VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MOBILNÍ APLIKACE PRO HRÁČSKOU PODPORU VE HŘE POKEMON GO

MOBILE APPLICATION FOR PLAYER SUPPORT IN POKEMON GO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK PIHRT**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN HRUBÝ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Pihrt Patrik**
Program: Informační technologie
Název: **Mobilní aplikace pro hráčskou podporu ve hře Pokemon Go**
iOS Application for Pokemon's GO Players Community
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte programování aplikací pro iOS, knihovny CloudKit, CoreData a MapKit. Prostudujte knihovny pro analýzu obrazu.
2. Navrhněte aplikaci, která bude evidovat hráčovy úspěchy ve hře Pokemon GO. Aplikace musí hráčovy vstupy přebírat také formou analýzy nasnímaných obrazovek ze hry Pokenom GO.
3. Aplikaci implementujte optimalizavanou pro iPhone.
4. Aplikaci testujte v rámci komunity hráčů Pokemon GO.

Literatura:

- Keur, Ch., Hillegass, A.: iOS Programming: The Big Nerd Ranch Guide, Big Nerd Ranch Guides; 6 edition (January 6, 2017), ISBN-13: 978-0134682334

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Hrubý Martin, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Pokémon Go je jedna z nejúspěšnějších mobilních herních aplikací na světě. Hra je podporována platformami Android a iOS. Tato bakalářská práce se zaměřuje hlavně na získávání potřebných dat pro výpočty ze snímků Pokémonů ze hry. Snímky jsou vkládány do aplikace a výstupem jsou data informující hráče o individuálních hodnotách jeho Pokémonů. Všechny vyhodnocené informace jsou ukládány a je možné k nim kdykoliv znovu přistupovat. Dále hráčům předává informace, které se často mění.

Abstract

Pokémon Go is one of the most successful mobile game applications ever created. The game is supported on Android and iOS platforms. This bachelor's thesis describes the way of finding out certain type of information from game's screenshot. Those screenshots are inserted to the application by user, datas are calculated and Pokémon's individual value is known and showed. Results are saved and can be seen anytime. The application provides the newest gaming information.

Klíčová slova

aplikace pro mobilní zařízení, iPhone, mobilní zařízení, iOS, Pokémon Go, rozpoznávání textu v obraze, mobilní aplikace

Keywords

application for mobile devices, iPhone, mobile device, iOS, Pokémon Go, text recognition, mobile application

Citace

PIHRT, Patrik. *Mobilní aplikace pro hráčskou podporu ve hře Pokemon Go*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Hrubý, Ph.D.

Mobilní aplikace pro hráčskou podporu ve hře Pokemon Go

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Patrik Pihrt
8. května 2019

Poděkování

Moc rád bych poděkoval svému vedoucímu Ing. Martinu Hrubému, Ph.D. za spoustu cenných rad, které mi pomohly k finálnímu zhotovení práce.

Obsah

Kapitola 1

Úvod

Mobilní hra Pokémon Go je po celém světě od roku 2016 stále velice oblíbenou. Spočívá v poměřování sil hráčů skrze nasbírané Pokémony. Síla Pokémona roste spolu s jeho úrovní a je ovlivněna jeho individuální hodnotou. V samotné hře individuální hodnota není příliš zjistitelná, a tak zkušení hráči k naleznutí nejlepších kousků používají kalkulátory dostupné na internetu či ve formě mobilních aplikací. Hra nabízí hráči několik způsobů, jak získat vzácnější Pokémony, například plněním úkolů nebo chozením vajec. Významná část hráčů hraje příležitostně a nestíhají registrovat často probíhající změny co se úkolů, vajec a Pokémonů z nich získaných týče.

Z vlastních zkušeností a postřehů z debat mezi hráči byly sestaveny prvky aplikace. Využito bylo prvků, které podle mého chybí v již existujících aplikacích na trhu. Způsob realizace prvků byl promyšlen, následně proveden, a tak vznikla aplikace obsahující prvky chtěné. Dle hráčské diskuze byla hlavním nedostatkem již existujících aplikací absence často měnících se informací a nedostatečné zaměření se na hráče začínající či mírně pokročilé.

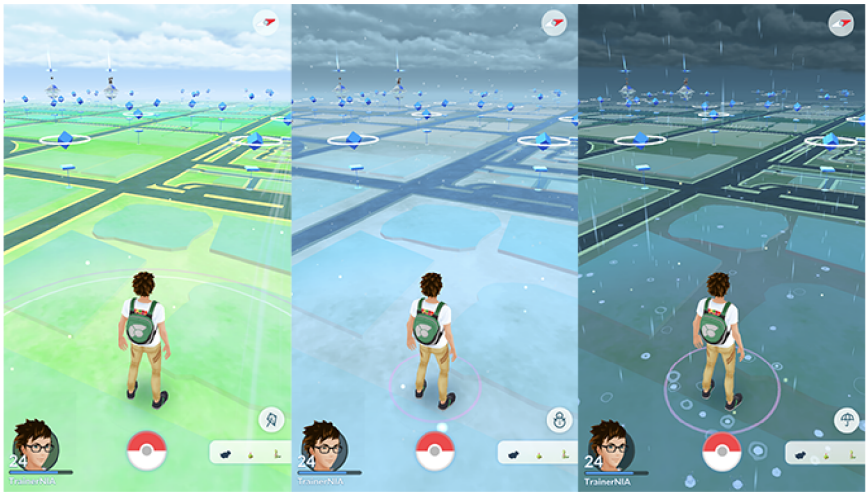
Cílem bakalářské práce bylo vytvořit aplikaci na mobilní zařízení od Applu, která splňuje požadavky komunity hráčů. Pro pochopení samotného smyslu vytvoření aplikace kapitola 2 uvádí základní informace o hře a o využitých prostředcích potřebných pro realizaci programu. Po seznámení se s tématikou a s využitými prvky při vytváření aplikace pro zařízení s operačním systémem iOS, se bakalářská práce zaměřuje na správné fungování aplikace a přívětivé používání. O návrhu se zmiňuje kapitola 3, je v ní popsán způsob získávání dat se snímku obrazovky a následný postup k prezentaci výsledku. Též pojednává o správné realizaci uživatelského rozhraní za pomoci typografických pravidel užitých pro přehlednost. Kapitola 4 se věnuje řešení návrhu v podobě kódu a použitým způsobům. Lze se z ní dozvědět například o způsobu práce s úložišti dat CoreData a UserDefaults nebo kódové realizaci získání dat se snímku. O testování aplikace se lze dočíst v kapitole 5, věnuje se poznatkům získaných z testování provedeného hráči, nebo z dob, kdy aplikace nebyla zatím kompletní. Finální kapitola 6 obsahuje zhodnocení práce a případné možné pozdější rozšíření aplikace.

Kapitola 2

Rozbor tématu

2.1 Pokémon Go

Pokémon Go je jedna z nejhranějších her na mobilní zařízení. Hlavní podmínkou pro hraní této hry je pohyb venku. Hra využívá GPS pro propojení herního prostředí s reálným světem, a umožňuje hráči reálnější požitek ze hry jako takové. Obrázek 2.1 názorně ukazuje snahu Pokémon Go přiblížit se realitě v podobě implementace počasí, která se řídí podle reality. Hra byla vydána 6. července 2016 americkou společností Niantic, Inc. a dodnes je stále na špičce co se týče počtu stažení [26].



Obrázek 2.1: Ukázka změn počasí ve hře podle počasí v reálném světě [17].

Niantic

Niantic je známá společnost vyvíjející hry na mobilní zařízení. Zaměřuje se na hry s rozšířenou realitou. Jako jediní na světě vytvořili platformu rozšířené reality. Za vytvořením platformy stojí touha propojit fyzický svět s digitálním. Vše začalo roku 2001, kdy skupina lidí vytvořila Keyhole, dnes znám spíše pod jménem Google Earth. V roce 2004 byl Keyhole koupen Googlem a přejmenován. Za zmínku stojí například i Google Maps a Street View, za kterými stojí stejná skupina tvůrců. Šest let na to vznikl projekt Niantic Labs pod záštitou Googlu, který měl za cíl propojit hraní her a jejich znalost map. Prvními aplikacemi byly

FieldTrip, jež byl založen na zajímavých lokacích a uživatele k nim naváděl, a Ingress, který poprvé přišel s konceptem spojení mapy reálného světa s hraním her. Ačkoliv se stahovaly a používaly docela v hojném počtu, to hlavní mělo teprve přijít. V roce 2015 se společnost stala nezávislou a od Googlu se odpojila. Rok na to přišel Niantic ve spolupráci s Nintendem a společností Pokémon s hrou Pokémon Go, která se přes noc stala naprostým fenoménem. Lidé byli nuceni chodit proto, aby zaznamenávali posun ve hře. K začátku roku 2018 byl počet nachozených kilometrů roven několika desítkám biliónům. Další připravované hry pro následující roky jsou Harry Potter ve spolupráci s Warner Bros, a také předělávka starší verze Ingressu s názvem Ingress Prime [19].

O hře

Na úplném začátku hry dostane hráč na výběr ze tří daných Pokémonů, stejně jako v prvních verzích her od Nintendo Co., Ltd. Poté si zvolí jeden tým z celkových tří, který v průběhu hraní reprezentuje. Každý tým má svého trenéra, který hráčům poskytuje bližší informace o chycených Pokémonech v průběhu hry. Nejvyšší úroveň hráče je 40. Celkový počet Pokémonů je 496 a jsou rozděleni do čtyř generací. Pokémoni dosahují taktéž maximální úrovně 40. Hráč se kromě postupného sbírání Pokémonů věnuje plnění různých úkolů. Dále má k dispozici vejce, ze kterých se pomocí chůze (2-10km) líhnou další úlovky. Ve hře je možné přidávat si ostatní hráče a mít je v seznamu přátel. S každým tímto kamarádem je možné provádět výměny Pokémonů či poměřovat své síly v podobě soubojů. Obě zmiňované akce můžou hráči provádět pouze v pevně dané blízkosti. Vyměňování Pokémonů může vyústit v obdržení šťastného tzv. lucky jedince. Pokémon poté na každé vycvičení potřebuje jen polovinu potřebného materiálu. Každá památka v okolí je ve hře vyobrazena jako stadion, který může být obsazen pouze jedním týmem. V praxi to znamená, že je zbarven buď do žluta, modra, nebo do červena. Boje o stadion probíhají velice často, týmy si tímto způsobem měří síly svých Pokémonů. V průběhu dne se koná na stadionech i jiný typ souboje (raid) – stadion je na hodinu obsazen silným (většinou zároveň raritním) Pokémonem. Pro takový boj je potřeba až deseti hráčů libovolného týmu, za vítězství je možnost poraženého Pokémona chytit. Hra oplývá i častými komunitními událostmi a akcemi. Cílem většiny hráčů je zaplnění tzv. Pokédexu – seznamu všech dostupných Pokémonů. Sekundárním cílem je získávání nejlepších jedinců, a to skrze sledování individuálních hodnot.

Pokémoni a jejich hodnoty

Každý Pokémon má své tři pevně dané hodnoty – útok, obranu a výdrž. Kombinací těchto hodnot, úrovně Pokémona (1 – 40), a jeho individuální hodnoty se získá finální síla Pokémona. Pokémon Go hráči zprostředkovává jen výslednou sílu, ale faktory k výpočtu nechává hráči zatajeny. Nejdůležitější z faktorů pro hráče je individuální hodnota (IV). Avšak tu není vůbec lehké zjistit [22, 23].

Individuální hodnota Pokémona

Individuální hodnota představuje bonus přidaný k základním hodnotám Pokémona (útok, obrana a výdrž). Rozmezí bonusu je 0–15 bodů. Např. Pokémon zvaný Eevee má základní hodnoty útok 104, obranu 114, výdrž 146 a bonus ke každé hodnotě je 0, čili jeho individuální hodnota se rovná 0,0,0 (0% – nejhorší možná Eevee). Naopak nejlepší možná (hráči žádaná) 100% Eevee má hodnoty útok 119(104+15), obranu 129(114+15) a výdrž 161(146+15). Ve výsledku je finální síla Pokémona s bonusovými hodnotami 15,15,15 průměrně o 10% lepší

jak s hodnotami 0,0,0 [22, 23].

Vliv individuální hodnoty spolu s úrovní Pokémona na finální sílu:

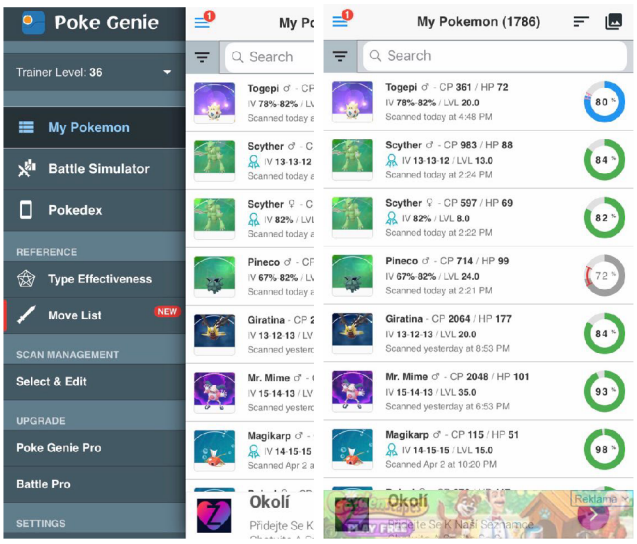
- Eevee s úrovní 20 a individuální hodnotou 0% má finální sílu 478
- Eevee s úrovní 20 a individuální hodnotou 100% má finální sílu 612

2.2 Existující aplikace

Většina zkušených hráčů používá minimálně jednu z dále uvedených aplikací. Jak již bylo zmíněno výše, Pokémon Go hráčům nesděluje chtěné informace, a proto jsou podpůrné aplikace skoro nutností. Při výběru aplikací, které bych zde rád zmínil, jsem vybral aplikace fungující bez vykazování chyb co se výpočtů a celkové funkcionality týče. Inspiroval jsem se první zmiňovanou aplikací z důvodu jednoduchosti uživatelského rozhraní a kvality zpracování dat ze snímků obrazovky.

PokeGenie

PokeGenie podporuje hráče na obou platformách. Primárně se zabývá získáváním dat ze snímků obrazovky a jejich prezentací. Další funkcí jsou např. simulace soubojů proti různým Pokémonům nebo přehled všech různých kombinací útoků. Z této aplikace jsem si vzal příklad a využil jsem postranního menu, které můžeme vidět v levé části obrázku 2.2, sloužící pro snadné vrácení se zpět na předešlou obrazovku. Značnou inspirací mi byl i hlavní seznam s naskenovanými Pokémony (pravá část obrázku 2.2) obsahující všechny důležité informace pro hráče.



Obrázek 2.2: Ukázka vzhledu PokeGenie.

PokeRater

PokeRater je méně známý mezi hráči, ale rozhodně stojí za zmínku. Hlavní funkcí stejně jako u PokeGenie je prezentace dat uživateli z naskenovaných snímků. PokeGenie má z mého

pohledu skenování vyřešené o dost lépe, a tak bych se raději zaměřil na vedlejší funkce PokeRateru. První z funkcí, kterou bych rád vyzdvihl, je přídávání nových kamarádů z celého světa přímo z aplikace. Další zajímavostí je klávesnice se speciálními znaky. Hráči používají specifické znaky k označení vyjímečných vlastností Pokémonů.

CalcyIV

Spolu s PokeGenie tvoří CalcyIV dvojici nejpoužívanějších doplňujících aplikací pro uživatele Androidu. Je určena pouze pro platformu Android. Kombinuje stejné prvky výše zmíněných, avšak vyniká možností být spuštěná zaráz s Pokémon Go, a tak působí více jako součást samotné hry jak můžeme vidět na obrázku 2.3.

Obrázek 2.3: Ukázka vzhledu CalcyIV.

2.3 Programování pro iOS

K vytvoření aplikace pro iOS je potřeba nezbytného přístupu k zařízení s operačním systémem OS X. Další nutnost představuje obstarání vývojového prostředí přímo od Applu jménem Xcode. Po splnění podmínek je na řadě výběr programovacího jazyka, ve kterém daná aplikace vznikne. Pro účely této bakalářské práce jsem si vybral programovací jazyk Swift z důvodu srozumitelnosti a dojmu, že se dá rychle a lehce naučit. Všechny aplikace pro Apple zařízení jsou založeny na architektuře MVC[1]. Architekturu je doporučeno dodržovat z důvodu lehčí znovupoužitelnosti a přehlednosti.

Programovací jazyky

Hlavní programovací jazyky k naprogramování iOS aplikace jsou Objective – C a Swift. První zmiňovaný je předchůdcem modernějšího Swiftu. Objective – C je objektově orientovaný jazyk vycházející z jazyka C a Smalltalku. Poprvé byl využit v počítačích NeXT s operačním systémem NeXTSTEP. Nová verze Objective – C s názvem Swift byla představena v roce 2014 na světové vývojářské konferenci společnosti Apple. Swiftu připadla přezdívka "Objective – C bez C"[18] , čímž se poukazuje na nepovolený přístup k ukazatelům

[1] Model – View – Controller

a ostatním nebezpečným prvkům. Dalším velkým rozdílem mezi Objective – C a Swiftem je absence Smalltalk způsobu volání metod. Tento způsob byl nahrazen tečkovými notacemi a jmenným prostorem, který je podobný dalším běžným objektově orientovaným programovacím jazykům. Specialitou Swiftu je možnost nazvat proměnnou volitelnou (optional). Volitelná proměnná může i nemusí nabývat hodnoty. Ve výsledku touto možností proměnných můžeme simulovat funkci ukazatelů v Objective – C [27].

iOS

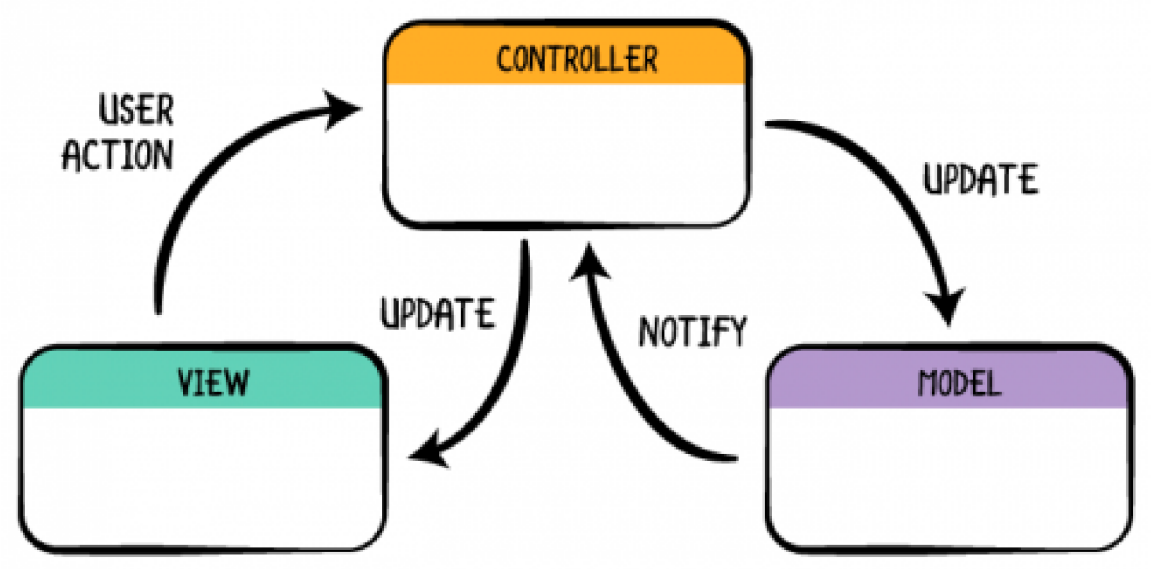
iOS, dříve iPhone OS, je mobilní operační systém vytvořen společností Apple Inc. Je druhým nejpopulárnějším systémem na světě po Androidu. První oficiálně představená verze byla roku 2007 pro mobilní zařízení iPhone a následně byl tentýž systém použit i v iPodu Touch v září toho roku, a o tři roky později byl využit i v iPadech (leden 2010). Od MacOS X a tvOS se iOS odlišuje možností práce na dotykové obrazovce (stejně jako tomu je u watchOS). Aktuální iOS verze je 12.2 a byla vydána koncem března roku 2019 [25].

Xcode

Xcode je vývojářské prostředí volně přístupné na App Store. Obsahuje profesionální vývojářské nástroje pro vytváření aplikací na platformy iOS, MacOS X, tvOS a watchOS. Aktuální nejnovější dostupná verze na App Store je 10.2, tato verze umožňuje vyvíjet aplikace pro nejnovější verze výše zmiňovaných platforem (iOS 12.2, watchOS 5.2, macOS 10.14.4, a tvOS 12.2.). Z důvodu využívání iOS verze 12.2 beta na testovacím zařízení bylo nunností pořídit verzi 10.2 beta 2 z internetu pro podporu této iOS verze. Xcode dovoluje vývojáři vyvíjet aplikace v jazyce C, Objective – C, C++, Objective – C++, Swift, Java, Python, Ruby, AppleScript a Rez [16]. Při vytváření nového projektu se automaticky vygeneruje určité množství (minimálně čtyř) souborů. Těmito soubory jsou AppDelegate, ViewController, Main.storyboard a LaunchScreen.storyboard. AppDelegate je přezdíván srdcem aplikace a zkráceně řečeno řídí celou aplikaci. Dále bych se rád zmínil o Main.storyboardu a LaunchScreen.storyboardu. Storyboardy umožňují jednoduchý návrh uživatelského rozhraní aplikace pomocí WYSIWYG editoru. LaunchScreen se využívá při spouštění aplikace a v Main souboru se dále tvoří veškeré uživatelské rozhraní aplikace. Xcode nabízí i možnost testování vlastních projektů. Testovat je možné na simulátoru jakéhokoliv zařízení a verze od firmy Apple, či na reálném přístroji připojeném k počítači. Po spuštění aplikace přibývá možnost sledování chování zařízení za běhu. Jedny ze sledovatelných faktorů jsou např. jak moc aplikace zatěžuje CPU nebo počet zapsaných a čtených dat z paměti zařízení spuštěné aplikace. Další z přívětivých funkcí je nápověda samotného Xcodu během psaní kódu.

Model – View – Controller

Návrhový vzor Model – View – Controller (jehož vizuální podoba je znázorněna na obrázku 2.4) přiřazuje objektům v aplikaci jednu ze tří rolí: Model, View (pohled) a Controller (řadič). Role by měly být navzájem co nejvíce nezávislé, to znamená, že pokud se upraví jedna, změna ovlivní zbylé co nejméně. Vzor nedefinuje pouze role, které objekty zastupují v aplikaci, ale také vzájemnou komunikaci těchto objektů. Komunikace probíhá přes abstraktní hranice, kterými jsou tyto typy objektů odděleny. Model – View – Controller je základem správné iOS aplikace. Snadno se rozšiřuje a lépe se v ní opravují chyby. Části kódu je možné bez žádných větších problémů znovu použít v jiné aplikaci [12].



Obrázek 2.4: Princip Model – View – Controlleru [21]

Model

Model zapouzdřuje data specifická pro aplikaci. Definuje logiku a výpočty, které data zpracovávají. Jako příklad Modelu může být postava ve hře. Nemělo by se stát, že Model bude přímo komunikovat s View [12].

View

View je vizuální část aplikace viděna uživatelem. Umí se vykreslit a reagovat na akce. Princip View je vizualizace dat z Modelu a umožnění získaná data modifikovat. View a Model jsou v architektuře od sebe odděleny a jejich komunikace vede přes Controller. View je při správné práci se vzorem lehce znova použitelný v jiných aplikacích [12].

Controller

Controller můžeme považovat za tzv. prostředníka. Spojuje View a Model a řídí komunikaci mezi nimi. Jakmile se něco ve View nebo Modelu změní, je pomocí Controlleru ihned informována druhá strana [12].

2.4 Knihovna UIKit

Je významný framework vytvořen pro vývoj aplikací na systémech iOS a watchOS. Poskytuje konstrukce a elementy uživatelského rozhraní pro vývoj kvalitně vypadajících aplikací. Hlavními zástupci elementů je **UIViewController**. Aplikaci často tvoří vícero ViewControllerů, které jsou spojeny pomocí přechodů. Obsahem **UIViewControllerů** jsou UIView, UIButton, UILabel, UISwitch, atd. Vývojář používá tyto objekty takové jaké jsou s případnými minimálními obměnami. Aplikace využívající UIKit jsou založeny na architektuře MVC, jejíž vysvětlení je v podkapitole 2.3. UIKit se angažuje ve View a Controller části aplikace. Jedněmi z prvků stojících za zmínku jsou **UIView** (třída uzpůsobena k prezentaci

obsahu aplikace uživateli. Patři pod View.) a **UIApplication** (objekt realizující hlavní cyklus událostí aplikace. Spravuje aplikaci po dobu jejího životního cyklu) [11].

Životní cyklus aplikace

Životní cyklus aplikce se dá rozdělit do pěti stavů [20]:

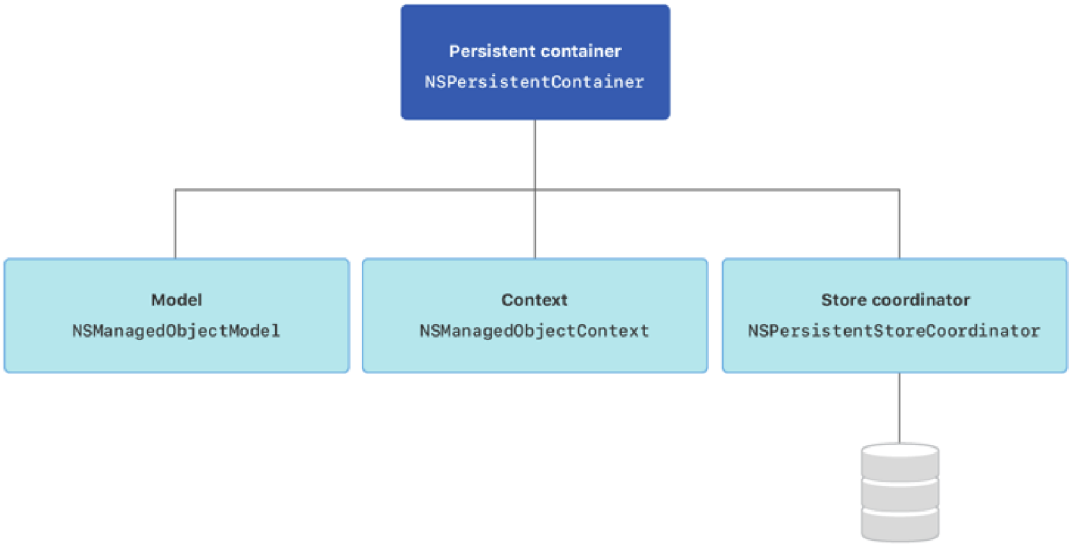
1. **Neběžící** – aplikace nebyla spuštěna nebo byla systémově přerušena.
2. **Neaktivní** – aplikace je již viditelná na obrazovce, ale není schopna přijímat události od uživatele.
3. **Aktivní** – stav, kdy je aplikace viditelná a plně schopna pracovat s uživatelem. Z aktivního stavu se často přesouvá do fáze neaktivní a nazpět z důvodu zpracovávání požadavků uživatele.
4. **Na pozadí** – aplikace přesunuta pryč z obrazovky (viditelného místa na zařízení), ale je stále aktivní
5. **Pozastavena** – aplikace je pozastavena a není prováděn žádný kód, ale je stále v paměti zařízení a je možno ji znovu obnovit.

2.5 Ukládání dat – CoreData a UserDefaults

Způsobů ukládání dat je u iOS celkem mnoho. Hlavními jsou CoreData, UserDefaults, CloudKit a SQLite. Při vytváření své aplikace jsem použil CoreData a UserDefaults [15].

CoreData

CoreData umožňuje aplikacím uchovávat data různých forem a načítat je. Data uchovává v SQLite databázi, avšak CoreData jako takový databází není. Je objektovým grafem umožňující vytvářet, ukládat a načítat objekty. Ty mají své atributy a vztahy k ostatním objektům grafu. Díky jeho jednoduchosti je umožněno ukládat data od jednoduchých po komplexní. Xcode nabízí vygenerování souboru pro práci s CoreData s koncovkou .xcdatamodeld. Zde můžeme vytvářet entity, jejich atributy a vztahy mezi ostatními entitami grafu. Každý atribut objektu může nabývat různého typu (integer16, decimal, float, string...). Vztahy mezi objekty mohou být navázány buďto jen s jedním či nekonečně mnoho dalšími objekty. Celý soubor je vizuální podoba **NSManagedObjectModel**, který je jedním ze tří objektů **CoreData stacku** 2.5 [14, 13].



Obrázek 2.5: CoreData stack

CoreData stack je nezbytnou součástí aplikací využívajících CoreData. Skládá se ze tří hlavních objektů [10]:

- **NSManagedObjectModel** – zastupuje databázové schéma popisující entity aplikace,
- **NSManagedObjectContext** – nabízí možnost vytváření, ukládání nebo načítání NSManagedObject z trvalého úložiště dat,
- **NSPersistentStoreCoordinator** – prostředník při akcích přistupujících k trvalému úložisti dat.

UserDefaults

UserDefaults je skvělé řešení pro ukládání malého objemu dat. Data se ukládají do interního souboru. Pro větší objemy dat je UserDefaults nevhodný z důvodu nutnosti načtení celého souboru se všemi daty pro získání byť i malého objemu dat. V aplikaci je využit pro ukládání osobního nastavení uživatele. Zakládá se na principu klíč hodnota. Hodnoty mohou nabývat základních datových typů (float, double, intiger atd.), nebo objektů (musí být typu list) [15].

2.6 Generická třída NSFetchedResultsController

NSFetchedResultsController se využívá na spravování výsledků načtených dat z CoreData. Je často využíván v kombinaci s buněčným seznamem, který data reprezentuje uživateli. Pro vytvoření Controlleru je zapotřebí:

1. **NSFetchRequest** – vytváří pole výsledků z CoreData splňující požadavky.
2. **NSManagedObjectContext** – kopie obsahu databáze.
3. **SortDescriptor** – určení, jakým způsobem řadit prvky seznamu.

Nabízí se i možnost výsledky filtrovat pomocí **NSPredicate**, ale není to nutností. NSFetchedResultsController nabízí ukládání dat do paměti cache. Data jsou v ní uložena i při znovuzapnutí aplikace. Cache je využívána při snaze vyhnout se opakovaní akce nastavování pole s výsledky načtení dat z databáze a následném seřazení. V případě, že jsou provedeny změny v databázi, které cache neobsahuje, je cache aktualizována pro budoucí používání. K NSFetchedResultsController lze implementovat delegát, který informuje o změnách na úrovni modelu a o nutnosti NSFetchedResultsControlleru upravit svá data [9].

2.7 Firebase ML – Rozpoznávání textu v obraze

Firebase je platforma vytvořena pro podporu vývoje aplikací na mobilní zařízení. Byla vytvořena roku 2011 firmou Firebase, Inc. Tři roky na to byla odkoupena firmou Google. Google si dále přivlastnil firmu Divshot, která se zabývá webovým hostingem a propojil ji s Firebase. Aktuálně poskytuje Firebase spoustu užitečných funkcí pro programátora. Za zmínku stojí realtime databáze, která nepoužívá HTTP, ale je propojena skrze WebSocket. Platforma poskytuje ukládání souborů do Cloud úložistě od Googlu. Dále také umožňuje zmiňovaný hosting, na kterém pracuje společně s firmou Divshot. Aplikace využívá vývojářskou sadu od Firebase s názvem Firebase ML (machine learning) [24, 2].

Firebase ML

Firebase ML poskytuje vývojáři řadu API[2], které může volně využít ve své aplikaci. Mezi API patří např. rozpoznávání textu v obraze, detekování obličejů nebo také skenování čárových kódů. Vývojář má možnost vybrat si ze dvou typů spuštění těchto API. Pokud vývojář upřednostňuje rychlost před kvalitou výsledku je správné použít běh na zařízení. Popisovaná varianta je úplně zdarma a nepotřebuje připojení k internetu. Druhou možností je běh v Cloudu. Zde je potřeba připojení k internetu a při bezplatném používání je počet použití omezen. Cloud nabízí kvalitnější výsledky, což může být někdy důležitější jak rychlost zpracování. V případě, že API nevyhovují podmínkám vývojáře, je možné si API přizpůsobit. Stačí nahrát svůj model do Firebase, který se postará o vše ostatní [4, 3].

Rozpoznávání textu v obraze

Rozpoznávání textu v obraze je jednou z vlastností Firebase ML. Běh na zařízení poskytuje dostačující výsledky rozpoznávání latinských znaků. Cloud varianta má omezení na 1000 použití měsíčně (co se bezplatné verze týče), ale poskytuje mnohem lepší výsledky a rozpoznává nejen latinské znaky. Na obrázku 2.6 jsou ukázány varianty párování rozpoznaného textu do bloků. Každý blok je předán s hodnotou x, y levého horního rohu. Souřadnice jsou užitečné při potřebě určení pozice v obraze vůči jiným blokům [5].

[2] Application Programing Interface

Obrázek 2.6: Ukázka detekce textu v obraze.[6]

2.8 MapKit

MapKit je framework[3] umožňující používat mapu světa v aplikaci. Mapu je možno centrovat, či obohacovat anotacemi zajímavých míst. Anotace je možno vkládat jak ručně, tak automaticky. MapKit umožňuje na mapě zobrazovat i polohu, na které se aktuálně osoba nachází, a průběžně ji aktualizovat [7].

MKMapView

MKMapView je rozhraní obsahující mapu, jejíž hlavními typy jsou satelitní, hybridní a standardní. Ukázku hybridního typu mapy můžeme vidět na obrázku 2.7. Obsah MKMapView se dá otáčet, přibližovat či oddalovat pomocí gest. Obrázek 2.7 dále znázorňuje anotaci v podobě špendlíku, aktuální pozici uživatele a centrování mapy k jeho poloze. Po povolení polohových služeb aplikaci je možné zpřístupnit kompas a aktuální polohu uživatele [8].

Obrázek 2.7: Ukázka MKMapView

[3] Aplikační rámec – softwarová struktura pro podporu programování

2.9 CocoaPods

CocoaPods spravují závislost knihoven na sobě v Xcode projektech. Soubor, ve kterém se nachází specifikace těchto závislostí, se nazývá Podfile. Vyřešením závislostí mezi knihovnami pomocí CocoaPods vzniká Xcode workspace. Vzniklý soubor obsahuje jak samotný projekt, tak i knihovny přidané pomocí CocoaPods. Při snaze použít CocoaPods je potřeba jejich instalace [1].

Následující příkaz automaticky stáhne vše potřebné :

```
$ sudo gem install cocoapods
```

Další fází při snaze připojit CocoaPods k projektu je vytvoření souboru Podfile. Jako příklad, jak by soubor mohl vypadat, jsem vybral Podfile své aplikace.

```
target 'PoGoHelper' do
use_frameworks!

pod 'Firebase/Core'
pod 'Firebase/MLVision'
pod 'Firebase/MLVisionTextModel'

end
```

Target je cílový projekt, ke kterému se připojí následující knihovny vypsané níže.

Jakmile je Podfile kompletní, je třeba využít příkazu:

```
$ pod install
```

Příkaz stáhne a nainstaluje všechny nové pody (i opakovaně), které doposud soubor Podfile.lock neobsahuje. Podfile.lock se generuje při prvním zadání příkazu výše a uchovává informace o verzi každého nainstalovaného podu.

Dalšími nabízenými příkazy jsou:

```
$ pod outdated
a
$ pod update
```

Outdated příkaz porovnává aktuální verze podů s verzemi uloženými v Podfile.lock. Vypíše ty, které jsou zastaralé. Update nahraje všechny nové verze podů, dá se spustit i s parametrem specifikující daný pod.

Kapitola 3

Návrh aplikace

V následující kapitole se věnuji aplikaci jako takové. Konkrétně se zde budu zabývat návrhem, od zadání přes navrhnuté řešení až po uživatelské rozhraní výsledné aplikace. Zahrnul jsem zde i strukturu objektového grafu CoreData z důvodu následné lepší orientace v části s implementací.

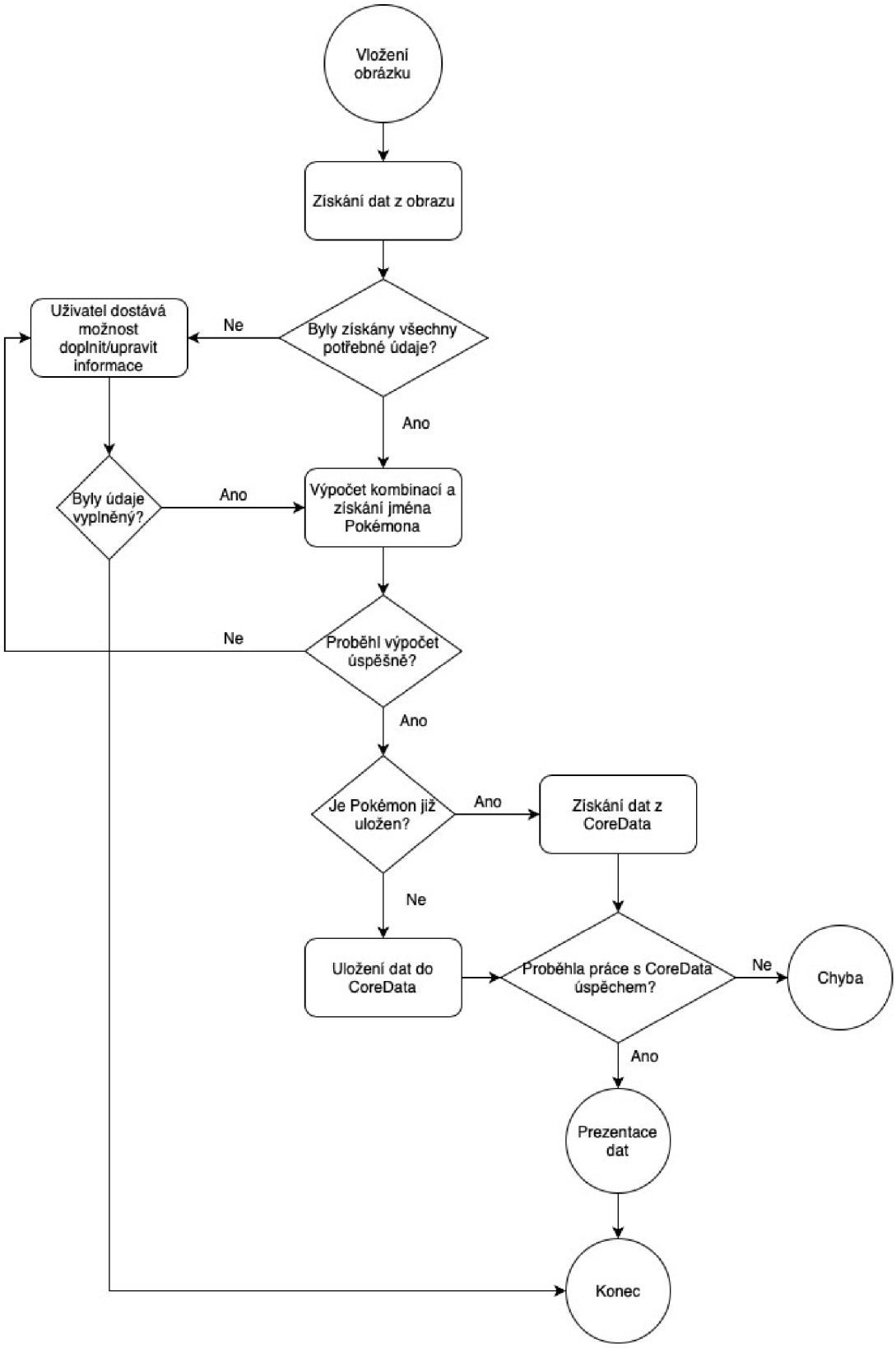
3.1 Rozbor zadání

Ze zadání plyne, že hlavním cílem bylo vytvořit aplikaci, která zpříjemní a usnadní hraní hráčům Pokémon Go. Základním kamenem aplikace je možnost skenování obrázků ze hry a následné vyhodnocení cenných dat pro hráče. Dalšími vlastnostmi aplikace jsou:

- detailní informace o všech dostupných Pokémonech ve hře,
- odpovědi na často měnící se obsah hry (aby hráč byl vždy o krok napřed a byl stále informován),
- užitečné rady a tipy.

3.2 Zpracování dat

Navržený způsob zpracování dat byl inspirován již existujícími aplikacemi. Uživatel nahraje snímek Pokémona, aplikace snímek zpracuje, a ze získaných dat si následně vytáhne potřebné informace k následným výpočtům. Výsledky dále prezentuje uživateli a ukládá je do CoreData. Podobnější postup zpracování dat je znázorněn pomocí diagramu na Obrázku 3.1. V něm lze vidět i způsob řešení nekompletního sběru dat z obrazu. V případě, že Pokémon kterého jsme již naskenovali, je obsažen v CoreData, jsou z něj vzata potřebná data a prezentuje naskenovaný obrázek jako již naskenovaný. Při možné chybě ukládání či vytažení dat z CoreData je uživatel obeznámen a požádán, aby akci opakoval.



Obrázek 3.1: Diagram zpracování dat ze snímku obrazovky

3.3 Základní data o Pokémonech

Pro uložení základních dat o všech Pokémonech bylo využito pole klíč hodnota, kde klíč znázorňuje jméno Pokémona a hodnota je struktura obsahující:

- *Číslo* – pořadí v Pokédexu,
- *Jméno* – název,
- *Atk* – základní útok,
- *Def* – základní obrana,
- *Stam* – základní výdrž,
- *Evolve* – vyšší vývin,
- *PodEvolve* – nižší vývin,
- *Lv15CpRange* – rozmezí finální síly na úrovňi 15,
- *Lv20CpRange* – rozmezí finální síly na úrovni 20,
- *Lv25CpRange* – rozmezí finální síly na úrovni 25.

Pole je využíváno ve většině částí aplikace, a proto byl kladen důraz na bezchybnost. Pole obsahuje přes 500 párů klíč hodnota a využívá se například u výpočtů individuální hodnoty jako dodavatel základních hodnot Pokémona.

3.4 Třídy s informacemi o Pokémonovi

Informace předané buďto skrze naskenovaný seznam Pokémonů, nebo pomocí vloženého obrázku, jsou uloženy do třídy PokemonScreenInfo. Jejím smyslem je možnost nad daty provádět menší úpravy a následně finální podobu úprav uložit do CoreData. Hraje velkou roli i při předávání informací mezi hlavním ViewControllerem a pop–up ViewControllery. Obsahuje :

- *CP* – finální hodnota,
- *HP* – životy,
- *Name* – jméno,
- *CandyType* – typ dobrot daného Pokémona,
- *Stardust* – prach potřebný pro vytrénování na vyšší úroveň,
- *CombinationList* – list třídy IVcombination, který obsahuje vyhovující kombinace bez jakéhokoliv upřesňování,
- *Edit* – třída Eedit,
- *Koordinaty* – třída Koordinatyy,
- *prumIV* – průměrná individuální hodnota rozmezí.

Třída obsahuje další tři třídy. CombinationList obsahující list tříd IVcombination je prázdný jen tehdy, když se iniciuje. Po naplnění třídy PokemonScreenInfo informacemi je nutností, aby CombinationList obsahoval minimálně jednu kombinaci. Ta se skládá z pěti atributů:

- *level* – úroveň, pro kterou platí následující hodnoty
- *atk* – individuální hodnota útoku,
- *def* – individuální hodnota obrany,
- *stam* – individuální hodnota výdrže,
- *afterEdit* – zda kombinace prošla kritérii upřesňování výsledků.

V případě, že CombinationList bude po naplnění prázdný, nastává chyba řešená různými způsoby. Například vyvoláním pop–up ViewControlleru s možností upravit získaná data z naskenovaného snímku. Vzorce pro výpočet individuálních hodnot Pokémona jsou k nalezení v sekci 3.5. Další třídou obsaženou v PokemonScreenInfo je Eedit. Obsahuje editační hodnoty pro upřesnění výsledku. Podrobnější obsah třídy je zde:

- *lucky* – zda je Pokémon šťastný,
- *level* – ručně zadaná úroveň,
- *ivrange* – určení jednoho ze čtyř rozmezí individuální hodnoty,
- *attack* – 1 pokud je útok dominantní hodnota, jinak 0,
- *deffence* – 1 pokud je obrana dominantní hodnota, jinak 0,
- *stamina* – 1 pokud je výdrž dominantní hodnota, jinak 0,
- *ivdominant* – určení jednoho ze čtyř rozmezí dominantní hodnoty.

PokemonScreenInfo obsahuje ještě jednu třídu označenou jako Koordinatyy. Jak už z názvu vyplývá, jedná se o třídu se zeměpisnou šířkou a délkou. Je využívána pro ukládání bodu, kde byl Pokémon nalezen a chycen.

3.5 Vzorce pro výpočet platných kombinací individuálních hodnot

Pro výpočet individuálních hodnot bylo potřeba využít dvou vzorců. Nejdůležitějším vzorcem je samotný výpočet finální síly (3.1).

$$CP = \frac{(ZkU + BonusU) * \sqrt{(ZkO + BonusO)} * \sqrt{(ZkV + BonusV)} * AktU^2}{10} \tag{3.1}$$

Význam zkratek je následující:

- *CP* – finální síla,
- *ZkU* – základní útok,
- *BonusU* – individuální hodnota útoku,

- ZkO – základní obrana,
- $BonusO$ – individuální hodnota obrany,
- ZkV – základní výdrž,
- $BonusV$ – individuální hodnota výdrže,
- $AktU$ – Stanovené číslo platné pro aktuální půl úroveň.

Ze vzorce vyplývá, že pro výpočet finální hodnoty je potřeba znát základní hodnoty, individuální hodnoty a stanovené číslo pro aktuální půl úroveň. Pro účely aplikace je výše zmiňovaný vzorec využit na výpočet individuálních hodnot Pokémona.

Dalším využitým vzorcem je vzorec na výpočet životů Pokémona.

$$HP = (ZkV + BonusV) * AktU \tag{3.2}$$

Význam zkratek je následující:

- HP – životy,
- ZkV – základní výdrž,
- $BonusV$ – individuální hodnota výdrže,
- $AktU$ – stanovené číslo platné pro aktuální. půl úroveň

Životy závisí pouze na celkové výdrži Pokémona, jiné hodnoty je neovlivňují. Pro získání platné kombinace je nutné použít oba vzorce.

Výsledky rovnic jsou vždy zarovnány na číslo 10. U Pokémonů s nízkou úrovní zarovnávání komplikuje výpočet individuální hodnoty, a je takřka nemožné vytvořit validní výsledek. Oproti všem možným kombinacím se počet platných kombinací snižuje jen o malé množství. Rozmezí individuální hodnoty tedy bude např. 0-90%, což není zrovna výsledek, který uživatel očekává.

Vzorci výše získáme seznam kombinací tří hodnot (útok, obrana a výdrž), které následně využijeme ve vzorci 3.3. Výsledná hodnota je v procentech a značí jeho kvalitu od 0 do 100%.

$$IV = ((Utk + Obr + Vyd)/maxSoucet) * 100 \tag{3.3}$$

Proměnné ve vzorci mají následující význam:

- Utk – hodnota útoku dané kombinace,
- Obr – hodnota obrany dané kombinace,
- Vyd – hodnota výdrže dané kombinace,
- $maxSoucet$ – maximální možný součet těchto hodnot (45).

3.6 Upřesňování výsledků

Pokémon může nabývat od své úrovně 1 až 40 celkem **323 584 kombinací**. Výpočty zredukují veliké množství, avšak zbylá škála může být stále velice obsáhlou. Pro nutnou potřebu zredukování kombinací byly navrhnuty dodatečné funkce aplikace pro upřesnění konkrétních výsledků.

Ohodnocení Pokémona

Pokémon Go nabízí možnost ohodnotit si Pokémona od svého trenéra (hlavní reprezentant zvoleného týmu). Trenér hráči sděluje 3-5 vět (jsou pro týmy odlišné, avšak jejich význam je identický), které jsou pouze orientační. V první větě je hráči sděleno rozmezí, ve kterém se daný Pokémon pohybuje co se individuální hodnoty týče. Ve větě není přesně rozmezí řečeno, ale je schováno za větu hodnotící sílu Pokémona. Věty určující rozmezí pro např. žlutý tým jsou:

- Looks like it can really battle with the best of them! – **82,2-100%**,
- Is really strong! – **66,7-80%**,
- Is pretty decent! – **51,1-64,4%**,
- Has room for improvement as far as battling goes! – **0-48,9%**.

Druhou až potenciálně čtvrtou větou trenér sděluje dominantní hodnotu Pokémona. Po dokončení sdělení o dominantních hodnotách přichází na řadu poslední věta. V ní se hráč dozvídá rozmezí dominantních hodnot zmíněných ve větách předtím. Ty jsou také prezentovány v podobě vět a není na první pohled vidět, co se snaží říct. Zde je ukázka těchto vět pro žlutý tým:

- Its stats are the best I've ever seen! No doubt about it! – **15**,
- Its stats are really strong! Impressive – **13-14**,
- It's definitely got some good stats. Definitely! – **8-12**,
- Its stats are all right, but kinda basic, as far s I can see. – **0-7**.

Hráč sdělené věty může použít v aplikaci, čímž zredukuje kombinace jen na ty, co splňují podmínky daných vět.

Určení úrovně

Určením úrovně má možnost uživatel ručně zadat aplikaci, jakou úroveň z nabízených Pokémon nabývá. Funkce pomůže hlavně hráčům, kteří už jsou ve hře zběhlí a vědí, jaké přesně úrovně je Pokémon. Funkci doporučuji používat jen v případě, že si je hráč jist danou úrovní. Při zvolení špatné úrovně se stanou výsledky nesprávné.

Vyfiltrování horších kombinací

V Pokémon Go je možné získat Pokémony několika různými způsoby:

- **Bežný** – Pokémon na obrazovce a jeho individuální hodnota je náhodná (0-100%),

- **Z úkolu** – Pokémon má úroveň 15 a jeho individuální hodnota je nejhůře 67%,
- **Z vajíčka** – Pokémon má úroveň 20 a jeho individuální hodnota je také nejhůř 67%,
- **Z raidu** – Pokémon má úroveň 20 či 25, a jeho individuální hodnota je také nejhůř 67%,
- **Výměnou s jinými hráči** – Při této akci je možnost získat šťastného Pokémona, který nabývá taktéž nejhůře 67% individuální hodnoty. V případě, že se výměnou Pokémoni nestanou šťastnými, pak není individuální hodnota nijak omezená.

Pro Pokémony z vajec, úkolů nebo šťastných výměn, jsou automaticky horší individuální hodnoty jak zmiňovaných 67% nereálné a je možné je bezpodmínečně zanedbat.

3.7 Kalkulace Pokémona

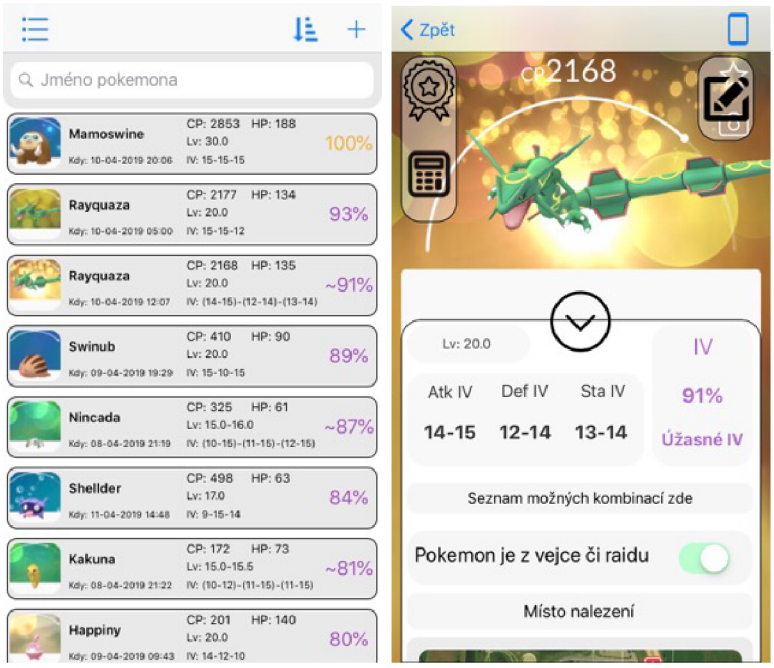
Po zpracování dat a případném upřesnění umožňuje aplikace předávat informace o síle Pokémona vlivem zvyšování jeho úrovně a zároveň ukazuje, kolik prostředků bude potřeba použít. Hráč získá kompletní přehled o nastávajících změnách síly Pokémona. Jediné, co hráč musí udělat pro získání informací, je pohybovat posouvačem, který simuluje zvyšování úrovní.

3.8 Uživatelské rozhraní

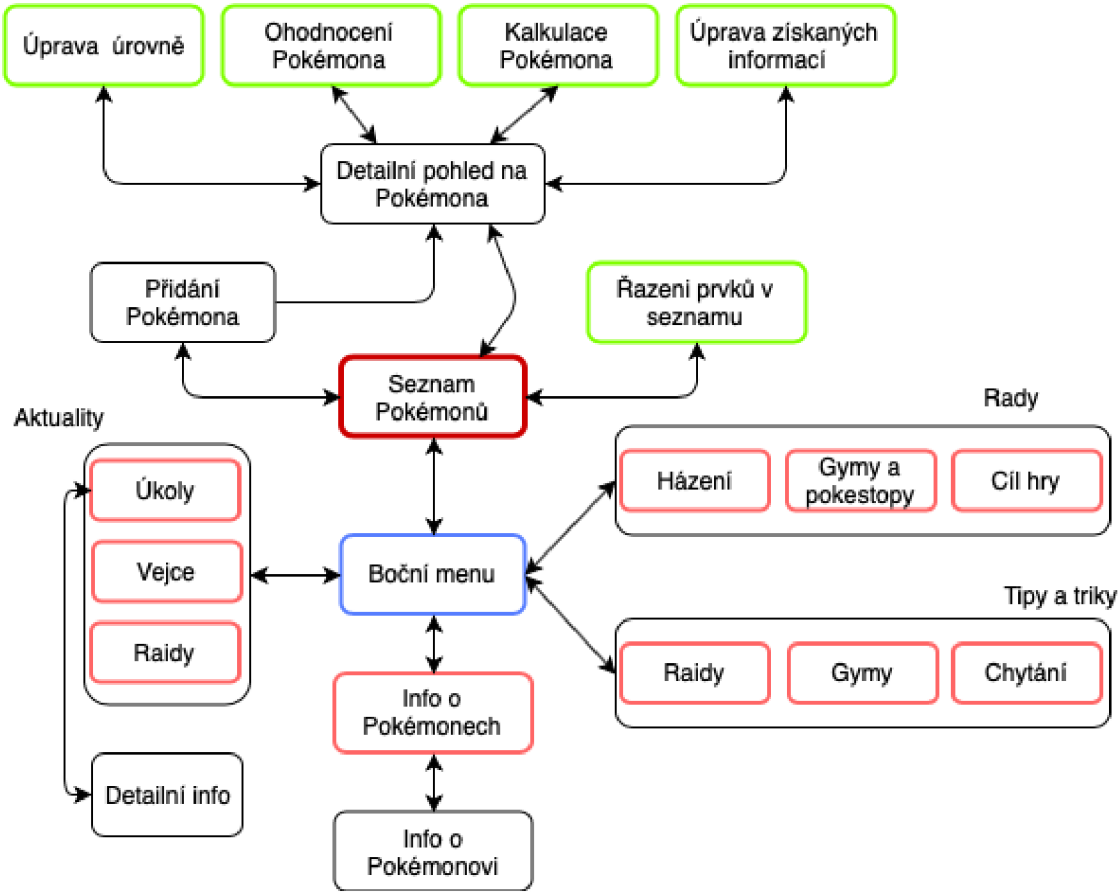
Uživatelské rozhraní je jeden z hlavních faktorů, podle kterých je aplikace hodnocena. Mělo by být přehledné a tlačítka s akcemi by měly mít vždy stejný vzhled. Uživatel na základě znalostí z předešlých částí aplikace již ví, co daná akce provede, i když se v dané části ocitá poprvé. Další, co by aplikace měla obsahovat, je rozlišování důležitosti informací. Na základě stanovených priorit se určuje velikost prvků v aplikaci. Nezohledněním významnosti zbytečně zaměstnáváme uživatele, což je nežádané. Na následujícím obrázku 3.2 bych rád ukázal využití těchto pravidel v praxi. V levé části obrázku je poukázáno na uspořádání informací do sloupců v jednotlivých buňkách. Důraz byl kladen i na zviditelnění nejdůležitejších informací jako například jméno nebo individuální hodnotu v procentech. Tlačítka v horní části byly vybrány na základě zkušeností z ostatních aplikací, kde plus znamená přidat, levé tlačítko připomíná menu a šipka s řádky představuje způsob řazení buněk v seznamu. Pravá část nabízí větší množství informací o Pokémonovi. Každá z nich je vizuálně oddělena pozadím. Využitý způsob uživateli zpřehlední pohled na aplikaci a ulehčí orientaci.

Mapa aplikace

Hlavní obrazovku aplikace zaplňuje seznam s již naskenovanými Pokémony. Z obrazovky je možné přesunout se na detaily buňky ze seznamu či navštívit stránku s přidáním nového Pokémona. Obrazovka má přístup i k bočnímu menu, které dále poskytuje vstup až na 12 různých obrazovek. Každá z nich nabízí návrat do menu. Obrázek 3.2 obsahuje hlavní seznam a detailní pohled. Pro ucelenější představu o propojení částí v aplikace jsem si připravil mapu 3.3.



Obrázek 3.2: Názorná ukázka využití pravidel

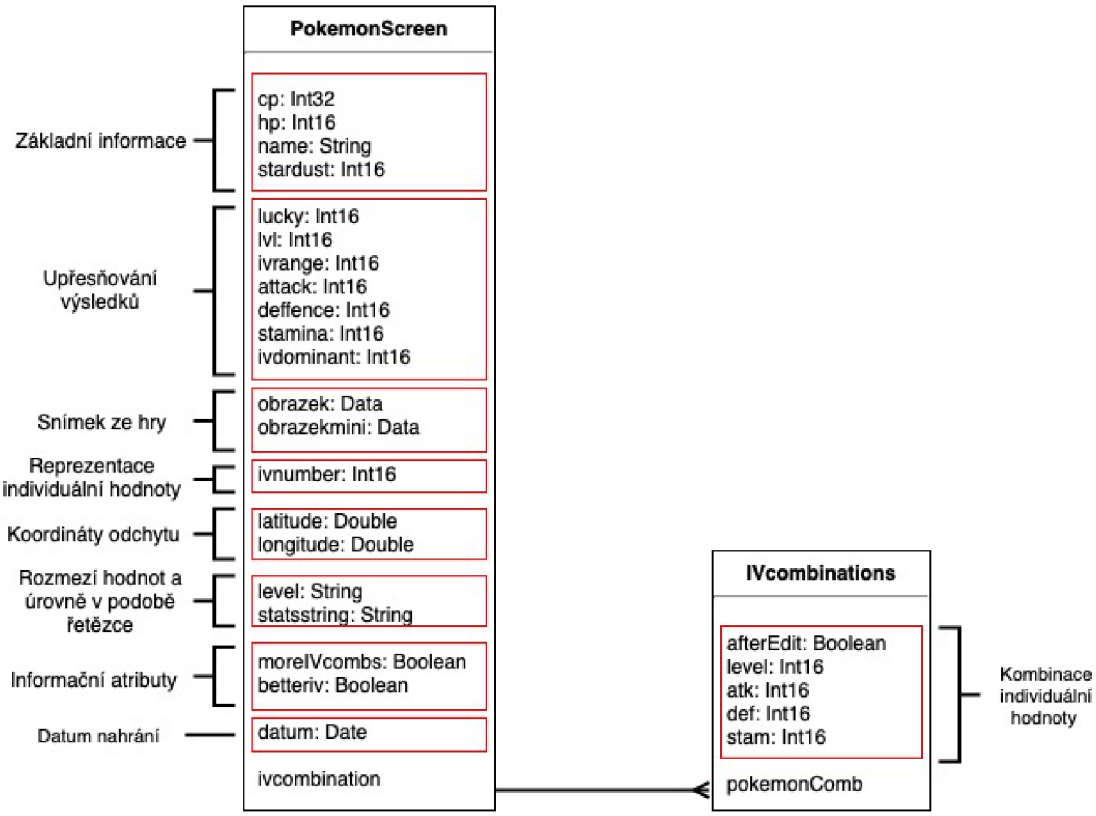


Obrázek 3.3: Mapa aplikace

Zaoblené malé odbdelníky představují různé typy obrazovek. Úvodní obrazovka je značena výraznou červenou barvou. Obrazovky, na které se lze dostat z bočního menu, jsou znázorněny červenou barvou. Obdelníky znázorněné zelenou představují takzvané pop–up[1] obrazovky, které jen překrývají tu aktuální. Šipky na obrázku znamenají přechody mezi obrazovkami. Kromě šipek směřujících od přidání Pokémona k detailnímu pohledu na něj jsou všechny šipky oboustranné. To znamená, že je možné se vždy vrátit zpět odkud uživatel přišel a nemůže se stát, že by zdlouhavě hledal cestu zpět. Větší obdelníky mají význam redukční. Redukují počet šipek v obrázku 3.3, aby byl přehlednější. Bez nich by šipky musely vést ke každému menšímu obdelníku obsažený v obdelníku větším.

3.9 Struktura CoreData

Pro ukládání snímků a dat z nich získaných, byl využit objektový graf CoreData. Důvodem výběru byla i snaha o co nejmenší spotřebu mobilních dat. Hráč již spotřebovává větší množství dat hraním hry a není příjemné, aby i podpůrná aplikace měla tutéž vlastnost. Graf je složen ze dvou objektů **PokemonScreen** a **IVcombinations**. První obsahuje všechna potřebná data o daném snímku obrazovky a Pokémonovi v něm. Druhý obsahuje přesně danou kombinaci. PokemonScreen může být ve vztahu s 0 až N **IVcombinations**. Právě jedna kombinace může být ve vztahu s právě jedním **PokemonScreen**. Co tyto objekty obsahují najdeme v Obrázku 3.4. Ve spodní části každého z objektů se nachází jejich vztahy. Obrázek 3.4 znázorňuje dvě použité entity. Každá z nich má rozdělené atributy do skupin



Obrázek 3.4: Objekty v CoreData

[1] Vyskakující okno překrývající aktuální obrazovku

podle příbuznosti a významu. První skupinou jsou základní informace, které obsahují to, co se dá vyčíst z naskenovaného obrázku. Obsahem této skupiny jsou:

- *cp* – finální síla Pokémona získaná ze snímku obrazovky,
- *hp* – životy Pokémona získané ze snímku obrazovky,
- *name* – konkrétní jméno Pokémona,
- *stardust* – hodnota potřebná pro výpočty a je ukládána pro možnost úprav nekorektního získání dat.

Všechny atributy je potřebné znát pro vypočtení vzorců 3.1 a 3.2. Díky znalosti jména vybereme správné základní hodnoty Pokémona a množství prachu udávající rozmezí jeho potenciální úrovně.

Další skupina atributů je složena z prvků upřesňující výsledky výpočtů. Nabývá hodnot:

- *lucky* – znalost, zda-li je Pokémon šťastný,
- *lvl* – určeno pro ukládání ručně zadané úrovně. Pokud není zadána, je hodnota automaticky 0,
- *ivrange* – stejně jako ivdominant obsahuje jedno číslo v rozmězí 0 až 4. Určuje bližší rozmezí, ve kterém se daná individuální hodnota pohybuje,
- *attack* – nabývá hodnoty 0 nebo 1 podle zadání uživatele v ohodnocení, zda-li jeho Pokémon má dominantní hodnotu útok,
- *defence* – stejně jako útok nabývá buďto 0 nebo 1. Je určen na základě hodnocení dominantní hodnoty,
- *stamina* – nabývá hodnot 0 nebo 1 stejně jako útok nebo obrana. Nastaví se na 1, když je dominantní hodnotou výdrž,
- *ivdominant* – říká, v jakém rozmezí hodnot se nachází dominantní hodnota. Může obsahovat čísla od 0 do 4, kde 0 – nespecifikováno, 1 – nejlepší až 4 – nejhorší.

Obsah těchto hodnot je využit vždy při jakékoliv jejich změně pro úpravu celkového počtu platných kombinací.
Následující skupinka nese informace o naskenovaném obrázku. Náplní skupiny jsou:

- *obrazek* – kompletní naskenovaný obrázek typu Data,
- *obrazekmini* – oříznutá verze naskenovaného obrázku pro reprezentaci vizuálu Pokémona v seznamu.

Obrázky jsou využívány pro vizualizaci v kolekci chycených Pokémonů a pro znovu zobrazení nasnímaného snímku při snaze přistoupit na jejich detailní náhled.

Po zjištění platných kombinací přichází na řadu uložení průměrné individuální hodnoty:

- *ivnumber* – průměrná individuální hodnota Pokémona.

PokemonScreen nabízí ukládání koordinátů nálezu úlovku v podobě:

- *latitude* – zeměpisná šířka místa nálezu úlovku,
- *logitude* – zeměpisná délka místa nálezu.

Pozice je reprezentována anotací v mapě na obrazovce detailního pohledu na Pokémona.

Pro účely rychlého zobrazení základních informací o Pokémonovi v seznamu uchovává PokemonScreen dva atributy:

- *level* – rozmezí, ve kterém se nachází úroveň Pokémona ve formě řetězce,
- *statsstring* – řetězec obsahující rozmezí hodnot útok, obrana, výdrž.

Eliminují potenciální nutnost jakéhokoliv výpočtu při vytváření seznamu.

Jednou z posledních skupin jsou dodatečné informace. Ty pomáhají v reprezentaci dat a nastavení tlačítek do pozice odpovídajícím právě těmto atributům.

- *moreIVcombs* – pomocný atribut říkající, že objekt má více kombinací individuální hodnoty,
- *betteriv* – určení, zda úlovek má automaticky lepší individuální hodnotu (66,7% a více). Nabývá taktéž hodnot 0 nebo 1.

Poslední skupinu tvoří jeden atribut, který informuje o posledínm přidání daného snímku:

- *datum* – datum a čas, kdy byla provedena. poslední manipulace s daným snímkem,

V moment, kdy uživatel přidá stejný obrázek znovu v jiný čas, aplikace pouze aktualizuje tento atribut a nepřidává nový objekt do databáze.

IVcombinations entita obsahuje jedinou skupinu, a tou je kombinace individuální hodnoty:

- *afterEdit* – informace, zda kombinace vyhovuje podmínkám ohodnocení a ručnímu zadání úrovně,
- *atk* – hodnota útoku kombinace,
- *def* – hodnota obrany kombinace,
- *level* – úroveň, pro kterou platí tyto hodnoty,
- *stam* – hodnota výdrže kombinace.

Využitím vzorce 3.3 na každý objekt svázaný s jedním Pokémonem a následném vyhledání minima a maxima, získáme procentuální rozmezí vybraného jedince.

Kapitola 4

Implemetace aplikace

Pro implemetaci bylo využito programovacího jazyka Swift verze 4.2. Celá práce byla vytvářena ve vývojovém prostředí Xcode verze 10.2. beta 2 na notebooku od firmy Apple. Krom knihoven UIKit (pro tvorbu uživatelského rozhraní) a CoreData (pro práci s perzistentními daty) od firmy Apple, bylo využito knihovny Firebase od firmy Google a open-source knihovny SWRevealViewController (pro implementaci postranního menu v jazyce Objective–C). První, čím se kapitola zabývá je, jakým způsobem bylo vytvořeno uživatelské rozhraní, a dále více do podrobna popisuje postup implementace sběru, a reprezentace dat od uživatele. Ukázány jsou i způsoby ukládání dat do CoreData a řešení možných problémů, které mohou při této akci nastat. Konec kapitoly se věnuje postupu zvolenému při implementaci rad, triků a aktuálních informací pro trenéry.

4.1 Uživatelské rozhraní

Tato podkapitola rozšiřuje informace předané v podkapitole 3.8 o návrhu uživatelského rozhraní. Z důvodu snadné orientace v aplikaci jsou všechny hlavní obrazovky realizovány pomocí UINavigationViewController a obsahují přístup k postrannínu menu, které je možné dosáhnout pomocí tlačítka v levém horním rohu nebo posunutím obrazovky prstem na pravou stranu. Díky UINavigationViewControlleru má uživatel vždy možnost se vrátit zpět na jednu z hlavních obrazovek. Primární display je realizován pomocí UITableView a je naplněn seznamem naskenovaných Pokémonů z CoreData, o tom více v podkapitole 4.3 o ukládání a načítání dat. Obsahuje možnost vyhledávání pomocí vyhledávače (realizován skrze UISearchBar) pro nalezení hledaného Pokémona. Funkce vyhledávání není jediné, čím se dá ulehčit uživatelovo hledání. Pro účely seřazení buněk bylo využito pop–upView s možnostmi řazení. Po výběru buňky s Pokémonem ze seznamu v UITableViewController je uživatel přesunut na obrazovku s detailním pohledem na daného Pokémona. Obrazovka se skládá z UIViewController a obsahuje dva hlavní a tři vedlejší prvky. Mezi hlavní se řadí UIImage obsahující naskenovaný úlovek hráče, a UIScrollView s informacemi získanými z obrázku a následných výpočtů. Tři vedlejší prvky jsou realizovány jako UIButton a jejich funkcí je vyvolat pop–upView s ohodnocením Pokémona (upřesnění výsledku), kalkulačkou s výpočty pro nadcházející úrovně a formulářem pro upravení možných chyb při sběru dat. Návratem zpět na úvodní obrazovku, tedy na seznam s Pokémony, je možné přistoupit k menu. Při přechodu na menu a snaze přejít na jiné UINavigationViewController je možno setkat se s UIColectionView nabývající horizontálního či vertikálního posouvání obsahu,

nebo také s dalšími UITableView. Realizace těchto dalších částí aplikace bude dále podrobně vysvětlena v podkapitole 4.8 a 4.9.

4.2 Získávání dat ze snímků obrazovky

U získávání dat ze snímku bylo zapotřebí najít způsob, který bude spolehlivě dodávat text z obrazu v dostatečné kvalitě na to, aby se s ním dalo dále pracovat. Pro tyto účely jsem vybral Tesseract OCR a následně Firebase ML text recognition. První zmiňovaný byl mou první volbou, ale po prvních testech se ukázalo, že nevykazuje dostatečných kvalit co se výsledného textu týče. Využita byla proto druhá varianta, a to Firebase. I když získávání dat nebylo úplně stoprocentní, výsledky byly dostačující, a proto byla API použita. Z důvodu možnosti používat rozpoznávání textu bez nutnosti připojení k internetu jsem vybral variantu běžící na zařízení. Návrh získávání dat je popsán na obrázku 3.1. Nadcházející podkapitoly se věnují jednotlivým částem návrhu.

Rozpoznání textu

Vše začíná výběrem obrázku z knihovny obrázků. Jakmile si uživatel vybere snímek ze hry, přichází na řadu Firebase API. Ukázka kódu níže znázorňuje potřebnou část kódu pro práci s rozpoznáváním textu od Firebase.

```
let vision = Vision.vision()
let textRecognizer = vision.onDeviceTextRecognizer()

let visionImage = VisionImage(image: pickedImage)
textRecognizer.process(visionImage){ features, error in
        guard error == nil, let features = features else {
        // kdyz se stane chyba
        return
        }
        // prace s rozpoznanym textem
}
```

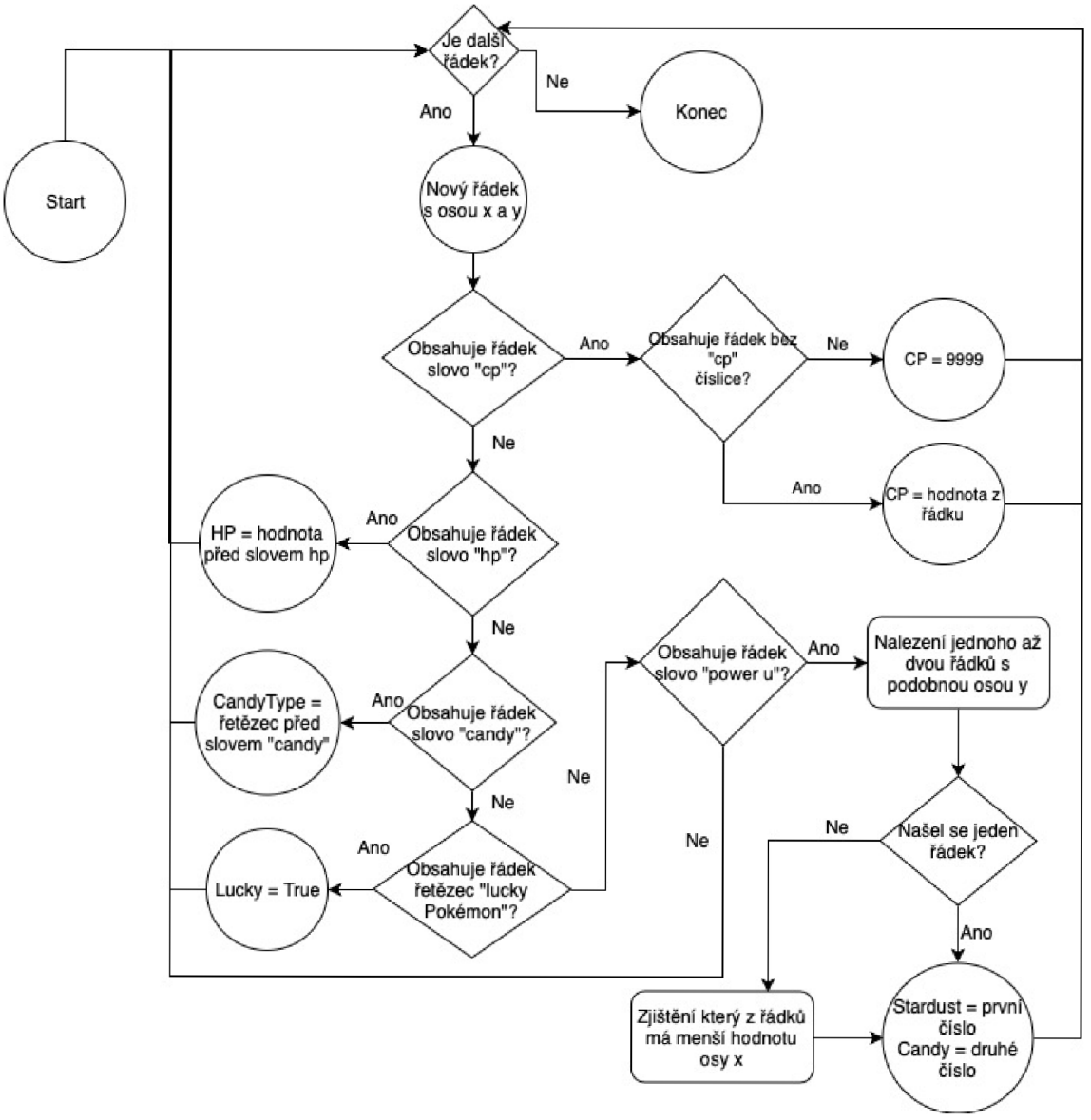
Výpis 4.1: Implementace Firebase ML text recognition

Proměnná pickedImage jako vybraný obrázek ze hry je vložen do procesu VisionTextRecognizer (v kódu vyobrazen jako textRecognizer). Následuje test na chybu při práci s obrázkem a po něm již možnost práce se získaným textem uloženým v proměnné features. Z nabízených možností seskupení dat, byl využit text seřazen po řádcích. Při této akci nastal problém, který byl nutný vyřešit. Text v jednom řádku na obrázku ve výsledku na řádku nebyl. Nejspíš z důvodu rozdíné velikosti písma a větší mezery mezi nimi. Problém byl vyřešen využitím os x a y získaných řádků, které napomohly tento řádek spojit v jeden.

Analýza dat

Po získání dat z obrazu je potřeba analyzovat a vybrat z nich jen to podstatné pro naše výpočty. Na analýzu dat byla vytvořena funkce *ParseData*, ve které se vypreparují důležitá data a ta jsou pak poslána dál. Obrázek 4.1 znázorňuje, jakým způsobem jsou data získávána. Kosočtverce reprezentují podmínky, které jsou kontrolovány. Kroužky znázorňují ukládání získaných dat a zaoblené obdelníky informují o prováděné akci mezi podmínkami či uložením dat. Zde se též zjišťuje, zda máme všechny potřebné data pro výpočty individuálních hodnot. V případě, že získané informace jsou nekompletní, přichází na scénu okno

umožňující uživateli doplnit vše chybějící. Nabízí i vše, co doposud aplikace sama zjistila. Uživatel nemusí informace doplnit a raději se rozhodnout pro opuštění upravování informací, což zapříčiní ukončení procesu a snímek s daty není uložen. V opačném případě jsou kompletní informace poslány dál na výpočty všech možných platných kombinací.



Obrázek 4.1: Vývojový diagram získání potřebných dat z rozpoznaného textu ve funkci ParseData

Výpočty individuálních hodnot

V této fázi by jsme měli mít již kompetní údaje potřebné k vypočítání individuálních hodnot možných pro naskenovaného Pokémona. Celou akci má na starost funkce *CalcIVandGetName*, která přijme znalosti z analýzy dat a využije je k výpočtům. V první řadě se určí rozmezí úrovní, ve kterém se Pokémon nachází, pomocí prachu potřebného k vytrénování na vyšší úroveň. Rozmezí je většinou o velikosti čtyř půl úrovní (každé vytrénování Poké-

mona zlepší o jednu půl úroveň). Následuje zjištění, o koho se přesně jedná. Každý jedinec a jeho podevoluce a evoluce mají stejný název dobrot, díky kterým aplikace určuje, o jakého Pokémona se jedná. Funkce projde všechny možné evoluce a podevoluce a vyzkouší, jestli je možné, aby do těchto čtyř půl úrovní spadaly. Při zjištění, že do rozmezí sedí více Pokémonů, je připraveno okno, které nabídne uživateli ruční určení. Jakmile máme jasno o jakého Pokémona jde, přichází na řadu testování všech možných kombinací. Zde jsou využity vzorce zmíněné v podkapitole 3.5. Konkrétně vzorec 3.1 a následně 3.2. Porovnáním výsledků rovnic a získaných dat z obrázku se uloží kombinace splňující rovnost u obou srovnání. Jakmile jsou vyzkoušeny všechny možnosti, je výsledek předán dál. Pokud se zjistí, že výsledkem není ani jedna platná kombinace, znovu je přivoláno okno s možností úprav a uživatel je nucen opravit chyby v získávání dat. Po dokončení této akce je znovu volána funkce *CalcIVandGetName* a celý proces je opakován znovu. Pokud je vše při výstupu z funkce v pořádku, je na řadě data uložit do databáze, a prezentovat je na obrazovku uživateli.

4.3 Využití CoreData

Pro ukládání dat s Pokémony bylo využito CoreData, hlavním důvodem je ukládání dat na zařízení, kdy není potřeba připojení k internetu. Entity s jejich atributy jsou popsány v podkapitole 3.9 a vztah mezi nimi je znázorněn v obrázu 3.4. Pro práci s CoreData byly vytvořeny tři funkce zabývajícími se přidávání, odebíráním a upravou objektů. Další funkce využívající CoreData jsou zjišťování existence objektu v databázi a vytažení jeho hodnot.

AddPokemonToCoreData

AddPokemonToCoreData se věnuje přidáním objektů do CoreData. Vytváří jeden objekt PokemonScreen a větší množství IVcombinations podle počtu platných kombinací.

UpdatePokemonInCoreData

UpdatePokemonInCoreData je funkce využívaná pro úpravu obsahu CoreData, žádné objekty nevytváří, jen mění hodnoty již existujících objektů. Funkce je využita například, když uživatel ohodnotní svého Pokémona ve snaze zredukovat počet kombinací.

DeletePokemonFromCoreData

DeletePokemonFromCoreData odstraní PokemonScreen objekt a všechny na něj závislé IVcombinations z CoreData. Využita byla na jediném místě aplikace, a to v seznamu s Pokémony. Při snaze posunout buňku z pravé do levé strany obrazovky se objeví možnost smazaní této buňky ze seznamu, a tím i celkově z paměti.

4.4 Využití UserDefaults

UserDefaults bylo využito pro nastavení individuálních preferencí uživatele. Projevují se u seřazení Pokémonů v seznamu a při ohodnocení úlovku. Seznam lze pomocí pop–up view, které je přístupné přes tlačítko s šipkou směřující směrem dolů řadit. Řazení je primárně nastaveno podle data nahrání od nejnovějšího po nejstarší. Uživatel si může vybrat z pěti možností (datum, individuální hodnota, finální síla, životy, jméno). Vybraný typ řazení zůstává díky UserDefaults i přes vypnutí aplikace. Dále se využívá při ohodnocení Pokémona,

a to pro upřesnění jaké věty zprostředkovat uživateli, jelikož se věty pro každý tým mění. Uživatel má možnost si svůj tým navolit v nastavení či při prvním styku s obrazovkou ohodnocení. Jakmile je jednou tým navolen, jde jednoduše změnit v nastavení.

4.5 Správa dat – NSFetchedResultsController

Správě dat v seznamu Pokémonů na úvodní obrazovce se věnuje NSFetchedResultsController. Delegát kontroluje změny v databázi a při jakékoliv změně upravuje obsah seznamu. Ukázku implementace Controlleru pro správu můžeme vidět ve Výpise 4.2.

```
lazy var fetchedResultsController:NSFetchedResultsController = {()->
 NSFetchedResultsController<NSFetchRequestResult> in

 let context = GetContext()

 let fetchRequest = NSFetchRequest<NSFetchRequestResult>
  (entityName: "PokemonScreen")

 fetchRequest.sortDescriptors = SortTable()
 let frc = NSFetchedResultsController(fetchRequest: fetchRequest,
  managedObjectContext: context, sectionNameKeyPath: nil,cacheName: nil)
 frc.delegate = self

 return frc
}()
```

Výpis 4.2: Implementace NSFetchedResultsControlleru pro srávu seznamu Pokémonů

Funkce *GetContext* vytvoří obraz databáze a uloží ho do proměnné context. Pomocí NSFetchedRequestu zažádáme o všechny objekty entity PokemonScreen. *SortTable* rozhodne podle jakého pravidla se bude seznam řadit. Záleží, zda–li je uložen osobní výběr řazení prvků v UserDefaults. V případě, že UserDefaults žádnou informaci o řazení neosahuje, je nastaveno řazení podle data nahrání do aplikace. Samotné vytvoření NSFetchedResultsControlleru je realizováno za pomoci konstruktoru, kterému je nutno předat fetchRequest a managedObjectContext. Za následné dva parametry je možno zadat nil v případě, že nehodláme mít více jak jednu sekci a nepoužijeme cache paměť.

4.6 Komunikace s pop–up obrazovkami

Jak je vidět na mapě aplikace 3.3, z obrazovky s informacemi je možné přistoupit až ke čtyřem různým pop–up pohledům. Tyto pop–upy dostávají informace, se kterými následně mohou pracovat pomocí přechodů, takzvaně segue. Každý přechod má svůj unikátní název, a proto je možné předat informace v dobu, kdy má být zavolán. Pro ukázku je ve výpisu z kódu aplikace 4.3 znázorněn způsob předání informací.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
  //Segue na popUpview pro ohodnoceni pokemona
  if segue.identifier == "PopUpAppraisal" {

    let popupappraisal = segue.destination as! AppraisalViewController

    popupappraisal.pokemon = self.PokemonInfo
    popupappraisal.BetterIV = self.PokemonBetterIV.isOn
```

```
        popupappraisal.delegate = self
    }
}
```

Výpis 4.3: Předání informací jinému UIViewControlleru promocí segue

Funkce je volána při jakémkoliv přechodu. Kontroluje, zda indentifikátor náhodou nesedí k některému z definovaných. V případě, že by se přešlo na pop–up s ohodnocení Pokémona, bude proveden kód znázorněný ve výpisu 4.3. Vytvoří se proměnná nesoucí všechny prvky Controlleru, na který je v plánu přejít, a do daných prvků jsou uloženy hodnoty z předchozího Controlleru, aby bylo možné s nimi dál pracovat.

Delegate

Při vrácení se zpět z pop–up pohledu je nutné informovat obrazovku, ke které se vracíme, o změnách provedených jinde. Problém byl v průběhu vývoje řešen dvěma způsoby. Nejdříve pomocí notification center, a poté delegate protokolů. Práce s protokoly mi přišla jednodušší a přehlednější, proto kompletně nahradila první způsob informující o změnách. Pro práci s delegate protokoly bylo nutné vytvořit protokol a nechat ViewController, na který se po ukončení změn v pop–upech třeba vrátit, aby protokol dědil. Pro přístup k funkcím protokolu z jiného Controlleru, než kde jsou definovány, je potřeba při přechodu předat i referenci na daný ViewController, jako tomu je ve výpisu 4.3. Nyní je pop–up schopen přistupovat k funkcím v protokolu a jsou vykonány jiným ViewControllerem.

4.7 Postranní menu – SWReavealViewController

Po zhodnocení způsobů realizace struktury aplikace bylo rozhodnuto, že nejlepším způsobem, kterým se uživatel bude pohybovat po aplikaci, je menu. Při průzkumu existujících aplikací se mi zalíbila varianta menu vyjíždějícího ze strany obrazovky a nezabírajícího celou plochu. Při řešení způsobu implementace jsem se setkal s dvěmi možnostmi realizace. Jednou z nich (nakonec nevyužitou) bylo vytvoření menu za pomoci Notification Center. Druhá v pořadí a poslední uvážená možnost byla SWReavealViewController. SWRevealViewController je opensource knihovna pro vytváření bočních menu v jazyce Objective-C. Pro zakomponování knihovny do aplikace je nutné vytvořit počáteční ViewController a přiřadit mu třídu SWRevealViewController. Dalším krokem je vytvoření dvou Controllerů. Jedním z nich je Controller zastupující obsah úvodní obrazovky a druhý realizující menu. Oba Controllery jsou spojeny přechodem s počátečním ViewControllerem. Přechod z úvodní obrazovky k ViewControlleru musí mít identifikátor **sw_front** a z menu **sw_rear**. O nastavení přístupu k menu se u každého hlavního ViewControlleru stará funkce *Menu*, kterou můžeme vidět ve Výpisu 4.4.

```
func Menu() {

    revealViewController().rearViewRevealWidth = 255

    MenuButton.target = revealViewController()
    MenuButton.action = #selector(SWRevealViewController.revealToggle(_:))

    view.addGestureRecognizer(self.revealViewController().panGestureRecognizer())
}
```

Výpis 4.4: Funkce pro nastavení přístupu k bočnímu menu

Přístupy jsou ve funkci nastaveny dva. Jeden pomocí tlačítka menu v horním levém rohu každé hlavní obrazovky a druhý při snaze zatáhnout za levý okraj. Menu se zpět zatáhne při stisknutí na jednu z nabídek v menu či zatažením menu z pravé strany na levou.

4.8 Způsob zobrazení tipů a triků pro trenéry

Ve snaze ukázat hráčům, kteří jsou buďto začátečníci, nebo už toho mají pár nachytáno, tipy popřípadě triky, bylo zapotřebí vybrat rozumný způsob prezentace těchto dat. Hlavním prvkem části aplikace se proto stalo UICollectionView. UICollection view dává možnost posouvání obsahu jak vertikálně, tak horizontálně. Při buňkách o velikosti CollectionView je navozován pocit listování jako v elektronické knize. Obsah je dynamický, proto je lehké ho doplňovat o nově získané informace, a nebo také odebírat ty zastaralé. CollectionView nabízí možnost stránkování, čehož bylo využito za účelem nenutnosti ručního zastavování posouvání obsahu. Pro listování je možné využít posouvání stran prstem na jednu či druhou stranu, nebo tlačítek na spodní části obrazovky.

4.9 Způsob zobrazení aktuálních informací pro trenéry

Společnost Niantic viditelně nerada dává hráčům vše potřebné pro hraní a odměny z různých typů úkolů, vajec či raidů nejsou výjimkou. Tyto informace jsou hráči hledány na internetových stránkách a ne vždy jsou tyto informace aktuální. Část aplikace s aktuálními informacemi pro trenéra byla vytvořena za účelem seskupit často měnící se informace ve hře hůře dohledatelné. Pro zprostředkování aktualit bylo využito UITableView a UISearchBar. TableView je tvořeno dynamicky na základě velikosti pole s úkoly a počty možných Pokémonů za splnění daného úkolu. Všechny tři TableView jsou tvořeny pomocí pole klíč hodnota, kde klíčem je enum podle kterého se poté i sekce řadí v seznamu a hodnota obsahuje všechny úkoly, možné Pokémony, či raid spadající do dané sekce. V případě úkolů a vajec je možné sekce srolovat a zmenšit tím celkovou velikost TableView. Srolování dosáhneme dotknutím se názvu dané sekce. Při opětovném dotknutí se sekce se znovu dostaneme k obsahu sekce. Informace o tom, zda je sekce viditelná, či ne, je uložena v hodnotě, která přísluší dané sekci. UISearchBar byl implementován pro snadnější nalezení potřebné informace na základně znalosti například názvu úkolu, pro který by jsme rádi znali jeho odměnu. Text vepsaný do UISearchBaru je použit pro hledání úkolu obsahující daný řetězec.

Kapitola 5

Testování aplikace

Nadcházející kapitola pojednává o průběhu testování při vývoji aplikace. Vybranými zařízeními pro testování byly iPhone SE, 8 a X. Jako první bude rozebráno testování funkčnosti aplikace při vytváření. Další částí je testování uživatelského rozhraní a jeho prvků. Poté, jak byla aplikace testována komunitou, jaké byly výsledky těchto testování, a jak ovlivnily vzhled aplikace či její chod.

5.1 Průběžné testování

Průběžné testování aplikace je nezbytné a nemělo by se zanedbávat. Z tohoto důvodu byla testována každá menší změna v aplikaci. Xcode nabízí možnost testovat aplikace na simulátoru různých typů zařízení. Pro účely práce byly vybrány mezi testovací zařízení iPhone SE, 8 a X z důvodu větší flexibility aplikace. Testováno bylo společně se simulátorem i na reálném zařízení, které bylo v pozdější fázi vývoje vysíláno i do terénu a zjišťovalo se pomocí něj, jaké nedostatky aplikace má. Prvotní větší testování probělo při zkoumání funkčnosti knihovny pro rozpoznávání textu. Při testech knihovny jsem si dával za cíl najít co nejvíce nedokonalostí při získávání dat. Nedokonalosti byly následně řádně ošetřeny ve funkci ParseData(), která z dat získaných z obrazu získává důležitá data pro výpočty. V průběhu vývoje prošlo aplikací kolem 500-600 snímků obrazovky s různými Pokémony. Otestována přibližně polovina Pokémonů, které jde ve hře aktuálně chytit, a pro všechny tyto Pokémony byla optimalizovaná ještě před dokončením celé funkcionality.

5.2 Vzhled uživatelského rozhraní

Uživatelskému rozhraní je potřeba věnovat velkou část času vývoje. Testování uživatelského rozhraní probíhalo jednak ohledně rozmístění prvků na obrazovce pro různá zařízení, jednak pro výběr té správné barvy pro daný prvek. Pro pozicování prvků bylo využito omezení pomocí autolayoutů, které se starají o správnou vizualizaci prvků na různých zařízeních, ne jen na daném zařízení pro které vyvíjíme. Pomocnými prvky při testování správného rozmístění prvků a jejich vizualizaci bylo využito preview (Xcodem nabízená možnost nahlednutí na vizuální stav obrazovky na námi zvolených zařízeních) a simulátor simulující námi zvolené zařízení. Nevýhodou preview byla realizace prvků jen ve storyboardu a všechny kódové změny nebyly vidět. Simulátor tyto změny již obsahoval a proto, když bylo potřeba, byla spuštěna simulace. V ostatních případech bylo vystačeno s preview. Co se výběru barev

týče, byla aplikace z prvu více barevná a odrážející kvalitu daného prvku. V průběhu byla barevnost prvků razantně zredukována z důvodu přehlednosti aplikace.

5.3 Komunitní testování

Komunitní testování proběhlo, jakmile byla aplikace hotová. Hráčům byla aplikace nahrána do zařízení a následně byli ponecháni s aplikací po dobu nejméně pěti dnů. Nejdříve po pěti dnech byl s hráčem proveden rozhovor, kde byl tázán na různé otázky od uživatelského rozhraní po celkovou funkčnost aplikace. Jednou z otázek bylo například, zda-li jim přijde rozmístění prvků na obrazovkách logické. Na tuto otázku většina dotazovaných odpověděla, že by něco málo změnili, ale hlavně by zvětšili buňky v seznamu Pokémonů. Podle mnohých nejsou vidět všechny informace v takové míře, aby si jich uživatel dostatečně všiml. Připomínka byla vzána v potaz a buňky byly zvětšeny na velikost, která již hráčům přišla dostačující. Další z otázek bylo, jestli se líbí barevné spektrum vybrané pro aplikaci. Po seskupení odpovědí na tuto otázku se přišlo na to, že uživatel ani tak neocení barevnost aplikace, jak spíše přehlednost. Na základě těchto znalostí byla většina barevných prvků odstraněna a zaměněna za odstíny šedé barvy, které nepůsobí tak agresivně jako více různých barev.

Při rozhovoru byl hráči podán i dotazník s uzavřenými odpověďmi, na které bylo možné odpovědět jedině číslem od jedné do deseti. Čím vyšší číslo, tím lepší hodnocení daného testovacího prvku. Výsledky hodnocení aplikace všech hráčů jsou znázorněny v Tabulce 5.1.

Prvky testování	Vendy	Adam	Petr	Víťa	Jirka
Funkčnost aplikace	9/10	8/10	8/10	9/10	8/10
Intuitivnost	10/10	9/10	10/10	10/10	10/10
Validita výsledků dat	9/10	7/10	7/10	9/10	8/10
Uživatelské rozhraní	9/10	8/10	9/10	8/10	8/10
Míra užitečnosti aktuálních informací	10/10	10/10	10/10	10/10	10/10

Tabulka 5.1: Výsledky komunitního testování

Uživatelé chválí využití známých tvarů tlačítek z již existujících aplikací, což se odráží na míře intuitivnosti používání aplikace. Mírné zklamání z většího rozmezí platných hodnot přineslo i ne tak dobré výsledky u validity výsledků. I když se ve výsledcích neobjevují žádné nevalidní výsedky, uživatelé by rádi věděli hned přesnou kombinaci bez nutnosti uživatelských úprav. Největším plusem bylo informování hráčů o aktuálních informacích. Každý z hráčů, který testoval aplikaci, si část s informacemi vychvaloval a přišla všem užitečná.

5.4 Výsledky testování

Díky průběžnémmu testování bylo chybám zabráněno takřka okamžitě. Našlo se jen pár nesrovnalostí, které bylo potřeba řešit dodatečně. Testování prokázalo, že aplikace je plně funkční pro 494 Pokémonů z 511. Pro zbylých 17 by bylo potřeba přidat rozšíření analyzátoru o kterého Pokémona se jedná. Někteří Pokémoni mají více forem a aplikace v tuto chvíli počítá jen se základními formami Pokémonů. Rozšíření je v plánu dodělat, aby byla aplikace funkční pro každého možného Pokémona.

Uživatelské rozhraní v průběhu vývoje prošlo změnami, které měly za cíl přehlednost a snadnější porozumění ohledně používání aplikace pro nastávající uživatele. Výsledkem změn bylo rozhraní rozdělené pomocí zaoblených View na různé části. I z ohlasů uživatelů při následném používání aplikace byly sbírány kladné komentáře na způsob rozmístění a rozdělení informací od sebe pomocí View.

Komunitní testování přineslo jak kladné komentáře na celkový chod aplikace, tak i individuální připomínky vůči obsahu, a co naopak vidí jako zbytečnost. Hlavní změnou, která proběhla, byla změna barevného spektra na základě připomínek hráčů. Barevné spektrum bylo s uživateli prodiskutováno a pozměněno.

Kapitola 6

Závěr

Cílem bakalářské práce je vytvoření podpůrné aplikace pro hráče užívající mobilní zařízení iPhone. Aplikace je kompatibilní na iPhone 5, 6, 6s, 7, 8 a X. Hlavním úkonem je získání dat ze snímku ze hry, která jsou následně uživateli prezentována. Aplikace obsahuje sekce informující hráče o aktuálním stavu měnících se informací. Výsledný program je spustitelný a použivatelný na výše zmíněná zařízení. Po sesbírání dat potřebných k vývoji aplikace bylo zapotřebí najít mezery v již existujících aplikacích, aby aplikace co nejvíce splňovala požadavky herní komunity.

Aplikací získaná data jsou uložena a uživatel si je může kdykoliv později prohlédnout znova. Pro účely ukládání dat bylo využito CoreData a pro menší data jako je osobní nastavení se využilo UserDefaults. Během vytváření aplikace byla přidána rozšíření ulehčující práci uživateli při vyhledávání. Jedno z rozšíření představuje zasouvací sekce u seznamu úkolů a vajec pro zredukování počtu buněk, a tím i urychlení hledání. Dalším rozšířením je seřazování prvků seznamu podle námi zvoleného pravidla. Na výběr je pět možných způsobů řazení a změny řazení si aplikace pamatuje i po vypnutí. Ke konci vývoje bylo přidáno nastavení. Dává možnost uživateli změnit výběr svého týmu, který navolil při prvním styku s ohodnocením Pokémona. Všechny doposud naskenované snímky obrazovek lze naráz smazat. Během komunitního testování a důkladnějšího zkoumání konkurenčních aplikací, vypluly napovrch možná rozšíření, která by aplikaci posunuly na vyšší úroveň.

Možné rozšíření

Velice atraktivní rozšíření, které bylo navrhnuto komunitou, může být komunitní hlášení úkolů na různých místech v České republice. Uživatel, který najde úkol, bude mít možnost nahlásit jeho polohu a přidat popisek. Ostatním hráčům by se úkol ukázal v jejich aplikacích. Úkoly se mění každý den o půlnoci, a proto je dobré dopředu vědět, kde jaký úkol v daný den je.

Při zkoumání konkurence byl nalezen mechanismus pro přesnější určování Pokémonovi úrovně, a tím snížení počtu platných kombinací. Vše spočívá ve znalosti hráčovi úrovně a správném nalezení pozice bílé kuličky ve snímku naskenovaného Pokémona.

Rozšíření jsou v plánu implementovat, ale finální verze bakalářské práce tato vylepšení neobsahuje.

Vyhlídky do budoucna

Do budoucna je v plánu nejen přidání rozšíření zmiňované výše, ale také vytvoření aplikace, která by byla použitelná i na zařízení s operačním systémem Android.

Literatura

[1] CocoaPods: CocoaPods Guides. [Online; navštíveno 29.4.2019].
URL https://guides.cocoapods.org

[2] Esplin, C.: What is Firebase? Říjen 2016, [Online; navštíveno 29.4.2019].
URL https://howtofirebase.com/what-is-firebase-fcb8614ba442

[3] Firebase: Machine learning for mobile developers. [Online; navštíveno 29.4.2019].
URL https://firebase.google.com/products/ml-kit/

[4] Firebase: ML Kit for Firebase. [Online; navštíveno 29.4.2019].
URL https://firebase.google.com/docs/ml-kit/

[5] Firebase: Text Recognition. [Online; navštíveno 29.4.2019].
URL https://firebase.google.com/docs/ml-kit/recognize-text

[6] Firebase: Recognize Text in Images with ML Kit on iOS (Firecasts). Prosinec 2018, [Online; navštíveno 9.4.2019].
URL https://www.youtube.com/watch?v=T9TAsurJdmk

[7] Inc., A.: MapKit. [Online; navštíveno 29.4.2019].
URL https://developer.apple.com/documentation/mapkit

[8] Inc., A.: MKMapView. [Online; navštíveno 29.4.2019].
URL https://developer.apple.com/documentation/mapkit/mkmapview

[9] Inc., A.: NSFetchedResultsController. [Online; navštíveno 29.4.2019].
URL https://developer.apple.com/documentation/coredata/nsfetchedresultscontroller

[10] Inc., A.: Setting Up a Core Data Stack. [Online; navštíveno 29.4.2019].
URL https://developer.apple.com/documentation/coredata/setting_up_a_core_data_stack

[11] Inc., A.: UIKit. [Online; navštíveno 29.4.2019].
URL https://developer.apple.com/documentation/uikit

[12] Inc., A.: Model-View-Controller. Červen 2018, [Online; navštíveno 29.4.2019].
URL https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[13] Jacobs, B.: What Is Core Data. Listopad 2017, [Online; navštíveno 29.4.2019].
URL https://cocoacasts.com/what-is-core-data

[14] Jagtap, S.: Core Data with Swift 4 for Beginners. Říjen 2017, [Online; navštíveno 29.4.2019].
URL https://medium.com/xcblog/core-data-with-swift-4-for-beginners-1fc067cca707

[15] Jutt, I.: Data Persistence in IOS – ways to store data in IOS world. Červen 2017, [Online; navštíveno 29.4.2019].
URL https://medium.com/@imranjutt/data-persistence-in-ios-2804d04bde62

[16] Kramár, A.: *DESKOVA HRA PRO APPLE TV*. Diplomová práce, Vysoké učení technické v Brně, 2017.

[17] Kumparak, G.: Pokémon GO's gameplay will soon change based on the real world weather around you. 2018, [Online; navštíveno 23.4.2019].
URL https://techcrunch.com/2017/12/06/pokemon-go-weather/?guccounter=1&guce_referrer_us=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_cs=CfO1UXHkO9Ai-_9lead3Yg

[18] Meltz, R.: Apple Seeks a Swift Way to Lure More Developers. Březen 2014, [Online; navštíveno 5.4.2019].
URL https://www.technologyreview.com/s/527821/apple-seeks-a-swift-way-to-lure-more-developers/

[19] Niantic: The Niantic Story. [Online; navštíveno 23.4.2019].
URL https://nianticlabs.com/about/

[20] Penumutchu, L. P.: The iOS Application Lifecycle. Červenec 2018, [Online; navštíveno 29.4.2019].
URL https://hackernoon.com/application-life-cycle-in-ios-12b6ba6af78b

[21] Peres, R.: Model-View-Controller (MVC) in iOS: A Modern Approach. Červen 2016, [Online; navštíveno 7.4.2019].
URL https://www.raywenderlich.com/1073-model-view-controller-mvc-in-ios-a-modern-approach

[22] Pokemongohub: Pokemon GO IV Guide Explained. Srpen 2016, [Online; navštíveno 23.4.2019].
URL https://pokemongohub.net/post/featured/pokemon-go-iv-guide-explained/

[23] Tips, T.: POKÉMON GO's HIDDEN MECHANICS: CP, IVs, LEVELS, STATS EXPLAINED. Srpen 2016, [Online; navštíveno 23.4.2019].
URL https://www.youtube.com/watch?v=GOQ2Co6nW3c&list=PLmT1HIfnmaWnICAzcaEQwzvnBmmL_n1uf

[24] Wikipedia: Firebase. Duben 2019, [Online; navštíveno 29.4.2019].
URL https://en.wikipedia.org/wiki/Firebase

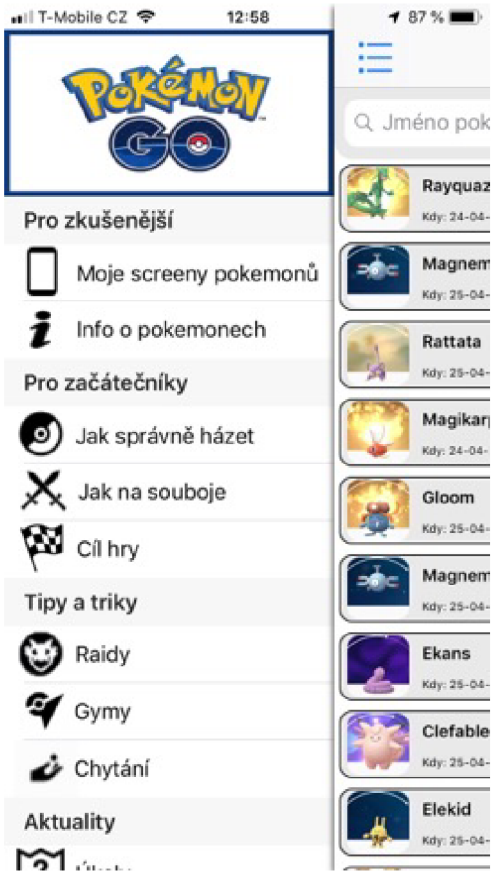
[25] Wikipedia: iOS (Apple). Duben 2019, [Online; navštíveno 29.4.2019].
URL https://cs.wikipedia.org/wiki/IOS_(Apple)

[26] Wikipedia: Pokémon Go. Duben 2019, [Online; navštíveno 23.4.2019].
URL https://cs.wikipedia.org/wiki/Pokémon_Go

[27] Wikipedia: Swift (programming language). Duben 2019, [Online; navštíveno 29.4.2019].
URL https://en.wikipedia.org/wiki/Swift_(programming_language)

Příloha A

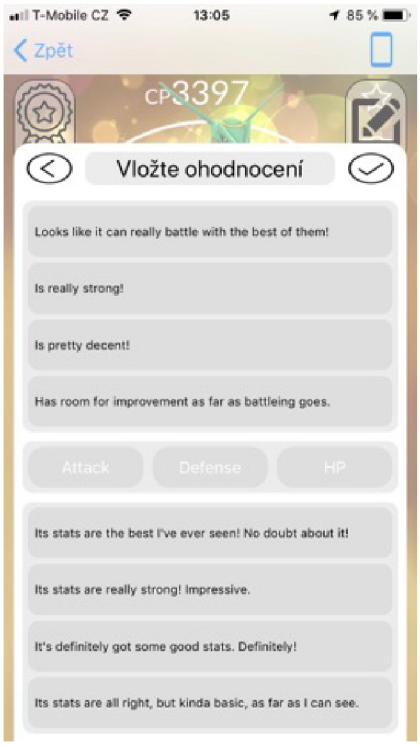
Obrázky aplikace



(a) Postraní menu

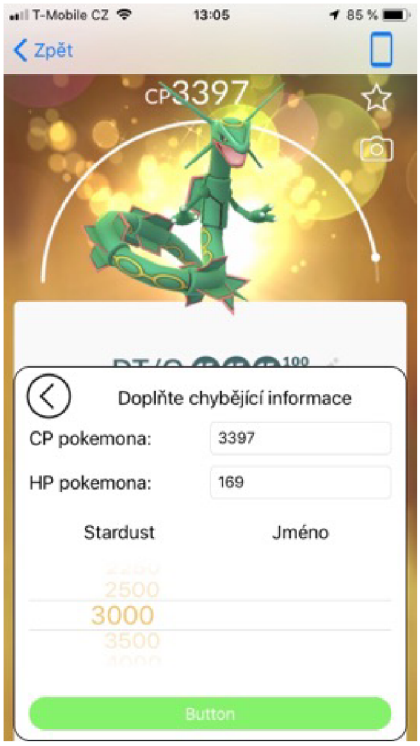
(b) Seznam Pokémonů

(a) Detailní pohled na Pokémona

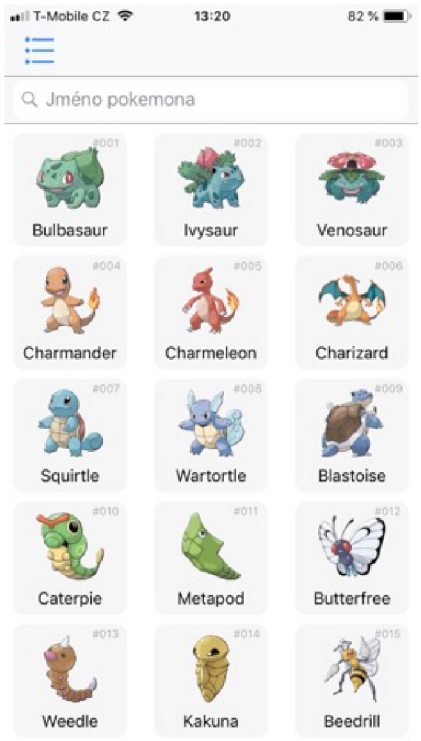


(b) Pop-up ohodnocení Pokémona

(a) Pop-up simulace vytrénování

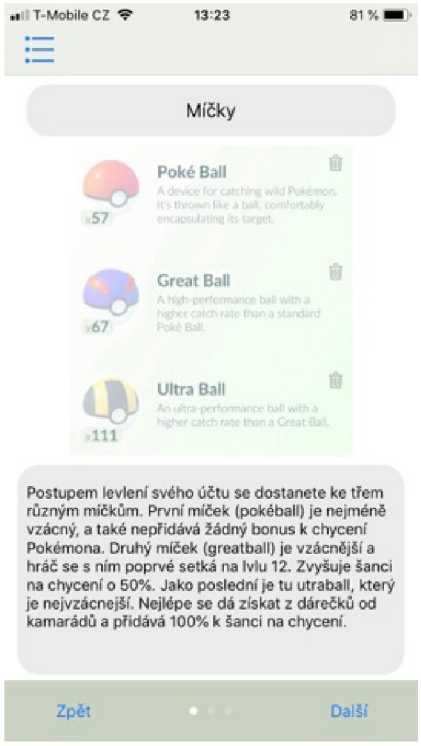


(b) Pop-up editace získaných dat



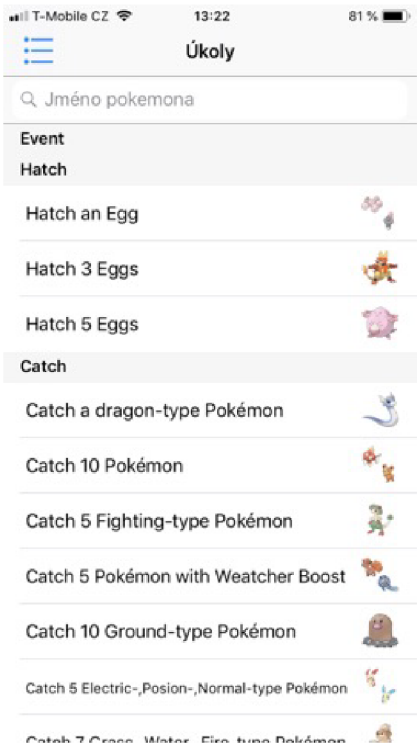
(a) Kolekce všech Pokémonů

(b) Pevně dané informace o Pokémonech

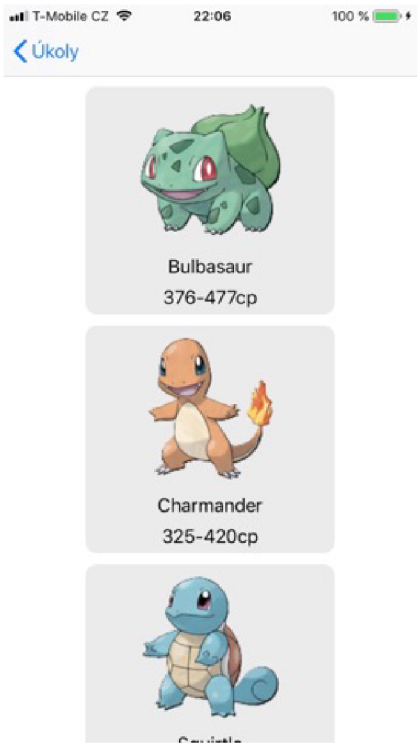


(a) Způsob realizace zprostředkování rad a triků

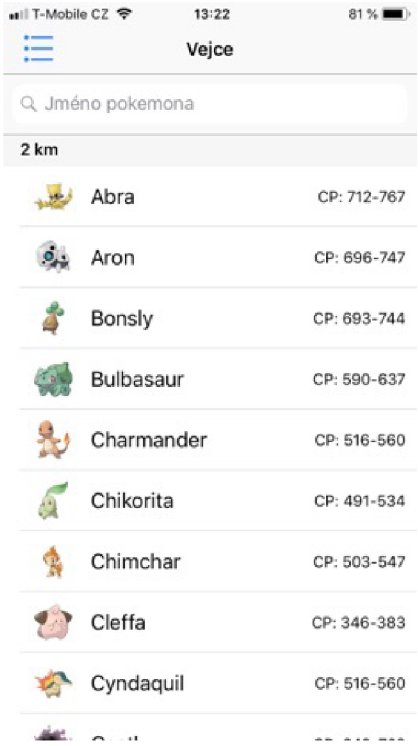
(b) Možnost změny řazeni seznamu



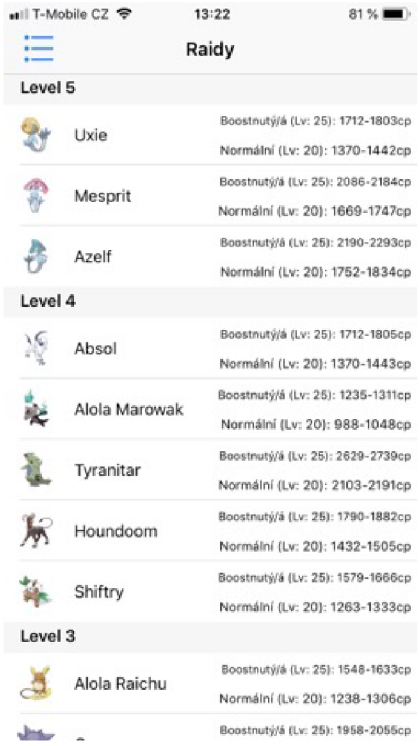
(a) Seznam aktuálních odměn za úkoly



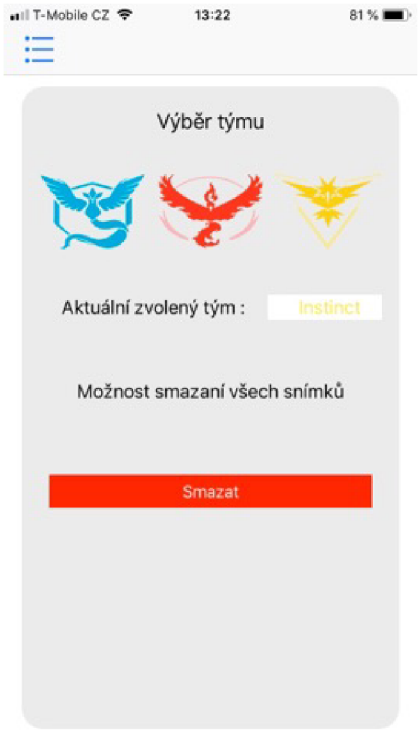
(b) Detailnější pohled na Pokémony z úkolu



(a) Aktuální obsah vajec



(b) Aktuální Pokémoni na raidech


T-Mobile CZ
13:22
81 %
Výběr týmu
Aktuální zvolený tým :
Instinct
Možnost smazaní všech snímků
Smazat

(a) Nastaveni