VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

# Možnosti XML – Definice pozicování komponent uživatelského rozhraní v Android

Bakalářská práce

Autor práce: Tomáš Šejstal

Vedoucí práce: Ing. Marek Musil

Jihlava 24.6.2024

# Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

# ZADÁNÍ BAKALÁŘSKÉ

| | |
|---|---|
| Autor práce: | **Tomáš Šejstal** |
| Studijní program: | Aplikovaná informatika |
| Obor: | Aplikovaná informatika |
| Garant studijního programu: | Ing. Lenka Kuklišová Pavelková, Ph.D. |
| Název práce: | **Možnosti XML-definice pozicování komponent uživatelského rozhraní v Android** |
| Vedoucí práce: | Ing. Marek Musil |
| Cíl práce: | Cílem práce je nastudovat dostupné zdroje (literatura, webové stránky s tutoriály) a představit možnosti a způsoby, kterými lze při implementaci uživatelského rozhraní mobilní aplikace definovat umístění (pozici) komponenty v okně aplikace. Bude provedeno posouzení jednotlivých řešení na základě dostupných informací a skutečností publikovaných v literatuře. A dále bude dle možností doplněna vlastní interpretace. Možné způsoby budou názorně ukázány prostřednictvím vhodné zvolených ukázek. Práce bude využitelná pro přípravu výuky předmětu Programování pro mobilní platformy. |

# Abstrakt

Bakalářská práce se zaměřuje na studium a analýzu pozicování komponent v uživatelském rozhraní mobilních aplikací a to ve frameworku Android. Diskutují se zde možnost pozicování a jejich interpretace v XML. Pozicování komponent je zásadním aspektem při tvorbě mobilních aplikací, dopad pozicování je důležitý z uživatelského hlediska při používání aplikace. Hlavním cílem práce je poskytnout komplexní přehled o různých typech layoutů v Androidu, včetně lineárního layoutu, relativního layoutu, constraint layoutu a jejich případných kombinací**.** Aktuálnost tématu je zdůrazněna v kontextu rychle rostoucího mobilního vývoje, zvýšené různorodosti zařízení a konkurence v oblasti mobilních aplikací. Práce má také potenciál být využita pro výuku předmětu Programování pro mobilní platformy.



# Klíčová slova

XML; Android; Pozicování; Layouty v androidu; Mobilní vývoj; Optimalizace uživatelského rozhraní; Uživatelská zkušenost; Grafické uživatelské rozhraní



# Abstract

The bachelor thesis focuses on the study and analysis of component positioning in the user interface of mobile applications using XML and the Android framework.

The main objective of the thesis is to provide a comprehensive overview of the different types of layouts in Android, including linear layout, relative layout, constraint layout and their possible combinations.

The relevance of the topic is highlighted in the context of rapidly growing mobile development, increased diversity of device and competition in mobile applications. The work also has the potential to be used for teaching the subject of Programming for Mobile Platforms.



# Keywords

XML; Android; Android Studio; Positioning; Layouts in Android; Mobile Development; User Interface Optimization; User Experience; Graphical User Interface

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle směrnice prorektora pro studium č. 2/2020, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 24. června 2024

…….

Podpis studenta/ky

# Poděkování

*V první řadě bych rád poděkoval svému vedoucím práce panu Ing. Marku Musilovi za výborné vedení, jeho cenné rady a poznámky k samotné práci.*

*Velké díky patří mojí rodině za podporu při studiu a v neposlední řadě mé přítelkyni, která mé strasti prožívala se mnou po celou dobu studia.*

# Obsah

**Seznam obrázků** ..... **8**
**Seznam tabulek** ..... **9**
**Seznam ukázek kódu** ..... **10**
**Seznam zkratek** ..... **11**
**Úvod** ..... **12**
**1 Základní terminologie pro pozicování** ..... **13**
1.1 XML (Extensible Markup Language) ..... 13
1.2 Android Studio ..... 13
1.3 Android ..... 14
1.4 Layout ..... 14
**2 Aktuálnost tématu** ..... **15**
2.1 Růst mobilního vývoje ..... 15
2.2 Různorodost zařízení ..... 16
2.3 Uživatelská zkušenost a konkurence ..... 16
**3 Layouty v Androidu** ..... **17**
3.1 Linear layout ..... 17
3.2 Relative layout ..... 17
3.3 Constraint layout ..... 18
3.4 Table layout ..... 18
3.5 Další layouty (Frame layout, Grid layout, Coordinator layout) ..... 19
3.5.1 Popis layoutů ..... 19
3.5.2 Základní Vlastnosti a Parametry ..... 19
3.6 Kombinace Layoutů ..... 20
3.6.1 Proč kombinovat layouty ..... 20
3.6.2 Příklady kombinací layoutů ..... 20
**4 Historie a vývoj layoutů pro Android** ..... **21**
4.1 Historický vývoj Android layoutů ..... 21
4.2 Trendy v návrhu uživatelského rozhraní ..... 22
**5 Návrh optimálního uživatelského rozhraní** ..... **23**
5.1 Jak dosáhnout optimálního uspořádání komponent ..... 23
5.2 Testování a zpětná vazba od uživatelů ..... 23
**6 Možnosti budoucího vývoje** ..... **24**
6.1 Nové technologie v mobilním vývoji ..... 24
**7 Tvorba layoutů** ..... **25**
7.1 Linear layout ..... 25
7.2 Relative layout ..... 28
7.3 Constraint layout ..... 30
7.4 Table layout ..... 32
**8 Kombinace layoutů** ..... **34**
8.1 Lineární horizontální a lineární vertikální ..... 34
8.2 Scroll view, constraint a linear layout ..... 36

**9 Pokročilé layouty v Androidu ..... 38**
9.1 Flexbox layout ..... 38
9.2 ConstraintSet layout ..... 40
9.3 Motion layout ..... 42
**10 Responzivní design ..... 44**
10.1 Různé velikosti a rozlišení obrazovek ..... 44
10.2 Adaptivní UI pro různé orientace ..... 45
**11 Testování a automatizace UI ..... 46**
11.1 Manuální testování ..... 46
11.2 Automatizované testování ..... 47
**Závěr ..... 48**
**Seznam použité literatury ..... 49**
**Přílohy ..... 51**

# Seznam obrázků

Obrázek 1 Android studio Logo ..... 13
Obrázek 2 Android Logo ..... 14
Obrázek 3 Lineární layout – Vertikální ..... 25
Obrázek 4 Lineární layout – Horizontální ..... 25
Obrázek 5 Použití gravitace na střed ..... 26
Obrázek 6 Rozložení pomocí váhy 1:2 ..... 27
Obrázek 7 Rozložení pomocí váhy 1:3 ..... 27
Obrázek 8 Kombinace pevné šířky a váhy 1:1 ..... 27
Obrázek 9 Relative layout ..... 28
Obrázek 10 Relative layout – kóty ..... 28
Obrázek 11 Constraint layout ..... 30
Obrázek 12 Constraint vazby 1 ..... 30
Obrázek 13 Constraint vazby 2 ..... 30
Obrázek 14 Table layout ..... 32
Obrázek 15 Kombinace lineárních layoutů ..... 35
Obrázek 16 Prvky scroll, linear layoutu ..... 36
Obrázek 17 Rozložení scroll, linear layoutus ..... 36
Obrázek 18 Scroll, linear layout – Tablet ..... 37
Obrázek 19 Flexbox ..... 38
Obrázek 20 ConstraintSet 1 ..... 40
Obrázek 21 ConstraintSet 2 ..... 40
Obrázek 22 Motion layout 1 ..... 42
Obrázek 23 Motion layout 2 ..... 42

# Seznam tabulek

Tabulka 1 - Počet aplikací v obchodě Play 2009-2023 ..... 15

# Seznam ukázek kódu

Ukázka 1 Vertikální uspořádání ..... 26
Ukázka 2 Zarovnání textu ..... 26
Ukázka 3 Rozdělení dle váhy ..... 27
Ukázka 4 Relative layout ..... 29
Ukázka 5 Constraint layout ..... 31
Ukázka 6 Table layout ..... 33
Ukázka 7 Vnoření lineárních layoutů ..... 35
Ukázka 8 implementace flexboxu ..... 38
Ukázka 9 Flexbox ..... 39
Ukázka 10 Změna orientace obrazovky ..... 45

# Seznam zkratek

| | |
|---|---|
| GUI | Graphical User Interface – Grafické uživatelské rozhraní |
| IDE | Integrated development environment – Integrované vývojové prostředí |
| OS | Operační systém |
| SDK | Software development kit – Sada vývojových nástrojů pro tvorbu softwaru |
| UI | User interface – Uživatelské prostředí |
| UX | User experience – Uživatelská zkušenost |
| XML | Extensible Markup Language |
| dp | density-independent pixels – jednotka pro určení velikosti na displeji |
| px | pixel |

# Úvod

V dnešní době mobilních aplikací a technologického pokroku je neustálým úkolem pro vývojáře zajistit uživatelům přívětivé uživatelské rozhraní. Jedním z klíčových aspektů tohoto úkolu je definování umístění komponenty v okně aplikace, což má výrazný vliv na uživatelský zážitek a efektivitu aplikace. K definování umístění komponent při vývoji mobilních aplikací existuje více možností, které lze i vzájemně kombinovat. Cílem bakalářské práce je důkladně nastudovat dostupné zdroje, včetně literatury a webových tutoriálů, a představit různé možnosti a způsoby, jak lze efektivně definovat umístění komponent v mobilních aplikacích.

Motivací k výběru tohoto tématu je stále rostoucí důležitost uživatelského rozhraní v oblasti mobilního vývoje. Správná konfigurace umístění komponenty může ovlivnit nejen vizuální stránku aplikace, ale také uživatelskou přívětivost a uživatelskou interakci. Téma je klíčové pro zlepšení uživatelských zkušeností a úspěšnost mobilních aplikací. Roli jistě hraje i velikost zobrazovacího zařízení a rozlišení zobrazení. Osobní motivací je dosavadní zkušenost ve vývoji mobilních aplikací. Když jsem se touto otázkou zabýval, zjistil jsem, že existuje několik možností pozicování komponent. Zamýšlel jsem se pak nad rozdíly ve výsledku pozicování a nad vhodností použití jednotlivých možností.

Práce bude zahrnovat důkladné zhodnocení různých řešení na základě dostupných informací a publikované literatury. Kromě toho se pokusíme přinést vlastní interpretaci a zpřesnění, což má potenciál vylepšit a obohatit existující poznatky v této oblasti. Jednotlivé způsoby a postupy budou názorně ilustrovány pomocí konkrétních příkladů.

Práce má také potenciál k využití při přípravě výuky předmětu Programování pro mobilní platformy, kde studenti budou moci získat hlubší porozumění problematice umístění komponenty v rámci vývoje mobilních aplikací.

V následujících kapitolách bakalářské práce se budeme detailněji zabývat studiem a analýzou různých aspektů spojených s umístěním komponent v uživatelském rozhraní mobilních aplikací.

# 1 Základní terminologie pro pozicování

Pozicování komponent v uživatelském rozhraní je klíčovým prvkem pro dosažení efektivní a uživatelsky přívětivé aplikace. Správné pozicování umožňuje vytvářet vizuálně přitažlivé rozložení, které podporuje logiku a účel aplikace. Uživatelé rychleji rozumí obsahu, když jsou prvky na obrazovce logicky a esteticky uspořádány. Přesné a promyšlené pozicování také umožňuje vývojářům vytvářet responzivní design, který se přizpůsobuje různým velikostem obrazovek a zařízení, což je klíčový prvek v éře rozmanitých mobilních zařízení.

Pozicování komponent má přímý vliv na uživatelskou zkušenost (UX). Dobře umístěné prvky zjednodušují navigaci a zvyšují přehlednost, což vede k plynulejší interakci uživatele s aplikací. Naopak, nevhodné pozicování může způsobit zmatení a snížit přístupnost funkčností. Kromě toho ovlivňuje i estetiku a dojem z aplikace. Vývojáři by měli věnovat pozornost každému detailu pozicování, aby vytvořili harmonické a intuitivní uživatelské rozhraní. Celkově lze říct, že správné pozicování komponent je klíčem k vytváření uživatelských rozhraní, která nejen vizuálně oslovují, ale také poskytují optimální uživatelskou zkušenost.

## 1.1 XML (Extensible Markup Language)

XML je značkovací jazyk umožňující hierarchické organizování dat prostřednictvím značek. Základní strukturní jednotkou je element, který může obsahovat text, další elementy nebo atributy. XML je představeno (popsáno), například v (XML pro začátečníky, 2024) a (XML 2024).

XML se v Android Studiu používá k definování uživatelského rozhraní aplikace. Pomocí XML můžete přesně určit, jak by měla aplikace vypadat a jaké prvky by měla obsahovat. Tyto soubory jsou součástí každého projektu. Pomocí XML souborů mohou být definovány různé prvky, jako jsou tlačítka, textová pole, seznamy a mnoho dalšího.

## 1.2 Android Studio

Android Studio je oficiální integrované vývojové prostředí (IDE) navržené speciálně pro vývoj Android aplikací. Postaveno je na IntelliJ IDEA, poskytuje komplexní sadu nástrojů pro tvorbu, ladění a nasazování mobilních aplikací pro platformu Android. S výhodami jako jsou emulátory zařízení a integrovanou podporou pro Android SDK, Android Studio usnadňuje vývoj pro tuto populární mobilní platformu.

I když Android Studio je primárním nástrojem pro vývoj Android aplikací, existují také alternativy, které mohou vyhovovat určitým potřebám vývojářů. Mezi ně patří Eclipse s pluginem pro Android vývoj, IntelliJ IDEA, Xamarin pro multiplatformní vývoj, a Visual Studio s Android Emulátorem.


**Obrázek 1 Android studio Logo**

*(zdroj: https://www.pngegg.com/en/png-purpz)*

## 1.3 Android

Android OS, vyvinutý společností Google, je operační systém navržený pro mobilní zařízení, jako jsou telefony, tablety a chytré hodinky. Jeho otevřený charakter umožňuje vývojářům vytvářet širokou škálu aplikací pro různá zařízení. Android OS je známý pro svou škálovatelnost, možnost přizpůsobení a otevřenost komunitě.

“Android je otevřený operační systém založený na Linuxu, který byl původně vytvořen pro dotykové obrazovky mobilních telefonů. Nyní je používán na zařízeních po celém světě, od mobilních telefonů a tabletů po televizory” (Open Handset Alliance, 2021).

Android hraje klíčovou roli v ekosystému mobilního vývoje. Je to univerzální platforma, která nabízí vývojářům široký dosah a možnost dosáhnout globálního publika. S rozsáhlou komunitou, kvalitní dokumentací a průběžným vývojem, Android poskytuje stabilní základ pro tvorbu inovativních a funkcemi nabitých mobilních aplikací pro uživatele po celém světě.

**Obrázek 2 Android Logo**
*(zdroj: https://www.pngegg.com/en/png-bzydr)*

## 1.4 Layout

„Grafické prvky v okně (tlačítka, textová pole atd.) v Androidu vkládáme do komponent, kterým se říká layout. Layout grafické prvky shromažďuje (funguje jako kontejner grafických prvků) a především určuje uspořádání prvků v okně a jejich vzájemnou polohu (funguje jako správce rozložení).“ (Publi.cz, 2023)

Layout je klíčovým konceptem v mobilním vývoji, který ovlivňuje uspořádání a vizuální prezentaci uživatelského rozhraní aplikace. Tato podkapitola podrobně rozebírá různé druhy layoutů dostupných v Androidu a jejich roli při vytváření responzivních a esteticky přitažlivých designů. Zároveň se zaměřuje na významné aspekty, jako jsou lineární layout, relativní layout, a constraint layout, poskytující vývojářům flexibilitu při modelování struktury aplikací. Dále probíráme strategie pro výběr vhodného layoutu v závislosti na konkrétních potřebách projektu a optimálních uživatelských zkušenostech. Podkapitola představuje čtenářům klíčové nástroje pro efektivní správu prostoru na obrazovce a vytváření intuitivních uživatelských rozhraní v rámci mobilního prostředí.

# 2 Aktuálnost tématu

V dnešní době rychlého rozvoje mobilních technologií zůstává téma "Definice pozicování komponent uživatelského rozhraní v Android" zásadní a aktuální. S rostoucí rozmanitostí mobilních zařízení, od telefonů po tablety, je klíčové porozumět strategiím pozicování komponent, aby uživatelské rozhraní zůstalo responzivní a optimalizované pro různé obrazovky. Zlepšení uživatelské zkušenosti je stále více spojeno s efektivním designem, a správné pozicování komponent hraje klíčovou roli v tomto úsilí. S technologickým pokrokem a neustálými aktualizacemi v oblasti mobilních platforem vyvstávají nové příležitosti a techniky, které vyžadují neustálou pozornost vývojářů. Toto téma je klíčové nejen pro profesionály v oblasti vývoje aplikací, ale i pro vzdělávací instituce, které chtějí připravovat novou generaci vývojářů na výzvy vytváření moderních uživatelských rozhraní pro mobilní zařízení.

## 2.1 Růst mobilního vývoje

Růst mobilního vývoje je v posledních letech nesmírně výrazný, což reflektuje neustále rostoucí poptávku po inovativních mobilních aplikacích. S širokým rozšířením chytrých telefonů a tabletů se staly mobilní aplikace klíčovým prvkem digitálního života, a tím pádem se vývojářské komunity stále více zaměřují na tuto dynamickou oblast. Rozmanitost dostupných platforem, zejména Android a iOS, nabízí vývojářům širokou paletu možností pro tvorbu aplikací, ať už pro potřeby podniků, zábavy nebo produktivity. Růst mobilního vývoje podporují také pokroky v technologii, zejména v oblasti umělé inteligence, rozšířené reality a internetu věcí, což vede k vytváření stále sofistikovanějších a interaktivnějších mobilních řešení. Tento dynamický trend zdůrazňuje nejen stále rostoucí důležitost mobilního vývoje ve světě digitální transformace, ale také potřebu neustálého vzdělávání a adaptace vývojářských dovedností, aby bylo možné udržet krok s rychlým tempem inovací v tomto odvětví.

**Tabulka 1 - Počet aplikací v obchodě Play 2009-2023**

| Rok | Počet aplikací (mil.) | Rok | Počet aplikací (mil.) |
|---|---|---|---|
| 2009 | 0.016 | 2017 | 3.5 |
| 2010 | 0.1 | 2018 | 2.6 |
| 2011 | 0.4 | 2019 | 2.8 |
| 2012 | 0.7 | 2020 | 2.95 |
| 2013 | 1 | 2021 | 2.605 |
| 2014 | 1.4 | 2022 | 2.694 |
| 2015 | 1.8 | 2023 | 3.718 |
| 2016 | 2.6 | | |

*Zdroj: www.bankmycell.com*

## 2.2 Různorodost zařízení

Různorodost zařízení v dnešním mobilním prostředí přináší vývojářům řadu výzev i příležitostí. S rozsáhlým sortimentem chytrých telefonů a tabletů od různých výrobců se setkáváme s odlišnými parametry výkonu, velikostí obrazovek, ovládacími mechanismy a operačními systémy. Tato heterogenita vyžaduje, aby vývojáři byli schopni vytvářet aplikace, které nejen plně využívají výkon moderních zařízení s vysokým rozlišením obrazovek, ale zároveň poskytují kvalitní uživatelský zážitek na zařízeních s odlišnými technickými specifikacemi.

Rozdíly v ovládacích mechanismech, jako jsou dotykové obrazovky, gesta nebo pera, vyžadují flexibilitu v návrhu uživatelských rozhraní, aby byly intuitivní a pohodlné pro všechny uživatele. Velikost obrazovek zase ovlivňuje vizuální prezentaci obsahu a layouty aplikací, což klade důraz na responzivní design a adaptabilitu.

Tato různorodost přináší pro vývojáře inspirativní výzvy, jak vytvářet univerzální aplikace, které si poradí s různými technickými specifikacemi, a přesto poskytují konzistentní a poutavý uživatelský zážitek napříč širokým spektrem mobilních zařízení.

## 2.3 Uživatelská zkušenost a konkurence

Uživatelská zkušenost je velice výrazně ovlivněna správným pozicováním komponent v uživatelském rozhraní. Stává se klíčovým faktorem pro dosažení konkurenční výhody v oblasti mobilního vývoje. Precizní a esteticky přitažlivé rozmístění prvků nejenže přispívá k vizuálnímu dojmu, ale také vytváří plynulý a intuitivní interakční tok pro uživatele. V dnešní konkurenční době je mnoho mobilních aplikací kde uživatelé očekávají jednoduchost a efektivitu, správně umístěné komponenty mohou vytvořit pozitivní uživatelský zážitek, což v konečném důsledku poskytuje výhodu nad konkurencí. Umění efektivně pozicovat prvky v uživatelském rozhraní se stává strategickým nástrojem pro vývojáře, kteří chtějí nejen splnit očekávání uživatelů, ale i vytvořit diferencovaný produkt schopný vyniknout ve stále se rozvíjejícím trhu mobilních aplikací.

**Aktuální stav poznání:** V dostupné literatuře není tématu pozicování věnována dostatečná pozornost. Uveďme několik případů, v knize (Vávrů a Ujbányai, 2013) jsou pouze představeny některé varianty layoutu bez hlubšího vysvětlení použití. Na webových stránkách zabývající se tvorbou mobilních aplikací není taktéž mnoho důkladných příkladů. Častou ukázkou použití layoutů je pouze základní lineární layout s horizontální formou nebo vertikální formou pozicování komponent. Pro vytváření atraktivních a dnes používaných aplikací není tato forma postačující. Lineární layout neumožňuje pokročilé pozicování komponent a omezuje také ve velikost komponenty při využití rozměru layoutu (viz wrap_parent). Využití absolutní velikosti má značné omezení pro responsitivitu[1] aplikace. Existují ale knihy (zdroje, které se tomuto tématu věnují podrobně. (GeeksforGeeks A Complete Guide to Learn XML for Android App Development, 2019)

[1] Responzitivita je vlastnost UI, která zajišťuje optimální zobrazení na různých zařízeních a obrazovkách různých velikostí nebo rozlišení.

# 3 Layouty v Androidu

Layouty v Androidu představují základní stavební bloky pro organizaci a uspořádání uživatelského rozhraní aplikací. Android nabízí různé druhy layoutů, které umožňují vývojářům flexibilně a efektivně definovat strukturu obrazovek. Zde jsou základní layouty používané v Androidu.

## 3.1 Linear layout

Linear layout v Androidu je jednoduchý a často používaný layout, který uspořádá své prvky nebo widgety buď horizontálně nebo vertikálně v rámci jednoho řádku nebo sloupce. Tato jednoduchá struktura umožňuje snadné uspořádání prvků v jednom směru, což je užitečné pro jednoduché rozložení obrazovek.

Využití Linear Layoutu:

- Jednoduché Seřazení Prvků: Lineární layout se často využívá pro jednoduché seřazení prvků ve vertikálním nebo horizontálním směru, což umožňuje jednoduché uspořádání uživatelského rozhraní.
- Rozložení Tlačítek nebo Položek Menu: V případech, kdy je potřeba jednoduché rozložení tlačítek nebo položek menu v jednom směru, lze lineární layout využít pro přehledné uspořádání prvků.
- Nevýhodou layout je přizpůsobení velikosti pozicované komponenty velikosti layoutu.

Základní Vlastnosti a Parametry:

- Orientation: Určuje směr, ve kterém budou prvky uspořádány. Může být nastaveno na "vertical" pro uspořádání ve sloupci nebo "horizontal" pro uspořádání v řádku.
- Layout Weight: Umožňuje určit relativní váhu jednotlivých prvků v lineárním layoutu. Pomocí této vlastnosti lze určit, jaký podíl prostoru má každý prvek zabírat vzhledem k ostatním.
- Gravity: Určuje umístění prvků uvnitř lineárního layoutu. Může být použito k nastavení zarovnání prvků na střed, doleva, doprava apod.

## 3.2 Relative layout

Relative layout v Androidu umožňuje uspořádat prvky relativně k sobě nebo k rodičovskému kontejneru, což poskytuje vývojářům flexibilitu při návrhu uživatelského rozhraní. Prvky v relative layoutu jsou pozicionovány na základě vzájemných vztahů a určené hierarchie, což usnadňuje vytváření komplexnějších obrazovek.

Využití Relativního Layoutu:

- Dynamické Rozložení Prvků: Relativní layout je vhodný pro situace, kdy je potřeba dynamicky měnit uspořádání prvků v závislosti na různých podmínkách nebo interakcích.
- Závislosti Mezi Prvky: Tento layout umožňuje definovat závislosti mezi prvky, což znamená, že pozice jednoho prvku může být určena vztahem k pozici jiného prvku.

Základní Vlastnosti a Parametry:

- Align Parent: Pomocí této vlastnosti lze určit, jakým způsobem bude prvek zarovnán vzhledem k rodičovskému kontejneru.
- Below/Above/ToRightOf/ToLeftOf: Tyto vlastnosti umožňují určit pozici prvku relativně k jinému prvku.
- Align With Parent: Nastavení zarovnání prvku vzhledem k hranici rodičovského kontejneru.

## 3.3 Constraint layout

Constraint Layout je v Androidu velice silný layout, který kombinuje výhody lineárního a relativního layoutu, což umožňuje vývojářům jednoduše a efektivně definovat složitější struktury uživatelských rozhraní. Tento layout využívá omezující vztahy (constraints), které určují polohu a velikost prvků vzhledem k ostatním prvkům nebo hranicím rodičovského kontejneru.

Využití Constraint Layoutu:

- Responzivní Design: Constraint Layout je ideální pro responzivní design, kde je potřeba vytvářet uživatelská rozhraní, která se automaticky přizpůsobují různým velikostem obrazovek a orientacím.
- Komplexní Rozložení: Tento layout je vhodný pro situace, kdy je potřeba vytvářet komplexní a detailnější rozložení, kde prvek může mít závislosti na více ostatních prvcích.

Základní Vlastnosti a Parametry:

- Omezení (Constraints): Definují vztahy mezi prvky, jako jsou umístění, zarovnání a velikost. Můžete například určit, že prvek bude umístěn vlevo od jiného prvku a zároveň bude zarovnán s horním okrajem.
- Chain (Řetězec): Pro umístění více prvků v řadě lze použít chain vlastnosti, které umožňují definovat vztahy mezi prvky ve formě řetězce.
- Guidelines (Orientační Čáry): Guidelines jsou neviditelné orientační čáry, které umožňují snadné umístění prvků vzhledem k relativním polohám.

## 3.4 Table layout

Table layout v Androidu je layout, který usnadňuje organizaci prvků do tabulkové struktury s řádky a sloupci. Každý řádek v TableLayout může obsahovat několik prvků (View), které jsou rozděleny do buněk. Každý TableLayout může obsahovat libovolný počet prvků TableRow (řádky buňky).

Využití Table Layoutu:

- Umožňuje snadné vytváření tabulkových struktur v uživatelském rozhraní, což je užitečné při organizaci dat do systematické formy.

Základní Vlastnosti a Parametry:

- shrinkColumns: Tato vlastnost definuje, které sloupce mají být zmenšeny, pokud je to nutné pro přizpůsobení šířce tabulky.
- stretchColumns: Tato vlastnost určuje, které sloupce mají být roztahovány, aby zaplnily dostupný prostor.
- layout_span: Tato vlastnost specifikuje, kolik sloupců nebo řádků by měl prvek zabírat.

## 3.5 Další layouty (Frame layout, Grid layout, Coordinator layout)

V této kapitole jsou představeny další typy layoutů, které rozšiřují základní sadu layoutů. Díky vlastnostem a možnostem rozšiřují tyto layouty oblast použití.

### 3.5.1 Popis layoutů

Frame layout

Frame layout je jednoduchý layout, který slouží k umístění jednoho prvku (nebo několika prvků) na obrazovku. Jeho základní vlastnost spočívá v tom, že jednotlivé prvky jsou překrývány, přičemž poslední přidaný prvek je vždy nahoře. Tato charakteristika je užitečná například pro zobrazení jednoho aktuálního prvku na vrcholu, jako například fragmentu nebo obrázku.

Grid layout

Grid layout umožňuje vytvářet mřížku, ve které jsou prvky uspořádány do řádků a sloupců. Každý prvek může zabírat jednu nebo více buněk mřížky. Tato flexibilita je užitečná pro vytváření responzivních rozložení, která se přizpůsobují různým velikostem obrazovek.

Coordinator layout

Coordinator layout je rozšířením frame layout a používá se ke koordinaci interakcí mezi různými prvky v uživatelském rozhraní. Poskytuje sofistikovaný způsob, jak spravovat animace, pohyby a chování prvků. Je často používán s AppBarLayout a CollapsingToolbarLayout pro implementaci efektních rozložení s efektem "skládání" (collapsing) nahoře na obrazovce.

### 3.5.2 Základní Vlastnosti a Parametry

FrameLayout

- Žádné specifické vlastnosti, ale je často používán pro jednoduchá rozložení s jedním hlavním prvkem.

GridLayout

- android:rowCount a android:columnCount určují počet řádků a sloupců v mřížce.
- layout_row a layout_column určují pozici prvku v rámci mřížky.
- layout_gravity určuje zarovnání prvku uvnitř své mřížkové buňky.

CoordinatorLayout

- Využívá speciálních chování (Behaviors) pro definici interakcí mezi prvky.
- layout_behavior určuje chování prvků.

## 3.6 Kombinace Layoutů

Kombinace layoutů v Androidu se často používají k dosažení složitějších a responzivních uživatelských rozhraní. Kombinování různých typů layoutů umožňuje vývojářům dosáhnout specifického uspořádání a chování prvků na obrazovce.

### 3.6.1 Proč kombinovat layouty

Kombinování poskytuje vývojářům flexibilitu a efektivitu při tvorbě uživatelských rozhraní. Tato praxe umožňuje lepší přizpůsobení rozložení podle konkrétních požadavků a optimalizaci pro různé části rozhraní. Různé layouty jsou často vhodné pro specifické účely, což umožňuje vývojářům optimalizovat výkon a čitelnost kódu. (itnetwork.cz 2023)

Kombinace layoutů také podporuje rychlou adaptaci na různá zařízení s rozdílnými velikostmi obrazovek a rozlišeními, což je klíčové pro poskytování optimální uživatelské zkušenosti. Tímto způsobem lze skládat složitější vizuální prvky a podporovat různé designové principy, například Material Design. (itnetwork.cz 2023)

Z hlediska vývojářské modularity je kombinování užitečné pro udržování kódu strukturovaného a čitelného. Každý layout může být samostatným modulem odpovědným za specifickou část rozhraní. Celkově tedy kombinace layoutů představuje klíčový prvek pro efektivní a moderní vývoj uživatelských rozhraní v prostředí Androidu.

### 3.6.2 Příklady kombinací layoutů

Frame layout s child layouts (frame layout s potomky):

- Používá se k umístění jednoho nebo více prvků na obrazovce. Potomci Frame layout mohou být jiné layouty, jako například linear layout nebo relative layout.

ScrollView s linear layout/relative layout:

- Tato kombinace se často používá, když je potřeba vytvořit posuvný seznam nebo oblast, která obsahuje mnoho prvků.

Relative layout s nested layouts:

- Relative layout je často používán jako nadřazený layout, který obsahuje další vnořené layouty. Toto uspořádání umožňuje flexibilní umístění prvků vzhledem k sobě.

Kombinace Constraint layout s ostatními layouty:

- Constraint layout může být použit jako nadřazený layout pro koordinaci složitých vzájemných vztahů mezi prvky. Může obsahovat další layouty nebo prvky.

# 4 Historie a vývoj layoutů pro Android

Layouty pro android se neustále mění a v průběhu historie prošli určitými zásadními změnami a vylepšeními. Jedním z hlavních prvků, proč se layouty mění a přizpůsobují jsou zvětšující se displeje mobilních zařízení, různorodost zařízení a rostoucí nároky na responzivní design. Vzhledem k tomu, že mobilní zařízení nabývají různé tvary, velikosti a rozlišení, vývojáři se snaží vytvářet layouty, které optimalizují uživatelskou zkušenost na všech typech obrazovek. Toto vyžaduje neustálý vývoj a zdokonalování layoutů, aby byla zajištěna konzistentní a efektivní prezentace obsahu na zařízeních s odlišnými charakteristikami.

## 4.1 Historický vývoj Android layoutů

Před Androidem 1.5 (Cupcake):

- První verze Androidu neměla příliš sofistikované nástroje pro práci s layouty. Vývojáři často vytvářeli uživatelská rozhraní pomocí XML kódu a Java kódu.

Android 1.5 (Cupcake) – Relative Layout:

- S verzí Cupcake byl do Androidu přidán relative layout, který umožňoval umisťovat prvky relativně k sobě. To přineslo větší flexibilitu v uspořádání prvků.

Android 2.2 (Froyo) – Frame layout a linear layout:

- Dalšími verzemi Androidu byly přidány layouty jako frame layout pro jednoduché překrývání prvků a linear layout pro jednoduchá lineární rozložení.

Android 3.0 (Honeycomb) – Grid layout:

- S Honeycomb byl představen grid layout, který umožňoval komplexní organizaci prvků do mřížky. To bylo vhodné pro tablety a zařízení s větší obrazovkou.

Android 4.0 (Ice Cream Sandwich) – Relative layout v podobě constraint layout:

- S Ice Cream Sandwich byl rozšířen relative layout a byl představen nový layout s názvem constraint layout, který poskytoval robustní a flexibilní možnosti definování vzájemných vztahů mezi prvky.

Android 5.0 (Lollipop) - Material design a card view:

- S příchodem Material design byly do Androidu zavedeny nové designové principy. Layouty, jako card view, byly přidány pro vytváření karet v uživatelském rozhraní.

Android 6.0 (Marshmallow) – Coordinator layout:

- Byl představen Coordinator layout, který umožňuje koordinaci pohybů a interakcí mezi prvky v uživatelském rozhraní.

Android 8.0 (Oreo) - Adaptive Icons a XML Layouts:

- S Oreo byly představeny adaptivní ikony a vývojáři získali větší kontrolu nad tvorbou uživatelských rozhraní pomocí XML layoutů.

Android 9.0 (Pie) – Constraint layout 2.0:

- Constraint layout získal aktualizaci na verzi 2.0, což přineslo ještě více funkcí a optimalizací pro tvorbu responzivních rozložení.

Android 10, 11, 12, 13 a 14 - Kontinuální vývoj a vylepšení:

- Poslední verze Androidu pokračují v kontinuálním vývoji layoutů a uživatelského rozhraní s důrazem na vylepšení výkonu, responzivity a možností designu.

(Google Developers, 2023)

## 4.2 Trendy v návrhu uživatelského rozhraní

Aktuální trendy v návrhu uživatelského rozhraní (UI) se neustále vyvíjejí v souladu s technologickým pokrokem a měnícími se uživatelskými očekáváními. V současné době jsou designéři a vývojáři zaměřeni na několik klíčových prvků, které ovlivňují vzhled a funkce mobilních aplikací. (Advisio.cz 2022)

Neomorfismus a Soft UI jsou trendy, které zdůrazňují vizuální dojem 3D objektů a jemné stíny, vytvářející dojem hmatového povrchu. Minimalismus a jednoduchý design zůstávají oblíbené, s důrazem na čistotu a přehlednost. Responzivní design je klíčovým faktorem, jak vzhledem k různým velikostem displejů, tak i rozmanitosti zařízení. (Advisio.cz 2022)

Dynamické pohyby a animace hrají stále větší roli ve zvýraznění interakce a přidávání intuitivity ovládání. Stínování a barevné průběhy jsou využívány pro vytvoření hloubky a estetického zájmu. Temný režim nebo "Dark Mode" získal na popularitě pro své energetické úspory a pohodlný noční režim používání.

Celkově lze konstatovat, že trendy v návrhu UI reflektují snahu vytvářet moderní, esteticky příjemné a uživatelsky efektivní mobilní aplikace, které splňují současné potřeby uživatelů a využívají nejnovější technologické možnosti.

# 5 Návrh optimálního uživatelského rozhraní

Optimální uživatelské rozhraní je koncipováno s důrazem na několik klíčových prvků, které mají zajišťovat co nejlepší uživatelskou zkušenost. Základními principy jsou jasnost a přehlednost, které umožňují uživatelům snadnou orientaci a rychlé porozumění obsahu. Konzistence, jak ve vizuálním stylu, tak ve struktuře, přispívá k celkovému komfortu a snadnějšímu používání.

## 5.1 Jak dosáhnout optimálního uspořádání komponent

Ovládací prvky by měly být navrženy tak, aby byly přístupné a pochopitelné bez složitého návodu. Důležitá je také rychlá odezva na interakce uživatele, což přispívá k plynulému průběhu práce.

Přizpůsobitelnost a responzivita zajišťují optimální zážitek na různých typech zařízení a obrazovek.

Vizuální prvky, včetně barev, typografie a grafiky, by měly být esteticky přitažlivé a příjemné pro uživatele. Přirozený tok práce je podporován minimalizací překážek a vytvořením prostředí, které respektuje uživatelský záměr.

Dostatek zpětné vazby je nezbytný pro informování uživatele o dění a udržení transparentnosti. Optimální UI zohledňuje i výkon, s důrazem na rychlou odezvu a minimalizaci čekacích dob.

Celkově je optimální uživatelské rozhraní navrženo tak, aby co nejlépe splňovalo potřeby uživatelů a poskytovalo jim pohodlný, intuitivní a efektivní prostředek pro interakci s aplikací.

## 5.2 Testování a zpětná vazba od uživatelů

Testování uživatelského rozhraní a uživatelské zkušenosti je klíčovým krokem ve vývoji aplikací. Umožňuje vývojářům a designérům získat cennou zpětnou vazbu od uživatelů a zajistit tak optimální uživatelské prostředí.

Usability Testing (Testování Použitelnosti):

Tato metoda zkoumá, jak snadno a efektivně mohou uživatelé interagovat s aplikací. Testování navigace, hledání informací a provádění úkonů pomáhá identifikovat případné překážky v používání.

Performance Testing (Testování Výkonu):

Zkoumá rychlost a efektivitu UI během různých podmínek. To zahrnuje testování odezvy, rychlosti načítání a celkové výkonnosti aplikace.

Accessibility Testing (Testování Dostupnosti):

Zajišťuje, že UI a UX jsou přístupné pro uživatele se speciálními potřebami. Testuje se například čitelnost pro uživatele se zrakovým postižením nebo ovladatelnost pro uživatele s omezenou pohyblivostí.

# 6 Možnosti budoucího vývoje

Budoucnost layoutů v oblasti uživatelského rozhraní se pravděpodobně bude vyvíjet v souladu s technologickým pokrokem, změnami ve způsobu interakce uživatelů s digitálními zařízeními a novými designovými trendy.

## 6.1 Nové technologie v mobilním vývoji

Rozšířená a Virtuální Realita (AR/VR):

- Technologie rozšířené a virtuální reality otevírají nové dimenze pro design layoutů. Vytváření prostoru a interakce v trojrozměrném prostoru přináší nové výzvy a příležitosti pro návrháře.

Umělá Inteligence (AI):

- Umělá inteligence může být využívána pro personalizaci layoutů na základě chování uživatelů, predikci jejich potřeb a automatizaci designových procesů.

Rychlý Vývoj Aplikací (Low-Code/No-Code):

- Nástroje pro rychlý vývoj aplikací umožňují návrhářům a vývojářům rychle vytvářet a upravovat layouty bez hlubších znalostí programování.

Interaktivní Grafika a Animace:

- Moderní grafické technologie a animace přinášejí možnost vytvářet pohyblivé, interaktivní prvky, které obohacují uživatelský zážitek.

Dostupnost a Inkluzivita:

- Technologie zaměřené na dostupnost a inkluzivitu, jako například čtečky obrazovky a hlasové ovládání, umožňují vytvářet layouty, které jsou přístupné pro co největší počet uživatelů.

Adaptivní Design a Responzivní Layouty:

- Technologie umožňující adaptivní design a responzivní layouty zajistí, že uživatelské rozhraní bude optimalizováno pro různé typy zařízení a obrazovek.

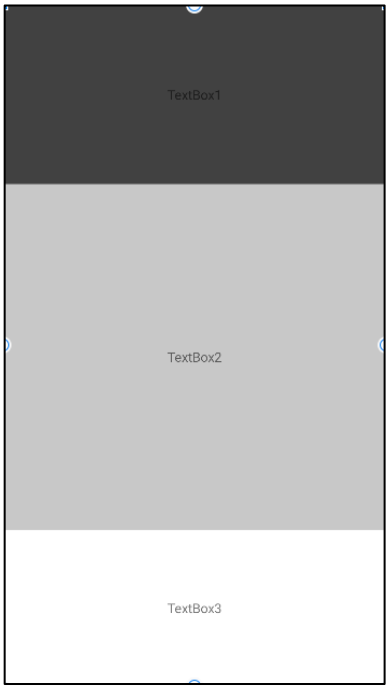
# 7 Tvorba layoutů

V první kapitole této praktické části se budeme podrobně věnovat procesu tvorby layoutů pro mobilní aplikace v prostředí Android. Tvorba layoutů je klíčovým prvkem designového procesu, který ovlivňuje celkový vzhled a strukturu uživatelského rozhraní. V této kapitole se zaměříme na základní prvky a koncepty spojené s tvorbou layoutů, včetně jednotlivých typů layoutů, které jsou k dispozici v Android Studio.

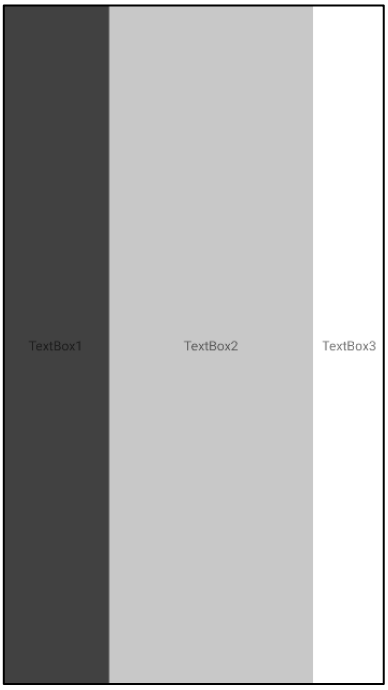
## 7.1 Linear layout

Lineární layout představuje skupinu prvků, která uspořádává všechny své potomky do jedné řady, a to dvěma možnými směry, horizontálně a nebo vertikálně. Směr se určuje pomocí atributu android:orientation.

Vertikální směr si lze představit jako tabulku která má pouze jeden sloupec a libovolný počet řádků. Horizontální směr je přesným opakem tudíž by měl pouze jeden řádek a prvky by se skládali do jednotlivých sloupců.




**Obrázek 3 Lineární layout – Vertikální**
*(zdroj: Vlastní)*




**Obrázek 4 Lineární layout – Horizontální**
*(zdroj: Vlastní)*

Rozdíly mezi horizontálním a vertikálním uspořádáním jsou po designové stránce pouze ve směru, v xml kódu najdeme minimum rozdílů, a to jen u jednotlivých prvků kde budou rozdíly v nastavení jejich šířky a výšky. Ukázky kódu jsou v příloze A.1. pro vertikální a A.2. pro horizontální rozložení.

Atributy pro lineárním layout

**android:orientation:**

- Tento atribut určuje orientaci layoutu a může nabývat hodnot **horizontal** nebo **vertical**. Podle této hodnoty jsou prvky umístěny vedle sebe nebo pod sebou.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
</LinearLayout>
<!-- Prvky budou uspořádány svisle -->
```

**Ukázka 1 Vertikální uspořádání**

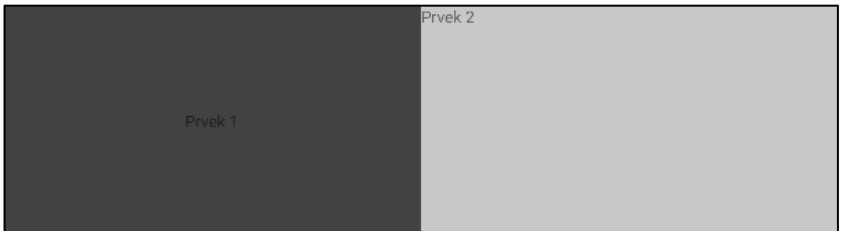
**android:gravity a android:layout_gravity:**

- Tyto atributy ovlivňují zarovnání textu nebo obrazu uvnitř každého prvku (**android:gravity**) a umístění samotného layoutu v rámci jeho rodiče (**android:layout_gravity**).

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Střední text" />
```

**Ukázka 2 Zarovnání textu**




**Obrázek 5 Použití gravitace na střed**
*(zdroj: Vlastní)*

**android:layout_weight:**

- Tento atribut umožňuje definovat, jaký podíl dostupného místa v layoutu má každý prvek. Používá se ve spojení s **android:layout_width** nebo **android:layout_height** nastaveným na 0dp. Například pokud máme dva prvky a první má nastavenou váhu 1 a druhý váhu 2 bude prostor rozdělen 1:2.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Prvek 1" />

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:text="Prvek 2" />

</LinearLayout>
```

**Ukázka 3 Rozdělení dle váhy**


**Obrázek 6 Rozložení pomocí váhy 1:2**
*(zdroj: Vlastní)*


**Obrázek 7 Rozložení pomocí váhy 1:3**
*(zdroj: Vlastní)*

**Použití android:layout_weight v kombinaci s prvkem s pevnou šířkou**

- Prvky s nastavenou váhou si rozdělí zbylé místo v poměru dle váhy, prvek s pevnou velikostí to neovlivní.


**Obrázek 8 Kombinace pevné šířky a váhy 1:1**
*(zdroj: Vlastní)*

XML kód pro obrázek 8 je v příloze A.3.

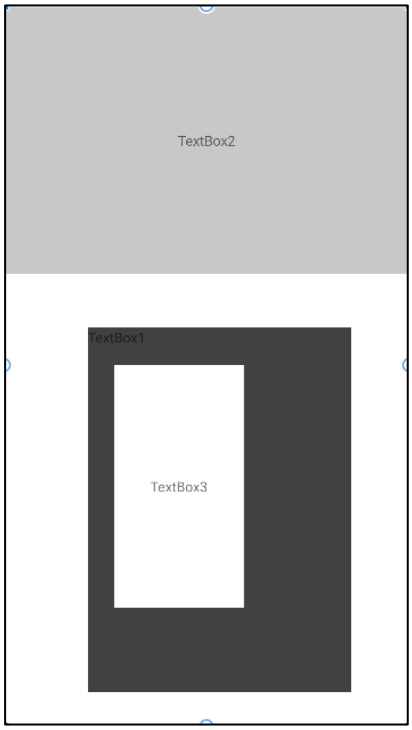
**Použití android:layout_weight v kombinaci s prvkem s šířkou nastavenou na "match_parent"**

- Tato kombinace je nepoužitelná protože prvek s šířkou match_parent zabere celou šíři rodičovského prvku tudíž nezbyde žádný prostor pro prvky s váhou.

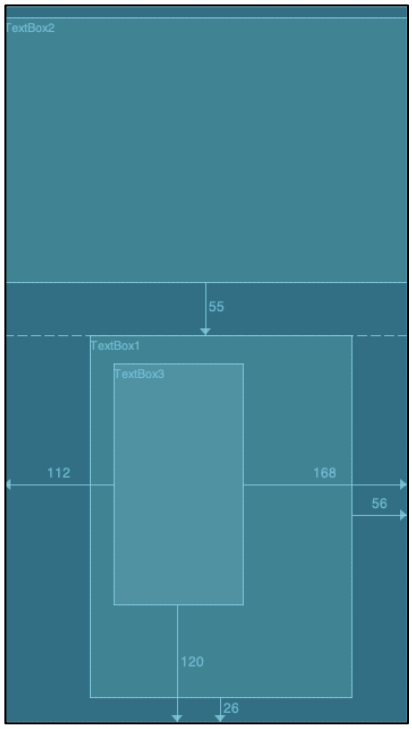
## 7.2 Relative layout

Relative layout je skupina prvků, který vytváří náhled na základě vzájemných pozic prvků. Pozice každého prvku lze určit relativně k ostatním prvkům (nad, pod, vlevo, vpravo) nebo relativně k nadřazené oblasti (k levému spodnímu okraji atp.).

Tento layout je velice výkonným nástrojem při složitějším návrhu, protože dokáže eliminovat vnořené skupiny prvků, udržet hierarchii prvků v návrhu a zvýšit výkon. Je vhodný jako náhrada v situaci, kdy je použito více vnořených lineárních layoutů.




**Obrázek 9 Relative layout**
*(zdroj: Vlastní)*




**Obrázek 10 Relative layout – kóty**
*(zdroj: Vlastní)*

Na obrázcích 5 a 6 můžeme vidět použití relativní pozice k rodičovskému prvku a to v případě TextBox1 a TextBox3 zároveň můžeme vidět, že TextBox3 je zobrazen před TextBox1 to je způsobeno hierarchií při zápisu. TextBox 2 je zarovnán na šířku rodičovského prvku a na výšku k prvku TextBox1 kde za jakýchkoliv podmínek bude vzdálenost 55px. Vzdálenost od jednotlivých prvků se dá zapisovat i v % což je užitečné, když chceme být flexibilní a přizpůsobit se různým velikostem obrazovek. Celá ukázka je v příloze číslo A.4.

```
<TextView
          android:id="@+id/textView"
          android:layout_width="267dp"
          android:layout_height="369dp"
          android:layout_alignParentEnd="true"
          android:layout_alignParentBottom="true"
          android:layout_marginEnd="59dp"
          android:layout_marginBottom="35dp"
          android:background="@color/cardview_dark_background"
          android:text="TextBox1"
          />
```

**Ukázka 4 Relative layout**

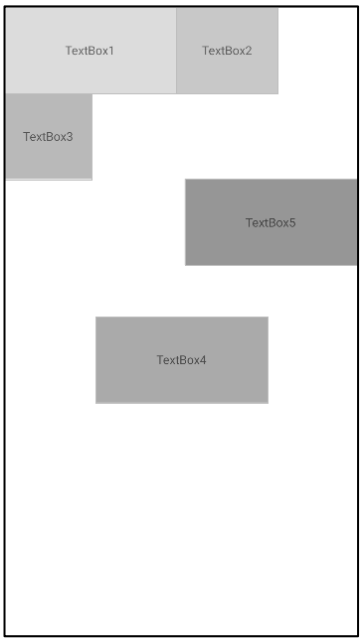
Specifické atributy pro relative layout

- **layout_alignParentTop:** Zarovná prvek na horní okraj rodičovského relative layout.
- **layout_alignParentBottom:** Zarovná prvek na dolní okraj rodičovského relative layout.
- **layout_alignParentStart:** Zarovná prvek na levý okraj rodičovského relative layout.
- **layout_alignParentEnd:** Zarovná prvek na pravý okraj rodičovského relative layout.
- **layout_above:** Umístí prvek nad určený prvek.
- **layout_below:** Umístí prvek pod určený prvek.
- **layout_toStartOf:** Umístí prvek vlevo od určeného prvku.
- **layout_toEndOf:** Umístí prvek vpravo od určeného prvku.
- **layout_alignTop:** Zarovná horní okraj prvního prvku s horním okrajem druhého prvku.
- **layout_alignBottom:** Zarovná dolní okraj prvního prvku s dolním okrajem druhého prvku.

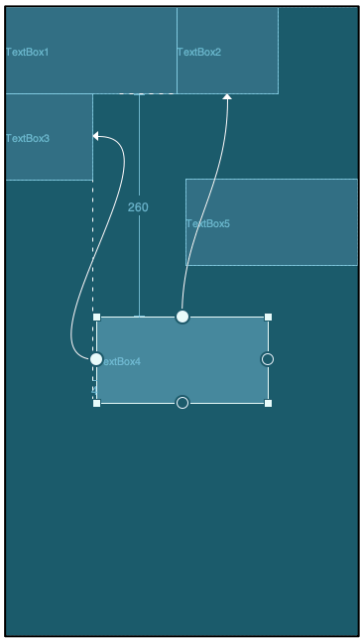
## 7.3 Constraint layout

Constraint layout je typ layoutu v Androidu, který umožňuje vytvářet flexibilní a komplexní uživatelská rozhraní. Tento layout je založen na konceptu omezujících vztahů (constraints) mezi prvky, což umožňuje definovat jejich umístění a velikost relativně k sobě a k rodičovskému layoutu. Tyto vztahy jsou definovány pomocí horizontálních a vertikálních omezujících linií a mohou určovat, jak jsou prvky zarovnány vůči sobě navzájem nebo k rodičovskému layoutu, jak jsou jejich okraje zarovnány a jaká je jejich relativní velikost. Díky tomu je možné vytvářet složité rozložení, aniž by bylo nutné vnořovat více layoutů.

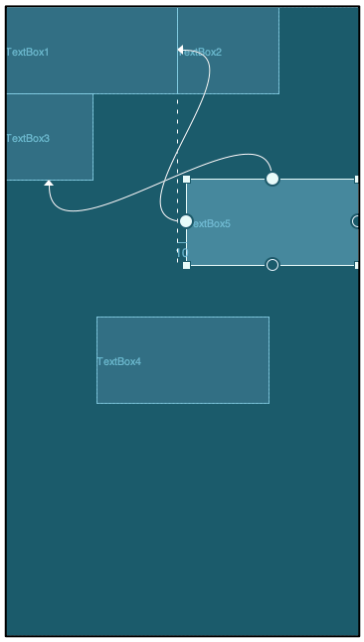
Constraint layout na první pohled připomíná relative layout, ale ve skutečnosti je mnohem komplexnější. Hlavním rozdílem mezi relative a constraint layoutem je přístup k uspořádání prvků, relative layout používá primárně zarovnání na okraje a relativní umístění, zatímco constraint layout umožňuje detailnější kontrolu nad umístěním a velikostí prvků pomocí omezujících vztahů.




**Obrázek 11 Constraint layout**
*(zdroj: Vlastní)*




**Obrázek 12 Constraint vazby 1**
*(zdroj: Vlastní)*




**Obrázek 13 Constraint vazby 2**
*(zdroj: Vlastní)*

TextBox1 je zarovnaný k rodičovskému constraint layoutu na horní straně a levém boku. TextBox2 a 3 je zarovnán jednou stranou ke constraint layoutu a druhou k TextBox1.

TextBox4 a 5 využívají omezeních a to tak, že levá strana TextBox4 nesmí být více vlevo, než je pravá hrana Textbox3 a horní hrana se nesmí dostat níže, než je spodní hrana TextBox2. TextBox 5 je nastaven obdobně, horní hrana se nesmí dostat níže než je spodní hrana TextBoxu3 a nesmí se dostat více vlevo než je pravá strana TextBox1 + 10px. XML kód je k dispozici v příloze A.5.

```
<TextView
      android:id="@+id/textBox1"
      android:layout_width="200dp"
      android:layout_height="100dp"
      android:background="#DDDDDD"
      android:gravity="center"
      android:text="TextBox1"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_marginStart="16dp"
      app:layout_marginTop="16dp"
      />
```

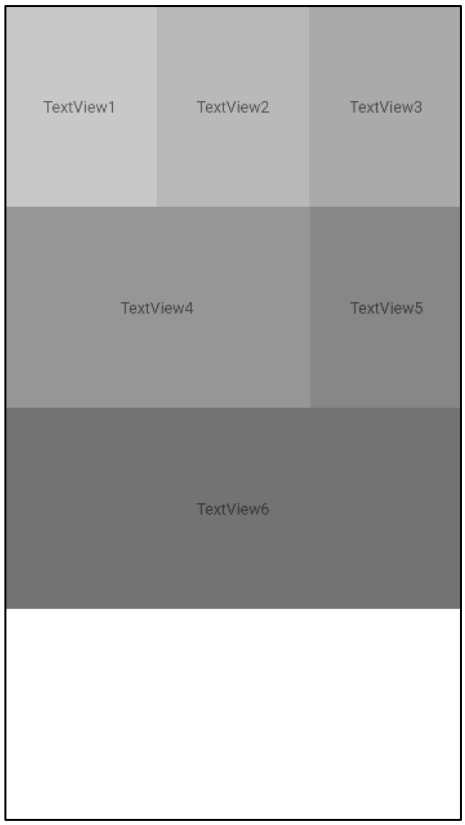
**Ukázka 5 Constraint layout**

Specifické atributy pro constraint layout

- **app:layout_constraintTop_toTopOf**: Určuje, že horní okraj aktuálního prvku je připojen k hornímu okraji jiného prvku.
- **app:layout_constraintStart_toStartOf**: Určuje, že levý (počáteční) okraj aktuálního prvku je připojen k levému (počátečnímu) okraji jiného prvku.
- **app:layout_constraintBottom_toBottomOf**: Určuje, že dolní okraj aktuálního prvku je připojen k dolnímu okraji jiného prvku.
- **app:layout_constraintHorizontal_bias**: Určuje horizontální poměr pozice prvku mezi okraji, kde 0 znamená, že je prvek zarovnán úplně doleva a 1, že je zarovnán úplně doprava.
- **app:layout_constraintVertical_bias**: Určuje vertikální poměr pozice prvku mezi okraji, kde 0 znamená, že je prvek zarovnán úplně nahoru a 1, že je zarovnán úplně dolů.
- **app:layout_constraintDimensionRatio**: Tento atribut umožňuje definovat poměr mezi šířkou a výškou prvku. Například hodnota "3:2" by znamenala, že šířka prvku je 3x větší než jeho výška.
- **app:layout_constraintGuide_begin** a **app:layout_constraintGuide_end**: Tyto atributy umožňují vytvoření pomocných vodících čar, které slouží k zarovnání prvků v layoutu.
- **app:layout_constraintVertical_chainStyle** a **app:layout_constraintHorizontal_chainStyle**: Tyto atributy řídí chování prvků v řetězcích (chains). Například umožňují nastavit, zda mají prvky v řetězci být zarovnány kolem prvního nebo posledního prvku v řetězci.
- **app:layout_constraintVertical_weight** a **app:layout_constraintHorizontal_weight:** Tyto atributy umožňují definovat váhu prvků ve vertikálním nebo horizontálním lineárním řetězci (chain). Váha určuje, jaký podíl dostává prvek z celkové dostupné velikosti.

## 7.4 Table layout

Jeden ze základních layoutů v androidu. Jak již název napovídá používá se k uspořádání uživatelského rozhrání do tabulkového formátu. Table layout obsahuje prvky Table row, které představují jednotlivé řádky v tabulce. Každý TableRow pak může obsahovat několik dalších prvků (např. TextView, Button) reprezentujících buňky v daném řádku.




**Obrázek 14 Table layout**
*(zdroj: Vlastní)*

V tomto kódu (příloha A.6.) jsou prvky v každém řádku rozmístěny vedle sebe s pomocí *layout_weight* atributu, který určuje jejich váhu v rámci řádku. Například, pokud má první prvek váhu 1 a druhý váhu 2, první prvek bude zabírat 1/3 šířky řádku a druhý prvek bude zabírat 2/3 šířky řádku. Tímto způsobem lze jednoduše rozdělit šířku řádku mezi jednotlivé prvky podle potřeby. Šířka se dá nastavit v % nebo dp k tomu slouží atribut *layout_width*, ten je zde nastavený na 0dp protože šířka jednotlivých prvků je určena váhou.

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow>
    <-- První řádek tabulky -->
    </TableRow>

    <TableRow>
    <-- Druhý řádek tabulky -->
    </TableRow>

    <TableRow>
    <-- Třetí řádek tabulky -->
    </TableRow>
</TableLayout>
```

**Ukázka 6 Table layout**

Specifické atributy pro table layout a table row

- **layout_column**: Určuje, ve kterém sloupci bude umístěn prvek.
- **stretchColumns**: Specifikuje, které sloupce budou automaticky roztaženy, aby vyplnily dostupný prostor.
- **shrinkColumns**: Specifikuje, které sloupce budou automaticky smrštěny, pokud je nedostatek prostoru.
- **collapseColumns**: Určuje, které sloupce budou automaticky skryty, pokud je nedostatek prostoru.
- **layout_span**: Určuje, kolik sloupců nebo řádků má prvek zabírat.
- **layout_gravity**: Nastavuje gravitaci (zarovnání) v rámci buňky.
- **layout_columnWeight**: Určuje relativní váhu sloupce vzhledem k ostatním sloupcům.

# 8 Kombinace layoutů

Kombinování layoutů umožňuje vývojářům vytvářet komplexní a flexibilní rozložení. Tím, že kombinují různé druhy layoutů, jako jsou linear layout, Relative layout, constraint layout a další, mohou vývojáři dosáhnout požadovaného vzhledu a chování aplikace. Například mohou využít vertikální nebo horizontální linear layout jako základní kontejner pro uspořádání prvků ve sloupcích nebo řádcích a poté použít relative layout nebo constraint layout k dalšímu jemnému ladění a umístění prvků v rámci těchto kontejnerů. Kombinování layoutů tak poskytuje vývojářům širokou škálu možností při tvorbě uživatelského rozhraní.

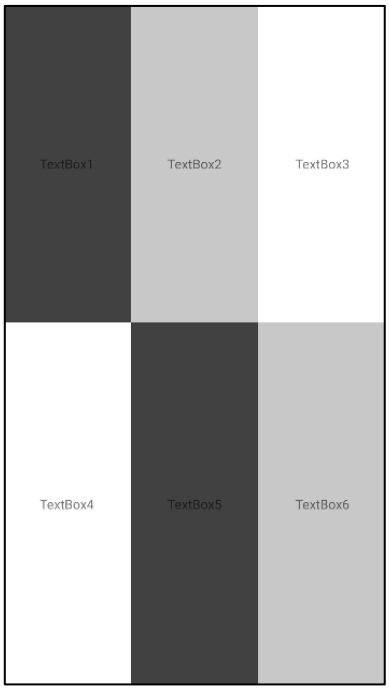
## 8.1 Lineární horizontální a lineární vertikální

Kombinace vertikálního a horizontálního lineárního layoutu je využívána pro vytvoření složitějších uživatelských rozhraní, která vyžadují uspořádání prvků ve více směrech. Vertikální layout může být využit pro uspořádání prvků pod sebou, zatímco horizontální layout může být využit pro uspořádání prvků vedle sebe. Je vhodně ho použít na statické tabulky, které nejsou příliš složité.

Tato kombinace ve výsledku vytvoří obdobu table layoutu. Ovšem pro utřídění dat do tabulek je table layout vhodnější.

Nevýhody oproti table layoutu při tvorbě tabulky.

- **Složitost kódu** – Kombinace layoutů může způsobit větší složitost kódu, zejména pokud tvoříme složitější strukturu tabulky. S TableLayout můžeme výrazně zjednodušit kód pro uspořádání tabulky.
- **Flexibilita** – TableLayout byl navržen speciálně pro uspořádání dat v tabulkové formě, což znamená, že může poskytnout určitou flexibilitu při práci s řádky a sloupci. S kombinací layoutů může být obtížnější dosáhnout některých specifických tabulkových funkcí.
- **Výkon** – Při používání více vrstev layoutů může dojít ke snížení výkonu, zejména pokud je struktura složitá. TableLayout může být optimalizován pro rychlé zobrazení tabulkových dat. *(Android Developers, 2024)*
- **Údržba** – Kombinace layoutů mohou být obtížnější na údržbu a změny, zejména pokud se požadavky na rozložení mění. TableLayout má obvykle přehlednější strukturu pro tabulkové uspořádání.




**Obrázek 15 Kombinace lineárních layoutů**

*(zdroj: Vlastní)*

```
<LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical">

      <LinearLayout
          android:layout_width="match_parent"
          android:layout_height="341dp"
          android:orientation="horizontal"> <!--První řádek-->
      ...
      </LinearLayout>

      <LinearLayout
          android:layout_width="match_parent"
          android:layout_height="390dp"
          android:orientation="horizontal"> <!—Druhý řádek-->
      ...
      </LinearLayout>
  </LinearLayout>
```

**Ukázka 7 Vnoření lineárních layoutů**

Celý kód k obrázku 15 je v příloze číslo A.7. Tato kombinace funguje tak, že první linear layout obalí dva další, které se pak chovají jako řádky v tabulce. TextView prvky se v tomto případě chovají jako buňky v tabulce.

## 8.2 Scroll view, constraint a linear layout

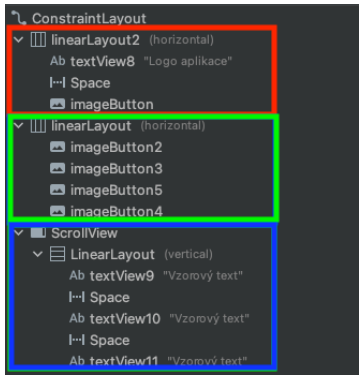
Kombinace constraint layoutu, linear layoutu a scroll view je často používaným přístupem k vytváření flexibilních uživatelských rozhraní v aplikacích pro Android.

ConstraintLayout se často používá jako kořenový prvek layoutu, který umožňuje snadné umístění a zarovnání prvků ve vztahu k sobě a ke zbytku rozložení.

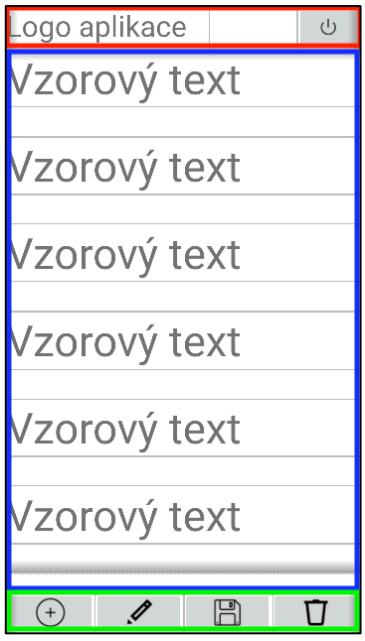
LinearLayout je užitečný pro uspořádání prvků v jednom směru, buď vertikálně nebo horizontálně. Je vhodný pro jednoduché seznamy nebo řádky prvků.

ScrollView umožňuje uživatelům posouvat obsah, který je větší než velikost obrazovky, což je užitečné pro dlouhé seznamy nebo obsah s možností posunu.

Kombinace těchto tří layoutů umožňuje vytvářet složitá a dynamická uživatelská rozhraní, která lze snadno přizpůsobit různým velikostem a orientacím obrazovky.

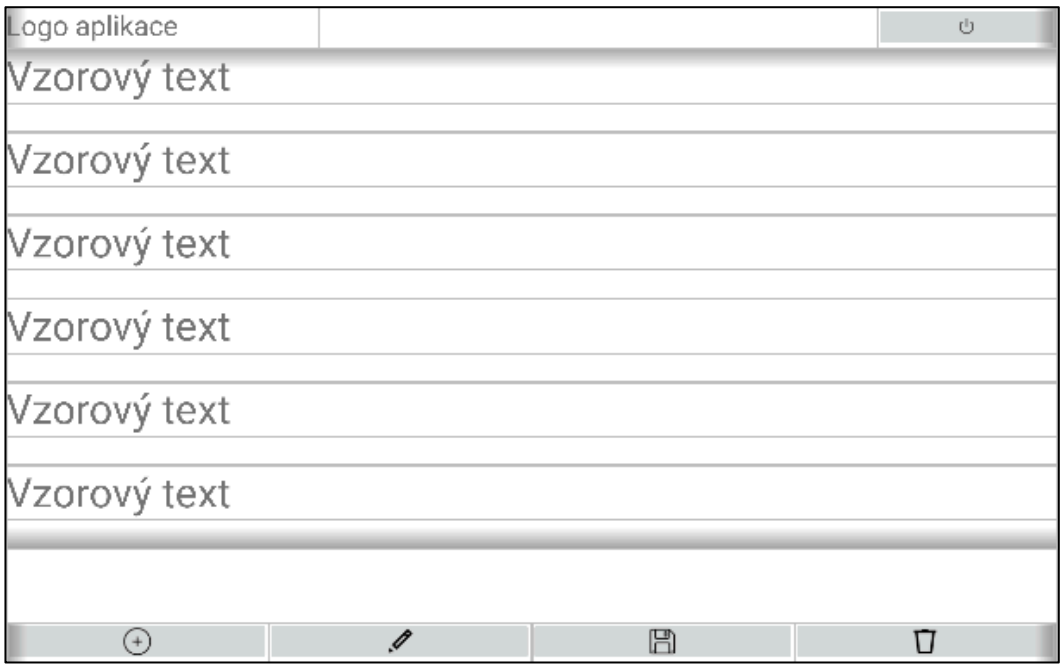



**Obrázek 16 Prvky scroll, linear layoutu**
(zdroj: Vlastní)




**Obrázek 17 Rozložení scroll, linear layoutus**
(zdroj: Vlastní)

- Constraint layout slouží k ukotvení všech prvků a zajišťuje kompatibilitu mezi různými velikostmi displejů. Všechny tři hlavní části jsou pevně ukotveny tak aby šířka byla vždy shodná s obrazovkou zařízení. Tudíž takto postavená aplikace může fungovat i v režimu na šířku. Jediný problém může nastat při použití příliš malého displeje kde by se prvky nebyli schopny vejít vedle sebe. Bylo by vhodné změnit nastavení těchto prvků tak aby neměli pevnou velikost, ale aby byla uvedena v procentech. Při použití pevné výšky se mohou prvky zdát příliš velké nebo naopak malé v závislosti na velikosti displeje.
- První část je tvořena linear layoutem se třemi prvky, tlačítko může být využito například pro vyvolání nastavení aplikace.
- Uprostřed je hlavní obsahová část, zde je reprezentována prvky TextView, v reálném použití může obsahovat data z databáze například příspěvky ze sociální sítě atp.
- Spodní část s tlačítky lze využít jako menu aplikace, popřípadě jako filtry obsahu v hlavní části. Verze pro na šířku by mohla obsahovat mezery mezi tlačítky tak aby kompenzovali větší šířku displeje a tlačítka si zachovali svoji velikost.




**Obrázek 18 Scroll, linear layout – Tablet**
(zdroj: Vlastní)

Kód k obrázku 17 a 18 je v příloze A.8.

# 9 Pokročilé layouty v Androidu

Rozšíření základních znalostí o tvorbě layoutu zahrnuje pokročilé techniky, které jsou velmi relevantní pro moderní vývoj mobilních aplikací. Flexbox layout a Motion layout jsou silné nástroje umožňující vytváření komplexních a responzivních rozložení.

Tyto pokročilé layoutu vyžadují import dalších knihoven bez nich nebudou fungovat.

## 9.1 Flexbox layout

Flexbox je moderní layout, který umožňuje flexibilní a dynamické uspořádání prvků. Inspiruje se CSS Flexboxem používaným při ve webovém vývoji.

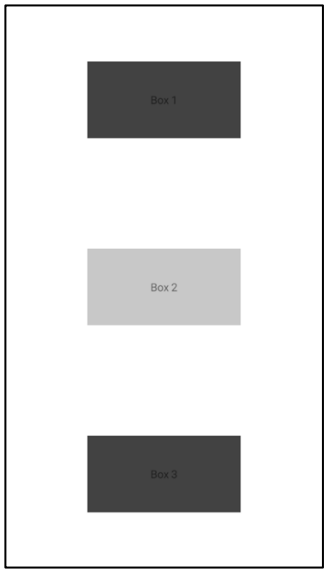
Hlavní vlastnosti:

- **Flex Direction**: Určuje směr, ve kterém jsou položky umístěny v kontejneru.
- **Justify Content**: Určuje zarovnání položek podél hlavní osy.
- **Align Items**: Určuje zarovnání položek podél vedlejší osy.
- **Flex Wrap**: Určuje, zda položky mají být zalamovány do dalších řádků.

Pro použití Flexboxu je nutné vložit do *build.gradle (Module: app)*, bez této implementace a správné synchronizace s projektem nebude tento flexbox funkční.

```
...
dependencies {
    ...
    implementation 'com.google.android.flexbox:flexbox:3.0.0'
    ...
}
```

**Ukázka 8 implementace flexboxu**




**Obrázek 19 Flexbox**
(zdroj: Vlastní)

Tento jednoduchý příklad Flexbox layoutu v Androidu má za cíl demonstrovat základní použití Flexbox layoutu k umístění prvků ve sloupci s rovnoměrným prostorem kolem nich. Celá ukázka je v příloze A.9.

```
<com.google.android.flexbox.FlexboxLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:flexbox="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    flexbox:flexDirection="column"
    flexbox:justifyContent="space_around"
    flexbox:alignItems="center">
...
```

**Ukázka 9 Flexbox**

- **flexDirection**: V tomto případě jsou prvky umístěny ve směru sloupce (column) pro směr řádku (row).
- **justifyContent**: Space_around rovnoměrně rozdělí prvky s rovnoměrným prostorem kolem nich.
- **alignItems**: Hodnota center zarovná prvky na střed.

Flexbox layout poskytuje vývojářům velkou flexibilitu při navrhování uživatelských rozhraní, která jsou dynamická a responzivní. Díky různým možnostem nastavení je možné snadno a rychle vytvářet složitější layouty bez potřeby vnořování více vrstev layoutů, což zjednodušuje správu a zvyšuje výkon aplikací.

## 9.2 ConstraintSet layout

ConstraintSet je součástí Constraint layout v Androidu a slouží k dynamické změně rozložení komponent uživatelského rozhraní během běhu aplikace. Pomocí ConstraintSet lze snadno a efektivně aktualizovat omezení (constraints) pro jednotlivé prvky ve výchozím rozložení, aniž by bylo nutné znovu vytvářet celé UI.

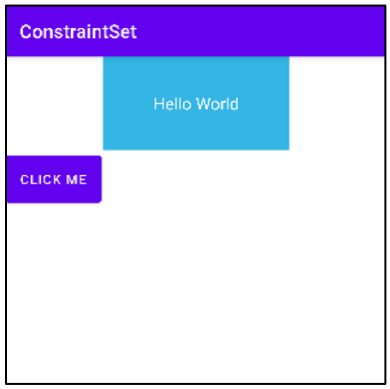
Hlavní vlastnosti ConstraintSet

Flexibilita: Umožňuje dynamické změny rozložení uživatelského rozhraní bez nutnosti reinicializace celého layoutu.

Jednoduchost: Poskytuje jednoduchý způsob, jak aktualizovat omezení (constraints) na základě určitých akcí nebo událostí v aplikaci.

Výkonnost: Pomáhá zlepšit výkonnost aplikace tím, že minimalizuje potřebu vytvářet a znovu vytvářet layouty.


**Obrázek 20 ConstraintSet 1**
(zdroj: Vlastní)




**Obrázek 21 ConstraintSet 2**
(zdroj: Vlastní)

První obrázek nám ukazuje stav před kliknutím na button a druhý obrázek ukazuje stav po kliknutí. Textbox „hello world“ se plynule přesune na novou pozici. Přesun je plynulý a pro uživatele vypadá jako animace. Kód k této layoutu je příloze číslo A.10.

Funkce java kódu:

**Inicializace ConstraintLayout**:

- ConstraintLayout je inicializován pomocí funkce *findViewById*. Tento layout bude použit pro dynamické změny.

**Vytvoření a klonování ConstraintSet**:

- ConstraintSet představuje stav rozložení ConstraintLayout.
- *initialSet.clone(constraintLayout):* Klonuje aktuální stav layoutu a ukládá jej jako počáteční.
- *changedSet.clone*(*constraintLayout):* Klonuje aktuální stav layoutu pro pozdější modifikace.

**Modifikace ConstraintSet**:

- *changedSet.clear(R.id.textView, ConstraintSet.TOP):* Odstraní horní vazbu pro TextView.
- *changedSet.connect(R.id.textView, ConstraintSet.BOTTOM, R.id.button, ConstraintSet.TOP):* Připojuje dolní část TextView k horní části Button.
- *changedSet.connect(R.id.textView, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END):* Připojuje pravou stranu TextView k pravé straně rodičovského layoutu.
- *changedSet.connect(R.id.textView, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START):* Připojuje levou stranu TextView k levé straně rodičovského layoutu.

**Nastavení OnClickListener pro tlačítko**:

- Button je inicializován a je mu přiřazen OnClickListener.
- Při kliknutí na tlačítko se zavolá *TransitionManager.beginDelayedTransition(constraintLayout),* což způsobí animovaný přechod mezi stavy layoutu.
- Pokud je *isChanged* nastaven na true, použije se *initialSet* k návratu do počátečního stavu. Pokud je false, použije se *changedSet* pro aplikaci nového stavu.
- *isChanged* se přepíná mezi true a false při každém kliknutí.

**Specifické příkazy a vlastnosti:**

- **clone():** Klonuje aktuální stav rozložení ConstraintLayout do ConstraintSet.
- **connect():** Nastaví nové omezení mezi dvěma prvky.
- **clear():** Odstraní všechna omezení z určitého prvku nebo specifické kotvy.
- **applyTo()**: Aplikuje všechny změny v ConstraintSet na cílový ConstraintLayout.
- **setMargin():** Nastaví okraj pro specifickou kotvu prvku.

ConstraintSet poskytuje zajímavý způsob, jak dynamicky a efektivně měnit rozložení komponent v aplikacích pro Android. Díky jednoduchým metodám pro nastavení a změnu omezení (constraints) je možné snadno implementovat složité změny layoutu a animace, čímž se zvyšuje uživatelský zážitek a optimalizuje výkon aplikace.

## 9.3 Motion layout

Motion layout je způsob použití ConstraintLayout a slouží k vytváření bohatých animací a interakcí v uživatelském rozhraní. Integruje schopnosti ConstraintSet pro manipulaci s rozložením a přidává možnosti pro animace mezi různými stavy layoutu.

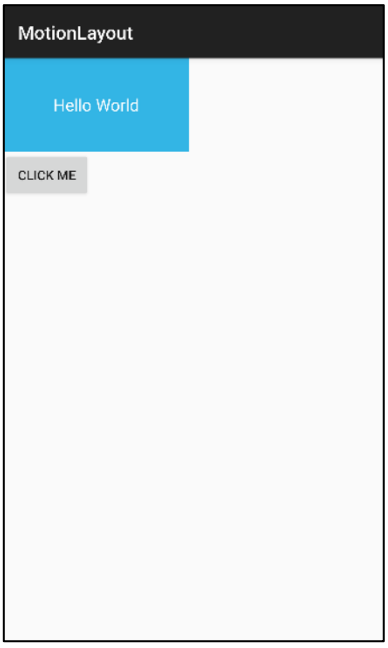
Hlavní vlastnosti Motion layoutu:

- Flexibilita: Umožňuje vytvářet složité animace a přechody mezi stavy.
- Jednoduchost: Používá XML soubory k definování animací, což usnadňuje práci s animacemi pro vývojáře.
- Integrace: Je plně integrovaný do ConstraintLayout, což umožňuje snadnou práci s existujícími layouty.

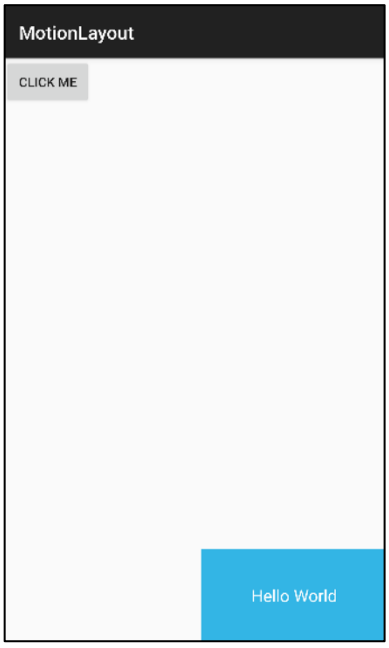
**Struktura Motion layout**

Motion layout používá tři hlavní komponenty:

- MotionScene: XML soubor, který definuje různé stavy a přechody mezi nimi.
- State: Definice jednotlivých stavů layoutu pomocí ConstraintSet.
- Transition: Definice přechodů mezi stavy, včetně animací.




**Obrázek 22 Motion layout 1**
(zdroj: Vlastní)




**Obrázek 23 Motion layout 2**
(zdroj: Vlastní)

Po kliknutí na button se tlačítko plynule posune nahoru a textBox se plynule posune do pravého spodního rohu. Celý kód je k dispozici v příloze A.11. Na rozdíl od constraintset layoutu se zde o přesun nestará kód v javě ale druhý XML soubor tzv. MotionScene soubor.

**Vysvětlení kódu**

**Motion layout v activity_main.xml**:

Motion layout je speciální typ ConstraintLayout, který umožňuje animace a přechody mezi různými stavy.

Atribut *app:layoutDescription="@xml/scene*" odkazuje na soubor scene.xml, který definuje animace a přechody.

**MotionScene v scene.xml**:

**Transition**: Definuje přechod mezi dvěma stavy (start a end) a jeho trvání. Přechod se aktivuje při kliknutí na tlačítko díky *OnClick* atributu.

**ConstraintSet start**: Definuje počáteční stav layoutu, kde TextView je umístěn nahoře a Button pod ním.

**ConstraintSet end**: Definuje konečný stav layoutu, kde TextView je umístěn dole uprostřed a Button je nahoře uprostřed.

**Princip fungování**

**Počáteční stav (start)**: TextView je umístěn nahoře a Button pod ním.

**Konečný stav (end)**: TextView je umístěn dole a Button nahoře.

**Přechod**: Když uživatel klikne na tlačítko, přechod se aktivuje a MotionLayout animuje změnu z počátečního do konečného stavu během 1000 milisekund (1 sekunda).

**Výhody Motion layout**

**Jednoduchost**: Animace a přechody lze definovat přímo v XML, což usnadňuje práci a údržbu.

**Flexibilita**: Umožňuje vytvářet složité animace a přechody mezi různými stavy uživatelského rozhraní.

**Výkon**: Motion layout je optimalizovaný pro výkon a umožňuje plynulé animace.

Specifické příkazy a vlastnosti:

- **Transition**: Definuje přechod mezi dvěma stavy layoutu.
- **OnClick**: Definuje akci, která spustí přechod.
- **ConstraintSet**: Definuje rozložení pro specifický stav.

Motion layout poskytuje vývojářům nástroj pro vytváření bohatých a složitých animací a přechodů v uživatelském rozhraní. Díky jednoduchému použití XML souborů pro definování animací a přechodů je možné snadno a rychle implementovat dynamické a interaktivní uživatelské rozhraní, které zlepší uživatelský zážitek.

# 10 Responzivní design

Vytvoření uživatelského rozhraní, které se přizpůsobí různým velikostem a typům zařízení, je klíčové pro úspěšné aplikace v dnešním různorodém ekosystému zařízení. Tento přístup zajišťuje, že aplikace bude dobře fungovat na telefonech, tabletech i dalších zařízeních.

## 10.1 Různé velikosti a rozlišení obrazovek

Pro zajištění responzivního designu na různých zařízeních je nutné využívat následující techniky a nástroje:

- **Resource Qualifiers**: Android umožňuje použití různých zdrojových souborů (layoutů, obrázků, atd.) pro různé velikosti a rozlišení obrazovek pomocí kvalifikátorů. Například:
    - **res/layout/main_activity.xml**: Výchozí layout.
    - **res/layout-sw600dp/main_activity.xml**: Layout pro obrazovky s šířkou 600dp a více (typicky tablety).
    - **res/layout-w820dp/main_activity.xml**: Layout pro obrazovky s šířkou 820dp a více.
    - **res/layout-land/main_activity.xml**: Layout pro zařízení v režimu na šířku.
- **ConstraintLayout**: Použití ConstraintLayout pro dynamické a flexibilní rozvržení prvků v uživatelském rozhraní. ConstraintLayout umožňuje vytvářet složitější rozložení, která se automaticky přizpůsobí různým velikostem obrazovek.
- **Density-independent Pixels (dp)**: Použití dp místo px (pixelů) pro definování rozměrů prvků v layoutu. Tím se zajistí konzistentní velikosti prvků na různých zařízeních s různou hustotou pixelů.
- **Vektorové grafiky**: Použití vektorových obrázků místo bitmapových pro ikony a jiné grafické prvky zajišťuje, že se grafika bude škálovat bez ztráty kvality na různých rozlišeních.

## 10.2 Adaptivní UI pro různé orientace

Pro zajištění správného zobrazení aplikace při změně orientace zařízení (na výšku a na šířku) je důležité:

- **Separate Layouts**: Vytvoření oddělených layoutů pro portrétní a krajinný režim.
    - **res/layout/main_activity.xml**: Layout pro portrétní režim.
    - **res/layout-land/main_activity.xml**: Layout pro krajinný režim.
- **Responsive Containers**: Použití kontejnerů, které se dynamicky přizpůsobují změně orientace, jako jsou ScrollView, FlexboxLayout a ConstraintLayout.

- **Handling Configuration Changes**: Implementace správného zachytávání a zpracování změn konfigurace v aktivitách.
    - Přidání **android:configChanges** atributu v manifestu:

```
<activity
    android:name=".MainActivity"
    android:configChanges="orientation|screenSize">
</activity>
```

    - Přepsání metody **onConfigurationChanged** v aktivitě:

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    if (newConfig.orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
        // Kód pro zpracování režimu na šířku
    } else if (newConfig.orientation ==
Configuration.ORIENTATION_PORTRAIT) {
        // Kód pro zpracování kódu na výšku
```

**Ukázka 10 Změna orientace obrazovky**

Implementace těchto technik a nástrojů zajistí, že uživatelské rozhraní bude plně responzivní a adaptivní pro různá zařízení a orientace, čímž se zlepší uživatelský zážitek a použitelnost aplikace.

# 11 Testování a automatizace UI

Testování uživatelského rozhraní je zásadní pro zajištění kvalitního uživatelského zážitku. Praktické informace o manuálním a automatizovaném testování jsou nezbytné pro správnou detekci a opravu chyb v uživatelském rozhraní.

Hlavními aspekty, které se testují jsou

- Vizuální konzistence
- Interakce s prvky UI
- Testování na různých zařízeních a velikostech obrazovky
- Kontrola dynamického obsahu

## 11.1 Manuální testování

Manuální testování uživatelského rozhraní zahrnuje kontrolu správného zobrazení, zarovnání a funkčnosti prvků v layoutu. Tento typ testování je zásadní pro zajištění, že aplikace poskytuje konzistentní a příjemný uživatelský zážitek na různých zařízeních a v různých orientacích obrazovky.

Testování může probíhat pouze vizuálně, jestli jsou prvky na svých místech, ale také můžeme testovat funkčnost, animace, přechody, interakce s uživatelem a responzivitu uživatelského rozhraní při různých změnách, jako je změna orientace zařízení nebo změna velikosti obrazovky. Testování by mělo být řízeno testovacím scénářem.

Výhody a nevýhody byly čerpány z (Skillmea, 2019).

Výhody:

1. Flexibilita a adaptabilita
- Možnost rychlé změny scénáře
- Specifické případy nebo obtížně automatizovatelné scénáře

2. Zaměření na uživatelskou zkušenost
- Lidský pohled na věc
- Vizuální problémy které by automatizovaným testem prošli

3. Nízké počáteční náklady
- Není počáteční investice do automatizovaných nástrojů a infrastruktury.
- Vhodné pro malé projekty nebo pro první fáze vývoje, kde se aplikace rychle mění.

Nevýhody:

1. Časová náročnost
- Manuální testování je časově náročné, zvláště u velkých a komplexních aplikací.
- Opakované testování stejné funkcionality může být únavné a zdlouhavé.

2. Subjektivita a variabilita
- Výsledky manuálního testování mohou být ovlivněny subjektivním pohledem testera a mohou se lišit mezi různými testery.
- Nedostatek konzistence v provádění testů může vést k nedůsledným výsledkům.

3. Náchylnost k lidským chybám
- Testeři mohou přehlédnout chyby nebo opomenout některé testovací kroky, což může vést k neúplnému testování.

Manuální testování hraje důležitou roli při zajišťování kvality softwaru, zejména v oblasti uživatelského rozhraní a použitelnosti. Přestože má své nevýhody, lidský faktor je zde spíše žádoucí vlastností. V ideálním případě by mělo být manuální testování kombinováno s automatizovaným testováním, aby byla zajištěna komplexní a efektivní kontrola kvality.

## 11.2 Automatizované testování

Automatizované testování uživatelského rozhraní a layoutů v Androidu zahrnuje použití nástrojů a skriptů k automatickému ověření správného fungování a zobrazení prvků UI. Tento přístup je zásadní pro zajištění, že aplikace je konzistentní a funkční na různých zařízeních a v různých orientacích.

Výhody a nevýhody automatizovaného testování jsou přesným opakem u manuálního testování.

Nástroje pro automatizované testování:

**Espresso**

Oficiální nástroj Google pro automatizované testování UI v Androidu. Umožňuje psaní jednoduchých a srozumitelných testovacích skriptů, které simulují interakce uživatele s aplikací.

**UI Automator**

Nástroj pro automatizované testování UI, který umožňuje interakci s prvky systému Android mimo aplikaci, jako jsou oznámení nebo nastavení systému.

**Appium**

Open-source nástroj pro automatizované testování mobilních aplikací, který podporuje testování na Android i iOS. Umožňuje psaní testů v různých programovacích jazycích.

# Závěr

Práce se zaměřuje na možnosti XML pro pozicování komponent v rozhraní Androidu, přičemž klade důraz na různé typy layoutů a jejich využití v praxi. Hlavním cílem je poskytnout komplexní přehled o metodách a technikách, které vývojářům umožňují efektivně a esteticky rozmístit prvky uživatelského rozhraní.

V úvodní částech práce jsou vysvětleny základní pojmy a teorie základních layoutů. Tato základní terminologie je klíčová pro pochopení následujících kapitol. Detailně byly rozebrány různé typy layoutů. Každý layout byl představen s konkrétními příklady a vysvětlením jeho výhod a nevýhod. Další část práce se zaměřila na kombinaci layoutů, což je technika využívaná pro dosažení složitějších a responzivních uživatelských rozhraní. Byly uvedeny příklady konkrétních kombinací layoutů, které ilustrují praktické využití těchto technik. V rámci návrhu optimálního uživatelského rozhraní byly uvedeny zásady pro navrhování optimálního uživatelského rozhraní, včetně důležitosti přehlednosti, konzistence, přizpůsobitelnosti a responzivity. Byla zdůrazněna role testování a zpětné vazby od uživatelů při tvorbě uživatelského rozhraní. V souvislosti s tím byla také popsána technika responzivního designu a metody zajištění, že aplikace bude správně fungovat na různých zařízeních s různými velikostmi a rozlišeními obrazovek. Na neposledním místě se práce věnuje metodám manuálního a automatizovaného testování uživatelského rozhraní, včetně konkrétních nástrojů a postupů pro testování v Androidu. Testování je zásadní pro zajištění, že aplikace je stabilní, funkční a poskytuje dobrý uživatelský zážitek.

Pro vývojáře je klíčové mít znalost různých typů layoutů, protože každý layout má své specifické využití a je důležité vybrat ten nejvhodnější pro konkrétní scénář. Kombinace layoutů může výrazně zvýšit flexibilitu a efektivitu návrhu uživatelského rozhraní. Responzivní design je nezbytný pro zajištění, že aplikace bude správně fungovat a vypadat na různých zařízeních, což zahrnuje testování na různých obrazovkách a v různých orientacích. Pravidelné testování a získávání zpětné vazby od uživatelů jsou klíčové pro zlepšování uživatelského rozhraní.

Budoucí práce by se mohla zaměřit na integraci nových technologií, jako jsou AR, VR a AI, které mohou ovlivnit návrh uživatelského rozhraní v Androidu. Dále by bylo vhodné vyvíjet a hodnotit nové nástroje a metody pro efektivnější testování uživatelského rozhraní a provádět hloubkové studie zaměřené na analýzu uživatelského chování a jeho vliv na návrh uživatelského rozhraní.

Závěrem lze konstatovat, že správné pozicování komponent v uživatelském rozhraní je klíčové pro úspěch mobilní aplikace. Tato práce poskytla ucelený přehled o možnostech XML-definice pozicování a ukázala praktické způsoby, jak tuto problematiku efektivně řešit.

# Seznam použité literatury


- Advisio.cz. (2022). 10 trendů v UX a UI pro rok 2022, které potřebujete znát. [online] Dostupné z: https://www.advisio.cz/blog/10-trendu-v-ux-a-ui-pro-rok-2022-ktere-potrebujete-znat/ [Citováno 28.11.2023].
- Android Developers. (2023) 'Android Developers.' [online] Dostupné na: https://developer.android.com/ [Citováno 19.11.2023].
- Android Developers. "Performance and view hierarchies." App quality. [Online]. [Citováno: 10. 6.2024] Dostupné na : https://developer.android.com/topic/performance/rendering/optimizing-view-hierarchies
- ALLEN, Grant. Android 4: průvodce programováním mobilních aplikací. Brno: Computer Press, 2013. ISBN 978-80-251-3782-6.
- BankMyCell. (2023). "How Many Apps Are on the Google Play Store?". BankMyCell Blog. [online] Dostupné na: https://www.bankmycell.com/blog/number-of-google-play-store-apps/ [Citováno 5.12.2023].
- CodePath. (2023). Android Guides - CodePath. [online] Dostupné z: https://guides.codepath.com/android [Citováno 20.11.2023].
- GeeksforGeeks. (2019) 'A Complete Guide to Learn XML for Android App Development'. [online] Dostupné na: https://www.geeksforgeeks.org/a-complete-guide-to-learn-xml-for-android-app-development/ [Citováno 19.11.2023].
- Google Developers. (2023). Android Developers Blog. [online] Dostupné z: https://android-developers.googleblog.com/ [Citováno 20.11.2023].
- Chiang, N. (2019) Android Studio Development Essentials. [online] Dostupné na: https://www.techotopia.com/index.php/Android_Studio_Development_Essentials [Citováno 20.11.2023].
- Itnetwork.cz (2023). [online] Dostupné z: Itnetwork.cz. [Citováno 2.12.2023].
- Nedbalek, S. (2016). Uživatelské rozhraní mobilních aplikací. Masarykova univerzita. [online] Dostupné z: https://www.muni.cz/ [Citováno 20. 12 2023].
- Open Handset Alliance. (2021). “O Androidu”. [online] Dostupné na: https://www.android.com/intl/cs_cz/about/ [Citováno 5.12.2023].
- Payne, I. (2013) AndroidTM User Interface Design. Brno: Computer Press. ISBN 978-80-251-3782-6.
- Phillips, B., Hardy, C. a Marsicano, K. (2017) Android Programming THE BIG NERD RANCH GUIDE. 3rd ed. Addison-Wesley Professional. ISBN 978-0-13-470605-4.
- Publi.cz (2023). 'Uspořádání prvků v okně (Layouts)'. [online] Dostupné na: https://publi.cz/books/275/06.html [citováno 5.12.2023].

- Skillmea (2019) 'Manuální vs. automatizované testování', Skillmea. [online] Dostupné na: https://skillmea.cz/blog/manualne-vs-automatizovane-testovanie [Citováno: 10. 6.2024].
- Stack Overflow. (2023). Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Dostupné z: https://stackoverflow.com/ [Citováno 20.11.2023].
- XML pro začátečníky. (2024) [online] Dostupné z: https://support.microsoft.com/cs-cz/office/xml-pro-za%C4%8D%C3%A1te%C4%8Dn%C3%ADky-a87d234d-4c2e-4409-9cbc-45e4eb857d44. [Citováno 6.6.2024].
- *XML*. (2024). [online] Dostupné z: https://en.wikipedia.org/wiki/XML. [Citováno 6.6.2024].
- Vávrů, Jiří a Miroslav Ujbányai. Programujeme pro Android. 2., rozš. vyd. Praha: Grada, 2013. Průvodce (Grada). ISBN 978-80-247-4863-4.

# Přílohy

V této sekci jsou zahrnuty veškeré zdrojové kódy použité v této práci. To zahrnuje XML soubory definující layouty popřípadě pokud jsou třeba k funkčnosti tak jsou zde Java soubory implementující funkce, a také soubory build.gradle obsahující konfiguraci projektu. Tyto soubory poskytují detailní přehled o struktuře a implementaci aplikace, a slouží jako referenční materiál pro lepší porozumění vývojovému procesu a použitým technologiím.

# A. Zdrojové kódy a definice layoutů

## A.1. Lineární layout vertikální

Ukázka lineárního vertikálního layoutu. Vertikální lineární layout se nastavuje ve vlastnosti orientation na hodnotu *vertical*. Layout zobrazuje komponenty vertikálně („pod sebou“). Pozicované komponenty přebírají šířku podle šířky layoutu a výška layoutu je přerozdělena mezi komponenty (pokud není zadána absolutní hodnota výšky).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"> <!--Určení směru layoutu-->

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="542dp"
            android:layout_weight="1"
            android:gravity="center"
            android:background="@color/cardview_dark_background"
            android:text="TextBox1"
        />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="721dp"
            android:layout_weight="1"
            android:gravity="center"
            android:background="@color/cardview_shadow_start_color"
            android:text="TextBox2" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="518dp"
            android:layout_weight="1"
            android:gravity="center"
            android:background="@color/cardview_light_background"
            android:text="TextBox3"/>
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## A.2. Lineární layout horizontální

Ukázka horizontálního linárního layoutu. Horizontální lineární layout se nastavuje ve vlastnosti orientation na hodnotu *horizontal*. Layout zobrazuje komponenty horizontálně („vedle sebe“). Pozicované komponenty přebírají výšku podle výšky layoutu a šířka layoutu je přerozdělena mezi komponenty (pokud není zadána absolutní hodnota šířky).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"> <!--Určení směru layoutu-->

        <TextView
            android:id="@+id/textView"
            android:layout_width="96dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"
            android:background="@color/cardview_dark_background"
            android:text="TextBox1" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="200dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"
            android:background="@color/cardview_shadow_start_color"
            android:text="TextBox2" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"

android:androidbackground="@color/cardview_light_background"
            android:text="TextBox3" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## A.3. Rozdělení prostoru podle váhy

Ukázka přerozdělení rozměrů podle váhy komponenty.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        tools:ignore="MissingConstraints">

        <TextView
            android:layout_width="200dp"
            android:layout_height="50dp"
            android:text="200dp"
            android:gravity="center"
            android:background="@color/cardview_dark_background"/>


        <TextView
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="2"
            android:text="Váha 1"
            android:gravity="center"
            android:background="@color/cardview_shadow_start_color"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="4"
            android:text="Váha 1"
            android:gravity="center"
            />

    </LinearLayout>

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

## A.4. Relative layout

Ukázka relativního layoutu. Umístění pozicovaných komponent je definováno pomocí atributů (alignParentStart, alignParentEnd, alignParentBottom, marginStart, marginEnd, marginBottom) a to vzhledem k jiné komponentě nebo layoutu.

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

        <TextView
            android:id="@+id/textView"
            android:layout_width="267dp"
            android:layout_height="369dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentBottom="true"
            android:layout_marginEnd="59dp"
            android:layout_marginBottom="35dp"
            android:background="@color/cardview_dark_background"
            android:text="TextBox1"/>

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="270dp"
            android:layout_above="@+id/textView"
            android:layout_marginBottom="55dp"
            android:background="@color/cardview_shadow_start_color"
            android:text="TextBox2"
            android:gravity="center"/>

        <TextView
            android:id="@+id/textView3"
            android:layout_width="130dp"
            android:layout_height="246dp"
            android:layout_alignParentStart="true"
            android:layout_alignParentEnd="true"
            android:layout_alignParentBottom="true"
            android:layout_marginStart="112dp"
            android:layout_marginEnd="168dp"
            android:layout_marginBottom="120dp"
            android:background="@color/cardview_light_background"
            android:text="TextBox3"
            android:gravity="center"/>
    </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## A.5. Constraint layout

Ukázka constrained layoutu. Pozicování komponent je provedeno pomocí atributů (layout_constraintStart_toStartOf, layout_constraintTop_toTopOf, layout_marginStart, layout_marginTop) a to vzhledem k layoutu (rodiči), hodnota *parent* nebo vzhledem k jiné komponentě.

```
<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/textBox1"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:background="#DDDDDD"
        android:gravity="center"
        android:text="TextBox1"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_marginStart="16dp"
        app:layout_marginTop="16dp" />

    <TextView
        android:id="@+id/textBox2"
        android:layout_width="118dp"
        android:layout_height="100dp"
        android:background="#CCCCCC"
        android:gravity="center"
        android:text="TextBox2"
        app:layout_constraintStart_toEndOf="@+id/textBox1"
        app:layout_constraintTop_toTopOf="@+id/textBox1" />

    <TextView
        android:id="@+id/textBox3"
        android:layout_width="102dp"
        android:layout_height="100dp"
        android:background="#BBBBBB"
        android:gravity="center"
        android:text="TextBox3"
        app:layout_constraintStart_toStartOf="@+id/textBox1"
        app:layout_constraintTop_toBottomOf="@+id/textBox1" />

    <TextView
        android:id="@+id/textBox4"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:layout_marginStart="4dp"
        android:layout_marginTop="260dp"
```

```
        android:background="#AAAAAA"
        android:gravity="center"
        android:text="TextBox4"
        app:layout_constraintStart_toEndOf="@+id/textBox3"
        app:layout_constraintTop_toBottomOf="@+id/textBox2" />

    <TextView
        android:id="@+id/textBox5"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:layout_marginStart="10dp"
        android:background="#999999"
        android:gravity="center"
        android:text="TextBox5"
        app:layout_constraintStart_toEndOf="@+id/textBox1"
        app:layout_constraintTop_toBottomOf="@+id/textBox3" />

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

## A.6. Table layout

Ukázka tabulkového layoutu. Komponenty jsou pozicovány po řádcích (TableRow) a to v horizontálním uspořádání. V ukázce tabulkového layoutu jsou komponenty pozicovány ve třech řádcích, a to v počtu 3 komponenty, 2 komponenty a jedna komponenta. Přerozdělení šířky komponent je analogické jako u horizontální lineárního layoutu.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TableRow>
            <TextView
                android:id="@+id/textView1"
                android:layout_width="0dp"
                android:layout_height="180dp"
                android:layout_weight="1"
                android:text="TextView1"
                android:background="#CCCCCC"
                android:gravity="center"/>

            <TextView
                android:id="@+id/textView2"
                android:layout_width="0dp"
                android:layout_height="180dp"
                android:layout_weight="1"
                android:text="TextView2"
                android:background="#BBBBBB"
                android:gravity="center"/>

            <TextView
                android:id="@+id/textView3"
                android:layout_width="0dp"
                android:layout_height="180dp"
                android:layout_weight="1"
                android:text="TextView3"
                android:background="#AAAAAA"
                android:gravity="center"/>
        </TableRow>

        <TableRow>
            <TextView
                android:id="@+id/textView4"
                android:layout_width="0dp"
                android:layout_height="180dp"
```

```
                android:layout_weight="2"
                android:text="TextView4"
                android:background="#999999"
                android:gravity="center"/>

            <TextView
                android:id="@+id/textView5"
                android:layout_width="0dp"
                android:layout_height="180dp"
                android:layout_weight="1"
                android:text="TextView5"
                android:background="#888888"
                android:gravity="center"/>
        </TableRow>

        <TableRow>
            <TextView
                android:id="@+id/textView6"
                android:layout_width="0dp"
                android:layout_height="180dp"
                android:layout_weight="1"
                android:text="TextView6"
                android:background="#777777"
                android:gravity="center"/>
        </TableRow>

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

## A.7. Kombinace lineárních layoutů

Ukázka kombinování layoutů různých typů. Ve vertikálním lineárním layoutu jsou umístěny horizontální lineární layouty představující jednotlivé řádky. Principiálně lze tímto způsobem kombinovat jakýkoliv typ layoutu.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"> <!--První sloupec-->

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="341dp"
            android:orientation="horizontal"> <!--První řádek-->

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="TextBox1"
                android:gravity="center"
                android:background="@color/cardview_dark_background"/>

            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center"
                android:text="TextBox2"

android:background="@color/cardview_shadow_start_color"/>

            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center"
                android:text="TextBox3"

android:androidbackground="@color/cardview_light_background"/>
        </LinearLayout>
```

```
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="390dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/textView4"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center"
                android:text="TextBox4" />

            <TextView
                android:id="@+id/textView5"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center"
                android:text="TextBox5"
                android:background="@color/cardview_dark_background"/>

            <TextView
                android:id="@+id/textView6"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center"
                android:text="TextBox6"

android:background="@color/cardview_shadow_start_color"/>
        </LinearLayout>
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## A.8. Scroll wiev, relative a linear layout

Ukázka scroll view spolu s lineárními layoutu. Scroll view zajišťuje možnost rolování na aktivitě, pokud se komponenty nevejdou na obrazovku. Lze použít pro seznamy.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/textView8"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="Logo aplikace"
            android:textSize="34sp" />

        <Space
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="4" />

        <ImageButton
            android:id="@+id/imageButton"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            app:srcCompat="@android:drawable/ic_lock_power_off" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintStart_toStartOf="parent">

    <ImageButton
        android:id="@+id/imageButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:srcCompat="@drawable/ic_add" />

    <ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:srcCompat="@drawable/ic_change" />

    <ImageButton
        android:id="@+id/imageButton5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:srcCompat="@drawable/ic_save" />

    <ImageButton
        android:id="@+id/imageButton4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:srcCompat="@drawable/ic_delete" />

</LinearLayout>

<ScrollView
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@+id/linearLayout"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout2">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView9"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Vzorový text"
            android:textSize="50dp" />
```

```
<Space
    android:layout_width="match_parent"
    android:layout_height="35dp"
    />

<TextView
    android:id="@+id/textView10"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Vzorový text"
    android:textSize="50dp"/>

<Space
    android:layout_width="match_parent"
    android:layout_height="35dp"
    />

<TextView
    android:id="@+id/textView11"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Vzorový text"
    android:textSize="50dp"/>

<Space
    android:layout_width="match_parent"
    android:layout_height="35dp"
    />

<TextView
    android:id="@+id/textView12"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Vzorový text"
    android:textSize="50dp" />

<Space
    android:layout_width="match_parent"
    android:layout_height="35dp" />

<TextView
    android:id="@+id/textView13"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Vzorový text"
    android:textSize="50dp" />

<Space
    android:layout_width="match_parent"
    android:layout_height="35dp" />
```

```
            <TextView
                android:id="@+id/textView14"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Vzorový text"
                android:textSize="50dp" />

            <Space
                android:layout_width="match_parent"
                android:layout_height="35dp" />

        </LinearLayout>
    </ScrollView>

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

## A.9. Flexbox

Ukázka FlexBox layoutu. Pro použití tohoto typu layoutu je nutné definovat závislost `implementation 'com.google.android.flexbox:flexbox:3.0.0'` v build gradle.

### A.9.1. XML

```
<com.google.android.flexbox.FlexboxLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:flexbox="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    flexbox:flexDirection="column"
    flexbox:justifyContent="space_around"
    flexbox:alignItems="center">

    <TextView
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:gravity="center"
        android:text="Box 1"
        android:padding="10dp"
        android:background="@color/cardview_dark_background"/>

    <TextView
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:gravity="center"
        android:text="Box 2"
        android:padding="10dp"
        android:background="@color/cardview_shadow_start_color" />

    <TextView
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:gravity="center"
        android:text="Box 3"
        android:padding="10dp"
        android:background="@color/cardview_dark_background" />
</com.google.android.flexbox.FlexboxLayout>
```

### A.9.2. build.gradle (Module :app)

```
plugins {
    id 'com.android.application'
}

android {
    namespace 'com.example.shopping_list'
    compileSdk 33

    defaultConfig {
        applicationId "com.SHApp.shopping_list"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.android.gms:play-services-
ads:23.1.0'//reklama
    implementation 'androidx.appcompat:appcompat:1.7.0'
    implementation 'com.google.android.flexbox:flexbox:3.0.0'
    implementation 'com.google.android.material:material:1.12.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.5.1'
```

## A.10. Constraintset layout

Ukázka constraintset layoutu. Constrainset layout je definován programově.

### A.10.1. XML

```
<!-- res/layout/activity_main.xml -->
<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"
    android:id="@+id/constraintLayout">

    <TextView
        android:id="@+id/textView"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:text="Hello World"
        android:background="@android:color/holo_blue_light"
        android:gravity="center"
        android:textColor="@android:color/white"
        android:textSize="18sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        android:padding="16dp"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        app:layout_constraintTop_toBottomOf="@id/textView"
        app:layout_constraintStart_toStartOf="parent"
        android:padding="16dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

### A.10.2. Java

```
package com.example.constraintset;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.ConstraintSet;
import androidx.transition.TransitionManager;

public class MainActivity extends AppCompatActivity {

    private boolean isChanged = false;
    private ConstraintLayout constraintLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Inicializace ConstraintLayout
        constraintLayout = findViewById(R.id.constraintLayout);

        // vytvoření a klonování ConstraintSets
        final ConstraintSet initialSet = new ConstraintSet();
        initialSet.clone(constraintLayout);

        final ConstraintSet changedSet = new ConstraintSet();
        changedSet.clone(constraintLayout);
        changedSet.clear(R.id.textView, ConstraintSet.TOP);
        changedSet.connect(R.id.textView, ConstraintSet.BOTTOM,
R.id.button, ConstraintSet.TOP);
        changedSet.connect(R.id.textView, ConstraintSet.END,
ConstraintSet.PARENT_ID, ConstraintSet.END);
        changedSet.connect(R.id.textView, ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START);

        // inicializace tlačítka a nastavení OnClickListener
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Použití TransitionManager pro animaci layoutu

TransitionManager.beginDelayedTransition(constraintLayout);
                if (isChanged) {
                    initialSet.applyTo(constraintLayout);
                } else {
                    changedSet.applyTo(constraintLayout);
                }
                isChanged = !isChanged;
            }
        });
    }
}
```

## A.11. Motion layout

Ukázka motion layoutu vhodná k vytváření složitých pohybových přechodů a animací v uživatelském rozhraní. Může být použit k implementaci animovaných přechodů mezi různými rozloženími pomocí XML.

### A.11.1. Main XML

```
<!-- res/layout/activity_main.xml -->
<androidx.constraintlayout.motion.widget.MotionLayout
    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"
    app:layoutDescription="@xml/scene"
    android:id="@+id/motionLayout">

    <TextView
        android:id="@+id/textView"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:text="Hello World"
        android:background="@android:color/holo_blue_light"
        android:gravity="center"
        android:textColor="@android:color/white"
        android:textSize="18sp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        android:padding="16dp"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        app:layout_constraintTop_toBottomOf="@id/textView"
        app:layout_constraintStart_toStartOf="parent"
        android:padding="16dp"/>
</androidx.constraintlayout.motion.widget.MotionLayout>
```

### A.11.2. Scene.xml

```
<!-- res/xml/scene.xml -->
<MotionScene
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <Transition
        app:constraintSetStart="@id/start"
        app:constraintSetEnd="@id/end"
        app:duration="1000">
        <OnClick
            app:targetId="@id/button"
            app:clickAction="transitionToEnd"/>
    </Transition>

    <ConstraintSet android:id="@+id/start">
        <Constraint
            android:id="@id/textView"
            android:layout_width="200dp"
            android:layout_height="100dp"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>

        <Constraint
            android:id="@id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toBottomOf="@id/textView"
            app:layout_constraintStart_toStartOf="parent"/>
    </ConstraintSet>

    <ConstraintSet android:id="@+id/end">
        <Constraint
            android:id="@id/textView"
            android:layout_width="200dp"
            android:layout_height="100dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"/>

        <Constraint
            android:id="@id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>
    </ConstraintSet>

</MotionScene>
```