UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Společenská hra pro Android

Martin Navrátil

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Navrátil**
Osobní číslo:	**I11135**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Společenská hra pro Android**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit adaptaci známé deskové společenské hry pro mobilní zařízení s operačním systémem Android.
Teoretická část:
Bude obsahovat popis programovacího jazyka Java se zaměřením na vývoj aplikací pro OS Android, popis technologie XML využité pro Android v závislosti na API a výhody využití interního databázového systému SQLite. Dále bude uveden soupis základních pravidel společenské hry Activity a výhody její mobilní verze.
Implementační část:
V implementační části bude zpracována adaptace na společenskou hru Activity pro 1 až 49 hráčů na jednom zařízení s OS Android. Bude vytvořena interní databáze, zabezpečení aplikace, grafické rozhraní a komunikace se serverem pro možnost aktualizace a rozšíření databáze.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

UJBÁNYAI, Mirosław. Programujeme pro Android. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.
MURPHY, Mark L. Android 2: průvodce programováním mobilních aplikací. Vyd. 1. Brno: Computer Press, 2011, 375 s. Průvodce (Grada). ISBN 978-80-251-3194-7.
HEROUT, Pavel. Java a XML: průvodce programováním mobilních aplikací. 1. vyd. České Budějovice: Kopp, 2007, 313 s. Průvodce (Grada). ISBN 978-80-7232-307-4.

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání bakalářské práce: **20. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 09. 05. 2014 Martin Navrátil

Poděkování

Děkuji vedoucímu své práce panu Ing. Zdeňku Šilarovi, Ph.D. za pomoc a vstřícný přístup. Děkuji rodině a přátelům za podporu a trpělivost a všem testerům, díky kterým jsem byl schopen práci dovést do finální podoby.

Anotace

Práce byla vytvořena s cílem přenesení oblíbené deskové společenské hry „Activity“ do zařízení s operačním systémem Android. Teoretická část pojednává o hře „Activity“, jejích pravidlech a možné nástavbě. Dále pak o operačním systému Android, databázi SQLite a jazyku XML, které byly v práci využity. Implementační část se zabývá realizací aplikace. Podrobněji je popsána struktura aplikace a vnitřní databáze. Dále jsou popsány možnosti, rozšíření a s tím spojené výhody aplikace oproti původní deskové hře. Závěr práce je zaměřen na zabezpečení aplikace.

Klíčová slova

Activity, Android, Java, SQLite, XML

Title

Social game for Android

Annotation

The bachelor thesis aims to transfer the popular board game „Activity“ to electronic device running on Android operating systém. The theoretical part focuses on the „Activity“ game, its rules and possible extension. Furthermore, it deals with the Android operating system, SQLite database and XML, which are used in it. The implementation section deals with the realization of the application. The structure of the application and internal databases are given a detailed descripton. Furthermore, the possibilities of extension and related benefits in comparison with the original board game are presented. Final part focuses on application security.

Keywords

Activity, Android, Java, SQLite, XML

Obsah

Seznam zkratek

OS	Operační Systém
XML	eXtensible Markup Language
SD	Secure Digital
APK	Application Package File
ER	Entity Relationship
PDA	Personal Digital Assistant
DVM	Dalvik Virtual Machine
SE	Standard Edition
JDK	Java Development Kit
SDK	Android Software Development Kit
ADT	Android Development Tool
DTD	Document Type Definition
UML	Unified Modeling Language
API	Application Programming Interface
SQL	Structured Query Language
ID	IDentification
USB	Universal Serial Bus
ES	Embledded Systems
SGML	Standard Generalized Markup Language

Seznam obrázků

Úvod

První mobilní přístroje se objevily již v 50. letech 20. století a od té doby prošly velkým vývojem do podoby, jakou známe dnes. Postupem času byly na přístroje měněny nároky od prostého přenosu zvuku po komplexní zařízení, od kterých je požadováno, aby jediný přístroj mohl sloužit k mnoha účelům. Velkým mezníkem se stal první mobilní přístroj s operačním systémem. Jedním z nejvýznamnějších operačních systémů se stal Android.

Nicméně komplexnost zařízení se nevztahuje pouze na účelnost. Snaha vývojářů se zaměřila i na vývoj a distribuci aplikací a na herní průmysl ve snaze, aby uživatel používal zařízení nejen k práci a účelným aktivitám, ale i k odpočinku a zábavě.

Cílem práce bylo vytvořit plně funkční samostatnou hru na motivy známé společenské hry s přihlédnutím na minimální požadavky na zařízení. Dále rozšířit aplikaci o nové nápady vyplývající ze zkušeností s hraním podobných společenských her a důrazem na svobodu uživatelů zakázáním striktních pravidel.

1 Hra Activity

Hra Activity je v dnešní době jednou z nejoblíbenějších společenských her. Základní verze je doporučena pro hráče od 12 let a je určena pro 3 – 16 hráčů. Z důvodu oblíbenosti hry vznikly rozšířené verze například pro děti. Hra na své popularitě získala především svou jednoduchostí a spoustou zábavy s ní spojenou.

1.1 Pravidla

Hra obsahuje hrací plán, karty, figurky a časomíru. Hráči se rozdělí do týmů a zvolí si barvu figurky, kterou postaví na startovní pole. Úkolem týmu je dostat svou figurku do cílového pole na druhé straně hracího plánu. Toho docílí správným hádáním zadaných pojmů. Jednotlivé týmy se střídají ve hře. Jeden hráč z týmu si vybere kartu dle obtížnosti. Karty jsou rozděleny na obtížnosti 3, 4 a 5. Po výběru karty ji otočí a přečte si pojem, který musí ostatní hráči daného týmu uhodnout. Na kartě je celkem 6 pojmů, které jsou barevně odlišeny podle tří typů úkonů, kde každý úkol má na kartě 2 pojmy. Nicméně i ty jsou od sebe barevně odlišeny. Podle typu a barvy pole, na kterém daný tým aktuálně stojí, musí být pojem předveden jedním ze tří způsobů:

- **mluvení** – pojem musí být vysvětlen bez vyslovení kořene slova. Při vysvětlování nesmí být použit žádný jiný způsob vysvětlení, například ukazováním na předměty.
- **kreslení** – pojem musí být vysvětlen kreslením. Nesmí být kresleny žádné znaky. Jediný povolený znak je počtem čar vyjádřen počet slov.
- **pantomima** – pomocí pantomimy je daný pojem předváděn. Při pantomimě je zakázáno vydávat zvuky.

Na uhodnutí pojmu má tým omezený čas, konkrétně 2,5 minuty, který je hlídán pomocí přesýpacích hodin. Pole „Start“ a „Cíl“ nemají barevné označení. Na poli „Start“ si hráč sám určí, který pojem ze šesti nabízených bude předvádět. Na poli „Cíl“ tuto volbu provedou protihráči. Pokud daný tým uhodne hádaný pojem v časovém limitu, posune se na hracím plánu o počet polí totožný s číslem na kartě, což znamená o 3, 4 nebo 5 polí. Jedinou výjimkou je situace, kdy by daný tým měl přeskočit pole „Cíl“. V tomto případě se tým posune právě na toto pole.

I přes velkou popularitu této hry jsou zde nevýhody omezující časté hraní této společenské hry. Jednou z hlavních nevýhod je, že je zde nepříliš velký počet karet. Z tohoto důvodu se hra rychle „ohraje“ a přestane bavit. Dalšími nevýhodami dále jsou: absence pomůcek pro kreslení, které jsou ne vždy a všude k dispozici, stejný neměnící se plán pro každou novou hru, text na kartách je malý, čímž je pro někoho nečitelný, a také fakt, že některé pojmy s nejvyšší obtížností jsou mnohem jednodušší než pojmy s nejnižší obtížností.

1.2 Rozšíření

Hra poskytuje velký potenciál a již vznikly další hry založené na stejném principu s drobnými úpravami. Vznikly jak deskové verze, tak elektronické verze pro zařízení desktopové i mobilní.

Hry poskytují jiné druhy úkolů, jiný způsob výběru polí a další drobné úpravy. I přes tyto změny daná hra často svazuje hráče nastavenými pravidly, která jsou s každou hrou naprosto stejná. Oproti deskovým hrám mají ty elektronické výhodu právě v možnostech dynamického nastavení hry a přizpůsobení se podmínkám. I přes to jsou současné hry často limitující a brzy se omrzí.

2 OS Android

Android je operační systém založený na open source[1] platformě a byl vytvořen společností Google. Je založen na verzích linuxových jader 2.6, která zajišťují zabezpečení systému, správu procesů, správu paměti a přístupy k ovladačům všech senzorů a k síti. Architektura neumožňuje přístup aplikací přímo k jádru, ale pouze skrze Android API [1], [2], [3], [4].

2.1 Vývoj aplikace

K vývoji aplikací pro operační systém android je nutné ovládat programovací jazyk Java. Vhodná je zde i znalost jazyku XML, kterým jsou definovány soubory sloužící pro zobrazování jednotlivých oken aplikace a jazyku SQL, pokud chce vývojář využívat vnitřní databázový systém SQLite. Při vytvoření projektu se automaticky vygeneruje adresářová struktura. Dokumenty napsané v programovacím jazyce Java se nachází v adresáři „src“ a v adresáři „res“ se nachází dokumenty, které se starají o vizuální stránku výsledné aplikace. Hlavními prvky každé aplikace v OS Android jsou „activity“.

„Activita“ reprezentuje základní vizuální komponentu, která zobrazuje jednu obrazovku aplikace. Aplikace se poté skládá z více „activit“, které jsou spolu provázané a vzájemně se volají. „Activita“ je definovaná v jazyce Java a při volání se spouští její metoda „onCreate“. V té se nejčastěji volá XML dokument, dle kterého se nastaví, co se má uživateli zobrazit. „Activita“ dále nejčastěji obsahuje metody, které obstarávají události například stisknutí tlačítka.

Nutnost znalosti XML závisí na vývojovém prostředí. Většina vývojových prostřední již umožňuje vytvářet XML soubory pomocí metody „Drag and Drop“ což znamená, že nám prostředí poskytuje náhled na konkrétní obrazovky a pouhým přesouváním komponent z palet na plochu automaticky generuje XML soubor. Po vytvoření základní struktury obrazovky se již stačí přepnout do kódu a upravit předdefinované informace.

Vizualizace aplikace se ovšem nemusí řešit pomocí XML souborů, ale je možné ji vytvořit přímo v „activitách“. Nemusí se ovšem jednat jen o sestavení vizualizace ze základních komponent, které se jinak tvoří pomocí XML, ale je možné grafickou podobu vytvořit například pomocí OpenGL ES, které umožňuje vykreslování 2D a 3D grafiky.

Po spuštění překladu se vygeneruje soubor „apk“, který slouží jako instalační balíček aplikace a je možné ho uložit do zařízení a aplikaci nainstalovat. Testování aplikace nabízí možnosti aplikaci spustit ve virtuálním přístroji nebo na reálném přístroji, který je propojený s vývojovým prostředím například pomocí kabelu USB. Virtuální přístroj pracuje pomaleji, ale nabízí vývojáři možnosti nastavení například velikost a rozlišení přístroje.

[1] Otevřený zdrojový kód. Reprezentuje snadnou licenční i technickou dostupnost.

2.2 Historie

Vývoj Androidu byl zaměřen pro zařízení PDA, mobilní telefony a tablety. Jedná se o jeden z nejmladších operačních systémů. Roku 2003 byla založena společnost Android, Inc. V roce 2005 byla společnost odkoupena společností Google a to včetně klíčových zaměstnanců. K prvnímu oficiálnímu zveřejnění softwarové platformy Android došlo roku 2007. Od prvního vydání OS Android prošel řadou změn. Originalitu Androidu potvrzují jména jednotlivých verzí, která představují jména jednotlivých zákusků seřazená podle abecedy.

První oficiální verzí byla verze Android 1.0. Tato verze již podporovala například webový prohlížeč, fotoaparát a služby k propojení s uživatelským účtem Google. Tato verze ještě neměla přívlastek. První verzí Androidu s přívlastkem byl Android 1.5 (Cupcake). Postupně docházelo k rozšiřování možností zařízení a opravování vzniklých chyb po v současné době poslední verzi Android 4.4 (KitKat). Tato verze přinesla několik novinek například klávesnici rozšířenou o sadu 153 smajlíků nebo upravený „launcher“.

2.3 Architektura

Architektura OS Android se skládá z pěti základních vrstev. Každá vrstva provádí jiné operace a vystupuje víceméně samostatně. V praxi však dochází ke spolupráci jednotlivých částí a vrstvy tímto nejsou mezi sebou striktně odděleny. Architektura operačního systému Android je znázorněna na Obrázek 1.

2.3.1 Linux Kernel

Nejnižší vrstva představuje jádro operačního systému. Slouží k implementaci abstrakce mezi softwarem ve vyšších vrstvách a hardwarem daného zařízení. Při startu zařízení je jádro zavedeno do operační paměti a dostává pravomoc nad řízením, kontrolou a koordinací systému a všech běžících procesů.

2.3.2 Libraries

Android poskytuje celou řadu knihoven sloužící pro vývoj aplikací. Mezi nejpoužívanější knihovny lze zařadit například knihovnu *android.util* obsahující nízkoúrovňové třídy pro specializované kontejnery, nástroje pro parsování XML atd. Dále například *android.graphics*, *android.text*, *android.database* a další často využívané knihovny.

2.3.3 Android Runtime

Vrstva obsahující DVM[2] a základní Java knihovny. Slouží ke koordinaci běžících procesů, správě paměti nebo práci s vlákny. Vývoj DVM byl zaměřen na úsporu energie mobilních zařízení. Knihovny programovacího jazyka Java lze srovnávat s platformou Java SE (Standard Edition).

2.3.4 Application Framework

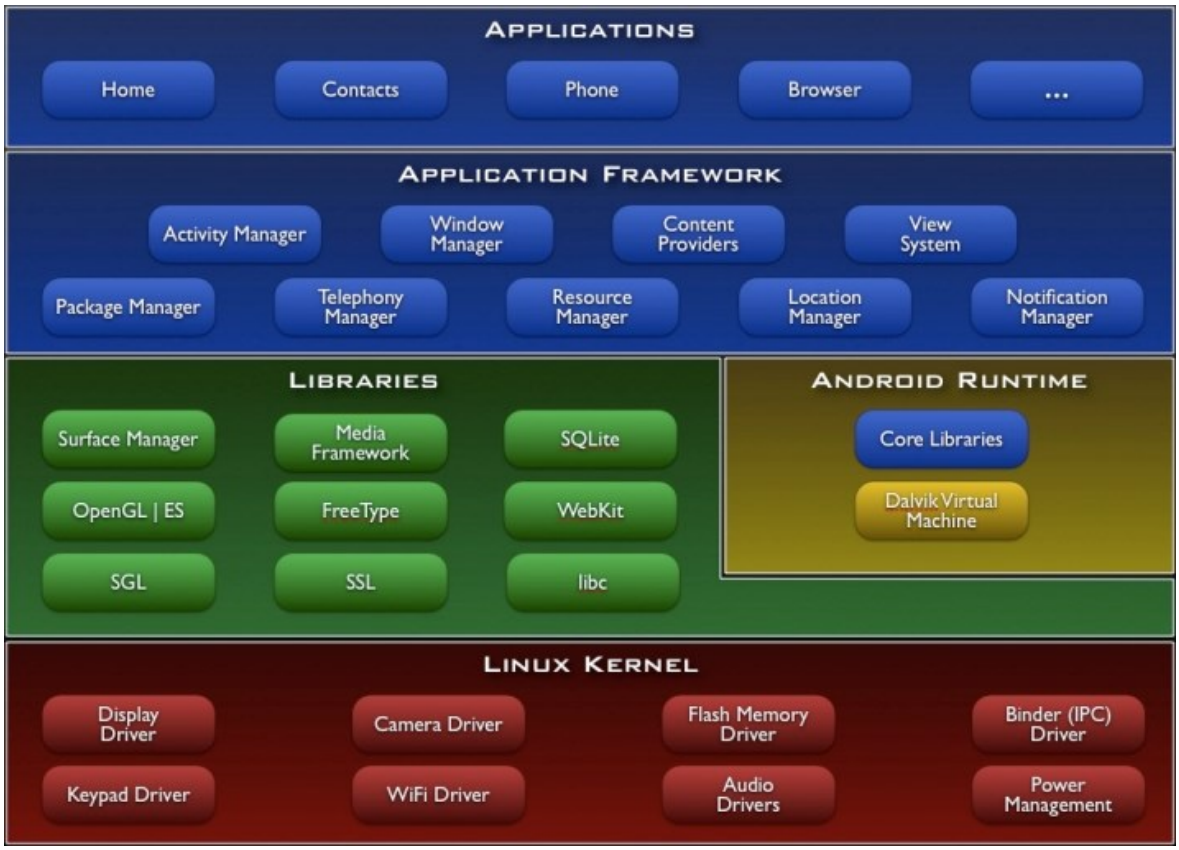
Tato vrstva je nejdůležitější vrstvou pro vývojáře. Android programátorům nabízí prostředí pro vývoj široké škály aplikací. Tento aplikační rámec umožňuje vývojářům využívat

[2] Dalvik Virtual Machine

nejrůznější služby přímo ve svých aplikacích. Příkladem může být používání hardware zařízení nebo spouštění jiných aplikací na pozadí.

2.3.5 Application

Nejvyšší vrstva slouží vlastním aplikacím, které jsou používány uživateli. Některé aplikace jsou již předinstalovány v zařízení, jiné uživatel může stáhnout a nainstalovat ručně nebo automaticky z online stránek, například z play.google.com.



Obrázek 1 – architektura OS Android [5]

2.4 Vývojové nástroje

Vývoj aplikací pro Android probíhá stejně jako vývoj aplikací pro veškerá mobilní zařízení ve vývojovém prostředí „Host-Target“. Tím je myšleno, že vývoj aplikace probíhá na počítači, který obsahuje vývojové prostředí a samotné spouštění a testování probíhá na mobilním zařízení. Testování může probíhat i v tzv. emulátoru, který simuluje reálné zařízení. Programování aplikací pro OS Android využívá nástroje:

- **Java Development Kit (JDK)** – soubor základních nástrojů a knihoven sloužící pro vývoj aplikací pro platformu Java.
- **Android Software Development Kit (SDK)** – balíček vývojových nástrojů, díky kterému je možné vytvářet aplikace pro konkrétní operační systémy či herní konzole. SDK obsahuje knihovny API, ukázky použití, dokumentaci atd. Dále se

zde nachází knihovny Javy a například také emulátor. Pro přehlednou správu SDK se používá „SDK manager“, ve kterém se určí instalované komponenty.

- **vývojové prostředí Eclipse** – jedním z nejrozšířenějších vývojových prostředí pro vývoj aplikací je Eclipse. Je určen primárně pro programování aplikací pomocí jazyku Java. Umožňuje rozšíření pro podporu dalších jazyků a poskytuje vizuální nástroj pro graficko-uživatelské rozhraní. Vše lze jednoduše získat instalací příslušných pluginů. Nejznámějšími alternativami jsou „IntelliJ IDEA“ a nyní nově vznikající „Android Studio“. Každé vývojové prostředí má své výhody a každému programátorovi bude vyhovovat něco jiného, proto nelze jednoznačně určit, které prostředí je nejlepší.

- **Android Development Tool (ADT)** – plugin starající se o propojení „Eclipse“ a „Android SDK“. Zaručuje výkonné integrované prostředí s editorem vizuálních aplikací, ladícími panely, tvorbou balíčků APK aj.

3 SQLite

SQLite je databázový systém, který je určený nejen pro OS Android. První verze vyšla roku 2000. Systém je využíván pro jeho rychlost a šetrnost k paměti. Je tvořen jedinou knihovnou, což je velmi praktické oproti často používaným způsobům klient-server. Je volně šiřitelný pod licencí „public domain“, což umožňuje používání systému bez jakékoli platby. SQLite využívá jazyk SQL podle standardu SQL-92. Jisté odlišnosti se zde však nachází. Oproti standardu SQL je zde hlavní problematika s neexistencí kontroly datových typů. Systém dovoluje například vkládání datového typu „integer“ do sloupce, který je typu „string“ [1], [6].

3.1 Vytvoření databáze

Pro vytváření databáze se nejčastěji využívá třída *SQLiteOpenHelper*, která se stará o veškerou logiku pro vytváření a aktualizaci databáze. Mimo tuto třídu SQLite nemá žádné předpřipravené databáze a je nutné vše vytvořit. *SQLiteOpenHelper* nabízí následující nejdůležitější metody:

- ***onCreate*** – je metoda pro vytváření a naplnění tabulek v databázi. Pracuje se zde s objektem *SQliteDatabase*.
- ***onUpgrade*** – je určena pro změnu struktury databáze a opět je zde využíván objekt *SQLiteDatabase*. Mimo tento objekt jsou zde využity dvě celočíselné proměnné označující číslo původní verze a nové verze.

Konstruktor zde slouží k tvorbě databáze a obsahuje parametry: objekt typu *Context*, název databáze, instanci třídy a číselnou proměnnou představující verzi databáze. Databázi lze vytvořit k využívání pro čtení nebo pro zápis.

3.2 Práce s tabulkami

Nejčastěji se tabulky databáze vytváří hned po vytvoření databáze. Je možné je vytvářet přímo v metodě *onCreate*. Dále je možné tabulky měnit a mazat. Příkazy pro práci s tabulkami jsou obdobné. Pro práci s tabulkami se využívá metoda *execSQL*. Příklad užití metody *execSQL*:

```
db.execSQL("CREATE TABLE karty (
    id INTEGER PRIMARY KEY,
    text_pojmu TEXT NOT NULL UNIQUE,
    hodnota INTEGER NOT NULL,
    typ_ulohy TEXT NOT NULL,
    sada TEXT NOT NULL,
    pouzita TEXT NOT NULL);");
```

3.3 Práce s daty

Databáze jsou vytvářeny pro strukturované ukládání dat a na jednoduchou práci s nimi. SQLite nabízí práci s daty pomocí jednoduchých metod.

3.3.1 Vyhledávání

Jednou z nejdůležitější části využívání databází je vyhledávání dat (Select). Systém SQLite nabízí tři metody, jak vyhledávání vytvořit: *rawQuery*, *query* a *SQLiteQueryBuilder*. Na příkladu metody *query* je vytvořené vyhledávání karet typu „mluvení“ ze sady „základní“, které ještě nebyly zahrány.

```
Cursor cursor = db.query(TABLE_CARDS, new String[] { KEY_ID, KEY_TEXT,
KEY_VALUE, KEY_TYPE, KEY_SET, KEY_USED },KEY_TYPE + " LIKE 'mluvení' AND
" + KEY_SET + " IN základní AND "+ KEY_USED + " LIKE 'false'",null, null,
null,"random()", "1");
```

Důležitou částí vyhledávání položek databáze jsou tzv. kurzory. Výsledek „selectu“ vrací instanci třídy *Cursor*. Tento objekt je třeba dále zpracovat metodami třídy. V daném objektu se metodami prochází mezi záznamy a čtou se získaná data.

3.3.2 Vkládání dat

SQLite nabízí dvě možnosti vkládání dat do databáze. K tomuto slouží metody *execSQL* a *insert*. Metoda *execSQL* opět pouze interpretuje vložený příkaz, metoda *insert* pak pracuje se třídou *ContentValues*. Do té se uloží jednotlivé údaje a příkazem *insert* se vloží do databáze. Vlastní metoda *insert* má tři parametry, přičemž nejdůležitějšími jsou: název tabulky, do které se bude řádek vkládat, a vlastní vytvoření *ContentValues*, ve kterém jsou uloženy vkládané údaje.

3.3.3 Změna dat

Pro změnu dat je možné opět využít metody *execSQL* nebo *update*. Metoda *update* též využívá třídy *ContentValues*. Využití metod pro změnu dat je výhodnější než data mazat a znovu je vytvářet pozměněná.

3.3.4 Mazání dat

Mazání slouží k odstranění dané položky z databáze. Pro tuto akci je možné využít metodu *delete* nebo je opět možné napsat příkaz přímo do metody *execSQL*. U mazání dat je důležité ošetřit, zda na daný prvek není jinde vytvořený odkaz.

4 XML technologie

XML bylo vytvořeno pro jednoduché tvoření vlastních formátů a díky své obecnosti je široce využíván pro mnoho odvětví informačních technologií. Nejvíce je používán pro vytváření webových stránek, definici vizualizace v operačním systému Android a v současnosti je jedním z nejdůležitějších formátů pro výměnu dat. XML technologie lze využít jako komunikační rozhraní pro prezentaci dat. Umožňuje definovat vlastní datové typy, které nejčastěji vznikají odvozením od již existujících datových typů. [7], [8], [9].

XML (eXtensible Markup Language) je překládáno jako rozšiřitelný značkovací jazyk. Pojem „rozšiřitelný“ označuje standard XML. Ten popisuje, jak má vypadat syntaxe XML dokumentu, přičemž záleží na samotném autorovi, jaké značky (tagy) zvolí pro daný dokument a jak bude dokument zpracován. „Značkovací“ označuje, že XML vznikl zjednodušením standardů SGML, aby bylo zjednodušeno vytváření a zpracovávání dokumentů a pojem „jazyk“ označuje, že formát dokumentů XML je možné prohlížet a editovat v libovolném editoru.

Dokumenty XML musí splňovat několik následujících pravidel:

- každý element (tag) musí být párový a začínat ve formátu *<jmeno_elementu>* a končit ve formátu *</jmeno_elementu>*,
- jednotlivé elementy se nesmí vzájemně překrývat, což znamená, že před ukončením elementu musí být ukončeny všechny elementy, které byly vytvořeny uvnitř tohoto elementu,
- počáteční element může obsahovat atributy,
- znakům elementů, které mají speciální význam, musí předcházet znak „\“,
- celý dokument musí obsahovat pouze jeden kořenový element (root element) a všechny ostatní elementy musí být uvnitř tohoto kořenového elementu.

4.1 Výhody XML

Jednou z hlavních výhod XML je jazyková podpora. Jedná se o jeden z prvních formátů, který nebyl zaměřen pouze na anglický jazyk, ale využívá jazykovou sadu „ISO 10646“, která je schopna pojmout znaky všech jazyků.

XML pomocí svých značek (tagů) jednoznačně a přehledně označuje význam částí textu. S tímto je spojena snadná konverze do jiných formátů, neboť XML je univerzální jazyk a většina programů, které zpracovávají data, umí importovat i exportovat právě do XML. Současně jediný styl můžeme aplikovat na velké množství dokumentů stejného typu, čímž lze dosáhnout stejného formátování. Příkladem může být definování tlačítka. Vytvořený dokument je využit současně pro všechna tlačítka, čímž je zaručeno stejné chování v celém modelu. Další velkou výhodou je potom automatická kontrola struktury dokumentů, která zaručuje korektnost vložených dat v dokumentech.

4.2 XML schema

Schema v XML slouží k definici a validaci XML souborů. Pomocí nich se definují jednotlivé datové typy a jejich použití zaručuje správnost dokumentů v závislosti na konkretizování požadavků ve schematu. Příkladem může být následující schema, které udává strukturu dokumentů zaměstnanců. Schema udává, že se bude týkat elementu „zaměstnanec“ a dále definuje čtyři argumenty. „Jméno“ a „příjmení“ typu textového řetězce, „plat“ typu desetinného čísla a identifikační číslo „id“ jako celočíselné číslo. *Sequence* udává, že elementy uvnitř tohoto elementu se musí vyskytovat přesně v tomto pořadí.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="zamestnanec">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="jmeno" type="xs:string"/>
        <xs:element name="prijmeni" type="xs:string"/>
        <xs:element name="plat" type="xs:decimal"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4.3 Připojení schematu k dokumentu

Dřívější aplikace podporovaly pouze způsob napojování pomocí atributů, novější aplikace už však podporují novou instrukci *<?xml-model?>*, která umožňuje připojování schemat bez používání speciálních atributů:

```
<?xml-model href="dokument.xsd"?>
<dokument>
…
</dokument>
```

Dřívější atributy *schemaLocation* a *noNamespaceSchemaLocation* se již nedoporučují používat.

4.4 Základní datové typy

XML schema obsahuje základní datové typy: řetězec, celá čísla, desetinná čísla, binární čísla, logické hodnoty, datum, čas a některé datové typy převzaté z DTD[3]. Ve většině případů jsou tyto datové typy plně dostačující. Pokud je ovšem vyžadována striktnější kontrola dat, je nutné od základních zabudovaných datových typů odvodit vlastní.

Nejčastějším vytvořením vlastního datového typu je tzv. restrikce, kdy je na datový typ aplikováno omezení. Jedním z nejjednodušších omezení je *length*, které umožní definici přesné délky řetězce. Je nutné použít klíčová slova *simpleType* pro definování nového

[3] Jazyk pro popis struktury XML

datového typu, dále *restriction* určující datový typ, od kterého je nový datový typ odvozen, a dále vlastní omezovací elementy. Nově definované typy se při vytváření elementů používají stejně jako u základních typů. Výsledný vlastní datový typ, ve kterém dochází k omezení délky řetězce mezi pět a deset znaků, může vypadat následovně:

```
<xs:simpleType name="jménoType">
  <xs:restriction base="xs:string">
    <xs:minLength value="5"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
```

Stejným způsobem je možné definovat omezování číselných typů. Je možné definovat nejen minimální a maximální hodnotu čísla, ale například i celkový počet platných číslic nebo počet míst za desetinnou čárkou. XML umí též pracovat s regulárními výrazy, které mohou pomoci pro jednoduchý zápis restrikce.

4.5 Komplexní datové typy

Mimo základní datové typy XML poskytuje i komplexní datové typy. Ty slouží k modelování struktury dokumentů. To je dáno tím, že se mohou skládat z elementů a atributů. Elementům lze například určit, kolikrát se mohou opakovat nebo v jakém pořadí. K definici typu se použije element *completType* a podobně, jako u jednoduchých typů, lze tento typ vytvořit samostatně, aby mohl být používán opakovaně. Příkladem může být případ, kdy nezáleží na pořadí elementů. Zde se využije kompozitura *all*, která ovšem limituje počet výskytů od nuly do jedné:

```
<xs:element name="osoba">
  <xs:complexType>
    <xs:all>
      <xs:element name="jméno" type="xs:string"/>
      <xs:element name="příjmení" type="xs:string"/>
      <xs:element name="titul" type="xs:string" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Takovéto schema potom umožní vytváření souborů, které mají velmi laxní přístup. To z toho důvodu, že není nastavena povinnost účasti všech informací v daném elementu a údaje mohou být v libovolném pořadí, což z hlediska hlídání správnosti dokumentu není příliš vhodné. Takto definovaný dokument pak může vypadat například takto:

```
<osoba>
  <jméno>Jan</jméno>
  <příjmení>Novák</příjmení>
</osoba>
<osoba>
<titul>Ing</titul>
  <příjmení>Nováková</příjmení>
  <jméno>Jana</jméno>
</osoba>
```

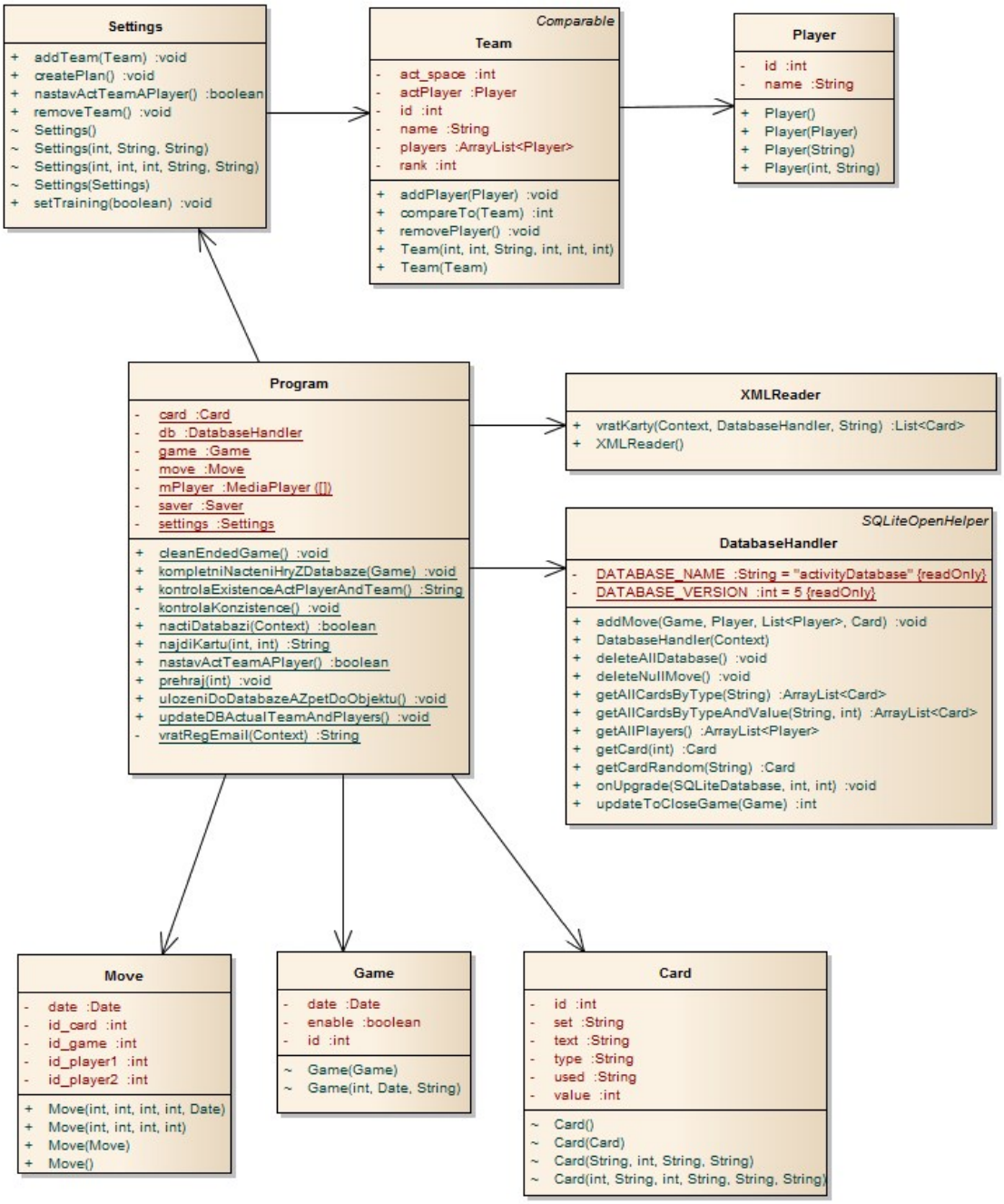
5 Implementace aplikace

Aplikace je implementována pomocí deseti tříd, které zpracovávají vlastní logiku programu, 29 aktivit starajících se o zobrazování aplikace a jedné třídy umožňující hráčům kreslit. Vývoj byl zaměřen na přehlednost a kompatibilitu i se staršími verzemi OS Android od API 9, která představuje verzi OS Android 2.3. Dalším důležitým aspektem vývoje byla minimální náročnost na systém.

5.1 Návrh aplikace

Aplikace pracuje v jednotlivých třídách pojmenovaných jako *activity*. *Activita* načte příslušný XML soubor, ve kterém je definováno, co se má uživateli zobrazit. Uživateli jsou poskytnuta tlačítka (buttons) pro ovládání aplikace. Po stisknutí tlačítka *activita* zareaguje, zpracuje požadavek a popřípadě zavolá jinou *activitu*. *Activity* si mohou předávat pouze omezené typy informací. Jsou jimi buď primitivní datové typy, nebo objekty, ovšem pouze objekty typu *Serializable*. Z tohoto důvodu je zde vytvořen logický modul aplikace hlavní statickou třídou *program*, kterou jednotlivé *activity* volají pro zpracování dané úlohy. Na Obrázek 2 jsou zobrazeny ořezané třídy a vazby mezi nimi. Úplný UML diagram je v Příloze A. Pro aplikaci byly vytvořeny následující třídy:

- ***program*** – třída vytváří páteř logiky programu, neboť propojuje jednotlivé části programu,
- ***databaseHandler*** – třída rozšiřující *SQLiteOpenHelper*, která pracuje s interní databází v zařízení,
- ***XMLReader*** – zodpovídá za aktualizace nových karet při spuštění aplikace,
- ***game*** – třída obstarávající základní informace o aktuálně načtené hře,
- ***settings*** – třída pro uložení stavu nastavení aktuální hry a týmů, které se dané hry účastní, včetně aktuálního týmu na tahu,
- ***team*** – uchovává informace o týmech a hráčích v daných týmech a aktuálních hráčích v týmech, pokud je tak nastaveno,
- ***player*** – obsahuje základní informace o hráčích,
- ***card*** – zpracovává údaje o jednotlivých kartách,
- ***move*** – pomocná třída pro vytvoření zahraného tahu pro uložení do databáze.



Obrázek 2 – ořezaný UML diagram

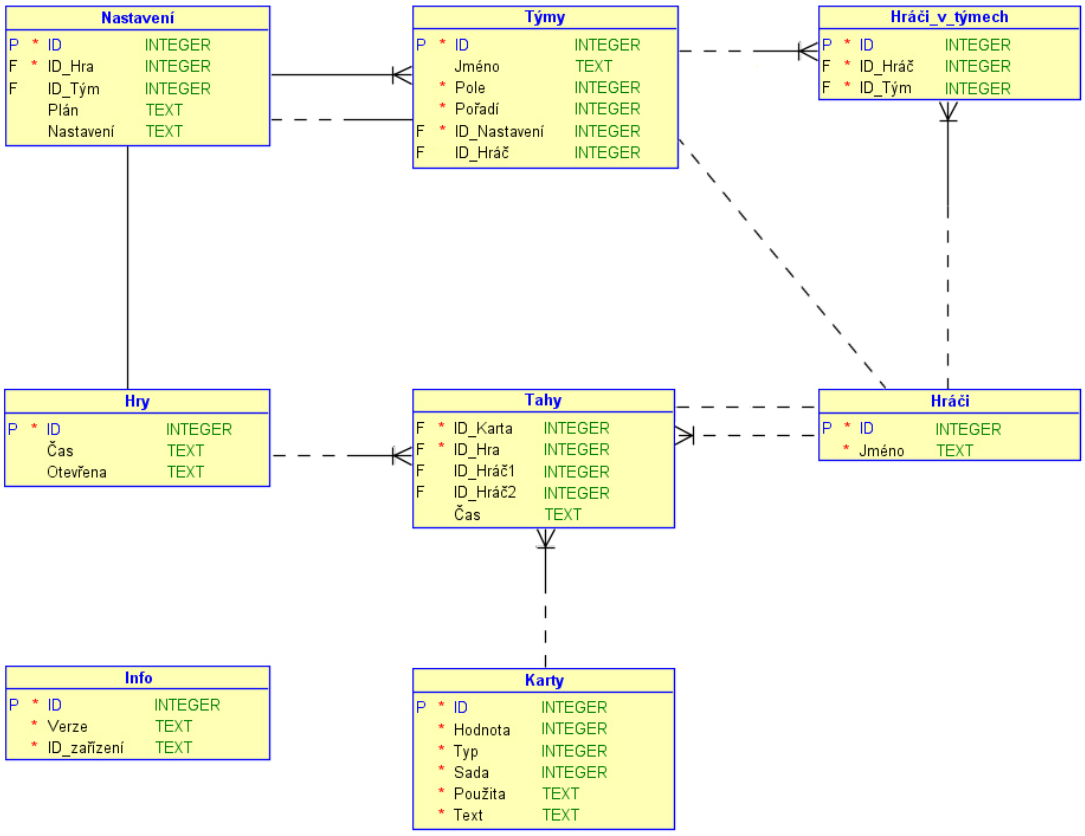
5.2 Struktura databáze

Aplikace využívá možností OS Android na vytvoření vnitřní databáze v zařízení. Databáze je navržena pro ukládání základních informací o jednotlivých hrách viz Obrázek 3. Databázová struktura umožňuje téměř kdykoli vypnout aplikaci bez ztráty dat. O práci s databází se stará třída *databaseHandler*. Tato třída je tvořena množstvím metod obstarávajících veškerý přístup do databáze. V Příloze B je uveden příklad tří metod.

Metoda *addPlayer* v parametru přijímá objekt hráče. Do vytvořeného objektu *values* uloží jméno hráče a pokusí se ji vložit do databáze pomocí metody *insertOrThrow*. Z důvodu toho, že jméno hráče musí být unikátní, je zde možnost výjimky, která vytvoří chybový výpis o již existujícím kontaktu.

Příkladem druhé metody je metoda *getCardRandom*, která obstarává vybrání náhodné karty z databáze specifikované pomocí dvou argumentů. Zde se vytvoří pomocný objekt *cursor* do kterého je uložen výsledek příkazu výběru z databáze. Tento výběr obstarává metoda *query* v jejímž příkazu jsou v tomto případě nejdůležitější poslední dva argumenty označující, že se jedná od náhodný výběr a že požadujeme právě jednu hodnotu. Následně se vytvoří pomocný objekt *card*, do kterého se uloží informace o získané kartě, a tento objekt se vrací.

Posledním příkladem je metoda *updateToCloseGame*, která obstará označení ukončené hry její změnou v databázi. Opět je zde využit objekt *values*, do kterého je vložen nejdůležitější parametr „false“ označující ukončené hry. Poté se volá metoda *update* pro aktualizování daného řádku v databázi.



Obrázek 3 – ER diagram

5.2.1 Tabulky

Databáze je rozdělena do osmi tabulek. Jedinou samostatnou tabulkou je tabulka „Info“, ostatní tabulky jsou provázány vazbami. Tabulky databáze jsou následující:

- **„Info“** – tabulka slouží pouze pro uchovávání informací o verzi aktuální použité databázi karet. Využívá se při kontrole aktuálnosti databáze, kdy tento údaj je kontrolován s údajem o verzi na Internetu. Druhým údajem v tabulce je „ID_zařízení“. Ten slouží k zabezpečení databáze.

- **„Hry“** – tabulka uchovává informace o času vytvoření a záznam, je-li daná hra otevřena pro hraní nebo se jedná o uzavřenou hru. S uzavřenými hrami uživatel již nemá možnost jakkoli pracovat, avšak slouží k výpočtům statistických údajů. Aktivní hru je možné načíst a pokračovat ve hraní, nebo hru smazat. Uživateli se jednotlivé rozehrané hry odlišují časovým údajem vzniku hry. Při smazání uživatelem nedochází ke skutečnému mazání hry, nýbrž jen k jejímu převedení do stavu ukončené hry. Uživateli tímto hra zmizí ze seznamu, ale do statistik se stále počítá.

- **„Nastavení“** – pomocná tabulka patřící k tabulce „Hry“. Obsahuje hrací plán, který je uložen v textovém řetězci jako posloupnost čísel symbolizující typ jednotlivých polí. Stejným způsobem je uloženo úplné nastavení hry. Tyto údaje jsou pro celou hru neměnné. Jeden „cizí klíč“ slouží k napojení na tabulku „Hry“. Druhý „cizí klíč“ udává odkaz na tým, který je aktuálně na tahu. Položky této tabulky jsou dočasné do chvíle, kdy se daná hra ukončí. Po této události nemá položka pro program další význam.

- **„Týmy“** – tabulka sloužící k uchovávání informací o týmech. Tabulka obsahuje sloupec pro jméno týmu, číslo pole, na kterém se tým nachází, a informaci o pořadí, aby aplikace po načtení z databáze rozeznala pořadí týmů v cíli, pokud některé týmy již cíle dosáhly a přesto se hra bude dohrávat. Dále jsou zde obsaženy dva „cizí klíče“. První pro navázání s tabulkou „Nastavení“ a druhý nepovinný slouží k uchování odkazu na hráče, který je v daném týmu aktuálně na tahu.

- **„Hráči“** – tabulka obsahující informace o jméně hráče. Toto jméno musí být unikátní, aby při hře nedocházelo k záměně. Jména lze během hry i mimo hru vytvářet a jsou v databázi uložena do doby, než je sám uživatel smaže. Mimo spuštěnou hru je možné tyto hráče přejmenovávat bez hrozby zásahu do her, kterých se hráč aktuálně účastní. Smazání hráče je povoleno pouze v případě, že se neúčastní žádné rozehrané hry, aby nedocházelo ke kolizním situacím. Pokud je uživatel smazán, smaže se jeho historie her, čímž přestane být započítáván do statistik.

- **„Hráči_v_týmech“** – tabulka vytvořena původní N*N[4] vazbou spojující tabulky „Hráči“ a „Týmy“.

- **„Karty“** – tabulka uchovávající informace o jednotlivých kartách. Tabulka obsahuje sloupce pro definici karet (hodnota, typ, text) a identifikátor použití. Sloupec „hodnota“ udává obtížnost dané karty, „typ“ o jaký typ úkonu se bude

[4] Vazba značící více výskytů k více výskytům.

jednat a „text“ obsahuje samotný zadaný pojem. Zde přidaný sloupec „sada“ je připraven pro budoucí rozšíření jednotlivých sad, aby bylo možné hrát například tematické hry nebo hry určený pro děti apod. Poslední sloupec „použita“ slouží k rozeznání karet, které už byly zahrány nikoli v rámci hry, ale v rámci celé aplikace, čímž je vyřešen problém často se opakujících pojmů k hádání.

- **„Tahy“** – propojující tabulka slouží pouze k ukládání dat pro statistické účely. V této tabulce je uloženo, v jaké hře došlo k tahu, jaká karta byla zahrána, čas provedení, kdo zadaný pojem předváděl a případně hráči, kteří tento pojem uhádli.

Her se mohou účastnit i hráči, kteří si nezaloží jméno, ale využijí dočasného jména „HostPlayer“. Po uzavření hry jsou tito hráči smazání z databáze, neboť pro statistiky nemají žádný význam a současně s nimi se smažou i řádky tabulky „Tahy“, kde se vyskytovali. Vše z důvodu snahy o minimální velikost databáze, aby v ní nebyly zbytečně nepotřebné informace.

5.2.2 Relace

Databázový model obsahuje deset relací mezi tabulkami. Tabulky „Hry“ a „Nastavení“ spojuje relace 1:1, neboť každé nastavení hry se váže právě na jednu hru a naopak. Tabulky „Nastavení“ a „Týmy“ propojují dvě relace. Relace 1:N určuje, že více týmů je napojeno právě na jedno nastavení. Druhá relace 1:1 ukládá informace o aktuálním týmu na tahu.

Tabulky „Týmy“ a „Hráči“ propojují dvě relace. První relací je relace N*N, neboť hráč se může vyskytnout ve více týmech a současně se v týmu může nacházet více hráčů. Pro tuto relaci vznikla pomocná propojující tabulka „Hráči_v_týmu“. Druhá relace slouží k uložení informace o aktuálním hráči v týmu.

Poslední skupina relací se týká tabulky „Tahy“, do které se ukládají informace o jednotlivých tazích. Relace k tabulkám „Karty“ a „Hry“ určují, že údaje odkaz na hru a kartu je povinný. Poslední dvě relace k tabulce „Hráči“ ukládají informace, který hráč úkon předváděl a kteří hráči pojem uhádli. Z důvodu toho, že může být smazán hráč „HostPlayer“, jsou tyto údaje nepovinné, stejně jako z důvodu toho, že aplikace ukládá informace i o neúspěšných tazích.

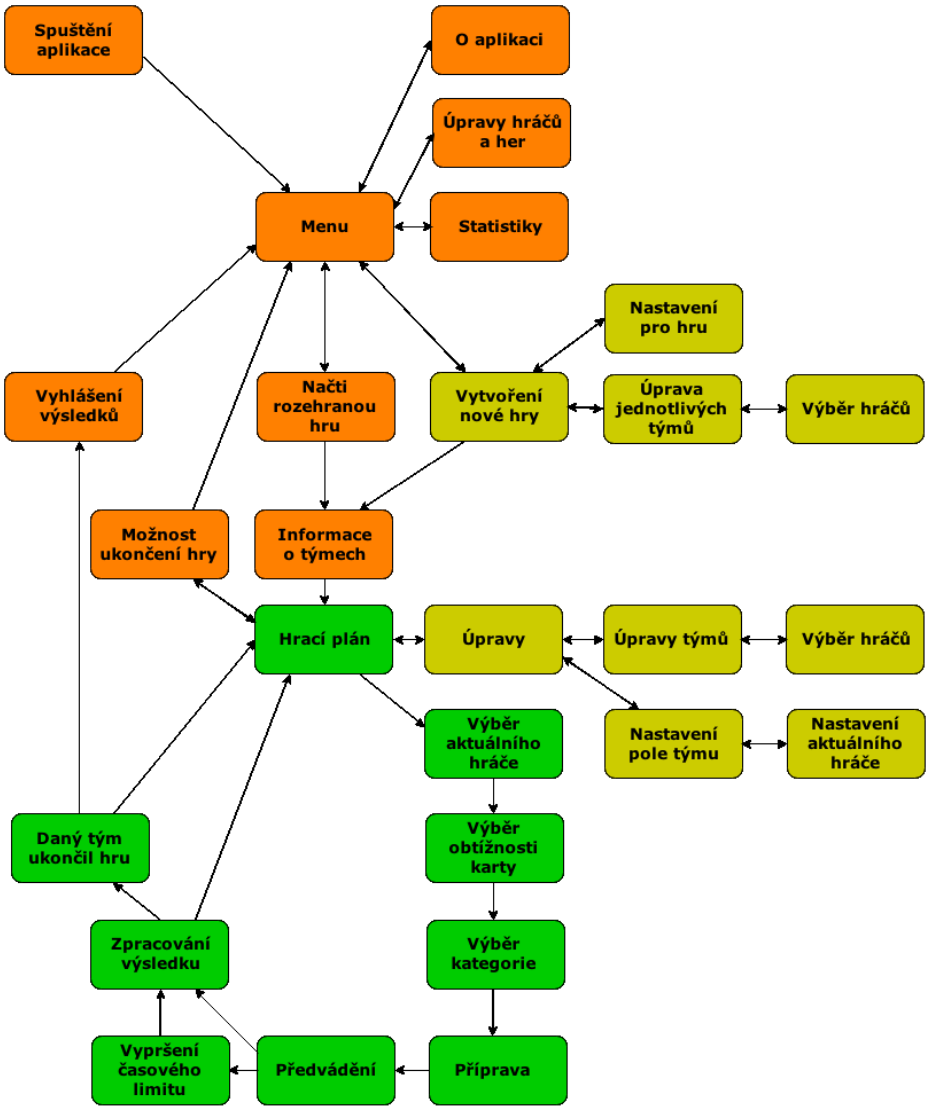
5.3 Rozšíření pravidel

Aplikace vychází ze základních pravidel a umožňuje hrát až sedmi týmům po sedmi hráčích. K základní koncepci hry byla přidána nová kategorie úkolů, konkrétně úkoly pojmenované „Ano/Ne“. Princip tohoto úkolu je následující: hráč obdrží pojem a během časového limitu na uhodnutí pojmu se spoluhráči ptají na hádaný pojem. Hráč držící kartu může používat pouze odpovědi „ano“ a „ne“, čímž spoluhráče musí navést na hádaný pojem.

5.4 Ovládání aplikace

Při vývoji aplikace byla snaha o intuitivní ovládání. Pohyb v aplikaci je znázorněn na Obrázek 4. Z tohoto pohledu se aplikace skládá z tří základních částí:

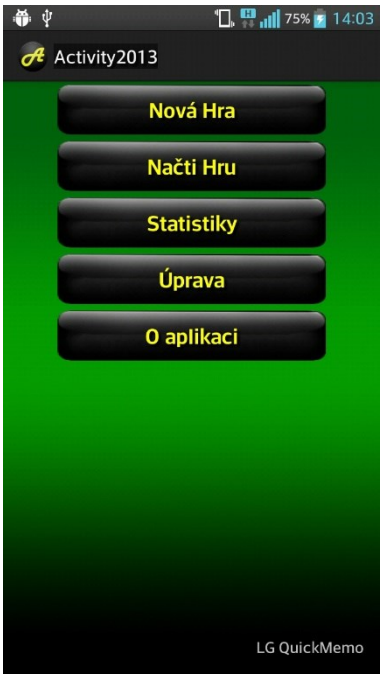
- *oranžovou* barvou je znázorněn pohyb po menu, které tvoří okolí vlastní hry a slouží spíše jako obal vlastní hry. Po spuštění aplikace se uživateli zobrazí menu, které slouží jako hlavní rozcestník aplikace,
- *žlutá* barva znázorňuje možnosti založení nové hry a dále nastavení ať už při založení hry nebo i nastavení v průběhu hry,
- *zelenou* barvou je označena nejdůležitější část aplikace, a to konkrétně životní cyklus každé hry. Ten se může lišit v závislosti na nastavení aplikace. Například pokud je hra nastavena tak, že se nebude hrát na obtížnost karet, aplikace se dle tohoto nastavení zachová a nedá hráči možnost výběru ze tří obtížností, ale toto výběrové okno přeskočí.



Obrázek 4 – diagram pohybu v aplikaci

Aplikace má přednastavená tlačítka pro pohyb mezi jednotlivými okny. Vypínání aplikace je řešeno pomocí správce aplikací v OS Android. Po spuštění se uživateli zobrazí základní menu, jak lze vidět na Obrázek 5. Zde je výběr pohybu na položky:

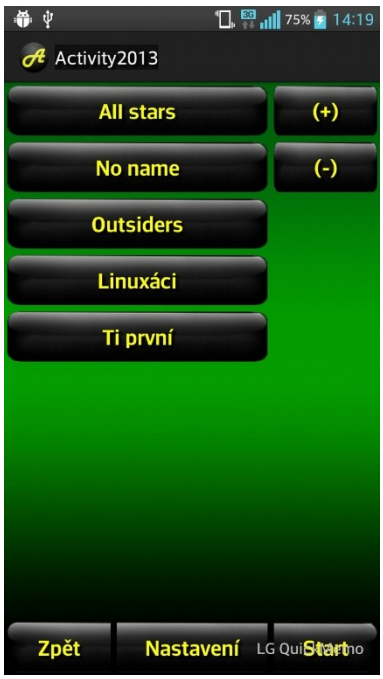
- **nová hra** – je načtena nová plocha s možnostmi nastavení hry, výběrem počtu týmů a po otevření týmu možnost změny jména týmu a nastavení počtu a jmen hráčů v týmu. Po ukončení nastavení lze tlačítkem „Start“ spustit nastavenou hru nebo se kdykoli vrátit do menu tlačítkem „Zpět“.
- **načti hru** – nabídne rozehrané hry, které lze výběrem spustit, nebo se vrátit do menu. Jednotlivé rozehrané hry jsou pojmenovány podle data vytvoření.
- **statistiky** – zobrazí okno statistik zúčastněných hráčů. Návrat je možný tlačítkem „Zpět“.
- **úpravy** – umožní modifikaci rozehraných her a hráčů. Návrat do menu je opět pomocí tlačítka „Zpět“.



Obrázek 5 – menu

- **o aplikaci** – zobrazí základní informace o aplikaci a pravidla hry.

Na Obrázek 6 je vytváření nové hry. Lze nastavit jeden až sedm týmů po jednom až sedmi hráčích. Týmy je možné pojmenovat a hráče nastavit jednoduše pomocí jmen uložených v databázi z předchozích her nebo si vytvořit nové jméno, pokud daný hráč hraje poprvé. Tím se jeho jméno uloží do databáze. Automaticky zde probíhá kontrola, aby hráč nemohl zadat jméno, které už je uložené v databázi, a taktéž aby se více týmů nemohlo jmenovat stejně. Aplikace se i zde snažila hráče příliš nelimitovat, proto je možné nastavit délku jména do dvanácti znaků, stejně tak i jména týmů. Současně však aplikace nezakazuje, aby jeden hráč hrál ve více týmech. Je zde možnost, aby hráč mohl hrát i sám a trénoval tak své dovednosti. Tento typ hry je zde označován jako trénink. Umožněno je zde i to, aby v jednom týmu bylo více hráčů a v jiném jen jeden. Není to pro hraní této hry vhodné, nicméně je to záležitost dohody hráčů a aplikace je v tomto neomezuje. Dalším kompromisem mezi jednoduchostí a ponecháním volnosti hráčům je skutečnost, že aplikace nabízí množství nastavení hry, kterému je věnována následující kapitola.



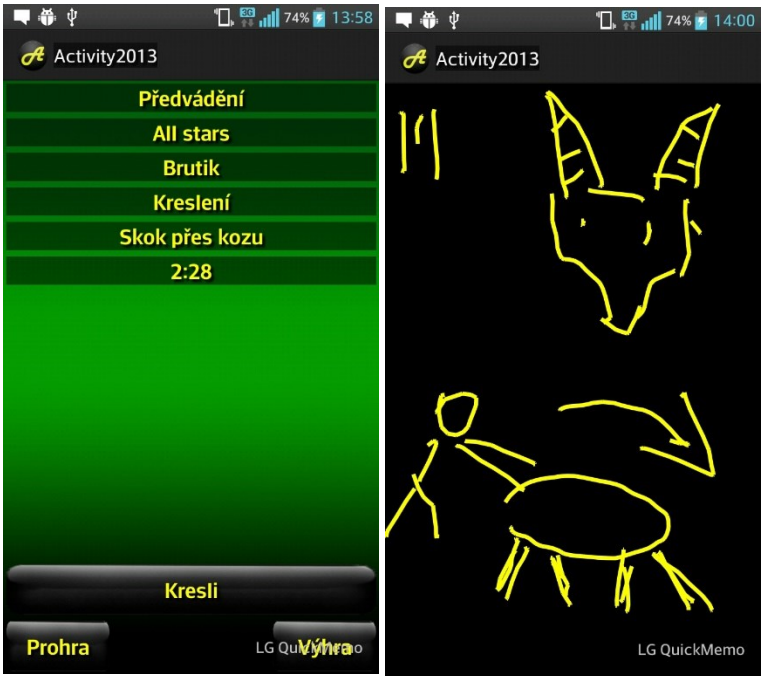
Obrázek 6 – nová hra

Po spuštění nové hry nebo načtení rozehrané hry se zobrazí nejprve informace o týmech a hráčích a po potvrzení hrací plán, kterým lze posouvat. Je barevně odlišený dle typu úkonu a jsou zde vypsána jména týmů na polích, na kterých se nachází. Příklad hracího plánu je možné vidět na Obrázek 7. Ve spodní části je poté napsán aktuální tým a hráč na tahu, který si má vzít zařízení. Hrací plán je vytvořen pomocí třídy *scrollview* a *tableview*. Díky tomuto lze plán „scrollovat" a prohlížet si tak celý hrací plán. Na obrázku lze vidět, že velikost polí se mění podle počtu týmů, které na daném poli stojí. V Příloze C se nachází část definice zobrazení hracího plánu. Při vstupu na plán se hrací pole automaticky nastaví na aktuální tým. V uvedeném příkladu je aktuálně na tahu tým „All stars", který je zvýrazněn na hracím plánu zvětšením. Pokud se hráči rozhlíží po hracím plánu a chtějí rychle najít aktuální tým, stačí kliknout na popis aktuálního týmu a mapa se automaticky zaměří zpět na aktuální tým. Pro zpřehlednění hracího plánu jsou pole barevně oddělena podle kategorií úkolů a dále rozdělena střídající se světlejší a tmavší barvou pro lepší orientaci i v místech, kde se nachází více polí stejné kategorie.

Obrázek 7 – hrací plán

Po kliknutí na „Hraj" se zobrazí okna závislá na nastavení. Pokud je nastaveno libovolné pořadí hráčů v týmu, zobrazí se výběr hráče, který chce danou kategorii předvádět. Poté se zobrazí nabídka obtížnosti od tří po pět, pokud opět není vypnuté nastavení obtížnosti. Po výběru se zobrazí příprava a časový limit na přípravu. Po přípravě se přejde na vlastní předvádění buď po uplynutí časového limitu na přípravu, nebo po stisknutí tlačítka „Start". Příklad zobrazení předvádění a hádání pojmu je na Obrázek 8. Je-li nastaven čas na předvádění, je tento čas zobrazen a doprovázen zvukovou signalizací a při vypršení času přístroj též zavibruje.

Pokud je za úkol kreslení, je možné kliknout na tlačítko „Kresli" a na čistou plochu dotykem kreslit, což je možné vidět též na Obrázek 8. Smazání plátna je poté možné tlačítkem přístroje „menu" a tlačítkem přístroje „zpět" se vrátit na plochu předvádění. Příklad kódu vytvoření kreslícího plátna se nachází v Příloze D. Nejdůležitější metodou je zde *onDraw* reagující na každou změnu. Každé volání této metody překreslí hrací plán pomocí jednotlivých bodů uložených v pomocném dynamickém poli *points*. Druhou nejdůležitější metodou je *onTouch*, která ukládá každý bod, který hráč vytvoří dotykem na kreslící plátno. Po uhodnutí či neuhodnutí se vybere příslušné tlačítko a dle modifikací nastavení je možnost hádání ostatních týmů. Poté se celá hra uloží a opět zobrazí hrací plán a zařízení se předává dalšímu týmu.



Obrázek 8 – předvádění

Důležitou vlastností aplikace je paměť odehraných karet. Každá karta, která se zobrazí, i kdyby jen na výběr (například na počátečním poli), je označena jako použitá, a tím se vyřadí z výběru karet pro následující kola. Tento stav se opakuje do chvíle, kdy v dané kategorii a obtížnosti nejsou žádné nepoužité karty. V tu chvíli se všechny karty pouze této kategorie označí jako nepoužité a opět se náhodně vybere jedna z nich. Pouze v tento okamžik hrozí výskyt dvou stejných pojmů. Ovšem při plánované velikosti databáze karet všech kategorií to lze považovat za nevýznamné.

Pokud tým dokončí hru a dostane se do cíle, objeví se nabídka pro ukončení hry, nebo dohrávání ostatních týmů. Pokud se hra ukončí, objeví se výpis konečného pořadí a přejde se zpět do menu.

Na hracím plánu je možné vstoupit do nastavení a upravit aktuální tým i hráče na tahu a umístění týmu na hracím plánu. Též je zde možnost úpravy týmů, pokud hráč odejde nebo naopak nový hráč přijde do hry. Navrácení na plán je možné aplikováním všech změn, nebo bez uložení změn.

5.5 Možnosti nastavení hry

Vývoj aplikace byl zaměřen na ponechání volnosti uživateli. V praxi každému uživateli vyhovuje trochu něco jiného, čímž je běžnými hrami svazován. Aplikace v tomto umožňuje určitou flexibilitu a uživatel má možnost provádět určité změny nejen při vytváření her, ale i v průběhu her, pokud dojde k neočekávanému stavu.

5.5.1 Nastavení pravidel

Pravidla každé hry lze nastavit jiná dle dohody hráčů, avšak z důvodu dodržení pravidel, aby nedocházelo ke zvýhodňování některých týmů, jsou poté tato pravidla neměnná po dobu celé hry. Hra umožňuje následující modifikace pravidel:

- **čas na přípravu** – od zakázaného času na přípravu po jednu minutu času na přípravu, popřípadě neomezený čas na přípravu. Původní hodnota je nastavena na půl minuty.

- **čas na hádání** – od deseti sekund (pro opravdu náročné hráče) do pěti minut, případě pak neomezený čas pro relaxační hru. Původní hodnota je nastavena na 2,5 minuty.

- **pořadí hráčů** – určuje pořadí hráčů v týmu dle toho, jak se budou střídat na úkoly. Hráči se mohou dohodnout, že až podle typu úkonu si budou vybírat, kdo se ujme předvádění. K tomu slouží možnost „libovolné“. Dále jsou možnosti „pevné“ a „náhodné“.

- **velikost hracího pole** – v rozmezí od 10 do 50 polí. Původní hodnota nastavena na 30 polí.

- **rozdělení úloh na hracím poli** – určuje rozdělení jednotlivých úloh a jejich výskyt na hracím plánu. Je zde výběr mezi rozdělením „po úkonech“, „pravidelné“ a „náhodné“. Z důvodu větší zábavnosti hry je přednastavena volba „náhodné“.

- **výběr prvního pole** – první pole hracího plánu nemá definovaný úkon. Tímto nastavením je zadáno, zda typ úkolu si vybírá hráč, nebo je mu přidělen aplikací. S tímto nastavením se váže další nastavení. Pokud je zde zvolena volba „hráč“, je tu další volba, zda hráč bude vybírat pouze typ úkonu, nebo mu bude zobrazena i konkrétní karta s textem úkonu. To znamená, jestli hráč uvidí například pouze „Malování“, nebo „Malování – Neplavec“.

- **výběr posledního pole** – Obdobným způsobem poslední pole nemá nastaven typ úkonu, ale úkol je hráči buď přiřazen, nebo mu úkon vyberou protihráči. S tímto je opět spojeno druhé nastavení, zda protihráči uvidí pouze typ úkonu, nebo i text jednotlivých úkonů.

- **hádání ostatních** – po uplynutí časového limitu je nabízena možnost hádání ostatních hráčů. Při uhodnutí získává body tým, který pojem uhodl.

- **obtížnost úkolů** – nabízí možnost vypnout obtížnost karet a k výběru dochází automaticky, čímž je zaručena větší role náhody. Při uhodnutí se tým posouvá pouze o jedno pole.

- **kreslení v mobilu** – plná přenositelnost hry je často limitována absencí papírů a tužek pro možnost kreslení. Toto nastavení umožňuje kreslit přímo na plochu přístroje, a tím je tento problém vyřešen a sám přístroj umožní plnou podporu hry bez nutnosti dalších pomůcek.

- **úkoly vysvětlování, kreslení, pantomima, ano/ne** – umožňuje vypnutí daného typu úlohy. Dané místo nemusí poskytovat možnost hrát některé typy úloh. Tímto nastavením je možné vybrat si jen ty typy, které je možné hrát a vypnout ty, které

hráče třeba nebaví. Nutností ovšem zůstává nechat alespoň jeden typ úkolů zapnutý, což aplikace kontroluje a nedovolí uživateli takový stav uložit.

5.5.2 Kreslení

Kreslení v přístroji je jedním z hlavních výhod digitální verze hry. K vlastní hře již není potřeba dalších pomůcek a jednoduchým ovládáním je uživatel schopen tohoto využít. Uživatel předvádějící úkol typu kreslení má k dispozici tlačítko „Kresli“, na které když klikne, dostane se na černé plátno plně připravené na kreslení. Na tomto plátně kreslí dotykem automaticky jednou barvou, jak je tomu v pravidlech. Pokud potřebuje plátno smazat, poslouží k tomu tlačítko „Menu“. Pro návrat na plochu předvádění slouží tlačítko „Zpět“.

5.5.3 Ukládání her

Při vytvoření nové hry se hra uloží do databáze. Od této chvíle ji nelze z databáze smazat, pouze ji upravovat. Úprava je zde jak ruční, tak i automatická. Jedna z možností ruční úpravy je popsána výše, kdy uživatel může hru tzv. smazat, ale hra se pouze nastaví do stavu ukončené hry. Další možností ruční úpravy je v průběhu hry, když hráčům nastane nečekaná událost. Například pokud jeden z hráčů musí odejít nebo naopak nový hráč přijde a chce se připojit do hry. Aplikace umožňuje úpravy týmů, které po uložení také upraví hru uloženou v databázi.

K automatickému ukládání dochází po každém odehraném tahu, kdy se aktualizuje stav databáze. Tím je zaručena stabilita systému, neboť při pádu aplikace, celého systému či například vybití baterie, je zajištěno uložení aktuální hry se ztrátou maximálně posledního nedokončeného tahu. K automatickému updatu dochází též při dohrání hry.

5.5.4 Úpravy týmů

U deskových her se občas stane, že někdo musí odejít, nebo naopak někdo nový přijde a rád by se zapojil do hry. U většiny her není problém na takovouto událost reagovat. Složitější už to bývá u aplikací, kde se vše nastaví na začátku hry.

Aplikace umožňuje tyto úpravy provádět přímo z hracího plánu, aniž by se musela hra ukončit. Umožňuje též dělat další úpravy v případě, že dojde k překlepu a tým se například ocitne na špatném políčku. Ke všem těmto úpravám se hráči dostanou pomocí tlačítka „Úpravy“ na hracím plánu. Zde je možné změnit aktuální tým na tahu, popřípadě si rozkliknout daný tým a nastavit mu aktuálního hráče na tahu a pole, na kterém tým stojí. Dále je zde možnost nastavení vlastních týmů. Jsou zde stejné možnosti jako při vytváření nové hry, tzn. nejen úpravy hráčů v týmech, ale i možnosti přidávání a odebírání celých týmů.

Po ukončení úprav jsou k dispozici dvě možnosti návratu. Možnost odchodu beze změn, kdy se nahraje uložený stav z databáze, nebo možnost aplikování změn, které se uloží do databáze.

5.6 Aktualizace

Po spuštění aplikace automaticky dochází ke kontrole stavu databáze. Tuto práci obstarává třída *XMLReader* viz Příloha E. Nejprve proběhne kontrola připojení k Internetu. Pokud tato kontrola proběhne úspěšně, stáhne se XML soubor, který je ve formátu:

```
<Aktualizace>2013.7.20.17.30</Aktualizace>
<Karta>
      <Sada>Základní</Sada>
      <Hodnota>3</Hodnota>
      <Ukon>Mluvení</Ukon>
      <Text>Pověst</Text>
</Karta>
```

Pokud dojde k chybě, načte se lokální soubor stejného formátu. Dochází k porovnání aktualizací, a pokud je verze databáze zastaralá, dojde ke zpracování souboru a uložení nových karet do databáze. XML soubor vystavený na Internetu je dočasným řešením a v průběhu dalšího vývoje aplikace bude nahrazen databázovým systémem. V Příloze E je v první části ukázka načtení lokálního souboru „Cards.xml“ a v druhé části poté zpracování načteného souboru. Je zde použita vnitřní třída *Element*. Pomocí této třídy dojde k elegantnímu rozčlenění XML souboru.

5.7 Statistiky

Aplikace poskytuje možnost náhledu na jednotlivé hráče a jejich celkové úspěchy v součtu všech zúčastněných her. Statistiky jsou rozděleny na kategorie celkových statistik a dále na statistiky jednotlivých typů úkonů. Každá kategorie je rozdělena na čtyři části:

- **celkové skóre** – vyjadřuje celkovou úspěšnost hráčů,
- **nejlepší aktér** – vyjadřuje hráče nejlepší v předvádění pojmů,
- **nejlepší hádač** – vyjadřuje hráče nejlepší v hádání pojmů,
- **nejlepší souhra** – vyjadřuje dvojici hráčů, kteří jsou nejlépe sehraní.

V každé části je zobrazeno celkové skóre tří nejúspěšnějších hráčů, popřípadě více hráčů, pokud se dělí o umístění. Ukázku je možné vidět na Obrázek 9. Úspěšnost je vypočtena v procentech poměrem všech odehraných kol a vyhraných kol. Procentuální úspěšnost může být někdy matoucí, neboť pokud hráč odehraje úspěšně jediný tah, má 100% úspěšnost. Proto je zde i údaj o počtu odehraných karet. Hra tím získává pro hráče nový rozměr, neboť zde nejsou hodnoceny týmy a hry jako takové, ale pouze hráči, a to i jednotlivých kategoriích. Nejzajímavějším údajem ve statistikách je poté ukazatel nejlepší souhry dvou hráčů.

Obrázek 9 – statistiky

Při smazání hráče z databáze se automaticky smažou veškeré záznamy o odehraných hrách, protože tyto údaje již nejsou pro aplikaci relevantní. Údaje v databázi se díky tomuto nehromadí a jsou zde uloženy pouze údaje, které aplikace využívá. Dalším aspektem je ignorování takzvaných „HostPlayerů“, kteří, i když mají vysokou úspěšnost, se nikdy neobjeví ve statistikách. Proto jsou při ukončení hry automaticky vymazáni z databáze. Spolu s nimi jsou smazány i údaje o jejich odehraných tazích.

5.8 Úpravy databáze

Z hlavního menu se lze dostat do úprav databáze. Tyto úpravy jsou pro uživatele velmi omezené a slouží ke zpřehlednění databázového systému. Z důvodu zabezpečení stability databázové struktury zde uživatelé mohou dělat jen drobné úpravy. Uživatelé mohou vstoupit do seznamu rozehraných her a vybrané hry smazat. Dále má uživatel možnost nechat si vypsat seznam hráčů uložených v databázi. Zde může uživatele vytvářet, upravovat a mazat. Mazat je ovšem může pouze v případě, že se uživatel neúčastní rozehrané hry. To jsou veškeré možnosti, kterými může uživatel upravovat databázi. Ostatní úpravy jsou prováděny automaticky.

5.9 Zabezpečení

Problematika zabezpečení OS Android a jeho aplikací je často diskutovaná a ochránit aplikace je takřka nemožné. Pokud se uživatel dostane k instalačnímu balíčku aplikace, stačí použít příslušné programy pro dekódování k získání zdrojových kódů.

Aplikace je zabezpečena proti hrubému nahrazení databází. Při vytvoření databáze se do ní automaticky uloží ID zařízení. Pokud se útočník poté pokusí danou databázi přenést na jiné zařízení, program automaticky rozpozná cizí databázi a automaticky ji smaže a nahradí základním objemem dat, které činí 192 karet z přiloženého souboru.

Při pokusu o nahrazení přiloženého souboru program nerozpozná jiný soubor, nicméně opět ze souboru načte pouze prvních 192 slov.

5.10 Požadavky na systém

Užívání aplikace je omezeno na určitý okruh zařízení. Kromě nutné minimální verze OS Android musí zařízení obsahovat kartu SD, vhodné je i připojení k Internetu, větší displej a celkově výkonnější zařízení. Předpoklady pro používání aplikace jsou uvedeny v řídícím souboru „Manifest.xml“ následujícími příkazy:

- ***android.permission.INTERNET*** – pro připojení k Internetu a možnosti stáhnutí aktualizací. Bez připojení k Internetu je aplikace funkční, pouze nebude provedena kontrola aktualizací databáze slov.
- ***android.permission.WRITE_EXTERNAL_STORAGE*** – pro ukládání databáze na kartu SD. Zde dochází k jistému omezení aplikace pro zařízení, které nemají možnost připojení karty SD.
- ***android.permission.VIBRATE*** – povolení vibrací pro aplikaci slouží jako další signalizace aplikace. V průběhu hraní hry nemusí zvukové signalizace vždy postačit, proto je vhodné tuto signalizaci doplnit.
- ***android.permission.GET_ACCOUNTS*** – povolení získání informací o ID zařízení a e-mailové adrese, na kterou je zařízení napojeno, slouží k lepší identifikaci uživatele a zabezpečení aplikace.
- ***android.permission.READ_PHONE_STATE*** – poskytnutí údajů ke kontrole verze OS Android.

Hra je též vhodnější pro větší a výkonnější zařízení z důvodů rychlosti aplikace zvláště při zpracování databáze. Dále pak pro kreslení na zařízení je vhodnější větší displej a případně je vhodné použití stylusu. Zařízení s nízkým výkonem mohou mít problémy s rychlostí práce s databází ve chvíli, kdy je odehráno více her a databáze dosáhne určitého objemu dat. Avšak stále z důvodu kompaktnosti je aplikace doporučena spíše pro mobilní zařízení oproti například tabletům.

Závěr

Účelem bakalářské práce bylo navrhnout mobilní aplikaci pro operační systém Android dle předlohy oblíbené společenské hry Activity. Z této hry byly vybrány části, díky kterým se hra stala velmi populární, a byly zde aplikovány úpravy k zamezení negativních vlastností deskových her. Největšími výhodami oproti původní hře jsou: lehká přenositelnost hry, nepotřebnost dalších pomůcek – konkrétně papír a tužka pro úkoly kreslení, a řešení práce s odehranými kartami. Aplikace dále automatickým stahováním aktuální databáze karet poskytuje neomezeně se rozšiřující databázi slov, čímž je vyřešena největší nevýhoda původní hry. Neopomenutelnou výhodou celé této hry jsou zajisté hodiny plné zábavy, které ovšem současně rozvíjí verbální i neverbální vyjadřování a slovní zásobu.

Hra poskytuje velký potenciál pro možné rozšíření:

- **vznik databázového systému na straně serveru** – je hlavním plánovaným rozšířením, který bude poskytovat vhodnější správu databáze karet. Uživatelé pak nebudou muset stahovat celou aktualizovanou databázi, ale pouze část databáze s novými pojmy.

- **rozdělení hry na jednotlivé sady karet** – je rozšíření, které již má v kódu své zastoupení, i když není ještě plně aplikované. Příkladem může být hra s dětmi, které neznají některé pojmy a to řeší sada určená pro děti, nebo sada s tématikou filmů. Dále je možné vytváření tematických sad například s oborovou tématikou. Výhodou pro aplikaci a její tvůrce je poté v možnosti tyto další sady karet vydávat zvlášť za případné další poplatky. Opět v rámci ponechání volnosti si uživatelé budou moci zapnout například všechny sady a získat tak již velmi rozsáhlou databázi slov.

- **rozšiřování databáze uživateli** – je dalším možným rozšířením. V aplikaci bude tlačítko na vytvoření nového pojmu a po zadání se automaticky uloží na straně serveru do speciální tabulky. Z tabulky poté administrátor pojem přijme a zkontroluje duplicitu, zda již není obdobný pojem zadán. Po vložení do databáze si všichni uživatelé automaticky stáhnou aktuální databázi při novém spuštění hry.

- **nové typy úloh** – se nabízí jako další velké rozšíření. Díky možnosti zapnout si libovolné typy úloh je vhodné uživatelům nabídnout větší výběr úloh. Inspiraci těchto úloh je opět možné hledat u jiných her a nabízí se například hry typu: vytváření rýmů, vydávání zvuků, vyjmenování minimálního počtu podstatných jmen s určitým omezením například tématem nebo počátečním písmenem atd.

- **hra na více zařízeních** – ať už s možností, že každý tým bude mít jedno zařízení, která budou synchronizována, nebo propojení s větším zařízením, například tablet. Mobilní zařízení zde může sloužit jako karty a tablet bude zobrazovat hrací plán.

Velkou problematikou vývoje aplikací pro OS Android se stává zabezpečení. Tento problém nutí vývojáře k výběru ze dvou možností. Vytvořit uživatelsky přívětivou aplikaci

s možností, že bude snadno prolomena a zneužita, nebo silně omezit uživatele a získat jistotu bezpečnosti aplikace. Aplikace lze v dnešní době ochránit v podstatě jedinou možností a tou je nutnost připojení aplikace k Internetu. To se v dnešní době stává stále běžnější, avšak stále například v zahraničí jsou internetové služby zpoplatněny bez ohledu na placený tarif. Tato aplikace byla navržena uživatelsky přívětivějším způsobem za cenu možného prolomení bezpečnostních ochran.

Bakalářkou prací vývoj této aplikace zdaleka nekončí a je počítáno s postupným rozvojem aplikace a uvedením aplikace na trh.

Literatura

[1] UJBÁNYAI, Miroslav. Programujeme pro Android. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.

[2] MURPHY, Mark L. Android 2: průvodce programováním mobilních aplikací. Vyd. 1. Brno: Computer Press, 2011, 375 s. Průvodce (Grada). ISBN 978-80-251-3194-7.

[3] DIMARZIO, J. *Programujeme hry pro Android 4*. Brno: Computer Press, 2012, 310 s. ISBN 978-80-251-3754-3.

[4] MAREK, Lukáš. Seriál Programování pro Android v příkladech. *Root.cz* [online]. 2001-01-23 [cit. 2014-04-25]. ISSN 1212-8309 Dostupné z: <http://www.root.cz/serialy/programovani-pro-android-v-prikladech/>

[5] POŠVIC, Kamil. Jak moc je Android vůbec Linux. *Root.cz* [online]. 2009-11-12 [cit. 2014-04-25]. ISSN 1212-8309 Dostupné z: <http://www.root.cz/serialy/programovani-pro-android-v-prikladech/>

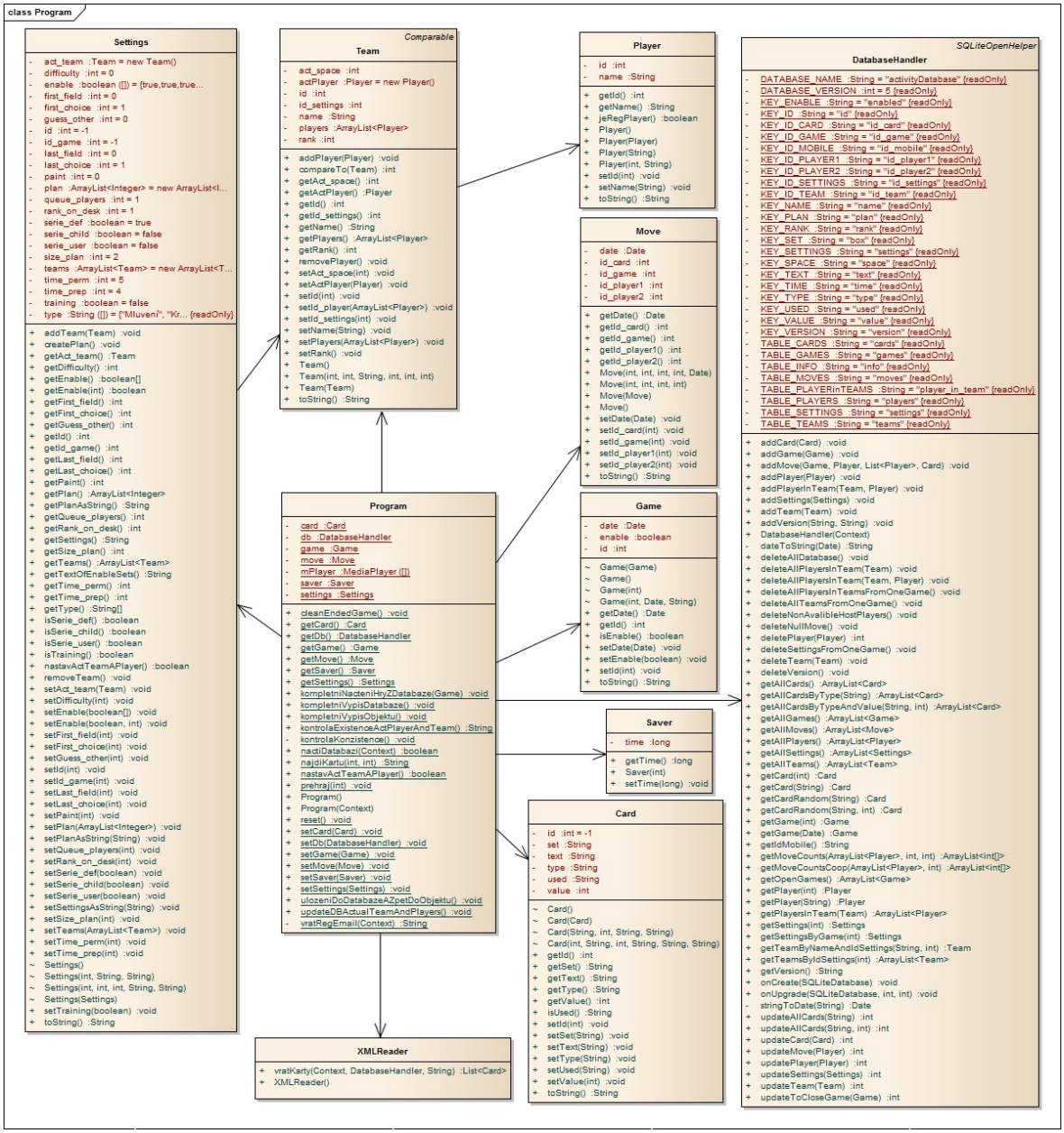
[6] NEWMAN, Chris. *SQLite*. Indianapolis, Ind.: Sams, 2005, 313 s. ISBN 06-723-2685-X.

[7] KOSEK, Jiří. *XML pro každého* [online]. 2006-04-03[cit. 2014-04-25]. Dostupné z: <http://www.kosek.cz/xml/>

[8] KOŠATA, Béďa. Seriál XML – eXtensible Markup Language. *Root.cz* [online]. 2001-01-23 [cit. 2014-04-25]. ISSN 1212-8309 Dostupné z: <http://www.root.cz/serialy/xml-extensible-markup-language/>

[9] HEROUT, Pavel. Java a XML: průvodce programováním mobilních aplikací. 1. vyd. České Budějovice: Kopp, 2007, 313 s. Průvodce (Grada). ISBN 978-80-7232-307-4.

Příloha A – UML diagram



Příloha B – Ukázka zdrojového kódu DatabaseHandler.java

```
…
public void addPlayer(Player player) {
      SQLiteDatabase db = this.getWritableDatabase();
      ContentValues values = new ContentValues();
      values.put(KEY_NAME, player.getName());
      try {
            db.insertOrThrow(TABLE_PLAYERS, null, values);
      } catch (SQLiteConstraintException e) {
            Log.e("AcErr:", "Nick " + player.getName()
                  + " is already in database");
      }
      db.close();
}

public Card getCardRandom(String typUkolu, int hodnotaKarty) {
      String sada = "";
      sada = Program.getSettings().getTextOfEnableSets();
      SQLiteDatabase db = this.getReadableDatabase();
Cursor cursor = db.query(TABLE_CARDS, new String[] {
KEY_ID,KEY_TEXT,KEY_VALUE, KEY_TYPE, KEY_SET, KEY_USED },
KEY_VALUE + "="+ hodnotaKarty + " AND " + KEY_TYPE + " LIKE '" +
typUkolu + "' AND " + KEY_SET + " IN " + sada + " AND " + KEY_USED + "
LIKE '" + "false'", null, null, null, "random()", "1");
      if (cursor != null)
            cursor.moveToFirst();
      if (cursor.getCount() < 1) {
            updateAllCards(typUkolu, hodnotaKarty);
            return getCardRandom(typUkolu, hodnotaKarty);
      }
      Card card = new Card(Integer.parseInt(cursor.getString(0)), // id
            cursor.getString(1), // text
            Integer.parseInt(cursor.getString(2)), // value
            cursor.getString(3), // type
            cursor.getString(4), // set
            cursor.getString(5)); // used
            card.setUsed("true");
            updateCard(card);
            return card;
      }

public int updateToCloseGame(Game game) {
      SQLiteDatabase db = this.getWritableDatabase();
      ContentValues values = new ContentValues();
      values.put(KEY_ID, game.getId());
      values.put(KEY_TIME, dateToString(game.getDate()));
      values.put(KEY_ENABLE, "false");
      return db.update(TABLE_GAMES, values, KEY_ID + " = ?",
                  new String[] { String.valueOf(game.getId()) });
}
…
```

Příloha C – Ukázka zdrojového kódu plan.xml

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/planTextView"
        android:orientation="vertical" >

        <ScrollView android:id="@+id/PlanScroll"
            android:layout_width="match_parent"
            android:layout_height="0dip"
            android:fillViewport="true"
            android:layout_weight="0.12"
            android:transcriptMode="alwaysScroll">

                <TableLayout
                    android:id="@+id/plan_table"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:gravity="left"
                    android:stretchColumns="0"
                    android:background="#000000">

                </TableLayout>
        </ScrollView>
    </LinearLayout>


      <TextView
          android:id="@+id/planTextView"
          style="@style/buttonStyle"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_above="@+id/planLeave"
          android:layout_alignParentLeft="true"
          android:layout_alignParentRight="true"
          android:gravity="center|bottom"
          android:text="@string/load" />

    <Button
        android:id="@+id/planLeave"
        style="@style/buttonStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:background="@drawable/buttons_left"
        android:onClick="planLeave"
        android:text="@string/leave" />
…
```

Příloha D – Ukázka zdrojového kódu graphics.java

```
public class PaintView extends View implements OnTouchListener {
  Paint paint = new Paint();
  public PaintView(Context context) {
    super(context);
    setFocusable(true);
    setFocusableInTouchMode(true);
    this.setOnTouchListener(this);

    paint.setColor(Color.YELLOW);
    paint.setAntiAlias(true);
    paint.setStrokeWidth(8);
  }

  @Override
  public void onDraw(Canvas canvas) {
    for (int i = 0; i < points.size() - 2; i++) {
      if (points.get(i).x < 0 || points.get(i + 1).x < 0){
        continue;
      }
      canvas.drawLine(points.get(i + 1).x, points.get(i+1).y,
                      points.get(i).x, points.get(i).y, paint);
    }
  }

  public boolean onTouch(View view, MotionEvent event) {
    Point point = new Point();
    switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
      toast.cancel();
      point.x = -1;
      point.y = -1;
      points.add(point);
      point = new Point();
      point.x = event.getX();
      point.y = event.getY();
      points.add(point);
    case MotionEvent.ACTION_MOVE:
      point.x = event.getX();
      point.y = event.getY();
    case MotionEvent.ACTION_UP:
      point.x = event.getX();
      point.y = event.getY();
    }
    points.add(point);
    invalidate();
    return true;
  }
…
```

Příloha E – Ukázka zdrojového kódu XMLReader.java

```
…
List<Card> cards = new ArrayList<Card>();
Document doc = null;
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
…
AssetManager asMan;
asMan = c.getApplicationContext().getAssets();
InputStream inStr;
try {
      inStr = asMan.open("Cards.xml");
      DocumentBuilder dBuilder;
      dBuilder = dbFactory.newDocumentBuilder();
      doc = dBuilder.parse(inStr);
      doc.getDocumentElement().normalize();
} catch (IOException e) {
      e.printStackTrace();
} catch (ParserConfigurationException e) {
      e.printStackTrace();
} catch (SAXException e) {
      e.printStackTrace();
}

String sada = "", hodnota = "", ukon = "", text = "";
nList = doc.getElementsByTagName("Karta");
int i = 192;
for (int temp = 0; temp < nList.getLength(); temp++) {
      nNode = nList.item(temp);
      if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            if (!pripojeni) {
                  i--;
                  if (i == 0) {
                        break;
                  }
            }
      Element eElement = (Element) nNode;
sada = eElement.getElementsByTagName("Sada").item(0).getTextContent();
hodnota=eElement.getElementsByTagName("Hodnota").item(0)
                                                      .getTextContent();
ukon = eElement.getElementsByTagName("Ukon").item(0).getTextContent();
text = eElement.getElementsByTagName("Text").item(0).getTextContent();

cards.add(new Card(text, Integer.parseInt(hodnota), ukon, sada));
      }
}
…
```