UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Ludek Tyler Derner

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Vývoj mobilní automatické klikací hry pro platformu Android

Bakalářská práce

2024 Ludek Tyler Derner

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Luděk Tyler Derner**
Osobní číslo:	**I21136**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Vývoj mobilní automatické klikací hry pro platformu Android**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je navrhnout a implementovat automatickou klikací hru pro platformu Android. Cílem hry je rozbít zeď za pomocí různých nástrojů. Dále bude implementován žebříček nejrychlejších časů. Toto bude implementováno za pomocí serveru, data z něj budou dostupná přes API.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

OSE, S. – KUNDU, A. – MUKHERJEE, M. – BENERJEE, M. A comparative study: Java vs Kotlin programming in android application development. International Journal of advanced research in computer science, Volume 9, No. 3, ISSN 0976-5697
HARKIRAN, Kaur. Top Programming Languages for Android App Development Dostupné z: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/
Joseph Ingeno, Software Architect's Handbook, Birmingham Packt Publishing Ltd., ISBN 978-1-78862-406-0
Kotlin documentation, Dostupné z https://developer.android.com/kotlin

Vedoucí bakalářské práce: **Ing. Martin Pozdílek, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem Vývoj mobilní automatické klikací hru pro platformu Android jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 4/2021 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 31. 3. 2024

Ludek Tyler Derner

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce Ing. Martinu Pozdílkovi, Ph.D. za odborné vedení a velice přínosné konzultace. Také bych chtěl poděkovat všem, kteří mě neustále podporují a jsou mou oporou.

ANOTACE

Tato práce se zabývá vývojem mobilní aplikace pro platformu Android. Jedná se o automatickou klikací hru a jejím cílem je zničit zeď. Aplikace je tvořena v Android Studiu za pomocí programovacího jazyka Kotlin. Uživatel má také možnost uložení svých dat po přihlášení přes Google za pomocí Firebase.

KLÍČOVÁ SLOVA

Mobilní aplikace, Android, Kotlin, JSON, Firebase, hra

TITLE

Development of a mobile auto-clicking game for the Android Platform

ANNOTATION

This work deals with development of an application for the Android platform. It is a auto-clicking style game and the purpose is to break the wall. Application is created in Android Studio with the help of the programming language Kotlin. User is also able to save their data after logging in with a google account thanks to the Firebase.

KEYWORDS

Mobile application, Android, Kotlin, JSON, Firebase, game

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API	Application Programming Interface
APS	Antimatter per second
CPS	Cookies per second
MVC	Model-View-Controller
OOP	Objected Oriented Programming
SDK	Software Development Kit
XML	Extensible Markup Language

ÚVOD

Každý z nás se určitě setkal s hrami na mobilních telefonech, ať už je to Snake na Nokii 3310, nebo Fruit Ninja na moderních chytrých mobilních zařízeních. Mobilní hry, jako nové technologie, se staly součástí našich životů. Pomáhají nám krátit dlouhé chvíle, například při čekání na autobus, či nás zabavit na hodiny při dlouhých zimních večerech.

Mezi mé oblíbené hry se řadí „automatické klikačky“, které spadají pod žánr incremental. Tento žánr mi přijde na mobilních zařízeních stagnující a bez nových inovací. Proto bych velice rád přidal nový náhled na tento žánr mobilních her.

Cílem této bakalářské práce tedy je vytvoření mobilní hry pro systém android, která bude novým náhledem na žánr incremental. Hra je tvořena v Android Studiu a k tomu byl zvolen programovací jazyk Kotlin. Hra disponuje automatickým ukládáním a možností přihlášení přes Google účet. Když se uživatel přihlásí, tak se jeho data uloží i na Firestore Database, která je součástí Firebase. Postupně se podíváme na hry s podobným žánrem a představím všechny použité technologie. Poté se podíváme na vývoj mobilní hry s mými novými funkcionalitami a problémy, které jsem při vývoji musel řešit. Popis vývoje mobilní hry bude rozdělen do částí, aby byly srozumitelné a lehce pochopitelné. Byl bych rád, kdyby tato práce mohla pomoci komukoliv, kdo by měl zájem o vývoj mobilní hry na platformu Android.

Cílem mé vytvořené hry je zničit zeď. K tomuto jsou dostupné různé nástroje. Některé nástroje jsou dostupné od začátku hry. Zbytek nástrojů je možné objevit v průběhu hry. Zeď má 100 vrstev, které je třeba zničit. Hra také disponuje podzemími, které se objevují průběžně během hry.

1 REŠERŠE

Zde se podíváme na hlavní představitele her s incremental žánrem. Zjistíme, jaké jsou jejich silné stránky, a jaké motivy se v nich opakují.

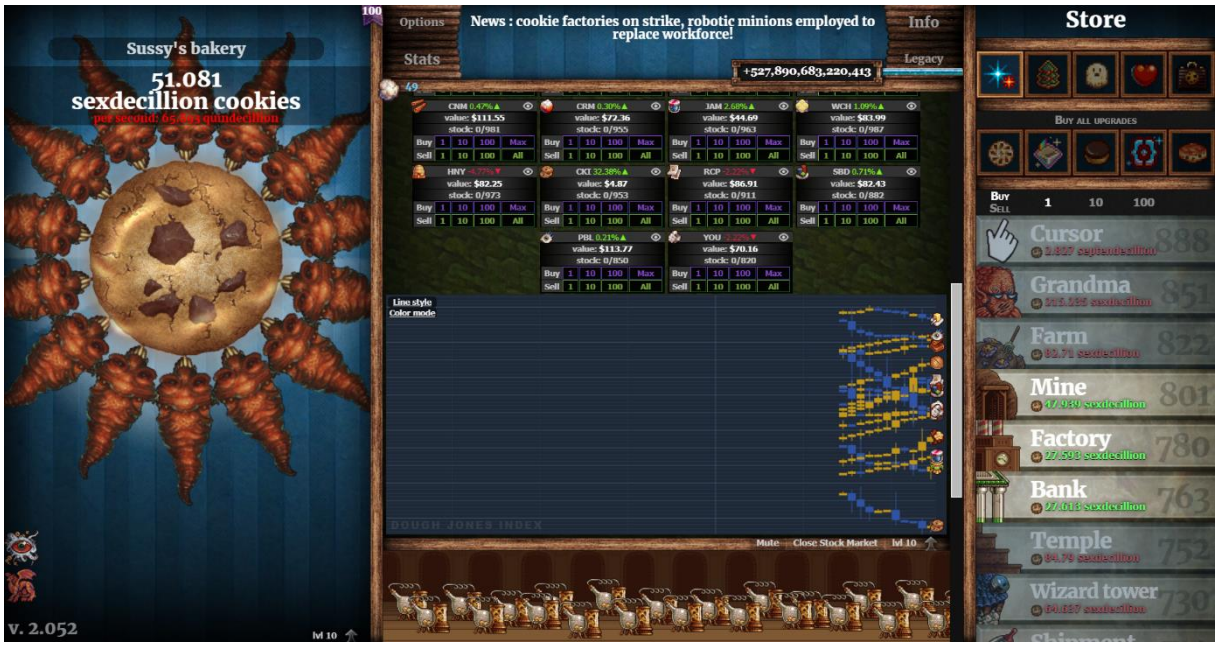
Incremental žánr je typ her, který mají za účel zvyšovat své prostředky inkrementálně. Je to způsob získávání herní měny a následně její používání k tomu, abychom byli schopni získávat prostředky rychleji a efektivněji. Incremental žánr také může mít blízko třeba i k jiným žánrům, nejčastěji se spojuje s žánrem klikacím. [1]

1.1 Počítačové hry

Počítačové hry jsou velkou součástí herní kultury a v tomto případě by bylo nemístné je vynechat. Mnoho her s inkrementálním žánrem je totiž vyvíjeno pro webové prohlížeče. Některé tyto hry sice lze spustit na mobilních telefonech, ale není to zrovna optimální uživatelský prožitek.

1.1.1 Cookie Clicker

Cookie Clicker se dá považovat jako základní benchmark všech her s tímto žánrem. První webová verze byla vydána v roce 2013 a od té doby je stále vyvíjena francouzským programátorem Julien „Orteil“ Thiennot. Později také byla vydána verze na platformě Steam v roce 2021, která oproti webové verzi je zpoplatněna. S touto verzí přichází možnost cloudového ukládání a hudba, která je tvořen umělcem C418. [1]

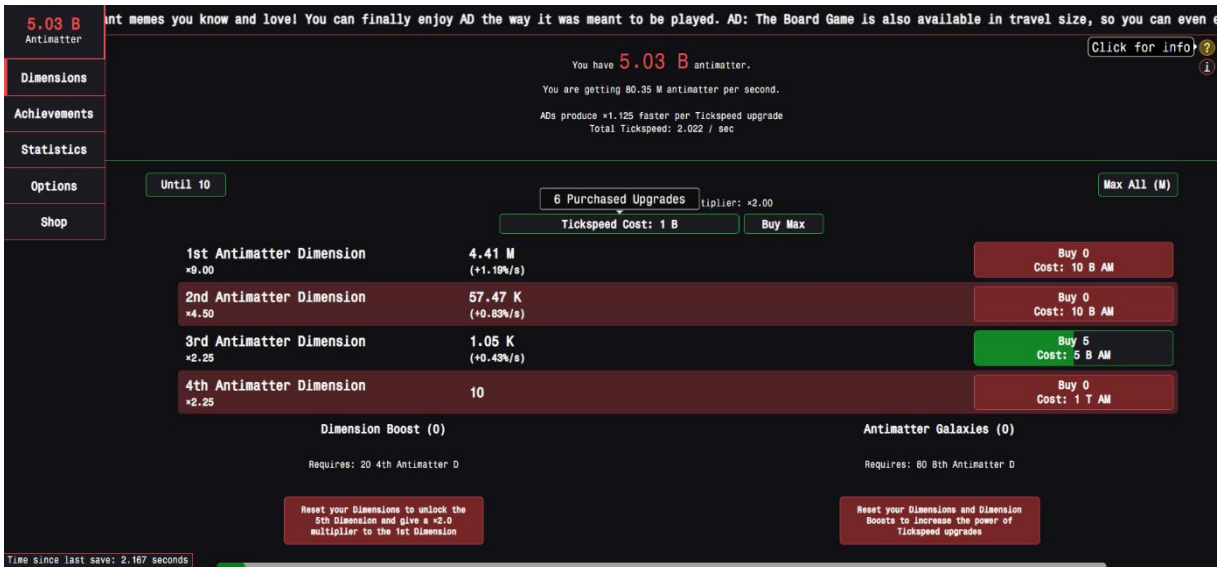


Obrázek 1 Cookie Clicker v. 2.052

Cílem této hry je klikat na sušenku. Tímto postupně získáváte další sušenky a za ty si můžete koupit budovy, které Vám pomáhají klikat na sušenku a tím zvyšují CPS (počet sušenek za sekundu). Daší možností, jak můžete zvyšovat své CPS je také kupováním upgradů, které se Vám postupně otevírají během hraní hry. Některé budovy mají interaktivní prostředí, například farmy Vám umožňují sázet různá semínka a tím můžete postupně mutovat nová semínka a ty znovu zasadit a tímto znovu zvyšujete své celkové CPS. Další budova Vám umožní obchodovat na burze s fiktivními akciemi. Těchto interaktivních budov se v Cookie Clickeru nachází mnoho, některé jsou více interaktivní a jiné naopak méně. Poslední, velice důležitá mechanika, je nazvána Ascend. Jedná se o resetování hry a při začátku můžete upgradovat znovu různé věci, za tzv prestižní levely, které jste nasbírali v průběhu hry. Tyto prestižní levely Vám zvyšují základní CPS a díky tomu jste schopni znovu pokročit ve hře.

1.1.2 Antimatter Dimensions

Antimatter Dimensions bylo originálně vytvořeno v roce 2016 developerem Hevipelle jako hra webového prohlížeče. Od té doby se přidalo mnoho nových developerů a následně i testerů. V dnešní době se vývoj na této hře oficiálně pozastavil. Poslední update byl vydán 18. 7. 2023 a má název „Final Official Patch“. Na hře se dohromady podílelo 5 různých vývojářů. [2]



Obrázek 2 Antimatter Dimensions, webová verze hry 2023-07-18

Cílem hry je nashromáždit co nejvíce antihmoty za pomocí dimenzí antihmoty. Tu můžete postupně nakupovat a tím zvyšovat APS (antihmoty za sekundu). Také je tu možnost restartování dimenzí a tím jste schopni otevřít nové dimenze a také přidat násobky předešlým dimenzím. Další možnost je resetování dimenzí a jejich upgradů za účelem vylepšení tickspeed

upgradů. Je tu mnohem více vrstev, než se na první pohled zdá, které se postupně otevírají během Vašeho postupu ve hře.

Velice zajímavá funkce této hry je možnost její úplné automatizace. V jedné fázi hry se totiž otevře The Automator a za jeho pomoci jste od této chvíle schopni vytvářet svoje vlastní scripty na plnou automatizaci hry.

Jelikož počítačová a mobilní hra, až na malé rozdíly, jsou identické. Tuto hru není třeba znovu zmiňovat v kapitole mobilní hry.

1.1.3 Clicker Heroes

Clicker Heroes bylo vytvořeno pro webové prohlížeče v roce 2014 a v roce 2015 bylo vydáno na platformě steam vývojářem a vydavatelem Playsaurus. Tato hra byla i několik měsíců v žebříčku top 10 her na celé platformě Steam. [3]

Obrázek 3 Clicker Heroes, webová verze hry 1.0e12-4824

Cílem této hry je nábor hrdinů a jejich postupné vylepšování. Díky tomu jste schopni ve hře postupovat v podobě zabíjení nepřátel a postupování do míst se silnějšími nepřáteli. Hra také má silnější nepřátele, které mají časový interval, na jejich zneškodnění. Jestli nejste dostatečně silní na jejich poražení, můžete se vrátit na slabší místo s nepřáteli, které nemají časový interval a postupně vylepšit své hrdiny. Jsou tu také různé vylepšení, které můžete hrdinům pořídit. Většinou se jedná o zvýšení jejich útoků, některé vylepšení jsou významnější a mohou Vám

výrazně pomáhat v postupu hry, jako je například dočasné zvýšení DPS na 30 sekund nebo automatické klikání na minutu.

1.2 Mobilní hry

Mobilní telefony v naší době jsou natolik dostupné, že skoro každý z nás vlastní chytrý telefon. [4]

To je také jeden z důvodů, proč zrovna mobilní hry jsou tak moc oblíbené. Temná stránka mobilních her může spočívat v mnoha aspektech. Jeden z příkladů je přehlcení reklamami, protože většina mobilních her jsou zdarma.

1.2.1 Cookie Clicker

Cookie Clicker jsme si sice již představili, bohužel verze na počítač a na mobilní telefon se liší natolik, že by nebylo fér vynechat tuto osekanou mobilní verzi hry.

Mobilní verze je dostupná pouze pro platformu Android a byla vydána v roce 2019. Této verzi chybí mnoho funkcionalit ze základní hry, některé vlastnosti jsou plánovaný v budoucích aktualizacích a jiné nejspíš nikdy nebudou implementovány. [5]

Oproti počítačové verzi, je hra rozdělena na záložky, mezi kterými se může hráč překlikávat. Také mobilní hra má dvě verze hry live a aplha. Live verze je stabilní mezitím alpha verze může být nebalancovaná a nestabilní.

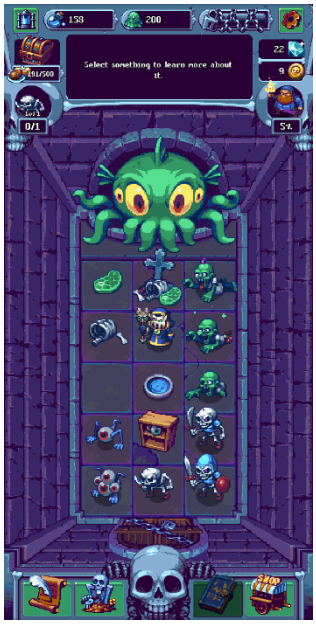
Obrázek 4 Cookie Clicker, mobilní live verze hry 7.17

Mezi chybějící věci patří všechny minihry a momentálně chybí čtyři poslední budovy. V mobilní aplikaci také nenaleznete velkou část achievementů, upgradů a nebeských upgradů. Klasických upgradů chybí zhruba polovina, nebeských upgradů chybí více jak dvě třetiny.

Uložená data hry nejsou kompatibilní s daty z počítačové verze.

1.2.2 Necro Merger

Necro Merger byl vydán v roce 2022 a je vydán a vyvíjen firmou Grumpy Rhino Games. Tato hra je čistě mobilní a je tedy dostupná pouze na Android a iOS. [6] [7]



Obrázek 5 Necro Merger, Version 1.47, Google Play Edition

Cílem hry je získávání surovin a krmení vykukujícího netvora. Získané suroviny stejného původu můžete spojovat a tím z nich tvořit nové a lepší suroviny. Jsou tu tři hlavní měny a tj. mana, sliz a temnota. Díky těmto měnám můžete generovat náhodně již zmíněné suroviny kliknutím na jejich příslušný zdroj. Čím větší úroveň vytvořené suroviny, tím více bonusů přináší. Ať už je to pomoc generování měny anebo jejich hodnota při krmení netvora.

1.2.3 Legend of Slime: Idle RPG War

Legend of Slime byl vydán v roce 2022 firmou AppQuantum, která se zabývá vývojem free-to-play her. Tato hra je také vydána pouze pro mobilní telefony a je dostupná na Android a iOS. [8]

Obrázek 6 Legend of Slime: Idle RPG War, verze 2.10.0

Hlavním cílem hry je, podobně jako v Clicker Heroes postupně porážet nepřátelé a následně bojovat se silnějšími nepřáteli, kteří mají také časový interval na jejich poražení. Rozdíl v podstatě mezi těmito hrami je, že v Legend of Slime stále vylepšujete jednoho vašeho hrdinu. Dále máte možnost dokoupení pomocníků, nebo oblíknutí brnění či různých zbraní.

1.3 Společné rysy

Jak lze vidět, většina těchto her mají alespoň jednu společnou vlastnost. V některých případech se hry mohou podobat jak nápadem, tak i vlastnostmi. To ale neznamená, že jsou stejné. Každá z těchto her se snaží docílit něčeho nového, čím by vynikli v moři her.

Počítačové hry s inkrementálním žánrem byli a stále jsou převážně tvořeny pro webový prohlížeč. Většina z nich jsou také zcela zdarma a ani nejsou zahlceny otravnými reklamami. Také mají možnosti lokálního uložení a načtení progresu, ať už se rozhodnete hrát na více zařízeních či chcete předat svůj progres ve hře kamarádovi.

Mobilní hry, oproti počítačovým mohou být až nezdravě zahlceni reklamami a jsou stavěny kolem jejich zneužívání. Všechny zmíněné mobilní hry disponují reklamami. Bohužel u her Necro Merger a Legend of Slime: Idle RPG War to spíš vypadá, že jsou stavěny kolem reklam. A to v takovém měřítku, že zabraňují normálnímu postupu ve hře, jelikož když shlédnete reklamu dostanete herní měnu, která Vám hru usnadní natolik, že hrát bez těchto výhod je skoro nemožné.

Všechny tyto hry disponují progresivním zvyšování výkonu anebo akumulace prostředků za účelem postupu. Ať už je to koupě dalších budov, které Vám budou produkovat další sušenky, či vylepšení síly hrdinů.

Dále tu máme postupné odemykání funkcí či upgradů. S postupem některých her získáváte nové schopnosti nebo upgrady, které znovu pomáhají se zvyšováním výkonu či akumulací prostředků.

Jedna z mých oblíbených vlastností je kompletní resetování hry za účelem pokroku. Toto nádherně provádí Cookie Clicker a Antimatter Dimensions a řekl bych, že implementace Cookie Clicker je téměř bezchybná.

2 POUŽITÉ TECHNOLOGIE

První důležité rozhodnutí bylo vybrání platformy, na kterou budu svoji hru vyvíjet. To také ovlivnilo technologie, které jsem si nakonec vybral. Na výběr je tedy android anebo iOS. Jelikož již mám zkušenosti s vývojem aplikace na android z jiných předmětů, rozhodl jsem se jít touto cestou.

2.1 Android Studio

Android Studio je oficiální vývojové prostředí pro platformu android a je založeno na základě IntelliJ IDEA.

V Android Studio si můžete vybrat ze tří programovacích jazyků a tj. Java, Kotlin a C++. Dříve byl výchozí programovací jazyk Java, ale Google jí nahradil v roce 2019 jazykem Kotlin. Můžete si stále vybrat, jaký budete používat jazyk ve Vašem projektu. [9]

Android Studio také disponuje mnoha užitečnými vlastnostmi. Jedna z jich je například vestavěný emulátor android telefonu, ze kterého můžete následně testovat svoji aplikaci, či se podívat, zda je správně navrhnutá na různých typů zařízení. Dále máme vestavěnou podporu Google Cloud Platform, Firebase, verzovacího systému GitHub a flexibilní Gradle-based Build Systém na automatizaci sestavování programu. [10]

2.2 Kotlin

Kotlin byl vytvořen v roce 2011 firmou JetBrains a je to alternativa k již ověřenému a populárnímu programovacímu jazyku Java. Tento jazyk byl vytvořen za účelem odstranění chyb, jako je NullPointerExceptions, které jsou stále součástí jazyku Java. Hlavně se jednalo o opravu null hodnot a celkové zjednodušení syntaxe jazyka. Kotlin byl také inspirovaný

mnohými jinými jazyky, jako ne například C#, Groovy nebo Swift. Z těchto jazyků se Kotlin snaží vybrat to nejlepší. [11]

Na výběr jsem měl dohromady ze tří programovacích jazyků, jak jsem již zmiňoval v kapitole Android Studio. Kotlin jsem si vybral díky jeho jednoduchosti a lehké čitelnosti. Také je to defaultní jazyk Android Studia a jeho integrace je natolik příjemná, že jsem ani vteřinu neuvažoval o klasické Javě. Dalším důvodem také je, již zmiňovaný NullPointerExceptions, který měl za následek necelých 30 % chyb v Android aplikací s jazykem Java. [12] Zajímavý poznatek také je, že když chcete vložit do Vašeho projektu kousek kódu v Javě, tak Vám Android Studio automaticky převede daný kód do jeho Kotlin podoby.

2.3 Gson

Gson je externí knihovna pro programovací jazyk Java a tím tedy i pro programovací jazyk Kotlin. Také je občas znám pod názvem Google Gson. Tato knihovna byla vytvořena firmou Google a je využívána na parsování a vytváření JSON objektů. Google tuto knihovnu zprvu vytvořil pouze pro jejich vlastní projekty. Později ji ale dali veřejně přístupnou. [13]

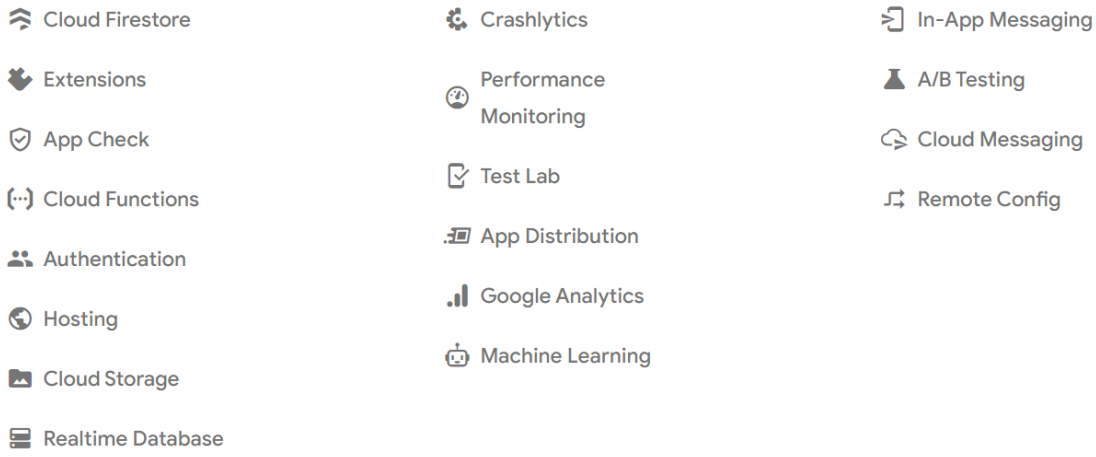
Gson si stanovuje, na jejich vlastní GitHub stránce, několik důležitých cílů. Hlavní z nich tedy je usnadnění převodu JSON souborů na objekty a následně z objektů vytvoření JSON souborů. Další důležitý cíl je podpora velice složitých objektů. To zahrnuje také dědičnost a hierarchii. [13]

Gson je také používán více než 222 000 projekty na platformě GitHub. [14]

2.4 Firebase

Firebase je platforma, která spojuje více různých technologií dohromady v jedné aplikaci. Mezi některé z těchto technologií patří například Google Authentication, Google Analytics a k tomu třeba cloudové databáze jako je Cloud Firestore anebo Realtime Database. Mimo to jsou součástí i jiné technologie viz obrázek 7. Firebase byl vytvořen v roce 2011 a v roce 2012 se projekt stal otevřený veřejnosti. [15] Důležitý krok pro Firebase také bylo spojení se společností Google. [16] Toto byl velice velký krok pro Firebase a umožnil tomu mnohé funkcionality, který z nás zná snad úplně každý.

Jedním z těchto velkých kroků bylo přidání Google Auth, což je ověření a možnost přihlášení s účtem Google.



Obrázek 7 Firebase, výčet včech produktů

2.4.1 Authentication

Authentication je možnost ověření uživatele. Ověření je dostupné jak přes klasické založení uživatelského účtu s emailem a heslem tak i přihlášení přes Google nebo Facebook. Také si můžete s Firebase spojit své vlastní služby. [17]

Díky Authentication lze identifikovat různé uživatele a aplikace získává možnost s těmito daty pracovat. Tyto data lze použít třeba při ukládání stavu aplikace pro daného uživatele.

2.4.2 Firestore Database

Firestore Database je řešení cloudového úložiště, které je zastřešeno již ověřeným cloudovým úložištěm od společnosti Google. Jedná se o NoSQL databázi. Firestore Database umožňuje ukládat různé datové typy jako je například string, number anebo boolean. Díky integraci od společnosti Google jsme schopni ukládat i fotky či videa. [17]

Firestore Database je tedy rozdělena na tři segmenty. Kolekce jsou první segment a mohou obsahovat dokumenty. Dokumenty jsou druhým segmentem. V třetím segmentu může být pole hodnot anebo lze znovu založit kolekci. Ve svém projektu používám kolekci pojmenovanou userData. Tato kolekce pak má dokumenty přihlášených uživatelů, dle jejich unikátního klíče. Tento klíč jsem získal za pomoví Authentication. Každý uživatel pak má několik polí na ukládání různých hodnot.

2.5 GitHub

GitHub byl vydán v roce 2008. Jedná se o webový nástroj pro správu verzovacího nástroje Git. Je to nástroj, který je v dnešní době používán většinou vývojářů a má mnoho zajímavých

vlastností. Každý uživatel si může vytvořit svůj vlastní repositář a ten si sám spravuje. Repositář je místo, kde jsou uloženy a spravovány soubory a zdrojové kódy projektu. Uživatel může repositáři přidávat různá oprávnění jiným uživatelům či spolupracovníkům. GitHub uchovává každou změnu, která byla v projektu provedena. Díky tomu GitHub uchovává komplexní historii vývoje. [18] GitHub je také využíván více než 100 miliony vývojáři. [19]

GitHub také představil způsob verzování spojený s příkazy fork a pull. Projekt se tedy může také větvit, každá větev může vyvíjet třeba novou funkcionalitu programu a poté, co je dokončena se může přidat do hlavní větve vývoje. Díky tomuto také může probíhat vývoj různých funckionalit, aniž by ovlivňovali hlavní vývoj projektu. Mimo hlavní funkcionalitu verzování, GitHub také nabízí sociální funkce, jako je sledování různých projektů či uživatelů. Také je tu mnoho projektů, ke kterým může kdokoliv navrhovat změny či rovnou přispívat k jejich vývoji. [20]

Jedna z nových funkcionalit, která byla přidána v roce 2021, je GitHub Copilot a je to AI pomocník, který je schopen vytvářet kousky kódu, funkce a pomáhat vývojáři s řešením různých problémů. Tato funkcionalita je bohužel placená, ale dle studií, které byly prováděny nestranně se ukázalo, že díky GitHub Copilot bylo mnoho vývojářů více produktivní a byly i více motivováni dokončit složitější úkoly. Copilot je využíván zhruba každou třetí firmou z žebříčku firem Fortune 500. [21]

2.6 Recraft

Recraft je firma založená v roce 2022, která vydala AI program se stejným jménem Recraft. Jedná se o generátor obrázků s pomocí AI. Recraft je schopen generovat jak rastrové, tak i vektorové obrázky. Je to program, který má až 20 miliard parametrů, což je oproti konkurenci, která má zhruba od 2,6 až do 5,1 miliard parametrů. Recraft není založen na žádném open-source modelu a je tvořen od základů. Jejich cíl při vývoji byl anatomická korektnost vygenerovaných obrázků. [22]

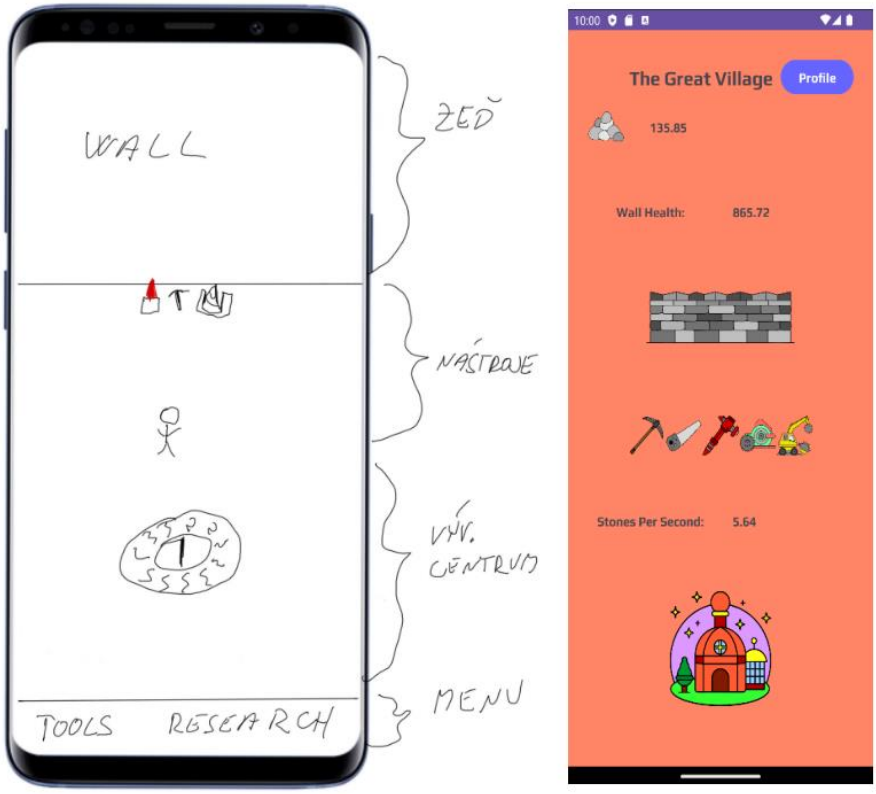
3 MOBILNÍ APLIKACE

V této kapitole si představíme a detailně popíšeme vývoj mé mobilní hry s incremental žánrem. Tato část bude strukturována jako návod pro začínajícího programátora se zájmem o vývoj mobilních aplikací, který bude popisovat základní principy objektově orientovaného programování a klíčové struktury, které tedy byly použity při vývoji. Budu se zabývat postupem vývoje mobilní hry, od samotného nápadu, až po jeho finální implementaci.

3.1 Nápad

Předtím, než vůbec jsme schopni začít s nějakým vývojem mobilní hry, je velice důležité udělat řádnou rešerši. Tam se podíváme na okolní hry a zjistíme, zda je náš nápad dostatečně originální a zda má smysl vůbec pokračovat s jeho vývojem.

Další důležitá část, je mít alespoň nějakou představu konečného produktu. V této fázi to vůbec nemusí znamenat to, že tak ta výsledná hra musí vypadat. Jen je důležité, abychom zhruba věděli, jakým směrem se budeme chtít ubírat a čeho chceme ve výsledné hře docílit. Vždy je dobré své nápady nějak zhmotnit či jim dát nějakou podobu. Proto je dobré si třeba nakreslit podobu, které byste chtěli docílit. Občas stačí i poznamenání na kus papíru.



Obrázek 8 Porovnání prvotního náčrtu (vlevo) a výsledné hry (vpravo)

Zde přidávám svůj prvotní náčrt a výslednou hru (Obrázek 8). Oproti prvnímu náčrtu jsem provedl pár grafických změn. Rozhodl jsem se nevytvářet žádné kontextové menu. Přidán byl pouze odkaz na profil hráče. Mimo to byl zbytek proveden dle náčrtu. Hlavní pointa hry zůstává a je tedy neměnná.

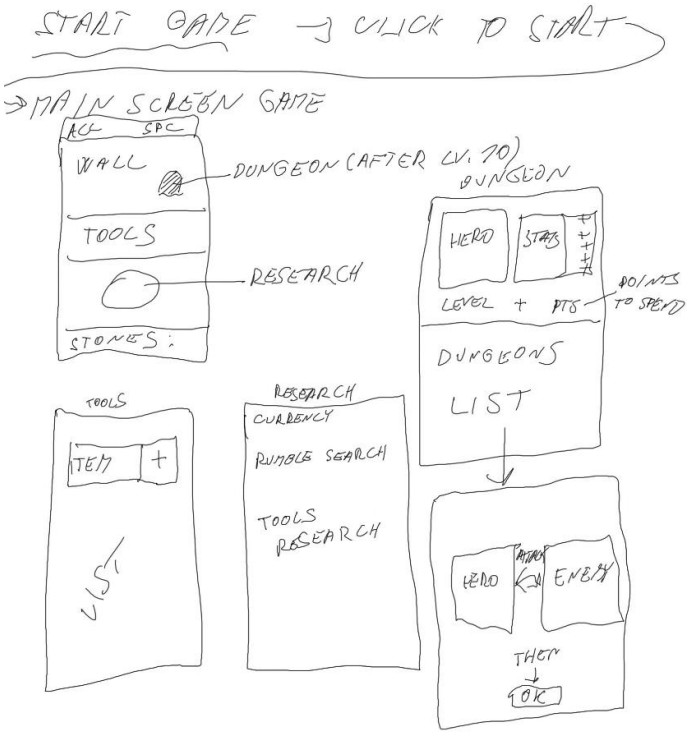
3.2 Plánování Projektu

Plánování projektu může být velice vyčerpávající. Máme nápad a je třeba ho zhmotnit do podoby, dle které bychom mohli projekt postupně implementovat. V této fázi je už důležité mít

nějakou představu. Neznamená, že při plánování projektu budeme schopni předvídat celý průběh implementace. Důležité je, abychom měli alespoň kostru celého projektu. Díky tomu budeme schopni nějak začít.

Stanovíme si pevné cíle. Můj cíl je vytvoření mobilní hry pro platformu android s incremental žánrem. Hra by měla být o zdi, kterou se hráč pokusí zničit dostupnými nástroji. Hráč by měl mít možnost dokoupení dostupných nástrojů. Nástroje mají možnost být vylepšeny a tím se sníží čas na extrakci kamene anebo se zvýší síla nástroje. Další funkcionalitou budou podzemí. Podzemí se objeví po zničení několika vrstev zdi. V podzemí hra představí hrdinu, kterého může hráč vylepšovat. Hrdina bude mít několik dostupných polí na vylepšení, jako je útok a obrana. Podzemí má více poschodí, každé poschodí disponuje sadou nepřátel. Když hrdina zdolá podzemí, tak objeví nový mýtický nástroj a ten se objeví v dostupných nástrojích.

Když už máme stanoveny hlavní cíle můžeme udělat i začáteční grafické návrhy hry. Převážně je to jen pro představu, ale i to pak může pomoci při implementaci. Zde jsem také prošel několika návrhy.



Obrázek 9 Plán grafického rozhraní hry

3.3 Implementace

Implementace projektu může začít mnohem dříve, než vůbec otevřete programovací prostředí. Stačí, když začnete přemýšlet nad tím, jak chcete, aby Vaše aplikace ve výsledku vypadala a

jaké funkcionality budou k její tvorbě potřeba. Jakou architekturu projektu zvolíte? Jaké třídy budete potřebovat? Nad tímto můžete přemýšlet ještě před založením projektu a alespoň Vám to usnadní práci v pozdějších částech implementace. Také je velice dobré si zapisovat zajímavé funkcionality nebo vztahy mezi třídami, které by mohli být užitečné či důležité.

Implementace bude také následovat programovací paradigma objektově orientovaného programování (OOP). Toto paradigma se zaměřuje na modelování programu pomocí objektů, které obsahují data a metody. Metody manipulují s daty.

Mezi hlavní koncepty objektově orientovaného programování patří:

- Třídy a objekty, kde třída je šablona pro vytváření objektů a objekt je konkrétní instance třídy
- Zapouzdření umožňuje skrytí vnitřního stavu objektu. Také umožňuje pouze přístup k datům pouze prostřednictvím definovaných rozhraní (metod)
- Dědičnost umožňuje třídě zdědit vlastnosti a metody jiné třídy. Dědící třída se nazývá potomek a děděná třída se nazývá rodič. Toto zabraňuje zbytečnou replikaci kódu a umožňuje znovu použít již existující kód či hierarchii tříd
- Polymorfismus umožňuje jednotlivým třídám poskytovat metody se stejným názvem, ale jinou implementací. Zjednodušuje se tak práce s různými typy objektů pomocí stejného rozhraní

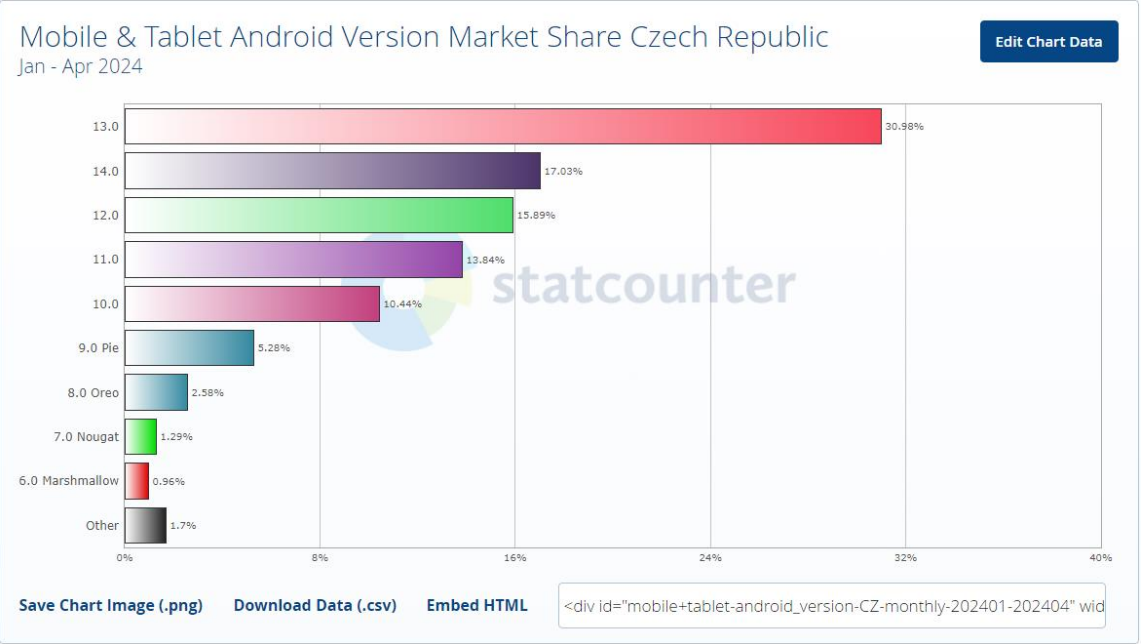
3.3.1 Založení projektu

Už při zakládání nového projektu máme před sebou pár rozhodnutí, která musíme udělat. Prvním z nich je, jaký programovací jazyk budeme používat a druhým z nich je, které minimální SDK si vybereme. Třetí rozhodnutí se týká zvolené softwarové architektury.

Vybral jsem si programovací jazyk Kotlin díky jeho jednoduchosti a čitelnosti, jak jsem již zmiňoval v předešlé kapitole.

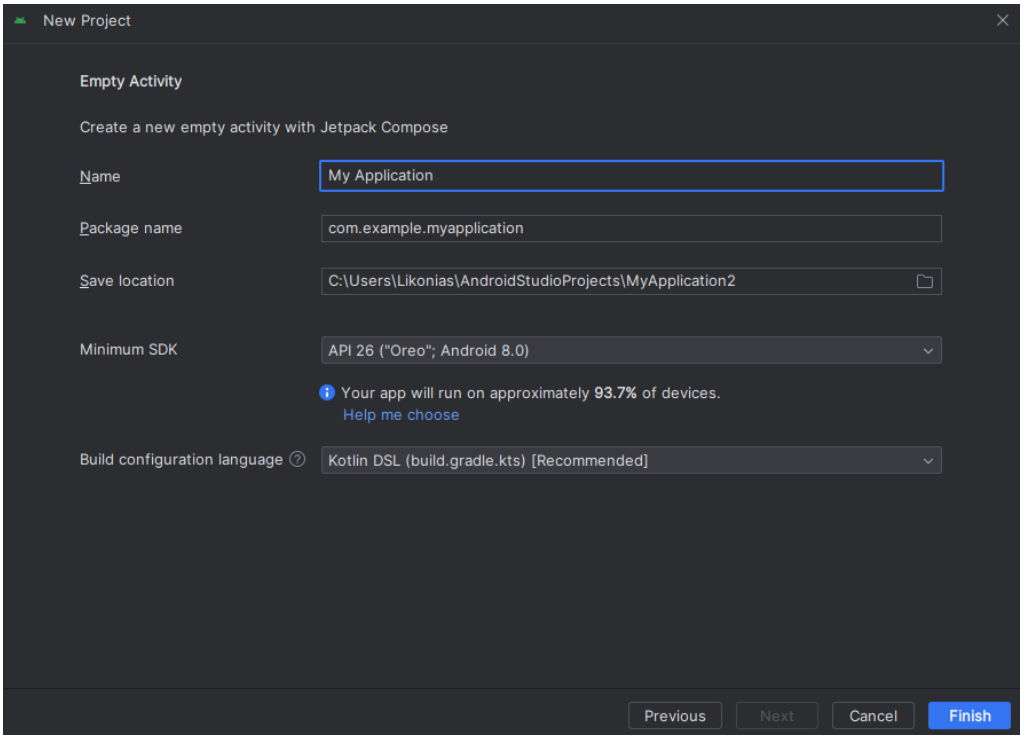
U výběru minimálního SDK to může být trochu složitější. SDK neboli Software Development Kit je balíček zdrojových kódů a nástrojů. Tento balíček umožňuje tvorbu aplikací pro platformu android. Jelikož s postupem verzí novějších SDK jsou přidávány i důležité funkce, jako jsou například nové notifikace či biometrické ověření, doporučuje se volit co nejvyšší možnou verzi. To také závisí o distribuci Android verzí v reálném světě. Naším úkolem tedy

je, zvolit takovou verzi SDK, aby byla dostupná co nejvíce lidem a taky aby měla co nejvíce potřebných funkcí.



Obrázek 10 Distribuce Android verzí v České republice [23]

Jako minimální SDK se tedy doporučuje API 26. To tedy odpovídá systému Android 8. Jedná se o verzi s dobrým pokrytí dostupných verzí, jak českým, tak i globálním. V této verzi už není třeba podporovat staré notifikace a jsou tam relativně nové bezpečnostní aktualizace. [24]



Obrázek 11 Ukázka založení projektu v Android Studio

Ve svém projektu jsem zvolil architekturu MVC. Jedná se o softwarovou architekturu, která vznikla v 70. letech. Dělí se na tři základní skupiny a tj. Model, View a Controller.

Model je jakýsi základní kámen. Zahrnuje datové struktury, operace a logiku potřebnou k manipulaci s daty.

View je to, co uživatel vidí na obrazovce. Je zodpovědný za vizuální prezentaci dat a často představuje grafické uživatelské rozhraní (GUI). V některých případech je možné zobrazování na příkazový řádek.

Controller spojuje Model a View. Přijímá vstupy od uživatele, jako je například kliknutí na tlačítko, a rozhoduje, jak reagovat. Poté aktualizuje Model a View podle toho, co se stalo. [25]

Projekt jsem si tedy rozdělil do tří balíčků. Do každého balíčku budu vkládat tvořené třídy dle jejich zařazení.

3.3.2 Tvorba tříd

V této fázi máme pouze vytvořené prázdné balíčky View, Model a Controller. Začneme tedy postupně vytvářet kostru naší hry dle plánu projektu. Ukážeme si, jak vytvořit různé typy tříd, které jsem ve svém projektu použil.

Class je třída, kterou každý z nás známe. Jedná o základní třídu, která je dostupná ve většině moderních programovacích prostředích. V Kotlinu se inicializuje klíčovým slovem Class. Klíčové slovo následuje název třídy a poté může být v kulatých závorkách uveden konstruktor. Tělo třídy ohraničují složené závorky. Tato třída neoslní svými funkcionalitami a ani neurazí. Tuto třídu jsem používal, když mi stačila základní struktura třídy. [26]

```
4  class Wall() {
5
6
7
8  }
```

Obrázek 12 Ukázka základní třídy

Nejdříve si vysvětlíme, jak fungují proměnné v jazyce Kotlin. Kotlin rozděluje proměnné do dvou hlavních kategorií. Jedná se o proměnné, které se označují klíčovými slovy var a val. Těmto proměnným lze přiřadit typ, ale není to podmínkou. Ve většině případech tedy stačí vložit hodnotu do proměnných a oni si sami přiřadí potřebný typ. Kotlin disponuje několika

základními typy. Čísla se dělí dle typu na Byte, Short, Int a Long pro celá čísla. Pro desetinná čísla jsou tu typy Float a Double. Mezi další typy patří Boolean, Char, String a Array. [27]

Proměnná s klíčovým slovem val je neměnná proměnná. To znamená, že s ní nelze nikterak manipulovat za běhu programu. Val se používá na hodnoty, které nejsou třeba měnit. Proměnná s klíčovým slovem var lze manipulovat za běhu programu. Tyto hodnoty lze měnit a jinak s nimi pracovat.

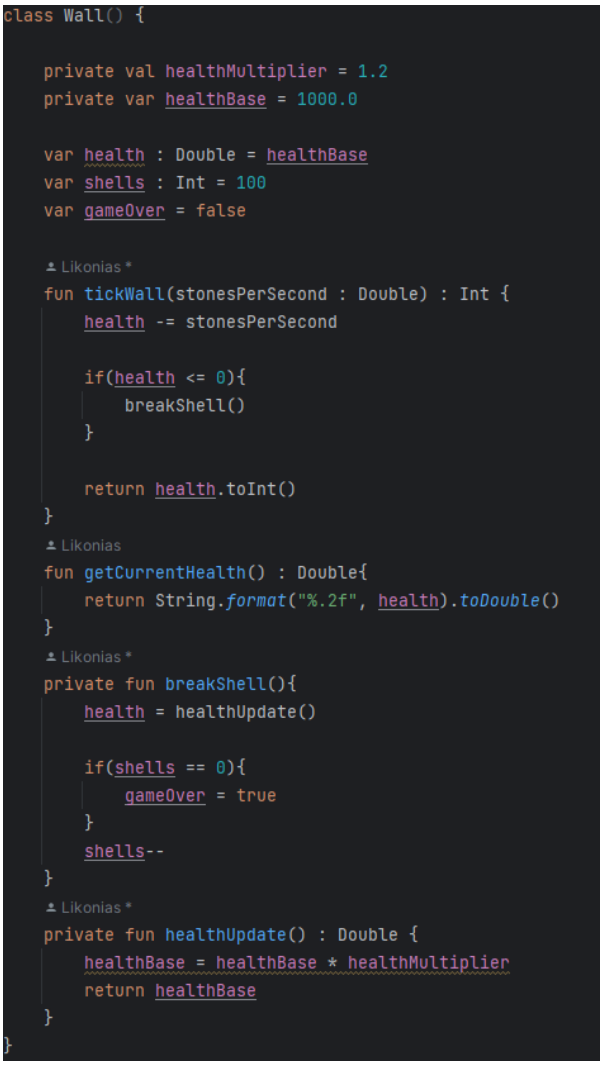
V praxi je doporučeno používat klíčové slovo val při inicializaci proměnných, pokud jejich hodnota bude pravděpodobně neměnná po celou dobu životnosti proměnné. Naopak, klíčové slovo var by se mělo používat pouze tehdy, když je známo, že hodnota proměnné bude měněna během běhu programu. Takovýmto přístupem lze zvýšit bezpečnost a čitelnost kódu a minimalizovat riziko nechtěných změn hodnot.

Podobně jako proměnné, funkce jsou klíčovou součástí jazyka Kotlin a umožňují programátorům strukturovat svůj kód do opakovaně použitelných částí, což zlepšuje čitelnost a údržbu kódu. Funkce mohou či nemusí mít návratovou hodnotu. Funkce se inicializuje klíčovým slovem fun. Dál následuje pojmenování funkce a jako při třídě může být konstruktor. Po konstruktoru lze nastavit návratovou hodnotu funkce. [28]

```
Likonias
fun tickWall(stonesPerSecond : Double) : Int {

    health -= stonesPerSecond

    if(health <= 0)
        breakShell()

    return health.toInt()

}
```

Obrázek 13 Ukázka funkce s návratovou hodnotou

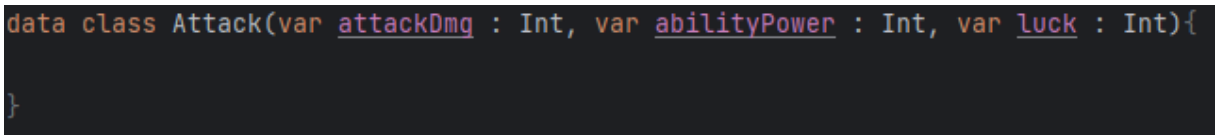
Jedna z prvních věcí, kterou jsem se ve svém projektu rozhodl vytvořit byla třída Wall. Tato třída bude mít několik proměnných, které budou hlídat životy, počet vrstev a neměnnou proměnnou na násobky zvyšování životů.



Obrázek 14 Ukázka základní třídy

Další třída, kterou si zde připravíme tak se jmenuje Data Class. Tato třída je poměrně speciální, jelikož automaticky vytváří metody equals, hashCode, toString a copy. Tyto metody se vytvářejí pouze k proměnným, které jsou uvedeny v hlavním konstruktoru třídy. Proměnné v hlavním konstruktoru se také stávají proměnnými třídy. Hlavní konstruktor třídy je ten, který je definován při deklaraci třídy. [29]

Data Class je třída, kterou jsem převážně používal ve svém projektu. Tato třída velice usnadňuje tvorbu tříd a jejich proměnných. Zde je ukázka třídy, kterou jsem tvořil v balíčku Characters. Tato třída slouží k udržování dat ohledně útoku postavy.



Obrázek 15 Ukázka třídy Data Class

Abstraktní třída v Kotlinu se tvoří klíčovým slovem Abstract Class. Jedná se o třídu, která slouží jako základ pro další třídy, které ji dědí. Tato třída nemůže být použita jako samostatná instance třídy. V abstraktní třídě mohou být předepsané proměnné a metody. Tyto metody a proměnné mohou být také abstraktní. Inicializace metod a proměnných v abstraktní třídě se používá v případě, kdy chceme docílit stejných hodnot proměnných nebo stejného chování metod ve všech odvozených třídách. Když je metoda nebo proměnná deklarována jako abstraktní, je třeba ji implementovat v odvozených třídách. To umožňuje každému potomkovi vytvořit vlastní implementaci těchto hodnot. [26]

Ve svém projektu používám jen jednu abstraktní třídu Character. Tuto třídu používám pro vytvoření základních vlastností postavy. Tato třída v sobě má mnoho metod, které se zaobírají logikou útoku a obrany. Třída obsahuje neměnné proměnné a abstraktní proměnné, které se inicializují v potomkovi. Tuto třídu dělí v mém projektu dvě další třídy a tj. Enemy a Player. Obě třídy inicializují abstraktní proměnné v hlavním konstruktoru.

```
abstract class Character{

    abstract var name : String?
    abstract var health: Double
    abstract var attack : Attack
    abstract var defense : Defense

    var currentHealth = 0.0

    private val defenseThreshold = 80.0
    private val criticalMultiplier = 1.5
    private val hundred = 100
    private val zero = 0

    fun isDead(): Boolean {
    }

    fun attack(character: Character) : Double{
    }

    fun resetHealth(){
    }

    private fun calculateDefensePercentage(character: Character) : Defense {
    }

    private fun calculateLuckPercentage() : Double {
    }

    private fun calculateAttackTrue(character: Character) : Attack {
    }

    //function calculates chance for a critical strike
    private fun isCritical (luck : Int) : Boolean {
    }

    private fun ruleOfThree(a : Double, b : Double, c : Double) : Double {
    }
}
```

Obrázek 16 Abstraktní třída Character, bez implementace funkcí

```
data class Enemy(val id : Int,
                 override var name: String?,
                 override var health: Double,
                 override var attack: Attack,
                 override var defense: Defense
) : Character(){

    Likonias
    init {
        currentHealth = health
    }

}
```

Obrázek 17 Ukázka dědění třídy Enemy

Další třída, kterou ve svém projektu používám je Enum Class. Jedná se o speciální datový typ, který reprezentuje pevně definovaný seznam hodnot. Tyto hodnoty mohou být definovány samostatně anebo mohou mít i své vlastní hodnoty. [30]

Já používám tuto třídu pro držení hodnot své vlastní měny, která se jmenuje CurrencyEnum. V konstruktoru mám dvě proměnné. První proměnná má v sobě hodnotu dané měny a v druhé proměnné se nachází šance na objevení dané měny.

```
enum class CurrencyEnum (val value : Int, val chance : Double) {

    BRONZE( value: 1000,  chance: 0.1),
    SILVER( value: 15000,  chance: 0.01),
    GOLD( value: 300000,  chance: 0.001)

}
```

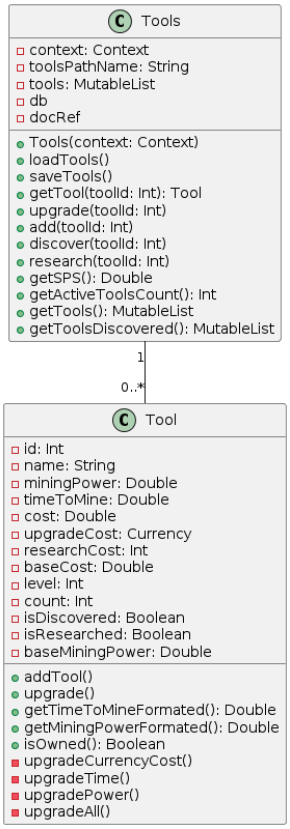
Obrázek 18 Ukázka třídy CurrencyEnum

Object je poslední z typů tříd, které jsem ve svém projektu používal. Object může znamenat mnoho věcí v jazyce Kotlin. Já používám Object jako Singleton, který uchovává hodnotu hry. Singleton je návrhový vzor v softwarovém inženýrství, který je používán k omezení instancí určité třídy na jediný objekt. Jeho hlavním účelem je zajistit, aby měla třída pouze jednu instanci a poskytuje globální bod přístupu k této instanci.

```
object GameSingleton {

    lateinit var game: Game

}
```

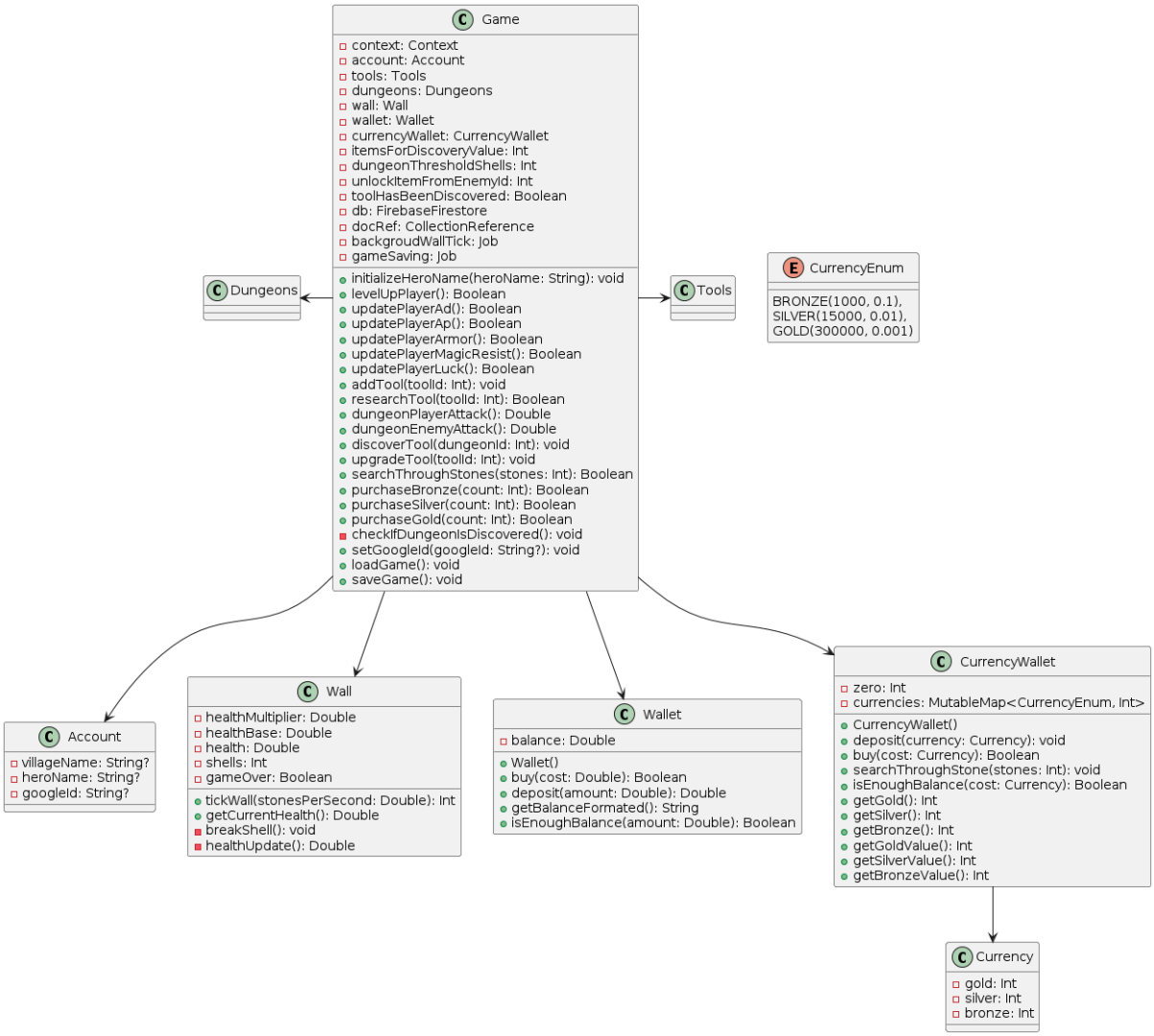
Obrázek 19 Ukázka použití Object jako Singleton

Za pomocí těchto tříd jsem byl schopen poskládat kompletní kostru mé hry. V balíčku Controller mám tři základní třídy, které se spojují všechny třídy z balíčku Model. V balíčku Controller se nachází třída Dungeons, Tools a Game. Zde si teď představíme UML diagramy těchto tříd a tím získáte lepší představu o struktuře mé mobilní hry. Jedná se o zjednodušené UML diagramy.



Obrázek 20 UML diagram třídy Tools

Obrázek 21 UML diagram třídy Dungeons



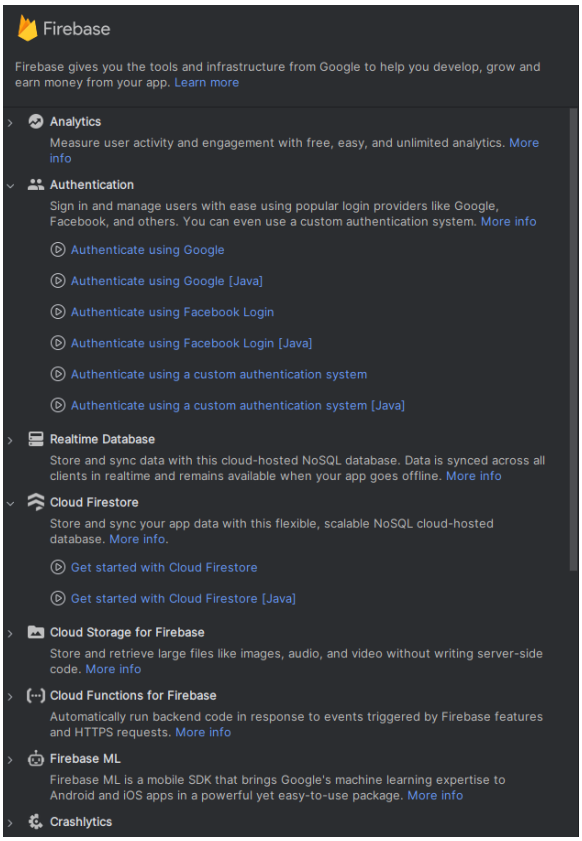
Obrázek 22 UML diagram třídy Game

3.3.3 Komunikace s Firebase

Jak jsem již zmiňoval při představování technologií, Firebase je integrován v programovacím prostředí Android Studio. Já jsem využil konkrétně dvou funkcionalit, které Firebase nabízí. Jedná se o Authentication a Firestore Database.

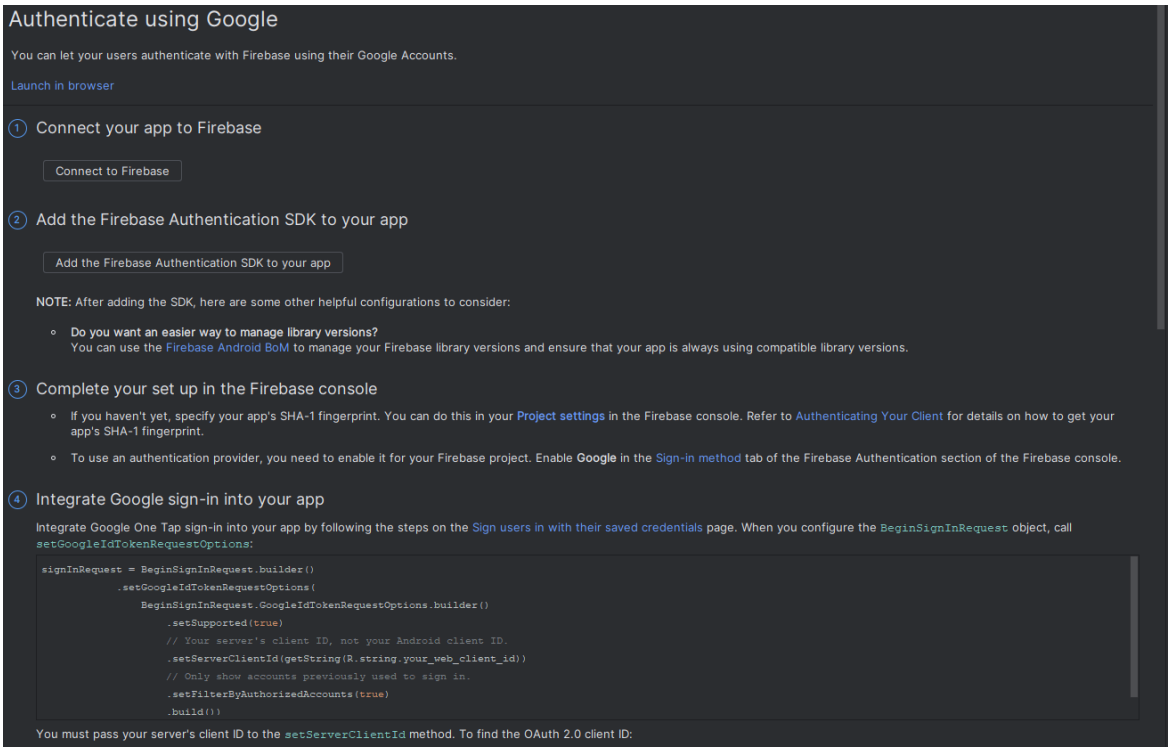
Nejdříve musíme založit Firebase projekt v jeho konzoli. Tato konzole je dostupná ve webovém prohlížeči. Při zaklání projektu musíme náš projekt pojmenovat a nastavíme Google Analytics.

Každá z těchto funkcionalit se musí individuálně nastavit. V Android Studiu je záložka nástrojů, kde se Firebase nachází. Po kliknutí na tento nástroj se zobrazí Firebase a s ní i výpis dostupných funkcionalit.

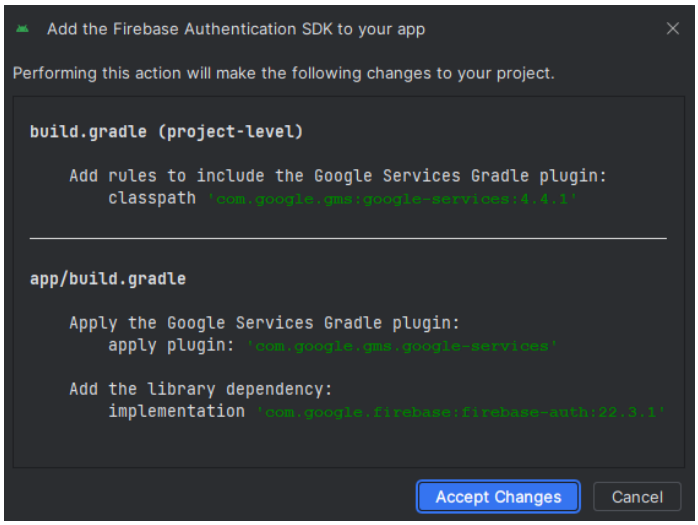


Obrázek 23 Firebase v Android Studiu

Záložka Firebase vše vysvětlí ohledně vybrané funkcionality a navede uživatele k její implementaci. První musí uživatel propojit Android Studio se svým Firebase účtem. Následně musíte přidat dané SDK do vašeho projektu. Všechny verze potřebných SDK jsou předem vyplněné a stačí jen potvrdit změny. Jako poslední krok zbývá přidat SHA-1 otisk vaší aplikace do projektu na Firebase.



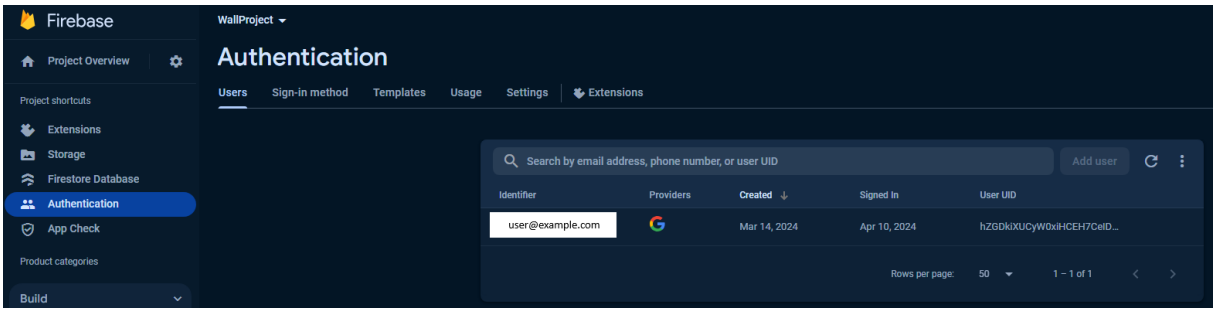
Obrázek 24 Popis kroků pro integraci Firebase Authentication



Obrázek 25 Ukázka Authentication SDK

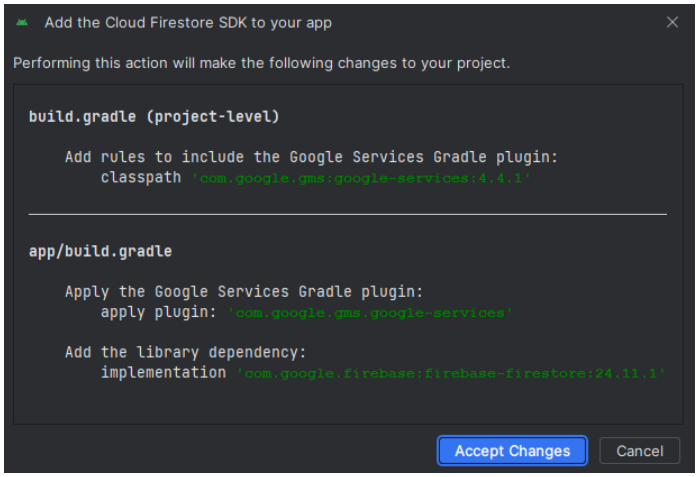
Po těchto krocích jsme připraveni na implementaci ověřování v našem projektu. Ve svém projektu používám již předem vytvořenou a doporučovanou verzi kódu ověření pro Kotlin. Jedná se již o předpřipravenou třídu a na nás zbývá její zakomponování do projektu. V mém projektu se nachází v balíčku Controller a nazývá se GoogleSignIn. [31]

V konzoli Firebase ve webovém prohlížeči v záložce Authentication se nachází všechny data ohledně uživatelů, kteří se přihlásili do aplikace. Hlavní údaje, které záložka uchovává jsou email uživatele a jeho unikátní User UID.



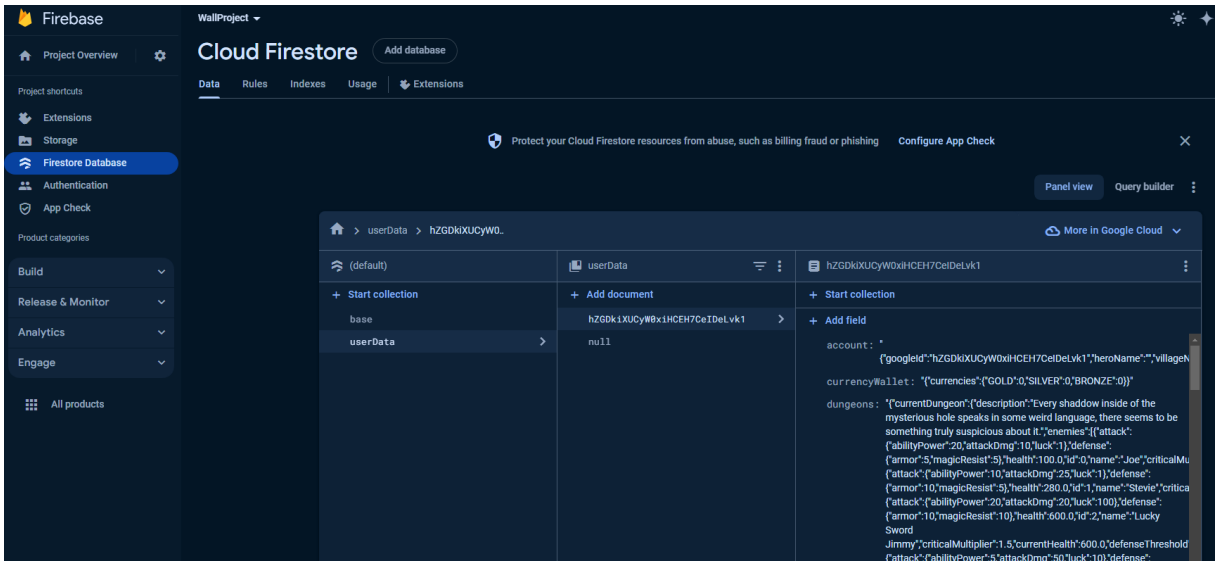
Obrázek 26 Ukázka Firebase Authentication záložky

Firestore Database je další technologie, kterou ve svém projektu používám. Podobně jako u Authentication, je třeba nejdříve přidat do našeho projektu Cloud Firestore SDK a dále nastavit dle instrukcí.



Obrázek 27 Ukázka Cloud Firestore SDK

Znovu si musíme otevřít Firebase v prohlížeči. U Firestore Database musíme nejdříve založit kolekci, do které budeme chtít ukládat data. Zde si určíte její unikátní ID. Dále Vás Firebase donutí vytvořit první dokument v kolekci. Já jsem dokument pojmenoval null. Jakmile máme založenou kolekci, už s ní můžeme pracovat v Android Studiu.



Obrázek 28 Ukázka Cloud Firestore záložky

V Android Studiu musíme nejdříve navázat spojení s databází. Poté se napojíme na kolekci, do které chceme data přidávat. Data se do databáze předávají za pomocí hash mapy. Následně ukládáme data do databáze pod unikátním User UID, které jsme získali díky Authentication.

```
private val db = Firebase.firestore
private val docRef = db.collection( collectionPath: "userData")
// Save to Firestore
val data = hashMapOf(
    "account" to accountJson,
    "tools" to toolsJson,
    "dungeons" to dungeonsJson,
    "wall" to wallJson,
    "wallet" to walletJson,
    "currencyWallet" to currencyWalletJson
)

docRef.document(account.googleId.toString()).set(data)
    .addOnSuccessListener { it: Void!
        println("Game data saved to Firestore successfully.")
    }
    .addOnFailureListener { e ->
        println("Error saving game data to Firestore: $e")
    }
```

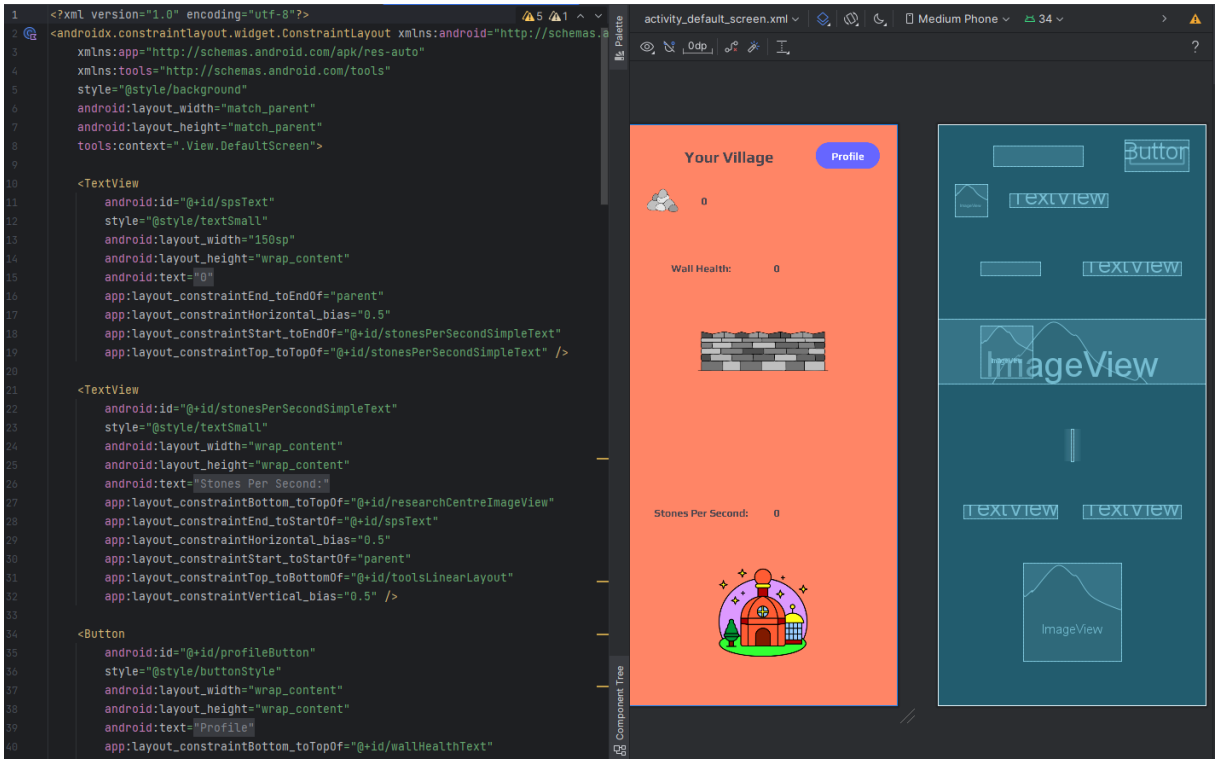
Obrázek 29 Ukázka implementace ukládání herní instance

3.3.4 Grafika

Grafika aplikace může být velice individuální. Já preferuji výstižnost a jednoduchost, a to se tedy odráží v mé implementaci. V Android Studiu se ke grafickému zobrazení používají aktivity. Jedná se o třídy, které dědí ze třídy AppCompatActivity. Při vytvoření nové aktivity

se také vytvoří XML, které odpovídá nově vytvořené třídě. Toto XML se nachází v balíčku layout, který se nachází v balíčku res. Balíček res obsahuje základní zdroje pro mobilní aplikaci. V tomto balíčku také můžeme najít fonty, obrázky, hodnoty, mipmapy a motivy.

Svoji aktivitu si nejdříve tedy navrhneme v XML souboru. XML soubor lze v Android Studiu upravovat dvěma způsoby. První z nich je čistě programátorský a druhý z nich je čistě grafický. Dobré je tyto způsoby kombinovat.



Obrázek 30 Ukázka activity_default_screen.xml

Zde si přidáme komponenty, které v dané aktivitě potřebujeme a nastavíme jejich umístění. Je velice důležité dbát na to, aby umístění dávalo smysl i při otočení zařízení. V některých případech je i nutné rozdělit implementaci rozložení na výšku a šířku.

Poté co máme vytvořený XML soubor se můžeme věnovat implementaci jeho View třídy. V mé aplikaci používám technologie data binding. Tuto funkcionalitu je nejdříve třeba povolit v build.gradle.kts. Jedná se o přístup ke komponentám v XML souboru za pomocí jejich ID.

```
buildFeatures { this: ApplicationBuildFeatures
    viewBinding = true
}
```

Obrázek 31 Povolení data binding

Teď můžeme konečně začít se spojováním grafické a praktické části. Přes binding získáme přístup k dané komponentě a následně s ní můžeme pracovat. V této fázi také využívám singletonu GameSingleton a spojuji instanci hry s pohledem.

```
private lateinit var binding: ActivityDefaultScreenBinding

Likonias
override fun onCreate(savedInstanceState: Bundle?) {

    super.onCreate(savedInstanceState)
    binding = ActivityDefaultScreenBinding.inflate(layoutInflater)
    setContentView(binding.root)

    //todo finish setting up game, loading or saving

    //using application context for some reason

    binding.profileButton.setOnClickListener { it: View!
        startActivity(Intent( packageContext: this, ProfileScreen::class.java))
    }

    binding.wallImageView.setOnClickListener { it: View!
        startActivity(Intent( packageContext: this, WallScreen::class.java))
    }

    binding.dungeonImageView.setOnClickListener { it: View!
        startActivity(Intent( packageContext: this, DungeonsScreen::class.java))
    }
```

Obrázek 32 Ukázka použití data binding

Ve svém projektu používám vektorové obrázky, jelikož vypadají na všech zařízeních stejně. Tyto obrázky jsou složitější na nakreslení. K vytváření těchto obrázků jsem využíval nové AI s názvem Recraft. Recraft se zaměřuje na generování obrázků všech druhů. Já jsem tedy generoval vektorové obrázky, které byly pouze obrysy. Obrázky jsem následně vybarvoval za pomocí Inkscape. Inkscape je open-source software, který se zaměřuje na úpravu vektorových obrázků.

3.3.5 Zajímavé funkcionality

V této části bych chtěl představit pár zajímavých funkcionalit, které jsem během programování využil anebo to mohou být útržky kódu, které mi přijdou přínosné.

Jelikož celá hra je založena na čase, je důležité toto zakomponovat do implementace. Ve své aplikaci používám Coroutines. Jedná se o nástroj pro asynchronní programování. Asynchronní programování je programovací paradigma, které umožňuje spuštění několika různých úkolů v jeden čas. Tyto úkoly se pak navzájem neovlivňují a mohou běžet paralelně. [32]

Při prvním spuštění hry se ve třídě Game spouští dva typy Coroutines. První řídí celý tok hry, tedy ničení zdi za pomocí všech dostupných nástrojů. Tato obnova se provádí každou sekundu. Druhý se stará o pravidelné ukládání hry v časovém intervalu jedné minuty. V hře je tedy nejmenší herní časová odezva stanovena na jednu sekundu.

```
val backgroudWallTick = GlobalScope.launch { this: CoroutineScope
    while (true) {
        if(wall.gameOver){
            break
        }
        delay( timeMillis: 1000)
        var sps = tools.getSPS()
        wall.tickWall(sps)
        checkIfDungeonIsDiscovered()
        wallet.deposit(sps)
        if(dungeons.dungeonsTimerSeconds > 0){
            dungeons.dungeonsTimerSeconds--
        }
    }
}
val gameSaving = GlobalScope.launch { this: CoroutineScope
    while (true) {
        delay( timeMillis: 60000)
        saveGame()
    }
}
```

Obrázek 33 Ukázka Coroutines

Coroutines jsou používány napříč celým projektem, ať už se jedná o obnovu herního prostředí či animací útoků v podzemích.

Další technologii, kterou jsem využil ve svém programování je převod objektů na jejich JSON podobu a naopak. Toto jsem využíval převážně při ukládání a načítání stavu hry. K tomuto jsem využíval externí knihovnu Gson.

```
// Convert game data to JSON strings
val accountJson = Gson().toJson(account)
val toolsJson = Gson().toJson(tools)
val dungeonsJson = Gson().toJson(dungeons)
val wallJson = Gson().toJson(wall)
val walletJson = Gson().toJson(wallet)
val currencyWalletJson = Gson().toJson(currencyWallet)
```

Obrázek 34 Použití knihovny Gson

Herní data jsem také ukládal na zařízení. Když není uživatel přihlášen do aplikace, je to jediné místo, kde se hra ukládá. Když se uživatel přihlásí a zjistí se, že už tuto hru hrál v minulosti, tak se nahraje stará uložená data.

```
// Save locally
val outputStream = context.openFileOutput( name: "game_data.json", Context.MODE_PRIVATE)
val writer = OutputStreamWriter(outputStream)
writer.use { it: OutputStreamWriter
    it.write( str: accountJson + "\n")
    it.write( str: toolsJson + "\n")
    it.write( str: dungeonsJson + "\n")
    it.write( str: wallJson + "\n")
    it.write( str: walletJson + "\n")
    it.write( str: currencyWalletJson + "\n")
}
```

Obrázek 35 Ukázka lokálního ukládání

3.4 Ladění hratelnosti

Při ladění hratelnosti jsem se především soustředil na plynulost hry a jejího postupu. Občas bylo velice složité upravovat tyto hodnoty. Jednalo se převážně o ceny nástrojů. Mým cílem bylo, aby hráč nemusel zbytečně dlouho čekat na zakoupení dalšího nástroje. Na druhou stranu to nemohlo být zase příliš lehké. To by pak mohlo hru rozbít. Musel jsem také vzít v úvahu možnost vylepšení nástrojů. Věřím, že hra je v této fázi hratelná a zajímavá.

Dále jsem se také musel zabývat podzemími a nastavení hodnot nepřátel. Toto se také odvíjelo od dostupných surovin v čase, kdy se podzemí odemknou. Zde jsem hledal přiměřenou cenu vylepšení hrdiny a příjemnému postupu v kontextu nepřátel. Nakonec jsem se rozhodl, že například první nepřítel druhého podzemí není silnější než konečný nepřítel prvního podzemí.

Plynulost kupování nástrojů byl další problém, na který jsem narazil. Jelikož celá hra běží v intervalu jedné sekundy, kupování nástrojů bylo krkolomné a vizuálně nehezké. Toto jsem se rozhodl vyřešit za použití asynchronního programování a nastavil jsem obnovu dat na 100 milisekund.

3.5 Testování

Testeři mé aplikace byli rodinný příslušníci a přátelé. Vysvětlil jsem jim základní principy hry a následně jsem je nechal hru hrát. Plynulost a návaznost herních úkonů byly velké plus. Někteří hráči u hry vydrželi i přes pár hodin.

Při testování jsem sice narazil na pár chyb, ale tyto chyby nebyly složité na opravení. Aplikace měla kladné ohlasy všech zúčastněných.

ZÁVĚR

Hlavním cílem této bakalářské práce bylo vytvořit mobilní hru, která bude odpovídat hernímu žánru incremental. Tato práce také slouží jako příručka pro začínajícího programátora, který by se chtěl věnovat vývoji mobilních her.

V rešerši jsme se podívali na hry s žánrem incremental. Podívali jsme se na jejich silné stránky a opakující se motivy. Představili jsme si hry jak na počítače, tak i na mobilní telefony. Jejich společné rysy mají mnoho podob.

Představili jsme si všechny technologie, které jsem k vývoji své mobilní hry potřeboval. V mém projektu jsem jako vývojové prostředí zvolil Android Studio s programovacím jazykem Kotlin. Na převod JSON souborů jsem použil externí knihovnu Gson. Firebase umožnil uživatelům přihlášení pomocí Google Authentication a ukládání JSON souborů do Firestore Database. V jakémkoliv projektu nesmí chybět verzování a já jsem využil GitHub. Na tvorbu grafických prvků jsem použil nové AI Recraft.

V mobilní aplikaci jsem se věnoval popisu mé mobilní hry. Jedná se o podrobné postupy. Popisuji, jak jsem se dostal od nápadu až k finální verzi mobilní hry. Od nápadu jsme schopni vytvořit první plán našeho projektu. Jakmile je plán projektu ve formě, se kterou jsme spokojeni můžeme začít s implementací. Zde si ukážeme, jak založit projekt. Vytvoříme první třídy a z nich se pokusíme vytvořit kostru projektu. Na tuto kostru můžeme postupně přidávat zbytek plánovaných tříd a funkcionalit. Ukážeme si implementaci Firebase a zajímavé funkcionality. Na závěr popíšu ladění hratelnosti a jaké testování jsem zvolil.

V budoucnu bych velice rád implementoval pokračování hry po dokončení. Tím bych chtěl rozšířit základy hry a také prodloužil její životnost z hráčského pohledu. S tím by bylo dobré také rozšířit dostupné nástroje. Také bych rád přidal další nepřátele v podzemích po zabití konečného nepřítele. Dalším možným rozšířením by mohlo být manuální práce s uloženými daty. Tedy schopnost načítat libovolná data a také schopnost uložit si vlastní progres.

POUŽITÁ LITERATURA

Bibliografie

[1] KERAJÄRVI, Ivar. Utilizing automation to reduce repetition in incremental games. 2023.

[2] *Antimatter Dimensions*. online. In: Steam. Dostupné z: https://store.steampowered.com/app/1399720/Antimatter_Dimensions/. [cit. 2024-04-05].

[3] KARLSEN, Faltin. Exploited or Engaged? Dark Game Design Patterns in Clicker Heroes, FarmVille 2, and World of Warcraft. online. In: *Transgression in Games and Play*. The MIT Press, 2019. ISBN 9780262348706. Dostupné z: https://doi.org/10.7551/mitpress/11550.003.0019. [cit. 2024-04-08].

[4] ELLIOTT, Rob. The demographics of student device ownership. *Educational Technology & Society*. 2023, roč. 26, č. 3, s. 129-140. ISSN 1176-3647.

[5] *Quick mobile Cookie Clicker FAQ*. online. 2019. Dostupné z: https://orteil42.tumblr.com/post/186867952730/quick-mobile-cookie-clicker-faq-expanding-on-the. [cit. 2024-04-06].

[6] *Necro Merger - Idle Merge Game*. online. In: Google Play. Dostupné z: https://play.google.com/store/apps/details?id=com.grumpyrhinogames.necromerger&hl=en&gl=US. [cit. 2024-04-09].

[7] *NecroMerger - Idle Merge Game*. online. In: App Store. Dostupné z: https://apps.apple.com/us/app/necromerger-idle-merge-game/id1611769159?platform=iphone. [cit. 2024-04-09].

[8] *Legend of Slime: Idle RPG War*. online. In: App Magic. Dostupné z: https://appmagic.rocks/google-play/legend-of-slime-idle-rpg-war/com.loadcomplete.slimeidle. [cit. 2024-04-09].

[9] *Google I/O 2019: Empowering developers to build the best experiences on Android + Play*. online. In: Android Developers Blog. Dostupné z: https://android-

developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html. [cit. 2024-04-07].

[10] *Meet Android Studio*. online. In: Android for Developers. Dostupné z: https://developer.android.com/studio/intro. [cit. 2024-04-07].

[11] MOSKALA, Marcin a WOJDA, Igor. *Android Development with Kotlin*. Packt Publishing Ltd, 2017. ISBN 1787128989.

[12] ARDITO, Luca; COPPOLA, Riccardo; MALNATI, Giovanni a TORCHIANO, Marco. Effectiveness of Kotlin vs. Java in android app development tasks. *Information and Software Technology*. 2020, roč. 127, s. 106374. ISSN 0950-5849. Dostupné z: https://doi.org/https://doi.org/10.1016/j.infsof.2020.106374.

[13] FRIESEN, Jeff. Parsing and Creating JSON Objects with Gson. In: . Berkeley, CA: Apress, 2019, s. 243-298. ISBN 978-1-4842-4330-5. Dostupné z: https://doi.org/10.1007/978-1-4842-4330-5_9.

[14] ALMULLA, Hussein a GAY, Gregory. Generating Diverse Test Suites for Gson Through Adaptive Fitness Function Selection. In: . Cham: Springer International Publishing, 2020, s. 246-252. ISBN 978-3-030-59762-7.

[15] *Developers, meet Firebase*. online. In: Firebase. Dostupné z: https://firebase.blog/posts/2012/04/developers-meet-firebase. [cit. 2024-04-10].

[16] *Firebase is joining Google!*. online. In: Firebase. Dostupné z: https://firebase.blog/posts/2014/10/firebase-is-joining-google. [cit. 2024-04-10].

[17] MORONEY, Laurence a MORONEY, Laurence. An Introduction to Firebase. *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*. 2017, s. 1-24. ISBN 1484229428.

[18] *GitHub Turns One!*. online. In: GitHub. Dostupné z: https://github.blog/2008-10-19-github-turns-one/. [cit. 2024-04-11].

[19] *100 million developers and counting*. online. In: GitHub. Dostupné z: https://github.blog/2023-01-25-100-million-developers-and-counting/. [cit. 2024-04-11].

[20] KALLIAMVAKOU, Eirini; GOUSIOS, Georgios; BLINCOE, Kelly; SINGER, Leif; GERMAN, Daniel et al. The Promises and Perils of Mining GitHub (Extended Version). *Empirical Software Engineering*. 2015.

[21] *GitHub Copilot*. online. In: GitHub. Dostupné z: https://github.com/features/copilot. [cit. 2024-04-11].

[22] *Recraft*. online. Dostupné z: https://www.recraft.ai/. [cit. 2024-04-11].

[23] *Mobile & Tablet Android Version Market Share Czech Republic*. online. In: Statcounter Global Stats. Dostupné z: https://gs.statcounter.com/android-version-market-share/mobile-tablet/czech-republic/#monthly-202401-202404-bar. [cit. 2024-04-16].

[24] *Recommended minimum SDK version for Android projects*. online. In: Megu Method. Dostupné z: https://www.megumethod.com/blog/recommended-minimum-sdk-version-for-android-projects. [cit. 2024-04-16].

[25] DEACON, John. Model-view-controller (mvc) architecture. *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*. 2009, roč. 28, s. 61.

[26] *Classes*. online. In: Kotlin. Dostupné z: https://kotlinlang.org/docs/classes.html. [cit. 2024-04-26].

[27] *Basic Types*. online. In: Kotlin. Dostupné z: https://kotlinlang.org/docs/basic-types.html. [cit. 2024-04-25].

[28] *Functions*. online. In: Kotlin. Dostupné z: https://kotlinlang.org/docs/functions.html. [cit. 2024-04-26].

[29] *Data Classes*. online. In: Kotlin. Dostupné z: https://kotlinlang.org/docs/data-classes.html. [cit. 2024-04-25].

[30] *Enum Classes*. online. In: Kotlin. Dostupné z: https://kotlinlang.org/docs/enum-classes.html. [cit. 2024-04-26].

[31] *Authenticate with Google on Android*. online. In: Firebase. Dostupné z: https://firebase.google.com/docs/auth/android/google-signin. [cit. 2024-04-26].

[32] *Asynchronous programming with coroutines*. online. In: Kotlin. Dostupné z: https://kotlinlang.org/spec/asynchronous-programming-with-coroutines.html. [cit. 2024-04-30].