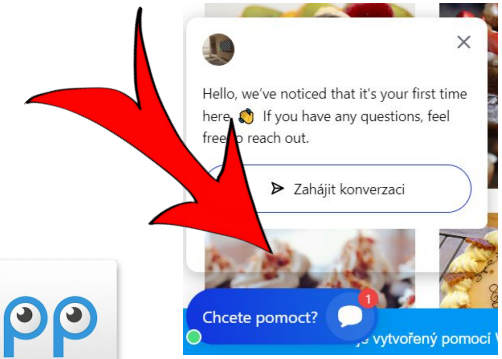
Úvod do Androidu

a možná i trochu víc

Jan Maděra

👋 Ahoj

- Android developer ve společnosti **Smartsupp**
- Vývoj pro Android přes 8 let
- LinkedIn



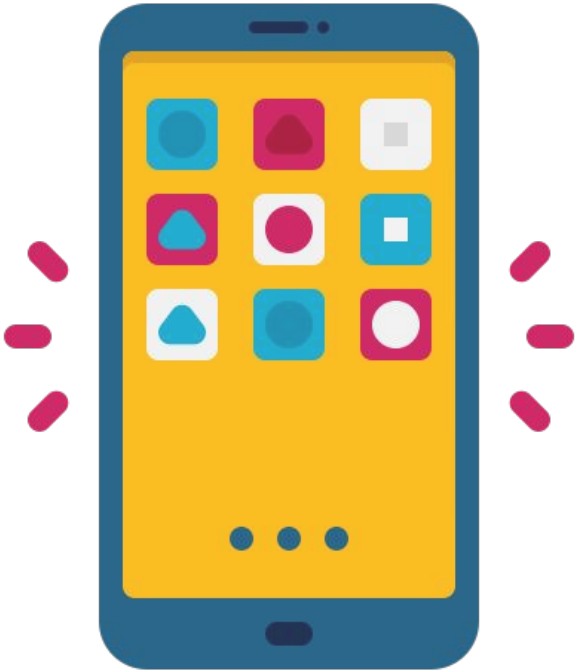
🎯 Osnova

- Nebojte se na cokoliv zeptat 😉
- Kde a jak se Android vyskytuje
- Jak vypadá vývoj pro Android
- Android vývoj v praxi
  - Architektury
  - Knihovny
  - Testing
- Cross-platform
  - Kotlin Multiplatform Mobile
- Koho sledovat a kam dál?

Android okolo nás

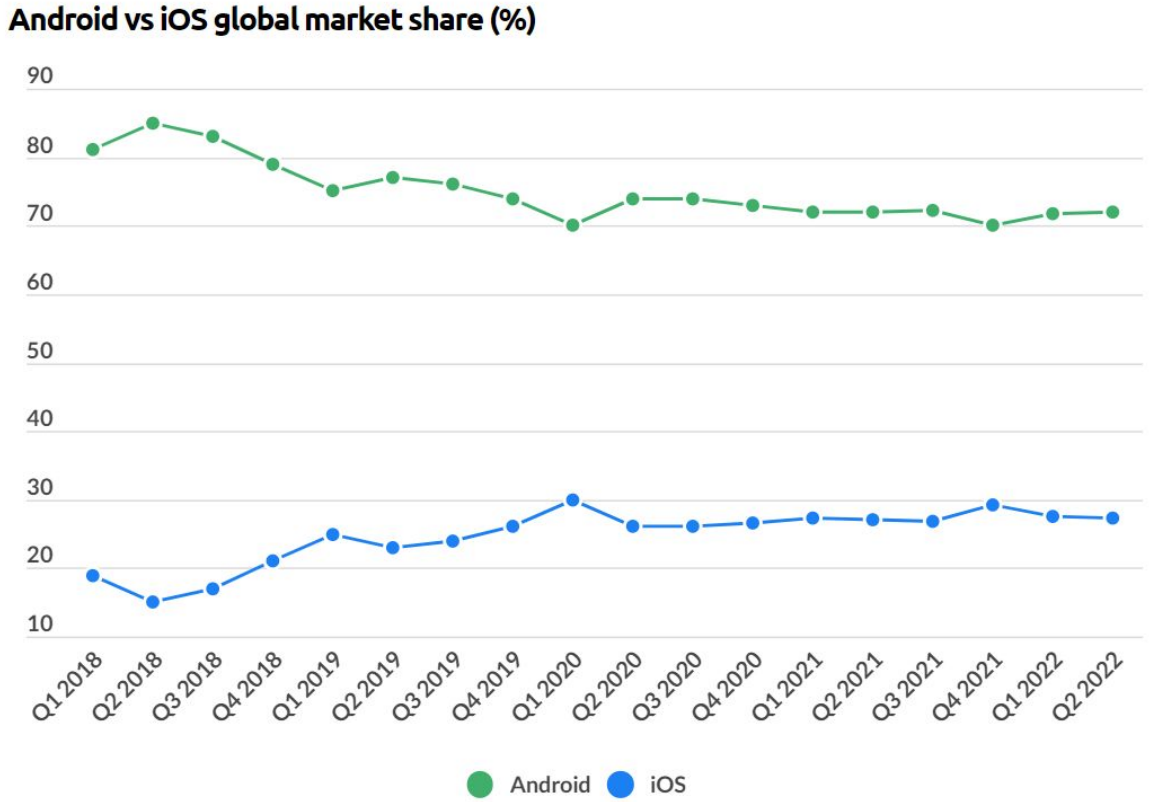
Kde všude dnes najdeme Android?

- Vlastně v podstatě **všude**
  - Telefony 📱
  - Tablety 💻
  - AndroidAuto 🚗
  - Brýle 👓



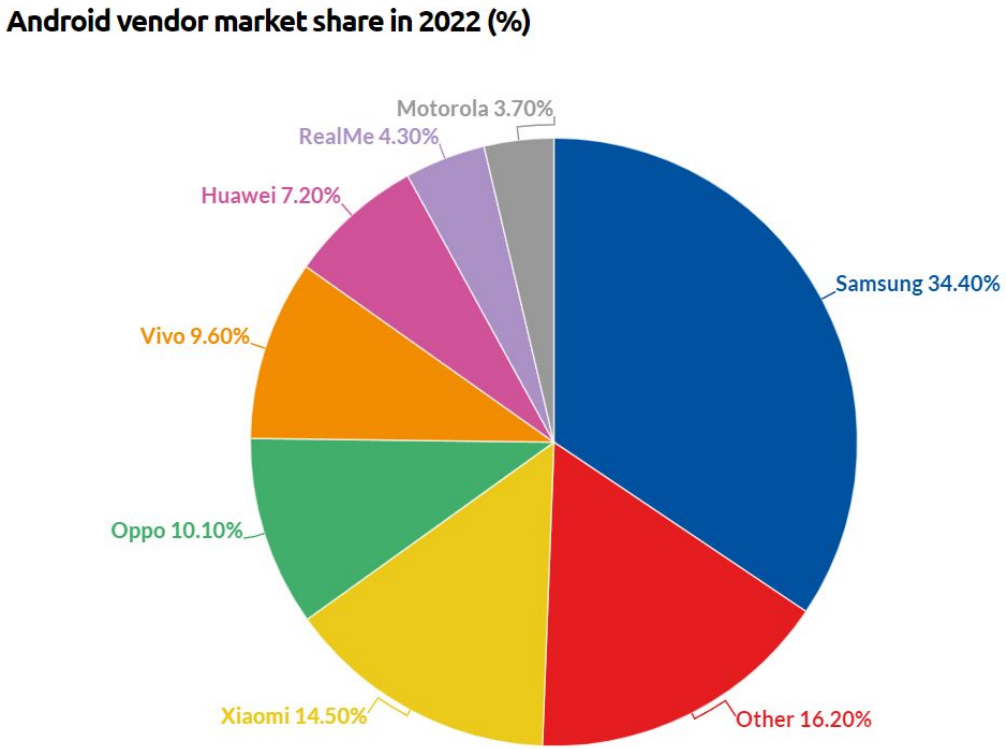
📺 Android najdete i tam kde byste ho možná nečekali

Android v číslech vs. iOS



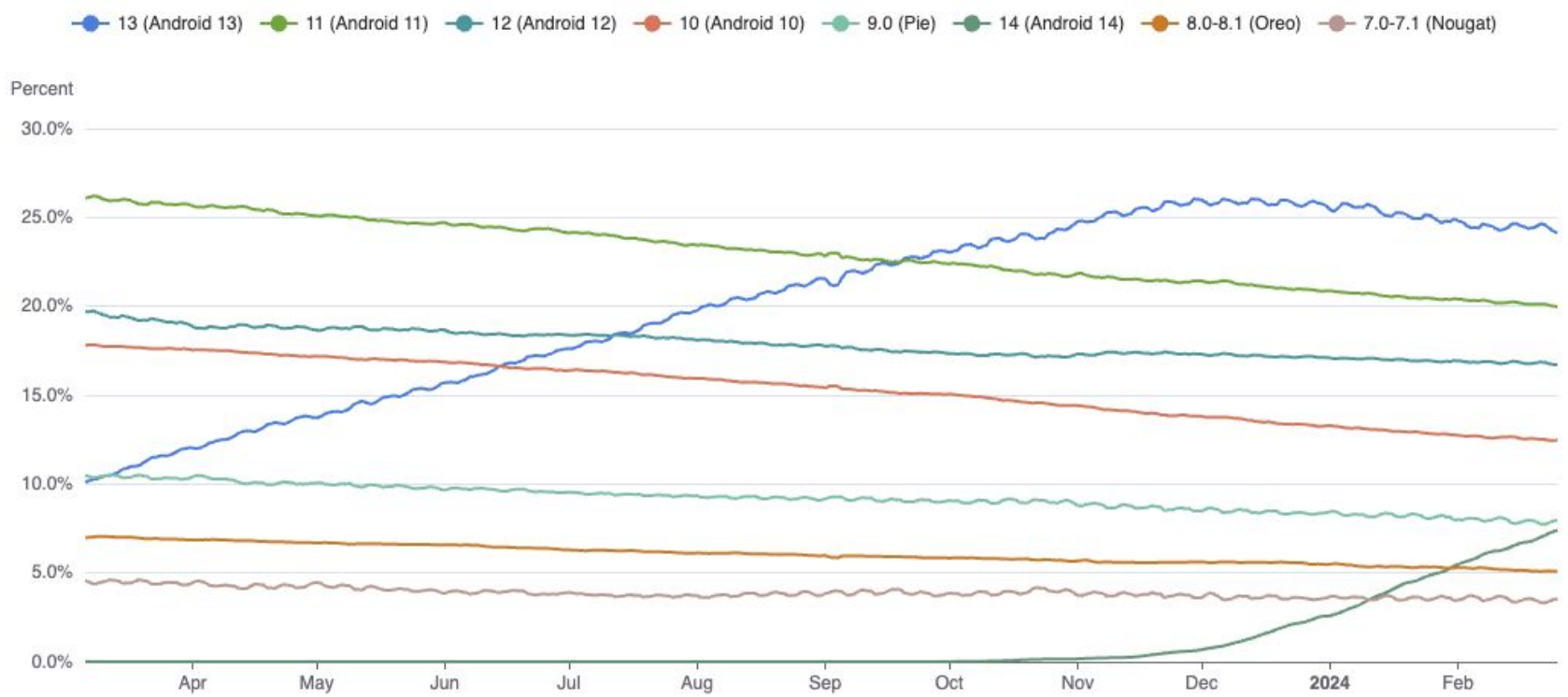
Zdroj

Android v číslech dle výrobce



Zdroj

Android v číslech dle verze



Zdroj

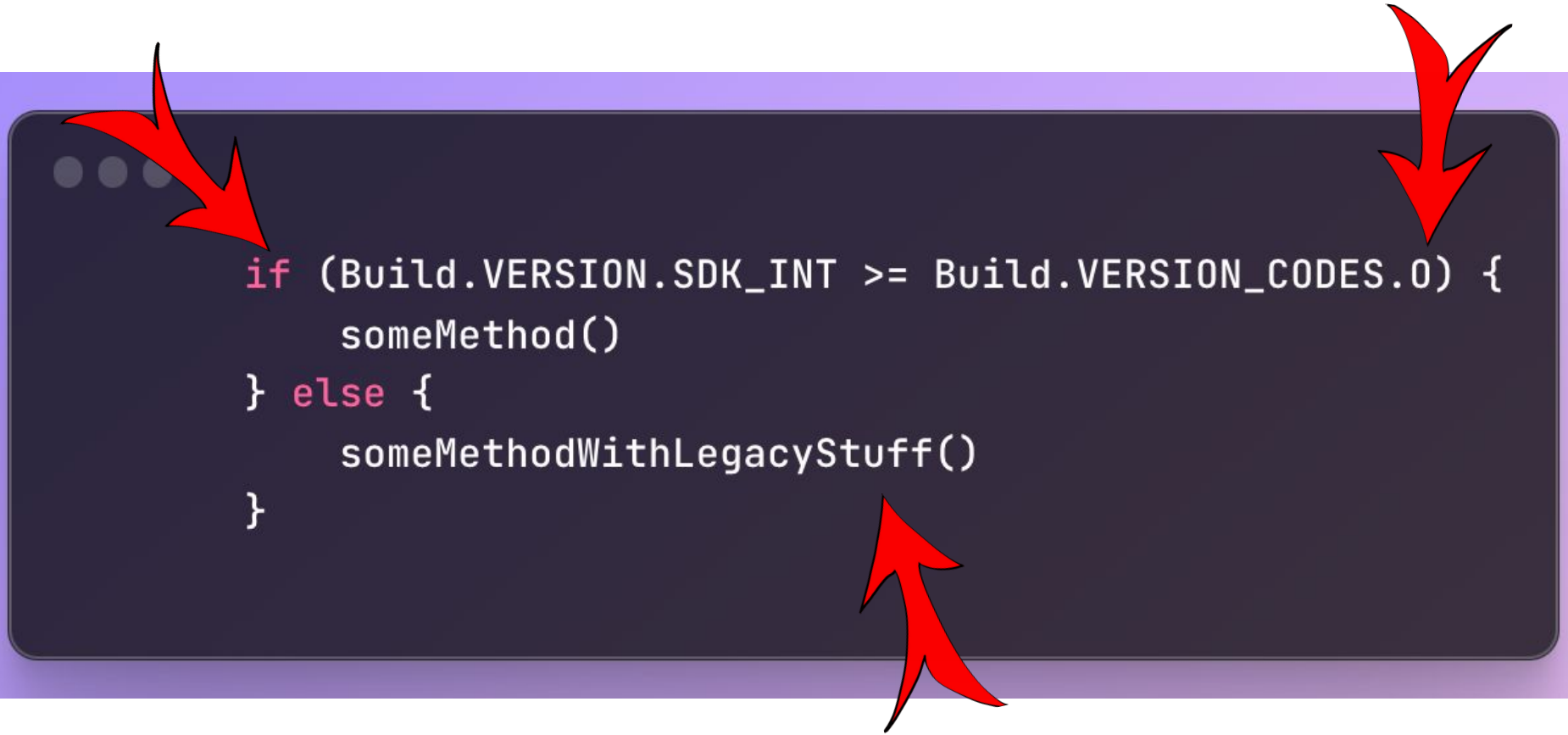
🤔 Vývoj s tolika verzemi je složitý!

- Ano i NE
- Google vývojáři se velmi snaží o zpětnou kompatibilitu
- Problém nastává pokud nová verze přinese nějakou **novou funkci** nebo naopak přinese nějaké **nové omezení** (zvýšení bezpečnosti)
- 🍎 *má podobný problém (né tak častý), ale Apple nehledí na zpětnou kompatibilitu, ale tlačí uživatele do updatu verze (tzn. může znamenat koupi nového modelu)*
  - *SwiftUI a UIKit*

❓Jak se takový problém řeší v praxi❓

⛔**Problém** - Zpětná kompatibilita

✅Řešení - Zpětná kompatibilita

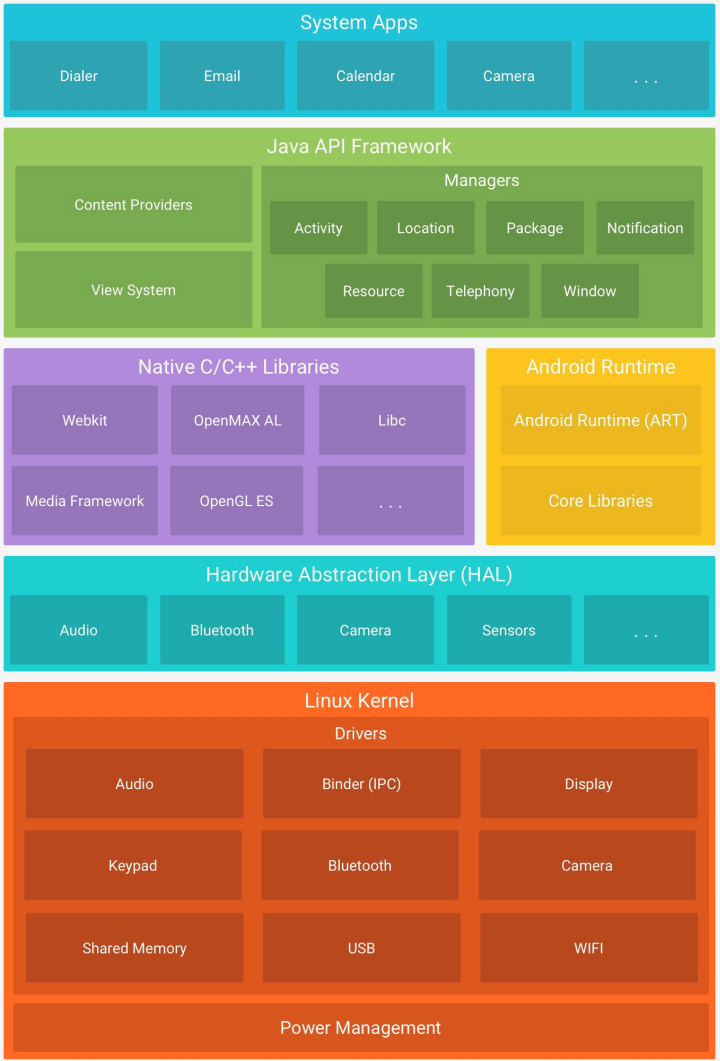


```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    someMethod()
} else {
    someMethodWithLegacyStuff()
}
```

👩‍💻Vývoj pro Android👨‍💻

🍳 Android pod pokličkou

- Založen na Linuxu
- Open Source Projekt (AOSP)
  - **source.android.com/**
  - neobsahuje kritické komponenty jako **Play Store**, **Google Play Service** a další Google aplikace
  - Pro vytvoření funkčního OS z AOSP jsou potřeba úpravy



Jak vypadá vývoj na Android?

- Android studio (JetBrains)
  - Plně zdarma
  - Časté aktualizace (Beta, Canary)
- Android SDK
  - Podmnožina JDK **+ Android** “classy” **+** knihovny (Apache, ...)
- **Kotlin**
  - Dříve Java (do května 2017), ale **starší code-base** se píší stále v Javě, ale firmy se snaží o refactoring
    - 6
    - 7 od 4.4 - syntaxe, ne plné API
    - 8 od 7.0 - syntaxe, ne plné API
- Gradle build system
  - Groovy, **Kotlin DSL**

Kotlin *(https://kotlinlang.org/)*

- *Moderní, stručný a bezpečný*
- **JetBrains**
- Plně kompatibilní s Javou
- Cross-Platform
  - **Android**
  - **Kotlin-Multiplatform** - **Android** + iOS
  - ServerSide
    - Spring
    - Ktor

VS.

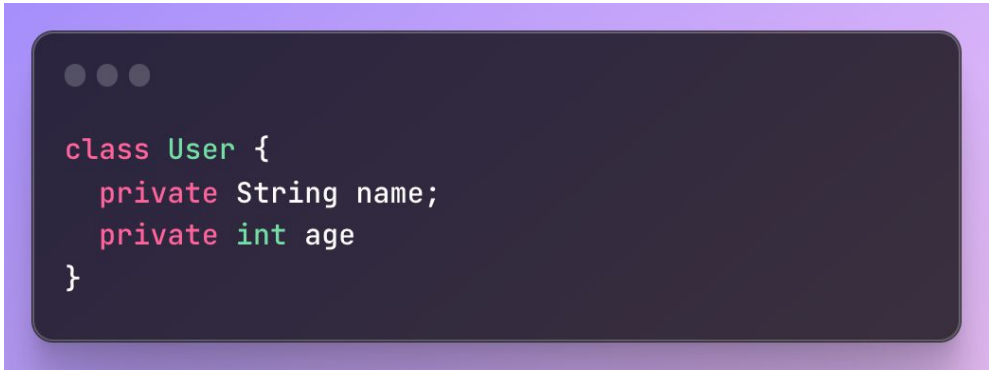
Java vs. Kotlin - Metody, Signatury, zápis proměnných

```java
public String foo(int input) {
    String testText = "";
    boolean testBoolean = false;
    int testInt = 5;
    double testDouble = 5.5;

    return "5";
}
```

```kotlin
fun foo(input: Int): String {
    val testText1 = ""
    val testText2: String = ""
    val testTextBoolean = ""
    val testTextInt = ""
    val testTextDouble = ""

    return "5"
}
```

Java vs. Kotlin - Třídy

- Java



```
class User {
  private String name;
  private int age
}
```

- **Kotlin**
  - **data class** - automaticky vygenerované **hashCode** a **copy**
  - **copy** - metoda vrací kopírovaný objekt v těle lze měnit parametry (vhodné pro immutable)

```
data class User(
  val name: String,
  val age: Int
)
```

❓Jak se píše UI pro Android❓

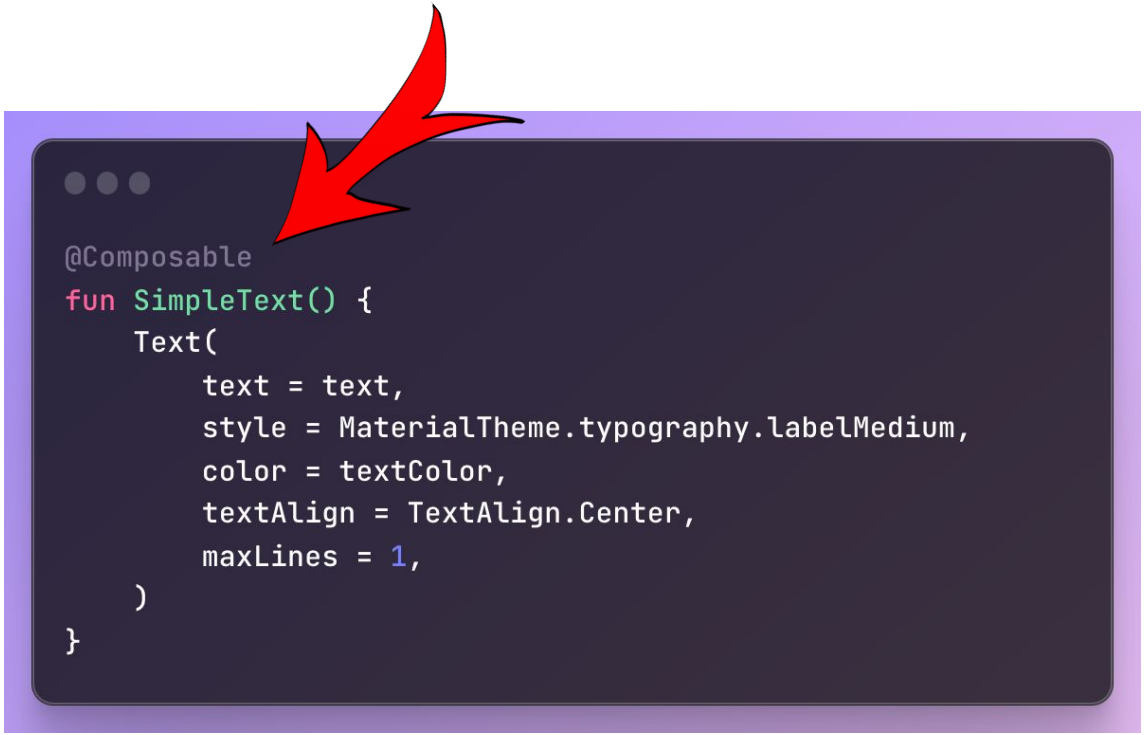
Staré dobré .xml

- Způsob používaný od prvopočátku
- Dnes se od něj pomalu odstupuje

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/test_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="🎸🥁"
        android:textSize="64dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

JetPack Compose

- Dnes už doporučený způsob jak psát moderní UI



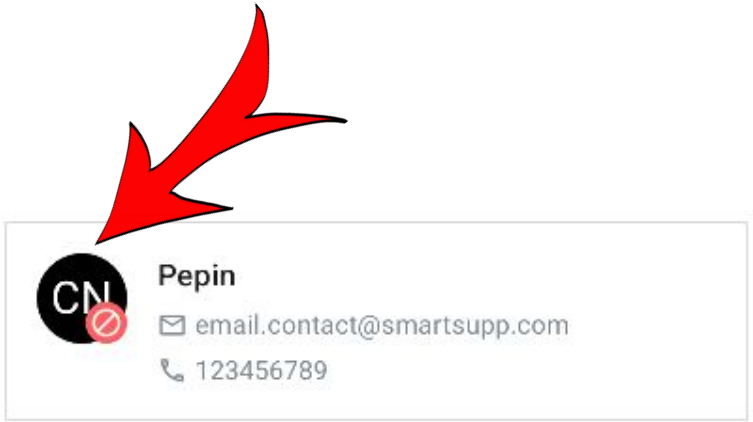
```
@Composable
fun SimpleText() {
    Text(
        text = text,
        style = MaterialTheme.typography.labelMedium,
        color = textColor,
        textAlign = TextAlign.Center,
        maxLines = 1,
    )
}
```

ChipSmallSin...

Text

JetPack Compose

```
@Composable
private fun ContactAvatar(contact: ContactHubContact) {
    ConstraintLayout {
        val (avatar, icon) = createRefs()
        SmartsuppAvatar(
            name = contact.name,
            textColor = contact.avatarInfo?.color?.text,
            backgroundColor = contact.avatarInfo?.color?.bg,
            modifier = Modifier.constrainAs(avatar) {
            },
        )

        if (contact.isBlocked) {
            Icon(
                painter = painterResource(id = R.drawable.ic_block),
                contentDescription = "",
                tint = MaterialTheme.colors.whiteConstant,
                modifier = Modifier
                    .size(22.dp)
                    .background(color = MaterialTheme.colors.red500, shape = CircleShape)
                    .padding(2.dp)
                    .constrainAs(icon) {
                        bottom.linkTo(avatar.bottom)
                        end.linkTo(avatar.end)
                    },
            )
        }
    }
}
```



Flutter vs. Android vs. iOS

```dart
class MyCard extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Row(children: [
      CircleAvatar(
        backgroundImage: AssetImage('assets/sundar_pichai.jpeg'),
        radius: 30,
      ), // CircleAvatar
      SizedBox(
        width: 10,
      ), // SizedBox
      Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            "Sundar Pichai",
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ), // Text
          Text(
            "Chief Executive Officer of Alphabet",
            style: TextStyle(
              fontSize: 16,
            ), // TextStyle
          ), // Text
        ],
      ) // Column
    ]); // Row
  }
}
```

Flutter

```kotlin
@Composable
fun MyCard() {
    Row(modifier = Modifier.fillMaxWidth().height(60.dp)) { this: RowScope
        Image(painter = painterResource(
            id = R.mipmap.sundar_pichai),
            contentDescription = null,
            modifier = Modifier
                .size(60.dp)
                .clip(CircleShape)
                .border(2.dp, Color.Gray, CircleShape)
        )
        Column(Modifier.fillMaxHeight().padding(horizontal = 10.dp),
            verticalArrangement = Arrangement.Center
        ) { this: ColumnScope
            Text(
                text = "Sundar Pichai"
            )
            Text(
                text = "Chief Executive Officer of Alphabet"
            )
        }
    }
}
```

Jetpack Compose

```swift
import SwiftUI

struct ContentView: View {
    var body: some View {
        HStack (
            spacing: 10
        ){
            Image("sundar_pichai_img").resizable().frame(width: 60, height: 60
            ).clipShape(Circle()).overlay(Circle().stroke(Color.gray, lineWidth: 2
            ))
            VStack(
                alignment: .leading,
                spacing:0
            ){
                Text("Sundar Pichai").font(.title2)
                Text("Chief Executive Officer of
                    Alphabet").font(.body).foregroundColor(.gray)

            }
        }
    }
}
```

Swift UI

Vydávání aplikace

Google Play/Play Console/Vydání aplikace

- Jednorázový poplatek **$25** 💰
- Co získáte?
  - Analytics v rámci vaší aplikace
    - rating, instalace, aktivní uživatele, atd…
  - Delivery přes Google Console až do Google play
    - testovací kanály, inkrementální delivery
  - Správa placených produktů a subscriptions
    - *ale taky platíte 15% z částky Googlu*
  - API
    - např. pro Continuous delivery

Google Play Console

Store listing performance

Store listing acquisitions

2.63K +3.95%

Explore →

Your KPIs

New users acquired

New users acquired

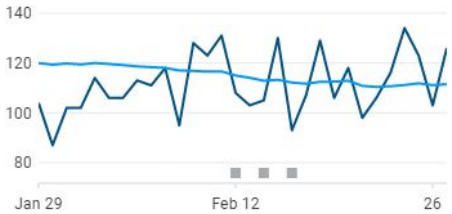
2.2K +5.61%

Explore →

User loss

User loss

3.35K -7.24%

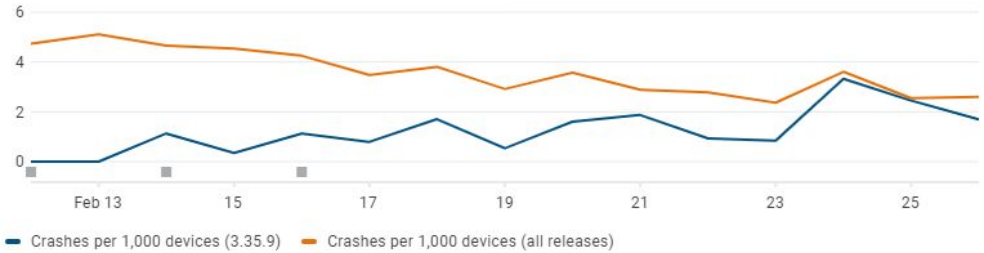


Explore →

Latest release performance

Crashes per 1,000 devices

1.412 -60.68% vs all releases



Android(Native)- Výhody a Nevýhody

👍 Výhody

- Snadný **vstup do vývoje**
- Skvělá **komunita** a dokumentace
- Mladé a stále se vyvíjející odvětví
- **Kotlin**
- Oproti iOS 🍎 větší možnost **využití v “průmyslu”**
  - Kasy, Scannery do skladu etc.

👎 Nevýhody

- **Výrobci si přizpůsobují/optimalizují své operační systémy**, které potom vedou k chybám na produkci
  - Typický příklad notifikace
  - Práce telefonu na pozadí
- Mnoho verzí Androidu občas komplikuje code-base

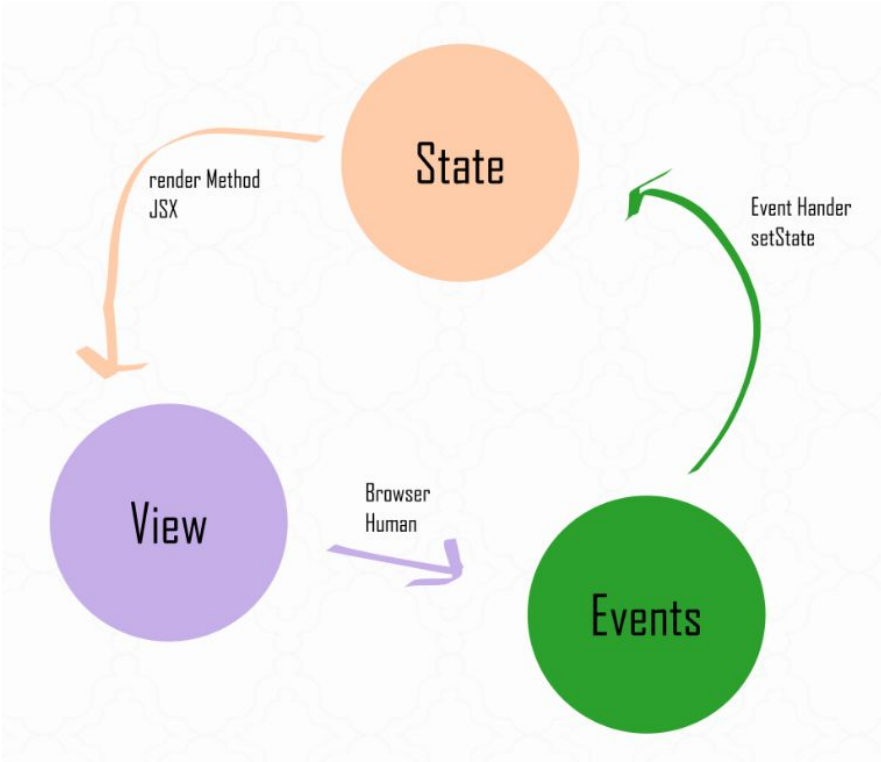
Vývoj pro Android dnes

➡️ Směr vývoje pro Android

- Na co můžete narazit, když budete hledat v práci jako **Android vývojář/ka?**
  - Filozofie UI
  - Architektura aplikace
  - Testing
- Kterým **směrem** se pohybuje Android vývoj?
- **Má smysl se ještě učit Nativní Android vývoj**, když existují multiplatformní řešení
  - AirBnb blog o vývoj v ReactNative

📕 Filozofie UI

- State rendering
- Programově deklarovatelné UI
- Podobný přístup lze vidět napříč všemi FrontEndovými frameworky
  - Flutter - Všechno je Widget
  - iOS - SwiftUI
  - React - Redux

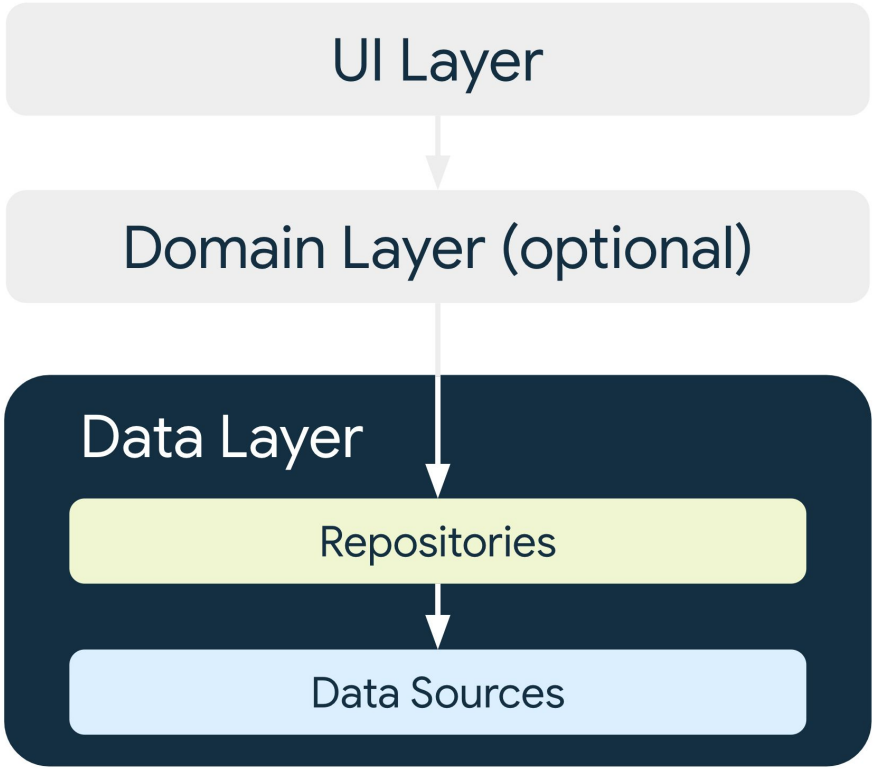


📏 Architektura aplikace - UI

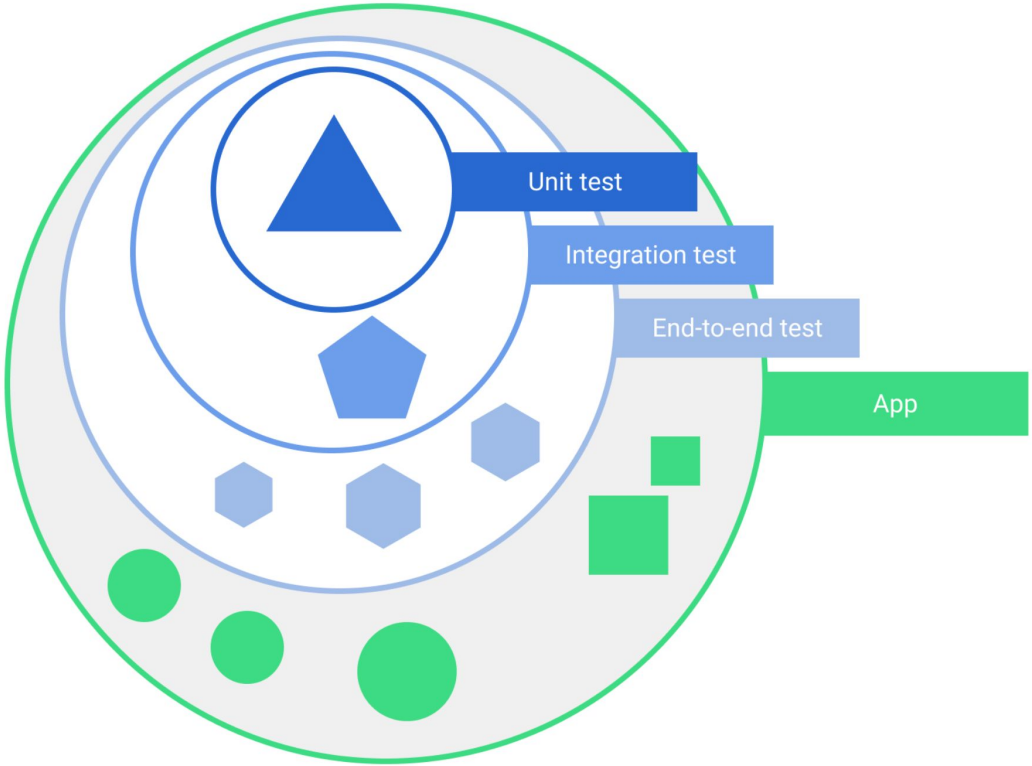
- V dnešní době se používají převážně dvě architektury
  - **MVVM**(dominantní) - Model-View-ViewModel
  - **MVI** - Model-View-Intent
- Využívají ViewModel pro práci se Statem, který je následně renderován
- Pracují s Observable patternem

📏 Architektura aplikace - Layers

https://developer.android.com/topic/architecture



🧪 Testing - Scope



- **Unit test**
  - Malé části aplikace
- **Medium**
  - Více části aplikace
- **End-to-End**
  - Flow obrazovek
- **App**
  - QA testing
  - Tester prochází ručně aplikaci

🧪 Testing - Unit Testing

```
@Test
fun isKeyValid_WhenPassedKeyAndRemoteKeyAreEqual_Success() {
    accountRepository.smartsuppKeyRemote = "validKey"

    testScope.runTest {
        sut.init("validKey", "")

        assert(accountRepository.isAccountValid)
    }
}
```

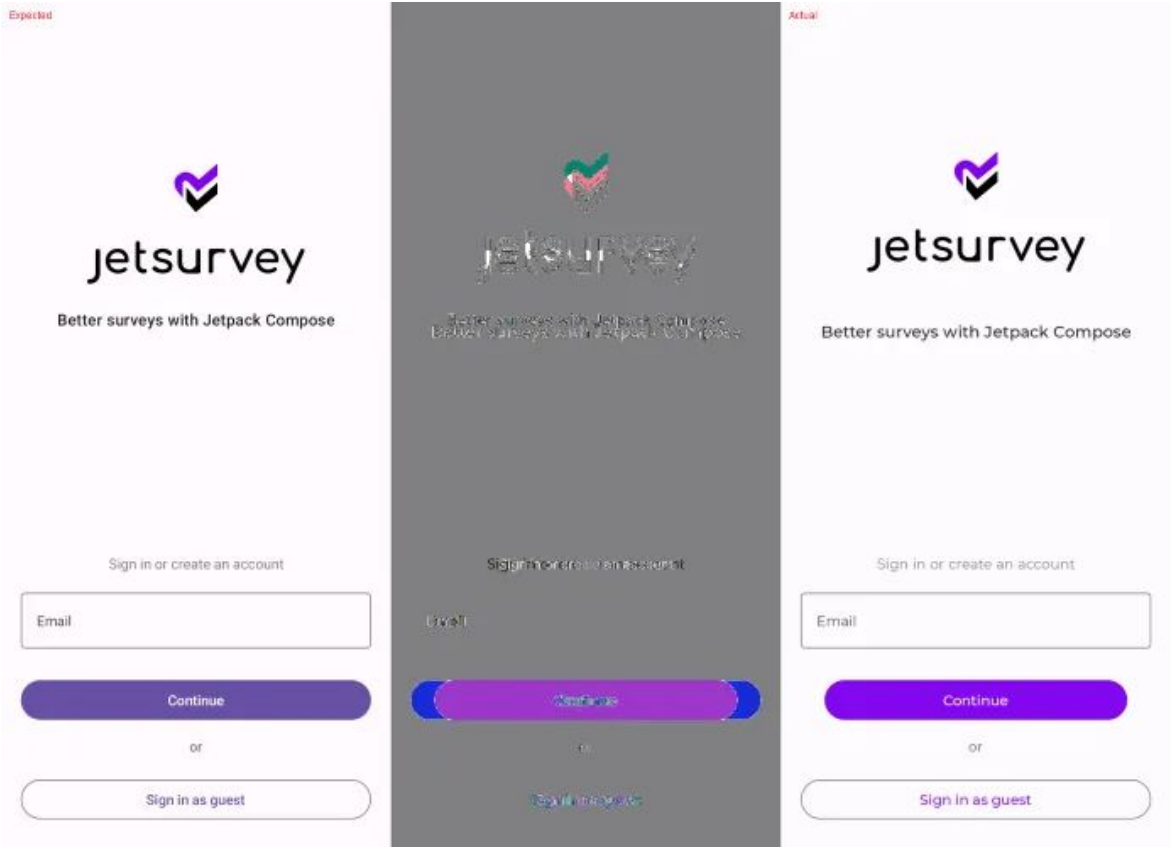
🧪 Testing - Integration Testing

```
@Test
fun areAnalytics_Inserted_InDatabase() {
    val analyticsCount = 5
    insertAnalytics(analyticsCount)

    analyticsDao.selectAllAnalyticsOrderDesc()
        .test()
        .assertValue { it.size == analyticsCount }
        .dispose()
}
```

🧪 Testing - Screenshot Testing

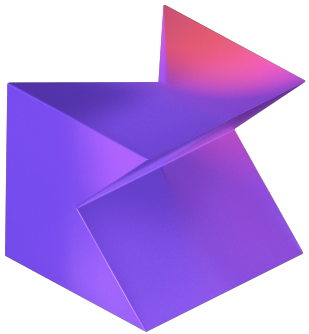
- Jedná se o proces, kde se porovnávají 2 screenshoty
- (obrazovky nebo části obrazovky)
  - Kde první snímek slouží jako zdroj pravdy
  - A druhý snímek se s ním porovává
- Umožňuje za “nízké náklady” testovat UI aplikace
  - Testy jsou velmi jednoduché na napsání
  - Někdy není ani nutné psát
- Knihovny
  - **Paparazzi** - https://github.com/cashapp/paparazzi
  - Roborazzi - https://github.com/takahirom/roborazzi

🧪 Testing - Screenshot Testing(Example)



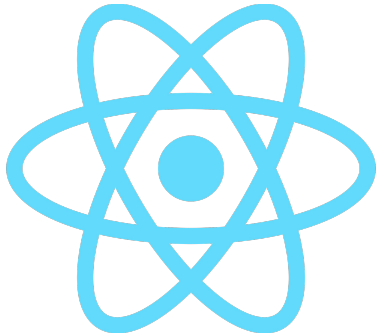
Zdroj

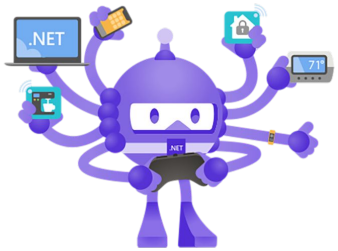
Kotlin Mutliplatform Mobile

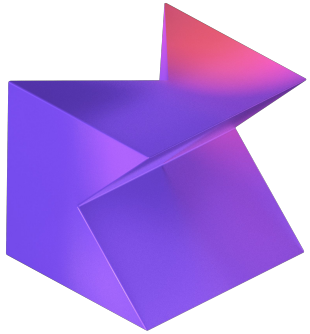


Cross-platform vývoj

- **Kotlin-Multiplatform**
  - **Kotlin**
- MAUI (Xamarin)
  - C#
- Flutter
  - Dart
- React Native
  - JavaScript
- Cordova
  - JavaScript



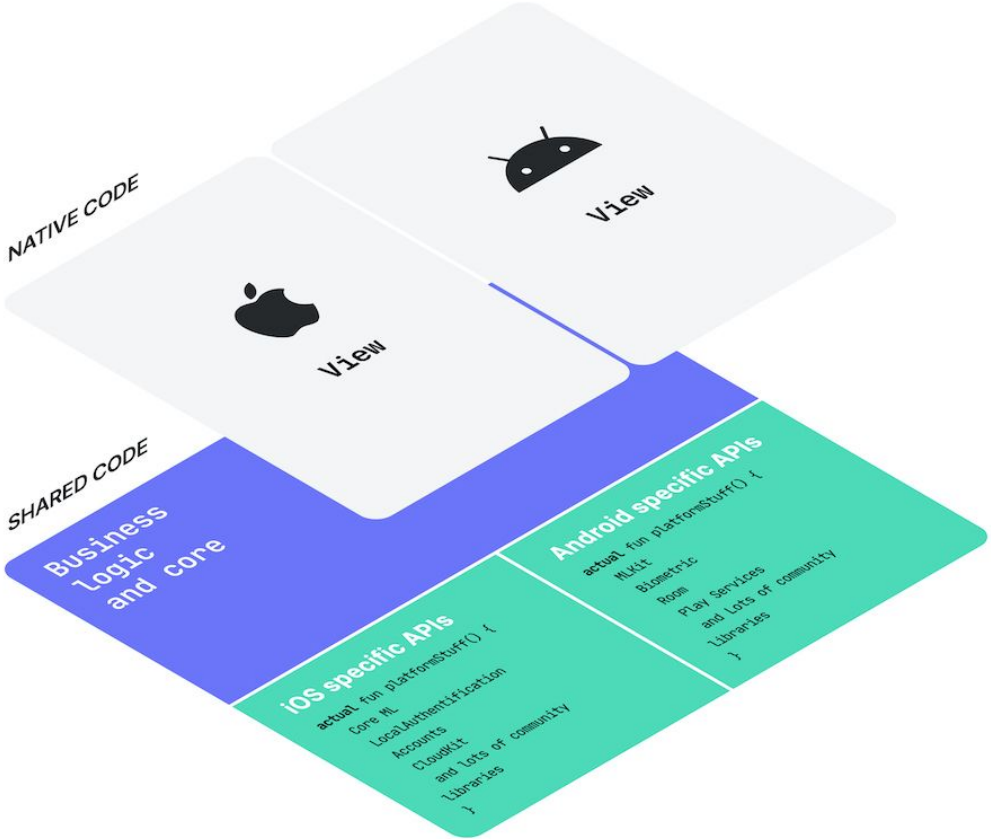




Kotlin-Multiplatform Mobile

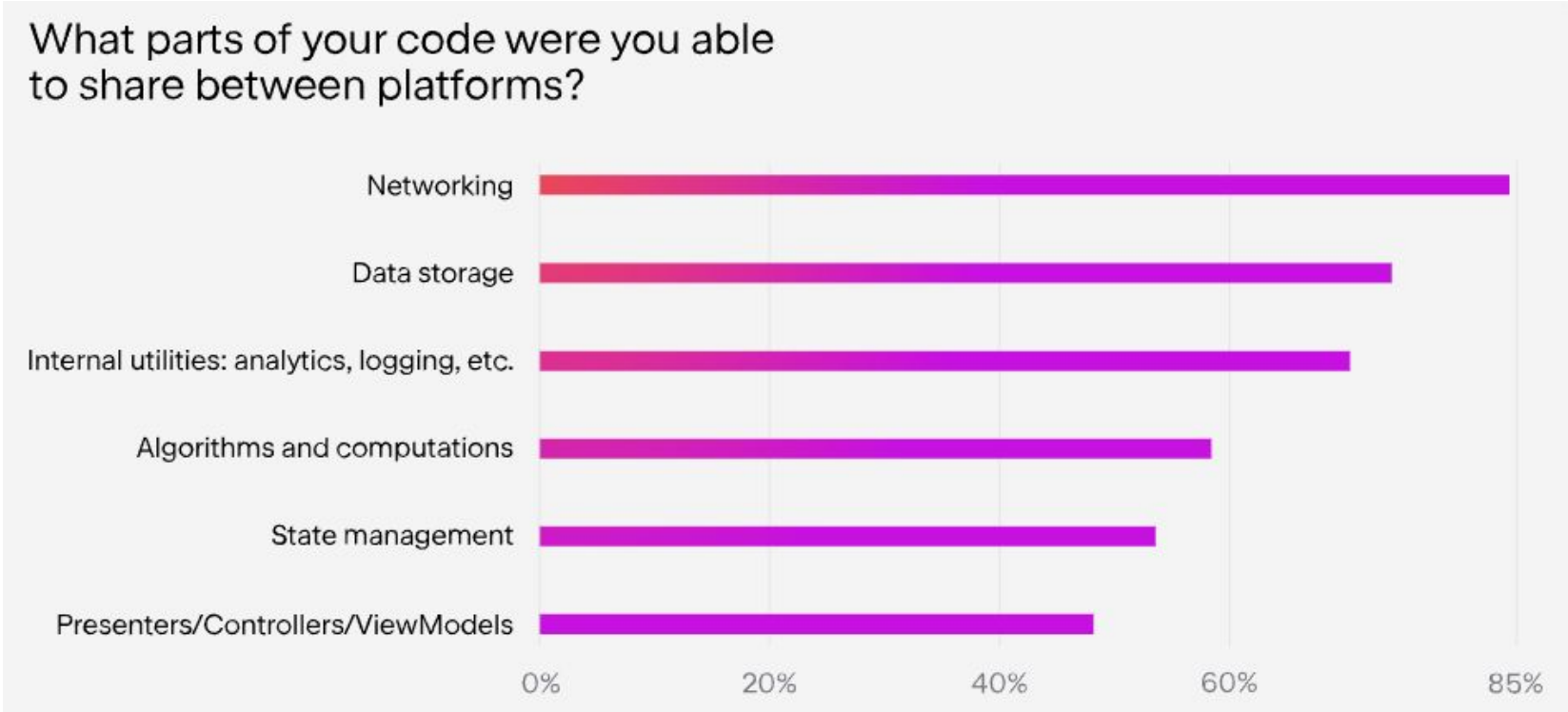
- Sdílení business logiky aplikace
- UI se píše nativně (Compose + SwiftUI)
  - UX uživatele stejná jako v případě nativního vývoje
  - Dnes je možnost používat Compose Multiplatform
- V současnosti používají KMM i velké firmy jako
  - LiveSport
  - Netflix
  - VMWare
  - etc…
- **Samples GIT**

KMM - Sdílení logiky



Zdroj

KMM - které částí kódu nejčastěji?



Zdroj

KMM - Výhody a Nevýhody

👍 Výhody

- **Sdílení business logiky**, tzn. menší prostor pro chyby (menší duplicita kódu)
- UX uživatele stejné jako při nativním vývoji
- **Code-Review** napříč týmy
- **Menší vývojová kapacita ==** menší nárok na rozpočet projektu
- Lze zavést postupně pomocí **modularizace** projektu

👎 Nevýhody

- Stále jsou zapotřebí platform specific vývojáři
  - **Ale tomu se u cross-platform vývoje často nevyhneme**
- KMM třídy
  - Je zapotřebí používat specifické třídy, které podporují KMM
- Na trhu není tolik specializovaných vývojářů

📔 Knihovny

- Retrofit | Fuel
  - HTTP client
- Room | SQLDelight
  - Databáze nad SQLite
- Hilt | Koin
  - Dependency Injection
- Coroutines| RxAndroid
  - Asynchronicita
- Glide | Piccasso | Coil
  - Práce s Obrázky
- Timber
  - Logování

👀 Koho sledovat

- **Jake Wharton**
  - jakewharton.com
- **Donn Felker and Kaushik Gopal**
  - https://fragmentedpodcast.com/
- **Adam McNeilly**
  - @AdamMc331
- **Chris Banes**
  - @chrisbanes
- **Alexey Bykov**
  - @nonewsss

Děkuji za pozornost 🙂