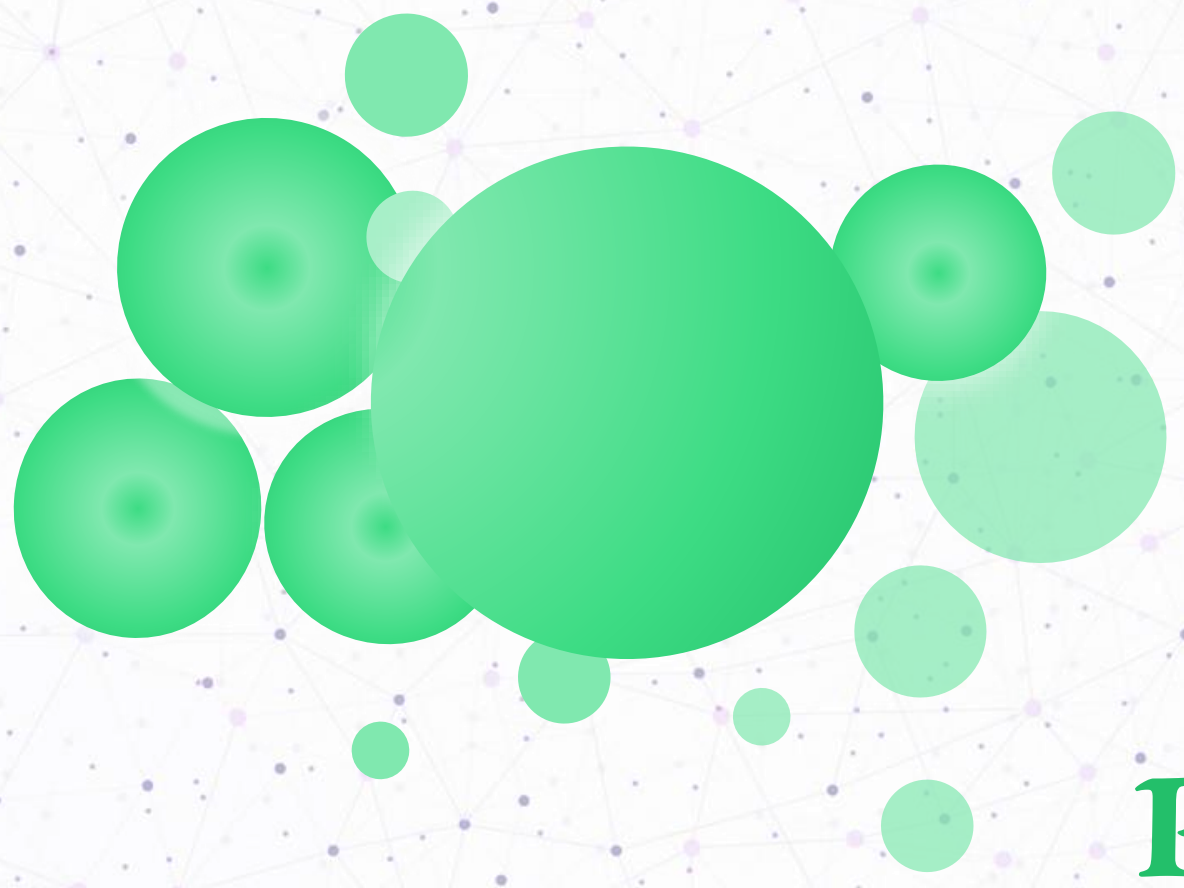


PV256
VÝVOJ NA ANDROID

Ing. Štefan Krajanec

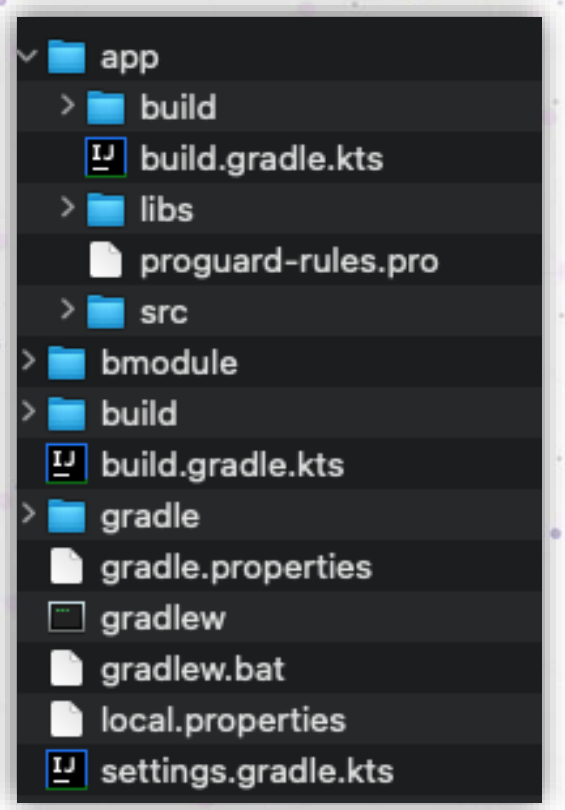
2. prednáška – 27. 2. 2024

Ciele prednášky

- Štruktúra projektu
- Resources
- Layouts
- Dobre vedieť

Štruktura projektu – schema

- app
- module
- build
- gradle



Štruktura projektu – gradle – project

‚build.gradle (Project)‘

- definuje konfiguráciu celého projektu a závislosti, ktoré sú spoločné pre všetky moduly(app

```
pluginManagement { this: PluginManagementSpec
    repositories { this: RepositoryHandler
        google()
        mavenCentral()
        gradlePluginPortal()
    }
}
dependencyResolutionManagement { this: DependencyResolutionManagement
    repositories { this: RepositoryHandler
        google()
        mavenCentral()
    }
}

rootProject.name = "P2"
include( ...projectPaths: ":app")
include( ...projectPaths: ":bmodule")
```

Štruktura projektu – gradle – module

Aplikácia vs. knižnica:

- dôležitou súčasťou konfigurácie je definovanie minimálnej (*minSdkVersion*), cieľovej (*targetSdkVersion*) a kompilačnej (*compileSdkVersion*) verzie Android SDK, ktoré ovplyvňujú kompatibilitu a funkcie dostupné pre aplikáciu

```
plugins { this: PluginDependenciesSpecScope
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
}
android { this: BaseAppModuleExtension
    namespace = "fi.paradox.p2"
    compileSdk = 34
    defaultConfig { this: ApplicationDefaultConfig
        applicationId = "fi.paradox.p2"
        minSdk = 27
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
        vectorDrawables { this: VectorDrawables
            useSupportLibrary = true
        }
    }
```

Štruktura projektu – gradle – module

Dependencies:

- build varianty a flavor
- kompilačné nastavenia
- prispôsobenie build procesu

```
flavorDimensions( ...dimensions: "version")
productFlavors { this: NamedDomainObjectContainer<ApplicationProductFlavor>
    create( name: "free") { this: ApplicationProductFlavor
        dimension = "version"
        applicationIdSuffix = ".free"
        versionNameSuffix = "-Free"
    }
    create( name: "paid") { this: ApplicationProductFlavor
        dimension = "version"
        applicationIdSuffix = ".paid"
        versionNameSuffix = "-Paid"
    }
}
```

```
dependencies { this: DependencyHandlerScope
    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")
    implementation("androidx.appcompat:appcompat:1.6.1")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

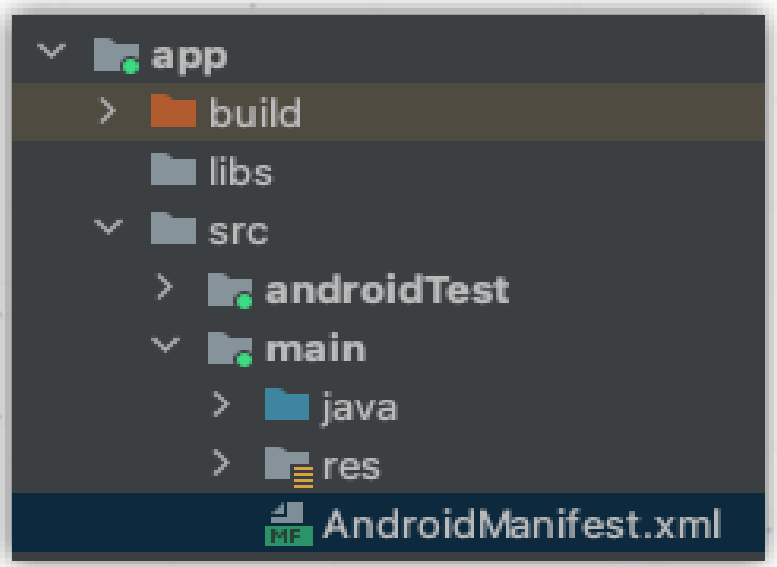
Štruktura projektu:

zhrnutie

```
plugins {...}
android { this: BaseAppModuleExtension
    namespace = "fi.paradox.p2"
    compileSdk = 34
    defaultConfig { this: ApplicationDefaultConfig
        applicationId = "fi.paradox.p2"
        minSdk = 27
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
        vectorDrawables {...}
    }
    buildTypes {...}
    compileOptions {...}
    kotlinOptions {...}
    buildFeatures {...}
    composeOptions {...}
    packaging {...}
}
dependencies {...}
```

Štruktura projektu – AndroidManifest.xml

- Manifestový súbor je srdcom každej Android aplikácie.
- Každá aplikácia musí mať tento súbor v koreňovom adresári.
- Definuje základné informácie o aplikácii pre Android systém pred jej spustením.



AndroidManifest.xml – základné komponenty

- **Package**: Unikátny identifikátor aplikácie, ktorý sa používa na jej identifikáciu v Google Play Store a na zariadení.
- **Application**: Koreňový element, ktorý definuje globálne vlastnosti aplikácie. Môžete tu špecifikovať ikonu aplikácie, tému a ďalšie komponenty.
- **Activities**: Definujú 'okná' cez ktoré používatelia interagujú s aplikáciou. Každá aktivita by mala byť deklarovaná v tomto súbore.
- **Services**: Pozadie komponenty, ktoré môžu vykonávať dlhotrvajúce operácie alebo pracovať, keď nie je používateľské rozhranie aktívne.
- **Broadcast Receivers**: Umožňujú aplikácii prijímať a reagovať na rôzne globálne oznámenia alebo udalosti.
- **Content Providers**: Spravujú prístup k štruktúrovaným dátam. Sú užitočné najmä pri zdieľaní dát medzi rôznymi aplikáciami.
- **Povolenia**: Manifest tiež definuje, aké systémové zdroje alebo informácie potrebuje aplikácia. Tieto povolenia musia byť výslovne požiadané a udelené používateľom.
- **Intent Filters**: Pomocou intent filtrov môžete deklarovať, pre aké akcie alebo dáta je vaša aplikácia schopná reagovať. Napríklad, môžete filtrovať určité URL, aby sa otvárali vo vašej aplikácii.

AndroidManifest.xml – základné komponenty

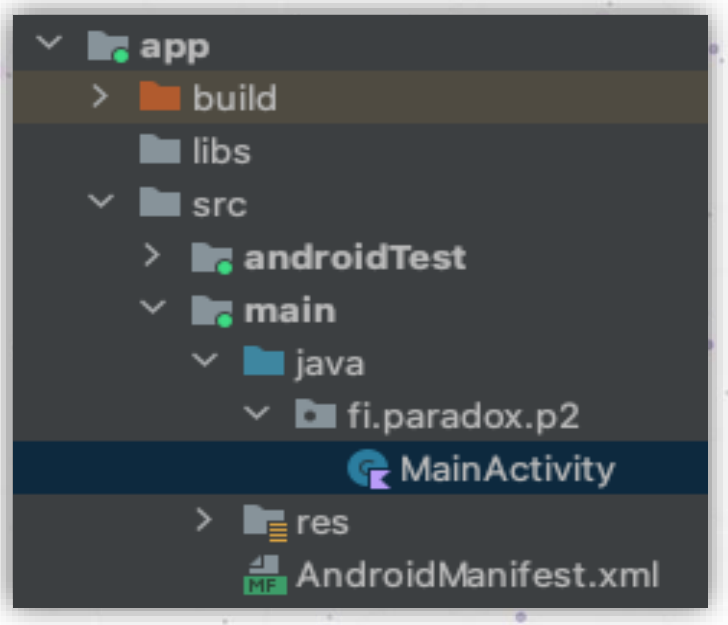
```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

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

Štruktura projektu – JAVA / Kotlin

- Priečinky *src/main/java* alebo *src/main/kotlin* sú určené pre zdrojové kódy vašej aplikácie.
- Tu umiestňujete všetky vaše Java alebo Kotlin súbory, ktoré obsahujú logiku vašich aktivít, služieb, broadcast prijímačov a iných komponentov aplikácie.



JAVA / Kotlin – organizácia kódu

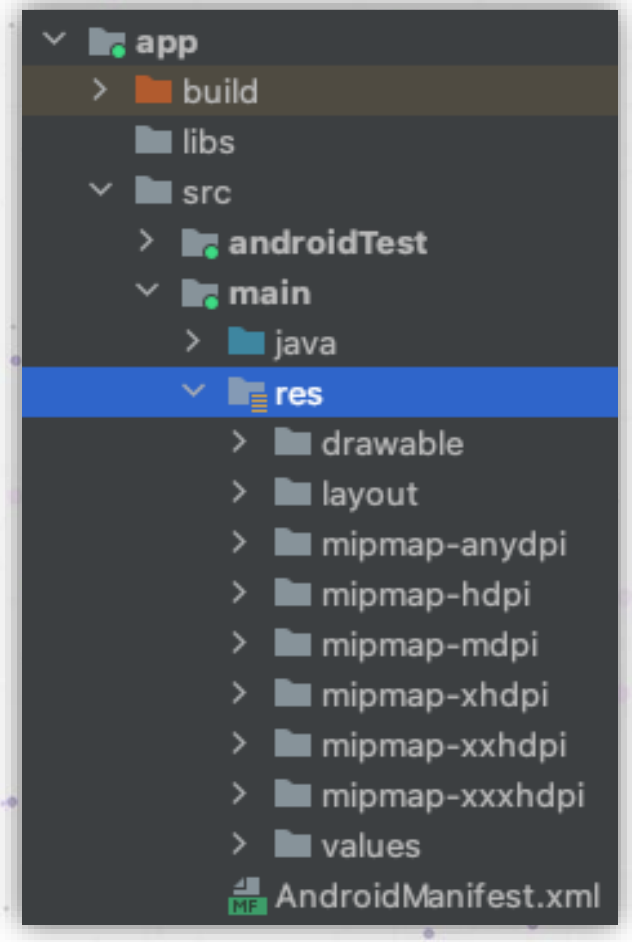
- Je dôležité udržiavať čistú a logickú štruktúru balíkov v týchto priečinkoch. To zahŕňa skupinovanie súvisiacich tried do balíkov podľa ich funkcionality alebo modulu, na ktorom pracujú, čo uľahčuje navigáciu a správu kódu.

- Okrem hlavných zdrojových priečinkov, *src/test/java* alebo *src/test/kotlin* sú určené pre jednotkové testy, ktoré testujú komponenty vašej aplikácie izolovane od Android frameworku.

- *src/androidTest/java* alebo *src/androidTest/kotlin* sú pre instrumentované testy, ktoré bežia na Android zariadeniach alebo emulátoroch.

Štruktura projektu – resources

- Priečinok *res* v Android projekte je určený na ukladanie 'zdrojov‘ – to sú súbory, ktoré definujú obsah a vzhľad vašej aplikácie, ako sú layouty, reťazce, obrázky a štýly
- assets Môže obsahovať surové súbory, ako sú fonty alebo dáta, ktoré aplikácia môže používať.



Štruktura projektu – resources

```xml
<resources>
    <string name="app_name">Moja Aplikácia</string>
    <string name="hello_world">Ahoj svet!</string>
    <string name="menu_settings">Nastavenia</string>
</resources>
```

```xml
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
    <color name="textColor">#FFFFFF</color>
</resources>
```

```xml
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="TextStyle" parent="TextAppearance.AppCompat">
        <item name="android:textSize">18sp</item>
        <item name="android:textColor">@color/textColor</item>
    </style>

    <style name="ButtonStyle" parent="Widget.AppCompat.Button">
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">@color/colorPrimary</item>
        <item name="android:background">@drawable/button_background</item>
    </style>
</resources>
```

Štruktura projektu – resources – príklady

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

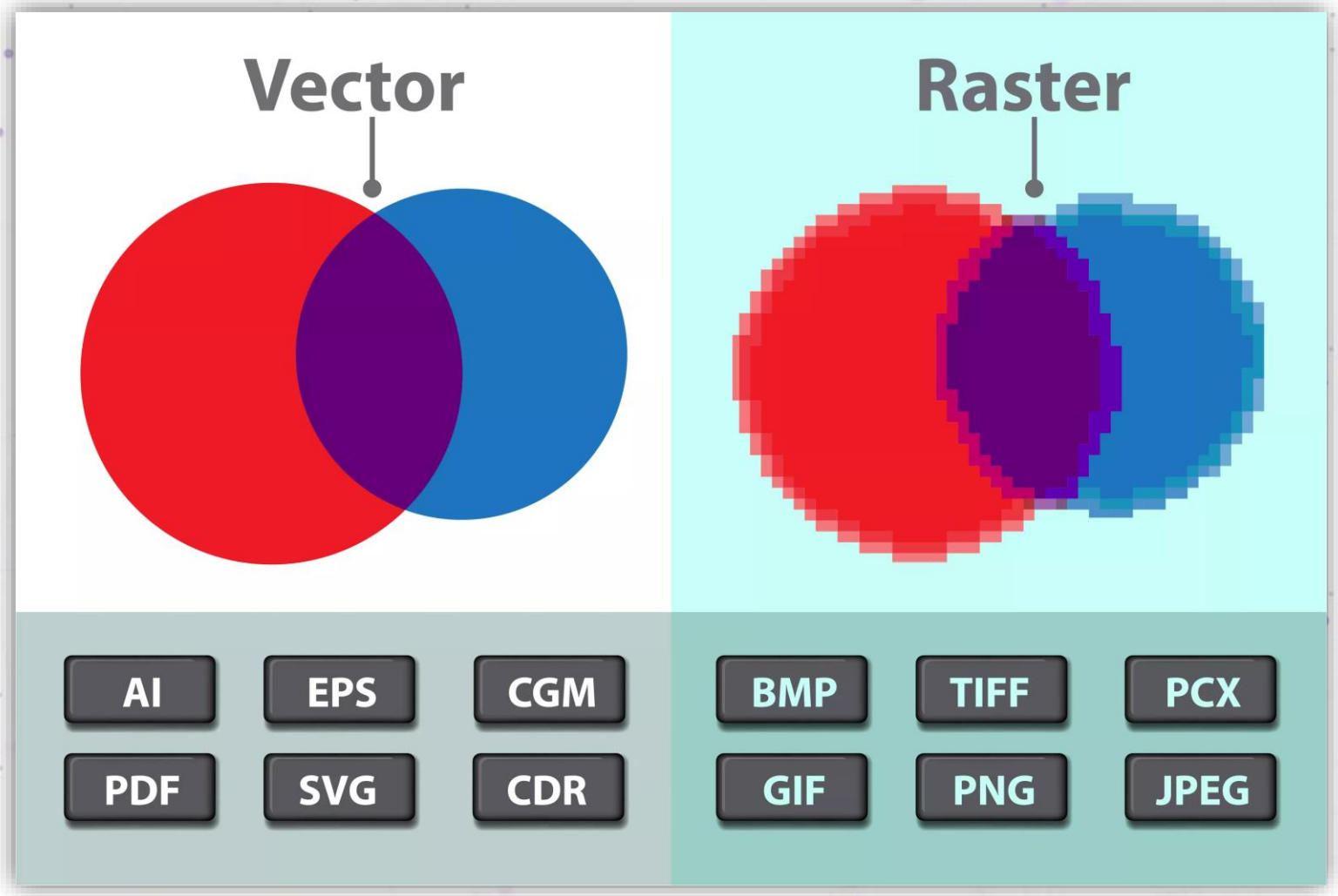
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"
    android:textColor="@color/colorPrimary" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/menu_settings"
    style="@style/ButtonStyle" />
```

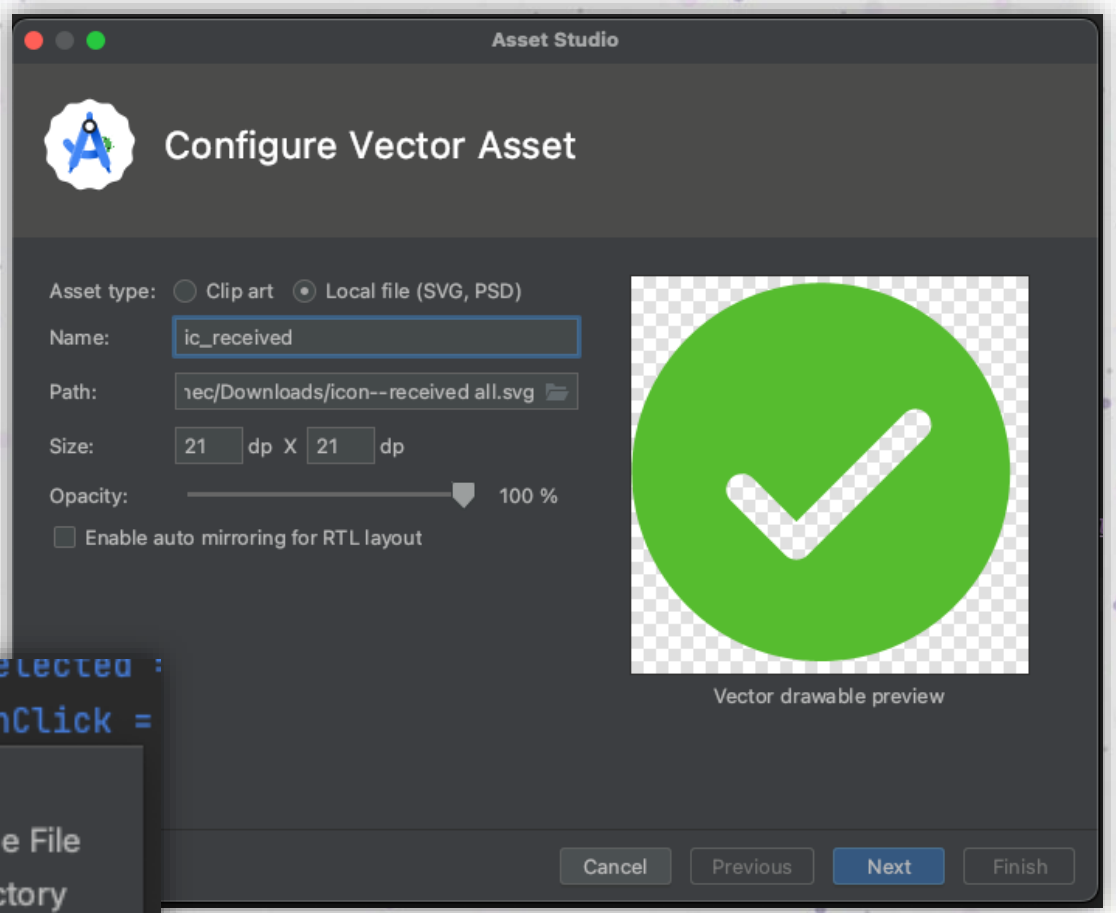
```
val helloWorldString = getString(R.string.hello_world)
```

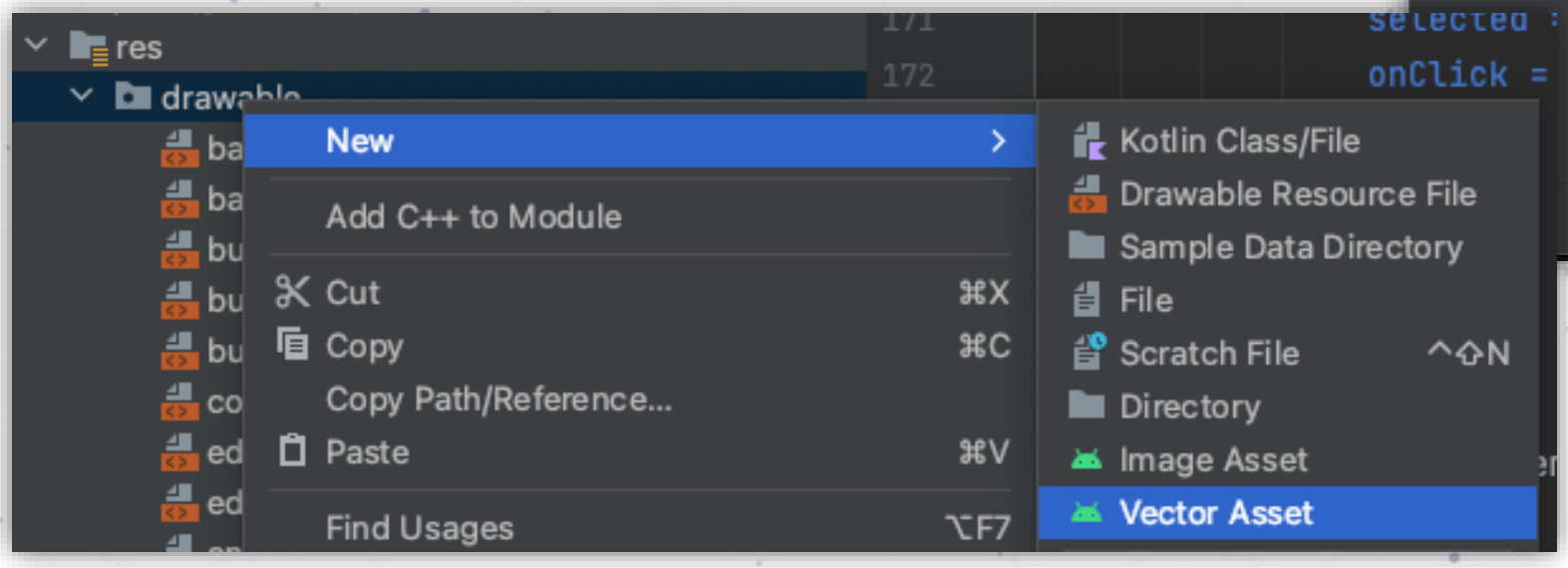
```
val primaryColor = ContextCompat.getColor(context, R.color.colorPrimary)
```

Resources – obrázky



Resources – obrázky





Štruktura projektu – testy

- jednotkové testy (*src/test/*)
- integračné testy
- UI Testy / End-to-End testy (*src/androidTest/*)

Layout

- **Základné komponenty:**

View: Základná stavebná jednotka pre UI komponenty ako sú textové polia, tlačidlá a obrázky.

ViewGroup: Kontajner, ktorý obsahuje a usporadúva viacero Views alebo ViewGroupov, napríklad LinearLayout, RelativeLayout, a ConstraintLayout.

- **Typy Layotov:**

LinearLayout: Usporiadáva komponenty v jednom smere, vertikálne alebo horizontálne.

RelativeLayout: Umožňuje umiestniť komponenty vzhľadom na vzťahy s ostatnými komponentmi v layoute.

ConstraintLayout: Poskytuje flexibilné a výkonné usporiadanie komponentov s použitím obmedzení pre definovanie ich pozície a veľkosti.

Layout – LinearLayout

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/click_me" />
</LinearLayout>
```

Layout
—
ConstraintLayout

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toTopOf="@id/button"
        app:layout_constraintVertical_chainStyle="packed" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/click_me"
        app:layout_constraintTop_toBottomOf="@id/textView"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />

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

Layout

- **Návrhové princípy a najlepšie praktiky:**

Opätovné použitie a modularita: Používajte <include> a <merge> tagy pre opätovné použitie častí layoutu a zvýšenie modularity.

Prispôsobiteľnosť: Využívajte kvalifikátory zdrojov (napr. -land, -port, -sw600dp) pre prispôsobenie layoutov rôznym veľkostiam a orientáciám obrazovky.

Pristupnosť: Zabezpečte, aby vaše layouty boli prístupné, s použitím správnych atribútov, ako sú contentDescription pre obrázky.

Layout – findViewByld

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val textView = findViewById<TextView>(R.id.textView)
        val button = findViewById<Button>(R.id.button)

        button.setOnClickListener {
            textView.text = "Kliknuté!"
        }
    }
}
```

Layout – Databinding

```
buildFeatures {
    dataBinding true
}
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@{dataModel.text}" />
```

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private val dataModel = DataModel()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = DataBindingUtil.setContentView(this, R.layout.activity_main)

        // Nastavenie dátového modelu pre binding
        binding.dataModel = dataModel

        // Nastavenie OnClickListenera pre tlačidlo
        binding.buttonShowText.setOnClickListener {
            // Aktualizácia dátového modelu podľa vstupu z EditText
            dataModel.text = binding.editTextInput.text.toString()
            // Explicitné oznámenie, že sa dáta zmenili, pre aktualizáciu UI
            binding.invalidateAll()
        }
    }
}
```

Layout – Compose

```
android {
    ...
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion compose_version
    }
}

dependencies {
    implementation "androidx.compose.ui:ui:$compose_version"
    implementation "androidx.compose.material:material:$compose_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_version"
    ...
}
```

Layout – Compose

```
@Composable
fun MyApp() {
    // Stav pre ukladanie viditeľnosti textu
    val showText = remember { mutableStateOf(false) }

    Column {
        Button(onClick = { showText.value = true }) {
            Text("Klikni na mňa")
        }
        if (showText.value) {
            Text("Ahoj svet!")
        }
    }
}
```

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

Informácie k pohovoru

- Ako znížiť velkosť *.apk*?

Optimalizacia resources

Odstránenie nepoužítých resources

Odstránenie nepotrebných zavislostí

Proguard / R8

Dynamic feature

App Bundle

Informácie k pohovoru

- Ako môžete znížiť čas potrebný na zostavenie Android aplikácie?

Opustiť Windows

org.gradle.configureondemand=true

org.gradle.daemon=true

org.gradle.parallel=true

org.gradle.jvmargs=-Xmx3072m -XX:MaxPermSize=512m

Dynamické závislosti

```
implementation 'com.android.support:appcompat-v7:27.0.2
implementation 'com.android.support:appcompat-v7:27.0.+
```

Zdroje

- **Kotlin**

  Oficiálna dokumentácia Kotlin: https://kotlinlang.org/docs/reference/
- **Android Štruktúra Projektu a Gradle**

  Štruktúra Android Projektu: https://developer.android.com/jetpack/guide

  Gradle pre Android: https://developer.android.com/studio/build
- **Android Layouty a UI**

  Práca s Layoutmi: https://developer.android.com/guide/topics/ui/declaring-layout

  View Binding: https://developer.android.com/topic/libraries/view-binding

  Data Binding: https://developer.android.com/topic/libraries/data-binding

  Jetpack Compose: https://developer.android.com/jetpack/compose

- Testovanie v Android: https://developer.android.com/training/testing/

ĎAKUJEM ZA POZORNOSŤ