FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Modifikace hry Tamagotchi pro Android.
Student:	Mgr. Adéla Lohonková
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Navrhněte a implementujte 2D hru pro operační systém Android. Hru implementujte v programovacím jazyce Kotlin. Hra bude inspirována hrou Tamagotchi. Modifikace bude spočívat ve využití rozšířené reality při pořizování fotografií v reálném prostředí a zahrnutí mini her do samotné hry. Pokyny pro vypracování:

1. Seznamte se s problémem vývoje mobilních aplikací v jazyce Kotlin a možném zapojení rozšířené reality.
2. Popište stávající verze hry Tamagotchi a navrhněte případně další vlastní modifikace této hry.
3. Návrh implementujte.
4. Výsledek podrobte uživatelskému testování a případně navrhněte další změny.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 30. září 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Modifikace hry Tamagotchi pro Android

Mgr. Adéla Lohonková

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

3. ledna 2020

Poděkování

Ráda bych poděkovala Ing. Miroslavu Balíkovi, Ph.D. za vedení práce a cenné rady. Dále bych chtěla poděkovat svému muži a zbytku své rodiny za trpělivost a spoustu připomínek.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 3. ledna 2020 .....

Odkaz na tuto práci

Lohonková, Adéla. *Modifikace hry Tamagotchi pro Android.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato bakalářská práce si klade za cíl zmapovat celý proces vývoje aplikace inspirované elektronickou hračkou Tamagoči. Hra je určena pro operační systém Android a je napsána v programovacím jazyku Kotlin. Oproti původní verzi, ze které vychází, je obohacena o prvek rozšířené reality, kdy si uživatel může své zvířátko vyfotografovat v reálném prostředí. Také jsou přidány tři minihry, které se zpřístupňují postupně s vývojem tvora. Při implementaci tvoří kostru celé aplikace návrhový vzor MVP. Hru si lze usnadnit nákupy v aplikaci, které jsou realizované pomocí platformy Google Play. Zde byla hra také testována a posléze vydána. V závěru práce jsou představeny možnosti budoucího rozvoje aplikace.

Klíčová slova mobilní aplikace, Android, virtuální zvířátko, DUGO , Tamagoči, rozšířená realita, Kotlin

Abstract

The bachelor thesis aims to map the whole process of application development. The application is inspired by an electronic game Tamagotchi. The game is meant for operation system Android and is written in the programming language Kotlin. In comparison to its original version it is enriched with an element of augmented reality, when the user can take a picture of his dinosaur in the real environment, and with three miniature games, that are enabled along with the creature development. Considering the implementation, the frame of the whole application consists of the architectural pattern MVP. The game can be facilitated by in-app purchases, that are implemented through Google Play platform, where the game was also tested and eventually published. At the end of the thesis, I introduce possibilities of application future development.

Keywords mobile application, Android, virtual pet, DUGO, Tamagotchi, augmented reality, Kotlin

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Běžnou součástí každého dne se v posledních desítkách let stalo používání mobilního telefonu. Ten již dávno neplní jenom komunikační funkci, ale stále častěji se stává zdrojem zábavy a prostředkem pro zpříjemnění času, který bychom jinak trávili čekáním či jízdou v dopravním prostředku. Hraní her pak patří mezi ty úplně nejčastější způsoby při krácení dlouhých chvil. Současně každým dnem narůstá počet her, které se dají stáhnout do mobilního telefonu pomocí distribuční služby Google Play. Mobilní hry jsou oblíbené nejen díky široké nabídce a snadnému a rychlému užívání, ale také díky cenové dostupnosti, protože mnoho modelů mobilních telefonů či tabletů stojí méně, než výkonný počítač či herní konzole.

Jedním z typů dostupných her je i tzv. virtuální zvířátko. Předlohou pro ně je kapesní elektronická hračka Tamagoči vytvořená v roce 1997 v Japonsku. Zde se hráč musí starat, aby opečovávaný tvor dobře prosperoval, netrpěl hlady, nedostatkem hygieny, lásky a aby pravidelně spal.

Původní název pochází z japonského slova „tamago“, což znamená vejce a anglického „watch“ tedy dívat se, starat se. Mojí původní myšlenkou bylo navázat na tento název pojmenováním hry „TamagoSaurus“, nicméně jsem následně vyhodnotila tento název za příliš dlouhý a tím i špatně zapamatovatelný. Konečný název DUGO se dá interpretovat jako spojení anglických slov „Dino You GO“, tedy ve volném výkladu, „dinosaurus, o kterého se staráš i za chůze“. Nicméně pravdou je, že název vznikl mnohem prozaičtěji, a to převzetím slova, které používají na východním Slovensku místo slova zátka, „ďugov“. Toto slovo mi přišlo dostatečně krátké a navíc i dobře použitelné pro hru, jejímž hlavním cílem je starat se o dinosaura, který se rodí jako „malý špunt“.

Hra samotná si klade za cíl navázat na původní hračku a přenést její myšlenku do prostředí operačního systému mobilního telefonu. Je určena pro celé spektrum uživatelů. Dětem může nahradit domácího mazlíčka, kterého jim rodiče nechtějí nebo nemohou pořídit. Pro dospělé je to retro vzpomínka

umocněná rastrovou grafikou, která je typická pro původní kapesní hračku. Starší generace pak může tuto hru vnímat jako vzpomínku na dětství svých potomků nebo jako prostředek saturace potřeby starat se o někoho, když vlastní děti už jsou dospělé.

Téma jsem zvolila, protože dle mého názoru není dostupná jiná adekvátní alternativa této hry s retro grafikou, která připomíná původní hračku a dinosaurem jako druhem virtuálního zvířátka. Při použití dinosaura dostávají totiž smysl i vývojová stádia zvířátka, kterými jsou vejce, ještěrka a dospělý jedinec. Současně tato hra poskytuje i obrovský potenciál pro další vývoj. Kromě péče o zvířátko umožňuje také hrát si s ním a využívat prvky rozšířené reality pro zapojení do reálného světa.

V následujících kapitolách prezentuji celý vývojový cyklus aplikace, od prvotní rešerše možností užití relativně nového programovacího jazyku Kotlin pro programování her pro operační systém Android (3.3), přes způsoby použití nástrojů distribuční služby Google Play pro monetizaci aplikace (3.6), až po srovnání již existujících řešení podobných aplikací (2). Dále představím vlastní návrh herního systému (3.1), uživatelského rozhraní přívětivého pro všechny generace hráčů (4.1) a budu se též věnovat struktuře aplikace sestavené pomocí návrhového vzoru MVP (Model-View-Presenter)(3.4). Také zhodnotím možnosti zapojení rozšířené reality do aplikace (3.5), způsoby ukládání dat (3.7) a také testovací strategie (3.8). Toto bude řešeno v kapitolách teoretické části této práce.

Praktická část bude popisovat samotnou implementaci aplikace (4), nasazení na Google Play, vyhodnocení dat získaných z prvotních testů a následné uvolnění aplikace pro veřejnost (5).

Kapitola 1

Cíl práce

Cílem práce je vytvořit 2D hru DUGO v programovacím jazyku Kotlin pro platformu Android, která bude inspirována původní japonskou hračkou Tamagoči, ale bude navíc obohacena o moderní technologii rozšířené reality, minihry a monetizační prvky, které budou realizovány ve spolupráci s distribuční službou Google Play.

V teoretické části práce nejprve srovnám stávající hry inspirované Tamagoči a navrhnu vlastní možné modifikace. Dále popíši problém vývoje mobilních aplikací v jazyce Kotlin a možné zapojení rozšířené reality. Také se budu krátce věnovat možnostem monetizace aplikací umístěných na Google Play a důvodům mé volby. V poslední části navrhnu vhodné způsoby testování aplikace.

V praktické části práce využiji informací z teoretické části a na jejich základě naimplementuji samotnou hru. Poté popíši testování aplikace pomocí platformy Google Play, jeho výsledky a vliv na další modifikace. V závěru představím možnosti budoucího rozvoje aplikace.

Kapitola 2

Srovnání podobných, již existujících aplikací podle různých kritérií

Primárním cílem této kapitoly je připravit si opěrné body pro samotnou implementaci aplikace. V první části této kapitoly představím historické pozadí této hry a následně se budu věnovat srovnání a analýze již existujících aplikací typu „virtuální zvířátko“.

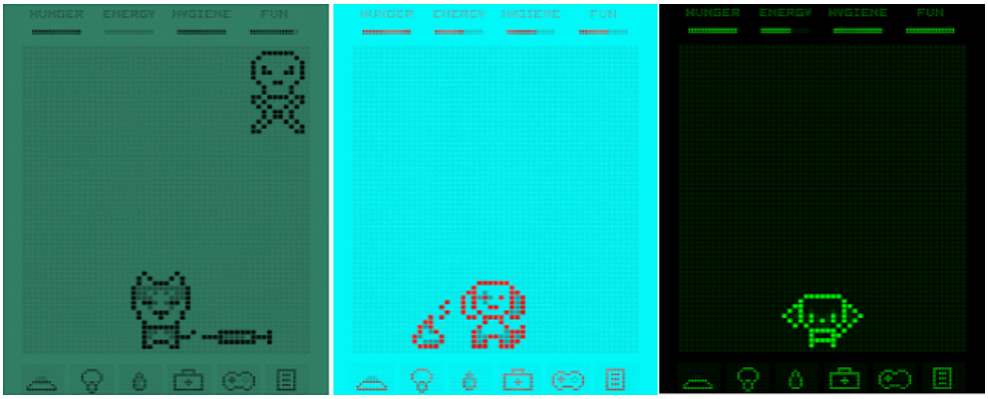
2.1 Fenomén Tamagoči

Téměř každé malé dítě v 90. letech vlastnilo kapesní elektronickou hračku Tamagoči. Ta byla vynalezena v roce 1997 japonským designérem Akihiro Yokoi ze společnosti Wiz a vývojářkou firmy Bandai Aki Maitou [1]. Oba za ni v tomtéž roce obdrželi Ig Nobelovu cenu. Toto americké ocenění paroduje Nobelovu cenu a je udělováno každý rok v říjnu časopisem Annals of Improbable Research za smysl pro humor a recesi ve vědecké profesi. Komise, která o cenách rozhoduje, je složena z nositelů skutečných Nobelových cen. [2]

Hra, která byla původně odmítnuta japonskými obchody jako nezábavná, se velmi rychle stala celosvětovým hitem. Spolu se hrami Dogz a Catz, které vznikly ještě o 2 roky dříve a ještě hrou Digimon, daly vzniknout novému druhu her nazývaných „virtuální zvířátko“.

2.2 Analýza stávajících her inspirovaných Tamagoči

Zde se budu věnovat srovnání her dostupných na platformě Google Play, které se také inspirovaly původní japonskou hračkou Tamagoči. Her existuje opravdu celá řada a na první pohled se liší zejména grafickým zpracováním.



Obrázek 2.1: Dogotchi: Virtual Pet [3]

Při bližším zkoumání také tím, o jaké zvířátko se staráte, délkou jeho vývoje a tím, jestli aplikace poskytuje další minihry, které si lze zahrát. Většina těchto her pak obsahuje nějakou formu monetizace.

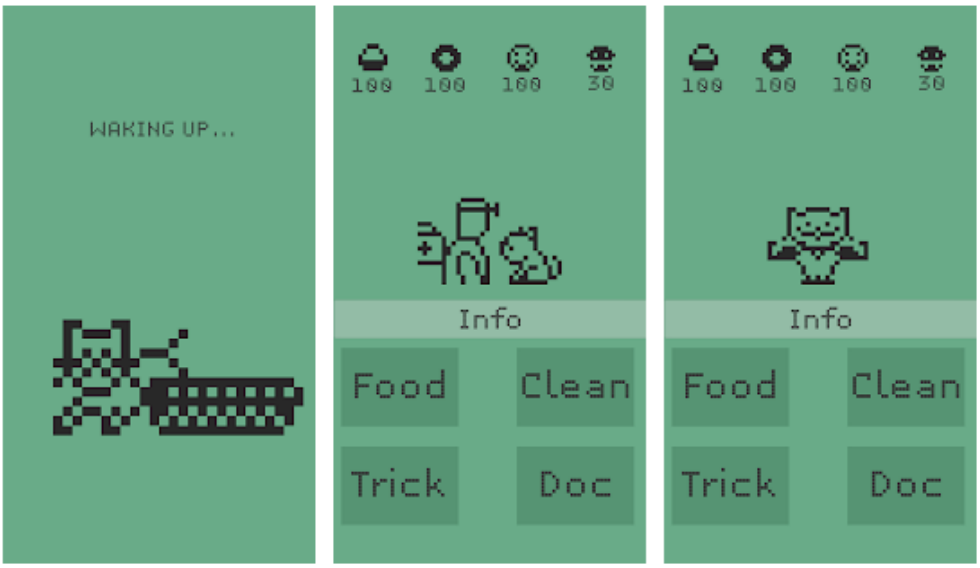
Graficky se dají rozdělit na aplikace s osmibitovou grafikou a na ty graficky vyspělejší. Mým cílem je se co nejvíce přiblížit originální hře, a proto budu volit první variantu. K vlastnímu srovnání aplikací si tedy vybírám pouze ty, jejichž grafické možnosti jsou srovnatelné s těmi mnou zvolenými.

2.2.1 Srovnání her dle grafiky a ovládacích prvků

Graficky nejblíže mé představě je hra „Dogotchi: Virtual Pet“. Zde je opečovávaným zvířátkem pes. Na začátku hry si může uživatel zvolit ze tří ras, jak je vidět na obrázku 2.1. Zároveň se dá měnit barva pozadí a zvlášť menu položek a psa. Veškeré ikony a zobrazení zvířete jsou jednoduché a uživatelsky srozumitelné. Nevhodně zvolené jsou, dle mého názoru, slovní popisky životních potřeb. Pro menší děti bude tato hra tedy hůře hratelná.

Velice podobná hra se jmenuje „Gig Pet - Virtual Cat“. Jak napovídá název, zde je třeba udržet při životě kočku. Tentokrát má hráč k dispozici pouze jedno zvíře, o které se stará. Hra je vytvořena pomocí herního engine Unity. Je zde opět použito hodně textu na akčních tlačítkách i při samotném startu hry. Aplikace je v tuto chvíli chybová, do základního módu byly zřejmě dodatečně zařazeny dvě minihry, které ovšem není možné hrát, při spuštění se aplikace vrací na základní obrazovku, případně se vypne. Poslední aktualizace hry je v roce 2017 a hra nemá slovní hodnocení, nevím tedy, jestli její vývoj ještě pokračuje, nebo se stala jednou z neúspěšných her na Google Play.

Dalším zástupcem této skupiny je hra „Cthulhu Virtual Pet“, kde je grafika již složitější, nicméně stále osmibitová, a tedy retro. Zde jsou ikony potřeb použity bez textu (viz obrázek 2.3), a tak lépe vyhovují širšímu okruhu hráčů. Ti se v tomto případě starají o vodního boha v podobě rozvinutější cho-



Obrázek 2.2: Gig Pet – Virtual Cat [4]

botnice Cthulha. Hra má oproti klasickému Tamagoči mnohem komplikovanější herní systém, kdy je třeba boha uctívat, dovolit mu bořit města a podobně. Bohužel jsem nenašla v aplikaci vysvětlivky významu ikon. Ty jsou sice vizuálně zdařilé, ale není úplně zřejmé, jakou plní funkci ve vztahu k Cthulhovi. I tato hra je vytvořena pomocí herního engine Unity dokonce menší firmou.

Hra „Neps: Virtual Pet“ spojuje opravdu velmi jednoduchou retro grafiku v popředí s pozadím s vyšším rozlišením. Ovládací prvky jsou nicméně osmibitové. Tato hra umožňuje bojovat se zvířátky kamarádů, ale bohužel tyto ovládací prvky jsou opět poněkud méně zdařile označené slovně, jak lze opět vidět na obrázku 2.4.

V neposlední řadě existuje aplikace „DiNostalgia Widget“, která si klade za cíl být kopií původní hry dokonce se shodnými tlačítky a vyobrazením celého přístroje. Toto zde uvádím zejména proto, že opečovávaným zvířetem je dinosaurus, kterého jsem zvolila také pro svou aplikaci. Způsob zobrazení mi však přijde nepraktický a reálně uživatelsky nepříjemný z důvodu malého displeje a špatně dostupných ovládacích prvků. V mobilním telefonu se zobrazuje jako widget, což umožňuje mít více různých zvířátek současně, viz obr. 2.5.

Obrázek 2.3: Cthulhu Virtual Pet [5]

2.2.2 Srovnání her dle času vývoje, miniher a možnosti hrát proti jinému hráči

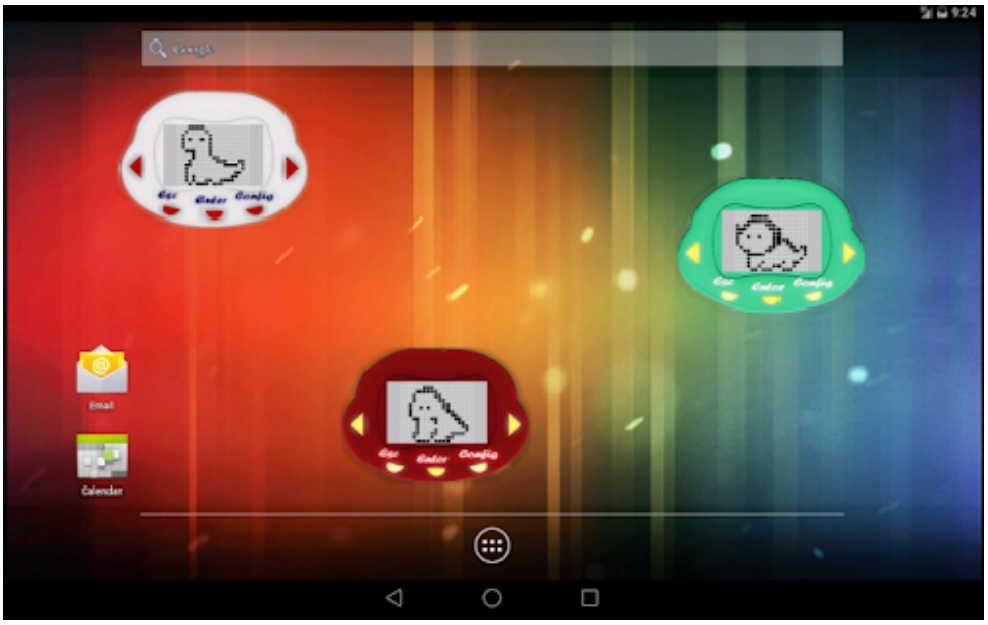
Většina zmíněných her zahrnuje vývoj zvířátka od mláděte po dospělce. Nejpomalejší se zdá hra „Cthulhu Virtual Pet“, kdy uživatelé v recenzích radí posouvat čas na mobilním telefonu pro urychlení. Je proto důležité se dobře zamyslet nad výběrem vhodné rychlosti vývoje zvířátka. Aby si hráč stihl vývojové stádium užít, ale zároveň, aby to nebylo příliš zdlouhavé.

Všechny uvedené hry poskytují uživateli minihry pro zvýšení dobré nálady zvířátka. Tyto hry slouží také pro diverzifikaci pozornosti hráče a také rozptýlení, aby nebyla péče pro uživatele monotónní. Většinou se jedná o vyhýbání se překážkám, střílení nebo sbírání předmětů, stavění kostek aj. Pokud hra poskytuje možnost interakce s jiným zvířátkem, jedná se vždy o vzájemný souboj. [8]

2.2.3 Srovnání her z hlediska použitých technologií

Pro vývoj her pro operační systém Android se dlouhodobě více a více využívají herní engine jako například Unity, které umožňují tvůrcům her maximalizovat výsledek s minimálními znalostmi programování [9]. I hry typu „virtuální zvířátko“ jsou mnohdy vytvořené za jeho použití, jak prozrazuje logo Unity

Obrázek 2.4: Neps: Virtual Pet [6]



Obrázek 2.5: DiNostalgia Widget [7]

při startu hry. Dříve byla majoritním programovacím jazykem Java hlavně s použitím aplikačního frameworku libGDX. V posledních letech se prosazuje i nový programovací jazyk Kotlin, který tento framework může též využít.

2.2.4 Shrnutí srovnání stávajících her

Na Google Play existuje opravdu velká řada her typu „virtuální zvířátko“, nicméně pouze minorita z nich se inspiruje původním Tamagoči co do grafického zpracování. Těchto pár zástupců mi posloužilo jako zdroj inspirace pro tvorbu vlastní aplikace.

Vzhledem k tomu, že tato hra bude jistě oslovovat i nejmladší uživatele mobilních telefonů, je pro mě zásadní co nejintuitivnější uživatelské rozhraní. Tedy zobrazení jak stavových ikon, tak akčních tlačítek obrázkem, který by měl být samovysvětlující.

Také při hraní miniher v uvedených aplikacích jsem došla k závěru, že pokud má aplikace hry pouze jednoho druhu (například postřehové), tak může velmi brzo odradit uživatele, které tyto hry nebaví. Rozhodla jsem se tedy zařadit minihry jak postřehové, tak pamětní a logické.

Žádná z dle uvedených kritérií vybraných a výše analyzovaných her neobsahuje prvky rozšířené reality. Takové hry se dají najít, ale potom zase nesplňují požadavek na osmibitovou grafiku. Protože je rozšířená realita velmi moderní a lákavá, myslím, že její zařazení do aplikace by mohlo znamenat zvýšení míry zájmu potenciálních uživatelů.

Aplikace	*Dogotchi*	*Gig Pet*	*Cthulhu*	*Neps*	*DiNostalgia*	*DUGO*
Retro grafika	ano	ano	ne	ne	ano	ano
Ovládací ikony	oboje	oboje	obrázky	oboje	text	obrázky
Obsahuje chyby	ne	ano	ne	ne	ne	ne
Monetizace	reklamy	reklamy	reklamy	freemium	nákupy	nákupy
Více zvířátek	ano	ne	ne	ne	ano	ne
Rozšířená realita	ne	ne	ne	ne	ne	ano

Tabulka 2.1: Shrnutí srovnání stávajících her

Kapitola 3

Analytické zpracování navrhované aplikace

Po prozkoumání a zhodnocení stávajících aplikací na principu „virtuálního zvířátka“ se nyní zaměřím na vlastní analýzu navrhované aplikace. V prvé řadě je třeba si stanovit pravidla, která bude hra splňovat, na jejich základě stanovit funkční i nefunkční požadavky, zamyslet se nad použitím návrhových vzorů, které usnadní implementaci, a následně zvolit vhodné technologie, které umožní samotnou realizaci. Jako poslední krok je nutné se zamyslet nad testováním vznikající aplikace.

3.1 Systém hry

Základní pravidla této hry vycházejí čistě z fungování původní elektronické hračky Tamagoči. Hráč má na starosti jednoho tvora, který postupně prochází různými vývojovými stádii. Stará se o jeho potřeby tak, aby nestrádal. Krmí ho, čímž navyšuje jeho sytost. Dává ho spát, čímž doplňuje jeho energii. Hraje si s ním, čímž uspokojuje jeho potřebu lásky. Pokud klesne zdraví zvířátka, tak ho léčí. Jakmile dojde k vyčerpání zdraví natolik, že ukazatel klesne na nulu, zvířátko umírá. V tom případě může neúspěšný chovatel zahájit hru od začátku.

Je ovšem jasné, že tato pravidla berou hru jako černou skříňku a neberou v potaz její vnitřní fungování. Pro to je třeba mít stanoven detailní **systém hry**:

Nová hra

Při spuštění nové hry je tvor ve stavu vejce. Všechny ukazatele stavu jsou nastaveny na 50 %. Zvuk je spuštěn.

Změna ukazatelů stavu

V závislosti na čase ubývají všechny ukazatele stavu tvora. Herní systém je nastaven tak, aby přibližně odpovídal dennímu rytmu člověka. Pro dítě může být výchovným prvkem v tom smyslu, že tvor jí, když dítě jí, tvor jde spát, pokud dítě jde spát. Je nastaven tak, aby bylo třeba ho nasytit cca čtyřikrát až pětkrát denně, při jídle vždy vykoná potřebu a desetihodinovým spánkem naplní ukazatel energie.

Systém porovnává časový rozdíl mezi odchodem z obrazovky a návratem na ni a převádí ho na celé hodiny pomocí horní celé části dle vzorce 3.1. Předpokládám, že hráč bude s tvorem buď aktivně interagovat, nebo jej kontrolovat rámcově jednou za několik hodin. Čím častěji hráč tvora kontroluje, tím více má zájem s ním interagovat, a i proto více ubývají ukazatele stavu.

$$T = \lceil \Delta t[hod] \rceil \tag{3.1}$$

Maximální hodnota je 10 dílků a minimální 0. Algoritmus úbytku je stanoven následovně.

- **Sytost:** Pokud je tvor vzhůru, sytost ubude více ($U_s(den)$) než v noci ($U_s(noc)$), jak je vidět dle 3.2. Jeho zvýšení se provádí krmením tvora různými potravinami. Každá z nich pak navyšuje sytost o jiný počet dílků: jablko o 2, kost o 4, voda o 1, hamburger o 6, dort o 5, ryba o 4, list o 3 a kýta o 5. Tyto hodnoty přibližně korelují s reálnou schopností potraviny zasytit.

$$\begin{aligned} U_s(den) &= 3 * T \\ U_s(noc) &= \lceil 0,75 * T \rceil \end{aligned} \tag{3.2}$$

- **Energie:** Ve dne energie klesá ($U_e(den)$), v noci naopak stoupá ($P_e(noc)$) dle vzorce 3.3.

$$\begin{aligned} U_e(den) &= \lceil 0,75 * T \rceil \\ P_e(noc) &= 1 * T \end{aligned} \tag{3.3}$$

- **Láska:** Ve dne ($U_l(den)$) klesá rychleji než v noci ($U_l(noc)$) dle vzorce 3.4. Její hodnota se zvýší, pokud si s tvorem zahrajete jednu z dostupných her. Přesný přepočet je uveden u sekce Hry 3.3. Ten vychází z průměrných hodnot dosažených při testování jednotlivých her.

$$\begin{aligned} U_l(den) &= 1 * T \\ U_l(noc) &= \lceil 0,75 * T \rceil \end{aligned} \tag{3.4}$$

- **Zdraví:** Tento ukazatel je závislý na hodnotách ostatních tří. Nezáleží na tom, zda tvor spí nebo bdí, klesá (U_z) vždy dle vzorce 3.5, kde (E) značí počet exkrementů. Pokud je některý z ukazatelů stavu úplně vyčerpán, pak klesá rychleji (U'_z). Zvyšování probíhá stisknutím ikony injekční stříkačky, kdy každé použití přidává vždy dva dílky až do dosažení maximální hodnoty. Jakmile klesne hodnota zdraví na nulu, tak tvor umírá.

$$\begin{aligned} U_z &= \lfloor (U_s + U_e + U_l)/5 + E \rfloor \\ U'_z &= \lfloor 2 * (U_s + U_e + U_l)/5 + E \rfloor \end{aligned} \tag{3.5}$$

Zobrazení speciálních symbolů

Pokud klesne zdraví tvora pod čtyři dílky a zároveň není noc, tak se nad jeho hlavou zobrazí lebka na znamení toho, že pomalu umírá a bylo by dobré ho uzdravit. Také pokud je hodnota sytosti rovna pěti nebo deseti dílkům a zároveň není noc, vedle tvora se objeví symbol exkrementu. Maximální počet zobrazených exkrementů jsou dva. Lze je uklidit stisknutím tlačítka s toaletním papírem.

Dospívání

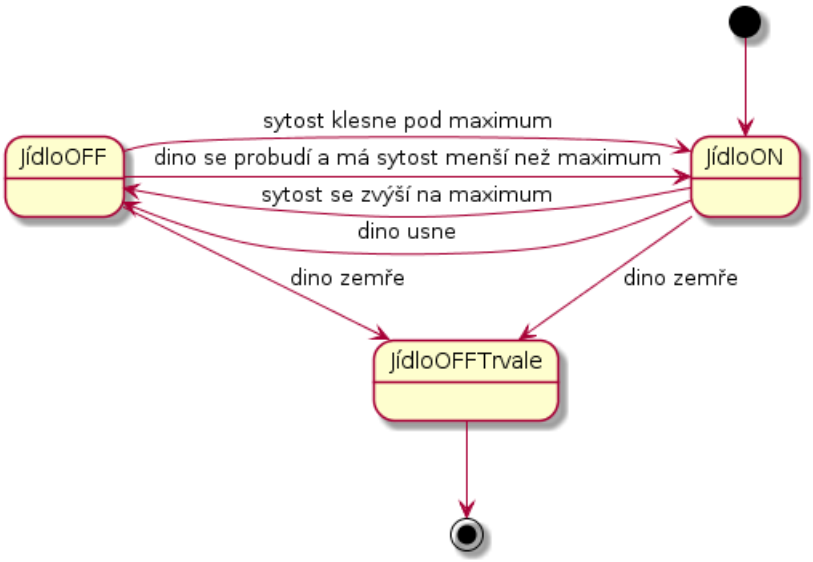
Při spuštění nové hry se tvor zhmotní v podobě pulzujícího vajíčka. V případě, že hodnoty všech ukazatelů stavu jsou hodnoty sedm a více, promění se v prasklé vajíčko. Jakmile jsou hodnoty doplněny do sta procent, promění se v ještěrku. Po třech dnech od spuštění nové hry se tvor změní v dospělce, ale pouze v případě, že je již ve stádiu ještěrky, jinak se tři dny začnou odpočítávat znovu od začátku.

Druhy dospělých dinosaurů

Hra poskytuje možnost získat čtyři druhy dospělých dinosaurů: Stegosaura, Tyrannosaura rexe, Pterodaktyla a Brontosaura. To, kterého uživatel získá, se rozhoduje do doby proměny v dospělého jedince pomocí stravy, kterou ho krmí. V případě, že tvor žere převážně hamburgery a kýty, tak se přemění v Tyrannosaura, pokud ryby a kosti, tak v Pterodaktyla, pokud listy a jablka, tak v Brontosaura a pokud dorty a vodu, tak ve Stegosaura.

Tlačítka k ovládání hry

Tlačítek je celkem osm: jídlo, toaletní papír, hry, injekční stříkačka, fotoaparát, den/noc, nákupy a nastavení. Tlačítka k ovládání hry je třeba zpřístupňovat a naopak za pevně daných situací. To vše slouží hráči k orientaci ve hře. Jediné vždy přístupné tlačítko je to sloužící k zobrazení nastavení aplikace,



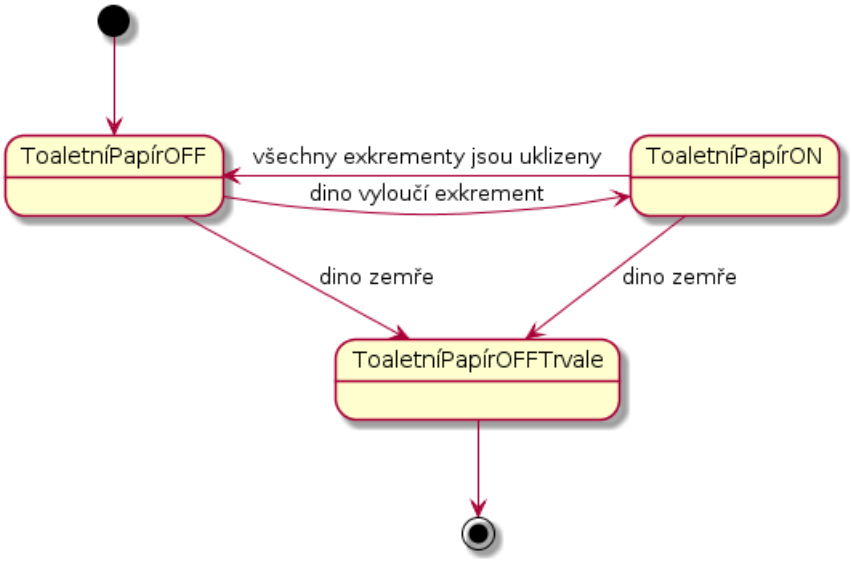
Obrázek 3.1: Stavový diagram tlačítka obsluhujícího jídlo

kde je i spuštění nové hry nebo odejití z aplikace. Všechny ostatní situace jsou zobrazeny na stavových diagramech 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7.

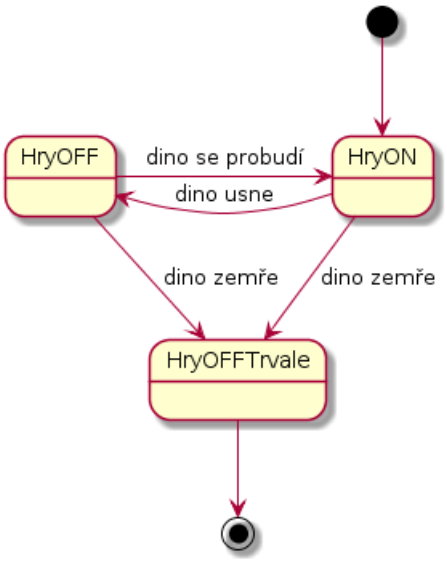
Minihry

Hra zahrnuje tři minihry, které se zpřístupňují postupně, jak tvor stárne. Ve stádiu vejce je dostupná pouze první hra, ve stádiu ještěrky je k dispozici také druhá hra, ve stádiu dospělce je možné hrát všechny tři minihry.

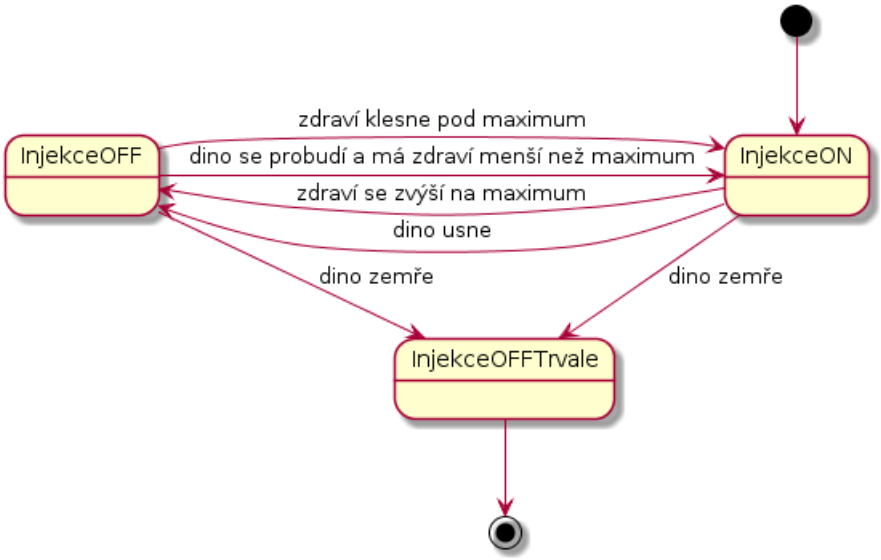
- **První hra:** „Chyť vejce“. Cílem této hry je v časovém limitu třiceti vteřin co nejvícekrát prstem chytit vejce, které se objevuje na obrazovce na různých místech. Skóre je pak přepočítáno a dochází k navýšení ukazatele lásky. Přepočet je dolní celá část poměru skóre a maximální hodnoty ukazatele, tedy deseti.
- **Druhá hra:** Pod číslem dva se skrývá klasické pexeso. Rozměr hrací plochy je třikrát čtyři obrázky. Celkem lze tedy spárovat šest druhů pixelových příšerek. Vždy lze dvě otočit, a pokud jsou stejné, zůstanou otočené a je možné otáčet další. Pokud to jsou různé obrázky, tak je třeba karty otočit zpátky a až potom lze zkusit další. Počítá se počet otočení tam i zpátky. Toto číslo je pak vyděleno šesti, vzata dolní celá část podílu, odečteno od maximální hodnoty stavu, tedy deseti a přičteno k množství lásky.
- **Třetí hra:** Obrazovka třetí hry je rozdělena na devět polí. Každé je kouskem obrázku. Pomocí vzájemného vyměňování pozic obrázků mezi



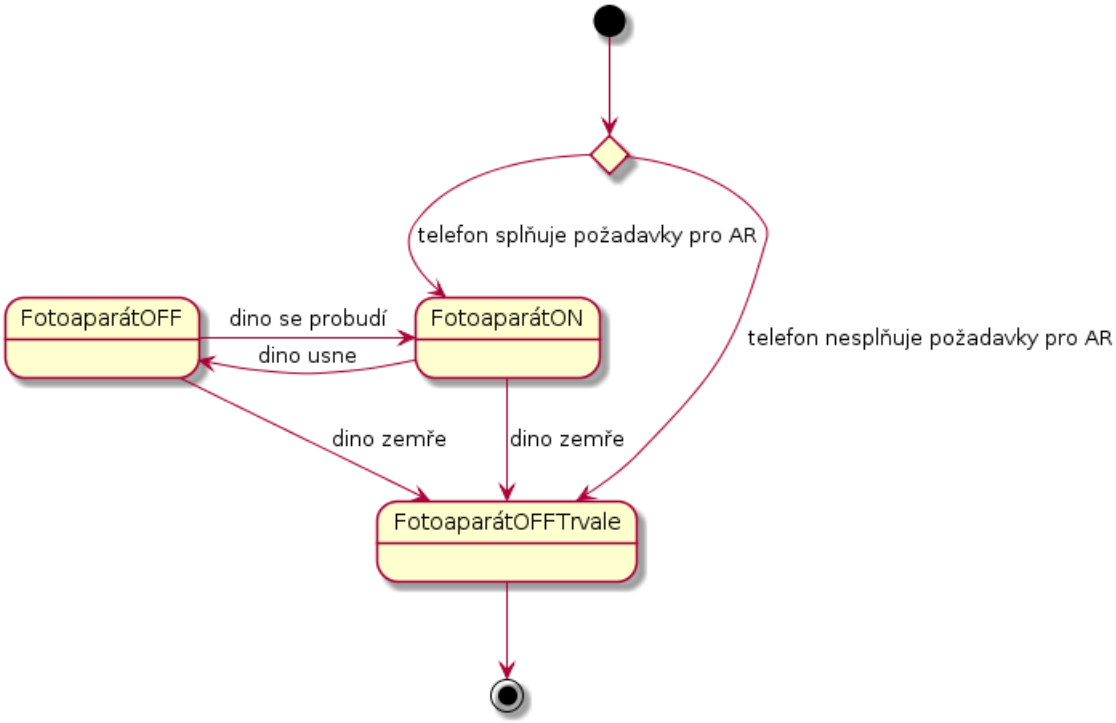
Obrázek 3.2: Stavový diagram tlačítka obsluhujícího toaletní papír



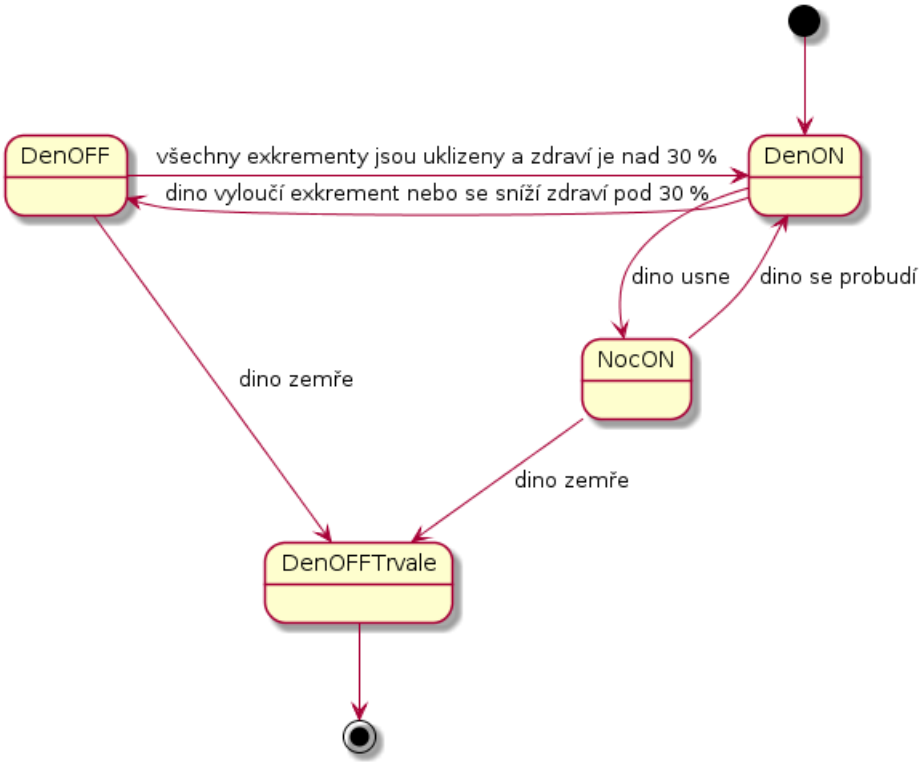
Obrázek 3.3: Stavový diagram tlačítka obsluhujícího hry



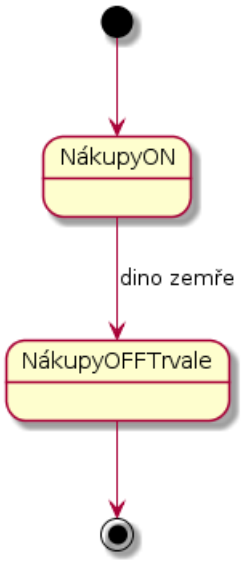
Obrázek 3.4: Stavový diagram tlačítka obsluhujícího injekci



Obrázek 3.5: Stavový diagram tlačítka obsluhujícího fotoaparát



Obrázek 3.6: Stavový diagram tlačítka obsluhujícího přepínání dne a noci



Obrázek 3.7: Stavový diagram tlačítka obsluhujícího nákupy v aplikaci

sebou potažením prstu je cílem sestavit celkový obrázek. Počet výměn určuje skóre. Skóre je přepočítáno podobně jako u druhé hry, pouze se nedělí šesti, ale čtyřmi.

Rozšířená realita

V jakékoliv fázi vývoje lze použít prvek rozšířené reality a s tvorem se vyfotit nebo ho samotného vyfotit v jakémkoliv prostředí. Stačí stisknout ikonu fotoaparátu a ta otevře obrazovku, kde je možné umístit dinosaura do prostředí, zvětšit ho nebo naopak zmenšit. Je také možné ho natočit do různých směrů. Dinosaurus není v tomto případě dvojrozměrný, ale trojrozměrný. Třetí rozměr vzniká vytažením osy z do prostoru. Vypadá to tedy, jako by byl dinosaurus tvořený z malých černých krychliček.

Nákupy v aplikaci

Jako naprostá většina podobných aplikací, má i tato možnost své monetizace. Vzhledem k analýze dostupných aplikací a jejich recenzí jsou prostředkem zisku financí nákupy položek v aplikaci. Lze si zakoupit celkem osm položek, které lze rozdělit do tří kategorií:

1. **Výběr dospělce:** Pokud chce mít hráč jistotu, který druh dinosaura bude jeho jedinec, až dospěje.

2. **Koupě hry:** Hry se zpřístupňují až během dospívání dinosaura. Pokud chce hráč mít dostupnou libovolnou hru v předstihu, může si toto zakoupit.

3. **Ostatní:** Zajištění nesmrtelnosti dinosaura na týden je vhodná investice například při odjezdu na dovolenou. Poslední možností je koupit pivo vývojáři. Tato položka sice neposkytuje žádnou herní výhodu, ale funguje jako ocenění práce na hře.

Nastavení

V sekci nastavení lze zahájit novou hru, ztišit nebo naopak zapnout zvuky, přečíst si informace o aplikaci a nebo aplikaci bezpečně opustit.

3.2 Analýza požadavků

Když máme představu o tom, jak by celá hra měla fungovat, je třeba si precizně stanovit funkční a nefunkční požadavky, které slouží jako vodítko při samotné implementaci aplikace. Funkční požadavky stanovují konkrétní funkcionality, které by systém měl poskytovat a nefunkční požadavky je doplňují o omezení kladené na proces vývoje nebo samotný systém.

3.2.1 Funkční požadavky

1. Založení nové hry: Aplikace umožní uživateli založit novou hru.
2. Pořizování fotek pomocí rozšířené reality: Aplikace umožní pořizování fotek tvora pomocí rozšířené reality.
3. Nákupy v aplikaci: Aplikace umožní uživateli nakupovat v aplikaci.
4. Péče o virtuálního tvora: Aplikace umožní uživateli pečovat o virtuálního tvora.
5. Hraní miniher: Aplikace umožní uživateli hrát minihry, které budou její součástí.

3.2.2 Nefunkční požadavky

1. Programovací jazyk Kotlin: Aplikace bude naprogramována v programovacím jazyce Kotlin.
2. Android 4.1 a vyšší: Aplikace bude fungovat na zařízeních s OS Android 4.1 a vyšší.
3. Návrhový vzor MVP: Aplikace bude dodržovat návrhový vzor MVP.
4. Publikace na Google Play: Aplikace bude publikována na distribuční platformě Google Play.

3.3 Volba operačního systému a programovacího jazyka

Při rozhodování mezi mobilním operačním systémem Android a iOS hrálo roli několik faktorů. Ten asi objektivně nejdůležitější je fakt, že systém Android má největší zastoupení na světě v konkurenci všech operačních systémů a to v současné době 87,0 % [10]. Tedy vznikající aplikace má mnohem větší šanci se rozšířit a oslovit více lidí. Abych co nejvíce podpořila tento fakt, nastavila jsem Minimum SDK na API 16: Android 4.1.x (Jelly Bean). To znamená, že 99,6 % zařízení, která jsou aktivní na Google Play, budou mít možnost si aplikaci stáhnout. Subjektivním důvodem je moje uživatelská zkušenost a to, že se v tomto operačním systému orientuji, což jistě urychluje vývoj. Ačkoliv většina mobilních aplikací pro operační systém Android je naprogramovaná v jazyce Java, současným hitem je jazyk Kotlin. Tento jazyk běžící na Java Virtual Machine (tedy interoperabilní s knihovnami Javy) s možností kompilace do JavaScriptu byl v roce 2017 vyhlášen firmou Google za oficiální jazyk Androidu [11].

Kotlin

Tento relativně nový programovací jazyk je nejčastěji srovnáván s Javou. Důvodem bude fakt, že je jejím nástupcem pro vývoj aplikací pro Android a také běží na JVM. Kotlin má samozřejmě jinou syntaxi než Java, ale jsou vzájemně interoperabilní. Lze je tedy v rámci jednoho projektu kombinovat. Kotlin je objektově orientovaný jazyk stejně jako Java.

Na rozdíl od Javy neobsahuje Kotlin ternární operátory pro podmíněné výrazy, statické metody a atributy, ty jsou nahrazené tzv. `companion object` a v neposlední řadě kontrolované výjimky, všechny třídy dědí z `Throwable`.

Naopak obsahuje některé prvky, které Java neobsahuje. Ta nejvíce popisovaná a oceňovaná je **typový systém**, který eliminuje problémy s hodnotami `null` v proměnných. Pokud je proměnné nullable, je za datový typ při deklaraci umístěn otazník. U těchto proměnných je třeba zajistit, že při přistupování k její hodnotě zde `null` nebude. To lze zajistit čtyřmi způsoby:

1. Podmínkou: Standardní kontrolou, že v proměnné není `null`. Kompilátor tuto kontrolu zachytí a povolí s proměnnou dále pracovat.

2. Safe call operátorem „?“: Ten zajistí, že pokud je proměnná, na kterou je funkce volána, `null`, tak celý výraz vrátí `null`. Tohoto se využívá hlavně v delších řetězcích.

3. Operátorem „!!“: Ten říká kompilátoru, že si je programátor jistý tím, že hodnota proměnné určitě není `null`. Pokud je, tak vyhodí výjimku.

4. Elvis operátorem „?:“: Ten zajišťuje, že pokud je v proměnné hodnota `null`, tak bude nahrazena jinou hodnotou nacházející se na pravé straně operátoru.

Druhým velmi užitečným novým prvkem v jazyce Kotlin jsou **datové třídy** tzv. `data class`. Ty slouží primárně k ukládání dat. Jejich vytvoření je velmi jednoduché, stačí deklarovat atributy třídy v konstruktoru. Dalším prvkem je tzv. destrukturalizace objektů, kdy lze jedním příkazem rozdělit jeden objekt do více proměnných.

V jazyce Kotlin jsou **proměnné** označené dvěma klíčovými slovy. Ty, jejichž hodnotu nelze dále měnit, slovem `val` a ty druhé `var`. Pokud je proměnná třídní, kompilátor k ní automaticky vygeneruje getter a setter.

Velice praktickým prvkem je také **odvození typu** proměnné. Pokud je kompilátor schopný odvodit, jaký datový typ má být do proměnné dosazen, není nutné jej deklarovat. Jinak zobrazí chybu.

Dále zde existuje také tzv. **chytré typování** „Smart cast“. Pokud v kódu zkontroluji, že proměnná je určitého typu, není třeba ji explicitně přetypovávat, kompilátor to udělá automaticky. Toto vylepšení funguje pouze v případě, že má kompilátor jistotu, že se proměnná nemůže mezi kontrolou typu a použitím změnit.

V Kotlinu lze také **přetížit** některé operátory a také **rozšiřovat existující třídy** novými funkcemi bez toho, aby bylo třeba z dané třídy dědit nebo do ní zasahovat. [12]

Android Studio

Jako vývojové prostředí jsem jednoznačně zvolila Android Studio, které je oficiálně určené pro tvorbu aplikací pro Android. Je založené na platformě IntelliJ IDEA, podporuje nativně jazyky Java, Kotlin a C++ a je v něm možnost spuštění vyvíjené aplikace v emulátoru, což je velmi užitečné hlavně pro různé testovací scénáře. Také umožňuje podepsání aplikace klíčem, což je naprostou nutností pro nahrání souboru na Google Play, dále vykreslení designu layoutu při návrhu UI nebo sledování různých parametrů aplikace za běhu pomocí nástroje Profiler.

Android

Programování aplikace pro operační systém Android má svoje specifika. Zde nastiňuji pouze rámcově ty nejdůležitější principy, které jsem využila při své práci.

Základním stavebním prvkem jsou **aktivity**. Ty reprezentují zobrazení uživatelského rozhraní, zpracovávají odezvy na akce uživatele a předávají jejich řešení do dalších vrstev systému. Většinou platí, že každá obrazovka je reprezentována jednou třídou, která dědí ze tříd typu `Activity`. Životní cyklus aktivity, který je zobrazen na obrázku 3.8, pomáhá v pochopení běhu aplikace a pomocí metod, které jsou volány v jednotlivých fázích, lze provádět potřebné operace v kódu v ten správný okamžik.

Podobnou funkci mají i **fragmenty**. Ty reprezentují menší stavební jednotku obrazovky, která má zpravidla nějaké specifické chování nebo je sdílena více aktivitami. Má také svůj životní cyklus, ten je ale velmi podobný tomu u aktivity.

Uživatelské rozhraní je definováno ve zvláštním XML souboru. V tomto souboru lze pomocí předdefinovaných komponent vytvářet různé typy zobrazení. Jednotlivé komponenty dědí ze třídy `View`: TextView zobrazuje text, ImageView obrázky. Další často používané komponenty jsou ještě například tlačítka Button. Z těchto souborů se pak tyto prvky načítají do tříd, hlavně aktivit a fragmentů, kde jsou dále použity. Dříve se pro vyhledávání komponent UI pro použití v aktivitě nebo fragmentu používala funkce `findViewById()`, ale nyní je možnost XML soubor přímo importovat a využívat `View` rovnou jako proměnnou. Tato funkcionalita je součástí knihovny Android KTX, která rozšiřuje Android knihovnu o funkcionality, které zpřehledňují kód a usnadňují programování. [13]

Uspořádání obrazovky je zajištěno pomocí různých typů **layoutů**. Dříve se hodně využívalo mnohonásobné zanořování menších layoutů, ale v současné

době je propagován jednoduchý layout: `ConstraintLayout`. Tato relativní novinka je součástí softwarových komponent souhrnně označovaných jako Android Jetpack, které mají usnadnit vývoj aplikací pro tuto platformu. Využívá navazování jednoduchých komponent do vzájemné interakce v horizontálním a vertikálním směru. Aby byla obrazovka responzivní pro širokou škálu velikostí displejů, používají se spolu s tímto layoutem ještě vodící čáry `Guideline`, ke kterým se dají také kotvit elementy a kterým se dá nastavit horizontální či vertikální pozice pomocí procent. Druhým hojně používaným zobrazením je `RecyclerView`, které umožňuje zobrazit seznam velkého množství prvků. [14]

3.4 Návrhový vzor

Protože programování pro operační systém Android má svá specifika, je nutné si o to lépe promyslet a využít efektivní mechanismus oddělování zodpovědností. Intuitivní programování totiž v tomto případě vede nutně k aplikaci, která bude pravděpodobně obtížně rozšiřovatelná, spravovatelná, testovatelná a její kód bude nepřehledný. Vývojáři pro Android sahají tedy k několika typům návrhových vzorů: MVC, MVP a MVVM. Postupně je všechny představím a potom si jeden z nich zvolím jako mně nejpřívětivější.

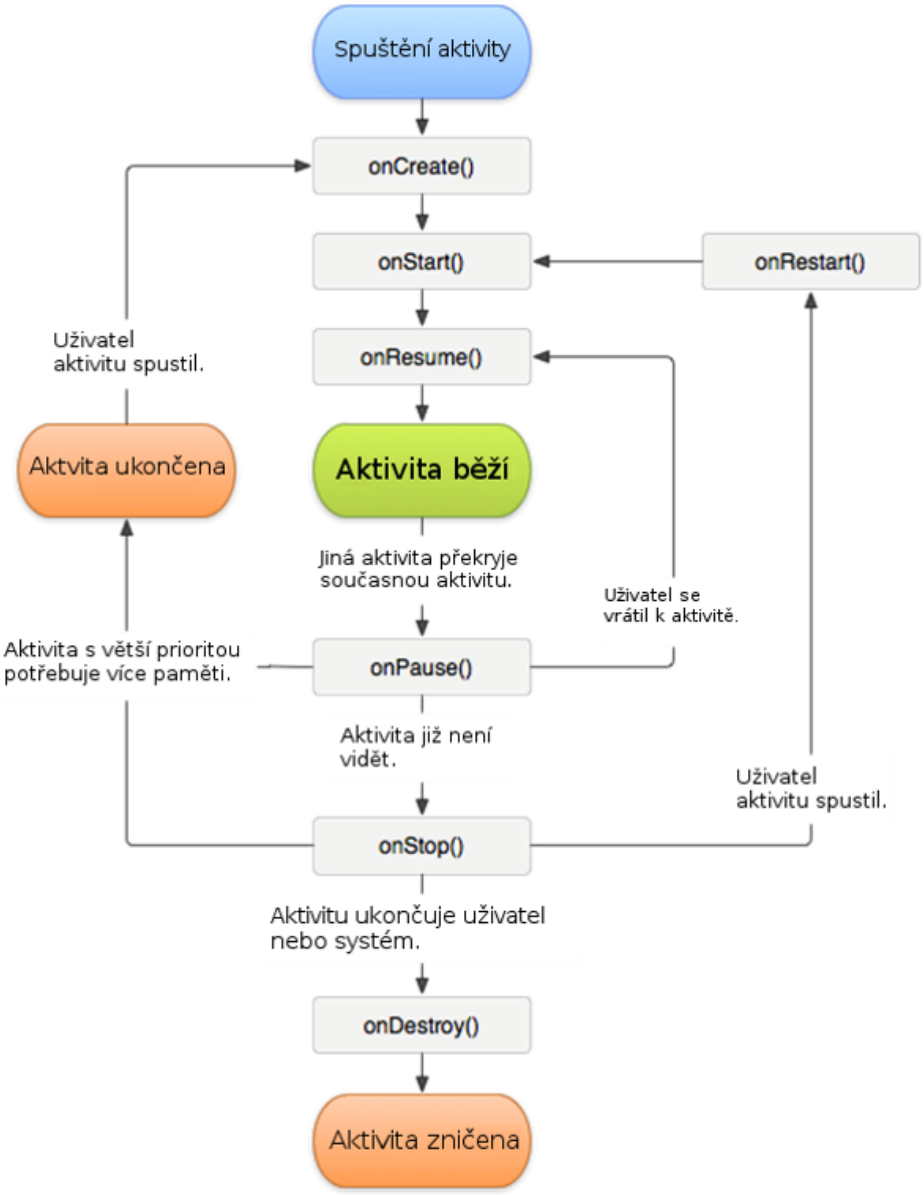
MVC

Model-View-Controller (MVC) je z uvedených návrhových vzorů historicky nejstarší používaný návrhový vzor. V literatuře se dá narazit na jeho různé interpretace, kdy největší třecí plochou je rozložení zodpovědností mezi třemi vrstvami, ze kterých se skládá. Zobrazuje se tak, jak je vidět na obrázku 3.9. Ten je zjednodušený v tom směru, že zatímco Model je typicky jeden, vrstev View a Presenter bývá standardně více.

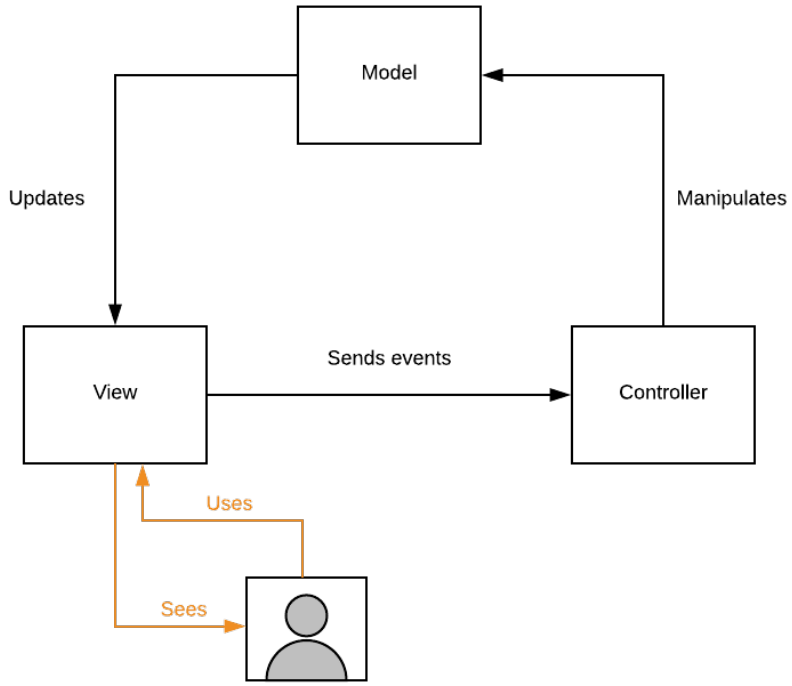
Model slouží hlavně jako nositel logiky aplikace. Jeho úkolem jsou hlavně výpočty a databázové dotazy, popřípadě samotné uložení dat. Nezajímá se o zobrazení dat, ani jejich přijetí. Zpracovává parametry, které dostává a vydává přepočtená data zpět do systému. Často se v tomto modelu využívá návrhového vzoru Observer, kdy upozorňuje vrstvu View, že se něco změnilo a má se překreslit.

View má za zodpovědnost zobrazovat výstup uživateli, starat se o funkčnost uživatelského rozhraní. Neví, odkud mu data přišla, jeho jediným úkolem je je zobrazit.

Controller slouží jako prostředník, přes kterého jdou všechny akce od uživatele. Zachytává je, zpracovává a předává dál Modelu.



Obrázek 3.8: Životní cyklus aktivity [15]



Obrázek 3.9: Zobrazení rozložení zodpovědnosti mezi vrstvami MVC návrhového vzoru [16]

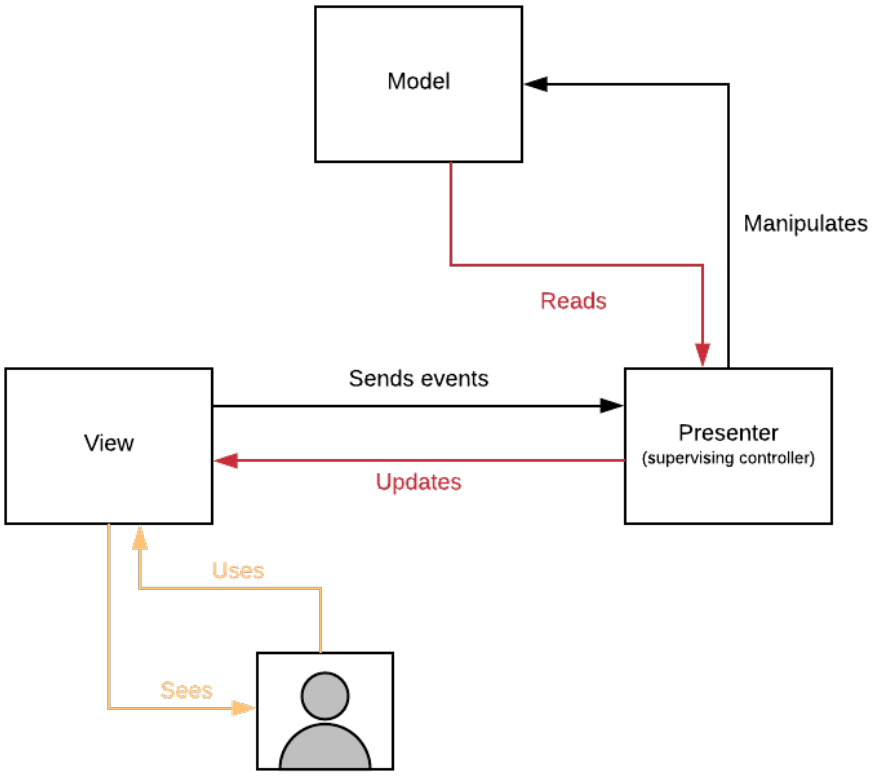
MVP

Model-View-Presenter (MVP) vychází ideově z předchozího modelu. Pouze přesouvá více zodpovědnosti na vrstvu Presenteru. Vše je vidět na obrázku 3.10.

Model má stejnou zodpovědnost jako ve vzoru MVC. Jeho jediná komunikace ovšem probíhá s vrstvou Presenteru.

View je stejně jako v předchozím případě zobrazovací vrstva. Jakékoliv akce uživatel provede, přesouvá k Presenteru.

Presenter v tomto případě komunikuje jak s Modelem, tak s View. Jeho úkolem je mediovat jejich komunikaci. Zpracovává akce od View a dotazuje se na data z Modelu. Jakmile dostane od Modelu odpověď, posouvá ji zpátky k View.



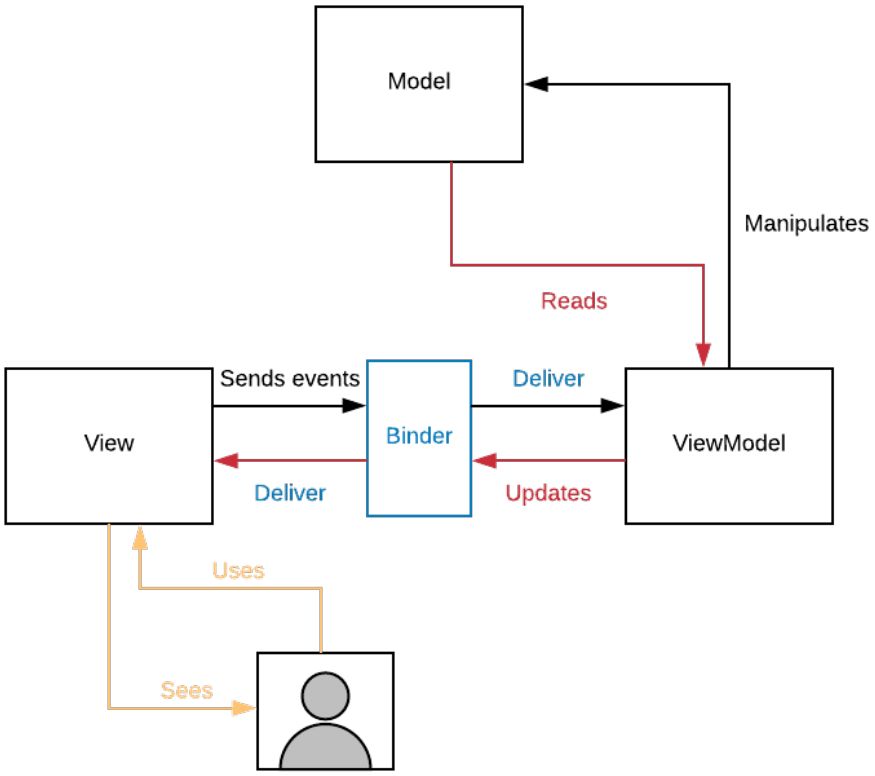
Obrázek 3.10: Zobrazení rozložení zodpovědnosti mezi vrstvami MVP návrhového vzoru [16]

MVVM

Model-View-ViewModel (MVVM) je nejmladším návrhovým vzorem. Začínal se rozvíjet někdy od roku 2005, kdy jej zformalizoval tým Microsoftu. Snaží se o ještě větší míru oddělení zodpovědností než předchozí dva modely. Představuje se sice jako třívrstvý, ale reálně má ještě čtvrtou vrstvu zvanou „Binder“. Ta představuje mechanismus provázání vrstvy View a ViewModel, viz obrázek 3.11. Tato vrstva je většinou spravována pomocí knihoven třetích stran, takže s ní vývojář nemá žádnou práci. U platformy Android je jí knihovna DataBinding [17]. [16]

Model se stává pouze držitelem dat aplikace.

View je u tohoto typu modelu velmi odlehčené. Stará se opravdu pouze o zobrazení uživatelského rozhraní.



Obrázek 3.11: Zobrazení rozložení zodpovědnosti mezi vrstvami MVVM návrhového vzoru [16]

ViewModel obstarává veškerou business logiku aplikace a částečně i zobrazování. Přizpůsobuje informace obsažené v Modelu, aby je View mohlo zobrazit. Také transformuje vstupní data, aby je Model mohl uložit.

Výběr

Vzor MVC je vhodný pro malé aplikace. U těch větších vede k tomu, že aktivity, tedy View vrstva, hromadí kód a stávají se nepřehlednými. View vrstva zobrazuje totiž uživatelské rozhraní a komunikuje ještě s Modelem. Vzor MVP již odlehčuje vrstvě View z předchozího příkladu, ale stále bojuje s velkou mírou zodpovědnosti vrstvy Presenter. MVVM vzor mi přijde pro můj typ aplikace zbytečně komplikovaný. Také nejsem schopná docenit často prezentovanou genialitu vrstvy ViewModel. Volím proto návrhový vzor MVP, který mi přijde jasně uchopitelný a pro moji aplikaci dobře použitelný. [16]

3.5 Rozšířená realita

Pojem rozšířená realita neboli „Augmented Reality“ (AR) se v současné době dostává do popředí zájmu veřejnosti laické i odborné. Z projektu původně pro vojenské a lékařské účely se postupně stává zábava pro masy. AR je definována jako technologie, která opravdový obraz v reálném čase doplňuje o počítačem generované prvky jakými jsou text, video či trojrozměrné objekty. Od virtuální reality se liší tím, že svět kolem nás zachovává svou opravdovost a je pouze doplněn o další prvky.

Aplikace pracující s rozšířenou realitou lze rozdělit do několika kategorií:

- **Marker-based** aplikace fungující s pomocí kotev, které fungují na principu detekce objektu v obraze. Jako kotvy se využívají unikátní objekty, na jejichž místo jsou pak umístěny vkládané objekty. Využívají se zejména QR kódy, obrázky či speciální tvary.

- **Markerless** aplikace bez kotevních prvků, kdy lze 3D objekt umístit kamkoliv do obrazu kamery bez jeho návaznosti na okolí.

- **Markerless with geometric environment understanding** aplikace bez kotevních prvků respektující geometrické tvary dokáže rozpoznat 3D kontext prostředí, do kterého je objekt vkládán. To slouží k lepší interakci s komponenty reálného světa.

- **Markerless with spatial understanding** aplikace bez kotevních prvků s prostorovou orientací, kdy je aplikace pomocí speciálních algoritmů nejen schopná detekovat tvary prostředí, ale dokonce je zařadit do různých kategorií a tím vytvořit ještě lepší provázanost generovaného objektu s reálným prostředím.

Tyto aplikace využívají ke svému fungování celé řady senzorů: akcelerometr, gyroskop, RGB kameru, světelný senzor, mikrofon, GPS a další. Proto jsou vhodnými médii pro tuto technologii mobilní telefony a tablety. Existuje tedy několik balení softwarových nástrojů (SDK) umožňujících integrovat AR do operačních systémů. Pro Windows existuje platforma Windows Mixed Reality, pro iOS ARKit a pro Android se primárně využívá ARCore, ačkoliv je multiplatformní. [18]

ARCore

Aplikace pro ARCore lze vyvíjet v jazyce Kotlin nebo Java. Proto se stal i mou primární volbou. Celý framework je velmi propracovaný, obsahuje detekci jak horizontálních, tak i vertikálních a šikmých povrchů. Pomocí technologie COM (Concurrent Odometry) je schopný sledovat pozici zařízení vzhledem

k okolí na 6 osách (6-degrees of freedom). Funkcionalitu zakládá na 3 principech: detekce osvětlení (Light Estimation), porozumění okolí (Environmental Understanding) a detekci pohybu (Motion Tracking). [19]

Jedinou jeho nevýhodu spatřuji v tom, že si Google sestavil seznam zařízení, která tuto technologii podporují [20]. Vždy při spuštění funkcionality ARCore ověřuje název zařízení a podle toho ji buď zpřístupní, nebo ne. Zdůvodňuje to nutností dostatečné hardwarové a softwarové vybavenosti přístroje. To dělá bohužel vývoj trochu problematický. Nicméně seznam zařízení se stále rozrůstá a AR je technologií budoucnosti, takže za pár let již bude nejspíše dostupná pro většinu mobilních zařízení.

3.6 Monetizační strategie

Na poli aplikací pro mobilní zařízení má vývojář v podstatě tři možnosti, jak získat za svou aplikaci nějaký zisk. Buď za ni budou uživatelé platit již při stažení nebo její součástí bude nějaká forma mikrotransakcí, popřípadě se v ní budou zobrazovat reklamy třetích stran. V dnešní době se využívá všech možností.

Hry, které produkují větší společnosti a již mají své jméno na trhu, což je pro uživatele zárukou kvality, mohou být **placené rovnou** (Minecraft, Machinarium, Assasin's Creed atd.). Často se také využívá tzv. **Freemium model** systémem placené prémium verze bez reklam nebo s funkcionalitami navíc, protože běžný uživatel Google Play je nedůvěřivý a nerad kupuje zajíce v pytli, ač se většinou jedná o investici v řádu sta korun.

Velmi často používaným monetizačním prvkem je zobrazování **reklam** třetích stran. Ty mohou být součástí vzhledu aplikace nebo narušovat její chod svým přehráváním, popřípadě při jejím shlédnutí odměnit hráče nějakou výhodou do hry. Je ovšem důležité dobře vyvážit jejich množství k obsahu hry samotné. Další možností jsou nákupy v aplikaci tzv. **In-App Purchases**. Aplikace je sice zdarma ke stažení, ale nabízí uživateli lákavé možnosti, jak si její užívání usnadnit:

Zpřístupnění dalšího obsahu nebo funkcionalit nabízí uživateli na začátku například několik úrovní obtížnosti nebo pouze několik obrázků, ze kterých si lze vybrat. Pokud ale zaplatí, odemknou se mu další úrovně nebo přidají do výběru další obrázky.

Prodej vylepšujících komponent je hlavně případem her, kdy je uživatel reprezentován virtuálním avatarem. Toho lze různě vylepšit a například obléct, upravit. Tento fenomén je proslulý zejména v Japonsku. Hráči, kteří si nezakoupí dané doplňky, mohou být vyloučeni z komunity nebo pro ně bude hra vlastně nehratelná, protože nebudou konkurenceschopní.

Urychlení času je velmi častým způsobem mikrotransakcí v aplikacích. Používají se hlavně ve hrách, kdy se děj odehrává v čase. Platba pak slouží

k urychlení hry nebo obejití časové restrikce. Hráč si může například zpřístupnit určité elementy hry dříve, dostávat zásoby dvakrát tak často, může být na nějakou dobu nesmrtelný atd.

Také se dá použít model předplatného tzv. **Subscriptions**. Ten využívají především aplikace s pravidelně se aktualizujícím obsahem jako například noviny, časopisy, zprávy atd. Pokračují tím vlastně v trendu nastaveném historicky předplácením si tiskovin měsíčně, ročně apod. [21]

Z vlastní zkušenosti mi přijde nejpřijatelnější model nákupů v aplikaci, který jsem zvolila i pro svou hru. Uživatele neobtěžuje a zároveň má potenciál generovat alespoň minimální zisk. K tomuto účelu použiji nastavení In-App Purchases přímo v Google Play Console při vydání aplikace.

3.7 Ukládání dat

Při programování pro platformu Android má vývojář tři možnosti, jak si ukládat data aplikace. Každá má své pro a proti a je třeba si dobře rozmyslet, která je ta nejvhodnější pro specifickou aplikaci.

Nejjednodušší je využít API **SharedPreferences**. Tento objekt ukládá data do souboru pomocí jednoduchého systému klíč-hodnota a poskytuje jednoduché metody na jejich zápis a čtení.

Druhým způsobem je ukládání **do souboru** ve formátu JSON nebo například XML. Tento způsob je použitelný pro komplikovanější data, která nelze uložit ve formě klíč-hodnota, ale stále nejsou v takovém množství, aby vývojář využil jako úložiště databázi.

Třetím způsobem je již zmíněná možnost ukládat data do **SQLite databáze**. S tou se dá pracovat přímo, nebo přes abstraktní vrstvu nad databází, která se jmenuje Room. Ta umožňuje plynulý přístup k databázi za současného zachování potenciálu SQLite. [22]

Vzhledem k tomu, že moje aplikace potřebuje ukládat pouze property třídy `Dino`, není třeba implementovat cokoliv komplikovanějšího než SharedPreferences.

3.8 Testování

Testování vyvíjených aplikací je naprostou nutností a musí probíhat během všech fází vývoje. Testovací strategie se rozděluje na dvě kategorie: automatickou a uživatelskou.

Automatické testování uživatelského rozhraní lze naimplementovat pomocí frameworku Espresso. Zde se definují jednotlivé testovací scénáře za využití připravených funkcí. Test může probíhat buď na emulátoru, nebo na připojeném zařízení. Pro jednotkové testování lze využít framework Mockito.

Uživatelské testování lze nastavit a distribuovat pomocí Google Play Console. Zde se aplikace nahraje nejdříve do interního testovacího kanálu, kde po schválení a nastavení skupiny testerů může vývojář sdílet odkaz, kde je možné aplikaci stáhnout. Testeři zároveň obdrží e-mailovou adresu vývojáře pro zpětnou vazbu, nejprve ale musí souhlasit se zařazením do testovací skupiny a jsou upozorněni na hrozící rizika neodladěné aplikace. Tato fáze je relativně rychlá. Druhým postupným krokem je opět uzavřené testování, tzv. Alfa verze. Další fází je uvolnění aplikace do otevřeného testování, tzv. Beta verze, kdy už si může aplikaci stáhnout z odkazu širší skupina testerů opět podávajících zpětnou vazbu.

Vzhledem k velikosti vyvíjené aplikace a mé potřebě konzultovat hlavně zábavnost a intuitivnost celého UI jsem testování obstarala nejprve velice pečlivě během vývoje, a následně využila možnosti uživatelského testování pomocí Google Play. Aplikace neprovádí žádné složité výpočty, které by musely být pokryty Unit testy, tento způsob testování mi tedy přijde dostatečný.

Kapitola 4

Implementace hry

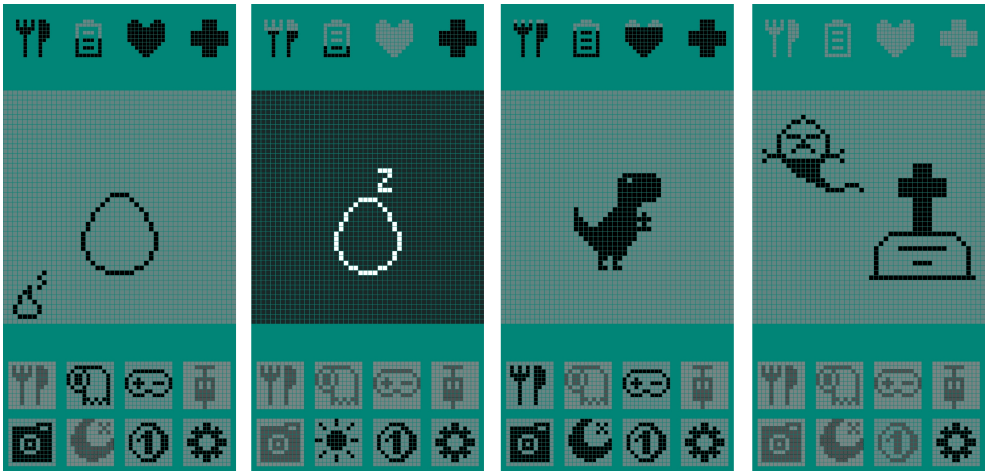
Po důkladné analýze všech možných hledisek důležitých pro tvorbu aplikace přichází samotná realizace. Prvním krokem je tvorba uživatelského rozhraní, které zaujme a bude co nejvíce uživatelsky přívětivé. To, co dělá hru zábavnou zejména pro mladší generaci uživatelů, jsou dobře zvolené zvuky, které dokreslují funkcionalitu uživatelského rozhraní. Dále je třeba si představit samotný systém MVP a také zdůraznit některé implementační zajímavosti. Posledním krokem je zhodnocení testování a samotné vydání hry.

4.1 Uživatelské rozhraní

Vzhledem k tomu, že mým přáním bylo se grafickým zpracováním co nejvíce přiblížit původní elektronické hračce Tamagoči, bylo nutné si grafické podklady vytvořit vlastnoručně pomocí programu Adobe Photoshop. Ten byl použit i pro generování 3D objektů pro rozšířenou realitu. Pro návrh uživatelského rozhraní bylo prioritou, aby hra byla přístupná všechny věkové skupiny, což znamená co nejvíce uživatelsky přívětivé prostředí a co nejméně textu.

Hlavní obrazovka je gró celé hry. Zde se odehrává celý vývoj tvora. Je zde umístěné i ovládání všech položek hry. Vizuálně se skládá ze tří částí, jak je vidět na obrázku 4.1:

1. **Horní lišta:** Zde jsou umístěny čtyři ukazatele stavu tvora, jeho sytosti, energie, lásky a zdraví. Když jsou plné, jsou černé a postupně od zhora šednou podle míry úbytku.
2. **Střední část:** Hlavní plocha hry. Slouží k zobrazování animací vývojových stádií tvora. Také se tu ukazují speciální symboly a při přepnutí na spánek tvora se mění do tmavého barevného schématu.
3. **Spodní tlačítka:** Dvě řady tlačítek slouží k ovládání všech akcí ve hře. Pokud je funkce nedostupná, tlačítko je zašedlé a neaktivní. Je zde tlačítko pro výběr jídla, úklid, volbu hry, podání medicíny,



Obrázek 4.1: Hlavní obrazovka

využití možnosti vyfotografovat tvora, přepnutí dne/noci, nákup a nastavení.

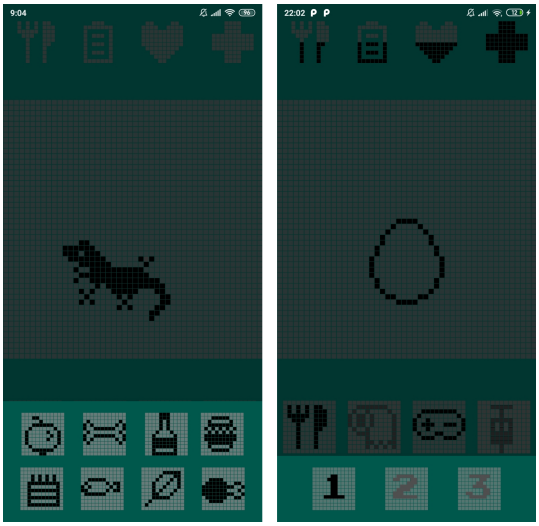
Fragmenty jsou použity na dvou místech UI. Jednak pro volbu potraviny, kterou tvor dostane a také pro výběr minihry, kterou si s tvorem lze zahrát. Jsou použity pro zvýšení dynamičnosti ovládání hry. Překrývají pouze část spodní obrazovky a vyjíždějí ze spodního okraje. Hráč tedy neztrácí kontakt se svým mazlíčkem. Jejich vzhled v praxi lze vidět na obrázku 4.2.

Hry jsou celkem tři a jejich dostupnost je ovlivněna vývojovým stádiem tvora. Jsou blíže popsány v kapitole 3.3. UI je co nejjednodušší, žádné zbytečné prvky. Vše tvoří elementy, se kterými se dá interagovat a které jsou nezbytné pro vyřešení hry. Ukázku si lze prohlédnout na obrázku 4.3.

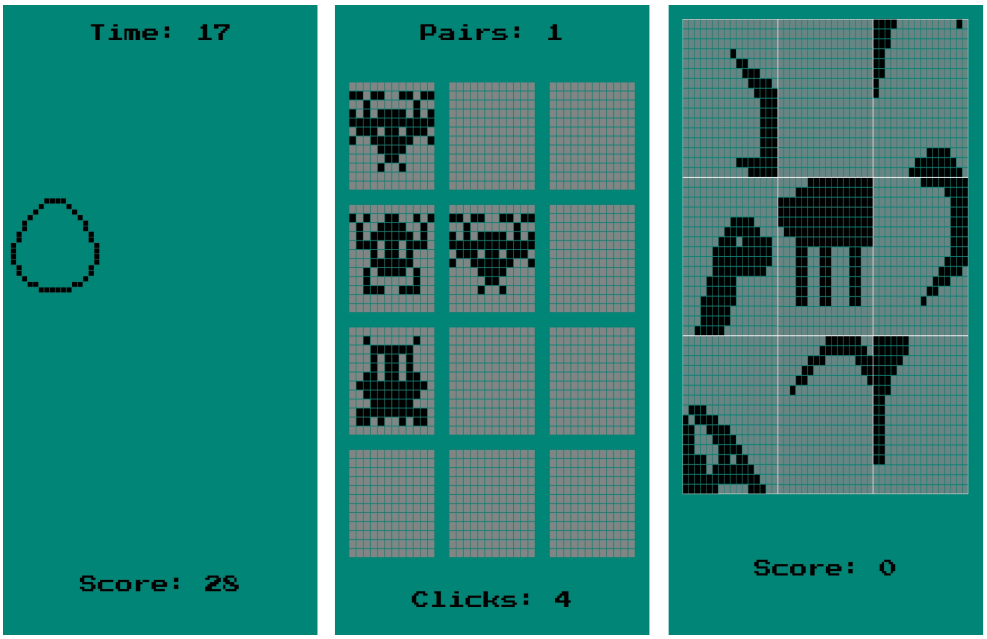
Další aktivity reprezentují rozšířenou realitu, nákupy a nastavení. Aby bylo celé ovládání opravdu co nejintuitivnější, je pro pořízení obrázku dostupné na obrazovce jen tlačítko spouště. U nákupů se načtou přístupné položky, které fungují jako tlačítka. Jejich stisknutím se načte dialog Google Play, který vyzve k platbě. Nastavení otevírá další jednoduchou obrazovku se čtyřmi výmluvnými tlačítky: nová hra, odejít, info a zvuk. Vše je zobrazeno na obrázku 4.4.

4.1.1 Zvuky

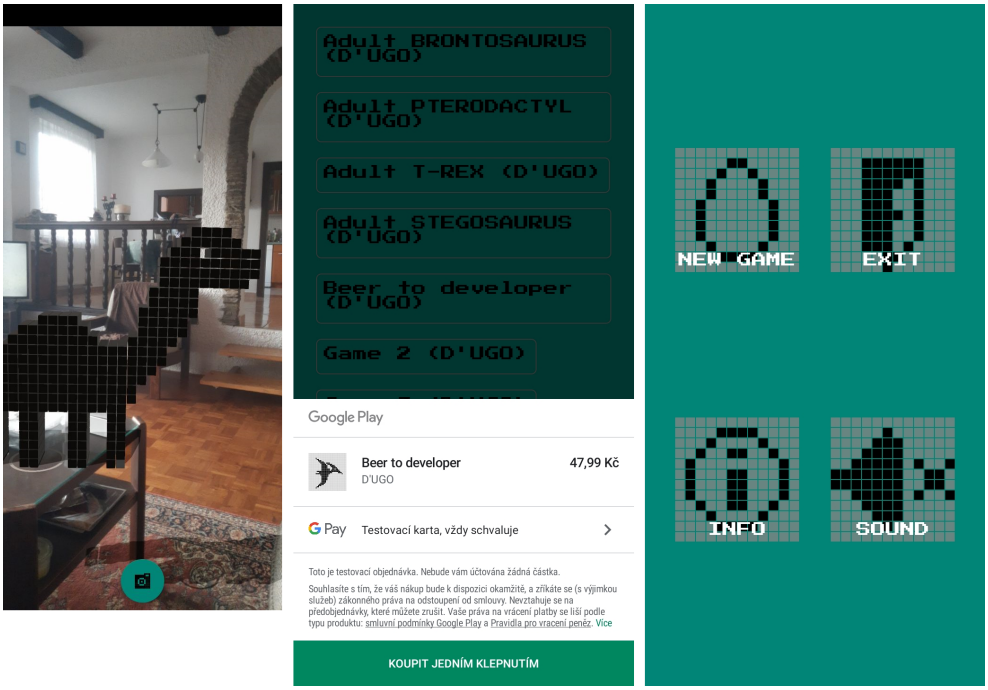
Atmosféra hry je dotvořena zvuky. Ty jsou přehrány při spuštění nové hry, změně vývojového stádia tvora, při přechodu do spánku, krmení, uklízení, podání medicíny, stisku spouště fotoaparátu, vzniku exkrementu a smrti. Jsou



Obrázek 4.2: Fragmenty sloužící k výběru jídla a minihry



Obrázek 4.3: Uživatelské rozhraní miniher



Obrázek 4.4: Uživatelské rozhraní rozšířené reality, nákupů v aplikaci a nastavení

také použity pro minihry. Zvuky jsem získala volně dostupné z internetové databáze *https://freesound.org/*.

Implementovány jsou pomocí Android knihovny MediaPlayer v pomocné třídě `SoundManager`. Pomocí té se pak v aktivitě, která zvuk vyžaduje, zvuk načte a v určený okamžik přehraje. V metodě `onDestroy()` dané aktivity je pak `SoundManager` zastaven a uvolněn, aby nedocházelo ke zbytečnému přeplňování paměti.

Zvuky lze v aplikaci samozřejmě zapnout a vypnout.

4.2 MVP

Dle předchozí analýzy jsem tedy zvolila návrhový vzor MVP, který je použit pro implementaci všech aktivit aplikace. Nejen tedy hlavní osy aplikace zahrnující model `Dino`, ale i aktivit všech miniher.

Zde uvádím bližší popis pouze hlavní osy s tím, že logika implementace ostatních míst výskytu vzoru MVP je analogická. Celkový obraz demonstruje obrázek 4.5. Zde je patrné, že každou vrstvu jsem umístila do vlastního balíčku.

Balíček view obsahuje třídu `DinoActivity`, která obsluhuje uživatelské akce a zobrazuje UI. Zastupuje hlavní obrazovku celé aplikace. Jako svou

třídní proměnnou obsahuje třídu `DinoPresenter`, která zajišťuje předávání uživatelských akcí dále do samotného modelu. Zároveň tato třída implementuje rozhraní `DinoContract.View`, které je jednou ze dvou částí rozhraní `DinoContract`.

Rozhraní `DinoContract` má za cíl zpřehlednit vztah mezi vrstvou view a presenter tak, že obsahuje dvě rozhraní. První rozhraní `View` obsahuje všechny metody, které používá presenter na své vlastní view a musí v něm být tedy implementovány. Druhé rozhraní `Presenter` popisuje naopak metody, kterými view volá na presenter a které v něm tedy musí být nutně implementovány. Výhodou tohoto přístupu je kromě větší přehlednosti i snadná možnost výměny celého view za jiné pouze implementací všech metod v `DinoContract.View` a zároveň tím, že se pro komunikaci s presenterem použijí pouze metody uvedené v `DinoContract.Presenter`. Další výhodou je snadná testovatelnost.

Třída `DinoPresenter` reprezentující samotný presenter samozřejmě implementuje `DinoContract.Presenter` dle výše zmíněné logiky. Zároveň si drží referenci na třídu `Dino`, která představuje vrstvu MVP model. Tím ovládá celou business logiku aplikace. Dále je potomkem abstraktní třídy `BasePresenter<V>`, která jí zajišťuje přístup k jejímu view.

Abstraktní třída `BasePresenter<V>` je společná pro všechny aktivity v aplikaci. Její cíl je jediný, a to obalit přijaté view třídou `WeakReference<V>` z balíčku `java.lang.ref`, a poté ho předávat svým potomkům. Používám slabou referenci proto, že konkrétní view je typicky aktivita nebo fragment a při jeho ukončení potřebuji, aby byla správně zpracována garbage collectorem, například když uživatel opustí aplikaci, nebo je přerušena spuštěním jiné aplikace (například hovorem), nebo uživatel přetočí display (tuto možnost tato aplikace nepodporuje).

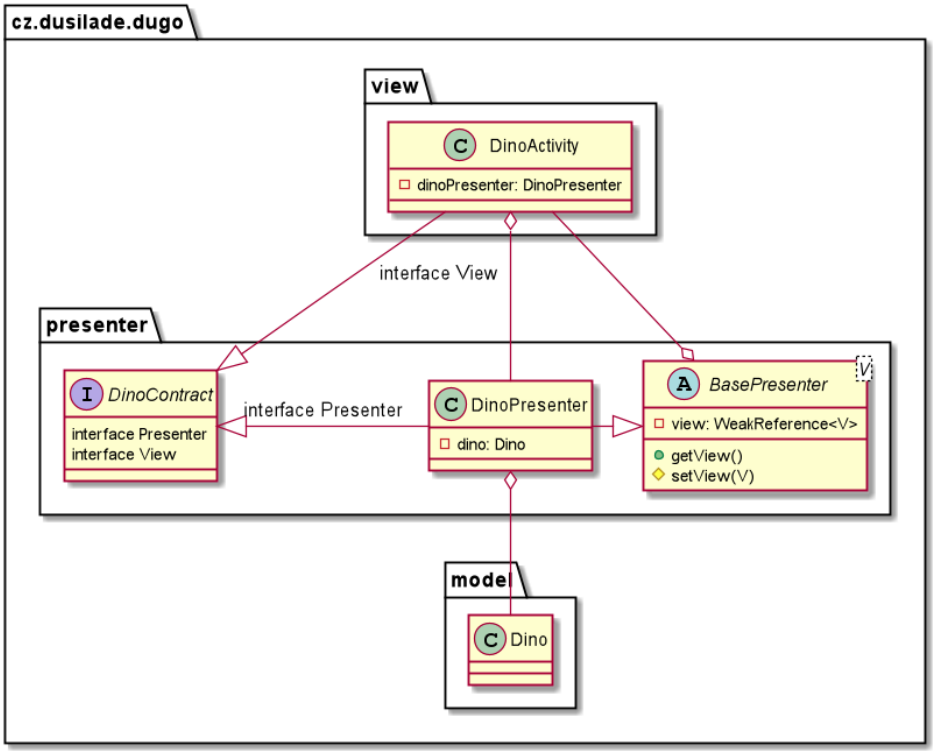
Třída `Dino` představuje vrstvu model a žádnou referenci na předchozí vrstvy nedrží, což je v souladu s návrhovým vzorem MVP. Funguje tedy pouze na principu přijetí požadavků, jejich zpracování, které se často projeví změnou jeho vnitřního stavu a po jejich vyřízení navrácení vypočtených hodnot.

4.3 Zajímavosti v implementaci

Celá aplikace je strukturou i funkcionalitou klasickým příkladem implementace pro operační systém Android. Přesto zde najdeme i zajímavé přístupy k realizaci, které bych chtěla představit.

4.3.1 Singleton

Singleton je návrhový vzor, který je tradičně používán pro třídy, u kterých stačí jedna instance pro celou aplikaci. Standardně se takto implementují třídy typu logování a různé managery. V programovacím jazyku Kotlin stačí třídu označit klíčovým slovem `object`, což oproti Javě nebo C++ jejich implementaci značně zjednodušuje.



Obrázek 4.5: UML diagram tříd - MVP

V mé aplikaci je tímto příkladem třída `TimeManager`, která slouží k různým přepočtům časových jednotek. Protože představuje pouze pomocnou třídu bez vlastního vnitřního stavu, je pro ni vhodné použít zmíněný návrhový vzor Singleton.

4.3.2 Čas

Vzhledem k tomu, že aplikace reprezentuje vývoj tvora od vylíhnutí po stav dospělosti, je nutné její velmi úzké spojení s časem. Tím se vlastně řídí ubývání nebo naopak přibývání stavových hodnot potřeb jedince i jeho přeměna na dospělce.

Potřebné sledování času jsem vyřešila jednoduše, ale efektivně. Celé řízení časového managementu se nachází v metodě `onResume()` hlavní aktivity aplikace `DinoActivity`. Ta se spouští pokaždé, když uživatel aplikaci spustí, ať už nově nebo ji pouze vyvolá z pozadí, zavolá se také při návratu na hlavní obrazovku po ukončení jiné aktivity aplikace. Zde dochází k přepočtu času stráveného mimo hlavní aktivitu a předání této informace presenteru a následně modelu. Ten vyhodnocuje tuto změnu dle výše popsaného systému

hry a vrací zpět informace presenteru, který požádá view o překreslení.

Tento přístup počítá s tím, že uživatel nebude trávit hodiny pozorováním hlavní obrazovky, kde probíhá pouze jednoduchá animace v pixelové grafice, ale jeho zájmem bude interagovat s UI hry, tedy plnit potřeby svého mazlíčka pomocí krmení a her nebo jej fotit v různých situacích apod. Jakákoliv tato akce vede následně zpět na hlavní obrazovku, tím vyvolá metodu `onResume()` a upraví se vše potřebné dle časové prodlevy.

Kapitola 5

Testování a vydání hry

V roce 2012 došlo k nahrazení služby Android Market distribuční platformou Google Play. Ta funguje na principu cloudu a stará se o to, aby na všech zařízeních, kde je použit stejný Google účet, docházelo k synchronizaci zakoupených položek. Google Play totiž umožňuje nákup a stahování nejen aplikací, ale i filmů, hudby a knih.

Vzhledem k tomu, že aplikace negeneruje žádnou relevantní zpětnou vazbu ani zisk, dokud není nahrána na Google Play, bylo třeba ji touto cestou nejprve důkladně otestovat, a potom vydat do veřejného prostoru. Tento proces se ukázal překvapivě komplikovaný, a proto jsem se rozhodla jej zařadit do samostatné kapitoly.

5.1 Vydání aplikace na Google Play

Pokud chce vývojář nahrát svou aplikaci na Google Play, musí projít níže popsaným řízením. Nejprve je třeba vytvořit vývojářský účet. Tedy souhlasit s distribuční smlouvou pro vývojáře, zaplatit registrační poplatek a vyplnit osobní údaje včetně adresy trvalého bydliště, která bude následně zobrazena na hlavní stránce aplikace.

Následně je třeba vyplnit název aplikace, její popis v krátké (80 znaků) a dlouhé verzi (4 000 znaků), nahrát její ikonu, grafiku (přesně 1024 px x 500 px), minimálně tři snímky obrazovky. Pak je třeba zvolit kategorii aplikace a vyplnit dotazník o hodnocení obsahu aplikace. Google Play poté vyhodnotí, pro jaké věkové skupiny mezinárodně má být dostupná. V případě, že by aplikace mohla být potenciálně zajímavá pro děti, obsahuje povolení pro GPS, kameru nebo další zdroj citlivých informací, je třeba ještě vytvořit webovou stránku s právním dokumentem o pravidlech užití a její url adresu přidat k registraci aplikace, jinak může dojít k jejímu neschválení, odstranění z Google Play a zablokování vývojářského účtu.

Vzhledem k tomu, že aplikace obsahuje In-App Purchases, bylo třeba tyto produkty také nastavit před vydáním aplikace. Vkládají se tam unikátní ID

jednotlivých položek, jejich název, popis a cena. ID se následně spáruje v kódu aplikace. Položky musí být nastaveny jako aktivní, aby bylo možné je do aplikace načíst.

Pak už je možné vytvořit vydání aplikace. Existuje několik možností, které jsem popsala v kapitole 3.8. Já jsem ještě využila možnosti vložit několik apk souborů pro jednu aplikaci. Většinou se toto využívá například pro různé lokalizace, ale pro mě to byla cesta, jak obejít selekci ARCore dostupných zařízení. Vzhledem k tomu, že rozšířená realita je jenom okrajovou funkcionalitou celé aplikace a hra je zajímavá i bez ní (původní verze hry ji také neobsahovala), přišlo mi nesmyslné, omezit uživatele jenom na ty, jejichž telefon je registrován na seznamu povolených mobilních telefonů. Nahrála jsem tedy dvě verze apk, jednu s ARCore a druhou bez její podpory. Aplikace je tedy dostupná pro všechny uživatele bez výjimky.

5.2 Testování pomocí Google Play

Nejprve jsem aplikaci vydala v kanálu interního testování a zpřístupnila ji deseti testerům. Z jejich zpětné vazby jsem zjistila, že aplikaci nelze nainstalovat, pokud nesplňují požadavky pro ARCore. V dalším vydání jsem toto vyřešila nahráním dvou apk souborů pro různá zařízení. Další zpětná vazba zaznamenala nefunkční načítání nákupních položek, pád aplikace při spuštění třetí hry, moc velké písmo v informaci o aplikaci a na tlačítkách výběru her. Opět jsem vydala další verzi. Zde jsem eliminovala nahlášené chyby, ale vyskytly se další. Nákupní položky se již načítaly, ale nebylo možné na ně kliknout a tedy ani cokoliv koupit. Dále testeři hlásili moc krátký interval po dohrání minihry a vrácení se na úvodní obrazovku, protože chtěli sdílet dosažená skóre a nedařilo se jim pořídit screenshot obrazovky. Také po nakrmení tvora do plného stavu se sice změnilo tlačítko pro jídlo na nepřístupné, ale stále jej šlo používat. To jsem také opravila.

Pak jsem aplikaci vydala již v beta verzi v kanálu otevřeného testování. Zde se zpětná vazba již zaměřila na budoucí vylepšení a nápady, ale na funkčnost nebyly stížnosti.

Aplikaci jsem tedy vydala v produkčním kanálu. Zatím má pouze patnáct stažení, ale publikace je velmi čerstvá a zatím jsem nepodnikala žádnou propagaci. Přírůstky uživatelů v čase za měsíc prosinec získané z konzole Google Play jsou zobrazeny na grafu 5.1.

Obrázek 5.1: Přírůstky uživatelů aplikace DUGO

Kapitola 6

Plánovaná a další budoucí vylepšení

Během testování se objevilo velké množství zajímavých nápadů, jak aplikaci vylepšit. Mým primárním cílem bude zpřístupnění podobné funkcionality, kterou poskytuje ARCore i uživatelům s telefonem, který nepatří do seznamu těch, na kterých je rozšířená realita povolena.

Dále bych chtěla udělat minihry více variabilní. U první hry by uživatel pronásledoval vždy to stádium tvora, ve kterém se zrovna nachází a možná i více úrovní, kdy se bude objekt zmenšovat a zrychlovat. U druhé hry a třetí hry pak vytvořit větší galerii obrázků. U třetí hry by si pak mohli uživatelé zvolit, jaký obrázek bude předmětem hry, u druhé hry pak vybrat ty obrázky, které se budou náhodně dosazovat na karty pexesa. U pexesa by se mi dále líbilo, aby si uživatelé mohli volit z více velikostí hrací plochy, tedy počtu kartiček. Třetí hra by mohla mít ještě variantu, kdy je jedna pozice obrázku prázdná a hráči posouvají části hracího pole pouze pomocí tohoto prázdného místa. To by pro ně znamenalo ztížení celé hry.

Také se samozřejmě nabízejí různé možnosti personalizace tvora, například volbou jeho jména, pohlaví apod. Také by bylo možné vytvořit galerii již vychovaných dinosaurů. Dle testerů velmi oceňovaným vylepšením by byla možnost interakce mezi dvěma dinosaury různých uživatelů ideálně prostřednictvím rozšířené reality (souboj) nebo například jejich páření a následně líhnutí potomků.

Jak je vidět, budoucích možných vylepšení je opravdu velká řada a já se opravdu velmi těším na další práci s touto aplikací.

Závěr

Celá práce obsahuje kompletní vývoj aplikace pro operační systém Android v programovacím jazyku Kotlin. Vzniklý produkt je hra inspirovaná elektronickou hračkou Tamagoči, která je vylepšená o prvek rozšířené reality a tři minihry, které se uživateli zpřístupňují postupně s vývojem tvora, o kterého se stará.

Cíle definované pro mou bakalářskou práci jsem naplnila beze zbytku. Oproti původnímu plánu se mi ještě podařilo navíc otestovat hru v distribuční platformě Google Play a publikovat ji. Google Play jsem také využila pro nastavení nákupů uvnitř aplikace, které jsou také nad rámec zadání.

V teoretické části práce jsem představila fenomén Tamagoči a následně provedla srovnání již existujících her na jeho principu. Zaměřila jsem se speciálně na hry s podobnou grafikou, jakou má původní hračka a jakou jsem zvolila pro svou hru i já. Grafické zpracování totiž z velké části určuje funkční možnosti aplikace.

V praktické části jsem nejprve detailně navrhla systém hry a pak hru implementovala.

Technologicky jsem se řídila zadáním. Nicméně jsem analyzovala zvolené možnosti a zdůvodnila konečnou volbu. Pro aplikaci jsem využila návrhový vzor MVP, který tvoří její kostru. Pro implementaci rozšířené reality jsem pracovala s platformou ARCore. Při nastavení nákupů v aplikaci jsem využila distribuční platformu Google Play, která toto podporuje.

Hru jsem otestovala pomocí několika verzí vydaných v kanálu interního testování na Google Play. Zpětnou vazbu od testerů jsem pečlivě analyzovala a zapracovala připomínky. Poté jsem vydala beta verzi v kanálu otevřeného testování, kde k ní mělo přístup více testerů. Hra byla již bez připomínek, a proto jsem ji publikovala.

Dále jsem zmínila možnosti další práce s aplikací. Některé návrhy plánuji realizovat sama v blízké budoucnosti, aby hra byla variabilnější a snad i přitažlivější pro uživatele. Jiné návrhy by mohly posloužit jako inspirace pro

další závěrečnou práci.

Celkově mohu zhodnotit, že práce na aplikaci byla velmi přínosná z mnoha hledisek. Pomocí ní jsem se seznámila se světem operačního systému Android, naučila se pracovat s pro mě novým programovacím jazykem Kotlin a prošla si celým vývojovým cyklem aplikace. Vzniklý produkt, hru DUGO, považuji za povedenou a doufám, že zaujme i ostatní. Zatím na ni mám veskrze pozitivní reakce.

Literatura

[1] Wudunn, S.: Hatchling Of Pet Lover Is the Rage Of Toylands. *The New York Times [online]*, září 1997, [cit. 2019-12-07]. Dostupné z: https://www.nytimes.com/1997/09/07/world/hatchling-of-pet-lover-is-the-rage-of-toylands.html

[2] Olivová, J.: Žertovné Ig Nobelovy ceny. *Český rozhlas [online]*, říjen 2011, [cit. 2019-12-07]. Dostupné z: https://vltava.rozhlas.cz/zertovne-ig-nobelovy-ceny-5122743

[3] Mawges: Dogotchi: Virtual Pet. *Google Play [online]*, červenec 2019, [cit. 2019-12-07]. Dostupné z: https://play.google.com/store/apps/details?id=com.mawges.dogotchi

[4] Mubashir, D.: Gig Pet - Virtual Cat. *Google Play [online]*, listopad 2017, [cit. 2019-12-07]. Dostupné z: https://play.google.com/store/apps/details?id=com.xeurikion.catgame

[5] Neurocreativa: Cthulhu Virtual Pet. *Google Play [online]*, leden 2017, [cit. 2019-12-07]. Dostupné z: https://play.google.com/store/apps/details?id=com.Neurocreativa.CthulhuVirtualPet

[6] Apps., J.: Neps: Virtual Pet. *Google Play [online]*, březen 2017, [cit. 2019-12-07]. Dostupné z: https://play.google.com/store/apps/details?id=appinventor.ai_jotakupo2.AAANeps

[7] Magri, R.: DiNostalgia Widget. *Google Play [online]*, květen 2018, [cit. 2019-12-07]. Dostupné z: https://play.google.com/store/apps/details?id=br.com.dinostalgia

[8] Hindy, J.: 10 best virtual pet apps and games for Android! *Android Authority [online]*, duben 2019, [cit. 2019-12-07]. Dostupné z: https://www.androidauthority.com/best-virtual-pet-apps-games-android-98122/

[9] kuma: Co je to herní engine. *České mody.cz [online]*, duben 2016, [cit. 2019-12-14]. Dostupné z: https://www.ceskemody.cz/clanky.php?clanek=56

[10] Smartphone Market Share. *IDC Research, Inc. [online]*, říjen 2019, [cit. 2019-12-14]. Dostupné z: https://www.idc.com/promo/smartphone-market-share

[11] Moskala, M.; Wojda, I.: *Android Development with Kotlin*. Packt Publishing, 2017, ISBN 1787123685 9781787123687.

[12] Reference - Kotlin Programming Language. *Kotlin Language Documentation [online]*, [cit. 2019-12-15]. Dostupné z: https://kotlinlang.org/docs/reference/

[13] Android KTX. *Android Developers [online]*, [cit. 2019-12-15]. Dostupné z: https://developer.android.com/kotlin/ktx

[14] Layouts. *Android Developers [online]*, [cit. 2019-12-15]. Dostupné z: https://developer.android.com/guide/topics/ui/declaring-layout

[15] Frank, J.: Životní cyklus aktivity. *ITnetwork.cz [online]*, [cit. 2019-12-15]. Dostupné z: https://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-zivotni-cyklus-a-novy-projekt

[16] Suica, M.: MVVM and DataBinding: Android Design Patterns. *raywenderlich.com [online]*, duben 2019, [cit. 2019-12-15]. Dostupné z: https://www.raywenderlich.com/636803-mvvm-and-databinding-android-design-patterns

[17] Data Binding Library. *Android Developers [online]*, [cit. 2019-12-15]. Dostupné z: https://developer.android.com/topic/libraries/data-binding/

[18] Smith, M.; Babilinski, K.: *Augmented Reality for Developers: Build practical augmented reality applications with Unity, ARCore, ARKit, and Vuforia*. Packt Publishing, 2017, ISBN 978-1-7872-8643-6.

[19] ARCore overview. *Google Developers [online]*, [cit. 2019-12-16]. Dostupné z: https://developers.google.com/ar/discover/

[20] ARCore supported devices. *Google Developers [online]*, [cit. 2019-12-16]. Dostupné z: https://developers.google.com/ar/discover/supported-devices

[21] 6 BEST APP MONETIZATION STRATEGIES OF 2018. *DA-14 Corp. [online]*, říjen 2017, [cit. 2019-12-16]. Dostupné z: https://da-14.com/blog/app-monetization-strategies

[22] App data and files. *Android Developers [online]*, [cit. 2019-12-18]. Dostupné z: https://developer.android.com/guide/topics/data

Příloha A

Seznam použitých zkratek

GUI Graphical user interface

UI User interface

XML Extensible markup language

MVC Model-View-Controller

MVP Model-View-Presenter

MVVM Model-View-ViewModel

AR Augmented Reality

2D Two-dimensional

3D Three-dimensional

SDK Software Development Kit

API Application Programming Interface

JVM Java Virtual Machine

GPS Global Positioning System

JSON JavaScript Object Notation

QR Quick Response

RGB Red Green Blue

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF