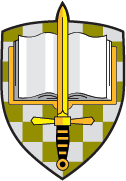
**UNIVERZITA OBRANY**

**Fakulta vojenských technologií**



**NÁZEV PRÁCE**

**Příjmení, jméno:**

**Učební skupina:**

**BRNO 2024**

**OBSAH**

[ÚVOD 4](#_Toc166764150)

[1 Operační systém Android 5](#_Toc166764151)

[1.1 Historie Android 5](#_Toc166764152)

[1.2 Architektura Operačního Systému 8](#_Toc166764153)

[1.3 Vývojové nástroje a prostředí 10](#_Toc166764154)

[2 Základy Androidových aplikací 11](#_Toc166764155)

[2.1 Programovací jazyky pro Android 11](#_Toc166764156)

[2.2 Typy aplikací [1] 13](#_Toc166764157)

[2.3 Komponenty aplikací 13](#_Toc166764158)

[2.3.1 Aktivity (Activities) 14](#_Toc166764159)

[2.3.2 Služby (Services) 14](#_Toc166764160)

[2.3.3 Přijímače vysílání (Broadcast receivers) 14](#_Toc166764161)

[2.3.4 Poskytovatelé obsahu (Content providers) 14](#_Toc166764162)

[2.3.5 Záměr (Intent) 15](#_Toc166764163)

[2.3.6 Android Manifest 16](#_Toc166764164)

[2.3.7 Android API 18](#_Toc166764165)

[3 Tvorba Aplikace 19](#_Toc166764166)

[3.1 Návrh aplikace 19](#_Toc166764167)

[3.2 Tvorba aplikace 22](#_Toc166764168)

[3.3 activity\_main.xml 27](#_Toc166764169)

[3.4 activity\_game.xml 30](#_Toc166764170)

[3.5 MainActivity.kt 33](#_Toc166764171)

[3.6 GameActivity.kt 35](#_Toc166764172)

[4 Testování 43](#_Toc166764173)

[5 Tvorba aplikace 2 45](#_Toc166764174)

[5.1 activity\_main.xml 2 46](#_Toc166764175)

[5.2 activity\_game.xml 2 46](#_Toc166764176)

[5.3 MainActivity.kt 2 46](#_Toc166764177)

[5.4 GameActivity.kt 2 48](#_Toc166764178)

[5.5 activity\_leaderboard.xml a item\_leaderboard\_header.xml 55](#_Toc166764179)

[5.6 LeaderboardActivity.kt 58](#_Toc166764180)

[ZÁVĚR 60](#_Toc166764181)

[SEZNAM POUŽITÉ LITERATURY (vzor) 61](#_Toc166764182)

[SEZNAM POUŽITÝCH ZKRATEK 62](#_Toc166764183)

[SEZNAM OBRÁZKŮ A TABULEK 63](#_Toc166764184)

[SEZNAM PŘÍLOH 64](#_Toc166764185)

# ÚVOD

Rozvoj technologií, které pronikají do každého aspektu našeho života vede mimo jiné k otevření nových možností v oblasti simulací a výcviku. Jedním ze směrů, se kterým je tato práce provázána, je využití dotykových diplejů a tabletů v leteckých simulátorech pro testování schopností pilota při simulovaném letu. Tato diplomová práce se zaměřuje na vývoj, optimalizaci a využití takové aplikace, která přináší jeden z možných způsobů měření.

V dnešní moderní době, kdy letecký průmysl podstupuje nepřetržitý vývoj a inovace, nabývá výcvik pilotů klíčového významu. Pilotní trenažery se staly nepostradatelným prvkem ve výcviku pilotů, které především ve vojenském prostředí nabízejí bezpečnou možnost přípravy na válečný konflikt a zároveň dovolují vyhodnotit jejich schopnost ovládat stroje pod vlivy vznikajícími v extrémních rychlostech.

Pilotní trenažéry jsou navrženy tak, aby co nejvíce simulovaly reálné letové podmínky. Stacionární simulátory poskytují statické prostředí, zatímco pohyblivé simulátory, umístěné na pohyblivých platformách, dokáží reprodukovat i pohyb letadla během letu. Tyto schopnosti trenažerů dokáží piloty seznámit a zdokonalit v jednotlivých fázích letu, od vzletu až po přistání, a naučit je reagovat na různé situace, jako jsou například turbulence.

S rychlým rozvojem technologií se mění jak podoba pilotních trenažerů samotných, tak i zařízeními s nimi spojenými.

1. Operační systém Android

Android je mobilní operační systém na jádře Linuxu vyvýjený firmou Google pod hlavičkou Open Handset Alliance v rámci projektu Android Open Source Project (AOSP) poprvé zveřejněný v roce 2008, který nastartoval revoluci na poli mobilních operačních systémů. Několik let po vzniku nahradil postupem času mobilní operační systém Symbian, címž se stal přímým konkurentem systému iOS. Android samotný zajišťuje pouze základní systémové funkce, které musí být pro dodatečné funkce zařízení doplněny o další proprietární software, neboli closed source software s licencí náležící nějaké firmě a většinou bez volně dostupného zdrojového kódu. Jedná se například o služby jako Google Play, YouTube nebo Amazon Appstore.

Popularita operačního systému Android u všech skupin, od uživatelů po vývojáře, spočívá v hned několika jeho vlastnostech. Jedná se o open-source software, což znamená, že zdrojový kód androidu je volně dostupný a šířitelný. Navíc se jedná o velmi stabilní a rychlý operační systém, což společně s jeho otevřeností vedlo k jeho oblíbení si mnoha vývojáři mobilních aplikací, kteří jej mohou využívat bez licenčních poplatků. Z uživatelského hlediska nabízí Android podporu u mnoha výrobců, což znamená rozmanitý výběr cenově dostupných zařízení s tímto operačním systémem. V neposlední řadě nabízí zařízení širokou nabídku aplikací mimojiné s podporou integrace Google služeb, jako je Gmail, Google Drive, Google Maps a další.

## Historie Android

Operační systém Android byl původně vyvinut malou společností Android Inc. založenou roku 2003 a později odkoupenou v roce 2005 Googlem, kde pokračoval vývoj operačního systému v alianci několika dalších firem pod společným názvem Open Handset Alliance. První verze Androidu byla zveřejněna roku 2008 a poskytovala na tu dobu pokročilé funkce, jako internetový prohlížeč a integrované mapy. První zařízení na trhu s operačním systémem Android byl mobilní telefon T-Mobile G1, který je považovaý za vůbec první skutečný smartphone.

Po prvním oficiálním vydání Androidu 1.0 začal Google vydávat nové verze operačního systému a od původní prošel mnohou řadou změn a vylepšení opravujících zjištěné chyby a přidávajícíh nové funkčnosti do systému. Jednotlivé vycházející verze dostaly název podle zákusků v abecedním pořadí. Každou z verzí rozebereme a uvedeme některé nové vlastnosti.

1. Android 1.0
   1. Jednalo se o první oficiální vydání operačního systému zveřejněnou 28. září 2008 a byl založen na jádře Linux 2.6.25.
   2. Obsahoval webový prohlížeč pro prohlížení HTML a XHTML stránek s funkcí zobrazení více stránek ve formě karet.
   3. Zařízení mělo fotoaparát bez jakýchkoli dalších funkcí.
   4. Obsahoval Android Market – on-line katalog plný aplikací a her.
   5. Podpora emailu s přístupem k emailovým serverům s podporou protokolů POP3, SMTP a IMAP4.
   6. Google aplikace jako Gmail, Google Contacts, Google Calendar a Google Maps atd.
2. Android 1.5 (Cupcake)
   1. Platforma vydaná 27. dubna 2009 založená na jádře Linux 2.6.27
   2. Přivedla zpřesnění všech dosavadních prvků uživatelského rozhraní.
   3. Celkové zrychlení všech softwarových komponent.
   4. Funkce otočení obrazovky při natočení zařízení.
   5. Možnost ve fotoaparátu navíc nahrávat videa.
   6. P5idání domovské obrazovky.
   7. Rozhraní API a Manifest prvky pro vývoj aplikací.
3. Android 1.6 (Donut)
   1. Verze systému vydaná 15. září 2009 na Linuxovém jádře 2.6.29
   2. Zlepšení grafického rozhraní pro fotoaparát a galerii.
   3. Přidání rychlého vyhledávání – možnost pro uživatele, jak z jednoho textového pole prohledávat více zdrojů (web, kontakty, aplikace, historii)
   4. Panel pro kofiguraci VPN (Virtual Private Network)
   5. Nově podpora rozlišení obrazovky WVGA (Wide video Graphics Array)
   6. Hlasové vyhledávání
   7. Možnost zobrazení, které aplikace využívají kolik energie, na základě čehož má uživatel možnost manuálně aplikace ukončit pro úsporu
4. Android 2.0/2.1 (Eclair)
   1. Verze zveřejněná chvíli po předchozí 26. října 2009 postavená na stejném jádře.
   2. Vylepšení telefonního seznamu.
   3. Podpora Microsoft Exchange.
   4. Podpora více rozlišení displeje, což mimojiné umožnilo uvést na trh jedny z prvních Android tabletů.
   5. Podpora Bluetooth 2.1.
   6. Přidání podpory HTML5 v prohlížeči.
5. Android 2.2 (Froyo)
   1. Vyšla dne 20. května 2010 na Linuxovém jádře 2.6.32.
   2. Přibyla možnost instalace aplikací na paměťovou kartu.
   3. Vylepšení fotoaparátu a kamery.
   4. Sdílení internetového připojení zapojením zařízení přes USB k PC.
   5. Možnost využít zařízení jako WiFi hotspot a sdílet skrze zařízení internetové připojení.
   6. JIT (Just-in-Time) kompilátor zrychlující celkový výkon systému.
   7. Vylepšená správa paměti RAM.
6. Android 2.3/2.3 (Gingerbread)
   1. Jedna z nejvyužívanějších verzí androidu zveřejněná 6. prosince 2010 na Linuxovém jádře 2.6.35.
   2. Podpora video formátů WebM a HTML5.
   3. Podpora IP telefonie v rámci protokolu SIP
7. Android 3.0/3.1/3.2 (Honeycomb)
   1. Verze určená pouze pro tablety s převážně grafickými upravami.
8. Android 4.0/4.0.1/4.0.2 (Ice Cream Sandwich)
   1. Verze vydaná v roce 2011 na Linuxové jádře 3.0.1. Opět nasazená na mobilní telefony a přináší si ty nejlepší funkce z verzí Honeycomb.
   2. Verze mají vylepšené rozpoznávání hlasu a možnost detekce obličeje.
   3. Zdokonalení webového přohlížeče a dalších aplikací, které lze nyní ovládat pohybovými gesty.

## Architektura Operačního Systému

Arhitektura OS Android se skládá z pěti vrstev, z nichž každá samostatně provádí v zařízení jiné operace. Toto oddělení vstev je ale především jen teoretické, protože v praxi dochází ke spolupráci jednotlivých částí systému a vrstvy nejsou mezi sebou striktně odděleny.

Linux Kernel

Nejnižší vrstvou architektury je Linux Kernel nebo též jádro operačního systému. Jedná se o nejdůležitější část všech operačních systémů fungujících na Linuxu. Základní funkcí této vrstvy je zodpovědnost za low-level operace systému jako je správa disku a alokování paměti operacím a vytváří základní rozhraní mezi hardwarem počítače a jeho funkcemi. Při startu zařízení je jádro zavedené do operační paměti a je mu předáno řízení, které zahrnuje například kontrolu nad koordinací činnosti všech běžících procesů, už zmíněná správa paměti, správa sítí a mnoho dalších.

Knihovny

Libraries (knihovny) jsou další vrstvou architektury Android. Obecně se jedná o sety instrukcí napsaných v programovacím jazyce. Obsahují většinou konfigurační data, dokumentaci, předepsaný kód atd. Android nabízí mnohá rozhraní API pro vývoj aplikací. Pro představu uvedeme některé nejčastěji používané, které jsou dostupné pro všechna android zařízení:

1. Android.util – hlavní balíček knihoven obsahující utility jako manipulace data/času, base64 enkodéry a dekodéry a nástroje pro parsování XML
2. Android.os – knihovny s přístupem k základním službám operačního systému, jako je předávání zpráv a meziprocesová komunikace
3. Android.graphics – knihovny pro zobrazování grafických prvků
4. Android.text – knihovny pro zpracování a zobrazení řetězců textu
5. Android.database – knihovny s třídamy potřebnými pro práci s kurzory datábází
6. Android.content – obsahují třídy pro přístup a publikování dat na zařízení [1]

Android runtime

Prostředí Android runtime je jednou z nedůležitějších částí operačního systému Android. Obsahuje komponenty jako jsou základní knihovny a Dalvik Virtual Machine (DVM). Především poskytuje základ pro application framework a pohání aplikace s pomocí základních knihoven. Základní knihovny nám umožňují implementovat aplikace pro Android pomocí standardních programovacích jazyků jako je Java a nebo Kotlin. [2]

DVM byl exkluzivně pro Android vyvíjen od roku 2005 společností Google a je pro něj speciálně navržený a optimalizovaný tak, aby zajistil efektivní běh více instancí na jednom zařízení. [1] Podobně jako Java Virtual Machine má Dalvik registrově orientovanou architekturu a využívá základní vlastnosti Linuxového jádra, jako je nízko-úrovňová správa paměti, koordinace běžících procesů a práce s vlákny (threading). [2]

Důvodem ke vzniku nového virtuálního stroje byla skutečnost, že programátoři vyvíjející aplikace pro operační systém Android používali často právě jazyk Java. Knihovny tohoto jazyka jsou licencovány jako open source, ale virtuální stroj (JVM), který slouží k překladu programu do spustitelné formy, již není volně šířitelný. Dalším důvodem vzniku virtuálního stroje Dalvik byla optimalizace virtuálního stroje pro potřeby mobilních zařízení, kde hrál klíčovou roli výkon a úspora energie.

Obsah knihoven programovacího jazyka Java, které tato vrstva zahrnuje, lze srovnat s platformou Java SE (Standard Edition), která zahrnuje JVM, API knihovny základních funkcí a API knihovny ori vytváření klientských desktopových aplikací (AWT, Swing). Na Androidu byly ale AWT a Swing nahrazeny knihovnami pro tvorbu uživatelského rozhraní a přibyly knihovny Apache pro práci se sítí.

Aplikace pro Android jsou psány například v jazyce Java, poté jsou přeloženy do Java bytecode a nakonec do mezikódu pomocí Dalvik kompilátoru. Výsledný bytecode je spuštěn na DVM a každá aplikace funguje jako samostatný proces s vlastní instancí DVM. [1]

Application framework

Aplikační rámec poskytuje několik důležitých tříd, které se používají k vytváření aplikací pro Android. Nabízí obecnou abstrakci pro přístup k hardwaru a také pomáhá spravovat uživatelské rozhraní pomocí zdrojů aplikace. Obvykle poskytuje služby, pomocí nichž můžeme vytvořit konkrétní třídu a udělal tuto třídu užitečnou pro vytváření aplikací. [2]

Dalo by se tedy říci, že se pro vývojáře jedná o nejdůležitější vrstvu architektury, jelikož umožňuje přistupovat k různým službám, které se dají používat přímo ve vytvořených aplikacích, a tím následně umožňují například přistupovat na prvky graficko-uživatelského rozhraní, používat hardware zařízení, spouštět služby na pozadí nebo používat určité služby zařízení, jako jsou hodiny, budík apod. [1] Navíc tato vrstva obsahuje různé typy služeb, jako jsou správce aktivit, správce oznámení, systém zobrazení, správce balíčků atd., které jsou užitečné pro vývoj aplikací podle požadavků. [2]

Applications

Aplikace představují poslední a nejvyšší vrstvu Android architektury. Tato vrstva zahrnuje jak předinstalované aplikace, jako jsou domovská obrazovka, kontakty, fotoaparát, galerie a další, tak i aplikace třetích stran stažené z obchodu Play, nebo odjinud z internetu, jako jsou chatovací aplikace, hry apod. Veškeré aplikace, bez ohledu na jejich původ, jsou nainstalovány a spouštěny právě na této vrstvě.

Spouštění aplikaci v této vrstvě probíhá v rámci Android runtime za pomocí tříd a služeb poskytovaných aplikačním rámcem. Tato integrace zajišťuje správné fungování a vzájemnou interakci aplikací a služeb v rámci Android ekosystému. [2]

## Vývojové nástroje a prostředí

Vývoj aplikací pro operační systém Android probíhá ve vývojovém prostředí tzv. *Host-Target*. Což je druh vývojového procesu, při kterém je prostředí, kde je aplikace vyvíjena, a prostředí pro realizaci aplikace naprosto odlišné. Jednoduše řečeno je, například v případě vývoje androidových aplikací, aplikace vytvářena ve vývojovém prostředí na počítači za pomocí dalších nástrojů a testování zároveň s implementací aplikace probíhá na jiném zařízení, ať už mobilní telefon, tablet atd. [1] Existuje samozřejmě možnost využití emulátorů a virtualizace, kde androidovský operační systém spustíme na počítači s například operačním systémem Windows, ale v takovém případě vyvstávají komplikace v podobě ku příkladu špatné kompatability v případě starších verzí OS, omezenému přístupu k hardwaru a nedostačujícímu výkonu a celkové zbytečné spotřeby zdrojů. Obecně je tedy možné provádět základní testování a ladění aplikací za pomocí emulátorů, ale vždy je lepší a dalo by se říct i nutné vytvořenou aplikaci testovat na zařízení a v prostředí, ke kterému byla designována. V našem případě tedy mobilní zařízení v podobě tabletu.

**[doplnit text podle nástrojů a prostředí, které použiju ve vývoji]**

# Základy Androidových aplikací

Před navrhováním a vývojem jakéhokoliv nástroje je nutné pochopit podstatu oboru a jeho okolí. Jinak tomu není ani při vývoji aplikací. Bez hlubšího porozumnění nelze efektivně identifikovat potřeby a požadavky uživatelů a je nutné pro vybrání klíčových funkcí a vlastností, které by do aplikace měly být zařazeny. Dále proto nahlédneme na nejčastější programovací jazyky používané při vývoji androidových aplikací, na životní cyklus aplikace, konstrukci aplikací, kterou je vhodné se řídit a na další komponenty s aplikacemi a vývojem svázané.

## Programovací jazyky pro Android

Obecně se pro Android používají nejčastěji dva velmi známé a rozšířené programovací jazyky. Těmi jsou Java nebo C++ a dodatečně XML (Extensible Markup Language), který ale ve své podstatě není přímo programovacím jazykem. V posledních letech se také rozšířil programovací jazyk Kotlin.

1. Java:

Java je všeobecný, oběktově orientovaný programovací jazyk poprvé představený v roce 1995 a prošel si od svého zrodu mnoha vývojovými fázemi, címž se stal jedním z nejrozsáhlejších programovacích jazyků na světě. Mezi jeho vlastnosti patří už zmíněná objektová orientace, což znamená, že veškerý kód v Java je organizován do objektů. Dále jazyk podporuje multi-threading, který umožňuje paralelní a asynchronní zpracovávání dat. Ohromnou výhodou jazyka je platformní nezávislost, která umožňuje spouštět kód napsaný v tomto jazyce na různých platformách bez nutnosti jakýchkoli úprav, tzv. Write Once, Run Anywhere. V neposlední řadě obsahuje Java garbage collector, který se stará o správu paměti a “vyhazuje“ nepoužívané objekty, což usnadňuje proces vývoje a zabraňuje úniku paměti kvůli přetečení.

* C++:

C++ je stejně jako Java vysokoúrovňový programovací jazyk, který byl v roce 1979 vyvinut jako rozšíření jazyka C s přidáním objektově orientovaných prvků a jako většina programovacích jazyků si také prošel řadou rozšíření, revizí a aktualizací. Podobně jako Java je tedy C++ také objektově orientovaný a podporuje tedy principy jako dědičnost a třídy. Navíc je tento jazyk znám svou vysokou výkonností, jelikož umožňuje přímý přístup k paměti a zkušeným programátorům tak dovoluje přímou kontrolu nad hardwarovými prostředky. Tato schopnost byla při vývoji ostatních zmíněných jazyků záměřně odepřena především z důvodu bezpečnosti, protože nesprávně řízený přímý přístup k paměti může být pro nezkušeného programátora rizikovým.

* XML:

eXtensible Markup Language není programovacím jazykem v pravém smyslu slova, ale jedná se o takzvaný značkovací jazyk, který slouží k značkování strukturovaných dat a je tedy používán k popisu struktury a označování informací. Značkování dat probíhá v XML vytvořením značek, které jsou ohraničeny závorkamy následovně:

<osoba>

<jmeno> Petr Novák </jmeno>

<vek> 25 </vek>

<povolani> Elektrikář </povolani>

</osoba>

Data jsou tímto způsoben organizována do hierarchické struktury a každá značka může obsahovat další podřadné značky a hodnoty. Standardizací formátu dat se v XML dosahuje pomocí XML schémat nebo také Document Type Definitions. Jelikož je XML platformě nezávislý, používá se často na výměnu dat mezi různými systémy. XML sám o sobě nemá vlastní programovací paradigma a pro manipulaci s daty se tak používají různé programy a knihovny. Jazyky jako například Java podporují nástroje pro práci s XML.

* Kotlin:

Kotlin je relativně nový plně objektový programovací jazyk vydaný v roce 2017 firmou JetBrains a byl navržen tak, aby byl interoperabilní s jazykem Java. Je tím pádem možné integrovat kód z jazyku Java do kódu napsaného v Kotlinu a naopak. Mimo to obsahuje Kotlin například některé funkcionální prvky, jako jsou lambdy výrazy vyšších řádů atd., obsahuje ochranu před velice běžnými chybami spojenými s hodnotami typu null (neexistence hodnoty v proměnné nebo objektu) a umožňuje rozšiřitelnost. Kotlin se stal nástupcem jazyka Java jako oficiální programovací jazyk k vývoji aplikací pro platformu Android.

## Typy aplikací [1]

Nejzákladnější rozlišení aplikací pro operašní systém Android je klasifikace podle toho, jak aplikace interagují s uživatelem a jak pracují ve vztahu k ostatním aplikacím běžících na systému. Podle těchto vlastností se aplikace mohou dělit do třech základních kategorií:

* Aplikace na popředí (Foreground apps)

Jedná se o aplikace, které jsou momentálně aktivní a užitečné pouze v případě, že běží na popředí. Jejich činnost je pozastavena, nebo ukončena, pokud je aplikace dána na pozadí. Zobrazují se na hlavní obrazovce nebo jsou otevřeny v režimu prohlížeče. Tyto aplikace mají přímý přístup ke zdrojům, jako jsou procesor, pamět a síťové funkce.

* Aplikace na pozadí (Background apps)

Tyto aplikace nemusí být aktivní přímo na popředí a nemají přímou interakci s uživatelem. Jejich hlavní činnost probíhá mimo momentální běh zařízení a mohou na pozadí provádět různé úkony, jako je sledování polohy, synchronizace, aktualizace, sledování příchozích hovorů atd. Systém většinou ovlivňuje přístup těchto aplikací k některým zdrojům uvedeným výše, aby šestřil energii a uvolnil výpočetní výkon jiným aplikacím.

* Aplikace s přerušovanou činností (Foreground services)

Aplikace řadící se pod tuto kategorii jsou na popředí, ale mohou provádět různé operace i poté, co uživatel aplikaci opustí a očekává se od nich komunikace s uživatelem skrze upozornění. Často tyto aplikace bývají spojeny se službami (services), které mohou fungovat na pozadí. Příkladem takovýchto aplikací je nejtypičtěji přehrávač hudby, jako Soundcloud a Spotify, které na popředí přehrávají hudbu, i když je zároveň aktivní jiná aplikace.

## Komponenty aplikací

Každá aplikace se skládá z několika komponent, které jsou jejich základními stavebními bloky. Každá komponenta je vstupním bodem, skrze kterou může systém nebo uživatel do aplikace vstupovat a komunikovat s ní. Každý druh má jasný účel a životní cyklus, jenž určuje, jak je komponenta vytvořena a zničena. Některé komponenty mohou být závislé na jiných a vzájemně spolu komunikují. [3]

Aktivity (Activities)

Aktivity představují přezentační vrstvu a umožňují interakci s uživatelem. Jedná se o vizuální komponentu aktuální obrazovky s uživatelským rozhraním. Ku příkladu se jedná o aplikaci Google Drive s několika aktivitami, z nichž jedna ukazuje náhled na soubory v disku, další na vytvoření (uložení) nového souboru a jinou na přecházení mezi jinými náhledy v aplikaci. Typicky je jedna aktivita určena jako hlavní a zobrazuje se uživateli ihned po spuštění. Aktivity by měly být vytvořeny s myšlenkou intuitivnosti a snadnosti použití. Nabízejí také možnost spolupráce s jinými aplikacemi. Například v případě Google Drive možnost přímého nahrání obrázků z galerie přes menu sdílení. [3]

Služby (Services)

Jedná se o části aplikace, které nejsou součástí uživatelského rozhraní, ale provádí operace na pozadí a tedy neumožňují uživatelům přímé ovládání. Fungují jako vstupní bod pro umožňení aplikaci fungování v pozadí pro provádění dlouhotrvajících operací nebo pro provádění práce pro vzdálené procesy.

Přijímače vysílání (Broadcast receivers)

Přijímače jsou jedním ze vstupů do aplikací umožňující systému doručovat aplikacím události mimo klasický uživatelský tok a dovolují tak reagovat na důležitá systémová oznámení. Důležitou vlastností přijímačů je schopnost přístoupit k aplikaci, přestože není právě spuštěna ani na pozadí a provádět k nim vysílání. Této vlastnosti se hojně využívá u mnohých aplikací, přestože samy přijímače vykonávají minimální možství práce. Příkladem jejich použití je kalendářová aplikace, která jej používá k posílání upozornění na napánované události, nebo na online synchronizaci dat s online kalendářem a to vše i v případě, kdy není spuštěna.

Poskytovatelé obsahu (Content providers)

Poskytovatelé obsahu provádějí operace s daty a slouží k jejich sdílení mezi aplikacemi. Aplikace ukládají data do svého souborového systému, do databáze, na web nebo na jiné druhy úložišť, a jelikož neexistuje žádné běžné úložiště, do kterého by měly přístup všechny Androidové balíčky, tak této funkci napomáhají poskytovatelé obsahu. Aplikace si, pokud k tomu má dostatečné oprávnění, jejich prostřednictvím může data vyžádat, nebo je modifikovat.

Unikátní prvek Androidových aplikací je schopnost aplikace vyžádat si určité komponenty a funkce z jakékoliv jiné aplikace. Například může chtít aplikace po uživateli, aby pořídil fotku (bankovní aplikace pro potvrzení identity fotkou občanského průkazu, seznamovací aplikace na vyfocení obličeje atd.). Většinou podobné funkcionality již existují v nějaké jiné aplikaci a nemusíte tedy vytvářet vlastní aktivity a kód pro focení nebo na ně odkazovat, ale pouze vytvoříte aktivitu ve fotoaparátu pro pořízení fotografie. Po pořízení se fotka vrátí aplikaci a lze s ní dále pracovat a pro uživatele se celý proces jeví, jako by fotoaparát byl přímo součástí aplikace.

Jelikož každá aplikace existuje v systému jako samostatný proces s vlastními oprávněními, nemůže aplikace sama přistoupit k obsahu aplikace jiné. O tyto požadavky se stará systém samotný. Aplikace si od něj zprávou se záměrem zažádá o přístup ke komponentu a systém jí reakcí daný komponent aktivuje.

Záměr (Intent)

Záměr, neboli Intent je asynchronní zpráva aktivující tři ze čtyř uvedených komponent: aktivity, služby a přijímače vysílání. Záměry na sebe vážou jednotlivé komponenty a fungují jako poslové, kteří od nich vyžadují činnosti. Jejich vytváření je prováděno přes objekt Intent, který nadefinuje zprávu pro aktivaci konkrétního komponentu, nebo pro aktivaci komponentu nějakého typu.

Záměry, jak bylo uvedeno u poskytovatelů obsahu a jak název napovídá, definují záměr aplikace k provedení nějaké akce, kterou může být třeba přístup k datům nebo k prvku nějaké jiné aplikace.

U aktivit a služeb může záměr sdělovat požadavek k otevření fotoaparátu nebo webové aplikace. V případě přijímačů vysílání většinou záměry značí požadavek k odvysílání nějakého upozornění, jako například upozornění z kalendáře nebo spuštění budíku.

Poskytovatelé obsahu fungují z bezpečnostních důvodů jinak a na rozdíl od aktivit, služeb a přijímačů jsou vyvolávány požadavkem Překladače obsahu (Content resolver), který zpracovává přímé transakce s poskytovateli obsahu.

Android Manifest

AndroidManifest je XML (Extensible Markup Language) soubor, uložený v kořenovém adresáři aplikace (typicky [název aplikace]>manifests>AndroidManifest.xml), který obsahuje informace o různých parametrech aplikace, jako je název a API verze, jednotlivé komponenty, potřebná oprávnění, softwarové závislosti pro použití komponentů jiných aplikací a další požadavky pro správný chod aplikace.

Ukážeme si strukturu jednoduchého souboru AndroidManifest.xml:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.myapp">  <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/AppTheme">  <activity android:name=".MainActivity">  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  <activity android:name=".SecondActivity" />  </application>  <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  </manifest> |

Nyní si vysvětlíme jednotlivé části:

1. Manifest element:
   * ‘<manifest>‘ je kořenový element souboru AndroidManifest.xml.
   * ‘xmlns:android=http://schemas.android.com/apk/res/android‘ slouží pro definování jmenného prostoru android a je povinné pro všechny prvky v manifestu.
2. Package:
   * ‘package="com.example.myapp"‘ definuje jméno balíčku aplikace.
3. Application:
   * ‘<application>‘ obsahuje globální informace o aplikaci a jejích komponentech (dříve zmíněné aktivity, služby, posluchače, poskytovatelé obsahu)
   * ‘android:...‘ jsou atributy pro nastavení vlastností aplikace.
4. Activity:
   * ‘<activity>‘ slouží pro definování jednotlivých aktivit aplikace.
   * ‘android:name‘ určuje název třídy aktivity.
   * ‘<intent-filter>‘ definuje typy záměrů pro každou aktivitu.
   * ‘"android.intent.action.MAIN"‘ říká, že se jedná o hlavní (MAIN) aktivitu a tedy se při spuštění aplikace obslouží jako první a ostatní aktivity jsou na ni dále vázány. Například se může jednat o spuštění hlavní nabídky.
   * ‘android.intent.category.LAUNCHER‘ říká, že aktivita bude zobrazena v seznamu aplikací (ikonka).
5. Permissions:
   * ‘<uses-permission> definuje, jaká povolení potřebuje aplikace k fungování. V tomto případě INTERNET umožňuje aplikaci připojení k síti a ACCESS\_NETWORK\_STATE umožňuje aplikaci získávat informace o stavu sítě

Android API

Android Application Programming Interface (API) je základním stavebním kamenem pro vývoj aplikací pro operační systém Android. Tato sada programových nástrojů je klíčová pro interakci aplikací s jádrem systému a hardwarovými komponenty zařízení. Jedná se o rozhraní, které umožňuje vývojářům využívat širokou škálu funkcí a služeb poskytovaných Androidem, včetně práce s uživatelským rozhraním, sítěmi, databázemi, multimédii a dalšími.

Android API je nedílnou součástí Android SDK, což je balík nástrojů, které vývojáři potřebují k tvorbě aplikací pro Android. Tato knihovna poskytuje různé funkce a abstrakce, které umožňují vývojářům psát kód pro Android aplikace bez nutnosti znalosti detailů implementace samotného systému.

S každou novou verzí operačního systému Android je aktualizována i verze Android API, což přináší nové funkce, vylepšení a opravy chyb. To vede k existenci několika nejčastěji používaných verzí API, které poskytují širokou škálu funkcí a jsou podporovány na velké části moderních zařízeních s Androidem.

Úroveň API

API je číselná hodnota nastavitelná v souboru AndroidManifest.xml, která aplikaci určuje verzi Android SDK. Android SDK a tedy i poskytované nástroje API jsou navrženy tak, aby byly obecně zpětně kompatibilní a vývojáři tedy mohou používat novější verze k vytváření aplikací, které jsou spustitelné na starších verzích Androidu. Pro vývojáře je tedy důležité zvážit aktuální trendy a potřeby uživatelů při výběru verze API pro své aplikace, aby mohli využít nejnovějších funkcí a zároveň zajistit maximální kompatibilitu se zařízeními uživatelů.

# Tvorba Aplikace

Prvním krokem při vytváření jakékoli aplikace je vybrat vhodné integrované vývojové prostředí (IDE), což je specializovaný program, který poskytuje vývojářům sadu nástrojů pro vývoj softwaru v jednom prostředí. Máme na výběr několik možností, jako je populární Visual Studio s použitím pluginu Xamarin, IntelliJ IDEA nebo Android Studio. Pro naše potřeby je Android Studio nejvhodnější volbou, protože poskytuje širokou škálu nástrojů pro usnadnění vývoje, včetně integrovaného asistenta AI ve svých nejnovějších verzích, a podporuje oba nejpoužívanější jazyky pro vývoj aplikací pro Android - Java a Kotlin.

Začneme tedy instalací z oficiálních stránek na adrese <https://developer.android.com/studio>, kde máme na výběr z několika verzí IDE.

## Návrh aplikace

Důležitým krokem před psaním samotné aplikace je rozhodnutí, jak by měla aplikace vůbec vypadat a co by měla dělat. Naše zadání je jasné – vytvořit aplikaci pro měření přesnosti doteku, ale i přes to nám zadání nechává velkou volnost v provedení. V této části si tedy projdeme všechny návrhy, které jsme zvážili, a zhodnotíme jejich výhody a možné problémy, které by mohly nastat během vývoje.

1. Simulátor ovládacího panelu letounu

První představa pro měření přesnosti doteku pro pilotní trenažér je samozřejmě věrná rekreace reálného ovládacího panelu, jaký se používá v skutečných letounech. Taková podoba aplikace by nejen v trenažeru působila věrohodně a profesionálně, ale také by piloty lépe připravila na situaci, kdy budou sedět za reálným ovládacím panelem. V tomto prvotním návrhu jsme narazili na několik problémů, které by vývoj takové aplikace nejen ztížily, ale pravděpodobně i znemožnily. Za prvé pracujeme s Androidovým tabletem malé velikosti, což by vedlo k nečitelnosti tlačítek a celý ovládací panel by mohl být nakonec nesrozumitelný a zbytečný. Na přiloženém obrázku lze vidět fotografii kokpitu pilotního trenažéru pro stíhačku Lockheed Martin F-35 Lightning II, z nějž jsou uvedené problémy zjevné.



Zároveň by takové provedení mohlo potenciálně posunout obsah diplomové práce na utajovaný, což by působilo jasný konflikt se zadáním.

1. „Hra“ s pohyblivým bodem

Jedním z navrhovaných řešení je vytvoření hry s pohyblivým bodem, na který by uživatel musel podle instrukcí klikat. Z těchto kliků by se následně vypočítala přesnost uživatele. Toto řešení by, spolu s rotací trenažeru, mohlo přinést praktickou a vypovídající aplikaci. Nicméně jako první problém tohoto řešení se může ukázat konstantní animace pohybujícího se bodu, která může vyvstávat kvůli hardwarovým limitacím tabletu.

1. Maticová „hra“ s měřením přesnosti dotyku

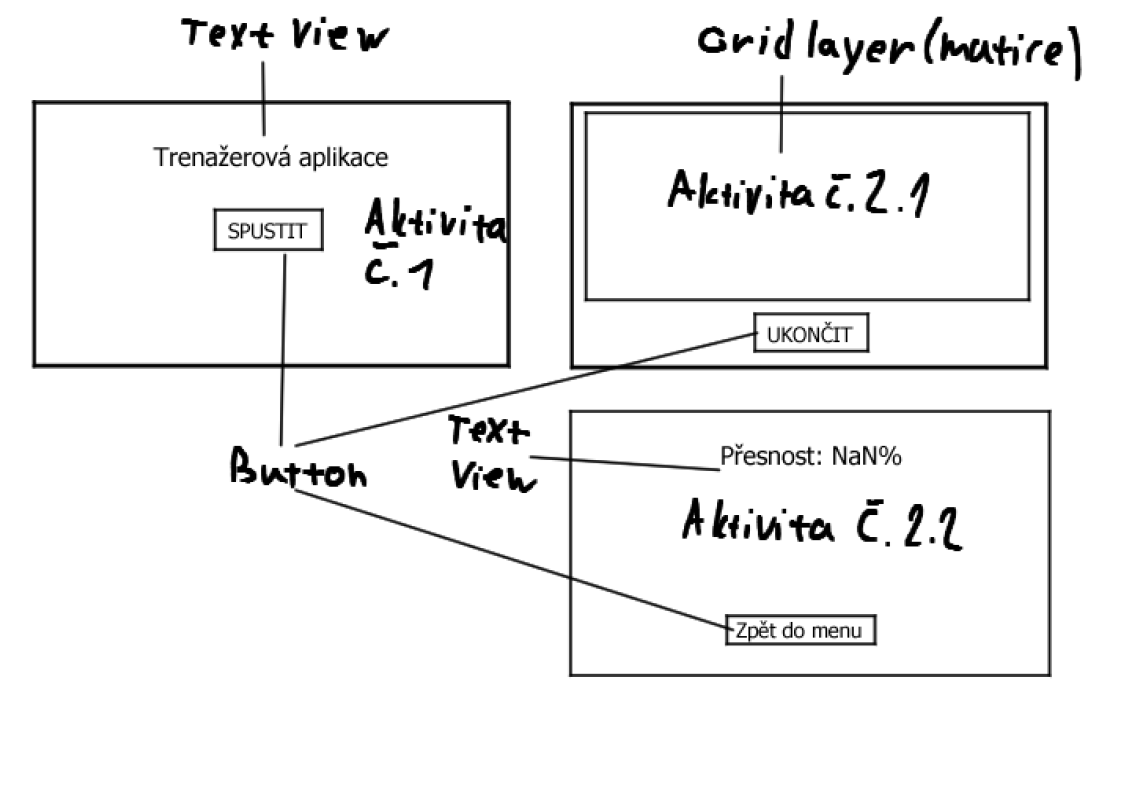
Po konzultaci s Centrem Tělesné Výchovy a Sportu a s kpt. Ing. Ondřejem Machem, který má bohaté zkušenosti s používáním trenažéru jsme došli k několika závěrům:

* Aplikace by neměla být náročná – tablet musí být schopný aplikaci provozovat minimálně několik hodin v kuse bez potřeby nabíjení.
* Aplikace by měla být jednoduchá – bez složitějších instrukcí a návodů by měla být uživatelsky přívětivá, jak pro učitele, tak pro žáka.

Pro inspiraci k naší aplikaci jsme se zaměřili na již používanou aplikaci od firmy Ad-libitum, která trenažery dodává.

**[rozeberu dál, jestli mi konečně dají zdroják]**

Finálním návrhem naší aplikace bude jednoduché a nenáročné provedení. Aplikace bude prezentovat jednu matici postupně se rozsvicujících tlačítek, na které musí uživatel kliknout v určeném pořadí. Po provedení předem určeného počtu kliknutí proběhne výpočet přesnosti, která se zobrazí na finální obrazovce jako procentuální hodnota úspěšnosti.

Po finálním výběru provedení je vhodné si vymyslet, jak na sebe budou navazovat jednotlivé aktivity aplikace a jak budou přibližně vypadat. K tomu nám postačí tužka a papír, nebo můžeme použít digitální prostředky. Tento proces je většinou prováděn dedikovaným designerem aplikace, ale pro vytvoření jednoduché aplikace si vystačíme sami.

Podle popisu výše jsme dospěli k následujícímu návrhu aplikace, ve kterém identifikujeme základní aktivity a prvky aplikace. První aktivitou je menu aplikace, které obsahuje úvodní text a tlačítko pro spuštění hry. Po kliknutí na toto tlačítko se přesuneme do druhé aktivity. V této aktivitě se automaticky vygeneruje herní pole ve formě matice tlačítek s předem určenými rozměry. Na spodní části této aktivity je umístěno tlačítko, které umožňuje uživateli ukončit aktuální hru a vrátit se zpět do menu.

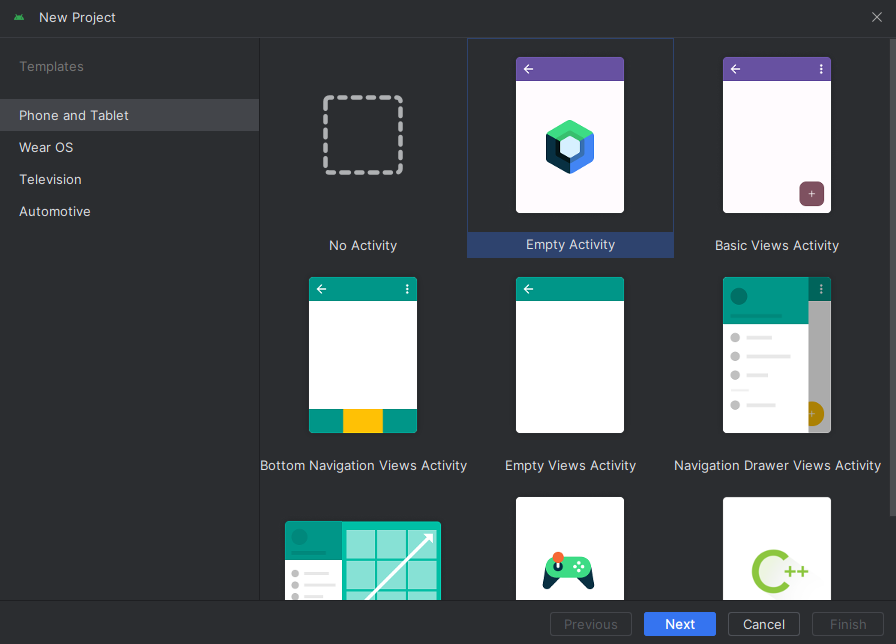
Po dosažení určitého cíle nebo podmínek pro ukončení hry dojde k vyhodnocení přesnosti výkonu hráče, která je vyjádřena v procentech. Kromě toho dojde ke změně textu na tlačítku pro ukončení hry, čímž se zlepší vizuální efekt a uživatelská zkušenost. Bylo by možné tuto druhou část aktivity oddělit do samostatné aktivity, avšak pro zachování přehlednosti a jednoduchosti celého kódu jsme se rozhodli ponechat tyto části společně.

## Tvorba aplikace

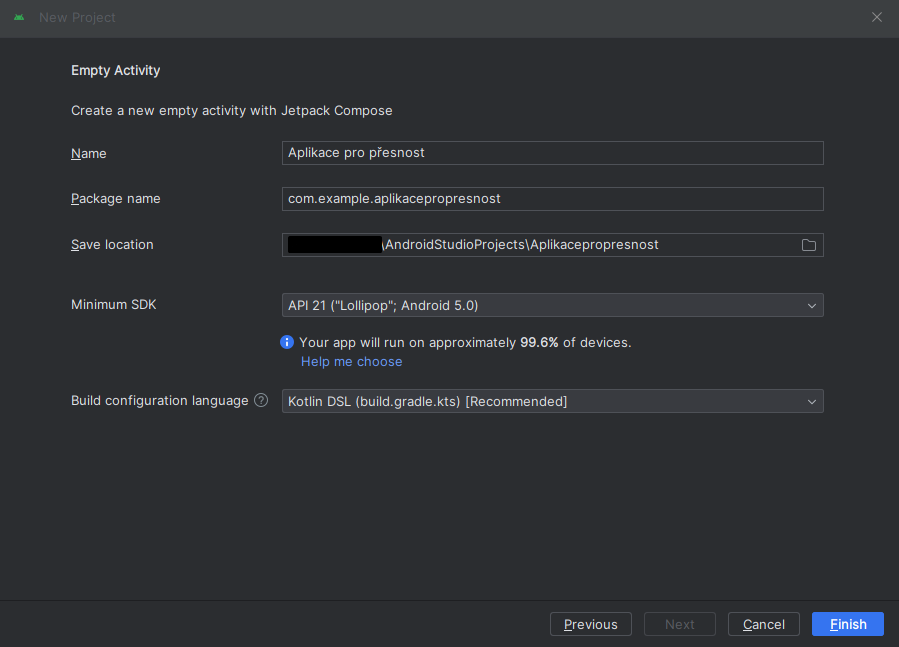
Jak už bylo zmíněno, aplikaci budeme psát v IDE Android Studio. Posledním krokem před začátkem vývoje aplikace je nezbytné pečlivě vybrat vhodný programovací jazyk. V našem případě jsme se rozhodli pro Kotlin, který v současné době nabízí několik výhod oproti svému předchůdci, Javě. Kotlin získává stále více podpory a uznání ze strany vývojářské komunity díky své moderní syntaxi a řadě užitečných funkcí.

Pro úpravu vizuální stránky naší aplikace budeme využívat CSS. Tento jazyk je běžně používán pro formátování webového obsahu, ale jeho flexibilita a možnosti ho činí užitečným nástrojem i při návrhu uživatelských rozhraní pro mobilní aplikace. S pomocí CSS můžeme efektivně definovat vzhled naší aplikace a zajistit, aby byla atraktivní a uživatelsky přívětivá.

Nyní tedy přejdeme přímo k psaní aplikace a v Android Studiu vytvoříme nový projekt.

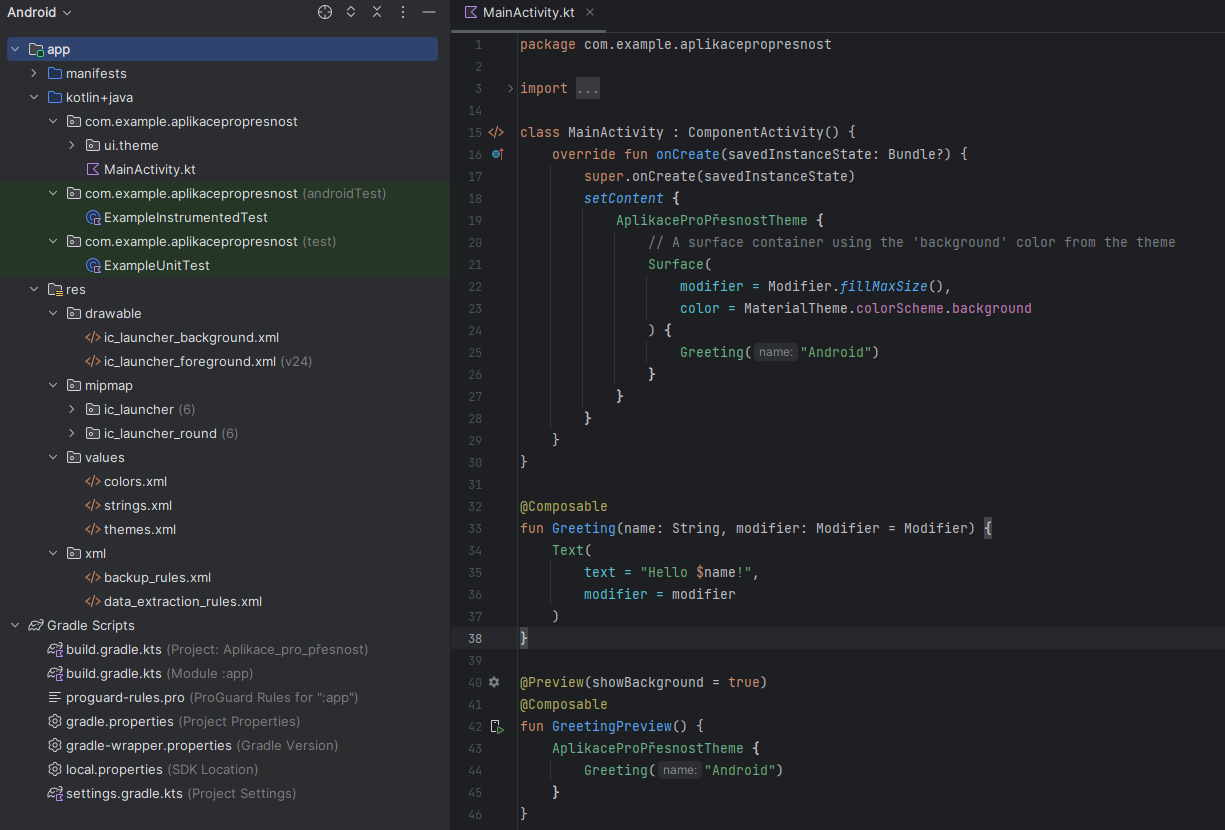


Dostaneme na výběr z několika šablon, což je jedna z mnohých výhod Android Studia v mnohých případech zjednodušuje vývoj. V našem případě ale vytvoříme prázdnou aktivitu.



V následující obrazovce aplikaci pojmenujeme, vybereme umístění a název balíčku a převděpodobně nejdůležitější částí je zde verze API a s tím spojená verze Android OS. V našem případě vytváříme aplikaci pro jeden konkrétní tablet s verzí Androidu 10, ale jelikož jsme k němu dostali přístup až v pozdější fázi vývoje, vybereme verzi Androidu 5, která má nejširší podporu zařízení a stále si uchovává nejdůležitější funkce. Toto by pro nás nemělo být z důvodu jednoduchosti aplikace problémové.

Tímto se nám vytvoří nový projekt, který už bude mít před vytvořené některé části aplikace, jako je hlavní aktivita podle vybrané šablony (v našem případě tedy prázdná) a provede se první synchronizace Gradle a zbytku souborů aplikace.



Jako první se nám po vytvoření projektu otevře hlavní aktivita, jak nám ji Android Studio navrhlo. Pro nás důležité části se nacházejí v první části kódu, kde je deklarace balíčku ‘com.example.aplikacepropresnost‘ a importy, které přinášejí různé třídy a funkce pro použití v kódu. V levé části můžeme vidět ostatní před vytvořené adresáře a soubory:

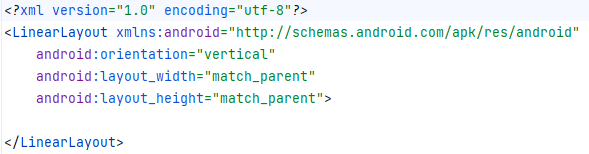
1. drawable: Tento adresář obsahuje obrázky, ikony a jiné grafické soubory používané v aplikaci. V Android Studiu je to běžný způsob, jak organizovat zdroje obrázků podle různých rozlišení displeje a jiných konfigurací.
2. colors.xml: Tento XML soubor obsahuje definice barev používaných v aplikaci. Může obsahovat hexadecimální kódy barev nebo odkazy na barvy definované jinde v aplikaci.
3. strings.xml: Tento XML soubor obsahuje řetězcové zdroje používané v aplikaci. Je to dobrá praxe oddělit řetězcové hodnoty od zdrojového kódu, což umožňuje jednodušší lokalizaci a správu textů ve vaší aplikaci.
4. themes.xml: Tento soubor obsahuje definice témat aplikace. Témata mohou obsahovat nastavení jako barvy pozadí, styly textu, velikosti písma atd. Tento soubor je klíčový pro nastavení vzhledu vaší aplikace.
5. backup\_rules.xml: Tento soubor může obsahovat pravidla pro zálohování dat aplikace. Záloha dat je důležitým aspektem vývoje aplikací, zejména pokud aplikace ukládá důležitá data, která by uživatel neměl ztratit.
6. data\_extraction\_rules.xml: Tento soubor může obsahovat pravidla pro extrakci dat z různých zdrojů. To může být užitečné, pokud aplikace pracuje s daty ze zdrojů jako jsou webové stránky, soubory nebo databáze.
7. build.gradle.kts: Tento soubor obsahuje skripty pro sestavení projektu pomocí nástroje Gradle. Gradle je nástroj používaný vývojáři Android aplikací pro automatizaci procesu sestavení a správu závislostí projektu. Soubor .kts znamená, že se jedná o soubor napsaný v programovacím jazyce Kotlin pro lepší čitelnost a flexibilitu. Sestává se v základní konfiguraci projektů ze dvou částí – projektová a modulová. Do modulové části budeme přidávat nekteré implementace nutné pro správnou funkci aplikace.

## activity\_main.xml

Při psaní kódu si nejdříve pro každou část aplikace projdeme její základní podobu, popíšeme vzniklé problémy při vývoji s jejich řešením a co se změnilo ve finální podobě. Kompletní kód v jeho finální podobě lze nalézt na konci celé práce.

První věc, kterou musíme po vytvoření aplikace udělat, je vytvoření layoutu první obrazovky, což je v našem případě menu. Toho dosáhneme vytvořením adresáře layout pod adresářem res a následně vytvořením tzv. Layout Resource File pod adresářem layouts. Pojmenujeme ho pro přehlednost activity\_main.

Tímto způsobem nám Android Studio vytvoří nový soubor, ve kterém předdefinuje displej s některými základními vlastnostmi.



První řádka definuje verzi XML a kódování dokumentu. Zbytek XML kódu popisuje rozložení uživatelského rozhraní v Android aplikaci pomocí prvku LinearLayout, který organizuje ostatní prvky ve vertikálním směru.

Velikou výhodou Android Studia je možnost grafického zobrazení designu aplikace. Momentálně je zobrazení prázdné, takže se na něj podíváme až vždy ve finální podobě.

Podle nákresu, který jsme si vytvořili při návrhu aplikace přidáme do activity\_main.xml nadpis a tlačítko pro spuštění.



Rozdílem oproti původnímu návrhu je naše použití RelativeLayout, který nám umožňuje definovat vztahy mezi prvky na základě jejich relativní polohy vůči sobě. Použili jsme vlastnosti match\_parent, aby se šířka a výška rovnala rodičovskému kontejneru.

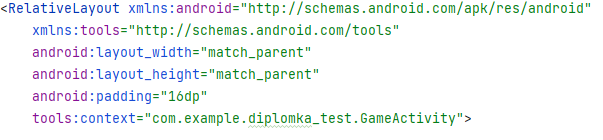
TextView je náš nadpis. Nejdříme mu vytvoříme identifikátor (id), který složí pro jeho pozdějí použití v kódu aplikace. Pro výšku a šířku jsme tentokrát použili vlastnosti wrap\_content, což povede k přizpůsobení textového prvku podle jeho obsahu tzn. zvětší a zmenší se podle potřeby. Velikost písma jsme nastavili na 24sp (scale-independent pixels), což je prvotní hodnota, kterou později podle potřeby změníme. Následně je text, který jsme nastavili podle našeho zadání. Nakonec zarovnáme celý text na střed pomocí layout\_centerHorizontal=“true“.

Button je naše tlačítko pro spuštění další aktivity s hrou. Identifikátorem tohoto tlačítka je startButton. Stejně jako u textu nastavíme výšku a šířku na wrap\_content, takže při jakýchkoliv úpravách na textu tlačítka se nám obal přizpůsobí. Další vlastností je layout\_bellow=“@id/textView“, což nám jednoduše určuje, že tlačítko se bude vždy naházet pod nadpisem a vlastností text=“Spustit“ nastavíme samotný text tlačítka. Opět určíme tlačítku zarovnání na střed displeje a vlastností layout\_marginTop=“32dp“ nastavíme odsazení tlačítka od nadpisu.

Opět musíme myslet na to, že toto rozložení je pouze prvotní a jako u ostatních částí kódu slouží pouze pro zprovoznění a testování. Většina prvků bude později změněna, aby byla přívětivější pro uživatele (velikost textu, velikost tlačítek apod.). Stejně tak i finální forma aplikace může být dále rozšířena a vytváření složitých animací, zvukových efektů a podobně nebudou součástí této práce.

## activity\_game.xml

Po vytvoření menu musíme vytvořit taktéž layout pro herní obrazovku. Stejně jako u menu vytváříme stále pouze soubor v XML, takže se jedná jen o vzhled a logika se nachází v odděleném souboru. Opět vytvoříme Layout Resource File v adresáři layout a pojmenujeme ho activity\_game. Již nebudeme rozebírat původní formu, jelikož je totožná s předchozí.



Začátek souboru je téměř stejný, jako v activity\_main.xml s v vyjímkou tools:context, který v našem případě nedefinuje MainActivity, ale GameActivity a udává nám tím jiný kontext, ve kterém má být návrh rozložení zobrazen. Jinak řečeno, jedná se o rozložení displeje pro budoucí GameActivity, který bude obsahovat kód s logikou.



V tomto souboru můžeme vidět podobné prvky, jako v předchozí části. TextView s id accuracyTextView, jak název napovídá, tentokrát neslouží jako nadpis obrazovky, ale pro zobrazení přesnosti uživatele po ukončení hry. Tento text jsme opět vycentrovali pomocí layout\_centerHorizontal, nastavili layout\_marginTop (prostor mezi prvkem a vrchní hranou obrazovky) na 16dp a ostatní vlastnosti jako v minulém případě. Můžeme si všimnout že Text je inicializován na prázdný řetězec. Pokud bychom vložili text už zde, tak by se nám zobrazoval po celou dobu hry, což by samozřejmě působilo rušivě. Zároveň je tento TextView speciální případ, protože do něj budeme chtít zapsat text až po ukončení hry a vypočítání přesnosti, takže TextView budeme aktualizovat dynamicky v kódu.

Následující GridLayout je nový středový prvek, který je obecně kontejner umožňující uspořádávat položky do řádků a sloupců. Náš prvek s id gridLayout bude po spuštění hry sloužit k uspořádání tlačítek do matice. Jeho šířka a výška jsou také nastaveny na wrap\_content pro přizpůsobení se obsahu a layout\_centerInParent slouží k umístění prvku přesně na střed obrazovky.

Tlačítko s id endButton slouží k okamžitému ukončení hry v jakékoliv fázi a vrácení se do menu. Pomocí alignParentBottom je umístěný na spodku obrazovky a přes centerHorizontal je zarovnán na střed. Nakonec má nastavenou mezeru od spodku obrazovky na 16dp.

## MainActivity.kt

V následující části jsme vytvořili MainActivity s příponou souboru .kt, což indikuje, že se jedná soubor s kódem napsaným v jazyce Kotlin. Tento soubor obsahuje kód a definici chování pro hlavní obrazovku naší aplikace, jejíž rozložení jsme vytvořili v activity\_main.

Soubory s kódem obsahují vždy importy nějakých knihoven dovolující fungování klíčových částí a proto si vysvětlíme všechny námi použité:

* 1. android.content.Intent: Knihovna, která je součástí Android SDK používaná k vytváření intentů a zahájení jiné aktivity nebo aplikace. Používáme k vytvoření intentu pro spuštění GameActivity (viz. následující podkapitola).
  2. android.os.Bundle: Datová struktura sloužící pro předávání dat mezi různými komponentami aplikace. V našem případě použitá v metodě onCreate() jako jeden z parametrů.
  3. android.widget.Button: Import sloužící pro manipulaci s tlačítky.
  4. android.view.View: Je základním stavebním blokem uživatelského rozhraní pro Android a poskytuje základní funkce pro manipulaci s uživatelskými prvky aplikace. Zde tento import konkrétně používáme pro skrytí status baru (vrchní lišta).
  5. androidx.appcompat.app.AppCompatActivity: Import pro třídu AppCompatActivity, která umožňuje podporu pro funkce moderního desihnu na starších verzích Androidu a používáme ji jako základ pro definici hlavní aktivity MainActivity.



Pod importy se nachází již zbytek kódu, jehož všechny části si vysvětlíme také.

Třída (class) MainActivity je odvozena od AppCompatActivity a tím, jak bylo popsáno výše, dědí všechny vlastnosti a metody této třídy, címž získává podporu pro novější designové prvky. Tímto způsobem mohou vývojáři “obejít“ nedostatky starších verzí Android a rozšířit si možnosti.

Metoda onCreate je metoda životního cyklu volána při vytvoření aktivity. V této metodě je inicializováno uživatelské rozhraní a nastavení tlačítka Spustit na hlavní obrazovce.

Zavoláním window.decorView.systemUiVisibility a supportActionBar?.hide zajistíme, aby naše aplikace byla roztáhnuta na celou plochu tabletu. Nejdříve skryjeme stavový panel na vrchní žásti displeje a posléze skryjeme akční panel, pokud je k dispozici..

Příkazem setContentView poté nastavíme rozložení pro tuto aktivity na základě dříve vytvořeného zdroje activity\_main. Z tohoto zdroje následně získáme id tlačítka Spustit před val startButton a nastavíme posluchač setOnClickListener pro toto tlačítko, aby při zmáčknutí spustil metodu startGame.

V metodě startGame vytváříme nový intent, kterým se jednoduše spouští GameActivity. Aktivita je specifikována podle její třídy GameActivity::class.java (třída je java, přestože pracujeme v jiném jazyce) a spouštěna pomocí metody startActivity(intent).+

Toto rozložení hlavní obrazovky je velice jednoduché, avšak pro naše účely plní všechny požadavky. Navrhování a práce na první části kódu byla prováděla s předpokladem pozdější úpravy a rozšíření, ale finální verze se od prvotní nijak značně neliší.

## GameActivity.kt

Na závěr vytvoříme soubor GameActivity, který obsahuje kód s logikou pro hlavní část celé aplikace a pracuje na základě zdroje z *activity\_game*. Jedná se tentokrát o značně dělší kód, takže bude rozpolcen do více obrázků s popisem.

GameActivity obsahuje importy knihoven totožné s MainActivity s několika přídavky:

1. android.graphics.drawable.ColorDrawable: Knihovna umožňující pracovat s barevnými pozadími (drawables) u prvků uživatelského rozhraní.
2. android.widget.GridLayout: Jedná se o layout manager, který dokáže organizovat své potomky do mřížky. Zde tento import samozřejmě používáme k vytvoření matice tlačítek.
3. android.widget.TextView: Import widgetu pro zobrazení textu na obrazovce, který zde využíváme po ukončení hry na zobrazení textu s přesností.
4. kotlin.random.Random: Často používaný import, kdykoliv je třeba generování náhodných čísel, který používáme pro náhodné zapínání tlačítek v matici.

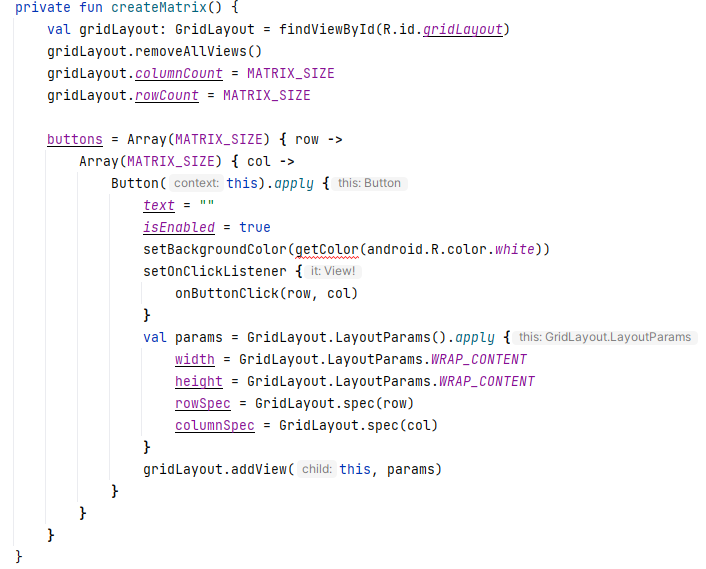


Hlavní částí je třída GameActivity, která je opět odvozena od AppCompatActivity a má tedy stejné rozšířené vlastnosti a metody. Pod touto třídou jsou obsaženy prakticky všechny komponenty kódu s jednou vyjímkou, ke které se dostaneme v následující kapitole.

Pod vytvořenou třídou jsme si definovali dvě konstanty *MATRIX*\_*SIZE* a *MAX*\_*ATTEMPTS*, které podle názvu definují hodnoty velikosti matice a počet pokusů, po kterých se hra ukončí. Pro zjednodušení má matice totožnou výšku i šířku a je nastavena na malý rozměr. Další definované proměnné jsou *buttons* v podobě dvourozměrného pole tlačítek (array) pro vytvoření herní plochy, *totalAttempts* pro počítání pokusů provedených hráčem a *successfulAttempts* pro počítání úspěšných pokusů, přičemž jsou na začátku obě nastavené na nula a aktualizované v průběhu hry a jako poslední proměnné *litButtonRow* a *litButtonCol*, které udržují informaci o tom, které tlačítko je aktuálně rozsvíceno.

Po definování proměnných následuje první metoda *onCreate*. Proměnná *savedInstanceState: Bundle?* slouží pro uchování stavu aktivity v případě, že je aktivita systémem zničena a poté opět vytvořena, což může nastat například při otočení obrazovky. V metodě opět skrýváme vrchní a spodní lištu pro roztáhnutí aplikace a nastavujeme layout aktivity na zdroj *activity\_game*.

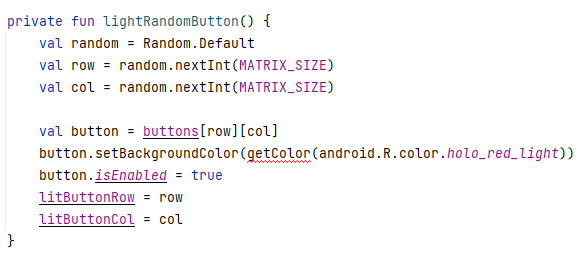
Vytváříme herní pole s maticí zavoláním metodou *createMatrix*, která je vysvětlena níže, nastavujeme posluchač na tlačítku *Ukončit*, které automaticky ukončí hru a zobrazí výsledky a nakonec rozsvítíme náhodné tlačítko zavoláním metody *lightRandomButton*, která je opět rozebrána později.



Následuje již jednou použitá funkce *createMatrix*, která je v třídě GameActivity zodpovědná za vytvoření herní matice, která obsahuje tlačítka, jež uživatel v průběhu hry mačká. Funkce nepřijímá žádné parametry a ani nevrací žádné hodnoty.

Na začátku funkce se pomocí *removeAllViews* odstraňují veškeré tlačítkové prvky z matice, čímž se zabrání překrývání stávajících prvků. Následně použijeme dříve nadefinovanou proměnnou *MATRIX\_SIZE* pro nastavení rozměrů matice. Bylo by v tomto případě samozřejmě možné vytvořit dvě proměnné a podle nich nastavit odlišný počet sloupců a řádků.

Poté jsou pomocí vnořené smyčky vytvořena tlačítka v každém řádku a sloupci matice. Používá se dvou úrovní smyček: jedna pro iteraci přes řádky a druhá pro iteraci přes sloupce. Každé tlačítko je inicializováno vnitřní smyčkou, která projde každý sloupec v daném řádku a vytváří tlačítka pomocí objektu třídy *Button*, nastavuje jejich vlastnosti a přiřazuje jim onClick posluchač. Po vytvoření tlačítka jsou každému přiřazeny nové parametry *params* a každé z nich je přiřazeno do mřížky pomocí objektu *GridLayout.LayoutParams*, který jim určí umístění. Pomocí metody *addView* jsou s použití parametrů tlačítka vložena do mřížky.



Jednoduchá funkce *lightRandomButton* slouží k rozsvicování náhodného tlačítka při spuštění hry a po zmáčknutí příslušného tlačítka. Definujeme si na začátku několik hodnot, které za použití generátoru náhodných čísel namátkově určí sloupec a řadu tlačítka. Poté dojde k zvýraznění tlačítka na námi vybranou hodnotu *holo\_red\_light* a nastavíme jeho stav na aktivní. Nakonec je pozice tlačítka *litButtonRow* a *litButtonCol* uložena do proměnných, což slouží k pozdějšímu ověření, zda uživatel stiskl správné tlačítko.



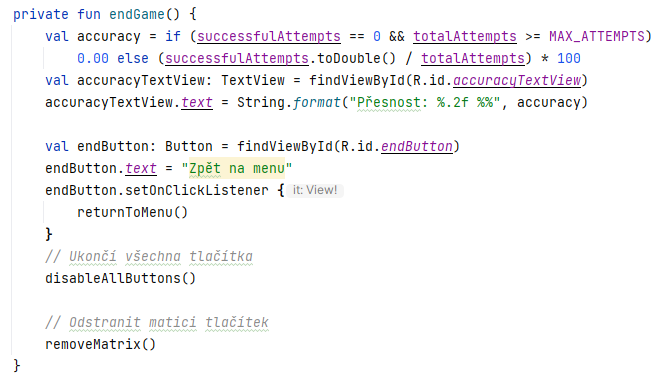
Následující funkce *onButtonClick* se volá po stisknutí tlačítka v herní matici a stará se o zpracování celé akce stisknutí, ověřuje, zda bylo stisknuté správné tlačítko a aktualizuje podle toho stav hry. Funkce přijímá vstupní parametry *row* a *col*, které udávájí pozici stisknutého tlačítka.

Hned na začátku funkce vidíme *totalAttempts++* čímž po každém stisknutí tlačítka hned na začátku zvyšujeme počet provedených pokusů, které samozřejmě nejsou nijak závislé na správnosti stisknutí.

Funkce dále porovnává pozici stisknutého tlačítka (*row* a *col*) s pozicí tlačítka, které bylo předtím zvýrazněno v metodě *lightRandomButton*. Pokud se hodnoty shodují, uživatel zmáčkl správné tlačítko.

Při vytváření aplikace jsme narazili na problém, kdy se stav tlačítka po zmáčknutí neměnil a jeho barva zůstávala stejná a zároveň po stisknutí již nebylo možné tlačítko použít. Proto jsme přešli k radikálnějšímu přístupu a po každém zmáčknutí tlačítka kompletní matici odstraníme a opět vytvoříme. Při menší velikosti matici se to nejevilo jako problémové a nedocházelo ke zpomalování.

Pokud bylo stisknuto správné tlačítko, zvýší se následně počet úspěšných pokusů. Na konci fukce dochází ke kontrole pro ukončení hry. Pokud je počet provedených pokusů menší počtu maximálních pokusů (*totalAttempts < MAX\_ATTEMPTS)* spustí se funkce od začátku a pokračuje se ve hře. Ve všech ostatních případech se spustí funkce *endGame* a dojde k ukončení hry.



Funkce *endGame* je volána po překročení pokusů a poskytuje uživateli zpětnou vazbu na jeho výkon ve hře a umožňuje se přes tlačítko vrátit zpět do menu.

Nejprve je ve funkci proveden výpočet přesnosti uživatele. Pokud nebyl proveden ani jeden úspěšný pokus a zároveň byl proveden alespoň jeden pokus, přesnost je nastavena na 0.00%. Pro ostatní případy je proveden výpočet vydělením úspěšných pokusů celkovým počtem pokusů krát 100, címž dostaneme procentuální úspěšnost. Bylo by možné použít celá čísla, ale pro speciální případy a větší precoznost je počet desetiných míst nastaven na dvě.

Přesnost je pomocí *TextView* zobrazena jako string v textovém poli *accuracyTextView*

Tlačítku *endButton* pro ukončení hry je změněn text na *Zpět na menu* a je mu aktualizován posluchač, aby při stisknutí zavolal metodu *returnToMenu*. Pro jasnost doteď tlačítko volalo metodu *endGame.*

Zároveň je s funkcí provedeno ukončení všech tlačítek a odstranení matice, aby se uvolnila plocha pro zobrazení výsledků.



Poslední částí celého kódu jsou tři jednoduché funkce.

Funkce *returnToMenu* je volána při stisknutí tlačítka pro ukončení hry a jejím úkolem je přesměrovat uživatele zpět na menu. Vytváří novou instanci *Intent* pro přechod na hlavní aktivitu *MainActivity* a zavolá metodu *startActivity*, která intent spustí.

Funkce *disableAllButtons* je volána při ukončení hry a deaktivují se s ní všechna tlačítka v matici. Pomocí vnořené smyčky projde kompletně všechna tlačítka matice a vypnou se nastavením vlastnosti *isEnabled* na *false*.

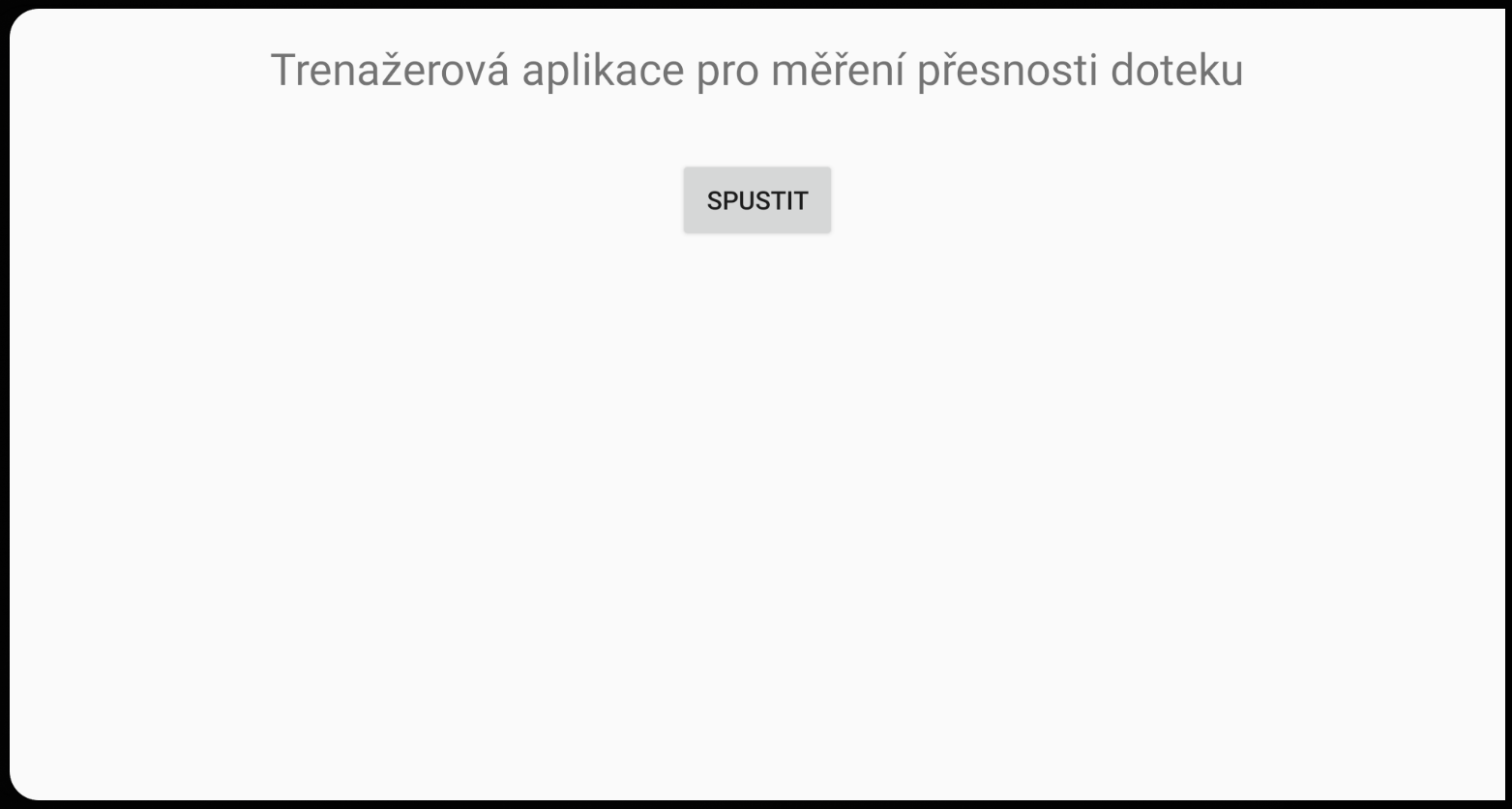
Poslední je funkce *removeMatrix*, která je také volána na konci hry a v této podobě kódu i při přetváření matice, a slouží k úplnému odstranění tlačítek matice. Najde si id matice *gridLayout* a metodou *removeAllViews* odstraní všechny její prvky.

# Testování

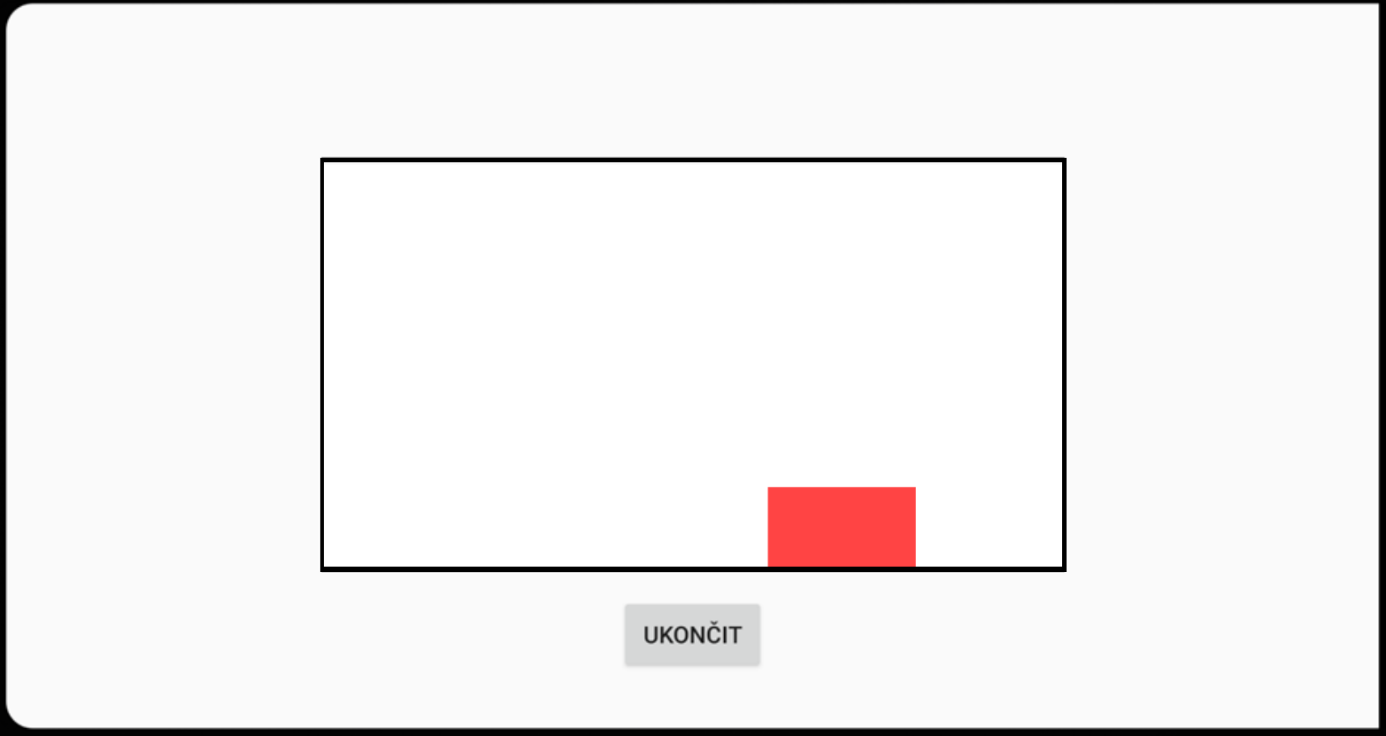
Součástí vývojového procesu každé aplikace je průběžné testování, při kterém ověřujeme správnou funkcionalitu, výkon a uživatelskou přívětivost aplikace. Naším cílem je při testování zajistit, že aplikace splňuje stanovené požadavky, očekávání a v našem případě i zadání.

Při vývoji aplikace bylo provedeno nespočet tzv. ručního testování na Androidovém emulátoru poskytovaném s Android Studiem, kde byla verze a rozměry zařízení vybrány co nejblíže skutečným parametrům tabletu gyroskopického trenažeru. Následně bylo provedeno testování přímo na tabletu, který byl propůjčen Centrem tělesné výchovy a sportu.

Praktické řešení všech problémů bude spolu s ostatními změnami rozvedeno v následující kapitole.



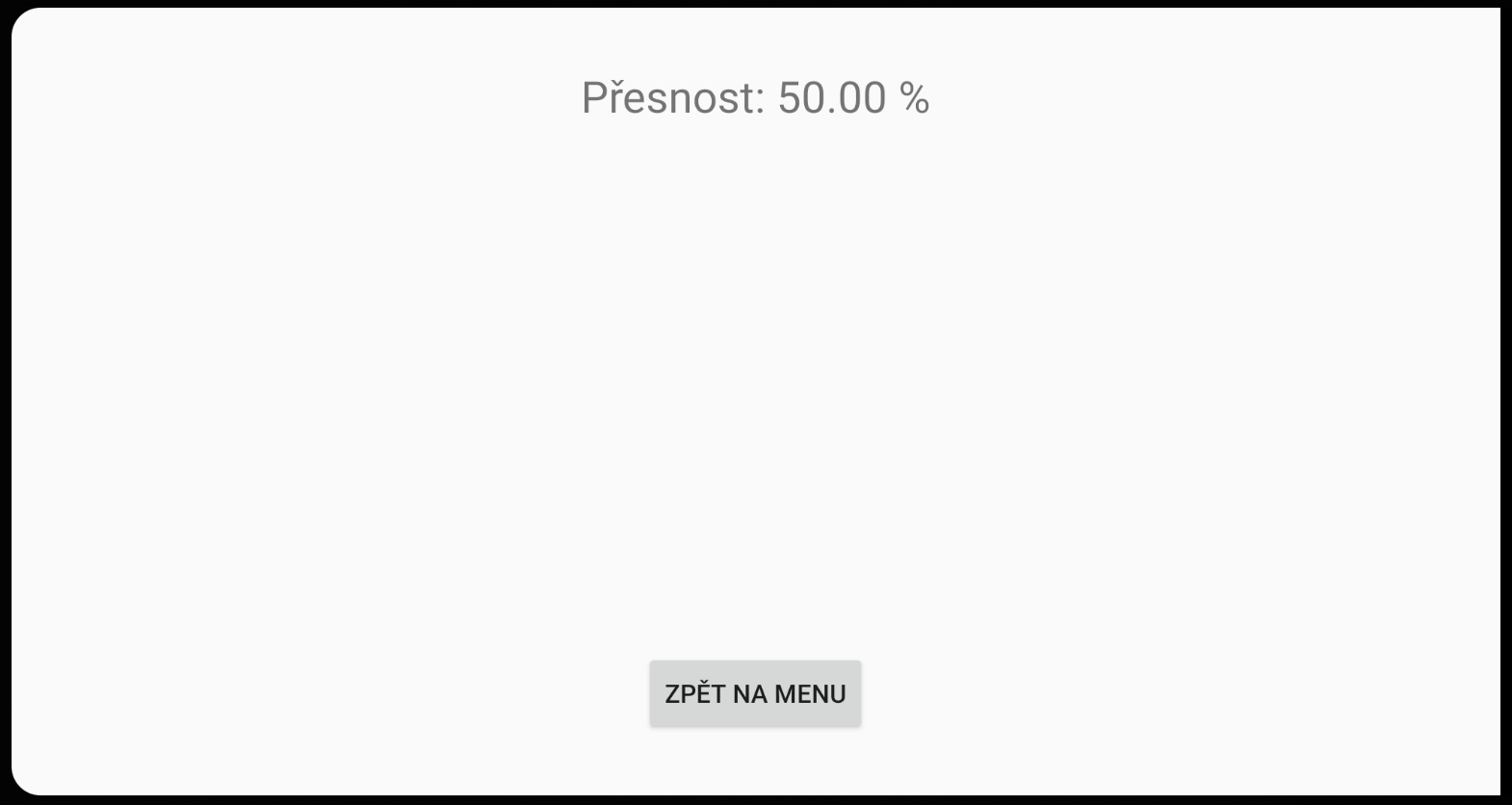
První verze hlavní obrazovky je velmi jednoduchá a neobsahuje proto v podstatě žádné prvky pro testování, s vyjímkou tlačítka pro spuštění, které funguje správně. Její hlavní a víceméně jedinou úlohou je přesměrovat uživatele na samotnou hru. Finální verze hlavní obrazovky se bude odvíjet především od rozšiřování samotné herní části aplikace.



Po stisknutí tlačítka pro spuštění dojde k odstartování samotné hry. Pro lepší čitelnost je matice tlačítek dodatečně černě zvýrazněna. Při vytváření herní matice se pro jednoduchost počítalo s malou maticí, aby byly všechny prvky aplikace snadno viditelné a nic se nepřekrývalo. Proto jsme při testování vyzkoušeli několik velikostí matice a došlo se k závěru, že nejlepší forma bude roztáhnutí na celý displej bez ohledu na počtu tlačítek, címž se dosáhne nejefektivnějšího využití herního prostoru. V závislosti na tomto závěru bude potřeba odstranit tlačítko pro ukončení pro odstranění rušících elementů. Ukončení hry může být nadále prováděno přes spodní lištu tabletu, takže nakonec nedojde k omezení možností.

Při otestování aplikace s různými velikostmi matice vyvstal problém při větších rozměrech, který nebyl ze začátku patrný. Neohrabané řešení dříve zmíněného problému s tlačítky matice ale vede při každém zmáčknutí aktivovaného tlačítka ke zpomalení aplikace, která musí pokaždé vytvořit novou matici. Celý systém generování tlačítek proto projde změnami.

Dalším dodatkem pro herní část bude možnost zvolení několika herních obtížností, které se budou odvíjet od počtu tlačítek v matici. Podle toho dojde i ke změně v hlavní nabídce, ve které se jedno tlačítko pro spuštění změní v několik tlačítek s nabídkou obtížnosti.



Poslední částí aplikace je obrazovka se zobrazením přesnosti. Výpočet přesnosti a taktéž tlačítko pro návrat do menu fungují správně a proto není třeba po technické stránce žádných oprav. Změny budou provedeny převážně pouze po estetické stránce, jako je velikost písma a tlačítka. Jediným vylepšením, které vyplní prázdný prostor obrazovky a navíc rozšíří schopnosti měření aplikace bude přidání časovače, který změří časy mezi jednotlivými stisky uživatele a nakonec zobrazí jeho průměrný čas mezi nimi.

# Tvorba aplikace 2

Poslední kapitola práce se bude věnovat finální formě aplikace, jejímu porovnání s verzí první a řešení problémů zjištěných v předešlé kapitole. Většina změn vycházela z postupného vývoje a testování aplikace a jejich velká část tedy nebyla popsána v předešlé kapitole.

Od první verze byla aplikace rozšířena o několik obtížností, které jsou odlišné velikostí matice, dále možnost volného zadání počtu pokusů, celkový vzhled herní části aplikace (rozšíření matice na celou obrazovku a automatické nastavení velikosti tlačítek) a vzhled ostatních aktivit vycházející z přidaných rozšíření.

Největší změnou ve finální verzi aplikace je možnost v menu „zaregistrování“ uživatele pod přezdívkou, pod kterou se následně ukládají jeho výsledky do textového souboru. Pro funkčnost tohoto vylepšení byla aplikace rozšířena o jednu aktivitu a tři xml soubory s jejich layoutem.

## activity\_main.xml 2

Největší změnou je v layoutu menu nahrazení jednoho tlačítka pro spuštění třemi tlačítky pro různé obtížnosti vnořenými do *Linear layout*, který je řadí vedle sebe.

Dále byly přidány dva formuláře *EditText*, které slouží pro zadání přezdívky uživatele, používané dále pro zapsaní výsledku pod zadaným jménem, a pro volné zadání počtu pokusů ve hře v číselné hodnotě.

Posledním důležitým přídavkem je nové tlačítko pro zobrazení žebříčku výsledků, který uživatele odkáže do nové aktivity *LeaderboardActivity.*

Na spodní část obrazovky bylo přidáno upozornění uživateli pro vypnutí automatického otáčení obrazovky, které by v pohybujícím se trenažéru mohlo působit komplikace při používání aplikace.

## activity\_game.xml 2

V souboru aktivity\_game nebyly provedeny žádné významné změny s vyjímkou velikosti textu a odstranení tlačítka pro ukončení hry, které se nyní zobrazuje pouze na konci.

## MainActivity.kt 2

V aktivitě pro menu byly provedeny změny spojené s přidanými elementy.

K MainActivity byla přidána knihovna *android.widget.EditText*, která aplikaci přidává funkce pro práci s textovými poli (EditText). Tím se umožňuje uživatelům zadávat textové vstupy, kterými jsou v tomto případě uživatelské jméno a počet pokusů.

A screenshot of a computer program

Description automatically generatedMísto jednoduchého spuštění hry pomocí *startGame()* nyní máme tři tlačítka s vlastními posluchači, které předávají jednu ze tří hodnot *matrixSize* odpovídající vybrané obtížnosti. Tlačítko *startButtonEasy* předává hodnotu 3, *startButtonMedium* hodnotu 10 a *startButtonHard* hodnotu 20. Navíc byl přidán posluchač pro tlačítko spouštějící žebříček výsledků.

A computer screen shot of a program code

Description automatically generated

Funkce *startGame* pro spuštění hry byla značně obohacena o několik parametrů předávaných do *GameActivity*. Těmi jsou již zmíněná velikost matice *MATRIX\_SIZE*, *USER\_NAME* s uloženou přezdívkou uživatele a *MAX\_ATTEMPTS* s číselnou hodnotou počtu pokusů.

A close up of a text

Description automatically generated

Posledním dodatkem je funkce, která při aktivování příslušného posluchače spouští aktivitu pro zobrazení žebříčku.

## GameActivity.kt 2

V současné verzi *GameActivity* bylo přidáno několik knihoven pro rozšíření funkcí aplikace:

1. android.content.Context: Obecně slouží k poskytování přístupu k systémovým informacím a prostředkům aplikace. V této třídě poskytuje přístup k zdrojům, jako jsou rozměry displeje, umožňuje spouštění nových instancí *Intent* a je také využita v metodě pro ukládání výsledků do souboru.
2. android.graphics.Color: Použita k manipulaci s barvami, konkrétně k nastavení barvy pozadí tlačítek.
3. android.os.SystemClock: Knihovna poskytující metody pro práci s časem a hodinami. Umožňuje nám měření měření času mezi kliknutími, z kterého je následně uživateli vypočítáná jeho průměrná reakční rychlost.
4. java.io.\*: Zahrnuje několik knihoven (hvězdička značí všechny knihovny pod java.io) s nástroji pro práci s vstupem výstupem v jazyce Java (Kotlin). Opět nám pomáhá při práci se soubory a zapisování do nich.

Dále budou uvedeny změny a vylepšení v poslední verzi *GameActivity*.

Dříve definovaná proměnná *MATRIX\_SIZE* na začátku kódu je nyní posílaná z *MainActivity*. Proměnné *litButtonRow* a *litButtonCol* pro pozdější vybrání umístění rozsvíceného tlačítka byly nahrazeny jedinou proměnnou *litButton* k uchování reference na aktuálně rozsvícené tlačítko. Přidány byly hodnoty *maxAttempts* a *userName* pro uchování hodnot pokusů a uživatelského jména a hodnoty *startTime*, *totalTime* a *clickCount* pro výpočet reakčního času uživatele.

A screenshot of a computer code

Description automatically generated

Metoda *onCreate* získává z *MainActivity* hodnoty *USER\_NAME, MATRIX\_SIZE* a *MAX\_ATTEMPTS*, které si ukládá do dříve nadefinovaných proměnných.

Metoda pro vytvoření matice tentokrát přijímá hodnotu *matrixSize*.

Nakonec proměnná *startTime* získává hodnotu času pro pozdější výpočet.

A screenshot of a computer program

Description automatically generated

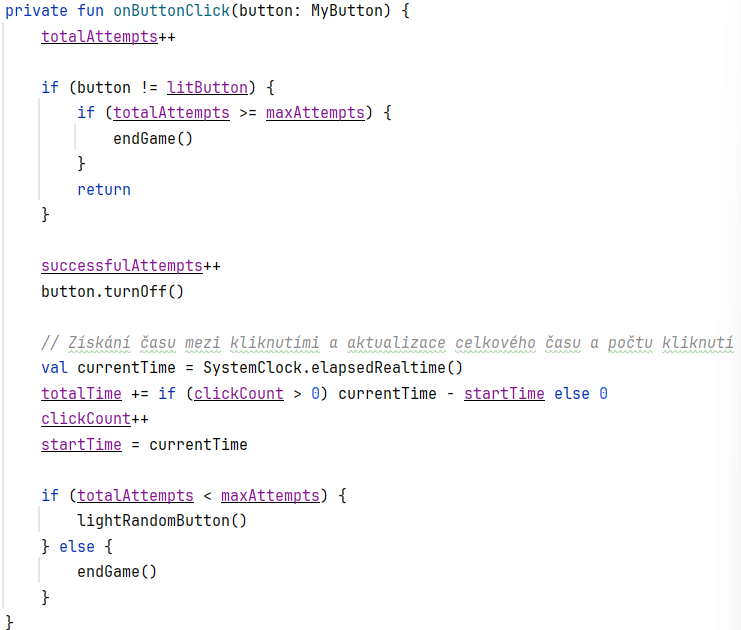
V metodě *createMatrix* nově získáváme od systému rozměry displeje a z nich vypočítáváme rozměry tlačítek, aby nehledě na jejich počet správně vyplnily obrazovku. Pro rezervu je od výšky a šířky displeje odečítáno 40 pixelů.

Vytvoření tlačítek bylo lehce zjednodušeno a posluchač je nyní nastaven s odkazem na aktuální tlačítko (this).

A screenshot of a computer code

Description automatically generated

Metoda pro rozsvícení už nenastavuje sama barvu tlačítka, ale používá metodu *lightUp*, která je součástí třídy *MyButton.*



Metoda *onButtonClick* byla podstatně osekána a řeší problém se spomalováním aplikace při větších rozměrech matice a opakované mazání a přetváření matice proto bylo úplně odstraněno a nahrazeno metodou pro vypnutí jednoho tlačítka, matice jako samotná tak po celou dobu od jejího vytvoření zůstává.

Vstupním parametrem pro metodu je namísto řádku a sloupce instance třídy *MyButton*.

A computer screen shot of text

Description automatically generated

Metoda *endGame* byla naopak viditelně rozšířena a obohacena o několik funkcí.

Jediná instance pro mazání matice *removeMatrix* je zde umístěn na začátku metody.

Výpočet přesnosti se už neprovádí v této metodě, ale byl přemístěn do vlastní metody *calculateAccuracy,* která je obsahově v podstatě totožná, takže nebude uvedena.

Nově vypočítává metoda průměrný čas kliknutí *averageTimePerClick* a pro zobrazení jej převádí do textové formy.

Největším rozšířením je ukládání výsledků uživatele a jejich ukládání pomocí třídy *DataManager*.

A screenshot of a computer program

Description automatically generated

Nově vytvořená třída *MyButton* slouží pro implementaci tlačítek a má dědičnost od třídy *AppCompatButton*, která obsahuje vlastnosti pro standardí tlačítko. Dále má kontruktor s parametry Context, row (řádek) a col (sloupec) tlačítka.

Obsahuje metody pro incializování (init) s výchozí bílou barvou, rozsvícení (*lightUp*) s červenou barvou a zhasnutí (*turnOff*) tlačítka zpět na bílou.

A screenshot of a computer code

Description automatically generated

Poslední novou částí je singleton objekt *DataManager* sloužící pro operace sloužící se správou dat, konkrétně načítání a ukládání výsledků.

Konstanta *FILENAME* určuje název souboru, do kterého jsou data ukládána. Třída *UserResult* obsahuje reprezentace výsledků s atributy *userName, matrixSize,* atd.

Metoda *saveResultsToFile* otevírá soubor pro zápis a vrátí OutputStream, do kterého lze poté zapisovat. Data jsou v textovém formátu a oddělována čárkou. Pro vzniklé vyjímky je použit blok try-catch, který vypíše trasovací informace vyjímky.

## activity\_leaderboard.xml a item\_leaderboard\_header.xml

Při postupném rozšiřování aplikace vznikla potřeba přidání nové třídy s aktivitou pro obsluhování žebříčku s výsledky uživatelů. Stejně tak byly přidány layouty pro vzhled této aktivity, kterými jsou activity\_leaderboard se samotným rozvržením a item\_leaderboard\_header sloužící pro upravení vzhledu tabulky s výsledky.

A screenshot of a computer code

Description automatically generated

Layout obsahuje rozložení pro tzv. spinner, který definuje drop-down nabídku pro změnu řazení tabulky.

A screenshot of a computer program

Description automatically generated

Problém s razením výsledků byla skutečnost, že od sebe nebyly odlišeny různé obtížnosti, takže se výsledky her „Jednoduchá“, „Střední“ i „Těžká“ řadily mezi sebe nehledě na použitou velikost matice. Tento problém byl vyřešen přidáním hodnoty velikosti matice do výsledků a vytvoření tlačítek, které zobrazí výsledky pouze z vybrané obtížnosti. Tlačítka jsou vložena do LinearLayout, takže jsou seřazena vedle sebe.

A screenshot of a computer program

Description automatically generated

Spodní část obrazovky obsahuje tlačítko pro vymazání výsledků, které po stisknutí otevírá textový soubor a maže jeho obsah.

První a nejdůležitější částí celého layoutu je tzv. recyclerView, který je v podstatě tabulkou, do které se vypisují výsledky z textového souboru podle vytvořené logiky. Prvek recyclerView automaticky obsahuje funkci scrollování, takže nehrozí zaplnění tabulky při větším počtu řádků.

Soubor *item\_leaderboard\_header* obsahuje pouze čtyři prvky TextView v LinearLayout s texty „Jméno“, „Počet Pokusů“, „Úspěšné Pokusy“ a „Průměrný Čas“, a slouží jako záhlaví pro celou tabulku.

## LeaderboardActivity.kt

Úplně poslední částí aplikace je samotná aktivita pro obsluhování žebříčku, jejíž layout byl popsán výše. Podíváme se na několik nových knihoven, které v ní byly použity:

1. android.view.LayoutInflater: Používá se v Androidu k naplnění XML layoutů do odpovídajících objektů v běžící aplikaci. Používáme ho v relaci s *aktivity\_leaderboard*, kde v metodě *OnCreate* inicializujeme různé prvky uživatelského rozhraní.
2. android.view.ViewGroup: Základní třída pro vytváření skupin prvků uživatelského rozhraní. V naší aktivitě je zodpovědná za zobrazení uživatelů v žebříčku.
3. androidx.recyclerview.widget.RecyclerView: Nástroj pro zobrazování většího objemu dat v seznamu, mřížce, apod. Používáme při zobrazování seznamu uživatelů.
4. androidx.recyclerview.widget.LinearLayoutManager: Opět nástroj pro zobrazování dat v nějakém seznamu, konkrétně u nás je jeho instance správcem rozložení pro RecyclerView.
5. android.widget.Toast: Umožňuje pro uživatele zobrazit krátkou zprávu jako pop-up. Zobrazuje nám zprávu při úspěšném vymazání výsledků ze souboru.
6. android.widget.ArrayAdapter: Adaptér pro převádění dat z datového zdroje na zobrazení položek v rozhraní. Používáme jej pro naplnění dat do spinneru pro výběr způsobu řazení žebříčku.
7. android.widget.AdapterView: Poskytuje obecné metody pro práci s daty a jejich zobrazování v rozhraní. Opět používáme při naplňování tabulky v RecyclerView.
8. android.widget.Spinner: Umožňuje použít drop-down menu (spinner), což je prvek uživatelského rozhraní, který dovoluje uživatelům vybírat hodnoty z rozbalovacího seznamu.

Aktivita žebříčku neplní pouze funkci zobrazování dat; musí mimo to být schopná číst data z otevřeného souboru, vložit je správně do tabulky, seřadit je podle pokynů uživatele a rozlišovat různé obtížnosti, podle kterých buď data do tabulky vloží, nebo je vynechá. Navíc aktivita obsahuje funkci kompletního smazání textového obsahu souboru.

A screenshot of a computer program

Description automatically generated

Na začátku třídy inicializujeme několik proměnných pro pozdější použití. Proměnná *recyclerView* slouží pro zobrazení kontejneru (seznamu) s výsledky. Správce rozložení *layoutManager* určuje, jak budou položky v tabulce uspořádány. Data o uživatelích uspořádává *leaderboardAdapter*, který je připravuje a podle pokynů zobrazuje. Poslední lateinit variable je *sortBySpinner*, který poskytuje funkčnost widgetu se spinnerem. Informace o velikosti matice, které se později zapisují do textového souboru jsou později ukládány do proměnné *currentMatrixSize*.

V metodě *onCreate* na začátku klasicky provádíme základní nastavení aktivity a nastavujeme layout podle souboru *aktivity\_leaderboard*. Dále proběhne inicializace RecyclerView pomocí ID, vytváříme nový LinearLayoutManager a přiřazujeme layout manager k RecyclerView. Následně najdeme podle ID Spinner a přiřadíme ho k příslušné proměnné a voláme metodu *setupSpinner* pro nastavení posluchačů.

A screenshot of a computer program

Description automatically generated

V druhé části třídy je provedeno nalezení tlačítek podle jejich ID v layoutu, jejich přiřazení proměnným a následné přiřazení posluchačů těmto tlačítkům. Tlačítka plní funkci filtrace výsledků a jejich smazání.

Poslední metoda obstará zobrazení všech výsledků, když není vybrána obtížnost.

A screenshot of a computer code

Description automatically generated

Metoda *loadAllUsers* použitá právě po spuštění posluchačů načítá pomocí metody *loadUsersFromStorage* data výsledků, předává je do LeaderboardAdapteru a následně nastavuje adapter pro RecyclerView pro jejich zobrazení.

Další metoda *loadUsersByMatrixSize* načítá opět data s výsledky, které filtruje podle obtížnosti (velikosti matice). Dále funguje jako předešlá.

Metoda *clearResults* si otevírá soubor s výsledky a buď jej maže, pročemž se zobrazí toast zpráva „Výsledky byly úspěšně smazány“, nebo zachytí vyjímku a zobrazí zprávu „Nastala chyba při mazání výsledků“.

A computer screen shot of a program code

Description automatically generated

Již použitá metoda *loadUserFromStorage* má za úkol načítat data v podobě textu ze souboru a vracet je ve formě seznamu. Nejprve je vytvořen prázdný seznam users pro postupné zapisování dat. Poté je zbytek uzavřen do try-catch bloku, který bude postupně monitorovat proces načítání. V tomto bloku je otevřen soubor s výsledky a vrácený InputStream je zpracován přes třídu BufferedReader, která umožňuje postupné čtení po řádcích. Jednotlivé části řádku jsou vidět v kódu a rozděleny jsou čárkou.

V případě, že soubor neexistuje, vyvolá se vyjímka, zachytí se do bloku catch a není provedena žádná operace.

Nakonec je seznam s daty vrácen z metody.

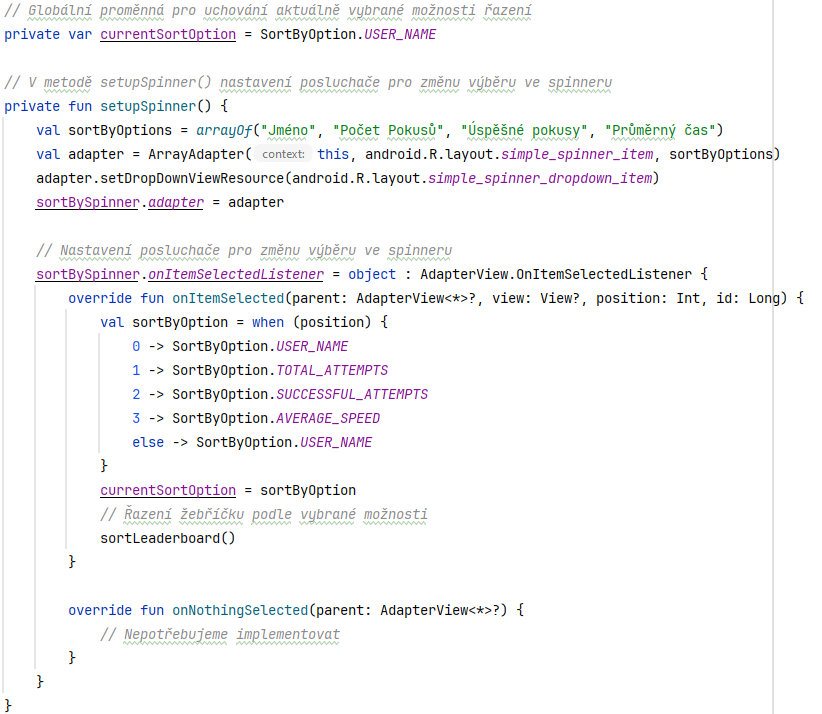
A screenshot of a computer program

Description automatically generated

Třída *LeaderboardAdadapter* je adapter pro propojení seznamu uživatelů s RecyclerView, který jej zobrazuje.

Konstruktor přijímá seznam uživatelů, který má poté RecyclerView zobrazovat. Metoda *onCreateViewHolder* je volána, když RecyclerView potřebuje při operacích vytvořit nový view holder, tedy buňku s jednou položkou ze seznamu. Metoda *onBindViewHolder* se volá při plnění nově vytvořeného view holder. Přijímá vytvořený *LeaderboardViewHolder* a index položky v seznamu. Metoda *getItemCount* zjišťuje počet položek v seznamu a určuje, kolikrát bude volána metoda *onBindViewHolder*.

Vnitřní třída *LeaderboardViewHolder* obsahuje odkazy na jednotlivé položky v seznamu a metodu bind, která určuje obsah těchto prvků.

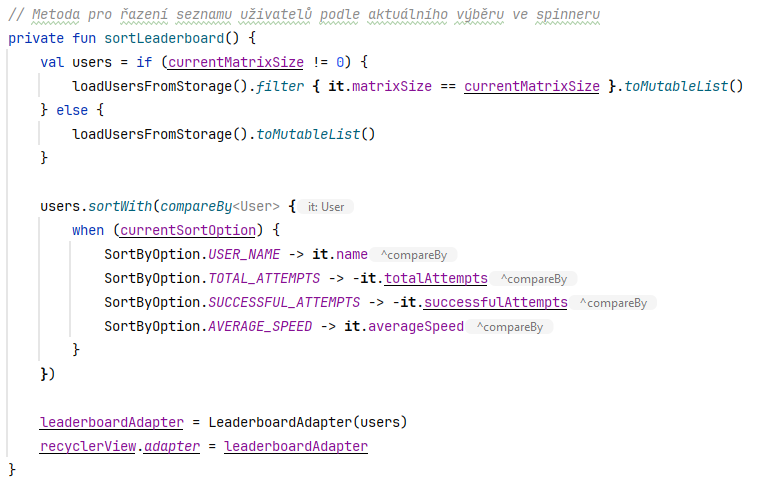


Metoda *setupSpinner* slouží k inicializaci a nastavení spinneru, který slouží k řazení položek v tabulce s výsledky.

Globální proměnná *currentSortOption* uchovává aktuální vybranou možnost spinneru a její výchozí řazení je podle USER\_NAME, neboli podle jmen uživatelů.

Samotná metoda začíná vytvořením pole řetězců s hodnotami Jméno, Počet Pokusů, atd., které reprezentují možnosti řazení. ArrayAdapter vytváří adapter, který propojí pole se spinnerem. Vzhled jednotlivých položek spinneru je nastaven přes *setDropDownViewResource* na základní vzhled (simple\_spinner\_dropdown\_item) a nakonec je kompletní adapter přiřazen.

Následuje nastavení posluchače událosté, které je v tomto případě o něco složitější, než jak tomu bylo u klasických tlačítek. Přes *sortBySpinner.onItemSelectedListener* je třeba nastavit posluchače pro každou položku ve spinneru metodou *onItemSelected*. Proměnná *sortByOption* rozlišuje, jaká z variant spinneru byla vybrána, následně ji ukládá do globální proměnné *currentSortOption* a metoda *sortLeadeboard* poté řadí žebříček podle vybrané možnosti. Metoda *onNothingSelected* v momentální verzi neplní žádnou úlohu, ale byla zanechána pro případné rozšíření a je volána, pokud není vybrána žádná z možností



Metoda *sortLeaderboard* slouží v předchozí metodě na seřazení tabulky podle vybrané možnosti spinneru. Metoda nejdříve rozlišuje momentálně vybranou obtížnost podle velikosti matice a zobrazuje pouze ty výsledky. Bez vybrané varianty zobrazuje všechny. V druhé části provádí řazení dle vybrané možnosti spinneru podle nastavených kritérií; USER\_NAME jsou řazeny abecedně, hodnoty TOTAL\_ATTEMPTS i SUCCESSFUL\_ATTEMPTS mají obě řazení záporné a řadí se tedy od největší a hodnoty AVERAGE\_SPEED mají řazení kladné, tedy budou postupně řazeny od nejmenší.

Nakonec metody je vytvořen nový adapter s aktuálně seřazenými uživateli a je nastaven pro *recyclerView*, čímž se aktualizuje zobrazení žebříčku.

# 

# ZÁVĚR

Je nutné věnovat stejnou, ne-li větší pozornost jako úvodu. V závěru musíte podat shrnutí, vyzvednout nejdůležitější postřehy a také vyhodnotit, zdali jste naplnili cíle, které jste si stanovili v úvodu své práce.

# SEZNAM POUŽITÉ LITERATURY (vzor)

**MONOGRAFIE**

1. ANTUŠÁK, Emil. *Strategie a ekonomika v bezpečnostním systému České republiky*. 1. vyd. Praha: Ministerstvo obrany České republiky, 2002. s. 141 – 180. ISBN 80-7278-143-x.
2. BRČÁK, Josef. *Česká republika ve světle ekonomických teorií*. 1. vyd. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, 2012. s. 78 – 150. ISBN 978-80-7380-369-8.
3. KADEŘÁBKOVÁ, Anna. *Základy makroekonomické analýzy: růst, konkurenceschopnost, rovnováha*. 1. vyd. Praha: Linde, 2003. 175 s. ISBN 80-86131-36-x.

**ELEKTRONICKÉ ZDROJE**

1. BALCAROVÁ, Pavlína. *Konkurenceschopnost ekonomiky a její měření* [online]. Brno, 2007 [cit. 2014-12-30]. Dostupné z: ˂<http://is.muni.cz/th/75962/prif_m/diplomova_prace.pdf>˃. Diplomová práce. Masarykova univerzita.
2. Česká republika: hlavní makroekonomické ukazatele. *Český statistický úřad* [online]. 2014 [cit. 2015-01-28]. Dostupné z: ˂[https://www.czso.cz/documents/10180/20555311/HLMAKRO.xls](http://www.czso.cz/csu/redakce.nsf/i/cr:_makroekonomicke_udaje/$File/HLMAKRO.xls)˃.

# SEZNAM POUŽITÝCH ZKRATEK

|  |  |
| --- | --- |
| AČR | Armáda České republiky (→ Styl seznam zkratek) |
| ISO | International Organization for Standardization  Mezinárodní organizace pro standardizaci |
|  |  |
|  |  |

# SEZNAM OBRÁZKŮ A TABULEK

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Strana |
| Obrázek č. 1 | − | Popis obrázku (→ styl Odstavec) | … XX |
| Obrázek č. 2 | − | Popis obrázku | … XX |
|  |  |  |  |
| Tabulka č. 1 | − | Popis tabulky | … XX |
| Tabulka č. 2 | − | Popis tabulky | … XX |

# SEZNAM PŘÍLOH

|  |  |  |  |
| --- | --- | --- | --- |
| Příloha č. 1 | − | Popis přílohy (→ styl Odstavec) |  |
| Příloha č. 2 | − | Popis přílohy (→ styl Odstavec) |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

(→ Styl Příloha – číslování) Příloha 1

Popis přílohy (→ styl Příloha – popis)

|  |
| --- |
|  |