Tvorba mobilních aplikací pro Android

4. seminář

Matěj Dostál

Univerzita Palackého v Olomouci

4. 3. 2025

Struktura projektu – Hlavní přehled

- Po vytvoření projektu vidíme v Android Studiu (v režimu zobrazení „Android“) dvě hlavní části

Struktura projektu – Hlavní přehled

- Po vytvoření projektu vidíme v Android Studiu (v režimu zobrazení „Android“) dvě hlavní části
- **Modul `app`** – Obsahuje veškerý kód a zdroje naší aplikace
  - `manifests/` – Konfigurace aplikace
  - `java/` – Zdrojové kódy
  - `res/` – Statické zdroje (obrázky, texty, UI)

Struktura projektu – Hlavní přehled

- Po vytvoření projektu vidíme v Android Studiu (v režimu zobrazení „Android“) dvě hlavní části
- **Modul `app`** – Obsahuje veškerý kód a zdroje naší aplikace
  - `manifests/` – Konfigurace aplikace
  - `java/` – Zdrojové kódy
  - `res/` – Statické zdroje (obrázky, texty, UI)
- **Gradle Scripts** – Skripty pro sestavení aplikace
  - Nástroj, který se stará o kompilaci kódu, správu knihoven a vytvoření výsledného APK/AAB souboru

AndroidManifest.xml

- Základní konfigurační soubor celé aplikace (v XML)

AndroidManifest.xml

- Základní konfigurační soubor celé aplikace (v XML)
- Co musí systém o aplikaci vědět, než ji vůbec dokáže spustit?
  - **Základní metadata:** Název balíčku (Package name), ikona aplikace, základní téma
  - **Komponenty aplikace:** Zde musí být deklarovány všechny Aktivity (`<activity>`), Služby (`<service>`), Broadcast Receivery a Content Providery
  - Bez deklarace v manifestu systém komponentu nespustí a aplikace spadne

AndroidManifest.xml

- Základní konfigurační soubor celé aplikace (v XML)
- Co musí systém o aplikaci vědět, než ji vůbec dokáže spustit?
  - **Základní metadata:** Název balíčku (Package name), ikona aplikace, základní téma
  - **Komponenty aplikace:** Zde musí být deklarovány všechny Aktivity (`<activity>`), Služby (`<service>`), Broadcast Receivery a Content Providery
  - Bez deklarace v manifestu systém komponentu nespustí a aplikace spadne
- **Oprávnění (Permissions)**
  - Definice přístupu k chráněným částem systému (např. `<uses-permission android:name="android.permission.INTERNET"/>`)
  - Zde deklarujeme jak install-time, tak runtime oprávnění

Zdrojové kódy (`.kt` soubory)

- Složka `java/`
  - Historický název, dnes se sem primárně umisťují **Kotlin** soubory (`.kt`)
  - Obsahuje veškerou aplikační logiku

Zdrojové kódy (`.kt` soubory)

- Složka `java/`
  - Historický název, dnes se sem primárně umisťují **Kotlin** soubory (`.kt`)
  - Obsahuje veškerou aplikační logiku
- Organizace kódu
  - Kód je strukturován do balíčků (packages), např. `com.moje.aplikace`
  - Zde vytváříme třídy pro Aktivity, Fragmenty, ViewModely, datové modely a pomocné funkce

Zdrojové kódy (`.kt` soubory)

- Složka `java/`
  - Historický název, dnes se sem primárně umisťují **Kotlin** soubory (`.kt`)
  - Obsahuje veškerou aplikační logiku
- Organizace kódu
  - Kód je strukturován do balíčků (packages), např. `com.moje.aplikace`
  - Zde vytváříme třídy pro Aktivity, Fragmenty, ViewModely, datové modely a pomocné funkce
- Testovací složky
  - `java (androidTest)` – Instrumentované testy běžící přímo na zařízení/emulátoru
  - `java (test)` – Lokální jednotkové (unit) testy běžící na počítači

Resources (složka `res/`) – 1. část

- Slouží k přísnému oddělení kódu od statického obsahu

Resources (složka `res/`) – 1. část

- Slouží k přísnému oddělení kódu od statického obsahu
- **`drawable/`** – Grafika a obrázky
  - Vektorová grafika (XML), PNG, JPEG, WebP
  - Také definice tvarů (shapes) a stavových seznamů (state lists, např. jak vypadá tlačítko po stisknutí)

Resources (složka `res/`) – 1. část

- Slouží k přísnému oddělení kódu od statického obsahu
- **`drawable/`** – Grafika a obrázky
  - Vektorová grafika (XML), PNG, JPEG, WebP
  - Také definice tvarů (shapes) a stavových seznamů (state lists, např. jak vypadá tlačítko po stisknutí)
- **`layout/`** – Definice uživatelského rozhraní
  - XML soubory definující vzhled obrazovek (pokud nepoužíváme moderní Jetpack Compose)
  - Popisuje rozmístění tlačítek, textů a dalších UI elementů

Resources (složka `res/`) – 2. část

- **`mipmap/`** – Ikony aplikace
  - Slouží výhradně pro ikony samotné aplikace (launcher icons)
  - Různé verze pro různé hustoty pixelů displejů (hdpi, xhdpi, xxhdpi, . . .), aby ikona byla vždy ostrá

Resources (složka `res/`) – 2. část

- **`mipmap/`** – Ikony aplikace
  - Slouží výhradně pro ikony samotné aplikace (launcher icons)
  - Různé verze pro různé hustoty pixelů displejů (hdpi, xhdpi, xxhdpi, . . .), aby ikona byla vždy ostrá
- **`values/`** – Datové hodnoty a konstanty
  - `strings.xml` – Všechny texty v aplikaci (klíčové pro snadnou lokalizaci/překlad do jiných jazyků)
  - `colors.xml` – Definice barev pomocí HEX kódů
  - `themes.xml` (`styles.xml`) – Definice celkového vzhledu, barevných palet (světlý/tmavý režim) a stylů komponent

Gradle Scripts – Základy sestavení

- Moderní Android projekty používají pro konfiguraci jazyk Kotlin (`.gradle.kts`)

Gradle Scripts – Základy sestavení

- Moderní Android projekty používají pro konfiguraci jazyk Kotlin (`.gradle.kts`)
- **`settings.gradle.kts`**
  - Definuje jméno celého projektu
  - Určuje, které moduly (např. `:app`) jsou do projektu zahrnuty

Gradle Scripts – Základy sestavení

- Moderní Android projekty používají pro konfiguraci jazyk Kotlin (`.gradle.kts`)
- **`settings.gradle.kts`**
  - Definuje jméno celého projektu
  - Určuje, které moduly (např. `:app`) jsou do projektu zahrnuty
- **`libs.versions.toml`** (Version Catalog)
  - Centralizované místo pro správu verzí všech knihoven a pluginů
  - Usnadňuje aktualizace závislostí

Gradle Scripts – Základy sestavení

- Moderní Android projekty používají pro konfiguraci jazyk Kotlin (`.gradle.kts`)
- **`settings.gradle.kts`**
  - Definuje jméno celého projektu
  - Určuje, které moduly (např. `:app`) jsou do projektu zahrnuty
- **`libs.versions.toml`** (Version Catalog)
  - Centralizované místo pro správu verzí všech knihoven a pluginů
  - Usnadňuje aktualizace závislostí
- **`gradle-wrapper.properties`**
  - Zajišťuje, že všichni vývojáři používají k sestavení stejnou verzi nástroje Gradle

Gradle Scripts – Build soubory

- **`build.gradle.kts` (Project)**
  - Konfigurace na úrovni celého projektu
  - Přidává repozitáře a definuje hlavní pluginy platné pro všechny moduly

Gradle Scripts – Build soubory

- **`build.gradle.kts` (Project)**
  - Konfigurace na úrovni celého projektu
  - Přidává repozitáře a definuje hlavní pluginy platné pro všechny moduly
- **`build.gradle.kts` (Module: app)** – Nejdůležitější Gradle soubor
  - *Základní nastavení:* `applicationId` (unikátní identifikátor v Google Play), verze aplikace (`versionCode`, `versionName`)
  - *SDK verze:* `minSdk` (nejstarší podporovaný Android), `targetSdk` (pro jakou verzi je aplikace optimalizována)
  - *Dependencies:* Zde přidáváme externí knihovny, které chceme v projektu používat (např. knihovny pro UI, stahování z internetu, . . .)

Tvorba UI: XML vs. Jetpack Compose

- Tradiční přístup (XML Layouty)
  - UI je definováno ve statických XML souborech (složka `res/layout/`)
  - Logika a vzhled jsou fyzicky odděleny (XML pro vzhled, Kotlin pro logiku)
  - Imperativní přístup: v kódu musíme UI prvky najít a ručně jim změnit stav (např. `button.text = "Kliknuto"`)

Tvorba UI: XML vs. Jetpack Compose

- Tradiční přístup (XML Layouty)
  - UI je definováno ve statických XML souborech (složka `res/layout/`)
  - Logika a vzhled jsou fyzicky odděleny (XML pro vzhled, Kotlin pro logiku)
  - Imperativní přístup: v kódu musíme UI prvky najít a ručně jim změnit stav (např. `button.text = "Kliknuto"`)
- Moderní přístup (Jetpack Compose)
  - Doporučovaný standard od Googlu pro nové aplikace
  - UI je definováno přímo v Kotlin kódu pomocí tzv. *Composable* funkcí (vše je ve složce `java/`)
  - Zcela odpadá nutnost používat složku `res/layout/` a XML

Tvorba UI: XML vs. Jetpack Compose

- Tradiční přístup (XML Layouty)
  - UI je definováno ve statických XML souborech (složka `res/layout/`)
  - Logika a vzhled jsou fyzicky odděleny (XML pro vzhled, Kotlin pro logiku)
  - Imperativní přístup: v kódu musíme UI prvky najít a ručně jim změnit stav (např. `button.text = "Kliknuto"`)
- Moderní přístup (Jetpack Compose)
  - Doporučovaný standard od Googlu pro nové aplikace
  - UI je definováno přímo v Kotlin kódu pomocí tzv. *Composable* funkcí (vše je ve složce `java/`)
  - Zcela odpadá nutnost používat složku `res/layout/` a XML
- Jaký to má dopad na strukturu projektu?
  - **Méně přepínání kontextu:** Logika i UI jsou napsány ve stejném jazyce (Kotlin)
  - Deklarativní přístup: pouze popíšeme, jak má UI vypadat na základě aktuálních dat, systém se sám postará o překreslení
  - Složka `res/` zůstává vyhrazena už jen pro čistě statické zdroje (ikony, obrázky, textové řetězce)

Témata a styly v Jetpack Compose

- Konec `themes.xml` a `styles.xml` – vše je nyní v Kotlinu

Témata a styly v Jetpack Compose

- Konec `themes.xml` a `styles.xml` – vše je nyní v Kotlinu
- **Material Design 3 (MaterialTheme)**
  - Compose nativně využívá návrhový systém Material Design
  - Téma aplikace je definováno jako speciální Composable funkce (např. `MyAppTheme { ... }`), do které obalíme celé UI

Témata a styly v Jetpack Compose

- Konec `themes.xml` a `styles.xml` – vše je nyní v Kotlinu
- **Material Design 3 (MaterialTheme)**
  - Compose nativně využívá návrhový systém Material Design
  - Téma aplikace je definováno jako speciální Composable funkce (např. `MyAppTheme { ... }`), do které obalíme celé UI
- **Tři pilíře tématu (Složka `ui/theme/`)**
  - `Color.kt` – Definice barevné palety (světlý vs. tmavý režim / `ColorScheme`)
  - `Type.kt` – Definice typografie (fonty, velikosti nadpisů a odstavců)
  - `Shape.kt` – Tvary komponent (např. jak moc zaoblené mají být rohy tlačítek)

Odbočka: Material Design 3 (MD3)

- Aktuální verze open-source designového systému od Googlu
  - Oficiální dokumentace a specifikace: `https://m3.material.io/`

Odbočka: Material Design 3 (MD3)

- Aktuální verze open-source designového systému od Googlu
  - Oficiální dokumentace a specifikace: `https://m3.material.io/`
- **K čemu slouží a co přináší?**
  - *Hotové UI komponenty* – Tlačítka, textová pole, karty, navigační lišty. Vše předprogramované, vypadající moderně a jednotně napříč Androidem
  - *Vestavěná přístupnost (Accessibility)* – Komponenty automaticky řeší správný kontrast barev, velikosti cílů pro dotyk a podporu pro čtečky obrazovky

Odbočka: Material Design 3 (MD3)

- Aktuální verze open-source designového systému od Googlu
  - Oficiální dokumentace a specifikace: `https://m3.material.io/`
- **K čemu slouží a co přináší?**
  - *Hotové UI komponenty* – Tlačítka, textová pole, karty, navigační lišty. Vše předprogramované, vypadající moderně a jednotně napříč Androidem
  - *Vestavěná přístupnost (Accessibility)* – Komponenty automaticky řeší správný kontrast barev, velikosti cílů pro dotyk a podporu pro čtečky obrazovky
- **Dynamic Color (Material You)**
  - Klíčová novinka verze 3 – barvy aplikace se dokážou automaticky odvodit z tapety, kterou má uživatel na domovské obrazovce telefonu
  - Vytváří tak silně personalizovaný vzhled aplikace

Odbočka: Material Design 3 (MD3)

- Aktuální verze open-source designového systému od Googlu
  - Oficiální dokumentace a specifikace: `https://m3.material.io/`
- **K čemu slouží a co přináší?**
  - *Hotové UI komponenty* – Tlačítka, textová pole, karty, navigační lišty. Vše předprogramované, vypadající moderně a jednotně napříč Androidem
  - *Vestavěná přístupnost (Accessibility)* – Komponenty automaticky řeší správný kontrast barev, velikosti cílů pro dotyk a podporu pro čtečky obrazovky
- **Dynamic Color (Material You)**
  - Klíčová novinka verze 3 – barvy aplikace se dokážou automaticky odvodit z tapety, kterou má uživatel na domovské obrazovce telefonu
  - Vytváří tak silně personalizovaný vzhled aplikace
- **Spojení s Jetpack Compose**
  - Compose obsahuje nativní knihovnu (`androidx.compose.material3`), takže vývojář může MD3 komponenty používat „out-of-the-box“

Práce s řetězci a barvami

- **Řetězce (Strings)** – zde se XML stále drží
  - I v Compose se doporučuje ukládat texty do `res/values/strings.xml`
  - Důvodem je snadná lokalizace a překlad aplikace do jiných jazyků
  - V Compose je načteme jednoduše: `text = stringResource(R.string.app_name)`

Práce s řetězci a barvami

- **Řetězce (Strings)** – zde se XML stále drží
  - I v Compose se doporučuje ukládat texty do `res/values/strings.xml`
  - Důvodem je snadná lokalizace a překlad aplikace do jiných jazyků
  - V Compose je načteme jednoduše: `text = stringResource(R.string.app_name)`
- **Barvy (Colors)**
  - Barvy definujeme v Kotlinu pomocí třídy `Color` (např. `Color(0xFFBB86FC)`)
  - Místo natvrdo zapsaných barev bychom v UI měli vždy používat barvy z našeho tématu: `color = MaterialTheme.colorScheme.primary`
  - Aplikace se pak automaticky přizpůsobí světlém/tmavému režimu

Měrné jednotky v Androidu

- Proč nepoužívat klasické pixely (`px`)?
  - Každý telefon má jinou hustotu pixelů na palec (ppi). Tlačítko velké 100 px by na starém telefonu bylo obrovské, na moderním 4K displeji miniaturní

Měrné jednotky v Androidu

- Proč nepoužívat klasické pixely (`px`)?
  - Každý telefon má jinou hustotu pixelů na palec (ppi). Tlačítko velké 100 px by na starém telefonu bylo obrovské, na moderním 4K displeji miniaturní
- **`dp` (Density-independent Pixels)**
  - Základní jednotka pro rozměry a odsazení v UI (šířka, výška, padding, margin)
  - Systém je automaticky přepočítá na pixely podle hustoty displeje daného zařízení, takže prvek vypadá všude fyzicky stejně velký
  - V Compose: `modifier = Modifier.padding(16.dp)`

Měrné jednotky v Androidu

- Proč nepoužívat klasické pixely (`px`)?
  - Každý telefon má jinou hustotu pixelů na palec (ppi). Tlačítko velké 100 px by na starém telefonu bylo obrovské, na moderním 4K displeji miniaturní
- **`dp` (Density-independent Pixels)**
  - Základní jednotka pro rozměry a odsazení v UI (šířka, výška, padding, margin)
  - Systém je automaticky přepočítá na pixely podle hustoty displeje daného zařízení, takže prvek vypadá všude fyzicky stejně velký
  - V Compose: `modifier = Modifier.padding(16.dp)`
- **`sp` (Scale-independent Pixels)**
  - Používá se **výhradně pro velikost textu**
  - Funguje stejně jako `dp`, ale navíc bere v úvahu nastavení velikosti písma, které si uživatel zvolil v nastavení systému (např. pro zrakově postižené)
  - V Compose: `fontSize = 18.sp`

Obrázky, ikonky a drawables

- **Složka `res/drawable/`**
  - Slouží k ukládání statických obrázků (PNG, JPEG)
  - V Compose je vykreslíme pomocí: `Image(painter = painterResource(id = R.drawable.muj_obrazek), ...)`

Obrázky, ikonky a drawables

- **Složka `res/drawable/`**
  - Slouží k ukládání statických obrázků (PNG, JPEG)
  - V Compose je vykreslíme pomocí: `Image(painter = painterResource(id = R.drawable.muj_obrazek), ...)`
- **Vektorová grafika (Doporučeno)**
  - Místo bitmap se v Androidu primárně používají vektory (`VectorDrawable` v XML)
  - Jsou ostré na každém displeji a zabírají minimum místa (malá velikost APK)

Obrázky, ikonky a drawables

- **Složka `res/drawable/`**
  - Slouží k ukládání statických obrázků (PNG, JPEG)
  - V Compose je vykreslíme pomocí: `Image(painter = painterResource(id = R.drawable.muj_obrazek), ...)`
- **Vektorová grafika (Doporučeno)**
  - Místo bitmap se v Androidu primárně používají vektory (`VectorDrawable` v XML)
  - Jsou ostré na každém displeji a zabírají minimum místa (malá velikost APK)
- **Materiálové ikony přímo v Compose**
  - Knihovna `material-icons-extended` obsahuje stovky předpřipravených vektorových ikon
  - Použití bez nutnosti cokoliv stahovat do `res/`: `Icon(imageVector = Icons.Default.Favorite, contentDescription = ...)`

Orientace displeje (Na výšku / Na šířku)

- Co se stane, když uživatel otočí telefon?
  - Nastane tzv. *Configuration Change* $\Rightarrow$ systém aktuální aktivitu „zabije“ a znovu vytvoří, aby mohl načíst případné jiné zdroje (např. širší layout)

Orientace displeje (Na výšku / Na šířku)

- Co se stane, když uživatel otočí telefon?
  - Nastane tzv. *Configuration Change* ⇒ systém aktuální aktivitu „zabije“ a znovu vytvoří, aby mohl načíst případné jiné zdroje (např. širší layout)
- **Jak to řešit v Compose?**
  - **Ztráta stavu:** Protože se obrazovka překresluje od nuly, musíme důležitý stav UI (např. co uživatel napsal do textového pole) obalit do `rememberSaveable`, aby přežil otočení displeje (později vyřešíme vhodnou architekturou)

Orientace displeje (Na výšku / Na šířku)

- Co se stane, když uživatel otočí telefon?
  - Nastane tzv. *Configuration Change* ⇒ systém aktuální aktivitu „zabije“ a znovu vytvoří, aby mohl načíst případné jiné zdroje (např. širší layout)
- **Jak to řešit v Compose?**
  - **Ztráta stavu:** Protože se obrazovka překresluje od nuly, musíme důležitý stav UI (např. co uživatel napsal do textového pole) obalit do `rememberSaveable`, aby přežil otočení displeje (později vyřešíme vhodnou architekturou)
- **Responzivní UI**
  - Pomocí `LocalConfiguration.current.orientation` můžeme přímo v kódu zjistit, jak je telefon otočený, a podle toho vykreslit jiné UI.
  - Pokročilejší přístup využívá tzv. *WindowSizeClass* (rozlišuje compact, medium, expanded obrazovky – vhodné pro podporu tabletů a skládacích telefonů)
- Ukázka

Aplikační sandbox

- Aplikace běží každá ve svém procesu – každé aplikaci je přiřazeno unikátní UID a jsou navzájem izolovány
- Android poté toto UID využívá pro vytvoření kernel-level aplikačního sandboxu
- Jádro poté zajišťuje izolaci aplikací a systému na úrovní procesů běžnými Linuxovými prostředky
- Ve výchozím stavu spolu aplikace nemohou interagovat a mají omezený přístup k OS
- Pouze základní funkcionality jsou aplikaci poskytnuty, o více specifické musíme zažádat

Permissions

- Ideální situace: nepotřebujeme žádné speciální oprávnění

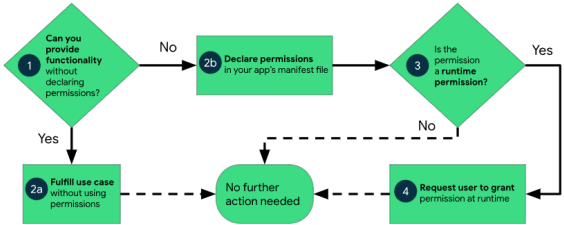
Permissions

- Ideální situace: nepotřebujeme žádné speciální oprávnění
- Pokud by naše aplikace mohla vyžadovat přístup k omezeným akcím/datům – můžeme to udělat tak, abychom speciální přístup nepotřebovali?
  - Mnoho toho lze udělat, aniž bychom nějaké speciální oprávnění potřebovali (např. vyfocení fotky nebo zastavení přehrávání média)

Permissions

- Ideální situace: nepotřebujeme žádné speciální oprávnění
- Pokud by naše aplikace mohla vyžadovat přístup k omezeným akcím/datům – můžeme to udělat tak, abychom speciální přístup nepotřebovali?
  - Mnoho toho lze udělat, aniž bychom nějaké speciální oprávnění potřebovali (např. vyfocení fotky nebo zastavení přehrávání média)
- Pokud přesto potřebujeme speciální oprávnění, musíme je definovat
  - Install-time permissions – automaticky uděleny při instalaci
    - Akce s minimálním dopadem na systém či jiné aplikace
    - Deklarovány v `AndroidManifest.xml`
    - Seznam těchto oprávnění je zobrazen na stránce s detaily o aplikaci
  - Runtime permissions – uživatel je musí explicitně udělat za běhu aplikace
    - „Nebezpečné“, větší potenciální dopad na systém či jiné aplikace
    - Potřeba zažádat o ně předtím, než je budeme potřebovat – nepředpokládat, že už byly uděleny
    - Často přistupují k *soukromým datům* uživatele, které můžou potenciálně obsahovat citlivé informace (např. lokace, kontakty, mikrofon, . . .)

Permissions – workflow



Permissions – best practices

- Vyžadovat pouze nezbytné množství oprávnění – aplikace by měla požadovat pouze nezbytné oprávnění k dokončení požadované akce

Permissions – best practices

- Vyžadovat pouze nezbytné množství oprávnění – aplikace by měla požadovat pouze nezbytné oprávnění k dokončení požadované akce
- Spojit runtime oprávnění se specifickými akcemi – vyžadovat oprávnění co nejpozději ve „flow“ aplikace, jak je to jen možné
  - Např. pokud máme aplikaci na chatování, která umožňuje i odesílání hlasových zpráv, o oprávnění přístupu k mikrofonu bychom měli zažádat až když uživatel stiskne tlačítko pro nahrávání hlasové zprávy

Permissions – best practices

- Vyžadovat pouze nezbytné množství oprávnění – aplikace by měla požadovat pouze nezbytné oprávnění k dokončení požadované akce
- Spojit runtime oprávnění se specifickými akcemi – vyžadovat oprávnění co nejpozději ve „flow“ aplikace, jak je to jen možné
  - Např. pokud máme aplikaci na chatování, která umožňuje i odesílání hlasových zpráv, o oprávnění přístupu k mikrofonu bychom měli zažádat až když uživatel stiskne tlačítko pro nahrávání hlasové zprávy
- Pozor na externí knihovny – při použití knihovny zároveň přebíráme její oprávnění, která vyžaduje $\Rightarrow$ měli bychom vědět na co je vyžaduje

Permissions – best practices

- Vyžadovat pouze nezbytné množství oprávnění – aplikace by měla požadovat pouze nezbytné oprávnění k dokončení požadované akce
- Spojit runtime oprávnění se specifickými akcemi – vyžadovat oprávnění co nejpozději ve „flow“ aplikace, jak je to jen možné
  - Např. pokud máme aplikaci na chatování, která umožňuje i odesílání hlasových zpráv, o oprávnění přístupu k mikrofonu bychom měli zažádat až když uživatel stiskne tlačítko pro nahrávání hlasové zprávy
- Pozor na externí knihovny – při použití knihovny zároveň přebíráme její oprávnění, která vyžaduje $\Rightarrow$ měli bychom vědět na co je vyžaduje
- Transparentnost – když vyžadujeme nějaké oprávnění, měli bychom být transparentní k čemu jej vyžadujeme, proč a co za funkcionality ovlivní případné odmítnutí

Příklad z praxe: Přístup k poloze

- Poloha je klasickým příkladem komplexního *Runtime oprávnění*

Příklad z praxe: Přístup k poloze

- Poloha je klasickým příkladem komplexního *Runtime oprávnění*
- Dnes už nejde jen o volbu „Povolit / Zamítnout“
  - **Přesnost (Android 12+):** Uživatel si může vybrat, zda aplikaci poskytne přesnou lokaci (`FINE_LOCATION` – GPS), nebo jen přibližnou oblast (`COARSE_LOCATION` – vysílače/Wi-Fi).
  - **Časové omezení:** Lze udělit přístup jen jednorázově, nebo pouze pokud je aplikace aktivně používána na obrazovce.

Příklad z praxe: Přístup k poloze

- Poloha je klasickým příkladem komplexního *Runtime oprávnění*
- Dnes už nejde jen o volbu „Povolit / Zamítnout“
  - **Přesnost (Android 12+):** Uživatel si může vybrat, zda aplikaci poskytne přesnou lokaci (`FINE_LOCATION` – GPS), nebo jen přibližnou oblast (`COARSE_LOCATION` – vysílače/Wi-Fi).
  - **Časové omezení:** Lze udělit přístup jen jednorázově, nebo pouze pokud je aplikace aktivně používána na obrazovce.
- **Lokace na pozadí (`BACKGROUND_LOCATION`)**
  - Získat polohu zavřené aplikace (např. fitness tracker) je dnes velmi přísně střeženo.
  - Aplikace o to nesmí požádat rovnou – nejprve musí získat polohu v popředí a až poté může uživatele přesměrovat hluboko do nastavení telefonu.

Úkol

1. Do aplikace z minulého úkolu přidejte obrazovku, která zobrazí mapu a na ní aktuální polohu uživatele
2. Ošetřete získání oprávnění od uživatele.
3. Pro odvážnější: při zavření aplikace se poloha dál trackuje a ve status baru se zobrazuje notifikace se souřadnicemi.