne wydanie otrzymuje kolejną literę alfabetu:

Nazwa kodowa	Numery wersji	Data pierwszego wydania	Poziomy API
Apple Pie	1.0	23 września 2008	1
Banana Bread[7]	1.1	9 lutego 2009	2
Cupcake	1.5	27 kwietnia 2009	3
Donut	1.6	15 września 2009	4
Eclair	2.0-2.1	26 października 2009	5-7
Froyo[8]	2.2-2.2.3	20 maja 2010	8
Gingerbread[8]	2.3-2.3.7	6 grudnia 2010	9-10
Honeycomb[8]	3.0-3.2.6	22 lutego 2011	11-13
Ice Cream Sandwich[8]	4.0-4.0.4	18 października 2011	14-15
Jelly Bean	4.1-4.3.1	9 lipca 2012	16-18
KitKat	4.4-4.4.4	31 października 2013	19-20
Lollipop	5.0-5.1.1	12 listopada 2014	21-22
Marshmallow	6.0-6.0.1	5 października 2015	23
Nougat	7.0-7.1.2	22 sierpnia 2016	24-25
Oreo	8.0-8.1	21 sierpnia 2017	26-27
Pie[9]	9	6 sierpnia 2018	28
Android 10 (wewnętrznie: *Quince Tart*[10])	10	3 września 2019	29
Android 11 (wewnętrznie: *Red Velvet Cake*[10])	11	8 września 2020	30
Android 12 (wewnętrznie: *Snow Cone*[11])	12-12L	4 października 2021	31-32
Android 13 (wewnętrznie: *Tiramisu*[12])	13	15 sierpnia 2022[1][2]	33
Android 14 (wewnętrznie: *Upside Down Cake*[13])	14	4 października 2023	34

22 sierpnia 2019 roku ogłoszono, że każda kolejna wersja systemu będzie nazywana po numerze wersji systemu[14]. Pomimo tego nazwy słodyczy są nadal wykorzystywane wewnętrznie przez Google[10].

https://pl.wikipedia.org/wiki/Wersje_systemu_Android

Android

Android 15 is released to AOSP

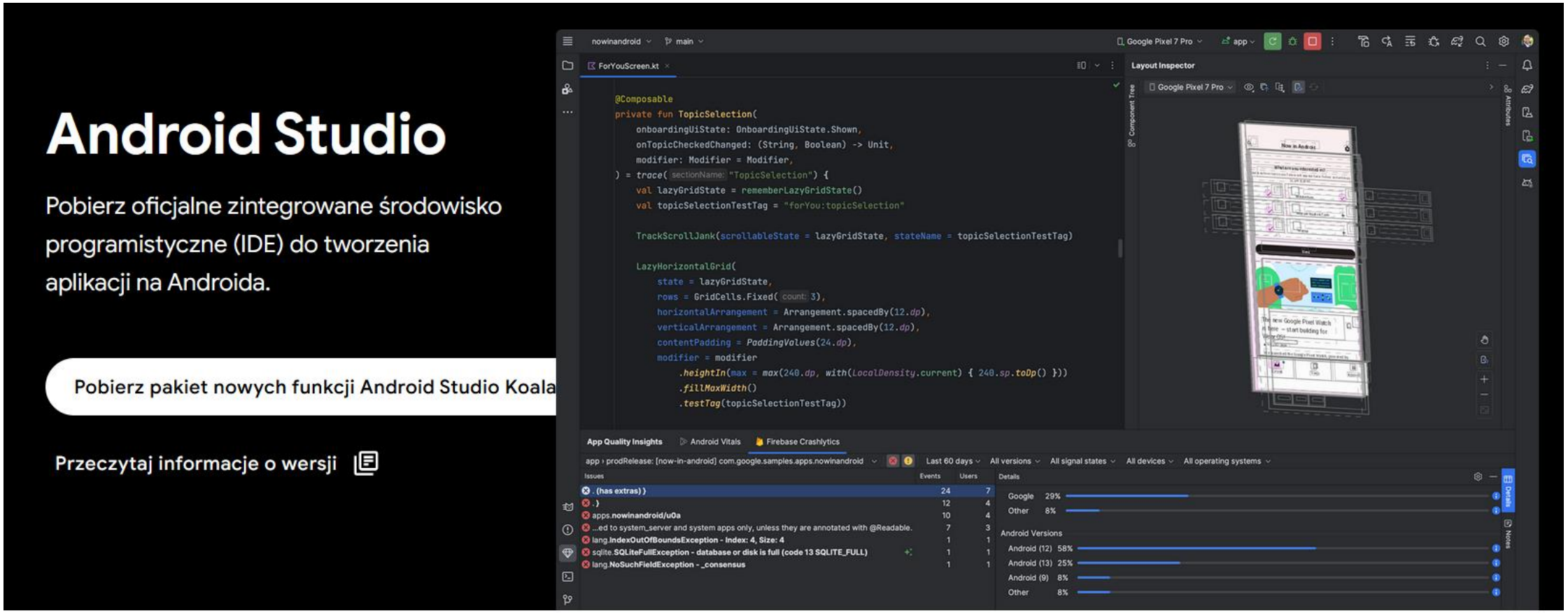
Today we're releasing Android 15 and making the source code available at the Android Open Source Project (AOSP). Android 15 will be available on supported Pixel devices in the coming weeks, as well as on select devices from Samsung, Honor, iQOO, Lenovo, Motorola, Nothing, OnePlus, Oppo, realme, Sharp, Sony, Tecno, vivo, and Xiaomi in the coming months.

We're proud to continue our work in open source through the AOSP. Open source allows anyone to build upon and contribute to Android, resulting in devices that are more diverse and innovative. You can leverage your app development skills in Android Studio with Jetpack Compose to create applications that thrive across the entire ecosystem. You can even examine the source code for a deeper understanding of how Android works.

Android 15 continues our mission of building a private and secure platform that helps improve your productivity while giving you new capabilities to produce beautiful apps, superior media and camera experiences, and an intuitive user experience, particularly on tablets and foldables.

https://android-developers.googleblog.com/2024/09/android-15-is-released-to-aosp.html

Android Studio



https://developer.android.com/studio?hl=pl

Android Studio - Kotlin

Hello world

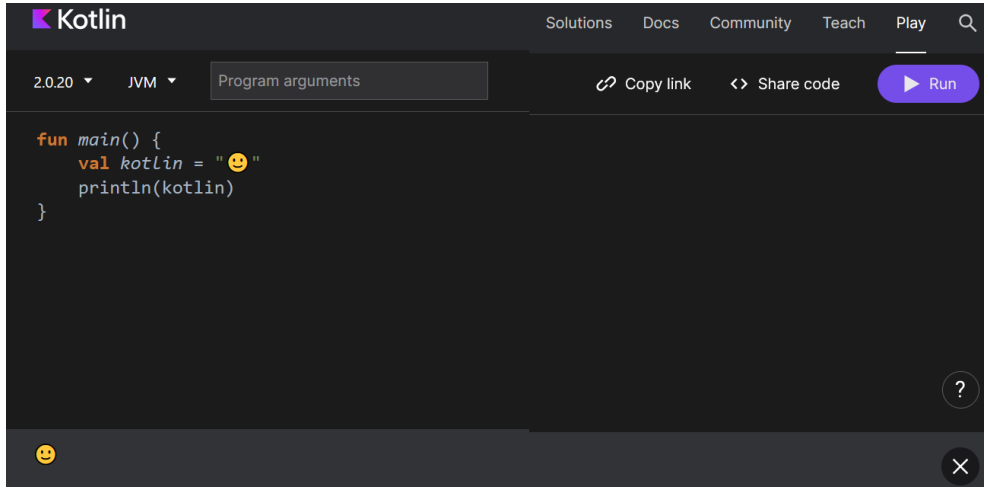
Edit page Last modified: 01 August 2024

1. **Hello world**
2. Basic types
3. Collections
4. Control flow
5. Functions
6. Classes
7. Null safety

Here is a simple program that prints "Hello, world!":

```
fun main() {
    println("Hello, world!")
    // Hello, world!
}
```

https://kotlinlang.org/docs/kotlin-tour-hello-world.html



https://play.kotlinlang.org/

Podstawy aplikacji

- Aplikacje na Androida można pisać w językach Kotlin, Java i C++.
- Android SDK kompiluje kod aplikacji wraz z wszelkimi plikami danych i zasobów do pliku APK lub pakietu Android App Bundle.
- plik APK (Android Package Kit, archiwum z sufiksem .apk), jest używany do dystrybucji oraz instalacji przez urządzenia z systemem Android.
- Android App Bundle (archiwum z sufiksem .aab) jest formatem programów wymaganym od programistów przez sklep Google Play. Na jego podstawie serwery Google generują finalny APK oraz podpisują go kluczem Google lub kluczem używanym w imieniu wydawcy.
- Każda aplikacja na Androida jest uruchamiana we własnej bezpiecznej piaskownicy (system przypisuje każdej aplikacji unikalny identyfikator Linux user ID, domyślnie każda aplikacja działa w oddzielnym procesie, każdy proces ma własną maszynę wirtualną (VM))
- System Android realizuje zasadę najmniejszych uprawnień. Oznacza to, że domyślnie każda aplikacja ma dostęp tylko do tych komponentów, których potrzebuje do swojej pracy i żadnych innych.

https://developer.android.com/guide/components/fundamentals

https://pl.wikipedia.org/wiki/APK

Składniki aplikacji

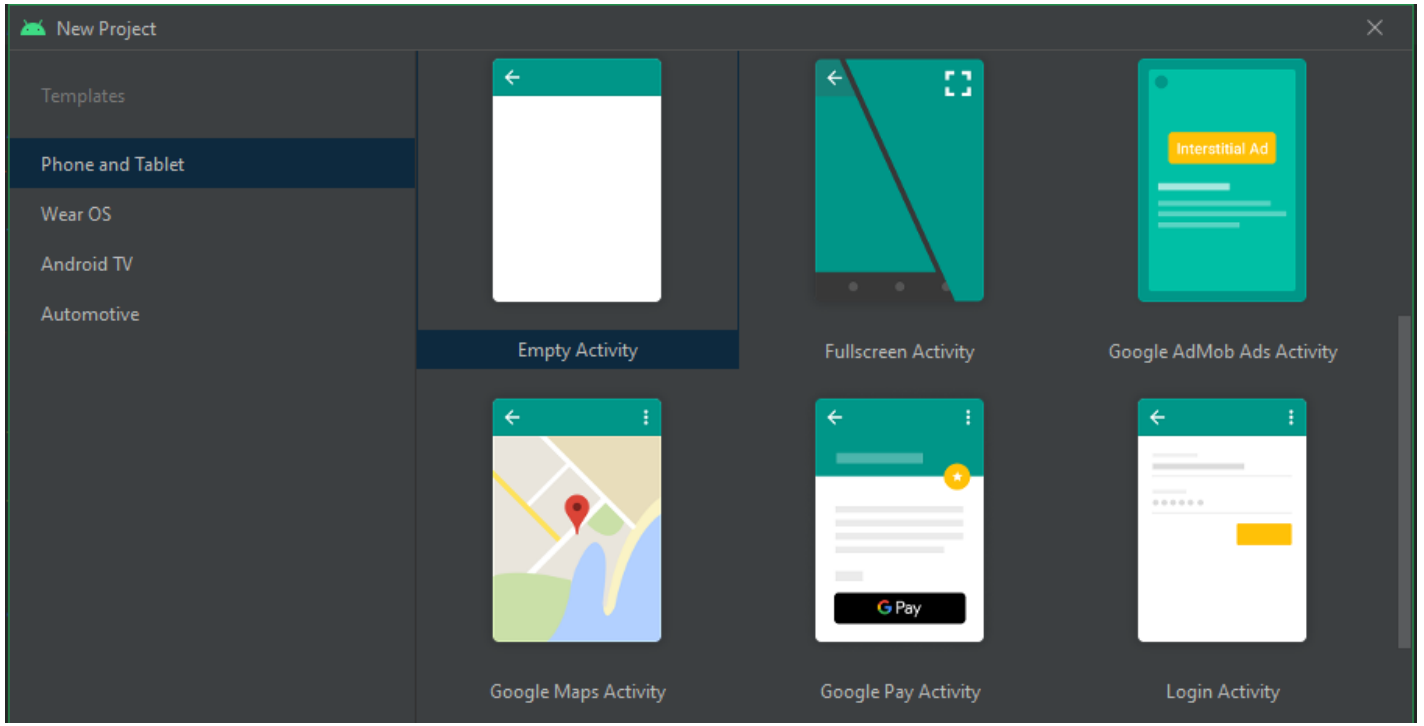
Składniki aplikacji to podstawowe elementy składowe aplikacji.

Istnieją cztery różne typy składników aplikacji:

- Activities (Aktywności)
- Services (Usługi)
- Broadcast receivers (Odbiorniki transmisji)
- Content providers (Dostawcy treści)

Activities (widoki, aktywności)

- Reprezentuje pojedynczy ekran z interfejsem użytkownika. Odpowiada za interakcję z użytkownikiem.
- Na przykład aplikacja poczty e-mail może mieć Activity, które pokazuje listę nowych wiadomości e-mail, inne Activity związane z tworzeniem wiadomości e-mail oraz inne Activity związane z czytaniem wiadomości e-mail.



https://developer.android.com/guide/components/activities/intro-activities

Services (usługi)

- Jest to komponent, który działa w tle w celu wykonywania długotrwałych operacji lub wykonywania pracy dla zdalnych procesów.
- Usługa nie zapewnia interfejsu użytkownika. Na przykład może odtwarzać muzykę w tle, gdy użytkownik używa innej aplikacji, lub może pobierać dane przez sieć bez blokowania interakcji użytkownika z aplikacją.
- Istnieją dwa rodzaje usług, które informują system, jak zarządzać aplikacją:
  - ✓ started services (usługi uruchomione)
  - ✓ bound services (usługi powiązane)
- started services informują system, aby działały, dopóki ich praca nie zostanie zakończona. Może to być synchronizacja niektórych danych w tle lub odtwarzanie muzyki nawet po opuszczeniu aplikacji przez użytkownika
- bound services działają, ponieważ jakaś inna aplikacja (lub system) powiedziała, że chce z niej skorzystać. Jest to w zasadzie usługa udostępniająca interfejs API dla innego procesu. W ten sposób system wie, że istnieje zależność między tymi procesami, więc jeśli proces A jest powiązany z usługą w procesie B, wie, że musi utrzymać proces B (i jego usługę) w działaniu dla A.

https://developer.android.com/guide/components/services

Broadcast receivers

broadcast receiver (odbiornik powiadomień) to składnik, który umożliwia systemowi dostarczanie powiadomień do aplikacji, dzięki czemu aplikacja może odpowiadać na ogólnosystemowe zdarzenia. System może dostarczać powiadomienia nawet do aplikacji, które nie są uruchomione. Wiele powiadomień pochodzi z systemu — na przykład powiadomienie informujące o wyłączeniu ekranu, niskim poziomie naładowania baterii lub zrobieniu zdjęcia. Chociaż *broadcast receivers* nie wyświetlają interfejsu użytkownika, mogą utworzyć powiadomienie na pasku stanu, aby informować użytkownika o wystąpieniu zdarzenia.

https://developer.android.com/guide/components/fundamentals

Content providers

Content providers (dostawca zawartości) zarządza udostępnionym zestawem danych aplikacji, które można przechowywać w systemie plików, w bazie danych SQLite, w sieci Web lub w dowolnej innej trwałej lokalizacji, do której aplikacja ma dostęp. Za pośrednictwem dostawcy treści inne aplikacje mogą wysyłać zapytania do danych lub modyfikować je, jeśli pozwala na to dostawca treści. Na przykład system Android zapewnia dostawcę treści, który zarządza informacjami kontaktowymi użytkownika.

https://developer.android.com/guide/topics/providers/content-providers

Intent

Trzy z czterech komponentów — activities, services, and broadcast receivers— są aktywowane przez asynchroniczną wiadomość zwaną *intent*. *Intents* wiążą ze sobą poszczególne składniki w czasie działania aplikacji. Można myśleć o nich jako o komunikatorach, które żądają akcji od innych składników, niezależnie od tego, czy składnik należy do danej aplikacji, czy do innej.

Intent jest tworzony za pomocą obiektu Intent, który definiuje komunikat, w celu aktywowania określonego komponentu lub określonego typu komponentu.

Intent może na przykład zawierać prośbę o pokazanie obrazu lub otwarcie strony internetowej.

Inny *intent* może umożliwić użytkownikowi wybranie kontaktu osobistego i zwrócenie go do naszej aplikacji. *Intent* zwrotu zawiera identyfikator URI wskazujący na wybrany kontakt.

https://developer.android.com/guide/components/intents-filters

plik AndroidManifest.xml

Zanim system Android będzie mógł uruchomić komponent aplikacji, system musi wiedzieć, że komponent istnieje, odczytując plik manifestu aplikacji, AndroidManifest.xml. Aplikacja musi zadeklarować w nim wszystkie używane przez nią komponenty. Plik manifestu musi znajdować się w katalogu głównym projektu aplikacji.
Manifest oprócz deklarowania komponentów aplikacji wykonuje szereg innych czynności, takich jak:

- Identyfikuje wszelkie uprawnienia użytkownika wymagane przez aplikację, takie jak dostęp do Internetu lub dostęp do odczytu kontaktów użytkownika.
- Deklaruje minimalny poziom interfejsu API wymagany przez aplikację, na podstawie używanych przez aplikację interfejsów API.
- Deklaruje funkcje sprzętu i oprogramowania używane lub wymagane przez aplikację, takie jak aparat, usługi Bluetooth lub ekran wielodotykowy.
- Deklaruje biblioteki API, z którymi aplikacja musi być połączona (inne niż interfejsy API platformy Android), takie jak biblioteka Google Maps.

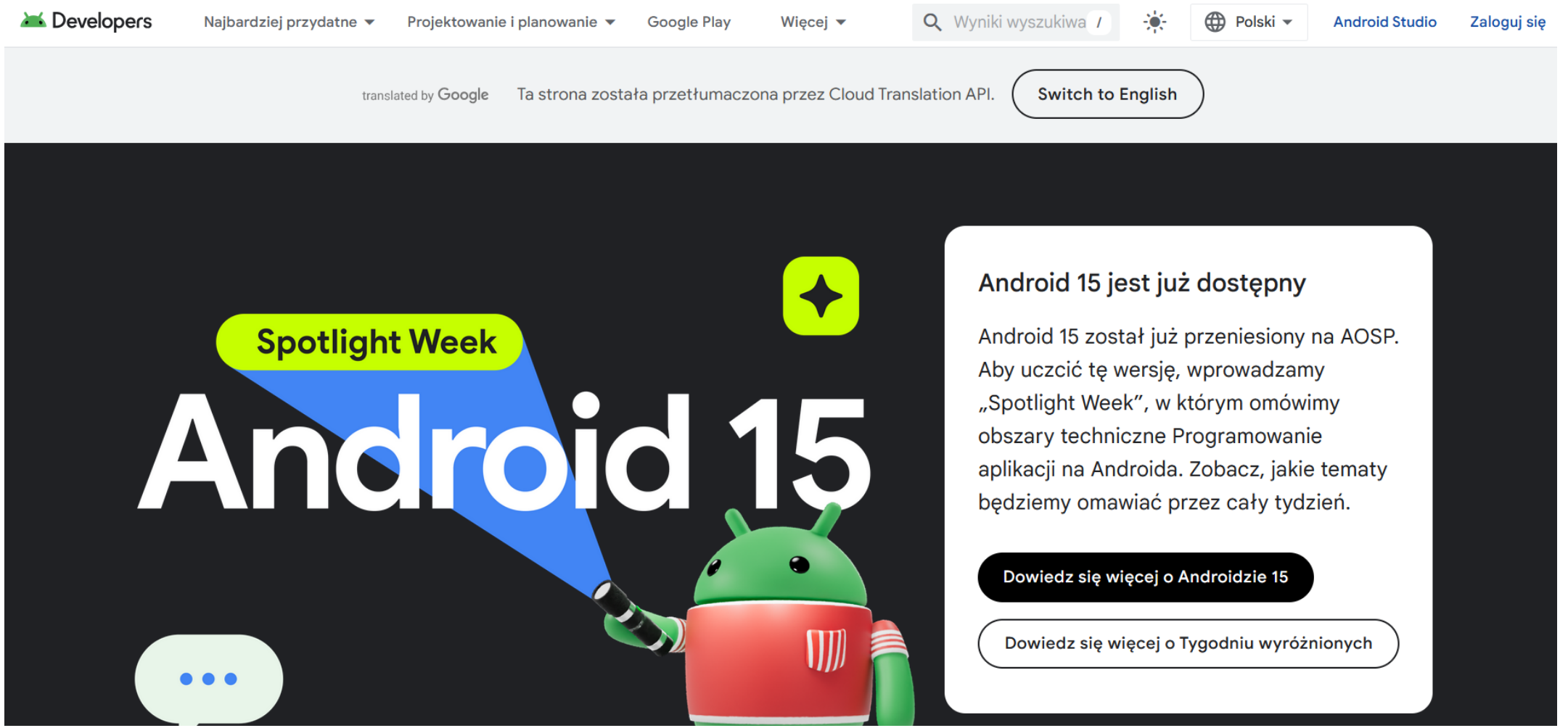
https://developer.android.com/guide/topics/manifest/manifest-intro

App resources
(zasoby aplikacji)

Aplikacja na Androida składa się nie tylko z kodu — wymaga zasobów odrębnych od kodu źródłowego, takich jak obrazy, pliki audio i wszystko, co dotyczy wizualnej prezentacji aplikacji. Za pomocą plików XML można zdefiniować animacje, menu, style, kolory i układ interfejsów użytkownika.
Korzystanie z zasobów aplikacji ułatwia aktualizowanie różnych cech aplikacji bez modyfikowania kodu. Udostępnianie zestawów alternatywnych zasobów umożliwia optymalizację aplikacji pod kątem różnych konfiguracji urządzeń, takich jak różne języki i rozmiary ekranu.
Dla każdego zasobu uwzględnionego w projekcie SDK definiuje unikalny identyfikator, którego można użyć do odwoływania się do zasobu z kodu aplikacji lub z innych zasobów zdefiniowanych w języku XML.
Na przykład, jeśli aplikacja zawiera plik obrazu o nazwie obrazek.png (zapisany w katalogu res/drawable/), SDK generuje identyfikator zasobu o nazwie R.drawable.obrazek. Ten identyfikator jest mapowany na liczbę całkowitą specyficzną dla aplikacji, której można użyć do odwoływania się do obrazu i wstawienia go do interfejsu użytkownika.

https://developer.android.com/guide/topics/resources/providing-resources

developer.android.com



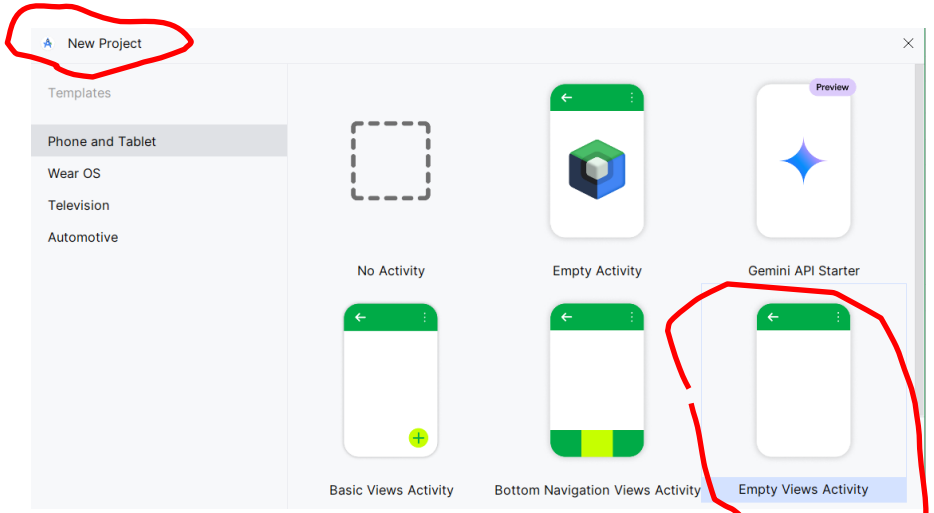
https://developer.android.com/?hl=pl

różne technologie tworzenia aplikacji w Android Studio

layout w plikach XML

Jetpack Compose

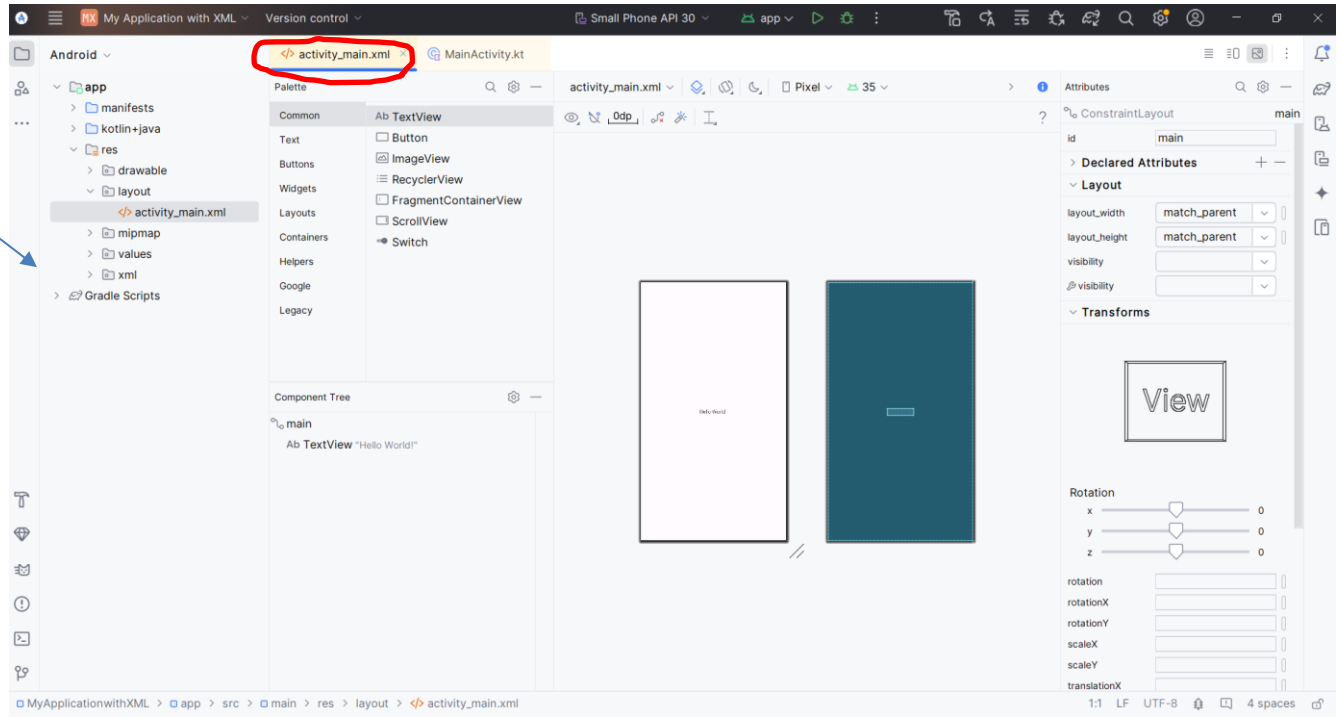
tworzenie aplikacji w Android Studio: layout w plikach XML



Empty Views Activity

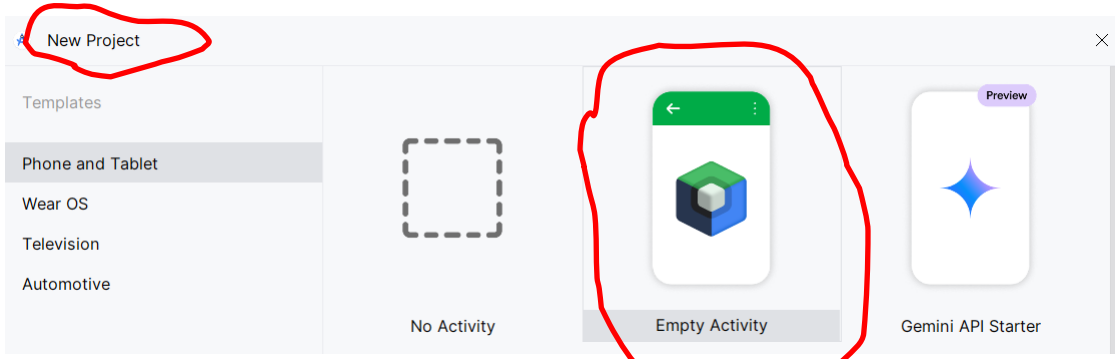
activity_main.xml

Design



tutoriale

tworzenie aplikacji w Android Studio: Jetpack Compose

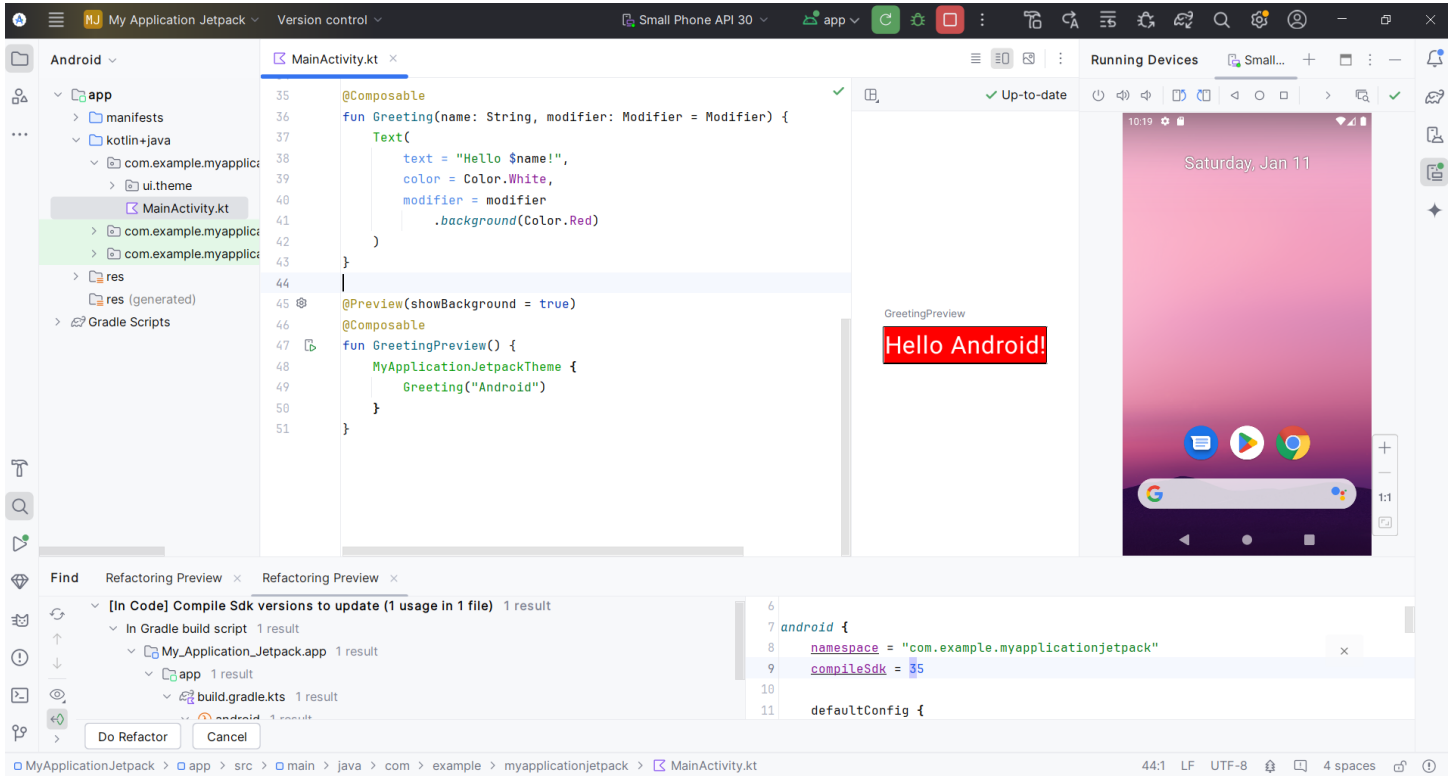


Empty Activity

MainActivity.kt

Design

Virtual device

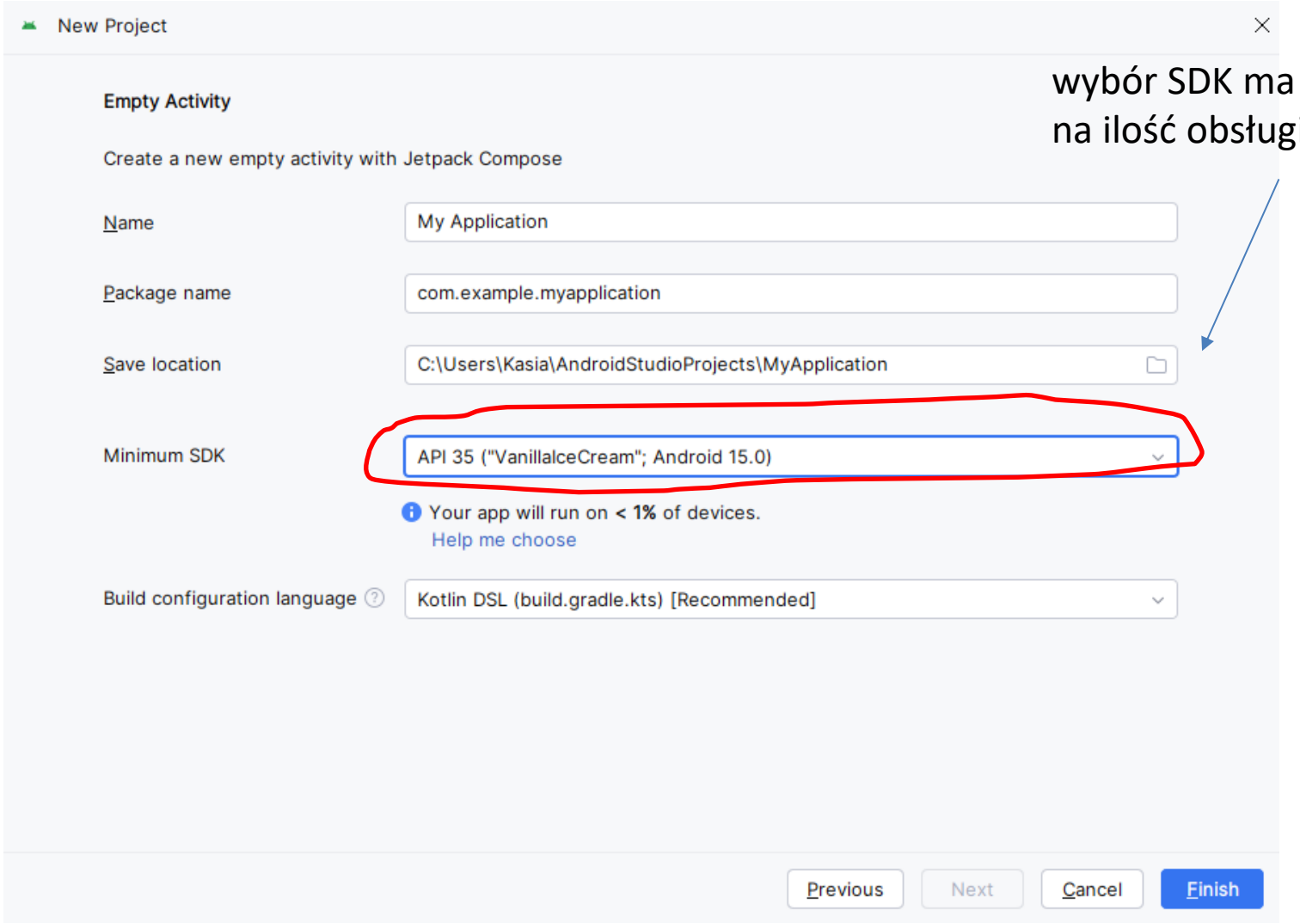


YouTube PL

https://www.youtube.com/watch?v=Ds2J1vmwUEI

https://www.youtube.com/watch?v=ERBEWmfz6h0&list=PLSrm9z4zp4mEWwyiuYgVMWcDFdsebhM-r&index=1

tworzenie aplikacji w Android Studio: Jetpack Compose

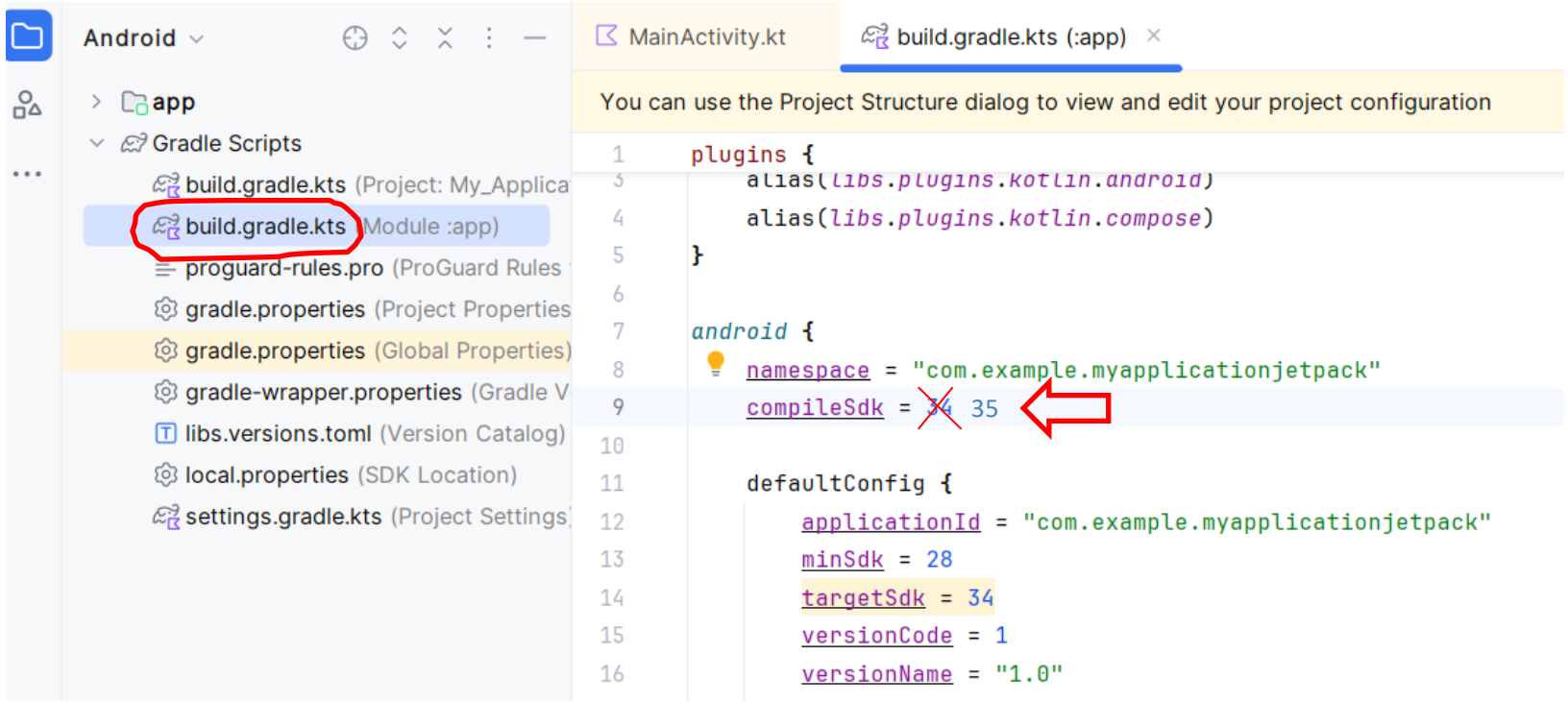


wybór SDK ma wpływ
na ilość obsługiwanych komórek

błąd przy *build & refresh*

```
Recommended action: Update this project to use a newer compileSdk
of at least 35, for example 35.

Note that updating a library or application's compileSdk (which
allows newer APIs to be used) can be done separately from updating
targetSdk (which opts the app in to new runtime behavior) and
minSdk (which determines which devices the app can be installed
on).
```

gdy wybierzemy *Minimum SDK* niższe niż 35 może pojawić się błąd, który naprawiamy



Jetpack Compose

Build better apps faster with Jetpack Compose

Jetpack Compose is Android's recommended modern toolkit for building native UI. It simplifies and accelerates UI development on Android. Quickly bring your app to life with less code, powerful tools, and intuitive Kotlin APIs.

https://developer.android.com/compose

Jetpack Compose is an open-source Kotlin-based declarative UI framework for Android developed by Google. The first preview was announced in May 2019, and the framework was made ready for production in July 2021.

wikipedia

The declarative programming paradigm

Compose to deklaratywny framework interfejsu użytkownika. Dzięki temu budowanie i aktualizowanie interfejsów użytkownika jest znacznie uproszczone. Technika ta polega na koncepcyjnej regeneracji całego ekranu od zera, a następnie zastosowaniu tylko niezbędnych zmian. Takie podejście pozwala uniknąć złożoności ręcznego aktualizowania hierarchii widoków stanowych.

https://developer.android.com/develop/ui/compose/mental-model#paradigm

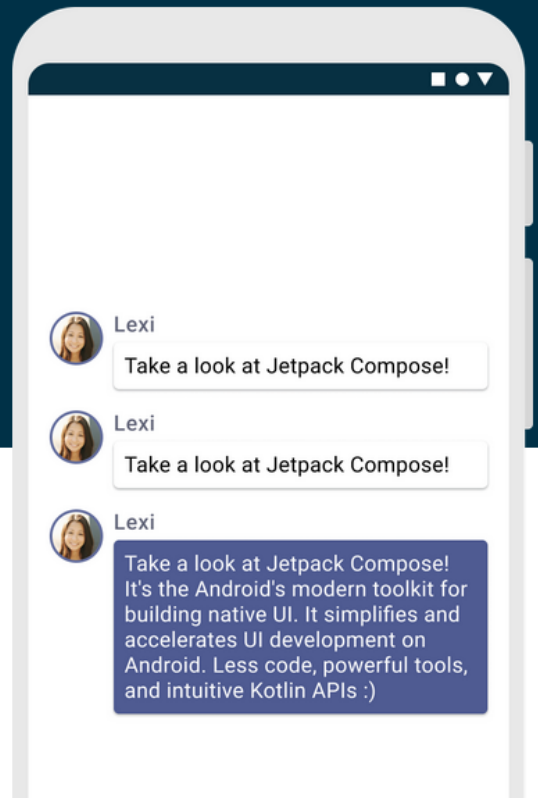
Learning the basics of Compose

SAMOUCZEK

Samouczek w Jetpack Compose

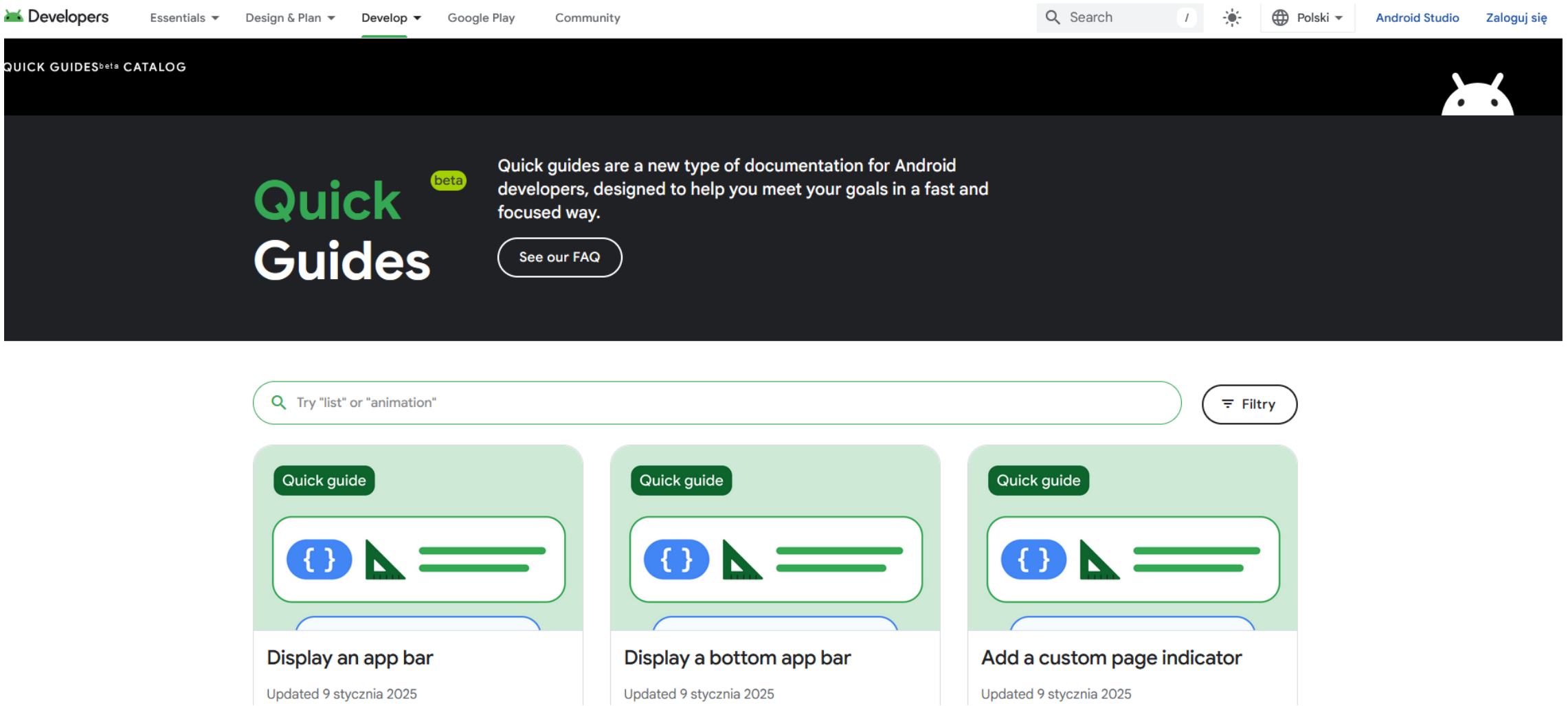
4 lekcje Konfiguracja >

Jetpack Compose to nowoczesny pakiet narzędzi do tworzenia natywnego interfejsu Androida. Jetpack Compose upraszcza i przyspiesza tworzenie UI na Androidzie przy użyciu mniejszej ilości kodu, zaawansowanych narzędzi i intuicyjnych interfejsów API Kotlin.



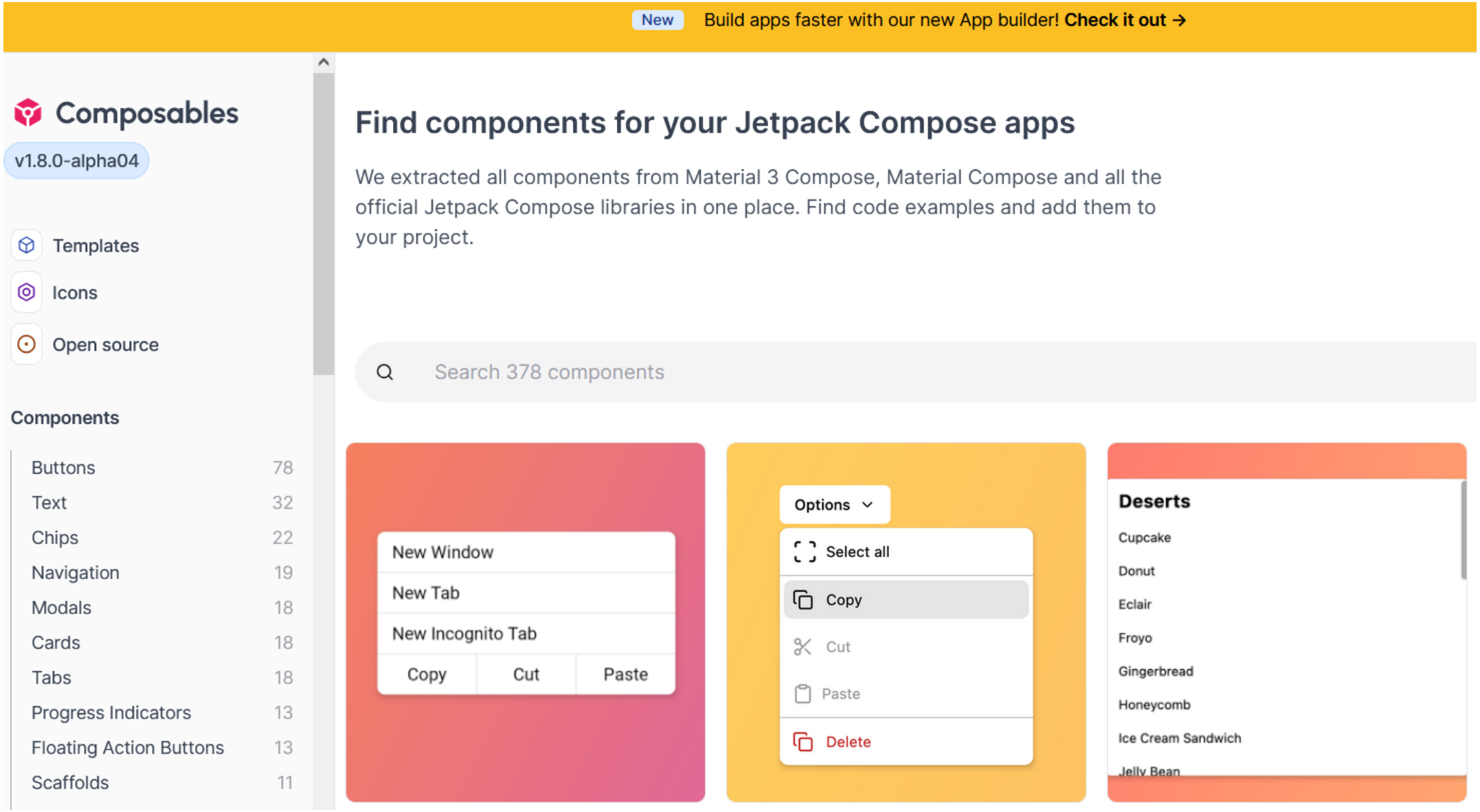
https://developer.android.com/develop/ui/compose/tutorial?hl=pl

Quick Guides Catalog



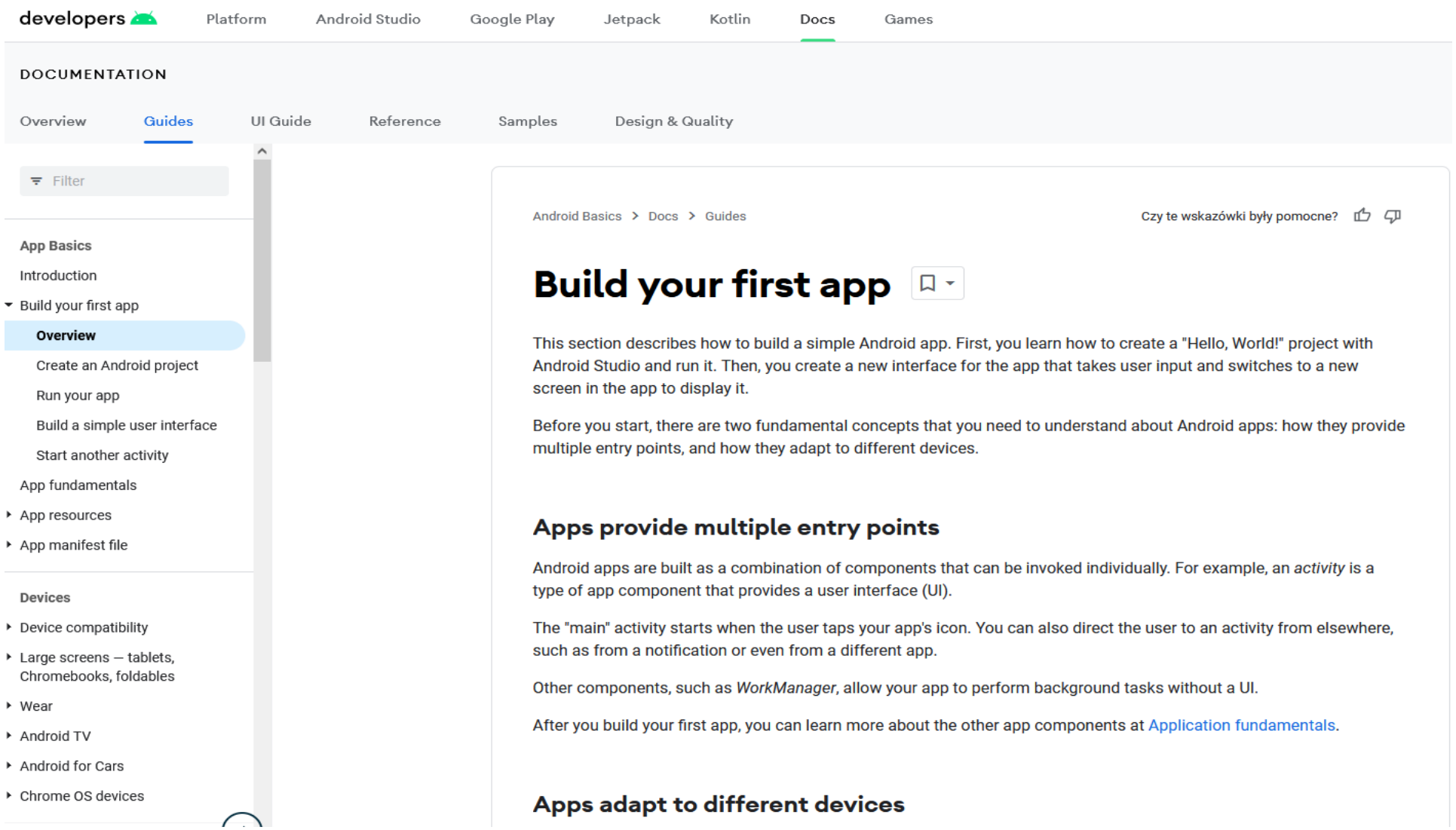
https://developer.android.com/quick-guides

Components



https://composables.com/

First app



https://developer.android.com/training/basics/firstapp

layout

Compose layout basics

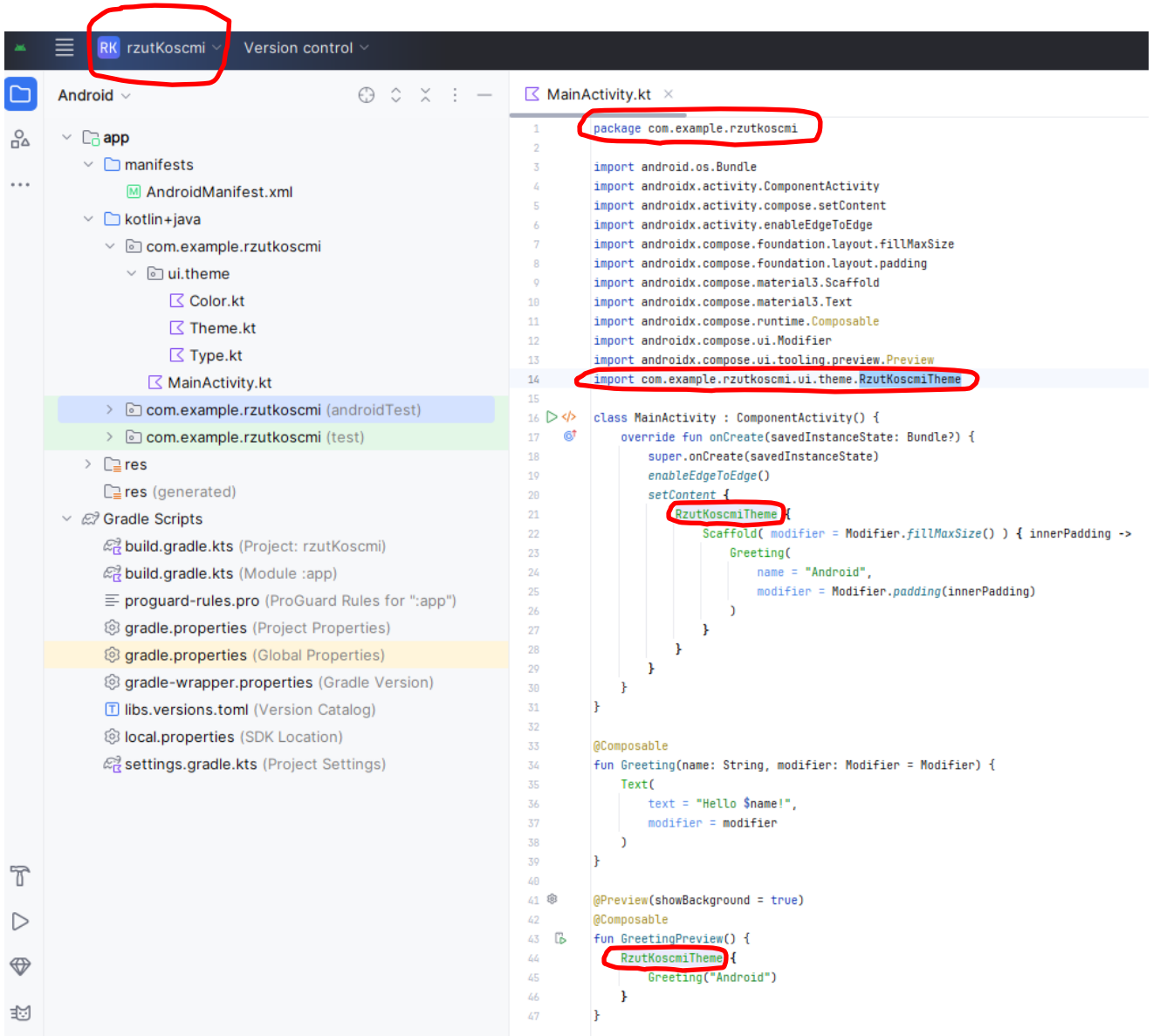
Jetpack Compose makes it much easier to design and build your app's UI. Compose transforms state into UI elements, via:

1. Composition of elements
2. Layout of elements
3. Drawing of elements

https://developer.android.com/develop/ui/compose/layouts/basics

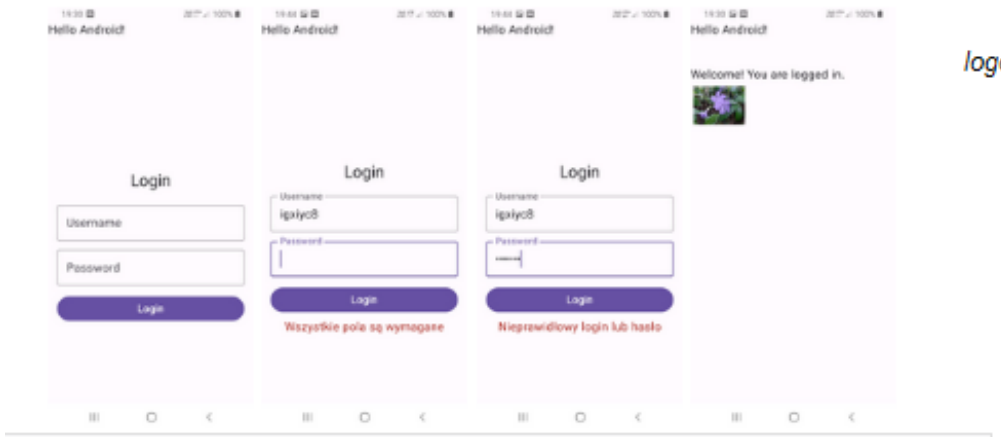
kopiowanie kodu z pałacowej strony (ze zmianą nazwy aplikacji)

utwórz aplikację np.
rzutKoscmi



zaznaczone nazwy przenosimy do skopiowanego kodu

kopiowanie kodu z pałacowej strony (ze zmianą nazwy aplikacji)



logowanie do galerii, komunikaty błędu

http://pracownia.palacmlodziezy.lodz.pl/pliki/materialy/aplikacjeMobilneAndroidStudio/

MainActivity.kt

nazwa aplikacji to ***login2***

kopiujemy kod ***logowanie do galerii***

```
package com.example.login2

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.login2.ui.theme.Login2Theme

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.res.painterResource

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Login2Theme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
```

kopiowanie kodu z pałacowej strony (ze zmianą nazwy aplikacji)

MainActivity.kt

linijka zostawiona z oryginalnego kodu

usunąć

wklejamy kod ***logowanie do galerii*** do naszej aplikacji ***rzutKoscmi*** (usuwamy oryginalny kod zostawiając pierwszą linijkę), ***pojawiają się błędy, które należy naprawić***

import com.example.rzutkoscmi.ui.theme.RzutKoscmiTheme

RzutKoscmiTheme

```
package com.example.rzutkoscmi


package com.example.login2

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.login2.ui.theme.Login2Theme

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.res.painterResource

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Login2Theme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}
```

kopiowanie kodu z pałacowej strony (ze zmianą nazwy aplikacji)

MainActivity.kt

```
42
43  @Composable
44  fun Greeting(name: String, modifier: Modifier = Modifier) {
45
46      Column() {
47          Text(
48              text = "Hello $name!",
49              modifier = modifier
50          )
51          var isLoggedIn by remember { mutableStateOf(false) }
52
53          if (isLoggedIn){
54              gallery()
55          } else {
56              LoginScreen(onLoginSuccess = { isLoggedIn = true })
57          }
58      }
59  }
60
61
62  // TODO
63  @Composable
64  fun gallery() {
65      Column() {
66          Text("Welcome! You are logged in.")
67          Image(
68              painter = painterResource(R.drawable._0220426_071357),
69              contentDescription = "kwiatek",
70              modifier = Modifier
71                  .width(100.dp)
72                  .padding(5.dp)
73          )
74      }
75  }
```

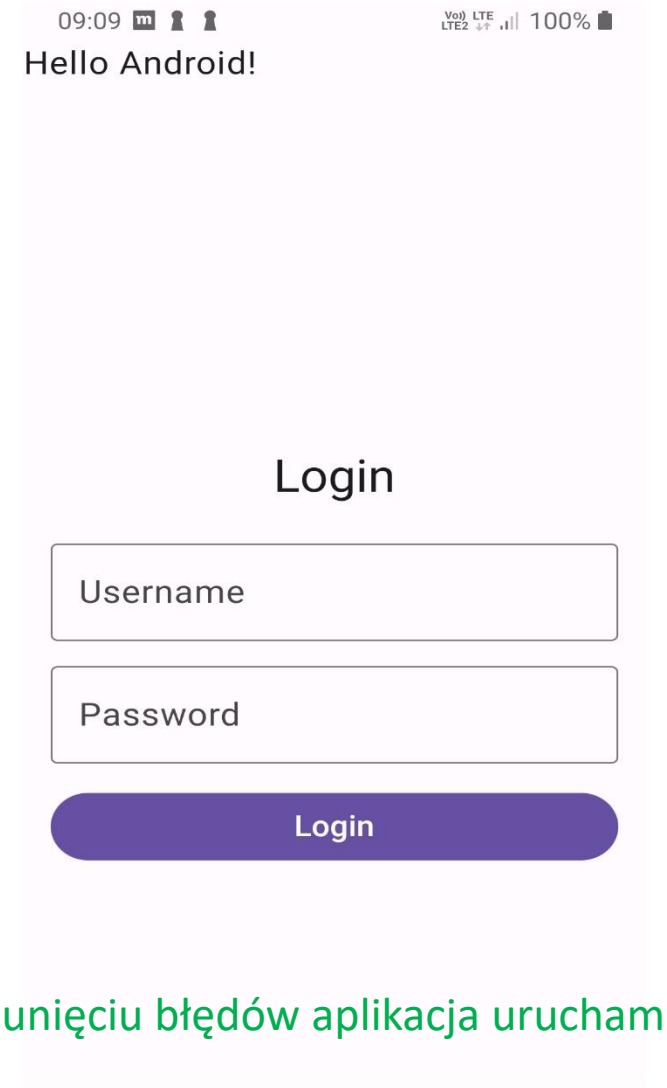
```
145 @Preview(showBackground = true)
146 @Composable
147 fun GreetingPreview() {
148     Login2Theme {
149         Greeting("Android")
150     }
151 }
152
153 source https://deepai.org/chat
154
```

RzutKoscmiTheme

usunąć

komponent ***Image*** usuwamy, gdyż nie mamy oryginalnego zdjęcia w Resource Manager (można wstawić ewentualnie inne)

kopiowanie kodu z pałacowej strony (ze zmianą nazwy aplikacji)



po usunięciu błędów aplikacja uruchamia się

pesel2 Version control Pixel 5 API 35 app

MainActivity.kt

```kotlin
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //enableEdgeToEdge()
        setContent {
            PreviewMessageCard()
        }
    }
}

data class Message(val author: String, val body: String)

@Composable
fun MessageCard(msg: Message){
    Row {
        Image(
            painter = painterResource(R.drawable._0220224_075737),
            contentDescription = "kwiatek",
            modifier = Modifier
                .width(100.dp) // obrazek był za duży
                .padding(5.dp)
        )
        Column (modifier = Modifier.padding(10.dp)){
            Text(msg.author,
                color = Color.Blue,
                modifier = Modifier
                    .background(Color.Yellow)
            )
            Text(msg.body,
                fontSize = 30.sp,
                modifier = Modifier
                    .padding(2.dp)
                    .border(width = 2.dp,
                            color = Color.Blue)
                    .padding(2.dp)
            )
        }
    }
}

@Preview
@Composable
fun PreviewMessageCard(){
    MessageCard(Message("Marek", "Hello Everybody"))
}
```

density-independent pixels

scaled pixels

Up-to-date

PreviewMessageCard

Marek

Hello Everybody

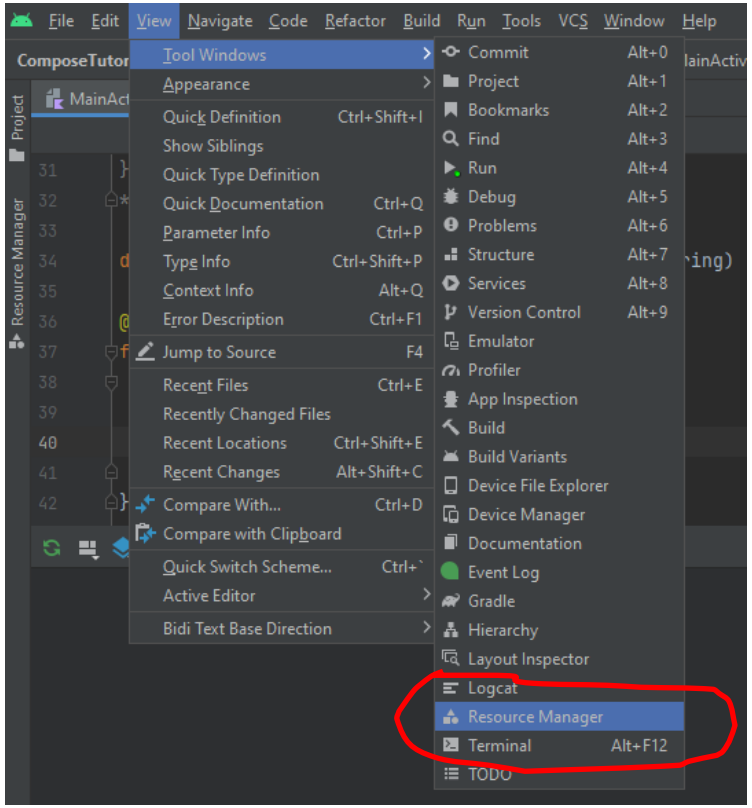
Running Devices

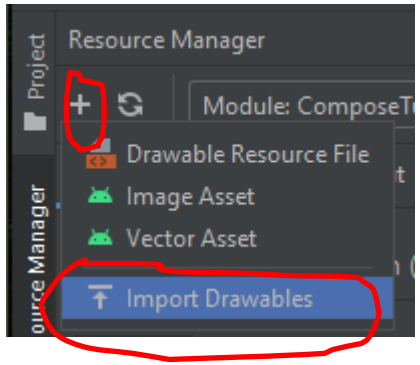
Problems File 4 Project Errors Compose 1

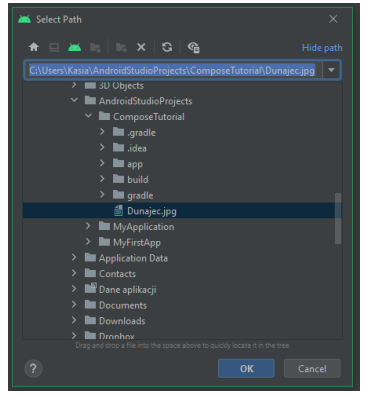
pesel2 > app > src > main > java > com > example > pesel2 > MainActivity.kt

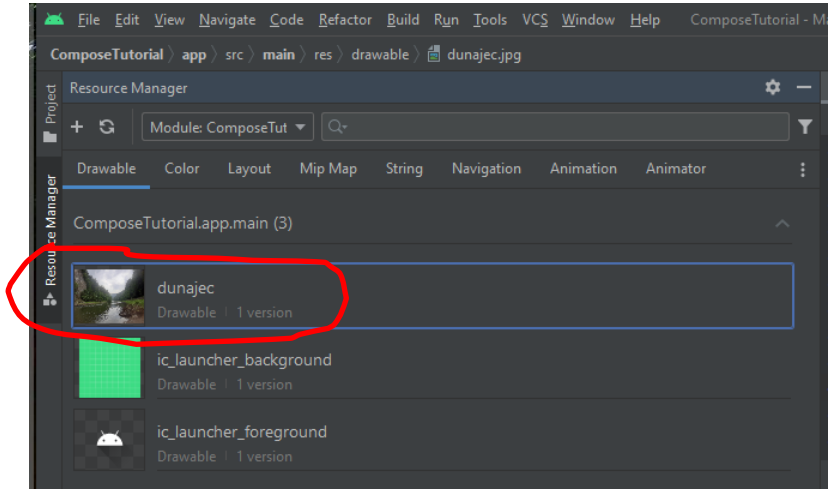
69:1 LF UTF-8 4 spaces

Resource Manager









https://developer.android.com/studio/write/resource-manager#import

Text() - argumenty

✓ **text: String**, modifier: Modifier = Modifier, color: Color = Color.Unspecified, fontSize: TextUnit = TextUnit.Unspecified, fontStyle: FontStyle? = null, fontWeight: FontWeight? = null, fontFamily: FontFamily? = null, letterSpacing: TextUnit = TextUnit.Unspecified, textDecoration: TextDecoration? = null, textAlign: TextAlign? = null, lineHeight: TextUnit = TextUnit.Unspecified, overflow: TextOverflow = TextOverflow.Clip, softWrap: Boolean = true, maxLines: Int = Int.MAX_VALUE, minLines: Int = 1, onTextLayout: ((TextLayoutResult) -> Unit)? = null, style: TextStyle = LocalTextStyle.current

```
Text(msg.author)
Text(msg.body)
}
```

Hello Everybody

```
@Composable
public fun Text(
    text: String,
    modifier: Modifier = Modifier,
    color: Color = Color.Unspecified,
    fontSize: TextUnit = TextUnit.Unspecified,
    fontStyle: FontStyle? = null,
    fontWeight: FontWeight? = null,
    fontFamily: FontFamily? = null,
    letterSpacing: TextUnit = TextUnit.Unspecified,
    textDecoration: TextDecoration? = null,
    textAlign: TextAlign? = null,
    lineHeight: TextUnit = TextUnit.Unspecified,
    overflow: TextOverflow = TextOverflow.Clip,
    softWrap: Boolean = true,
    maxLines: Int = Int.MAX_VALUE,
    minLines: Int = 1,
    onTextLayout: ((TextLayoutResult) -> Unit)? = null,
    style: TextStyle = LocalTextStyle.current
): Unit
```

High level element that displays text and provides semantics / accessibility information.

The default `style` uses the `LocalTextStyle` provided by the `MaterialTheme` / components. If you are setting your own style, you may want to consider first retrieving `LocalTextStyle` and using

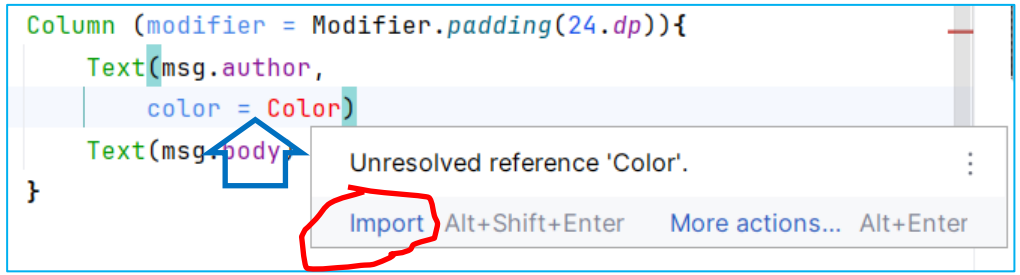
wstaw kursor do wnętrza nawiasów i przyciśnij CTRL + p

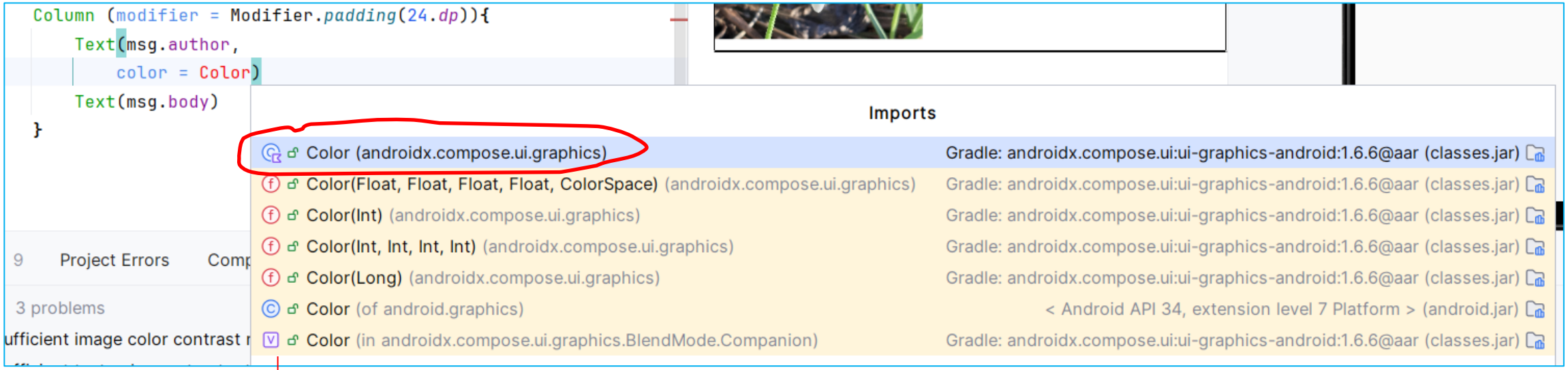
najedź kursorem na słowo ***Text***

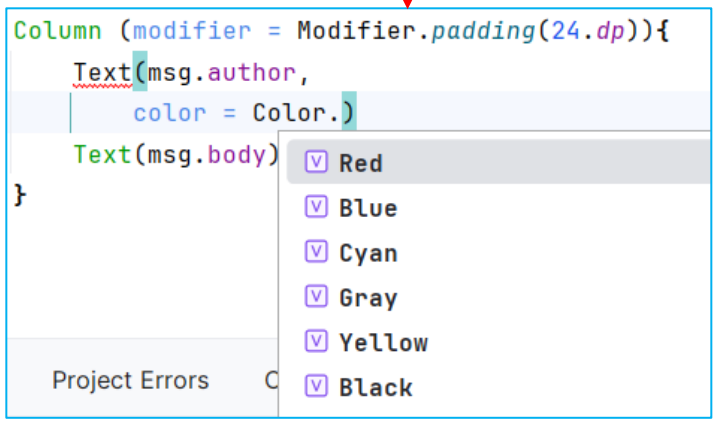
[style text]

poeksperymentuj ☺

Unresolved reference







wypisywanie logów w konsoli Logcat

MainActivity.kt

```kotlin
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            LogiTheme {
                Logi()
            }
        }
    }
}

@Composable
fun Logi() {
    // lista prezentów
    val prezenty = listOf("hulajnoga", "rower", "książka", "smartwatch", "tablet")

    // LazyColumn wyświetla listę prezentów
    LazyColumn() {
        items(prezenty) { item -> // 'item' to element listy
            // Wyświetlanie pojedynczego elementu
            Text(
                text = item,
                modifier = Modifier
                    .clickable {
                        Log.i("==== mój log ======","kliknięto $item")
                    }
            )
        }
    }

}

@Preview(showBackground = true)
@Composable
fun LogiPreview() {
    LogiTheme {
        Logi()
    }
}
```

MainActivity.kt

```kotlin
package com.example.logi

import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.clickable
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items   // items
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.example.logi.ui.theme.LogiTheme
```

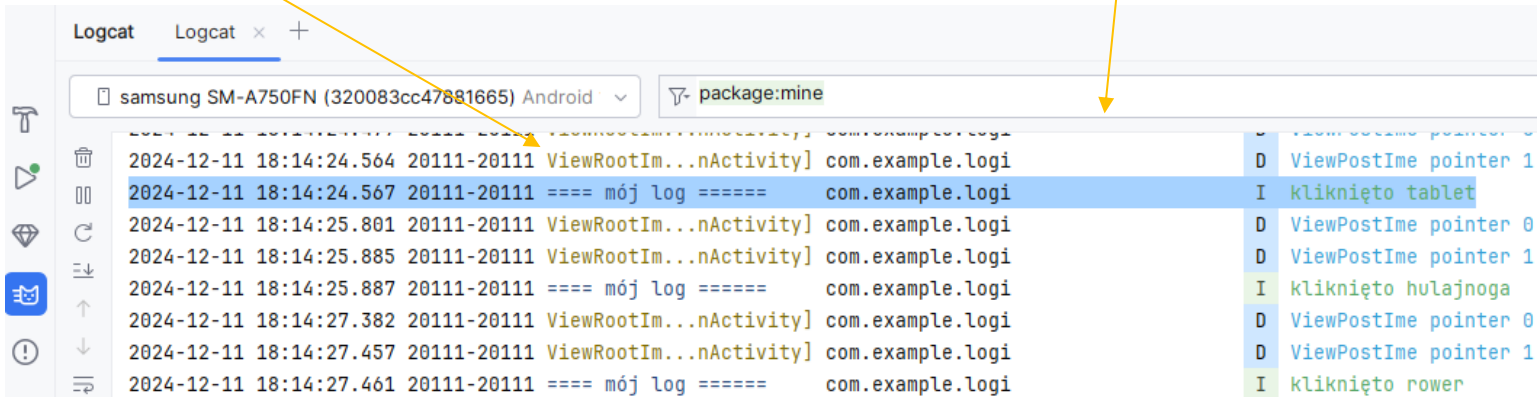
klikalna lista elementów

po kliknięciu na element listy pojawia się log w Logcat

android developer

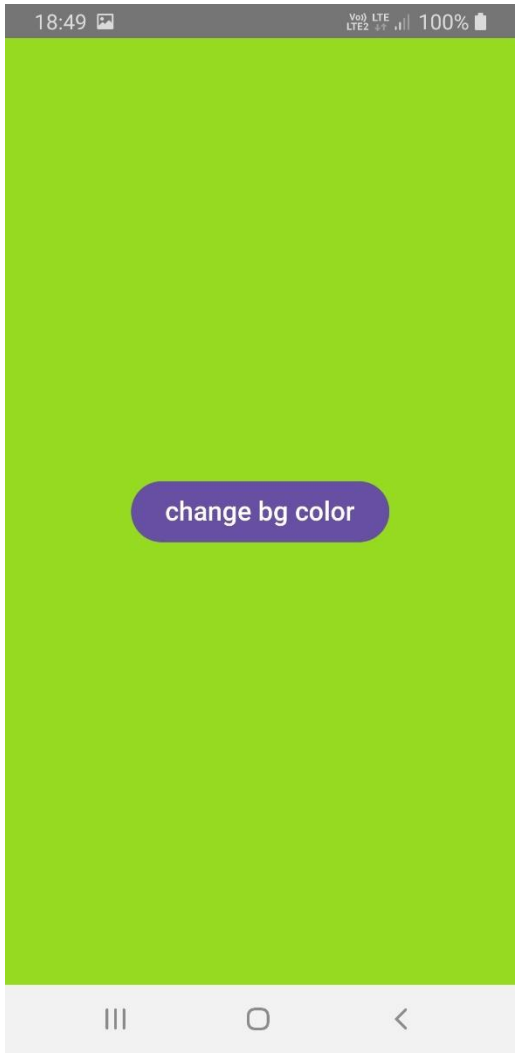
log informacyjny

Logcat



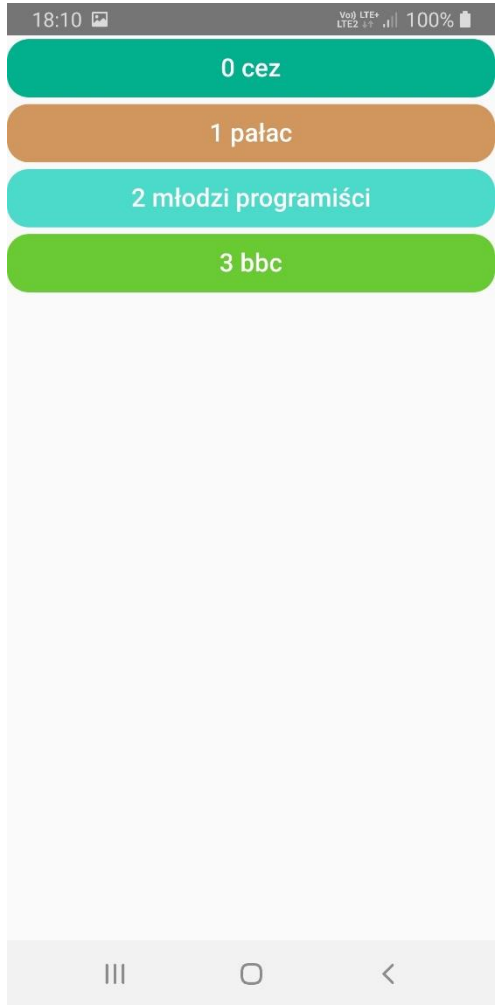
button

```
@Composable
fun  MyButtons(){
    // stan przycisku - zmienna bgColor - gdy klikniemy na przycisk zmieni się -> upadate aplikacji
    // Compose is declarative and as such the only way to update it is by calling the same composable with new arguments.
    // These arguments are representations of the UI state.
    /*
    "by" - allows the variable bgColor to delegate its getter and setter to the underlying state holder, which in this case is mutableStateOf.
    "remember" - Composable functions can use the remember API to store an object in memory.
    A value computed by remember is stored in the Composition during initial composition,
    and the stored value is returned during recomposition.
    remember can be used to store both mutable and immutable objects.
    "mutableStateOf(Color.Green)" - This creates a mutable state holder that holds a value (in this case, a color).
    mutableStateOf creates a state that can be read and modified,
    and when the value changes, it will inform the Compose UI to recompose (update) any composables that are dependent on that state.
    The initial value of this state is set to Color.Green.
    source :https://deepai.org/chat, https://developer.android.com/
     */
    var bgColor by remember { mutableStateOf(Color.Green) }

    // funkcja wywoływana po przyciśnieciu przycisku
    fun onClick(){
        // Generate a random color
        /*
        0xFF shl 24 is a way to set the alpha channel of a color to fully opaque in a 32-bit ARGB color value.
        The hexadecimal value 0xFF represents the alpha channel (opacity) and is 11111111 in binary.
        After shifting left by 24 bits: 11111111000000000000000000000000 in binary, or 0xFF000000 in hexadecimal.
        Math.random() - generates a random floating-point number between 0.0 (inclusive) and 1.0 (exclusive).
        The multiplication 0xFFFFFF * Math.random() gives you a floating-point number in the range [0, 0xFFFFFF)
        (or [0, 16777215) in decimal), which means it can represent any RGB color value.
        OR operation - combines all components into a single ARGB
        source :https://deepai.org/chat, https://developer.android.com/
        */
        bgColor = Color((0xFF shl 24) or (0xFFFFFF * Math.random()).toInt())
    }

    // layout
    Column(modifier = Modifier              // layout - obiekty układają sie jeden pod drugim (stos)
        .fillMaxSize()                      // cały ekran
        .background(bgColor)                // kolor tła jest pobierany ze zmiennej stanu bgColor
        .wrapContentSize(Alignment.Center)  // przycisk na środku ekranu
    ) {
        Button(onClick = {onClick()}){      // przycisk, po kliknieciu wywoływana jest funkcja onClick() zmieniająca zmienną stanu bgColor
            Text("change bg color")         // napis na przycisku
        }
    }
}
```

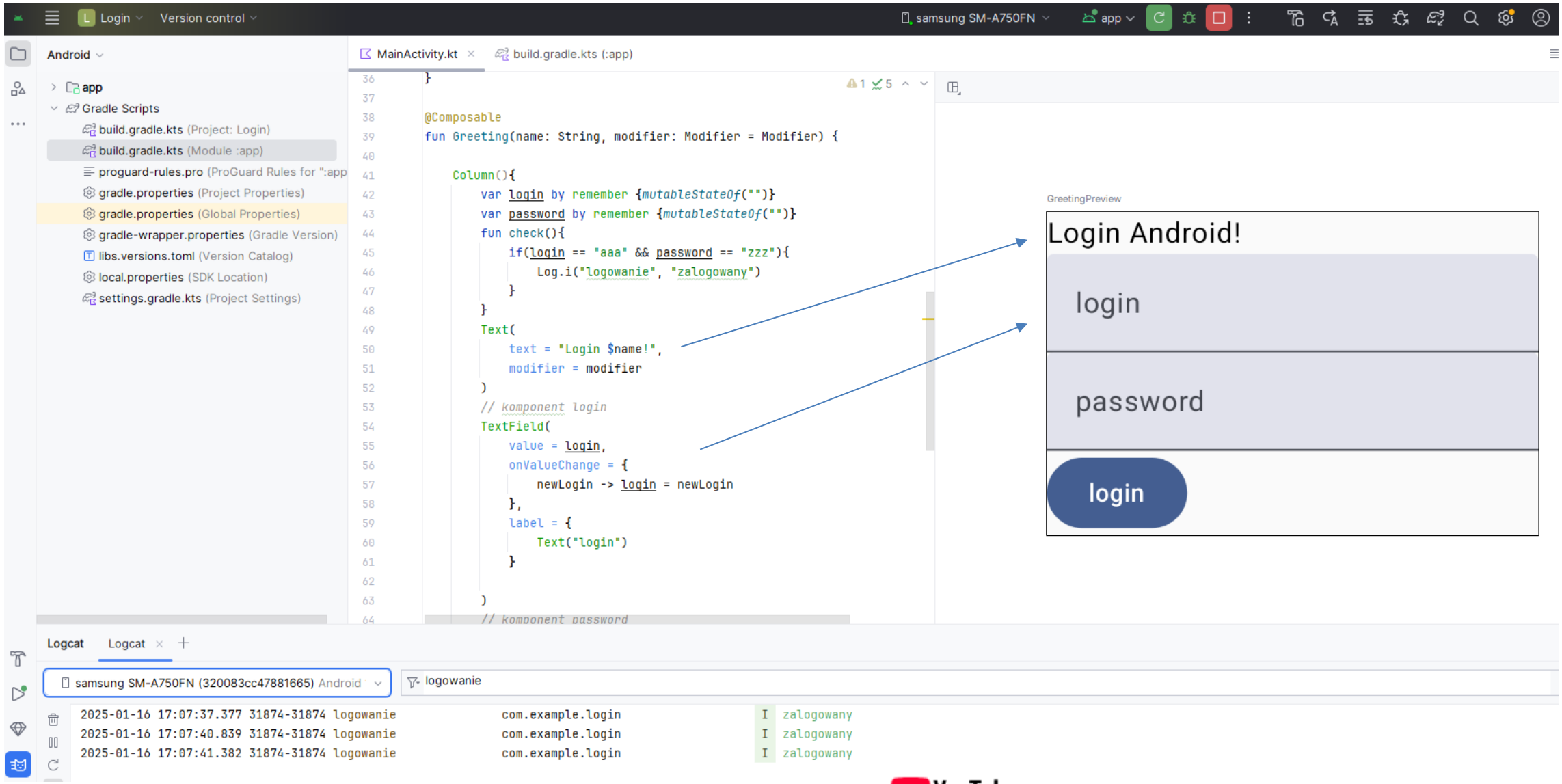


otwieranie przeglądarki

```
@Composable
fun WebPages() {
    val context = LocalContext.current    //  dostęp do kontekstu w komponencie
    val urls = listOf(
        "https://cez.lodz.pl",
        "http://www.palacmlodziezy.lodz.pl",
        "http://pracownia.palacmlodziezy.lodz.pl",
        "http://www.bbc.com"
    )

    val buttonLabels = listOf(
        "cez",
        "pałac",
        "młodzi programiści",
        "bbc"
    )

    Column {
        urls.forEachIndexed { index, url ->    // petla generująca przyciski
            val label = buttonLabels[index]
            //Log.i("==== przycisk ======", "Przycisk $label na URL: $url")

            Button(
                colors = ButtonDefaults.buttonColors(
                    containerColor = Color((0xFF shl 24) or (0xFFFFFF * Math.random()).toInt()), // Kolor tła
                    contentColor = Color.White // Kolor tekstu
                ),
                shape = MaterialTheme.shapes.large,
                onClick = {
                    val intent = Intent(Intent.ACTION_VIEW).apply {   // tworzymy nowy Intent z akcją Intent.ACTION_VIEW,
                        data = Uri.parse(url)                         // jest to standardowy sposób na otwieranie stron internetowych.
                    }
                    context.startActivity(intent)                     // uruchamiamy przeglądarkę
                },
                modifier = Modifier.fillMaxWidth()                    // przycisk wypełnia całą szerokość
            ) {
                Text(text = "$index $label")                          // tekst na przycisku
            }
        }
    }
```



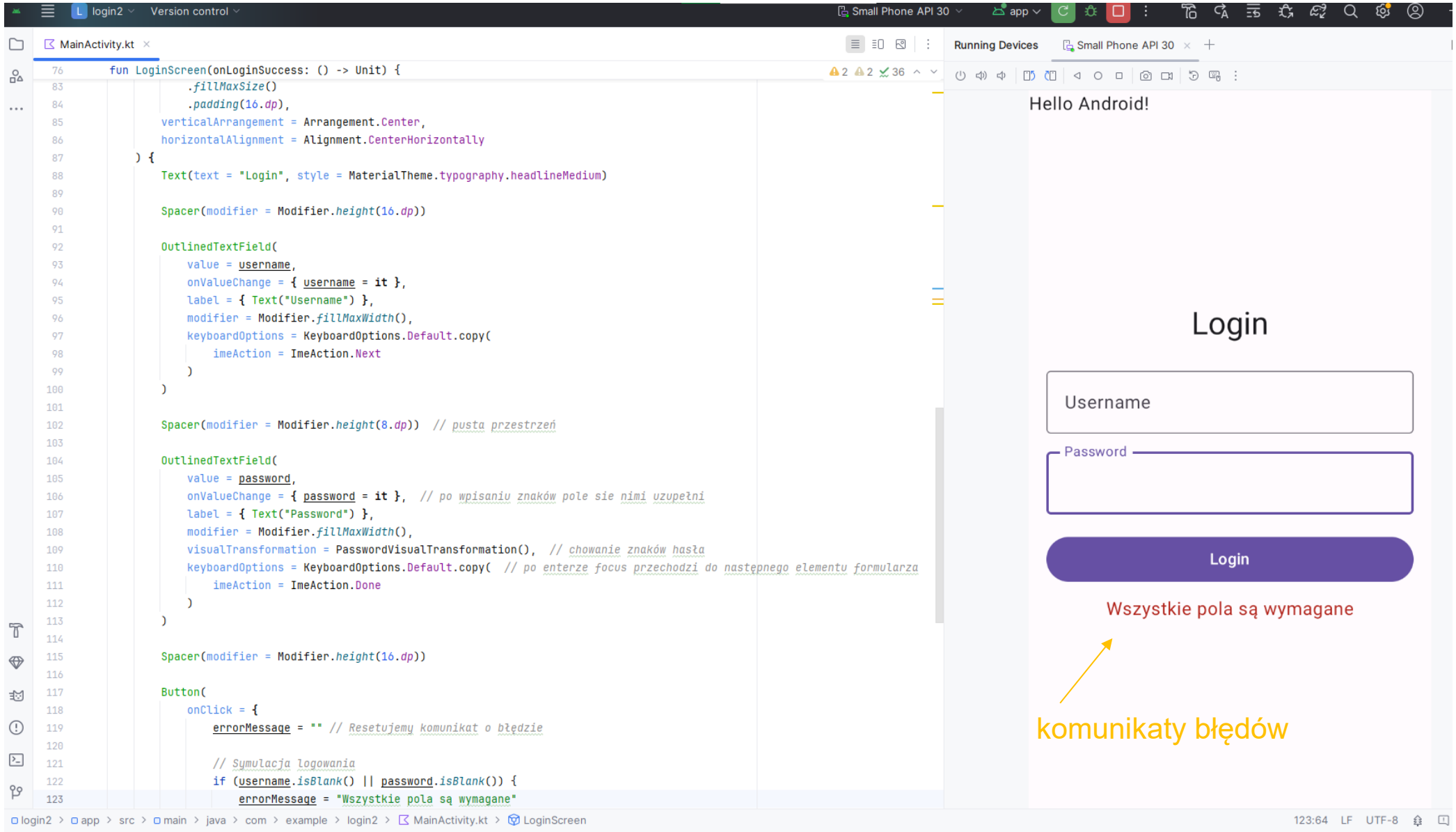
logowanie



YouTube

https://www.youtube.com/watch?v=6w4l-3jC21E&list=PLSrm9z4zp4mEWwyiuYgVMWcDFdsebhM-r&index=11

logowanie 2


login2
Version control
Small Phone API 30
app
MainActivity.kt
fun LoginScreen(onLoginSuccess: () -> Unit) {
.fillMaxSize()
.padding(16.dp),
verticalArrangement = Arrangement.Center,
horizontalAlignment = Alignment.CenterHorizontally
) {
Text(text = "Login", style = MaterialTheme.typography.headlineMedium)
Spacer(modifier = Modifier.height(16.dp))
OutlinedTextField(
value = username,
onValueChange = { username = it },
label = { Text("Username") },
modifier = Modifier.fillMaxWidth(),
keyboardOptions = KeyboardOptions.Default.copy(
imeAction = ImeAction.Next
)
)
Spacer(modifier = Modifier.height(8.dp)) // pusta przestrzeń
OutlinedTextField(
value = password,
onValueChange = { password = it }, // po wpisaniu znaków pole sie nimi uzupełni
label = { Text("Password") },
modifier = Modifier.fillMaxWidth(),
visualTransformation = PasswordVisualTransformation(), // chowanie znaków hasła
keyboardOptions = KeyboardOptions.Default.copy(// po enterze focus przechodzi do nastepnego elementu formularza
imeAction = ImeAction.Done
)
)
Spacer(modifier = Modifier.height(16.dp))
Button(
onClick = {
errorMessage = "" // Resetujemy komunikat o błędzie
// Symulacja logowania
if (username.isBlank() || password.isBlank()) {
errorMessage = "Wszystkie pola są wymagane"
Running Devices
Small Phone API 30
Hello Android!
Login
Username
Password
Login
Wszystkie pola są wymagane
komunikaty błędów
login2 > app > src > main > java > com > example > login2 > MainActivity.kt > LoginScreen
123:64 LF UTF-8

```
OutlinedTextField(
    keyboardOptions = KeyboardOptions.Default.copy(
        imeAction = ImeAction.Done
    )
)
```

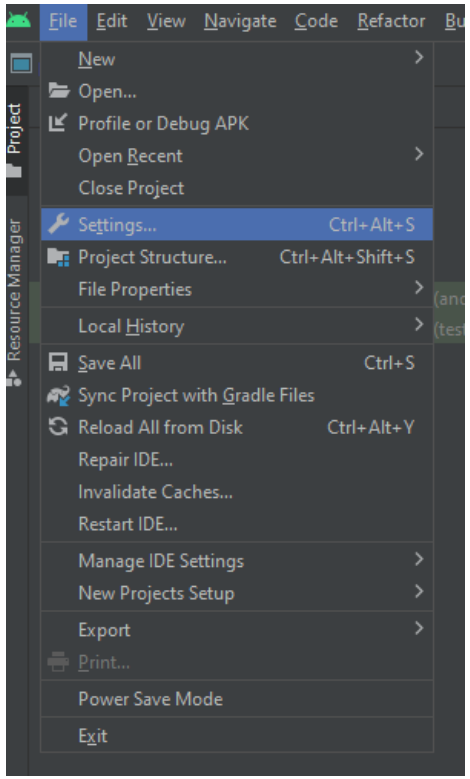
1. `KeyboardOptions.Default`:
    - To jest domyślna instancja obiektu `KeyboardOptions`. `KeyboardOptions` to klasa, która definiuje różne opcje związane z klawiaturą, takie jak typ klawiatury, akcja IME (Input Method Editor) i inne. Korzystając z `KeyboardOptions.Default`, zaczynamy z zestawem predefiniowanych opcji, które są standardowo używane w klawiaturze.

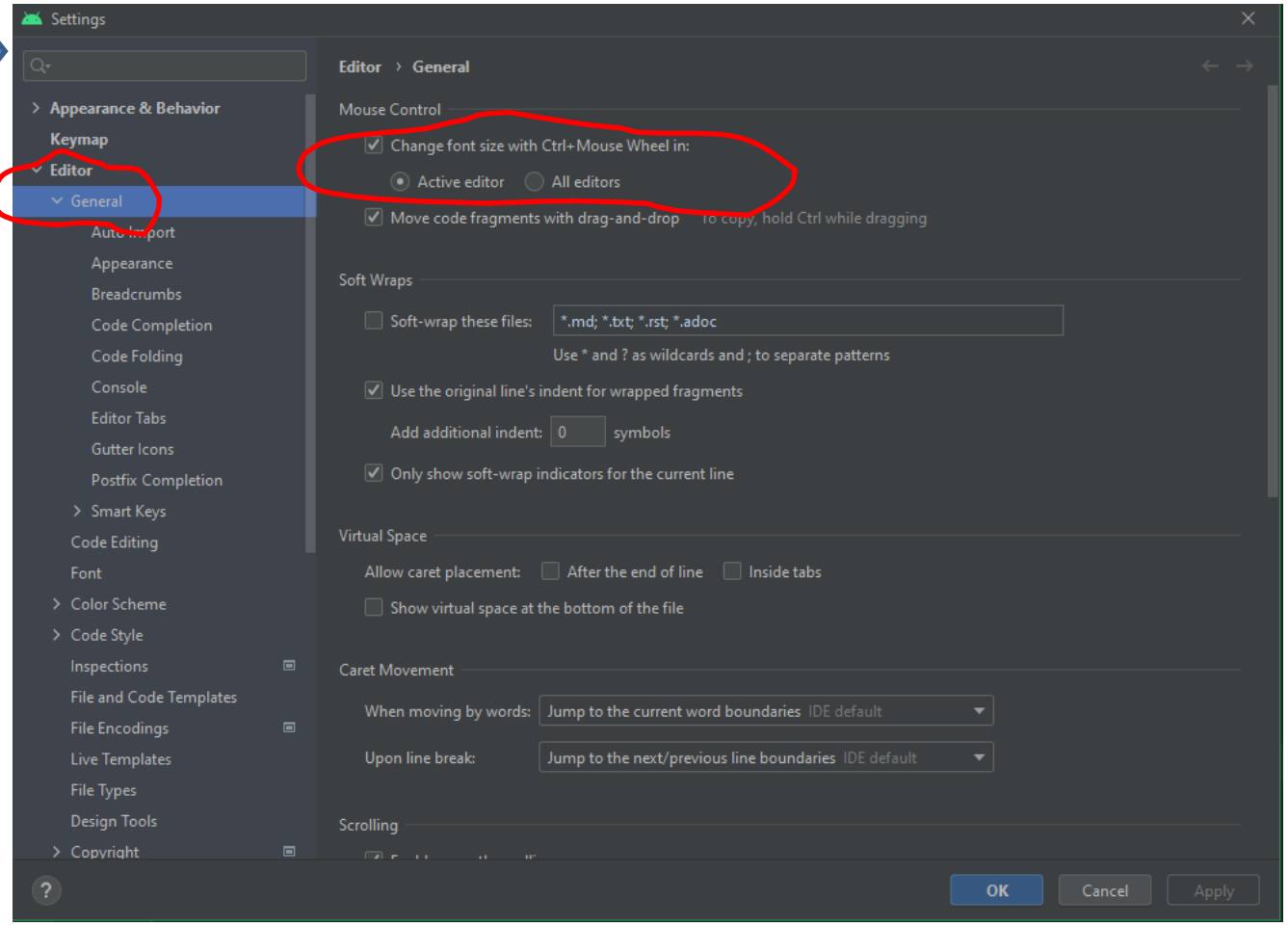
2. `.copy(...)`:
    - `copy` to metoda, która jest dostępna w wielu klasach danych w Kotlinie. Umożliwia ona utworzenie nowej instancji obiektu na podstawie istniejącego obiektu, ale z możliwością zmiany niektórych właściwości. Tym sposobem możemy modyfikować tylko te parametry, które nas interesują, a resztę pozostawić niezmienioną.

3. `imeAction = ImeAction.Next`:
    - To jest konkretna modyfikacja, którą dokonujemy na domyślnych opcjach klawiatury. `imeAction` określa, co ma się stać, gdy użytkownik naciśnie przycisk „Enter” na klawiaturze. Ustawienie `ImeAction.Next` wskazuje, że po naciśnięciu przycisku „Next” (zwykle oznaczonego ikoną strzałki lub napisem „Dalej”), fokus przechodzi do następnego elementu formularza (np. do następnego pola tekstowego).

source: https://deepai.org/chat

Android Studio zmiana wielkości czcionek
CTRL Mouse Wheel





Device connecting

translated by Google API. Ta strona została przetłumaczona przez Cloud Translation

Switch to English

Android Developers > Programowanie > Android Studio > Edytor Android Studio

Czy te wskazówki były pomocne?

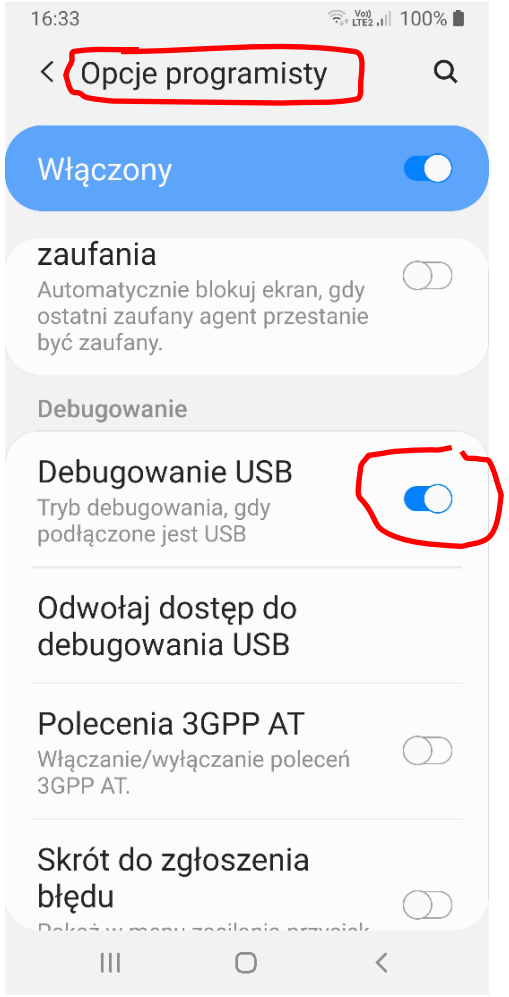
Uruchamianie aplikacji na urządzeniu sprzętowym

Przed udostępnieniem aplikacji użytkownikom zawsze testuj ją na prawdziwym urządzeniu. Na tej stronie opisaliśmy, jak skonfigurować środowisko programistyczne i urządzenie z Androidem do testowania i debugowania za pomocą połączenia Android Debug Bridge (ADB).

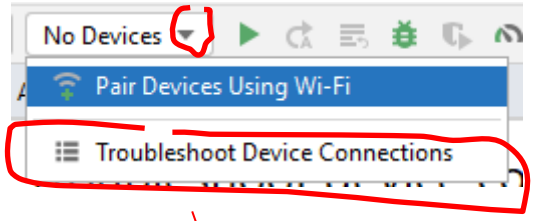
Uwaga: do testowania aplikacji użyj emulatora Androida różnych wersji platformy Androida i różnych rozmiarów ekranu. Rozważ użycie Laboratorium Firebase do uruchomienia aplikacji na wielu różnych prawdziwych urządzeniach hostowanych w infrastrukturze opartej na chmurze.

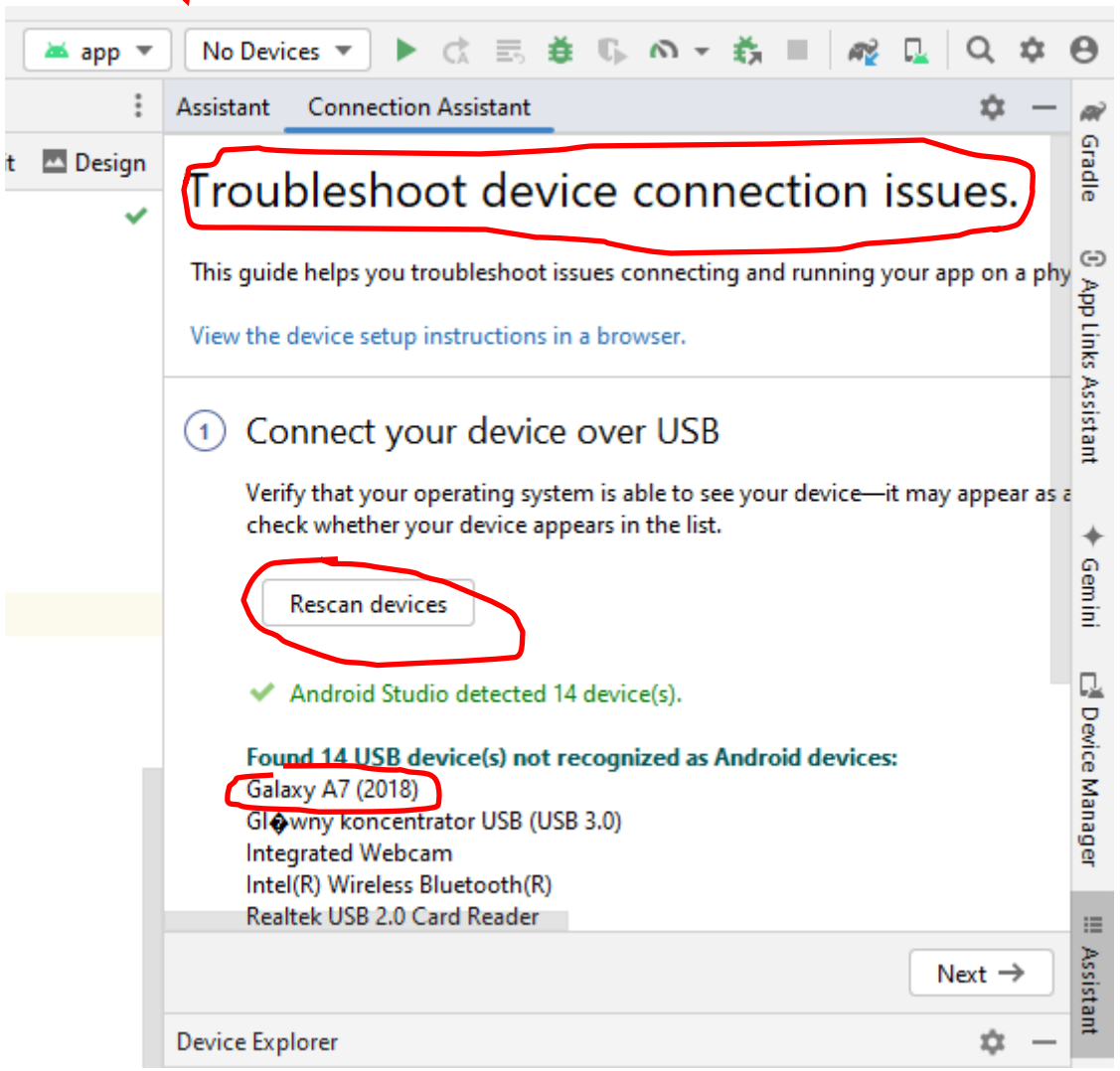
https://developer.android.com/studio/run/device?hl=pl

Device connecting problem



Device connecting problem





Zainstaluj sterownik USB

Najpierw w tabeli Sterowniki OEM znajdź odpowiedni sterownik dla swojego urządzenia.

Po pobraniu sterownika USB postępuj zgodnie z poniższymi instrukcjami, aby zainstalować lub uaktualnić sterownik w zależności od wersji systemu Windows i tego, czy instalujesz sterownik po raz pierwszy, czy uaktualniasz dotychczasowy. Następnie w sekcji Korzystanie ze sprzętu znajdziesz inne ważne informacje o korzystaniu z urządzenia z Androidem podczas prac programistycznych.

Uwaga: można wprowadzić zmiany w pliku **`android_winusb.inf`** znajdującym się w **`usb_driver\`** (na przykład aby dodać obsługę innych urządzeń), jednak podczas instalowania lub uaktualniania sterownika pojawią się ostrzeżenia dotyczące bezpieczeństwa. Wprowadzanie innych zmian w plikach sterowników może zakłócić proces instalacji.

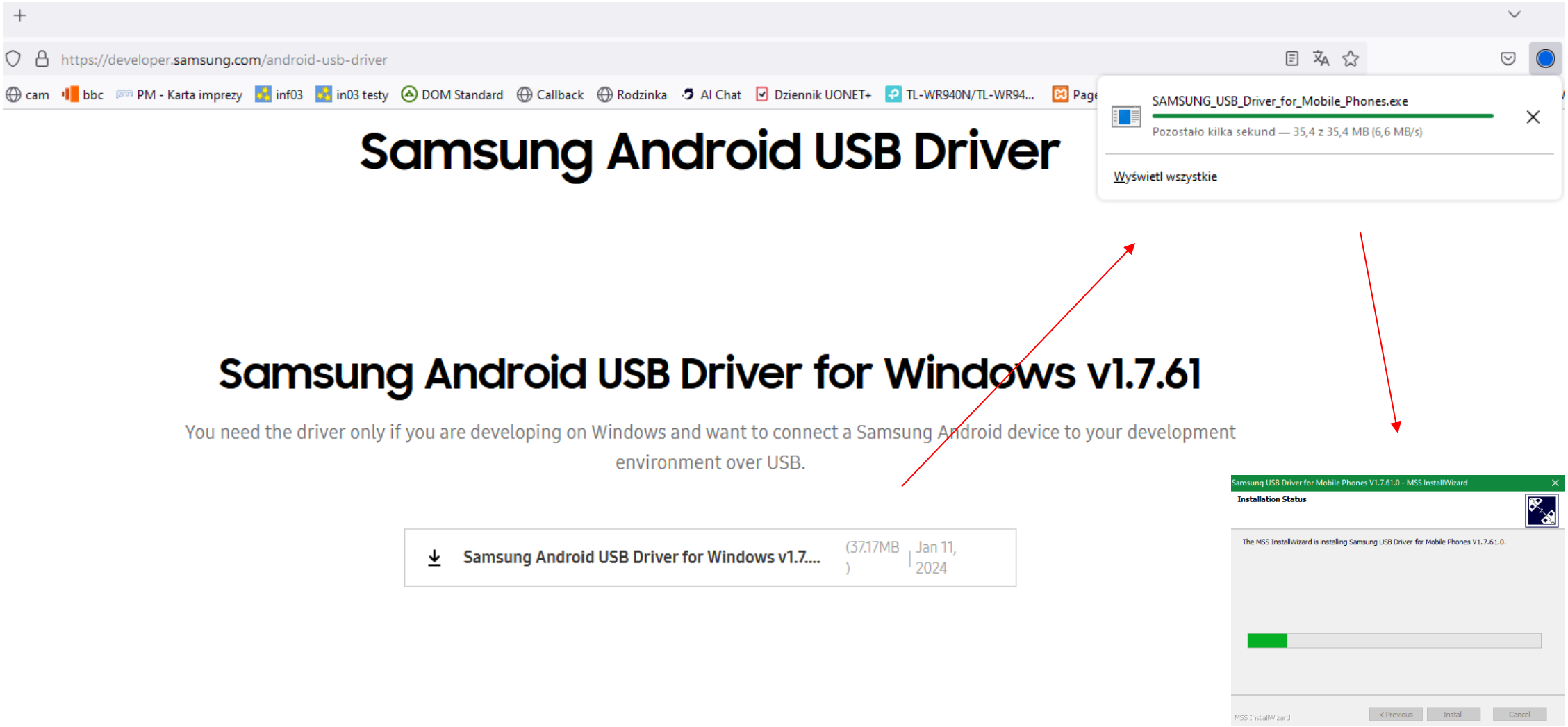
Windows 10

Aby po raz pierwszy zainstalować sterownik USB na urządzeniu z Androidem w systemie Windows 10, wykonaj te czynności:

1. Podłącz urządzenie z Androidem do portu USB komputera.
2. W Eksploratorze Windows otwórz **Zarządzanie komputerem**.
3. W panelu **Zarządzanie komputerem** po lewej stronie kliknij **Menedżer urządzeń**.
4. W panelu **Menedżer urządzeń** po prawej stronie znajdź i rozwiń **Urządzenia przenośne** lub **Inne urządzenia**, w zależności od tego, które urządzenie widzisz.

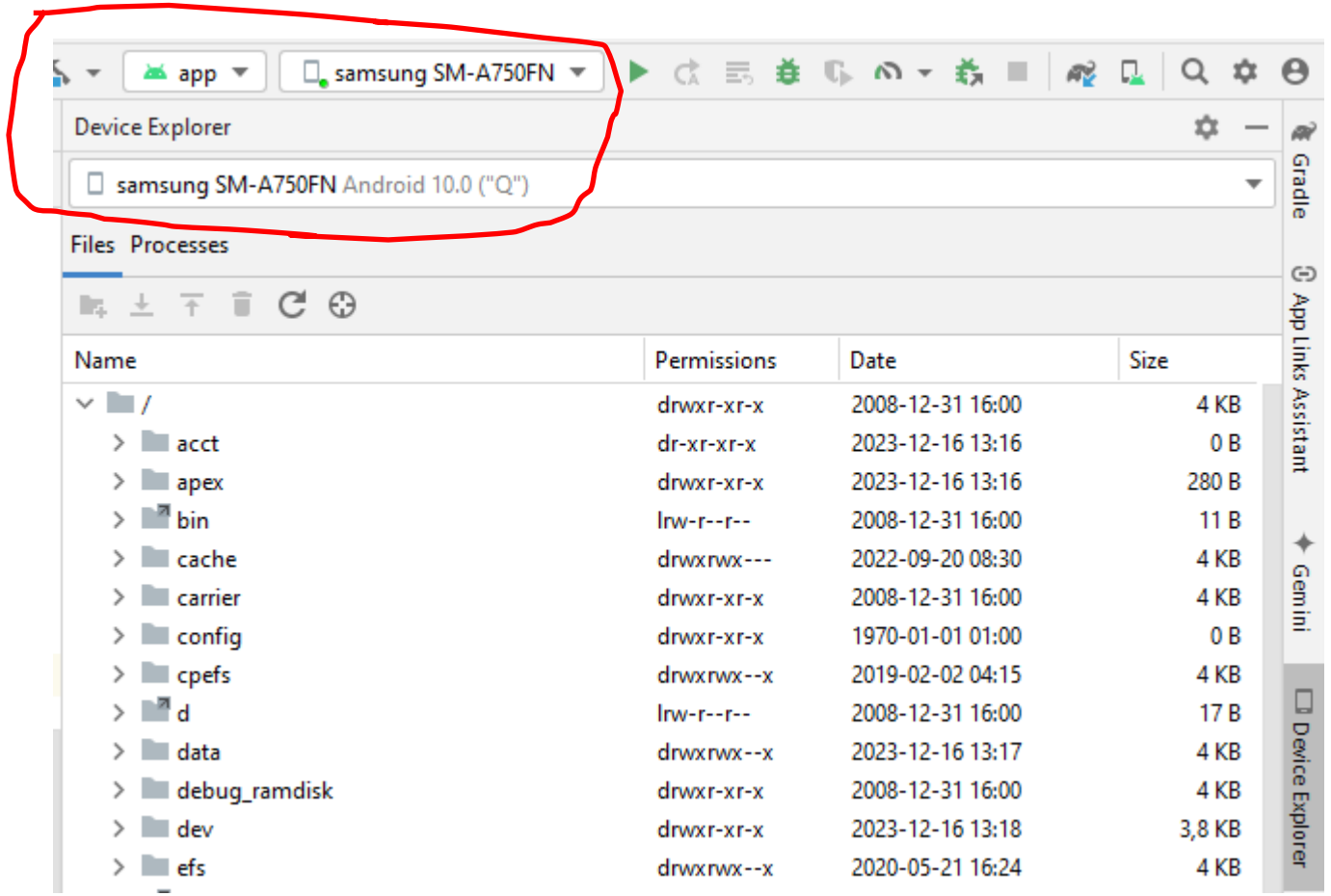
https://developer.android.com/studio/run/oem-usb?hl=pl

Device connecting problem



https://developer.android.com/studio/run/oem-usb?hl=pl#Drivers

Device connecting



// first app

https://developer.android.com/training/basics/firstapp

// samples

https://developer.android.com/samples?language=kotlin&skill=beginner

https://developer.android.com/guide/components/fundamentals

https://developer.android.com/guide/practices/compatibility

https://developer.android.com/courses/android-basics-kotlin/course

// **Build a Responsive UI with ConstraintLayout**

https://developer.android.com/develop/ui/views/layout/constraint-layout