

Aplikace pro záznam průběhu sportovních utkání

Bc. Martin Macko

Diplomová práce
2016



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin Macko**
Osobní číslo: **A14403**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Počítačové a komunikační systémy**
Forma studia: **prezenční**

Téma práce: **Aplikace pro záznam průběhu sportovních utkání**
Téma anglicky: **An Application for Tracking the Course of Sporting Events**

Zásady pro vypracování:

1. Seznamte se s pravidly míčové hry softball a baseball.
2. Vypracujte literární rešerši na téma software využívaný pro zápis průběhu utkání.
3. Navrhněte vlastní aplikaci záznamu hry pro platformu Android.
4. Umožněte sledovat aktivitu pálkařů, běžců i polařů.
5. Zpracujte automatické generováním statistik a export výsledků.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Lacko, L'. Vývoj aplikací pro Android. 1. vyd. Brno: Computer Press, 2015, 472 s. ISBN 978-80-251-4347-6.
2. Griffiths, D., Griffiths, D. Head First Android Development. O'Reilly Media, 2015. ISBN 978-1-4493-6213-3.
3. Schildt, H. Mistrovství – Java. 1. vyd. Brno: Computer Press, 2014, 1224 s. Mistrovství. ISBN 978-80-251-4145-8.
4. Aditya, S. a Karn, V.. Android SQLite Essentials. Packt Publishing, 2014, 127 s. ISBN 978-1783282951.
5. Castro, E., Bruce, H. HTML5 a CSS3: názorný průvodce tvorbou WWW stránek. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.
6. Süß, V., Votinský, J. Příručka pro zapisovatele v softbalu [online]. Praha, 2005 [cit. 2016-01-28]. Dostupné z: <http://www.pegas.sedlcany.cz/files/documents/zapisovatel.pdf>

Vedoucí diplomové práce:

doc. Ing. Martin Sysel, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

5. února 2016

Termín odevzdání diplomové práce:

23. května 2016

Ve Zlíně dne 5. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



Ing. Miroslav Matýšek, Ph.D.
ředitel ústavu


Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen přípouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 17.05.2016


.....
podpis diplomanta

ABSTRAKT

Predmetom diplomovej práce je vytvoriť mobilnú aplikáciu umožňujúcu zaznamenávanie priebehu športového zápasu v hre softball. Teoretická časť obsahuje úvod a krátke predstavenie platformy Android, pre ktorú bola aplikácia vyvinutá. Ďalej sú v nej stručne popísané pravidlá hry, spôsoby zápisu a dostupné aplikácie na samotný zápis. V poslednej kapitole teoretickej časti sú popísané dynamické dátové štruktúry, ktoré boli použité v samotnej aplikácii. Praktická časť práce sa potom skladá z popisu štruktúry vytvorenej aplikácie a popisu funkcionality z pohľadu užívateľa. Súčasťou práce je taktiež užívateľská príručka.

Kľúčová slova: android, softball, dynamické dátové štruktúry, lineárny zoznam

ABSTRACT

The subject of diploma thesis is to create mobile application that enables recording of softball match. Theoretical part contains a short introduction to the Android platform on which was application developed. Next part consist of brief description of game rules, procedures of hand record and description of available applications for record. In last chapter of theoretical part are described dynamic data structures that were used in the application. Practical part of thesis consists of application strucutre description and description of the functionality from the users perspective. Part of the thesis is also user manual.

Keywords: android, softball, dynamic data structures, linked list

Chcel by som sa poďakovať doc.Ing. Martinovi Syslovi Phd. Za vedenie mojej diplomovej práce, cenné rady a odborný dohľad. Taktiež mojej rodine a priateľke za podporu a pomoc ohľadom gramatickej stránky práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 PLATFORMA ANDROID	11
1.1 ARCHITEKTÚRA ANDROIDU	15
1.2 ARCHITEKTÚRA ANDROID APLIKÁCIE	17
1.2.1 Životný cyklus aktivity	20
1.2.2 Moduly Android aplikacie	22
1.3 VÝVOJOVÉ PROSTREDIE	23
2 ZÁZNAM ŠPORTOVÝCH ZÁPASOV	26
2.1 EXISTUJÚCE ANDROID APLIKÁCIE PRE ZÁZNAM ŠPORTOVÉHO ZÁPASU	28
2.1.1 iScore.....	28
2.1.2 ScoreFinger	31
3 DYNAMICKÉ DATOVÉ ŠTRUKTÚRY	33
II PRAKTICKÁ ČÁST	39
4 POPIS APLIKÁCIE SOFTBALL TRACKER	40
4.1 OBRAZOVKY SPRÁVCA TÝMOV	40
4.2 OBRAZOVKY VÝBERU TÝMOV	42
4.3 OBRAZOVKY ZÁPISU HRY	43
5 PROGRAMÁTORSKÁ PRÍRUČKA	48
5.1 DATABÁZA	50
5.2 ZÁPIS HRY	53
5.2.1 Parser zápisu.....	57
5.3 AKTIVITY	58
5.3.1 Aktivity spravujúce databázu	59
5.3.2 Aktivity zápisu hry	61
5.4 XML LAYOUT.....	65
ZÁVĚR	67
SEZNAM POUŽITÉ LITERATURY	68
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	70
SEZNAM OBRÁZKŮ	71
SEZNAM PŘÍLOH	73
5.5 SPRÁVCA TÝMOV	74
5.6 SPUSTENIE HRY	75
5.7 ZÁPIS HRY	76
5.7.1 Odpal do ľavej časti vnútorného poľa, páľkar out	78
5.7.2 Odpal do ľavej časti vnútorného poľa, páľkar safe	79
5.7.3 Vysoký odpal do pravého zadného poľa	80
5.7.4 Odpal po zemi do stredného vonkajšieho poľa	81
5.7.5 Prvá meta obsadená, páľkar BB	82
5.7.6 Druhá meta obsadená páľkar SO	83
5.7.7 Druhá meta obsadená, páľkar HB.....	84

5.7.8	Prvé dve méty obsadené, bežci outovaný	85
5.7.9	Prvá meta obsadená, páľkar SH.....	86
5.7.10	Ukážka opravy zápisu	87

ÚVOD

V dnešnej dobe sú smartphony, alebo tablety základnou výbavou každého človeka. Ich funkcionality sa stále rozširujú a tak pomaly vytlačujú iné technológie. Ľudia si bežne prečítajú noviny na svojom mobile, odfotia sa s ním na dovolenke, alebo ho použijú ako navigáciu pri cestovaní. Technologický pokrok je už v štádiu, kedy takéto zariadenia vďaka vysokému výkonu nahrádzajú pri mnohých činnostiach personálne počítače. Tým sa zvyšuje dopyt po aplikáciách, ktoré bežia na mobilných zariadeniach. Táto diplomová práca sa skladá z dvoch častí zaoberajúcimi sa práve aplikáciami na spomínané mobilné zariadenia.

Prenejšie sa zaoberá aplikáciami vytváranými natívne pre platformu Android, ktorá je dnes s veľkým náskokom najpoužívanejším operačným systémom pre smartphony. V prvom bode teoretickej časti je predstavený operačný systém Android a okolnosti, ktoré stáli za jeho vznikom. Taktiež je načrtnutá história vývoja tohoto operačného systému a architektúra na ktorej pracuje. V ďalšej kapitole je popísaná architektúra aplikácií vyvíjaných na platforme Android a dostupné vývojové prostriedky, či odporúčané postupy pri vývoji aplikácií. Druhý bod obsahuje stručné zoznámenie s loptovou hrou softbal. Praktická časť práce sa totiž zaoberá návrhom aplikácie na zaznamenávanie a spracovanie priebehu športového zápasu v tejto hre. Kvôli tomu sa tretí a posledný bod teoretickej časti práce venuje dynamickým dátovým štruktúram, ktoré sú použité na zaznamenávanie, následne spracovanie a prehľadnú interpretáciu komplexných dát.

Ako už bolo spomenuté, praktická časť sa venuje popisu aplikácie na zaznamenávanie priebehu zápasu vyvinutej pre platformu Android. Bežne sa priebeh zápasu zaznačuje písomne, perom na špeciálny hárok o zázname. Takýto záznam je však relatívne zložitý a vyžaduje značné skúsenosti s touto hrou. Preto bola vyvinutá aplikácia, ktorá intuitívne umožňuje zápis zápasu. Aplikácia je navrhnutá tak, aby zápis zvládol aj poučený užívateľ, ktorý nemá skúsenosti so samotnou hrou. Na rozdiel od aplikácií s podobnou funkcionalitou je kladený dôraz na jednoduchosť zápisu tak, aby sa každá herná situácia dala zaznamenať na „pár klikov“ a nie zdĺhavým vyplňovaním formulárov. Táto časť práce je rozdelená do dvoch podsekcí. Prvou je popis aplikácie z pohľadu užívateľa, súčasťou tejto časti je aj príloha užívateľská príručka, ktorá popisuje rôzne herné situácie a spôsob ich zápisu. Druhou je potom štruktúra a architektúra aplikácie z pohľadu vývojára.

I. TEORETICKÁ ČÁST

1 PLATFORMA ANDROID

Android je voľný open source systém vyvíjaný spoločnosťou Google. Za to, že je Android skutočne open source môže situácia na trhu so smartfónmi v roku 2007. V tom čase bol dominantným výrobcom smartfónov spoločnosť Blackberry so svojim vlastným operačným systémom. A tak sa Google uchýlil k drastickému riešeniu. Odkúpil od malej skupiny vývojárov operačný systém Android, ktorý bol postavený na Linuxe a jednoducho ho sprístupnil všetkým. Google sa spoliehal na to, že výrobcovia mobilných zariadení siahnu po voľnom operačnom systéme a Android sa tak stane dominantným. Dnes už sa vie, že táto stratégia zafungovala na výbornú. Android sa stal jasnou voľbou pre výrobcov, ktorý hľadali lacný a jednoducho prispôsobiteľný systém pre svoj hardvér. Ani Google nezostal stratový. Pomocou platformy Android Market (neskôr Google Play) začal na Androide zarábať cez predaj aplikácií, filmov a hudby. V dnešných dňoch, keď už Android kraluje trhu so smartfónmi (minimálne v počte zariadení používajúcich Android) už systém nie je tak „voľný“ ako vo svojich začiatkoch, avšak jeho zdrojový kód zostáva stále otvorený pre všetkých. Odvtedy vydal niekoľko verzií systému Android. Každá nová verzia Androidu je pomenovaná podľa sladkosti alebo dezertu, pričom názvy na seba navezujú v abecednom poradí.[2,3]



Obrázok 1. Prehľad verzií Androidu.[28]

Android 1.X vyšiel v roku 2008. Najznámejšia verzia Android 1.5 Cupcake potom v roku 2009. Už v tejto dobe mal Android omnoho viac funkcií, ako jeho hlavný rivalovia iPhone a Windows mobile, hlavne vďaka dobrej integrácii s už vtedy rozsiahlymi Google službami. V tej dobe však ešte systém trpel na mnoho chýb, ktoré ho značne zrážali dolu. Spomínaný iPhone mal v tej dobe napríklad omnoho menej funkcií, avšak ako celok fungoval lepšie a prinášal kvalitnejší užívateľský zážitok. Ako už bolo spomenuté, táto verzia Androidu vychádzala s Linuxu 2.6.25. [1,2,3]

V roku 2009 bola vydaná druhá generácia systému Android. V tejto generácii sa vývojárom konečne podarilo spojiť funkcionálnosť s príjemným užívateľským zážitkom z používania. Systém, však stále trpel na to, že oproti iPhone bol stále značne pomalý. V tejto generácii vyšli tri verzie. Android 2.0/2.1 Eclair, Android 2.2 Froyo a hlavne Android 2.3 Gingerbread. Táto generácia priniesla mnohé novinky, medzi ktoré patrí napríklad podpora viacerých účtov na jednom zariadení, prevod textu na hovorenú reč, podpora USB a Wi-fi tetheringu (zdieľanie internetového pripojenia), podpora fotoaparátu na čelnej strane zariadenia, vylepšenú správu energie, podporu NFC alebo natívnu podporu senzorov, ako barometer, gyroskop a podobne. Verzie Froyo a Gingerbread sú potom verzie, ktoré majú ešte stále nepatrný podiel na trhu. [1,2]

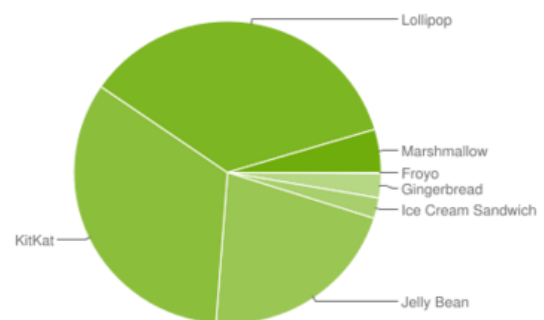
Následne prišla pre Android veľká výzva, Applom bol vydaný prvý iPad. Zároveň presadzoval stratégiu, že aplikácie vydávané na iPad musia byť špeciálne vyvinuté na väčšiu obrazovku. Google teda prišiel s treťou generáciou Androidu, ktorá sa stala tak trochu slepou vývojovou vetvou toho systému. Google totiž použil opačnú stratégiu ako Apple. Všetky Android aplikácie museli bežať na všetkých Android zariadeniach nehládajúc na ich rozlíšenie či veľkosť obrazovky. Táto stratégia, však značne zlihala, nakoľko vývojári spravili veľmi málo pre to, aby Android aplikácie boli prispôbené veľkej obrazovke. Aplikácie vyvíjané na troj palcové displeje potom vyzerali na sedem palcových značne komicky. Dá sa však povedať, že Android 3.0 Honeycomb do istej miery naznačil smer, akým sa Android neskôr v roku 2011 začal uberať. Táto verzia priniesla napríklad vylepšený multitasking (paralelný beh viacerých aplikácií), systémový panel s rýchlym prístupom k oznámeniam, softvérové tlačítka na displeji, možnosť plného šifrovania užívateľských dát alebo podporu príslušenstva pripojeného cez USB (USB On-The-Go). [2,3]

V ďalšej generácii zobral Google vylepšené rozhranie Honeycombu a vytvoril prvý systém štvrtej generácie Android 4.0 Ice Cream Sandwich. Touto generáciou sa Google podarilo

premostiť priepasť medzi mobilmi a tabletmi. Taktiež začal tlačiť vývojárov k vývoju aplikácii optimalizovaných pre tablety. Ice Cream Sandwich priniesol taktiež zásadnú zmenu designu. Priniesol vzhľad Holo a nový font Roboto. Medzi zásadné novinky ďalej patrí kontrola využívania dát, sfunkčnená metóda drag and drop, spúšťania aplikácii rovno z úvodnej obrazovky, prevod hovorenej reči na text v reálnom čase a značné vylepšenie fotoaparátu. Skrátenie oneskorenia, režim panoramatického snímku či možnosť približovania počas natáčania. [1,2]

Následujú 3 verzie, ktoré vyšli pod kódovým menom, avšak každá vychádza z novšieho API. Jellybean je zároveň prvá spomenutá verzia Androidu, ktorá si drží nezanedbateľný podiel na trhu do dnes (viz. *Obrázok 2*). Okrem novej funkcionality bolo veľkým plusom celkové zýchlenie a vyčistenie systému. Podľa mnohých kritikov sa touto verziou podarilo Google konečne spojiť naleštené a rýchle prostredie, ktorým bol známy najmä iOS s výkonnou sadou funkcií, ktorou Android už dávno disponoval. Rozhranie sa od tejto verzie prispôbovalo zariadeniu na ktorom bežalo. Aj v tejto verzii sa ďalej vylepšovala práca s fotoaparátom, vylepšila sa notifikačná lišta a pridala sa podpora viac kanálového zvuku, čo prinieslo prehrávanie hudby bez počuteľných medzier. [1,2,3]

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%



Obrázok 2. Prehľad rozšírenia jednotlivých verzií.[28]

Posledná vydaná verzia zo štvrtej generácie je Android 4.4 KitKat. Tu už sa dostávame do nie až tak vzdialenej minulosti. KitKat pomenovaný po obľúbenej čokoládovej tyčinke je momentálne druhou najrozšírenejšou verziou Androidu. Beží na API 19 a z pohľadu

rozšírenia verzií API, je API verzie 19 je momentálne vôbec najpoužívanejšie. Opäť sa vylepšilo užívateľské rozhranie. Google od tejto verzie začal integrovať kompatibilitu s „nositeľnou elektronikou“ tzv. Android Wear. Po značnej kritike užívateľov prišlo obmedzenie prístupu aplikácií do externého úložiska. Pribudol taktiež NFC emulátor, ktorý povolil používanie mobilu suplovať karty, alebo podpora senzorov na počítanie krokov. Ďalším veľkým krokom, ktorý však bežný užívateľ veľmi nepostrehol bolo predstavenie ART (Android Runtime). Išlo o virtuálny stroj, ktorý v Anroide vytváral runtime prostredie pre aplikácie napísané v Jave. [1,2]

Kitkat uzavrel štvrtú generáciu a na svetlo sveta sa následne mohol dostať Android 5.0/5.1 s kódovým označením Lollipop. To už sme na konci roku 2014 a z histórie postupne prechádzame do súčasnosti. Lollipop priniesol v prvom rade ďalšiu veľkú revolúciu vo vzhľade. Googlom bol predstavený tzv. Material Design. Jednotný vzhľad pre všetky aplikácie, ktorý sa okrem Androidu dostal aj do mnoho iných Google produktov. Bol taktiež soustavený projekt Volta, ktorého cieľom mala byť vylepšená spotreba batérie. Spomenutý ART už plne nahrádza stávajúci Dalvik. Medzi ďalšie novoty patrila podpora 64 bitových procesorov, možnosť rýchleho prenesenia nastavení pomocou NFC a funkcie Tap Go, zoznam posledných spustených aplikácií si systém pamätá aj po reštarte a aplikácie tretích strán možnosť čítať a upravovať dáta. [1,2]

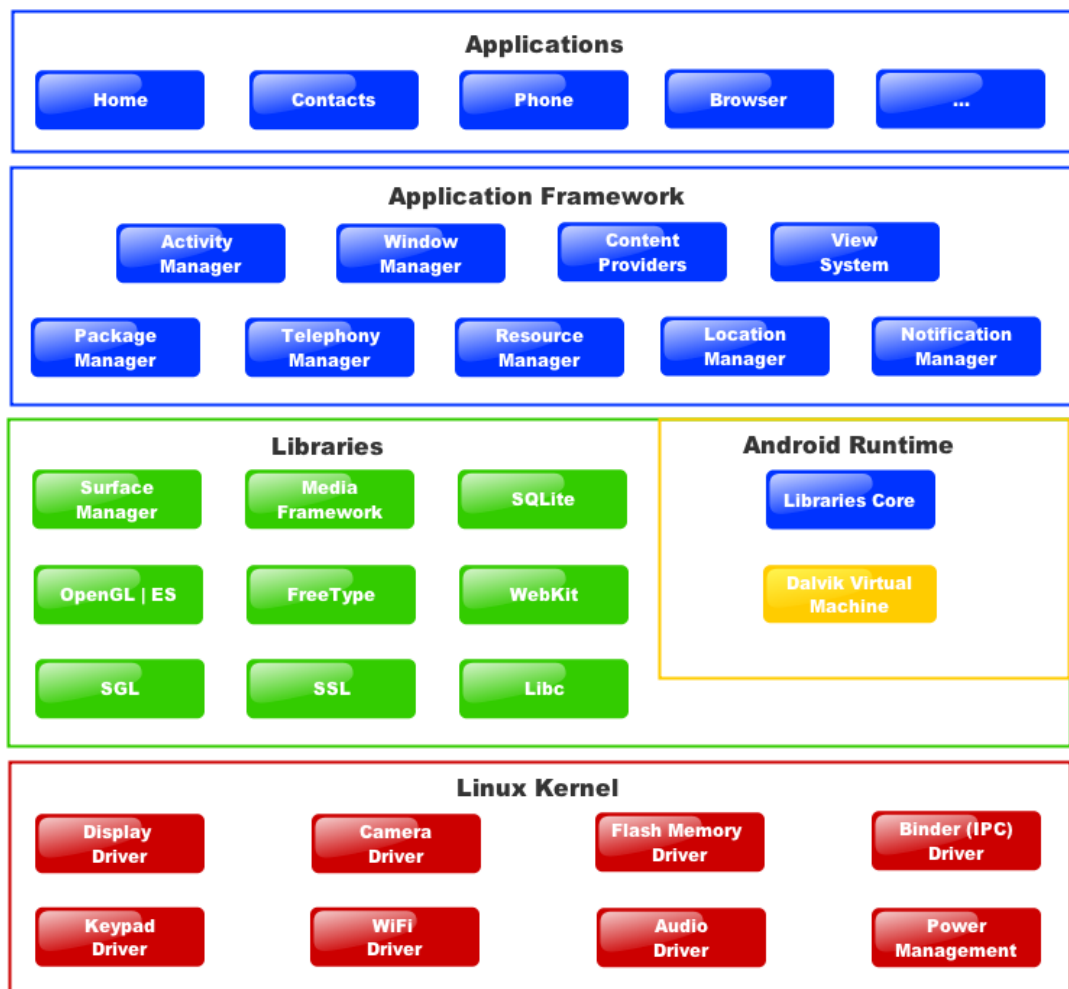


Obrázok 3. Ukážka Android systémov od rôznych výrobcov.[3]

Týmto sa dostávame do súčasnosti a systému Android 6.0 Marshmallow, ktorý bude bližšie rozobraný v ďalšej kapitole. Ako už bolo spomenuté, Google sprístupnil Android všetkým výrobcam zadarmo. Z toho však plynulo, že každý výrobca na základnú verziu Androidu pridal ďalšiu funkcionálnosť a Kitkat na zariadení od Samsungu, tak nebude to isté ako Kitkat na zariadení od HTC.

1.1 Architektúra Androidu

V tejto kapitole bude v skratke vysvetlená architektúra a fungovanie Android systému. Operačný systém Android sa skladá zo softwarových komponent, ktoré je možné rozdeliť do piatich kategórií. Na spodu pomyselné pyramídy sa nachádza Linuxové jadro, na vyššej úrovni sa nachádzajú knižnice a ART emulátor. Nasleduje aplikačný framework a na samom vrchu sa nachádzajú samotné aplikácie. [4,5]



Obrázok 4. Architektúra systému Android.[4]

Brané od spodu pyramídy. Ako prvá vrstva systému pracuje Linuxový Kernel, teda jadro. Jadro sa stará o komunikáciu medzi hardvérom zariadenia a samotným systémom. V jadre sa nachádzajú všetky ovládače, ktoré zabezpečujú prístup systému k fotoaparátu, klávesnici, displeju atď. Kernel sa taktiež stará o veci, v ktorých je Linux všeobecne dobrý, ako je napríklad spojenie zariadenia so sieťou. [4,6]

Nad jadrom systému pokračujú Android knižnice. Ide o knižnice založené na Jave, ktoré sprístupňujú vývojárom prácu s kľúčovými komponentami zariadenia. Medzi hlavné knižnice patrí:

- Android.app – Sprostredkuje prístup k modelu aplikácie a je hlavným stavebným kameňom každej aplikácie.
- Android.content – Sprostredkuje komunikáciu medzi aplikáciou a jednotlivými komponentami systému.
- Android.database – Používa sa na prácu so vstavaným databázovým systémom SQLite.
- Android.opengl – Java rozhranie na 3D grafické renderovanie.
- Android.view – Obsahuje základné stavebné bloky aplikácie potrebné na zobrazenie užívateľského rozhrania.
- Android.webkit – Sada tried, ktoré pre aplikácie sprostredkujú prístup k webovým službám.

Tretou komponentou, ktorá leží tak ako Android knižnice na druhej vrstve systému je už spomínaný ART. ART obsahuje kľúčový prvok tzv. Dalvik Virtual Machine čo je Java virtuálny stroj špeciálne upravený a optimalizovaný pre Android. Tento virtuálny stroj používa hlavné funkcie Linuxového jadra, ako je správa pamäte alebo spracovanie viacerých vlákien, ktoré sú pre jazyk Java bežné. Každá aplikácia, ktorá na systéme beží potom existuje vo svojom vlastnom procese s vlastnou inštanciou Dalvik Virtual Machine. ART v neposlednom rade obsahuje sadu knižníc, ktoré umožňujú vývojárom Android aplikácii písať aplikácie v štandardnom Java jazyku.

Na tretej vrstve sa nachádza aplikačný framework, ktorý ponúka mnoho Java knižníc a služieb na vyššej úrovni abstrakcie. Vývojári sú potom schopní tieto funkcie a knižnice použiť vo svojich aplikáciách. Medzi tieto knižnice patrí napríklad:

- Activity manager – Ovláda životný cyklus aplikácie.
- Content providers – Prostredkuje komunikáciu medzi aplikáciami.
- Notifications Manager – Umožňuje aplikácii zobrazovanie notifikácií.

Nakoniec na najvyššej štvrtej vrstve Android systému sa nachádzajú samotné aplikácie. Architektúra a životný cyklus aplikácie bude rozobraný v samostatnej kapitole. [6,7,16]

1.2 Architektúra Android aplikácie

Každá natívna Android aplikácia je písaná v Java programovacím jazyku. Android SDK (Software Development Kit) potom aplikáciu skompiluje spolu so všetkými dátovými zdrojmi, ktoré aplikácia k svojmu behu potrebuje a vytvorí z nej APK súbor. APK súbor je balíček, ktorý obsahuje všetky súčasti, ktoré systém potrebuje, aby mohol danú aplikáciu nainštalovať a používať. Nainštalovaná aplikácia potom ďalej beží vo svojom vlastnom procese. Každá aplikácia má prístup iba ku komponentom systému a zariadenia, ktoré potrebuje ku svojej činnosti. Týmto vzniká bezpečné prostredie, v ktorom jednotlivé aplikácie nemajú prístup k službám, ktoré nepotrebujú. Každá aplikácia má taktiež pridelené jedinečné Linux id, ktoré ju oprávňuje k prístupu k svojim vlastným súborom, pričom ostatné aplikácie s rozdielnym ID k daným súborom nemajú prístup.

Každá aplikácie sa skladá zo základných stavebných prvkov, ktoré sa nazývajú aplikačné komponenty. Každá zo základných komponent je rozdielnym vstupom, cez ktorý môže systém vstupovať do aplikácie. Nie každý s týchto prvkov je používaný daným užívateľom a niektoré prvky sú na sebe navzájom závislé, avšak každý z nich je jedinečný a v aplikácii má svoje nezastúpiteľné miesto. Celkovo existujú štyri základné prvky každej aplikácie. Každý z nich plní rozdielny účel a má rozdielny životný cyklus, ktorý hovorí ako je prvok vytvorený, používaný a následne zrušený.

Prvým z nich je aktivita (Activity). Aktivita je samostatná obrazovka, ktorá zobrazuje užívateľské rozhranie. Napríklad aplikácie na správu kontaktov môže mať jednu aktivitu (obrazovku), ktorá zobrazuje kontakty, ďalšiu aktivitu, ktorá umožňuje pridanie nového kontaktu atď. Tieto rozdielne aktivity v rámci jednej aplikácie však pracujú spolu, aby vytvorili ucelený užívateľský zážitok. Ak to daná aplikácie dovoľí, iná aplikácia môže spustiť aktivitu správy kontaktov a podobne. Pokiaľ chcem pri vkladaní kontaktu vložiť zároveň fotku, aplikácia na správu kontaktov môže spustiť aktivitu aplikácie na správu fotiek a podobne.

Ďalším prvkom je služba (Service). Ide o komponentu, ktorá na rozdiel od aktivity pracuje na pozadí aplikácie, kde prevádza operácie náročné na čas alebo prácu na vzdialených procesoch. Služba teda nezobrazuje žiadne užívateľské rozhranie. Používa sa napríklad na prehrávanie hudby na pozadí, kým beží úplne iná aplikácia, prípadne sťahuje dáta cez internet, pričom inak neblokuje interakciu užívateľa s bežiacou aktivitou. Iné prvky ako napríklad aktivita potom môže službu spúšťať, rušiť a prijímať jej výsledky a zobrazovať ich ďalej užívateľovi.

Tretím základným prvkom sú tzv. Content providers, čo by sa dalo voľne preložiť ako poskytovateľ obsahu. V jednoduchosti by sa dalo povedať, že Content provider spravuje zdieľané dáta danej aplikácie. Tieto dáta môžu byť uložené v súborovom systéme, v SQLite databázy, či na vzdialenom webovom úložisku alebo inom dátovom úložisku, na ktoré má daná aplikácia prístup. Cez Content provider môžu aplikácie pristupovať prípadne modifikovať dané dáta, za predpokladu, že k tomu majú povolenie. Ako príklad by sa dala znova uviesť správa kontaktov. Android poskytuje content provider, ktorý umožňuje prístup ku kontaktom uloženým na mobilnom zariadení. Tým pádom môže každá aplikácia s povolením čítať a ďalej používať, alebo spracovávať užívateľové kontaktné dáta. Content provider sa používa taktiež na čítanie a zapisovanie dát, ktoré slúžia iba pre aplikáciu, ktorá ich zapisuje.

Posledným avšak nie menej dôležitým prvkom je Broadcast receiver. Úlohou Broadcast receiveru je prijímať a spracovávať systémové oznámenia. Medzi takéto oznámenia môže patriť napríklad oznámenie o zablokovaní obrazovky, stave batérie prípadne o tom, že bola fotoaparátom zachytená snímka. Receiver teda odchyť takéto oznámenia a umožní na ne aplikácii vhodne reagovať. Funguje aj opačným smerom, napríklad môže aplikácii oznámiť ostatným aplikáciám a systému, že sťahovanie dát bolo dokončené a dáta sú momentálne uložené v súborovom systéme a ostatné aplikácie k nim môžu pristupovať. Receiver nezobrazujú užívateľské rozhranie, avšak môžu užívateľa informovať formou status barov o tom, že nastala taká či onaká situácia alebo systémové hlásenie.

Unikátnou vlastnosťou Android systému je to, že aplikácia môže spustiť prvok inej aplikácie. Keď chce vývojár vo svojej aplikácii napríklad zachytiť fotku pomocou fotoaparátu zariadenia nemusí vyvíjať aktivitu, ktorá fotku spraví, môže jednoducho použiť funkcionality inej aplikácie, v tomto prípade aplikácie fotoaparát. Nie je pritom potrebné inak aplikácie prepájať, skratka stačí spustiť aktivitu aplikácie fotoaparát na zachytenie snímky. Po zachytení fotky je dokonca fotka poslaná ako návratová hodnota

späť do vašej aplikácie. Užívateľovi však celá táto interakcia pripadá ako plynulý chod jednej aplikácie. Táto interakcia neprebíha priamo medzi jednotlivými aplikáciami. Ako už bolo spomenuté každá aplikácia beží v samostatnom procese a má obmedzený prístup k systémovým prvkom. Pokiaľ teda chceme využiť funkcionality inej aplikácie musíme poslať správu systému, ktorý sprostredkuje interakciu s druhou aplikáciou a späťne.

Tri zo štyroch spomenutých prvkov – aktivita, služba a broadcast receiver- sú aktivované asynchrónnou správou, ktorá sa nazýva intent (zámer). Intent spája jednotlivé komponenty či už v rámci jednej aplikácie alebo medzi viacerými. Dá sa naň pozerat' ako na poslov, ktorý vyžadujú služby od iných Android prvkov. Intent vytvára intent objekt, ktorý definuje správu k aktivácii špecifického prvku alebo špecifického typu prvku. Pre aktivity a služby intent definuje činnosť, ktorá sa má vykonať, napríklad niečo poslať prípadne zobrazit' a taktiež dáta, ktoré má k tomuto použiť. Aplikácia môže napríklad vytvorit' intent na zobrazenie určitej webovej stránky. A naopak, niekedy chceme začať aktivitu, aby sme dostali späť výsledok. Môžeme teda vytvorit' intent, v ktorom si užívateľ vyberie kontakt z adresára. Tento intent sa potom vráti aplikácii spoločne s URI, ktorá ukazuje na vybraný kontakt. Pre broadcast receive intent definguje oznámenie. V prípade slabej batérie teda príde intent, ktorý bude obsahovat' reťazec znakov „battery is low“. Posledný prvok, content provider, nie je aktivovaný intentom. O jeho aktiváciu sa stára tzv. ContentResolver a to v dobe požiadavky na prístup k dátam. ContentResolver slúži ako určitá abstrakcia medzi danými dátami a content providerom. Táto abstrakcia vytvára bezpečnostný prvok keď sa o prístup k dátam stará ContentResolver a nie jednotlivé content providere. Tým je zachovaná integrita dát.

Predtým ako môže Android systém spustiť jeden zo štyroch základných prvkov, musí najskôr zistiť, či daný prvok existuje tým, že prečíta tzv. Android manifest. Android manifest je uložený v AndroidManifest.xml súbore. Každá aplikácia v tomto súbore deklaruje všetky svoje prvky. Tento súbor musí byť umiestnený vždy v root zložke aplikácie. Okrem toho, že deklaruje jednotlivé prvky manifest ďalej:

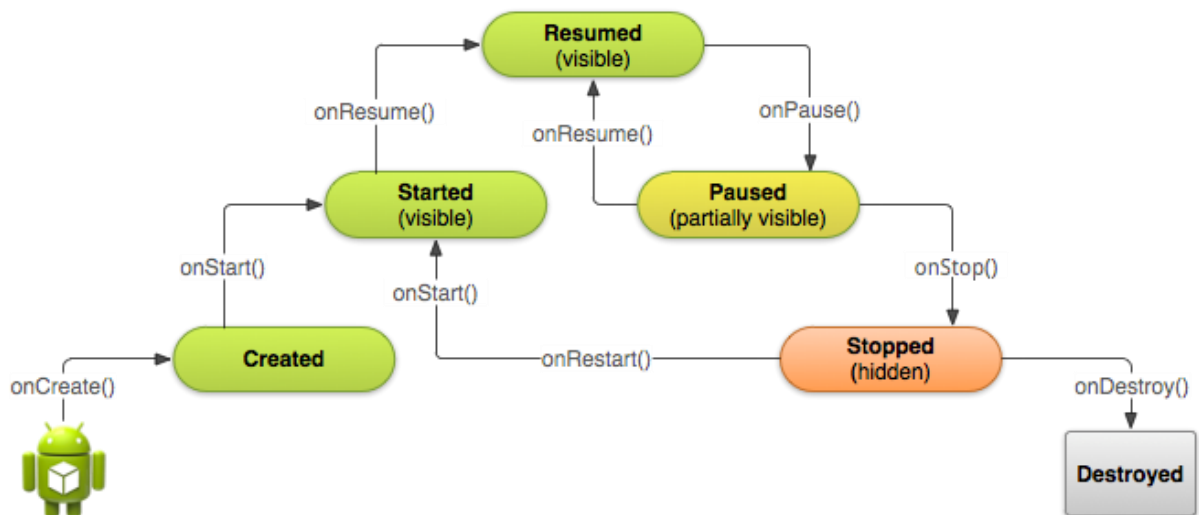
- Zaznamenáva všetky užívateľské povolenia, ktoré daná aplikácia vyžaduje (prístup na internet, prístup ku kontaktom).
- Deklaruje minimálne API ktoré aplikácia potrebuje ku svojej činnosti.
- Deklaruje hardverové a softverové funkcie, ktoré sú potrebné prípadne používané aplikáciou (fotoaparát, bluetooth, gps).

- Knižnice, ktoré aplikácia potrebuje ku svojej činnosti (okrem Android framework APIs).

Aktivity, služby či content providere, ktoré aplikácia používa avšak nie sú deklarované v manifeste, systém nerozpozná a tým pádom nemôžu byť spustené. Jediná výnimka sú broadcast recievere, ktoré môžu byť deklarované v manifeste, avšak taktiež môže byť inicializované dynamicky v kóde a následne registrované v systéme metódou `registerReceiver()`.

1.2.1 Životný cyklus aktivity

V minulej kapitole boli popísané štyri základné prvky aplikácie. Tým najdôležitejším a nepostrádateľným pre každú aplikáciu je však aktivita. Na rozdiel od iných programovacích vzorov, kde sú aplikácie spúšťané z metódy `main()`, Android systém spúšťa kód z inštalácie aktivity vyvolávaním špecifických metód, ktoré súvisia so špecifickým bodom v životnom cykle aktivity. V tejto kapitole bude tento cyklus popísaný. Existuje sekvencia metód, ktoré sú volané pri vytvorení aktivity a taktiež sekvencia, ktorá sa volá pri jej rušení. V priebehu života aktivity je volaná sada metód v poradí podobnému pyramíde (viz. *Obrázok 5*). Každý krok v životnom cykle aktivity je rozdielny krok v tejto pyramíde. [8,9]



Obrázok 5. Životný cyklus aktivity.[8]

Ako jesystémom vytváraná nová inštalácia aktivity, každá metóda posúva aktivitu o krok vyššie v pyramíde až na jej vrchol. Vrchol pyramídy je bod, kedy aktivity beží na pozadí

a uživatel s ňou nemůže pracovat. V momente ako je daná aktivita opustená užívateľom, systém začne pomocou série metód posúvať aktivitu smerom z vrchu dole za cieľom aktivitu zničiť. V niektorých prípadoch je aktivita posunutá iba do stavu „Paused“, teda do stavu pozastavená. To môže nastať napríklad v prípade, keď je užívateľom otvorená iná aplikácia a súčasná aktivita je tak presunutá na pozadie. Tam potom čaká na volanie ďalšej metódy. Pokiaľ bude ďalšia metóda `onStop()`, teda aplikácia je užívateľom vypnutá, aktivita pokračuje dole pyramídou k svojmu zničeniu. Pokiaľ však užívateľ naopak znovu posunie aktivitu na popredie, volá sa metódy `onResume()` a aktivita sa posúva opäť na vrchol pyramídy.

Vzhľadom na zložitosť aplikácie potom nie je potrebné implementovať všetky metódy životného cyklu. Avšak implementáciou týchto metód je zaistené fungovanie aplikácie, tak ako by to užívateľ očakával a to napríklad:

- Aplikácia nespadne v momente, keď užívateľ prijme hovor alebo otvorí inú aplikáciu.
- Aplikácia nezaberá systémové zdroje v momente, keď ju užívateľ aktívne nepoužíva.
- Aplikácia nestratí kontext v momente, keď ju užívateľ odsunie na pozadie a neskôr opäť používa.
- Aplikácia nespadne v momente, keď užívateľ rotuje z obrazovkou.

Iba tri zo stavov zobrazených na *obrázku 5* sú stavy statické, teda také v ktorých môže aplikácia pretrvávajúť dlhší čas.

- Resumed voľne preložené bežiaci aktivita. V tomto bode je aktivita na popredí a užívateľ s ňou môže interagovať.
- Paused (zapauzovaná) - V tomto stave je aktivita čiastočne zakrytá inou aktivitou, ktorá je momentálne na popredí. Zapauzovaná aktivita neprijíma žiadne užívateľské vstupy a nemôže vykonať žiadny kód.
- Stopped (zastavená) – Tu už je aktivita kompletne skrytá za inou aktivitou a nie je pre užívateľa viditeľná. Beží teda iba na pozadí. V tomto momente sú všetky stavové informácie a užívateľské premenné zachované, ale nie je vykonávaný žiadny kód.

Ostatné dva stavy (Created a Started) sú stavy prechodné a systém ich volá po sebe v rýchлом slede. Keď je teda užívateľom otvorená aplikácia systémom je volaná onCreate() metóda aktivity, ktorá je v manifeste aplikácie definovaná ako „launcher“ aktivita, teda aktivita, ktorá má byť spustená ako prvá pri spustení aplikácie. Táto aktivita teda slúži ako vstupný bod do aplikácie. Pri spustení ďalšej aktivity v rámci aplikácie sa potom opäť volá onCreate() metóda príslušnej aktivity. Metóda onCreate() teda musí byť implementovaná pre každú aktivitu. V tejto metóde sú zvyčajne inicializované členské premenné a je nastavený „Layout“ teda rozloženie aktivity, ktoré je použité. Poslednou metódou pre každú aktivitu je metóda onDestroy(), ktorá je volaná pri úplnom zničení aktivity. Táto metóda nemusí byť nutne implementovaná, pretože veľká časť „upratovania“ tried a premenných prebieha už pri onPause() a onStop() metódach. Každopádne, pokiaľ sú aktivitou používané vlákna na pozadí, prípadne iné zdroje, pri ktorých môže potenciálne viesť k pretečeniu pamäte je dobré aby boli tieto zdroje dealokované práve v metóde onDestroy(). [8,9,14]

1.2.2 Moduly Android aplikácie

Moduly Android aplikácie sú súčasťou, z ktorých je nakoniec pri builde vytvorený .apk inštalateľný súbor danej aplikácie. Moduly sa skladajú hlavne zo zdrojového kódu a zdrojov aplikácie. Väčšina týchto modulov je pre vývojára generovaná automaticky vývojovým prostredím, zatiaľ čo iné musia byť vytvorené. Nasledujúce súbory a priečinky tvoria moduly Android aplikácie:

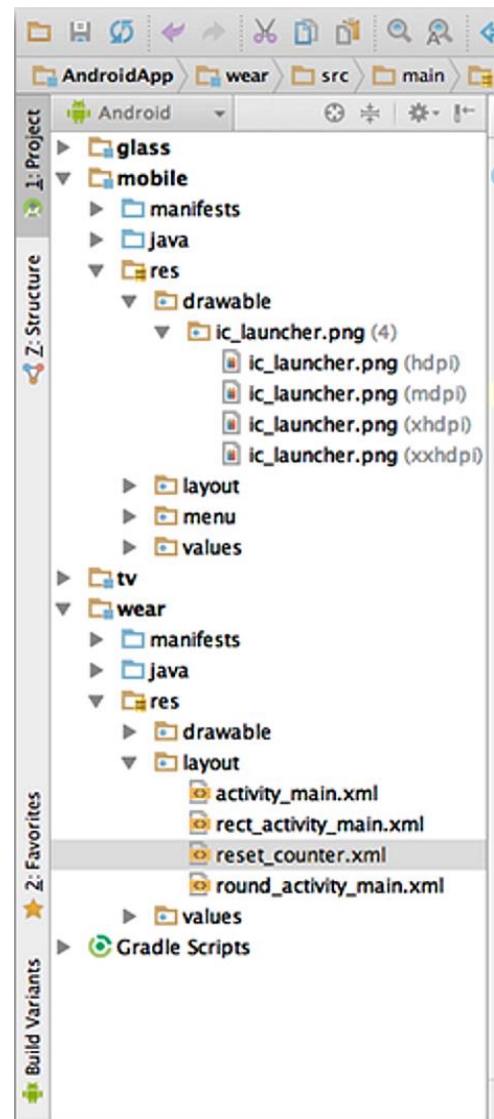
- build/ - Súbory obsahujúce informácie o builde aplikácie.
- libs/ - Privátne knižnice.
- src/ - Súbor koreňovej aktivity.
- androidTest/ - Súbory na testovanie zariadenia.
- main/java/com.<project>.<app> - Java zdrojové súbory pre aktivity aplikácie.
- main/assets/ - Tento priečinok je prázdny. Používaný je na ukladanie zdrojov aplikácie. Súbory uložené v tomto priečinku sú následne skompilované do .apk súbory v podobe akej sú v súbore vložené.
- main/res – Obsahujú ďalšie zdroje aplikácie:
 - o anim/ - XML súbory ktoré sú skompilované na animácie.

- color/ - XML súbory ktoré popisujú farby.
- drawable/ - Pre bitmapové súbory (PNG,JPG,GIF).
- mipmap/ - Ikony používané pri štarte aplikácie.
- layout/ - XML súbory popisujúce rozloženie prvkov v aktivite.
- menu/ - XML súbory popisujúce menu aplikácie.

A v neposlednom rade už spomenutý a opísaný AndroidManifest.xml. Týmto členením je v rámci Android aplikácie podporená modulárnosť celého systému. Jednoznačne je totiž oddelená logika aplikácie (backend) teda Java súbory pre dané aktivity a vzhľad danej aplikácie (frontend), teda hlavne XML súbory popisujúce vzhľad aplikácie. Je teda jednoducho možné zmeniť XML súbor, a tým je zmenený vzhľad určitej aplikácie avšak logikaktorá pracuje na pozadí je zachovaná. A taktiež naopak je možné zmeniť výpočty alebo logiku akou sa pristupuje na sieť bez toho, aby bol akokoľvek zmenený vzhľad. Vývojári su taktiež Androidom tlačení k používaniu spomenutého main/res priečinka na všetky zdroje v aplikácii. Farby, textové reťazce, obrázky či animácie sú uložené na jednom mieste a následne cez ich ID používané v rámci aplikácie. Keď je potom potrebné zmeniť farbu pozadia, prípadne zmeniť názov položky v menu, stačí mu túto zmenu previesť na jednom mieste a zmena sa automaticky premietne do celej aplikácie. [8,9,10,15]

1.3 Vývojové prostredie

V začiatkoch Android systému sa na vývoj používalo niekoľko vývojových prostredí, ktoré neboli špeciálne usporiadané na vývoj Android aplikácii. Týmto trpela hlavne kvalita daných aplikácii a tým aj povest' celého Android systému.



Obrázok 6. Štruktúra projektu v AS.[11]

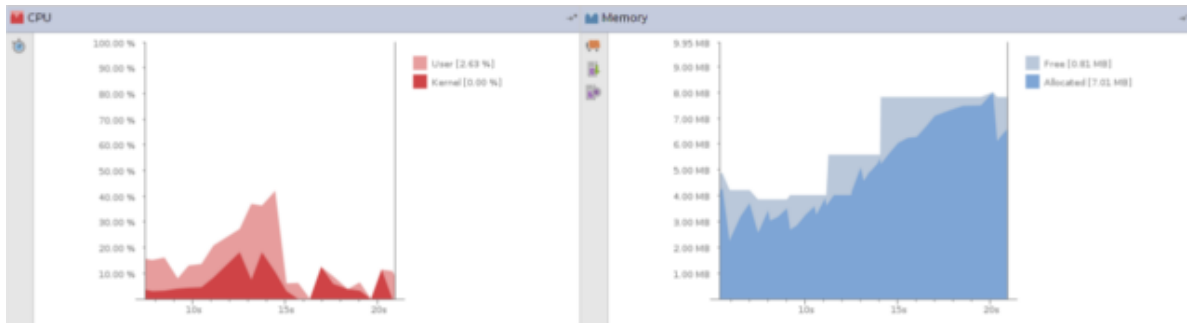
Preto sa Google rozhodol zakročiť. Bola zvolená rovnaká stratégia, ktorá už predtým slávila úspech v rámci celého Androidu. Odkúpil teda of firmy IntelliJ IDEA ich vývojové prostredie a postavil na ňom úplne nové vývojové prostredie špecializované na vývoj Android aplikácii. Plná verziu nového Android Studio následne bola sprístupnená všetkým ako freeware. Tým pádom dostali vývojári robustné vývojové prostredie špecializované na vývoj aplikácií a podľa jednotnej normy. A Google dostal naspäť o mnoho kvalitnejšie aplikácie. Samotné Android studio potom vývojárov tlačí k používaniu „best-practices“ k vývoju kvalitných aplikácii. Každý projekt sa v Android studiu skladá z modulov popísaných v predchádzajúcej kapitole. Zobrazené sú v ľavej lište v Android projekt prieskumníku. Tento prieskumník umožňuje rýchly prechod medzi jednotlivými modulmi. Na *obrázku 6* je vidieť členenie projektu. Na najvyššej úrovni sú priečinky, ktoré delia projekt na rozdielne zariadenia (mobil, tv ...), každý tento priečinko potom obsahuje tri základné priečinky a to manifests, ktorý obsahuje Manifest súbory danej aplikácie. Priečinko java v ktorom sú všetky Java triedy a Java súbory a priečinko res, v ktorom sú obsiahnuté všetky zdroje aplikácie. Na najvyššej úrovni sa taktiež nachádza priečinko Gradle Scripts, ktorý obsahuje súbory potrebné k buildu. Android studio taktiež obsahuje šablóny na vývoj bežnej funkcionality a grafický „drag and drop“ editor na návrh designu aplikácie. Nástroje na refaktorovanie kódu alebo zmenšovanie a komprimovanie zdrojov aplikácie. A v neposlednom rade taktiež nástroje na testovanie a debugovanie aplikácií.

Na build aplikácií používa Android studio Gradle systém, ktorý je obohatený o špecifické funkcie pre Android. Tento Gradle systém funguje ako integrovaný nástroj v rámci Android studia a taktiež samostatná jednotka pomocou príkazového riadku. Medzi funkcie systému patrí:

- Prispôsobovanie a konfigurácie build procesu.
- Vytváranie rozličných .apk súborov pre aplikáciu s rozličnými funkciami z toho istého projektu a za použitia tých istých modulov.
- Znovu používania kódu a zdrojov.

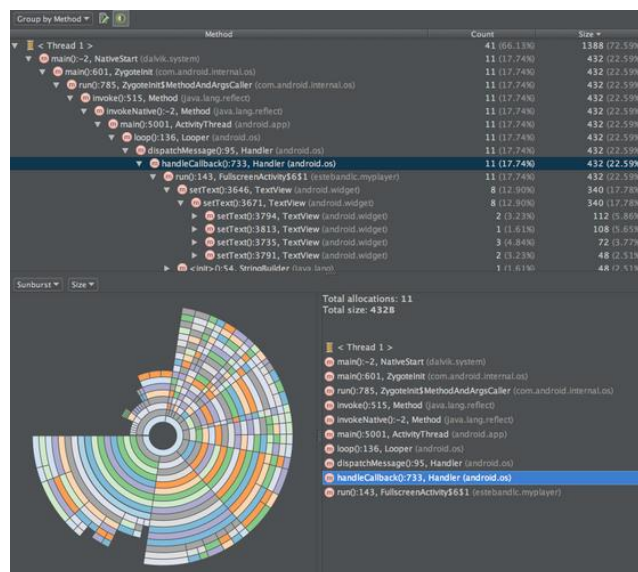
Flexibilita Gradlu umožňuje všetko tieto funkcie bez zásahu do zdrojových súborov aplikácie.

Ďalšou dôležitou súčasťou Android studia sú nástroje na debugovanie a monitorovanie výkonu aplikácie. Debugger Android studia umožňuje sledovať hodnoty premenných za behu aplikácie, návratové hodnoty metód, lambda a operátorové výrazy a mnoho ďalších vecí. Taktiež je možné monitorovať výkon procesoru a pamäte zariadenia. Týmto je možné hľadať dealokované a nepoužívané objekty, lokalizovať pretečenia pamäte alebo nájsť miesta v kóde, na ktorých aplikácie strávi najviac výpočetného času (bottle neck).



Obrázok 7. Monitorovanie výkonu v Android studiu.[12]

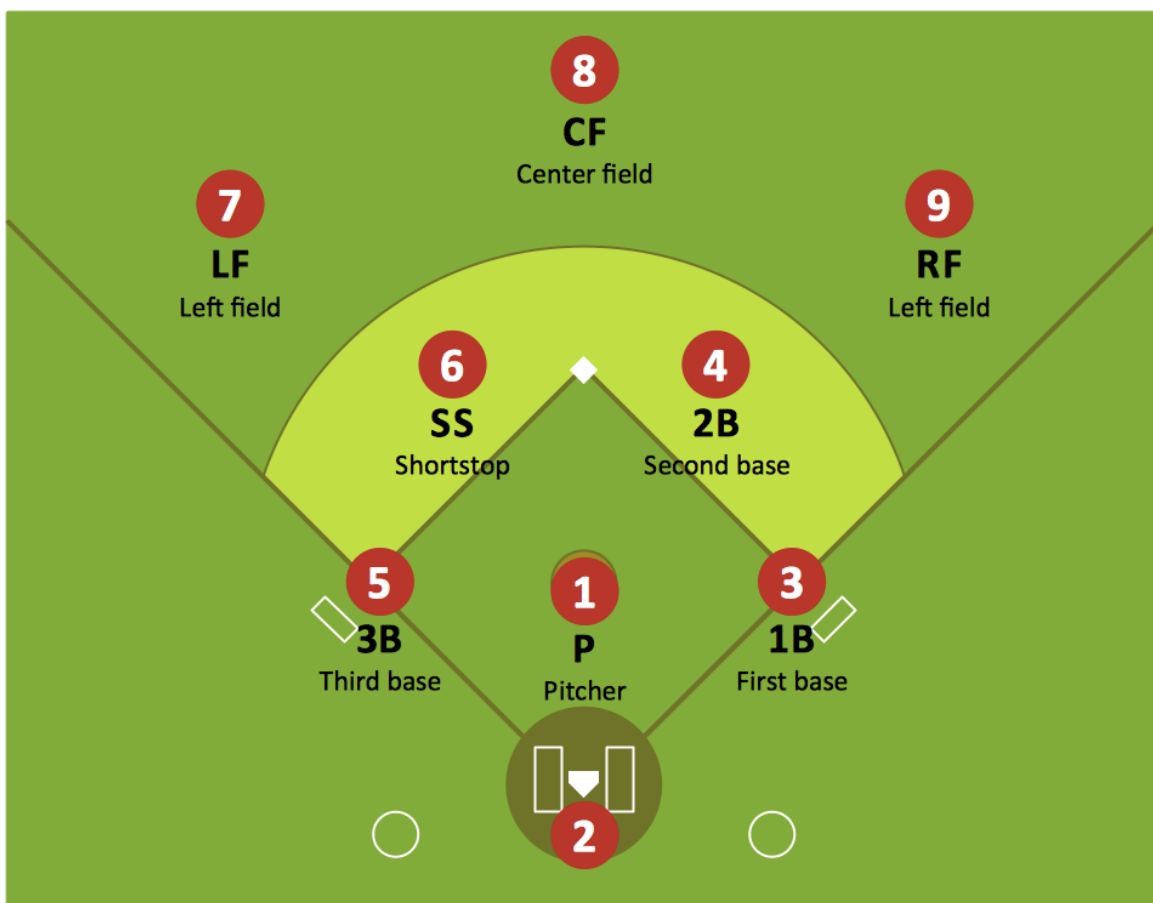
Skvelým nástrojom, ktorý Android studio ponúka je tzv. Allocation tracker. Tento nástroj umožňuje sledovať pridelenie pamäte jednotlivým zdrojom. Toto vývojárom umožní sledovať pridelenia pamäte ako následok vykonania určitej akcie. Z vedomosťou tohoto pridelenia dovoľí vývojárovi upraviť volania metód tak, aby optimalizoval výkon a využitie pamäte. Všetky tieto informácie sú zobrazované prehľadne v stromovej štruktúre vlákna a súčasne v grafickom zobrazení (viz obrázok 8). [11,12]



Obrázok 8. Allocation tracker v Android Studiu.[12]

2 ZÁZNAM ŠPORTOVÝCH ZÁPASOV

Softbal je loptová hra, ktorú hrajú dve mužstvá o deviatich hráčoch. Jeden tím vždy útočí, takže sa jeho hráči striedajú pri odpalovaní a druhý bráni. Jeho hráči sú rozložený v poli a snažia sa odpálenú loptu chytiť. Pravidlá softballu sa v mnohom nelíšia od tých baseballových. Hlavnými zmenami je veľkosť ihriska (softbalové je menšie), veľkosť pálky a lopty (softbalové sú väčšie), spôsob nadhodu a pozície bežca pri odpale. Páľkari, teda hráči útočiaceho tímu, podľa predom stanoveného poradia nastupujú do páľkarského územia (šedý kruh nad pozíciou č. 2). Následne nadhadzovač brániaceho tímu nadhadzuje loptu na páľkara pričom nadhoz musí prejsť tzv. strike zónou čo je pomyselný obdĺžnik medzi páľkarovými kolenami a ramenami.



Obrázok 9. Schema softbalového ihriska.[27]

Úlohou pálkara je potom loptu odpáliť do priestoru ihriska. Po úspešnom odpale sa rola pálkara mení na rolu bežca. Na vrcholoch kosoštvorca na obrázku sú umiestnené jednotlivé méty. Štvrtkruh nad týmto kosoštvorcom je vnútorné pole. Štvrtkruh nad vnútorným polom potom tvorí vonkajšie pole. Nadhadzovač (pitcher) nadhadzuje zo svojej pozície (1) smerom na pálkara pričom chytač (2) musí loptu chytiť. Poliar 3,4,5,6 sú rozmiestnený v rámci vnútorného pola. Poliar 7,8,9 v rámci vonkajšieho pola, ako je vidieť na *obrázku 9*. Presné pozície poliarov vo vonkajšom a vnútornom poli sa môžu mierne líšiť, ich postavenie nie je pevne dané.

Po odpale bežec postupuje po obode vnútorného pola čo najďalej, avšak minimálne na prvú métu. Ďalšia méty potom získava pálkár priamo po svojom odpale prípadne po odpale nasledujúceho pálkara z jeho tímu, na každej méte môže byť iba jeden hráč. Bežec, ktorý prebehne všetky štyri méty získava pre svoje družstvo jeden bod. Nadhadzovač nadhadzuje loptu zo svojej méty spodom jednou rukou s vykročením jedným krokom smerom k pálkarovi. Pokiaľ nadhodená lopta prejde cez strike zónu, a pálkarovi sa ju nepodarí odpáliť, získava pálkár jeden strike. Strike získava tak isto v prípade, že lopta neprechádza cez strike zónu a pálkár sa po nej zaženie. V momente troch strikov je pálkár vyoutovaný a na pátku nastupuje ďalší hráč jeho tímu. Každý nadhod, ktorý neprechádza strike zónou je potom hlásený ako ball a v prípade piatich ballov, teda chybných nadhodou získava pálkár prvú métu zdarma. Pri situácii keď je pálkár bez vlastného zavinenia trafený nadhodom získava prvú métu automaticky zdarma. V prípade, že je prvá méta obsadená hráči na ostatných metách sa posunú tak, aby pálkár mohol prejsť na prvú métu.

Po úspešnom odpale sa brániaci tím snaží bežca vyoutovať. A to hlavne tak, že príhodou dostane odpálenú loptu na métu skôr ako k nej dobehne bežec, prípadne sa loptou dotkne bežca, ktorý sa nenachádza na méte. V prípade, že pálkár odpáli loptu priamo za vonkajšie pole, a obránci tak loptu nemôžu chytiť, môže pálkár obehnúť všetky méty a získať pre svoj tím automaticky bod. V prípade, že loptu brániaci tím chytí priamo zo vzduchu je pálkár out. Po vyradení (vyutovaní) troch súperov si tími menia role. Brániaci tím ide na pátku a naopak. Zápas býva spravidla omedzený siedmi zmenami (každé družstvo je sedem krát v poli a sedem krát na pálke). V prípade nerozhodného výsledku po siedmych zmenách, sa nastavuje vždy jedna zmena až do dosiahnutia rozhodného stavu. Pravidlami hry je predpísané iba postavenie dvoch hráčov. Nadhadzovač sa v okamihu zahájenia nadhodu musí dotýkať oboma nohami nahadzovacej méty. A zadák musí byť v okamihu zahájenia nadhodu v svojom území. Pre ostatných poliarov platí iba pravidlo, že musia byť

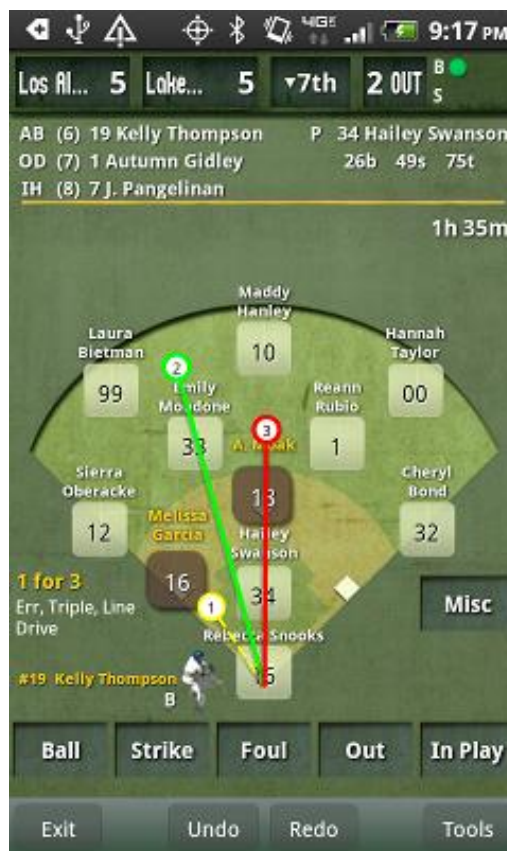
pri zahájení nadhodu v poli. Bežci na métach sa potom v dobe pred odpalom musia nachádzať maximálne jeden meter od svojej méty na rozdiel od baseballu, kde bežec týmto nie je obmedzený. Záverečné skóre je potom súčet bodov (dobehov hráčov), ktoré získal daný tím v jednotlivých zmenách. [19,20]

2.1 Existujúce Android aplikácie pre záznam športového zápasu

Existuje niekoľko aplikácií, ktoré umožňujú zápis softballového prípadne baseballového zápasu. Medzi najznámejšie patria aplikácie iScore a Scorefinger. Aplikácia iScore je platená a aplikácia Scorefinger je voľne stiahnuteľná.

2.1.1 iScore

Aplikácia je vyvinutá na všetky tri hlavné operačné systémy (Android, iOS a Windows phone). iScore umožňuje vytvorenie tímov, hráčov a samostatných líg. Na obrázku 10 je vidieť hlavnú obrazovku aplikácie, ktorá slúži na zápis samotného zápasu. V spodnej lište sa nachádzajú tlačítka, ktoré umožňujú späťne zmazať zapísané kroky prípadne ukončiť zápas.



Obrázok 10. Obrazovka pre záznam.[21]

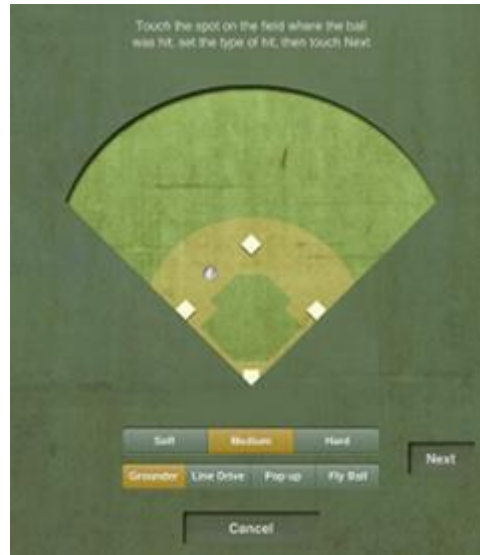
zápisu iScore.

Nad touto lištou sa nachádzajú tlačítka, ktoré slúžia k samotnému zápisu zápasu. Každé z tlačítiek otvára dodatočný samostatný dialóg, v ktorom sa upresňuje situácia ktorá nastala. Na *obrázku 11* je vidieť dodatočné menu, ktoré sa otvorí pri stlačení tlačidla out. Podobný dialóg nastáva aj pri ostatných tlačidlách. Zapísanie každej situácie, teda aj jednoduchého striku vyžaduje niekoľko krokov pričom zapisovateľ musí mať znalosti v oblasti situácií, ktoré môžu pri hre nastať.



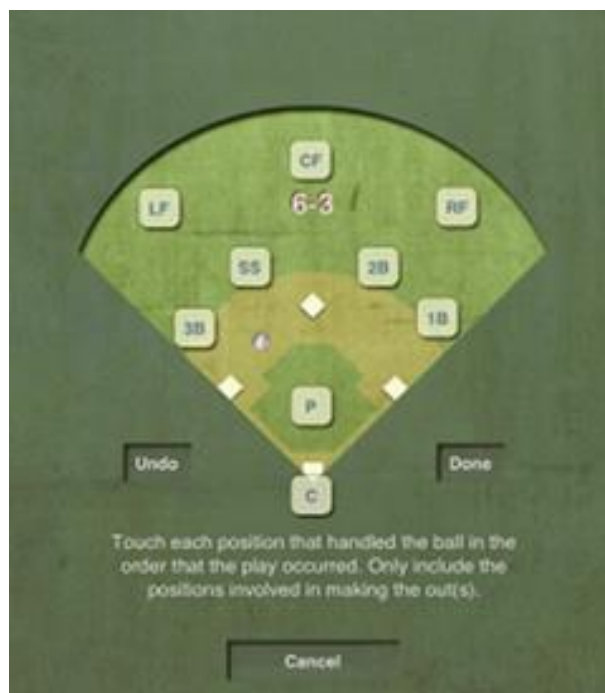
Obrázok 11. Dodatočná tabuľka na zápis vyoutovania hráča. [21]

V momente, kedy sa dostane lopta do hry, je následne potrebné zaznamenať pozíciu, do ktorej bola lopta odpálená. Po zaznamenaní tejto pozície sa pokračuje stlačením tlačítka Next na ďalšiu obrazovku.



Obrázok 12. Obrazovka pri odpale lopty. [21]

Následne sa prechádza na ďalšiu obrazovku (obrázok 13), na ktorej je potrebné zaznamenať pohyb lopty po odpale. Pokiaľ pred odpalom neboli žiadny bežci na métach, zápis pre daný odpal je ukončený. V opačnom prípade sa prechádza na obrazovku, kde sa bežci na jednotlivých métach posunú na svoje nové pozície.



Obrázok 13. Obrazovka k záznamu pohybu lopty. [21]

Táto obrazovka je vidieť na *obrázku 14*. Na nej je potom potrebné určiť čo sa stalo s bežcami na jednotlivých métach.

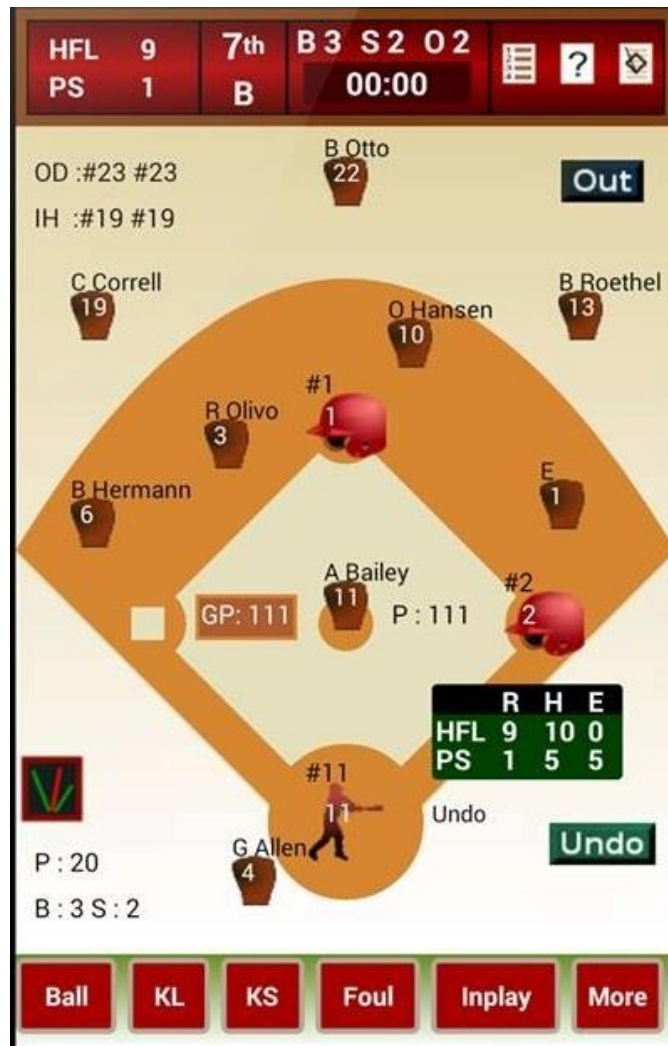


Obrázok 14. Obrazovka pre záznam pohybu bežcov.[21]

Aplikácie iScore ešte ďalej umožňuje zaznamenať polohu a druh nadhodu. Ako je vidieť z predchádzajúcich strán, iScore je komplexnou aplikáciou. Na zápis hernej situácie je potrebných mnoho relatívne zložitých krokov. Zapisovateľ, preto musí mať skúsenosti so samotnou hrou a taktiež so zápisom pomocou tejto aplikácie. [21]

2.1.2 ScoreFinger

Druhou rozšírenou aplikáciou pre Android je aplikácia ScoreFinger. Aplikácie tak ako iScore umožňuje vytváranie tímov, hráčov a samostatných líg ako aj export výsledkov. Na *obrázku 15* je potom vidieť hlavnú obrazovku. Na vrchu obrazovky sa nachádzajú informácie o súčasnom stave zápasu.



Obrázok 15. Obrazovka pre záznam zápasu Scorefinger. [22]

V strede obrazovky sa nachádza ihrisko s hráčmi v poli. Na spodnej časti obrazovky opäť tlačítka na zaznamenanie herných situácií. Každé tlačítko opäť otvára menu na upresnenie zadania situácie, ktorá nastala. Na obrazovke sa taktiež nachádza špeciálne tlačítko, ktoré dovoľí označiť miesto odpalu a následný pohyb lopty v rámci ihriska. Bežcov je následne možné presunúť na méty. Taktiež sa dodatočne upresňuje situácia vďaka, ktorej sa daný hráč dostal na danú métu. Opäť teda ide o relatívne zložitý zápis hernej situácie, ktorý zaberie relatívne dlhý čas. Scorefinger taktiež ako iScore spracuje záznam zápasu a ponúka prehľadné štatistiky hráčov a tímov. Možné je sledovať štatistiky nadhadzovačov, bežcov, poliarov a taktiež pálkarov. [22]

3 DYNAMICKÉ DATOVÉ ŠTRUKTÚRY

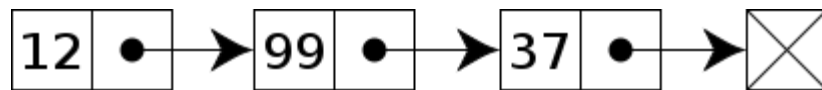
V praktickej časti aplikácie je záznam zápasu ukladaný a spracovávaný pomocou rôznych dynamických dátových štruktúr. V tejto kapitole sú popísané základné dynamické dátové štruktúry. Sú vysvetlené ich výhody, nevýhody a prípady užitia. Vo vytvorenej aplikácii boli tieto štruktúry použité na zaznamenávanie zápasu.

Dynamické dátové štruktúry sú také štruktúry, ktorých obsah sa behom vykonávania programu môže meniť. Ako príklad môže slúžiť obyčajná rada v obchode. Program je potom proces platenia nakupujúcich. Behom tohoto procesu sa rad nakupujúcich znižuje (nakupujúci platia rýchlejšie ako prichádzajú noví), zväčšuje (noví zákazníci prichádzajú rýchlejšie ako sťahujú starí platit') alebo ostáva rovnaká (k pokladne prichádza rovnaký počet nakupujúcich ako od nej odchádza). Takto dynamicky sa dĺžka radu mení behom prevádzania procesu platenia a takto si možno predstaviť dynamickú dátovú štruktúru. Spomínaný rad je iba jedným ľahko uchopiteľným príkladom, v tejto kapitole budú popísané základné dynamické dátové štruktúry. [23]

Prvou a základnou dynamickou štruktúrou je dynamické pole. V Jave, ale aj všeobecne v ostatných programovacích jazykoch má pole statickú predom definovanú veľkosť. Predom sa teda alokuje množstvo pamäte, ktoré potom ostáva nemenné. Dynamické pole je potom pole, ktoré môže meniť svoju veľkosť za behu programu. Pri dynamickom poli sa taktiež alokuje pamäť určitej veľkosti. V prípade je dané pole zaplnené dynamicky sa alokuje pole o dvojnásobnej veľkosti a prekopíruje do neho všetky prvky z poľa starého. Pôvodne pole sa potom dealokuje. V momente, kedy je značná časť poľa nevyužívaná a tým pádom je alokovaná, zbytočná pamäť prebehne tento proces obrátene. Vytvorí sa menšie pole do ktorého sa prekopírujú všetky prvky a ušetríme pamäť. Značnou nevýhodou tohoto postupu je jeho neefektívnosť. V okamžiku, keď dojde miesto totiž musíme všetko skopírovať, asymptotická zložitosť vkladania prvku je preto $O(n)$. Asymptotická zložitosť nám hovorí koľko operácií je potrebné vykonať na vloženie dát o veľkosti (n) . [23]

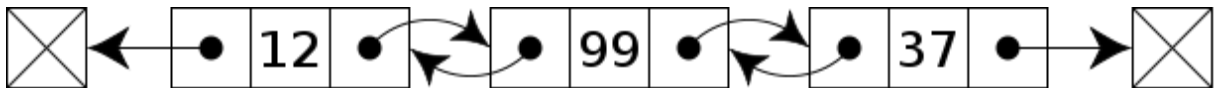
Ďalšou s často používaných dynamických dátových štruktúr je lineárny zoznam. Samotný lineárny zoznam má mnoho variácií a slúži ako základ pre iné dynamické štruktúry. Implementačne je pomerne jednoduchý. Prvky zoznamu sú dynamické premenné rovnakého typu (záznam). Každá položka ukazuje na následníka daného prvku zoznamu. Celý zoznam je potom referovaný ako ukazateľ na prvý prvok v zozname, pomocou

ktorého sme schopný pristúpiť na hociktorý iný. Posledný prvok zoznamu potom často ukazuje na null, teda prázdny prvok a jednoznačne tak identifikuje koniec zoznamu. [23,24]



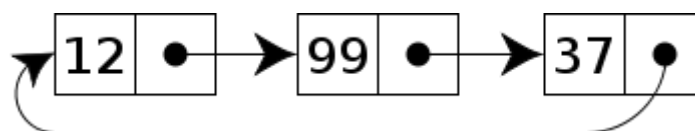
Obrázok 16. Jednosmerný lineárny zoznam.[24]

Ďalšou variantou zoznamu je obojsmerný zoznam. Každý uzol v tejto štruktúre obsahuje samotné dáta a dva ukazatele, jeden na nasledujúci a ďalší na predchádzajúci prvok zoznamu. Prvý a posledný prvok potom opäť ukazujú na null. Touto úpravou zoznamu sa komplikuje implementácia, avšak zoznam je potom možné prechádzať v oboch smeroch. [23,24]



Obrázok 17. Obojsmerný lineárny zoznam[24].

Poslednou implementáciou lineárneho zoznamu je cyklický zoznam. V tejto štruktúre obsahuje každý prvok dáta a jeden ukazateľ na nasledujúci prvok. Posledný a prvý prvok v zozname však ukazujú na špeciálny prvok, ktorý slúži ako prvý a zároveň ako posledný. Týmto sa v podstate dosiahne štruktúra z obrázku 18. Zároveň je potom vloženie na začiatok a koniec a medzi prvky v podstate jedna operácia. Malou nevýhodou tohoto prístupu sú dodatočné pamäťové nároky na udržiavanie špeciálneho prvku. [23,24]

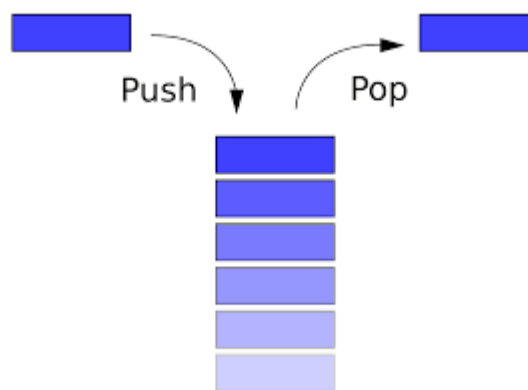


Obrázok 18. Cyklický lineárny zoznam.[24]

Pre efektívne využívanie ľubovoľného typu lineárneho zoznamu je potrebný objekt tzv. iterátor. Iterátor prechádza zoznamom a udržiava odkaz na aktuálnu pozíciu v zozname. Ďalej potom umožňuje prechádzať zoznam, jedným alebo oboma smermi podľa typu zoznamu. Bez takejto štruktúry by bol prechod zoznamom pamäťovo drahou operáciou.

Pri každom prístupe na ďalšiu položku, by sa musel tento prvok najskôr vyhľadať, čo sa rovná prejdenu celého zoznamu od začiatku až po daný prvok. Vkladanie prvku na začiatok zoznamu prebieha s asymptotickou zložitou $O(1)$. Rovná sa v podstate iba alokácii nového prvku a priradeniu ukazateľa. Operácie mazania a čítania na indexe (i) prebehne v čase $O(i)$ nakoľko k danému prvku je potrebné doiterovať cez (i) prvku, lineárny zoznam neponúka náhodný prístup. Z toho jasne plynie že lineárny zoznam je výhodné použiť v prípade že od dátovej štruktúry nie je vyžadované mazanie na náhodnej pozícii či vyhľadávanie prvkov. Medzi typické operácie teda patrí pridanie prvku (Add) a prechádzanie celého zoznamu pomocou iterátora pričom jednotlivé prvky môžeme mazať. [23,24]

Fronta je dátový typ, ktorý už bol spomenutý na začiatku kapitoly v príkladne s obchodom. Je jedným zo základných dátových typov a vychádza z lineárneho zoznamu. Služi k ukladaniu a výberu dát takým spôsobom, že prvý vložený prvok je ako prvý taktiež vybraný. Tomuto princípu sa hovorí FIFO (First In First Out). Špeciálnym vylepšením formy je tzv. prioritná fronta v ktorej má každý prvok určenú prioritu a prvky s vyššou prioritou môžu „predbehnúť“ prvky s prioritou menšou. Fronte je často používaná pri prenose dát cez sieť prípadne pri spracovaní videa či zvuku. V týchto oblastiach služi ako synchronizačné primitívum či kruhový buffer (vyrovnanie pamäte pre dátové toky). Ďalej sa využíva na komunikáciu medzi procesmi v operačných systémoch. [24,25]

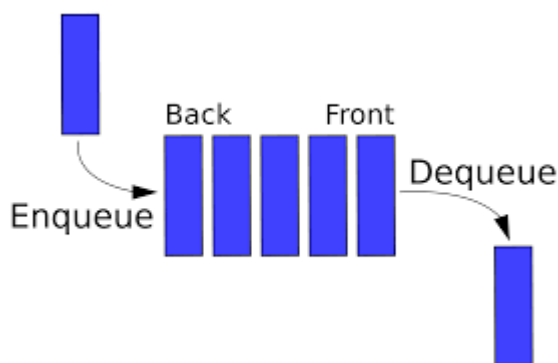


Obrázok 19. Štruktúra typu FIFO.[24]

Medzi typické operácie patrí vloženie prvku do fronty (enqueue) a vybranie prvku z fronty (dequeue). Ďalej nahliadnutie na prvý prvok fronty (peek). A dotazy na veľkosť fronty (size) a zistenie prázdnoty (isEmpty).

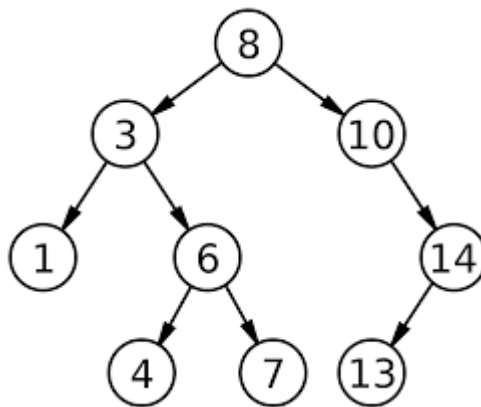
Opakom fronty je dátový typ zásobník. Predstaviť si ho je možné ako zásobník na náboje v strelnej zbrani. Posledný vložený prvok je ten, ktorý je prvý vybraný, teda LIFO (Last In

First Out). Medzi základné operácie patria operácie vloženia (push) a vybraní (pull). Pri vložení sa vkladá prvok na vrchol zásobníka a pri vyberaní sa prvok z vrchola vyberie. Taktiež ako pri fronte sú častými operáciami nahliadnutia na vrchol (top) a dotazy na prázdnosť (isEmpty) a veľkosť (size). V programovaní sa využíva vo všeobecnosti hlavne na dočasné ukladanie dát v priebehu výpočtu. Taktiež pre ukladanie momentálneho stavu algoritmu či programu a implicitne sa používa pri rekurzívnych algoritmoch. Na zásobníkovej architektúre je taktiež postavená celá Java. [24,25]



Obrázok 20. Štruktúra typu LIFO.[24]

Predposledným dynamickým dátovým typom, ktorý bude rozobraný je strom. Ide o hierarchickú štruktúru, kde každý rodič (uzol) má 0 až n potomkov (uzlov) a každý potomok má práve jedného rodiča, pričom v celej štruktúre nie sú cykly. Uzol na najvyššej pozícii hierarchie sa nazýva koreňový uzol (root). Uzol, ktorý nemá žiadnych potomkov sa nazýva list. Každý podstrom stromu je taktiež strom, ide teda o rekurzívnu štruktúru. Strom je veľmi obľúbenou štruktúrou pre svoju jednoduchosť. [24,25]

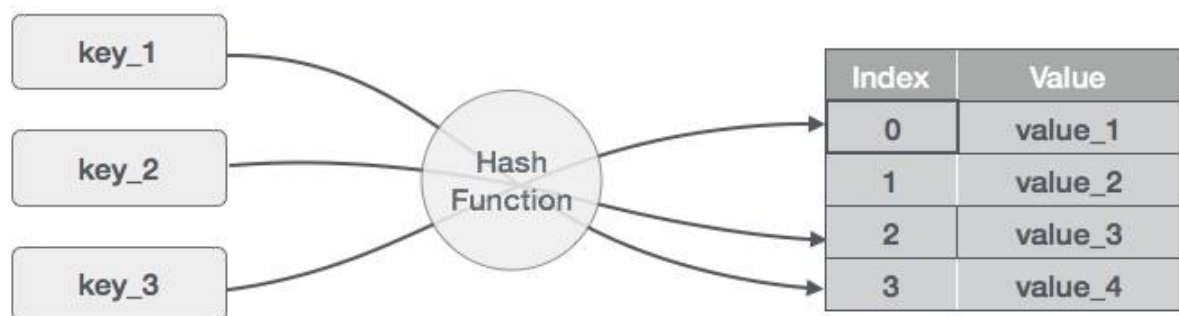


Obrázok 21. Binárny strom.[26]

Medzi základné parametre stromu patrí N-arita. Tento údaj hovorí, koľko potomkov môže mať maximálne každý rodič. Z tohoto hľadiska je najpoužívanejším binárny strom, kde má každý uzol žiadneho, jedného alebo dvoch potomkov. Ďalším parametrom je hĺbka, tento parameter hovorí na akej úrovni sa nachádza daný uzol. Strom by mal byť taktiež pravidelný, každý uzol v N-árnom strome musí mať 0 alebo N potomkov. N-árny strom je potom vyvážený, pokiaľ pre všetky listy platí, že nie sú hlbšie ako ostatné. Pri prehľadávaní a prechádzaní stromu sa potom môže postupovať troma rôznymi stratégiami. Preoder, prechádza sa najskôr koreň, následne ľavý podstrom a na koniec pravý podstrom. Inorder, najskôr ľavý podstrom, následne koreň a na koniec pravý. Postorder, najprv ľavý podstrom, následne pravý a na koniec koreň. [24,25]

Poslednou popísanou dynamickou štruktúrou bude Hashovacia tabuľka (Hash table). Hashovacia tabuľka je dátová štruktúra, ktorá slúži na ukládanie dvojice kľúč - hodnota. Táto štruktúra kombinuje výhody vyhľadávania pomocou indexu (zložitosť $O(1)$) a prechádzania zoznamu (nízke nároky na pamäť). K ukládaniu dvojice kľúč – hodnota sa používa hashovacia tabuľka. Tabuľka je štruktúra, ktorá je postavená nad poľom obmedzenej veľkosti a ktorá pre adresáciu používa hashovacej funkcie. Nájdenie prvku pre daný kľúč zaberie priemerne $O(1)$ operácií. Samotná hashovacia funkcia potom musí mať tieto vlastnosti:

- Konzistentnosť – pre rovnaké vstupy musí dávať rovnaké výstupy.
- Využíva celý priestor adres s rovnakou pravdepodobnosťou.
- Rýchly výpočet adresy.
- Nezaručuje že pre dva rôzne objekty nevráti rovnakú adresu.



Obrázok 22. Princíp Hashovacej tabuľky.[24]

Z posledného bodu je jasné, že pri ukladaní môžu nastať kolízie. V tomto prípade sa používa niekoľko postupov. Najjednoduchším z nich je zret'azené rozptylovanie, kde hodnoty ukladáme ako lineárny zoznam za seba. V prípade kolízie sa teda pridá ďalší prvok do zoznamu. Alternatívou je potom otvorené rozptylovanie, kde sa hodnoty vkladajú priamo do poľa. Z *obrázka 22* je vidieť princíp vyhľadávania. Pri ňom sa pomocou kľúču cez hashovacia funkciu získa index v poli a tým pádom aj uložená hodnota, s ktorou chceme pracovať. Týmto spôsobom je potom jednoduché a pamäťovo nenáročné pristupovať náhodne na jednotlivé prvky, prípadne dané prvky mazať. [24,25]

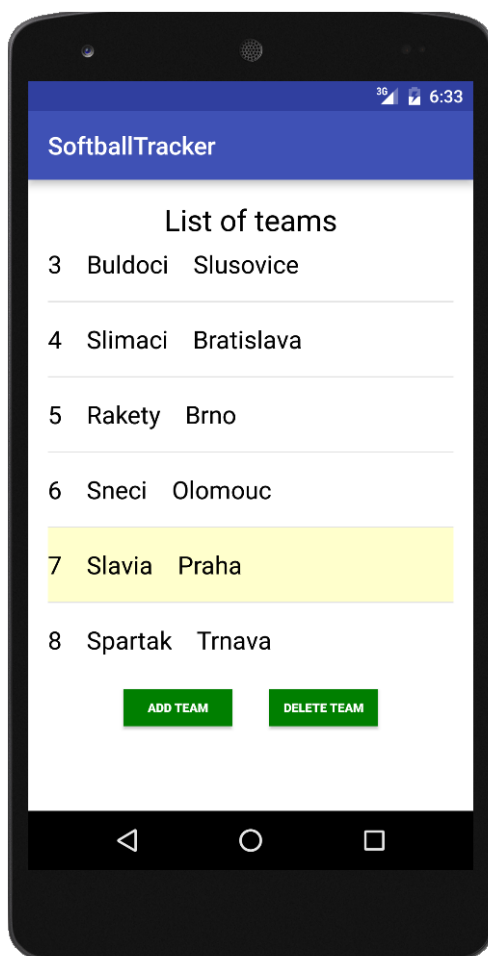
II. PRAKTICKÁ ČÁST

4 POPIS APLIKÁCIE SOFTBALL TRACKER

Na zápis priebehu zápasu bola vyvinutá aplikácia pre mobilné zariadenia. Aplikácia je vyvinutá natívne pre Android v programovacom jazyku Java. Na vývoj bolo použité vývojové prostredie Android Studio od spoločnosti Google. Samotná aplikácia má niekoľko funkcií. Prvou je jednoduchá správa tímov a hráčov, čím si môže užívateľ vytvoriť vlastný tím a hráčov. Ďalšou funkciou je možnosť zápisu priebehu zápasu a jeho následné spracovanie, vytváranie štatistík a export výsledkov.

4.1 Obrazovky správca tímov

Aplikácia obsahuje jednoduchého manažéra na správu tímov a hráčov, ktorý je prepojený zo SQLite databázou. Na obrázku je vidieť náhľad na prvú obrazovku manažéra, na tejto obrazovke je možné označiť ľubovoľný tím a následne ho zmazať, pridať nový tím, čím sa otvorí nové menu na pridanie tímu (*obrázok 23.*)



Obrázok 23. Náhľad na obrazovku správy tímov.

případně kliknutím na tým, prejsť na detail daného tímu. Pri prechode na pridanie nového tímu užívateľ zadá názov tímu a následne tím uloží kliknutím na tlačidlo Save. Tím je automaticky pridaný do zoznamu tímov a zapísaný do databáze.



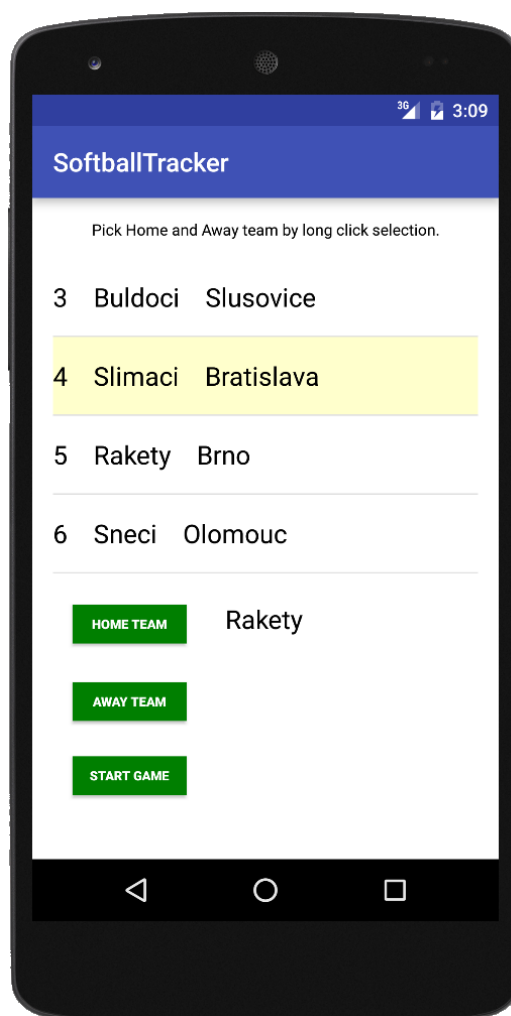
Obrázok 24. Náhľad na obrazovku správy hráčov.

Po označení tímu a kliku na tlačidlo zmazania je tím odobraný zo zoznamu a zmazaný z databáze. Pri krátkom kliku na tím, aplikácia prejde na Aktivitu detail tímu. Vo vrchnej časti obrazovky tejto aktivity sa nachádza meno, id a mesto tímu. Pod týmto záhlavím je zobrazený zoznam hráčov daného tímu. Na spodu obrazovky sa nachádzajú tlačidlá na pridanie a odstránenie hráča. Označením hráča a kliknutím na tlačidlo zmazania je hráč podobne ako tím vymazaný zo zoznamu a taktiež databáze. Kliknutím na tlačidlo pridania hráča sa otvorí formulár pomocou ktorého je možné zadať informácie o novom hráčovi. Formuláre na pridanie tímu aj hráča vyžadujú vyplnenie všetkých položiek, inak nie je

umůžnené záznam uložit. Pri zmazaní tímu z databáze sú automaticky kaskádovo zmazaní všetci hráči, ktorí patria do daného tímu.

4.2 Obrazovky výberu tímov

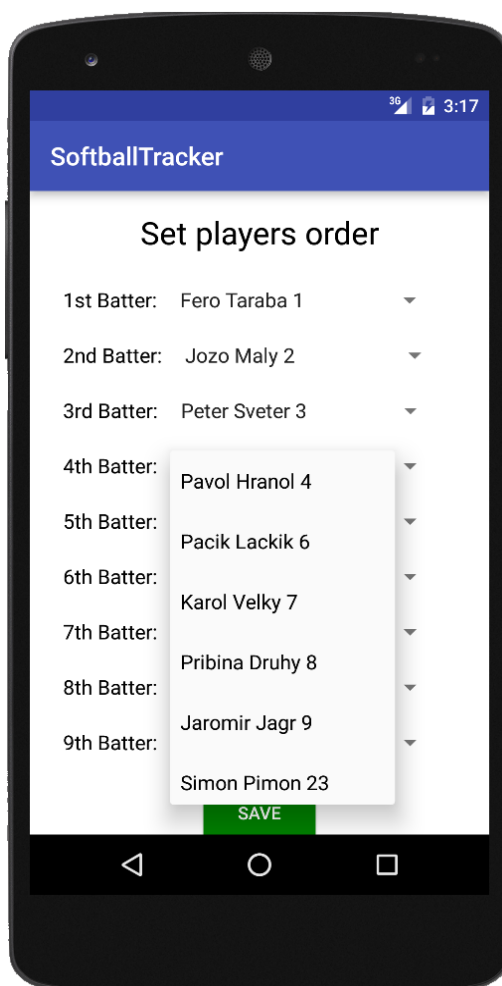
Ďalšou časťou aplikácie z pohľadu užívateľa je samotný zápis priebehu zápasu. Pri prechode na novú hru je užívateľ vyzvaný na výber tímov. Na tejto obrazovke sú vybrané tímy, ktoré sa zúčastnia zápasu. Tím sa pridržením označí a nastaví sa, ako domáci alebo hosť.



Obrázok 25. Náhľad na obrazovku výberu tímov.

Po tom, čo je daný tím označený, ako domáci alebo hosť je možné kliknutím na tento tím v zozname prejsť na ďalšiu obrazovku. Na tejto obrazovke je zobrazené poradie pákarov od prvého až po posledného. Ku každému miestu je potom možné priradiť hráča z daného tímu. K uloženiu nastaveného poradia je potrebné správne zadať hráčov na všetky pozície,

pričom jeden hráč sa nemôže nachádzať na viacerých pozíciách. Pokiaľ poradie nie je správne zadané aplikácia použije predvolené poradie hráčov z databáze.

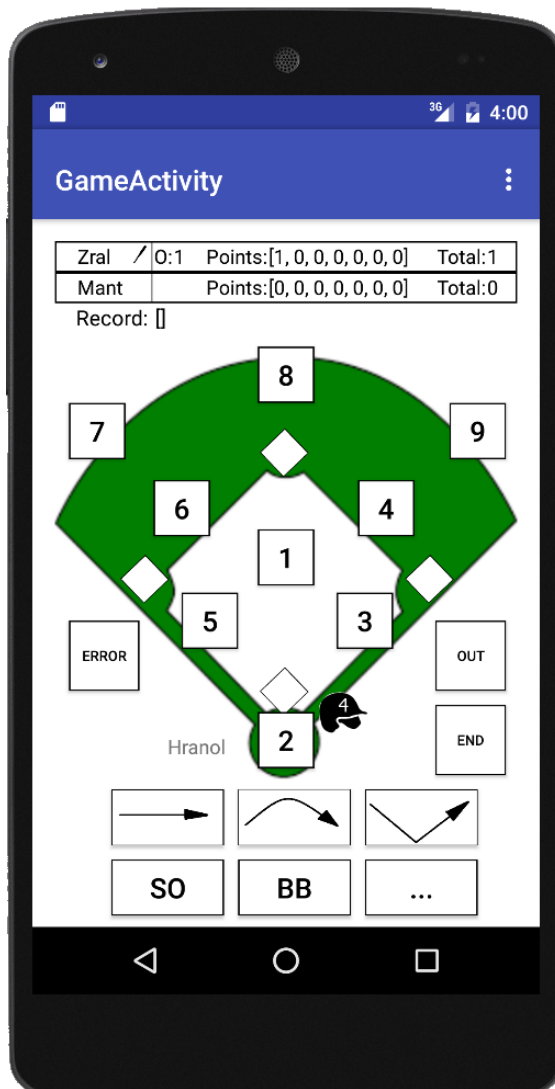


Obrázok 26. Náhľad na obrazovku nastavenia poradia páľkarov.

4.3 Obrazovky zápisu hry

Následne sa spustí hra na hlavnú obrazovku. Samotný zápis priebehu zápasu sa skladá zo štyroch obrazoviek. Medzi obrazovkami je možné listovať. Hlavná obrazovka slúži na plný zápis priebehu zápasu a je teda obrazovkou, na ktorej strávi užívateľ najviac času. Na vrchu tejto obrazovky sa nachádza tabuľka skóre. Na tejto tabuľke sú zapísané mená oboch tímov, aktuálny počet outov útočiaceho tímu, skóre v jednotlivých zmenách a celkové skóre zápasu.

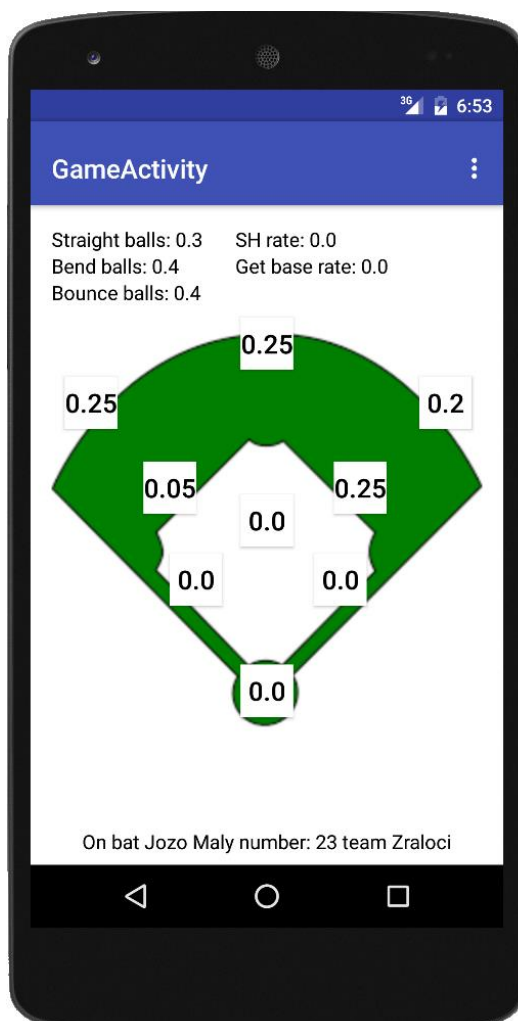
Údaje v tejto tabuľke sú generované automaticky v prebehu zaznamenávania zápasu. Pod tabuľkou skóre je schematicky znázornené ihrisko a tlačidlá s číslami jednotlivých hráčov brániaceho tímu v poli.



Obrázok 27. Náhľad na obrazovku zápisu zápasu.

Pomocou týchto tlačidiel užívateľ zadáva miesto, kde bola odpálená lopta a následný pohyb lopty medzi poliarmi po tomto odpale. Na schéme ihriska sú v rohoch umiestnené jednotlivé méty. Vedľa domácej méty sa nachádza piktogram bežca. Ten je užívateľom ťahaný medzi jednotlivými métami, podľa toho ako postupuje v hre. Pod a vedľa ihriska sú umiestnené tlačítka, ktoré popisujú bežné situácie, ktoré môžu nastať v hre. Situácie, ktoré nastávajú výnimočne je možné zadať pomocou tlačítka, ktoré otvára menu s týmito situáciami. Užívateľ pomocou týchto tlačidiel a presúvania

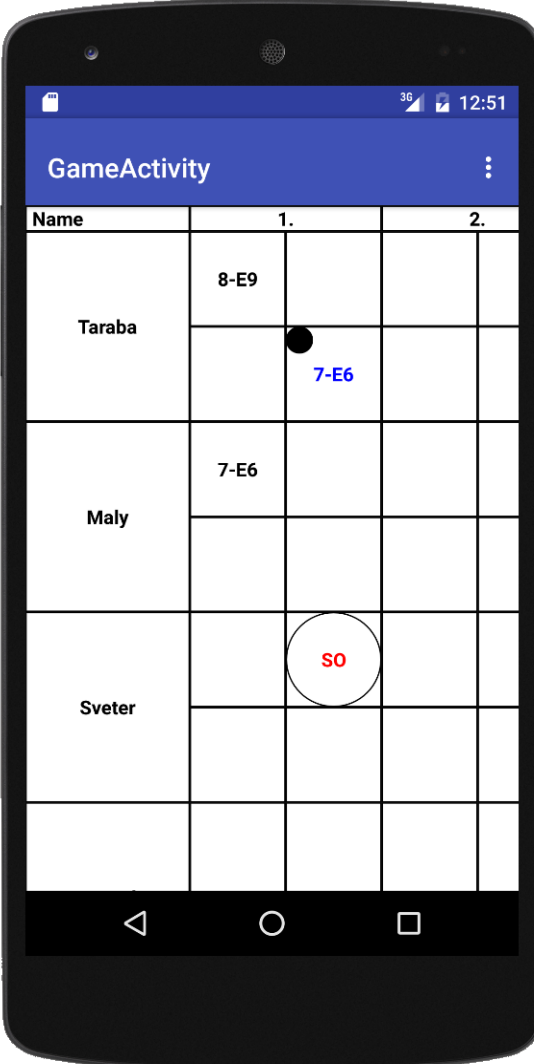
bežcov zaznamenáva priebeh zápasu. Aktuálny záznam, ktorý užívateľ zadal je zobrazený pod tabuľkou skóre.



Obrázok 28. Náhľad na obrazovku pre zobrazenie štatistik.

Druhá obrazovka slúži na zobrazenie štatistík aktuálneho pálkara. Hlavnou funkciou je teda zapisujúcemu užívateľovi poskytnúť informácie o aktuálnom pálkarovi. Na spodu tejto obrazovky sú umiestnené údaje aktuálneho pálkara. V strednej časti sa opäť nachádza schematicky zobrazené hracie pole spolu s pozíciami poliarov. Na rozdiel od prvej obrazovky sú však na týchto pozíciách zobrazené pravdepodobnosti odpalu do danej zóny. Na vrchu obrazovky sa nachádzajú ďalšie údaje. Medzi ne patrí pravdepodobnosť typu odpalu (vysoký, priamy, odrazený) a pravdepodobnosť tzv. uliatej lopty a homerunu. Všetky dáta na obrazovke štatistiky sú čerpané z databáze a aktuálnej hry. Čím viac zápasov bolo pomocou aplikácie hráčom zaznamenaných, tým sú zobrazované štatistiky presnejšie.

V poradí treťou obrazovku pri zapisovaní hry je obrazovka s výsledkovou tabuľkou. Ide o tabuľku, ktorá zobrazuje informácie o priebehu zápasu v podobnom formáte ako ručný zápis. V tejto tabuľke sú v riadkoch rozložení jednotliví hráči. Stĺpce tejto tabuľky sa menia v prípade ďalšej zmeny a v prípade, kedy útočiaci tím vystriedal všetkých hráčov na pálke a hráči pokračujú na pálke opätovne.

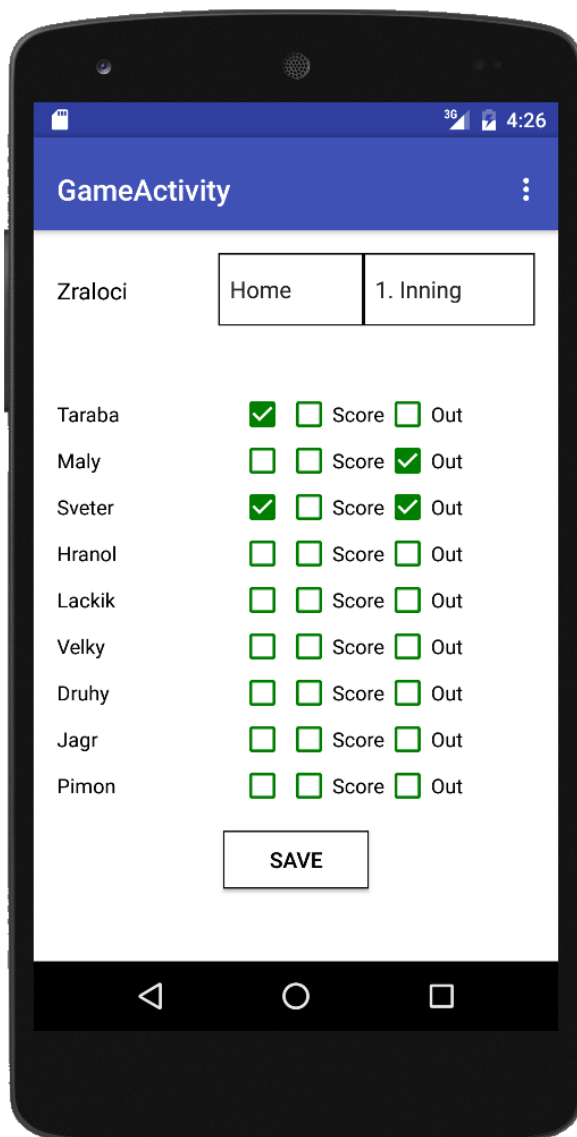


Name	1.	2.
Taraba	8-E9	
		7-E6
Maly	7-E6	
Sveter		SO

Obrázok 29. Náhľad na obrazovku zobrazujúcu priebeh zápasu.

Pre každý tím sa vytvára vlastná tabuľka a vždy je zobrazená tabuľka aktuálne útočiaceho tímu, teda tá do ktorej sa práve zapisuje. Každá bunka v tabuľke je ďalej rozdelená krížom na štyri bunky. Tie znázorňujú jednotlivé méty, pričom pravá horná je méta číslo a proti smere hodinových ručičiek pokračuje číslovanie druhou až po domácu. Herná situácia, ktorá nastala, keď bol na pálke príslušný hráč sa zapisuje do riadka pálkara. Pohyb ostatných hráčov sa potom zapisuje späť. Pokiaľ sa teda bežec na hru aktuálneho pálkara

dostal na určitú métu, v jeho riadku sa na tejto méte objaví hra, ktorá prebehla pri pátkarovom odpale. Tým pádom sa dá späťne zrekonštruovať priebeh celého zápasu.

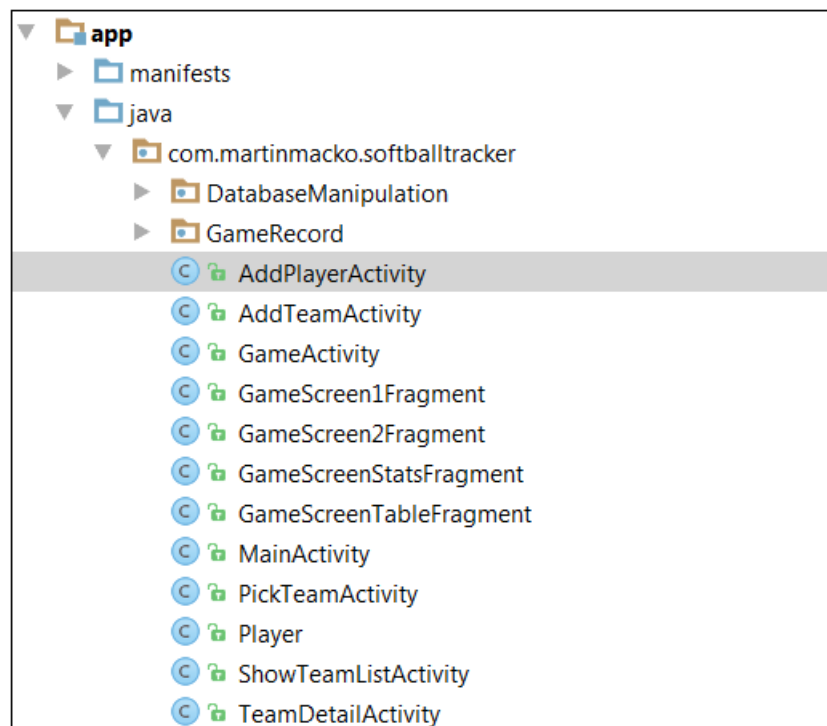


Obrázok 30. Náhľad na obrazovku zrýchleného zápisu.

Posledná herná obrazovka zobrazuje zoznam hráčov a ich body a outy v jednotlivých zmenách. Na vrchu obrazovky sa nachádzajú spinnere, ktoré umožňujú vybrať tím a zmenu. Pod nimi sa následne zobrazí zoznam hráčov a ich výsledky v danej zmene. V prípade, že v priebehu zápasu došlo k chybe a hráčovi bol zle zapísaný out alebo bod je pomocou tejto obrazovky možné túto chybu opraviť. Táto zmena sa prejaví v obrazovke, ktorá zobrazuje tabuľku. Detailný popis zapisovania jednotlivých herných situácií sa nachádza v Prílohe číslo 1 (Užívateľská príručka).

5 PROGRAMÁTORSKÁ PRÍRUČKA

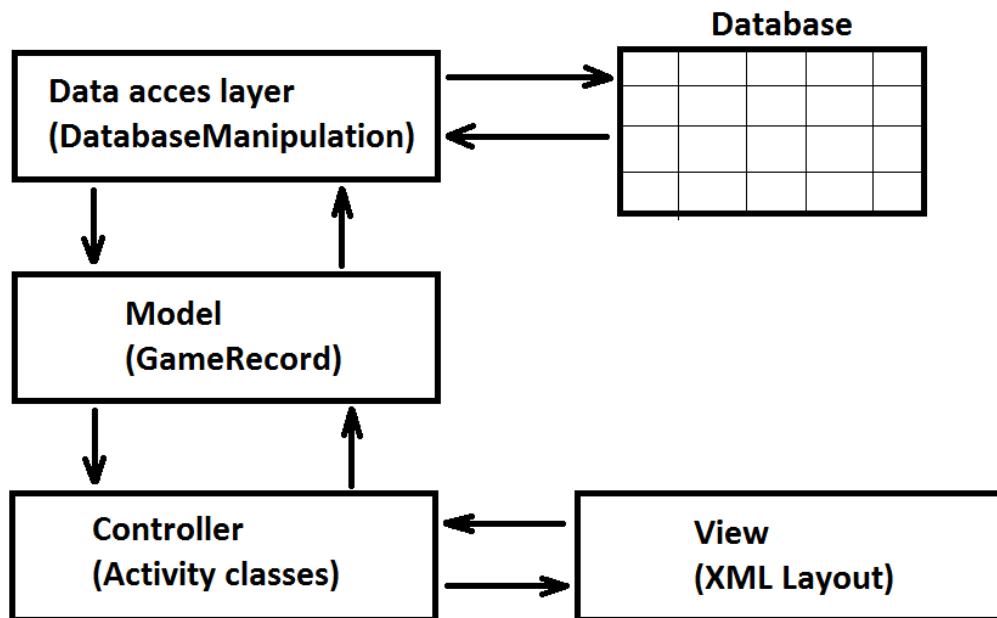
V predošlej kapitole bola aplikácie popísaná z pohľadu užívateľa, v kapitole 5 bude popísané programovanie a použité techniky z pohľadu programátora. Ako už bolo spomenuté, aplikácia bola vyvinutá natívne pre platformu Android. Použité boli natívne podporované programovacie jazyky. Backend aplikácie je teda napísaný v Jave zatiaľ čo Frontend pomocou značkovacieho jazyka XML. Minimálne podporované API aplikácie je API 16, takže aplikácia je podľa štatistík firmy Google spustiteľná na vyše 90 % aktuálne používaných Android zariadení. Na *obrázku 29*. je vidieť štruktúra Java tried. V koreňovom adresári sa nachádzajú všetky triedy, ktoré patria k jednotlivým aktivitám a sprostredkujú komunikáciu zo samotným View aplikácie, teda XML layoutmi.



Obrázok 31. Štruktúra Java tried aplikácie.

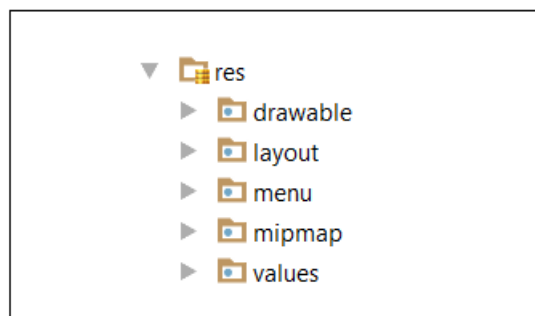
V štruktúre je vidieť ďalšie dva Package, čo sú v podstate priečinky, ktoré združujú triedy jedného druhu. Package GameRecord obsahuje triedy, ktoré pracujú na pozadí aplikácie a obsahujú logiku, ktorá sa stará o správny zápis priebehu zápasu. V poslednom package DatabaseManipulation sa nachádzajú triedy, ktoré zabezpečujú zápis, čítanie a správu databáze SQLite, slúžia teda ako vrstva, ktorá získava dáta a posúva ich na ďalšie spracovanie modelu (GameRecorder), ktorý následne tieto dáta cez triedy jednotlivých

aktivít pomocou XML layoutu zobrazuje užívateľovi. Aktivitné triedy potom naopak zbierajú vstupy od užívateľa a posúvajú ich do modelu, ktorý spätne relevantné dáta zapisuje do databáze.



Obrázok 32. Schéma jednotlivých častí aplikácie..

Použitý je teda MVC návrhový vzor. Jeho výhodou je modularnosť jednotlivých prvkov. Zmena designu a vzhľadu aplikácie prebieha čisto vo View a nie je závislá na logike. Naopak pokiaľ sa zmení model výpočtu štatistík alebo zápisu jednotlivých situácií vzhľad aplikácie týmto nie je niako zasiahnutý. V neposlednom rade sa v budúcnosti môže zmeniť spôsob, akým aplikácia príma dáta. Miesto databáze to môže byť napríklad cez sieť. V tomto prípade sa opäť zmení iba Dátová vrstva. Model teda dostane dáta z iného zdroja čo ho, ale nijak neomedzuje, jednotlivé časti aplikácie sú teda plne oddelené.

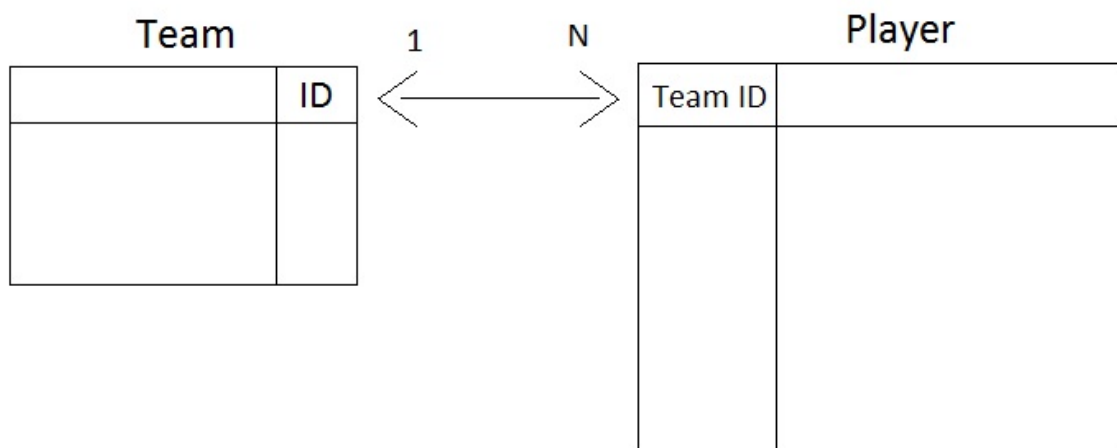


Obrázok 33. Štruktúra zdrojov aplikácie.

Ďalší koreňový adresár obsahuje zdroje aplikácie. V package drawable sa nachádzajú všetky obrázkové zdroje a XML súbory, ktoré sú v aplikácii použité na vykresľovanie. Najdôležitejším packageom v zdrojoch je samozrejme package layout, v ktorom sa nachádzajú XML súbory popisujúce rozloženie prvkov aplikácie. V ďalšom package menu sa nachádzajú XML súbory, ktoré popisujú menu položky použité v aplikácii. Package mipmap obsahuje podsúbory, v ktorých sú uložené obrázky pre jednotlivé rozlíšenia obrazovky. V poslednom package values sú pomocou XML definované použité farby a textové reťazce. V prípade zmeny farby, designu alebo textovej hodnoty v programe je potom možné túto zmenu previesť iba na jednom mieste. Zmena sa potom automaticky premietne do celého programu.

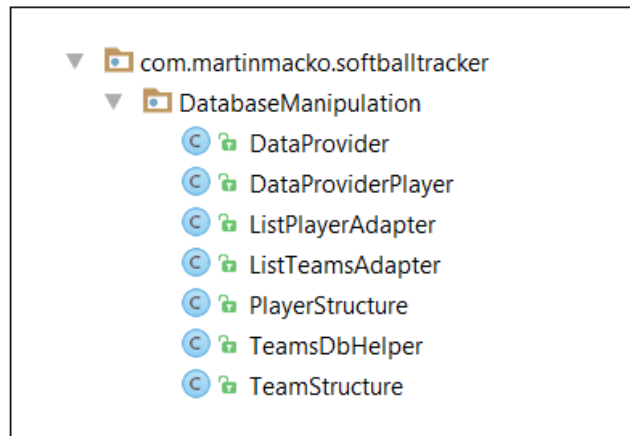
5.1 Databáza

Aplikácia svoje dáta ukladá do SQLite databáze. Týmto sa zaručí, že štatistické dáta a dáta o hráčoch a tímoch zostanú zachované aj po ukončení aplikácie. Databáza sa skladá z dvoch tabuliek. Tabuľka tímov skladuje dáta o jednotlivých tímoch. A tabuľka hráči udržiava základné informácie o hráčoch a taktiež ich štatistiky z predošlých zápasov. Tabuľky sú prepojené reláciou 1:N a spojené cez id tímu. Jeden tím teda obsahuje N hráčov avšak jeden hráč môže byť iba v jednom tíme.



Obrázok 34. Prepojenie tabuliek v databázi.

O správu databáze sa stará package DatabaseManipulation, ktorý obsahuje 7 tried. Každá trieda zastáva špecifickú úlohu v správe databáze. Triedy TeamStructure a PlayerStructure sú statické triedy, ktoré popisujú štruktúru databáze. V prípade zmeny štruktúry databáze je teda túto zmenu previesť v prvom rade v týchto dvoch triedach.



Obrázok 35. Triedy balíčka na správu databáze.

Trieda TeamsDbHelper obsahuje metódy na vytvorenie databáze a taktiež metódy, ktoré do databáze prístupujú. Teda metódy na vkladanie, mazanie a úpravu stávajúcich záznamov. Tabuľky sú vytvorené pomocou štandardného SQL dotazu (ako je vidieť na ukážke), ktorý čerpá data práve z TeamStructure triedy.

```
private static final String CREATE_TEAMS_QUERY =
"CREATE TABLE "+TeamStructure.NewTeamInfo.TABLE_NAME+" ("
+TeamStructure.NewTeamInfo.ID+" INTEGER PRIMARY KEY AUTOINCREMENT,"+
TeamStructure.NewTeamInfo.TEAM_NAME+" TEXT,"+
TeamStructure.NewTeamInfo.CITY_NAME+ " TEXT)";

db.execSQL(CREATE_TEAMS_QUERY);
```

Ostatné operácie z databázou už prebiehajú pomocou vstavaných tried, ktoré zabezpečujú a kontrolujú správnosť vkladajúcich údajov a následne sami vytvárajú dotaz na databázu. Pri takomto postupe sa najskôr do triedy ContentValues vložia údaje, ktoré chceme do databáze poslať. Následne sa vytvorí premenná, ktorá popisuje podmienku (supluje where podmienku v bežnom SQL dotaze.). V poslednom rade sa zavolá príslušná metóda, ktorá vytvorí dotaz. Ako je vidieť na zjednodušenej metóde aktualizácie záznamu v tabuľke hráčov.

```
public void updatePlayer(int straightCount, int bendCount, SQLiteDatabase
db, int playerId){
ContentValues contentValues = new ContentValues();

contentValues.put(PlayerStructure.NewPlayerInfo.STRAIGHT_COUNT, straightCo
unt);
contentValues.put(PlayerStructure.NewPlayerInfo.BEND_COUNT, bendCount);

String id = String.valueOf(playerId);

db.update(PlayerStructure.NewPlayerInfo.TABLE_NAME, contentValues, PlayerSt
ructure.NewPlayerInfo.ID+" = ?", new String[]{id});}
```

Posledným druhom metód v tejto triede sú metódy, ktoré sa dotazujú na dáta z tabuľky.

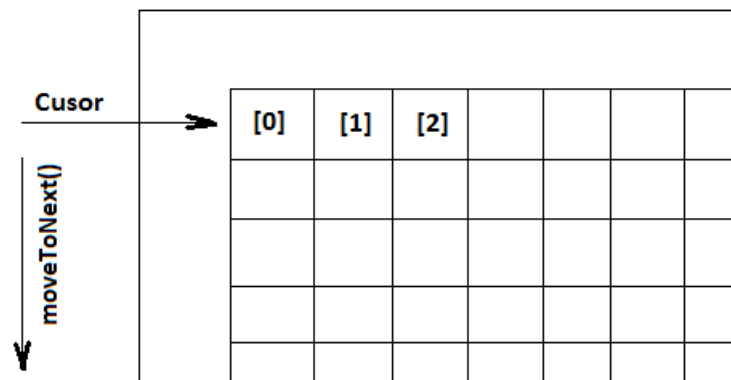
```
public Cursor getInformations(SQLiteDatabase db){
    Cursor cursor;

    String[] projections =
    {TeamStructure.NewTeamInfo.ID, TeamStructure.NewTeamInfo.TEAM_NAME, TeamStr
    ucture.NewTeamInfo.CITY_NAME};

    cursor =
    db.query(TeamStructure.NewTeamInfo.TABLE_NAME, projections, null, null, null,
    null, null);
    return cursor;}

```

Ako je vidieť na poslednej ukážkovej metóde, metóda najskôr naplní pole projekcií, čo sú stĺpce, na ktoré sa dotazuje, následne priradí do triedy Cursor dotaz so zadanými projekciami. Táto metóda potom objekt Cursoru vracia. Cursor je v podstate určitý ukazovateľ do databáze. Tento ukazovateľ je objektom a po jeho vrátení je s ním možné ďalej pracovať. Na začiatku Cursor ukazuje na prvý riadok v tabuľke, ktorý spĺňa zadané projekcie a následne je s ním možné prechádzať databázu a získať dáta, ktoré aplikácia v danom bode programu potrebuje. Indexy Cursoru odkazujú na jednotlivé projekcie a metódou `moveToNext()` je možné prechádzať riadky tabuľky. Pokiaľ sa aplikácie dotazuje na špecifické dáta, napríklad hráčov daného tímu, Cursor sa pohybuje iba po riadkoch splňujúcich podmienku.

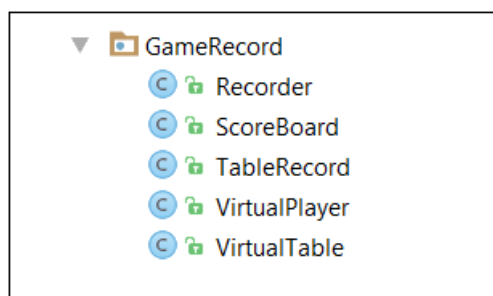


Obrázok 36. Ukážka práce s Cursorom.

Ďalšie dve triedy `DataProvider` a `DataProviderPlayer` poskytujú štruktúru pre dáta získané z databáze. Dáta v tejto štruktúre sú následne v posledných dvoch triedach `ListPlayerAdapter` a `ListTeamAdapter` uložené a metódy v týchto triedach umožňujú ich ďalšie spracovanie v modeli aplikácie.

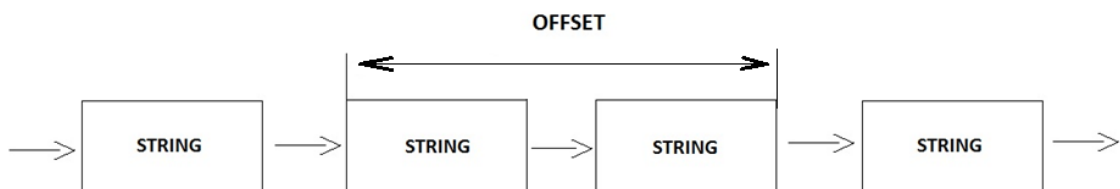
5.2 Zápis hry

Ďalším balíčkom je balíček GameRecord, ten sa stará o všetku logiku celej aplikácie. Najdôležitejšiou triedou tohoto balíčku je trieda Recorder. Tá má na starosti niekoľko kľúčových úloh pre správny chod aplikácie. V prvom rade sa stará o jednotlivých hráčov. Pri inicializácii tejto triedy sa z databázovej vrstvy načítajú údaje o hráčoch, a títo hráči sa uložia do statického poľa hráčov. Recorder sa ďalej stará o všetku logiku priebehu zápasu a záznam priebehu, ktorý následne pomocou parseru spracováva a posiela cez triedy Aktivít na View a tieto informácie sú následne zobrazované užívateľovi.



Obrázok 37. Triedy balíčka starajúce sa o logiku zápisu.

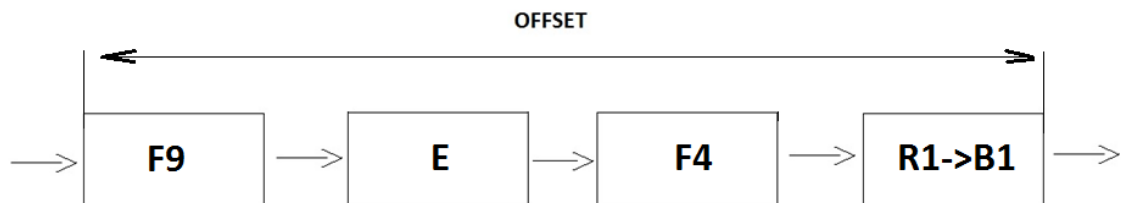
Samotný priebeh zápasu je v triede Recorder zaznamenávaný pomocou niekoľkých dynamických dátových štruktúr ktoré boli popísané v teoretickej časti. Základným zápisom je zápis textového reťazca do lineárneho zoznamu. Štruktúra tohoto lineárneho zoznamu je vidieť na obrázku 38.



Obrázok 38. Dátová štruktúra na záznam priebehu

V každej položke lineárneho zoznamu je zapísaná jedna atomická herná situácia. Príklad je možné vidieť na obrázku 39. V tomto príklade je každá operácie zápisu zaznamenaná v jednej položke. Zápis znamená, že lopta bola odpálená na poliara 9, ten prihrával

poliarovy 4, ktorý spravil chybu a prvý bežec prebehol na prvú métu. Tento zápis prebehol klikmi na tri tlačítka a presunutím bežca na prvú métu. Položka offset potom označuje položky zoznamu, ktoré súvisia s jednou rozohrou. Ide o číslo, ktoré si v pamäti udržuje trieda Recorder a je potom možné lineárny zoznam rozdeliť na jednotlivé rozohry.



Obrázok 39. Príklad zápisu do lineárneho zoznamu.

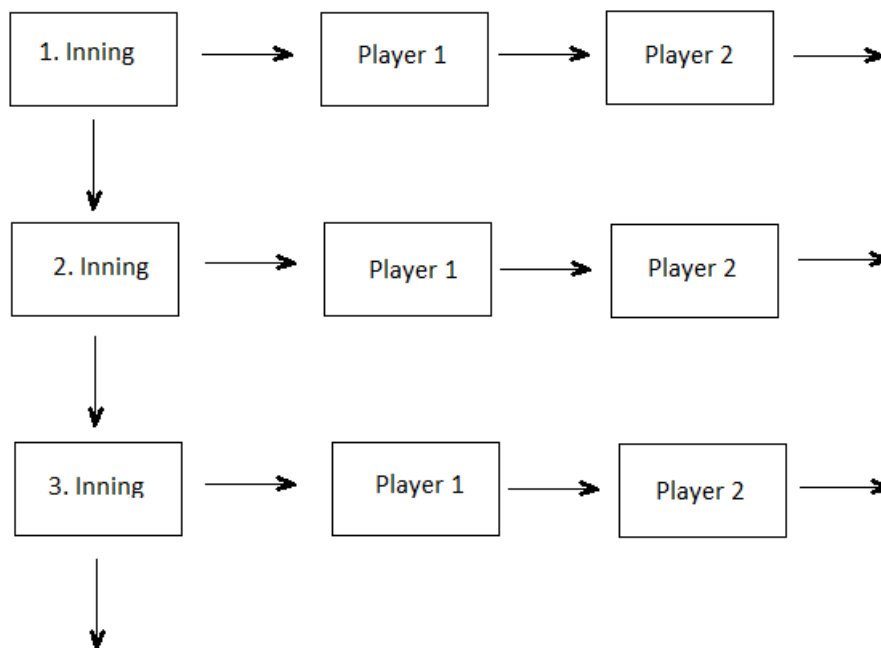
Táto štruktúra je základom pre zápis zápisu. Pre ďalšie zobrazovanie zápisu sa táto štruktúra v triede Recorder pomocou metódy `parseRecordForTable()` spracuje a upraví do podoby štruktúry, ktorú je možné následne zapisovať do tabuľky priebehu zápisu a do tabuľky, ktorá je použitá v aktivite na skrátenejší zápis. Parsovanie prebehne vždy po stlačení tlačidla End. V tomto momente Controller, teda aktivita plného zápisu pošle informáciu o ukončení zápisu. Parser si v prvom kroku pomocou offsetu vytiahne časť lineárneho zoznamu, ktorá prislúcha poslednej rozohre. Následne prejde záznam, separuje a upraví položky, ktoré tvoria obsah zápisu, ktorý bude neskôr zapísaný do prislúchajúcej bunky v tabuľke. Tieto položky zo zoznamu zmaže a upraví do finálnej podoby. Položky nakoniec uloží do členskej premennej triedy `TableRecord mStringRecord` (viz štruktúra triedy). Pokiaľ je súčasťou zápisu aj druh odpalu (pri zápise ide o dobrovolný údaj) do premennej `mStrokeType` sa uloží číslo podľa druhu odpalu. Pomocou tohoto údaju sa neskôr vykreslí v tabuľke odpovedajúca grafika.

```
public class TableRecord {
    public String mBaseNumber;
    public String mPlayerNumber;
    public String mColumnIndex;
    public String mStringRecord;
    public boolean mDidScore;
    public boolean mIsOut;
    public int mStrokeType; }
```

V ďalšom kroku je potrebné získať ostatné premenné triedy `TableRecord`. Zostávajúca časť záznamu sa teda prejde opätovne. V zozname momentálne ostali iba pohyby a outovanie hráčov. Z týchto údajov sa následne získajú premenné číslo mety a hráča, ku ktorému sa pridá členská premenná samotného Recorderu, ktorá drží informáciu o aktuálnom stĺpci

tabuľky. Tieto tri čísla potom udávajú presnú pozíciu bunky, do ktorej zápis prislúcha. Pokiaľ sa v jednej rozohre posúva po mete viac hráčov vytvorí sa viacej záznamov typu TableRecord, ktoré budú mať rovnaký obsah bunky v podobe premennej mStringRecord avšak rozdielne premenné mBaseNumber, mPlayerNumber. Tým sa teda rozohra, ktorá sa udiala na ihrisku objaví vo viacerých bunkách. Poslednými dvoma premennými, ktoré sú potrebné k plnému zápisu sú boolean hodnoty či daný hráč získal bod prípadne či bol outovaný. Tie parser získa jednoducho zo zápisu. Pokiaľ hráč dobehol na štvrtú metu premenná mDidScore sa nastaví na hodnotu true. A naopak pokiaľ má daný hráč pri svojom čísle príznak (O) premenná mIsOut sa nastaví na hodnotu true.

Po tomto kroku je položka triedy TableRecord kompletná. Táto jedna položka je následne zapísaná do lineárneho zoznamu. Recorder udržiava takýto lineárny zoznam pre každý tím. Záznam je parsovaný ešte do poslednej dátovej štruktúry, ktorá je použitá na zobrazenie v aktivite skráteneho záznamu.



Obrázok 40. Dátová štruktúra pre skrátene zápis.

Jej štruktúra je vidieť na obrázku 40. Ide o jeden lineárny zoznam, ktorého položky sú samostatné lineárne zoznamy. V koreňovom zozname sú jednotlivé Inningy, teda zmeny hry, v každej zmene je potom lineárny zoznam o deviatich položkách, ktoré predstavujú

jednotlivých hráčov. V každej tejto položke, ktorá predstavuje daného hráča je potom premenná triedy VirtualPlayer o danej štruktúre:

```
public class VirtualPlayer {
    public int mIndex;
    public int mColumn;
    public int mPoints;
    public boolean mOut; }
```

Premenná mIndex udáva poradové číslo hráča, mColumn odpovedajúci stĺpec v tabuľke a premenné mPoints a mOut hovoria o počte bodov, respektíve o tom či bol hráč outovaný. Táto dátová štruktúra je taktiež vytvorená v metóde, ktorá sa stará o parsovanie záznamu. Ide v podstate o maticu, na ktorej riadkoch sú jednotlivé zmeny, na stĺpcoch daný hráči a v samotných bunkách matice premenná typu VirtualPlayer, ktorá hovorí ako si daný hráč viedol v danej zmene.

Poslednou úlohou triedy Recorder je udržiavať informácie o priebehu hry pomocou, ktorých môže následne Controller aplikácie nastaviť View na požadovaný stav. Medzi informácie, ktoré Recorder uchováva patria:

1. Aktuálne útočiaci tím (mActualAttackingTeam).
2. Hráč, ktorý je aktuálne na pálke (mActualBatter).
3. Posledný hráči, ktorí boli na pálke za jednotlivé tími(mLastOnBatTeamA/B).
4. Označený hráč (mClickedPlayerIndex).
5. Aktuálny Inning (mInning).
6. Aktuálny počet outov útočiaceho tímu (mActualOutCount).
7. Body jednotlivých tímov v jednotlivých zmenách (mTeamA/BPoints).

Trieda potom obsahuje metódy, ktoré udržujú tieto členské premenné podľa vnútornej logiky hry v aktuálnom stave, medzi tieto metódy patria metódy:

1. getPlayerOnBase(int base) - Vracia hráča na danej méte.
2. freeBase(int player) - Uvolní métu dosiahnutú hráčom.
3. nextBatter() - Posunie na pálku ďalšieho hráča.
4. resetBatters() - Otočí zoznam hráčov v prípade, že sa na pálke otočili všetci hráči.
5. freeAllBases() - Uvolní všetky méty (pri zmene inningu).
6. switchAttackingTeam() - Premení útočiaci tím a brániaci tím (pri zmene inningu).
7. playerThatScored() - Vráti hráča, ktorý skóroval.

V popise modelu nebola spomenutá ešte jedna trieda a to trieda ScoreBoard. Úlohou tejto triedy je taktiež zaznamenávať priebeh zápasu. Na rozdiel od Recorderu však dáta udržiava vo forme čitateľnej pre užívateľa a zobrazuje ich v hlavnej aktivite pre zápis.

Uživatel takto vie akú sekvenciu zaznamenal. Po dokončení každej rozohry (stlačení tlačidla End) je tento záznam zmazaný.

5.2.1 Parser zápisu

Spomínaný lineárny zoznam, ktorého príklad bol ukázaný na *obrázku 39* je k úprave do výsledných dátových štruktúr modelu spracovený pomocou parseru. Každá herná situácia má v lineárnom zozname špecifické označenie, aby ju bolo ďalej možné spracovať týmto parserom.

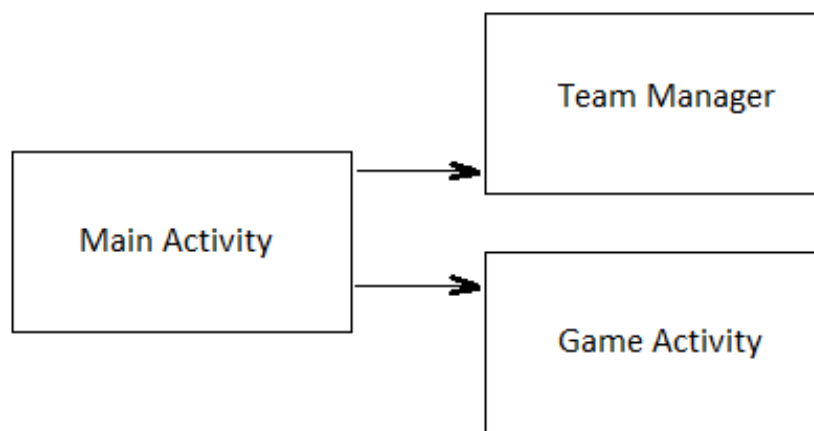
Označenia herných situácií:

1. Hra poliarara (Stlačenie tlačidla [1-9]) – „FX“, kde X je číslo daného poliarara.
2. Chyba poliarara (Stlačenie tlačidla [E]) – „E“, vzťahuje sa k nasledujúcemu poliararovy.
3. Priamy odpal (Stlačenie tlačidla priameho odpalu) – „SS“.
4. Vysoký odpal (Stlačenie tlačidla vysokého odpalu) – „SE“.
5. Odrazený odpal (Stlačenie tlačidla odrazeného odpalu) – „SB“.
6. Trafený nadhadzovač (Stlačenie tlačidla HB) – „HB“.
7. Strike Out (Stlačenie tlačidla SO) – „SO“.
8. 4 chybné nadhody (Stlačenie tlačidla BB) – „BB“, Base on Ball
9. Úlievka (Stlačenie tlačidla SH) – „SH“.
10. Homerun (Stlačenie tlačidla HR) – „HR“.
11. Presun hráča po mete (Presunutie hráča na metu) – „R2->B3“ Presunutie hráča 2 na metu číslo 3.
12. Vyoutovanie hráča (Označenie hráča + Stlačenie tlačidla [Out]) – „IOR1OB1“ vyoutovanie hráča 1 na méte 1.

Princíp parseru, ktorý tento záznam spracováva je možné preniesť do iných systémov nakoľko princíp spracovania zostáva rovnaký bez ohľadu na systém.

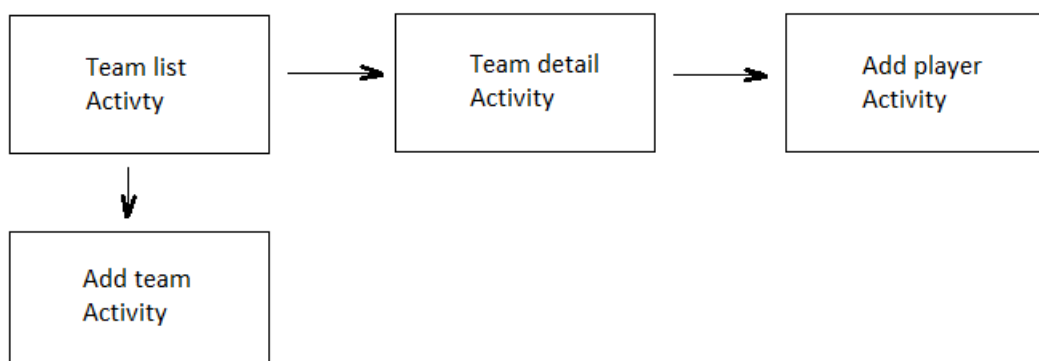
5.3 Aktivity

Ďalšou súčasťou aplikácie sú triedy aktivít. Tie v aplikácií zastávajú funkciu Controlleru. V prvom rade sprostredkujú dáta z modelu na samotné View. Ich druhou úlohou je odchytať užívateľské interakcie ako označovanie, stlačenia tlačidiel alebo posúvanie hráčov a reagovať na ne vhodnými akciami v modely.



Obrázok 41. Schéma prepojenia aktivít.

Úvodnou aktivitou pri spustení aplikácie je aktivita Main. Z tej je možné prejsť na aktivitu Team Manageru. Presun medzi týmito aktivitami je vidieť na obrázku 40. Prvou aktivitou je aktivita Team list.



Obrázok 42. Prechod medzi aktivitami Team managera.

5.3.1 Aktivity spravujúce databázu

Táto aktivita pristupuje priamo na dátovú vrstvu. Zavolá metódu, ktorá jej vráti inštanciu triedy Cursor. Táto aktivita potom v metóde reloadList() použije objekt cursora a prečíta zoznam všetkých tímov z databáze. Každý prečítaný riadok uloží do objektu triedy DataProvider, ktorý je vložený do adaptéra. Tento adaptér je následne použitý ako zdroj položiek do XML prvku ListView, ktoré tvorí zoznam všetkých tímov.

Ukážka čítania údajov o jednotlivých tímoch z databáze:

```
if(cursor.moveToFirst()){
    do{
        String id,name,city;
        id = cursor.getString(0);
        name = cursor.getString(1);
        city = cursor.getString(2);
        DataProvider dataProvider = new DataProvider(id,name,city);
        listTeamsAdapter.add(dataProvider);

    }while (cursor.moveToNext());
    cursor.close();
}
```

V poslednom kroku je po prečítaní dát z databáze potrebné objektu cursoru uzavrieť. Po úspešnom načítaní dát je štruktúra ListView naplnená dátami a je zobrazený zoznam tímov. Z tejto aktivity je ďalej možné prejsť na aktivitu pridania tímu. Ide o aktivitu formulára, ktorá obsahuje elementy EditText na zadanie informácií o tíme. Pre vloženie tímu musia byť všetky polia vyplnené. Po následnom potvrdení vloženia je táto aktivita uzavretá a aplikácia automaticky prechádza naspäť na aktivitu zoznamu tímov. Všetky prvky v ListView majú nastavené dva listenery. Listenery sú objekty, ktoré zaznamenávajú akcie vykonávané nad týmto zoznamom tímov a umožňujú na ne programátorovy reagovať. Po pridržaní danej položky sa položka označí a následne ju je možné zmazať kliknutím na tlačidlo Delete. Po tomto kliku sa zavolá metóda, ktorá podľa vybraného prvku zo zoznamu vytvorí dotaz na jeho zmazanie z databáze. Pri zmazaní sa kaskádovo zmažú aj odpovedajúce položky v tabuľke hráčov. Po krátkom kliknutí na daný tím aplikácia prejde na aktivitu Team detail.

Ukážka prechodu na ďalšiu aktivitu s prenosom dát :

```
Intent i = new Intent(this, TeamDetailActivity.class);
Bundle bundle = new Bundle();
bundle.putString("TeamName", teamName);
```

```
bundle.putString("TeamCity", teamCity);  
bundle.putString("TeamId", teamId);  
  
i.putExtras(bundle);  
startActivity(i);
```

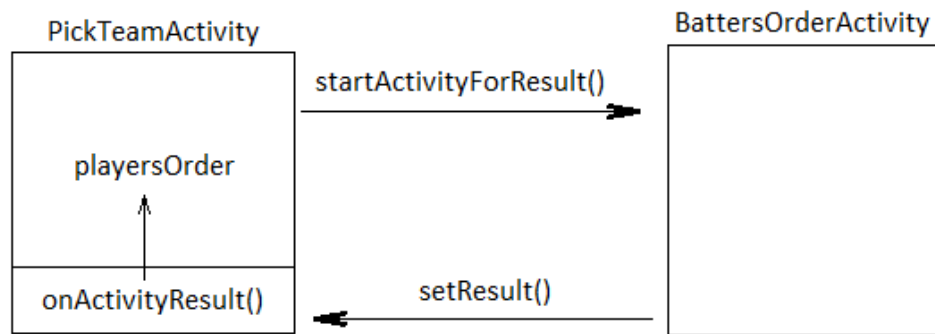
Pri prechode na túto aktivitu sa vytvorí triedy intent a trieda bundle. Trieda bundle je kontajner, do ktorého je možné vložiť dáta, ktoré je potrebné predať medzi danými aktivitami. V tomto prípade ide o dáta vybraného tímu. Tie sú potrebné k tomu, aby aktivita detailu daného tímu vedela získať potrebný zoznam hráčov.

Ukážka prijatia dát aktivitou :

```
Bundle bundle = getIntent().getExtras();  
  
mTeamName = bundle.getString("TeamName");  
mTeamCity = bundle.getString("TeamCity");  
mTeamId = bundle.getString("TeamId");
```

V konštruktoře nasledujúcej aktivity sa následne dáta z triedy bundle extrahujú a na databázu sa pošle ďalší dotaz, tento ktát na odpovedajúci zoznam hráčov. Tieto dáta sú opäť cursorom prečítané a uložené do adaptéra, ktorý slúži ako zdroj ListView na zoznam hráčov. Z tejto aktivity je opäť možné prejsť na formulár k pridaniu hráča do databáze a po označení daného hráča zmazať.

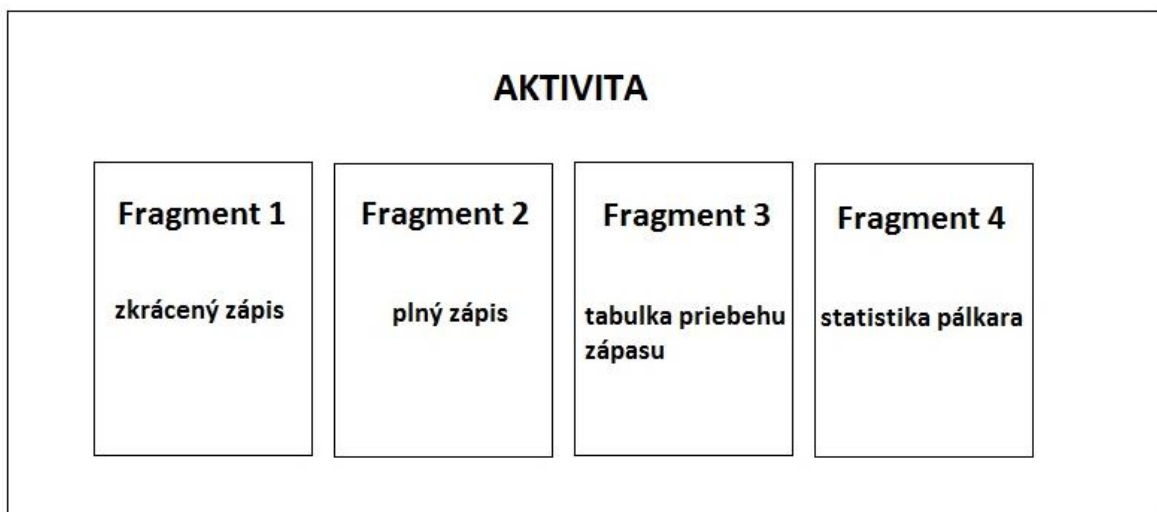
Z hlavnej aktivity je ďalej možné prejsť cez aktivitu výberu tímu na samotnú aktivitu, ktorá sa stará o zaznamenávanie zápasu. Na aktivite pre výber tímu sa vygeneruje zoznam tímov, ktorý je podobne ako v Team Managerovy vygenerovaný z databáze. Následne je po výbere domáceho a hosťujúceho tímu možné prejsť na aktivitu poradia páľkarov. Aktivita na určenie poradia páľkarov sa z aktivity výberu tímov volá ako aktivita na získanie výsledkov. To znamená že po jej ukončení uložením nastaveného poradia aktivita pošle volajúcej aktivite výsledok. Ten sa skladá z troch informácií. Prvou je tím, pre ktorý bolo poradie zmenené. Druhou je informácie či bolo nastavené poradie validné a treťou (pokiaľ poradie validné bolo) samotné poradie hráčov.



Obrázok 43. Schéma komunikácie medzi aktivitami.

Tieto informácie sú spracované v metóde `onActivityResult()`, ktorá podľa navrátených výsledkov nastaví poradie páikarov. V prípade, že bolo výsledné poradie validné je nastavené dané poradie, pokiaľ nie, je použité poradie z databáze. Vybrané tími a nastavené poradie sa následne odošlú hernej aktivite a hra začína.

5.3.2 Aktivity zápisu hry



Obrázok 44. zloženie akvity hry.

Táto aktivita je špeciálne prispôsobená na to, aby mohla obsahovať viac obrazoviek. Každá obrazovka je tvorená fragmentom. Fragment je samostatná jednotka, ktorá je popísaná vlastnou triedou a má vlastný XML layout, ktorý popisuje jej design. Ide o súčasť aktivity a životný cyklus fragmentu je priamo prepojený so životným cyklom aktivity, ktorej prislúcha. Jeden fragment je možné použiť vo viacerých aktivitách, avšak každá inštancia fragmentu je spätá práve s jednou aktivitou. V aktivite pre zápis je použitý adaptér, ktorý obsahuje štyri fragmenty, každý fragment predstavuje jednu obrazovku. Pri geste prechodu medzi obrazovkami sa z adaptéru vyberie prislúchajúca inštancia fragmentu, ktorá sa následne zobrazí na obrazovke. V jednom okamihu je vždy na obrazovke viditeľný iba

jeden fragment. Zloženie fragmentov je vidieť na obrázku 41. Okrem toho, že má každý fragment prisluchajúcu triedu má samostatnú triedu taktiež celá aktivita. Táto trieda sa stará o všetky eventy stlačení tlačidiel a reaguje na ne volaním príslušných metód. Každé tlačidlo má vlastný event handler, ktorý reaguje na stlačenie špecifického tlačidla.

Ukážka handleru ktorý reaguje na stlačenie tlačidla Out :

```
public void onOutButtonClick(View view){
    // Pokial nie je nikto označený vyskoč z handleru a varuj užívateľa
    if(mRecorder.mClickedPlayerIndex >= 8){
        Toast.makeText(this, "You didnt select any
            player!", Toast.LENGTH_LONG).show();
        return;}
    // Získaj z modelu pozíciu a index outovaného hráča
    int[] positions =
mRecorder.countPlayerAndBase(mRecorder.mClickedPlayerIndex);
    // Zapiš údaje o outovanom hráčovi do Modelu a na ScoreBoard
mScoreBoard.addOutedPlayerToBoard(positions);
mRecorder.addOutedPlayerToRecord(positions);
    // Vymaž piktogram outovaného hráča
gameScreen1Fragment.outBatter(mRecorder.mClickedPlayerIndex);
    // Zruš označenie hráča
gameScreen1Fragment.resetSelection();
    // Zapiš do Modelu že méta je voľná
mRecorder.freeBase(mRecorder.mClickedPlayerIndex);
    // Zapiš do Modelu že sa zvýšil aktuálny počet Outov
mRecorder.mActualOutCount += 1;
    // Zobraz záznam
displayScore();}
```

Na ukážku kódu vyššie je popísaná reakcia aplikácie na stlačenie tlačidla Out, ktoré outuje označeného hráča. Hráč sa označuje kliknutím na piktogram. Okrem handlerov na všetky stlačenie tlačidiel táto trieda obsahuje taktiež metódu, ktorá prepisuje funkcionality systémového tlačidla „späť“. Toto tlačidlo v aplikácii slúži na krok späť v zápise. Odmaže teda poslednú položku v zázname a prípadne spätne prepíše akcie, ktoré sú s ňou spojené.

Ukážka časti metódy na vrátenie posledného zápisu :

```
@Override
public void onBackPressed(){
    // Ak bol posledný zápis vyutovanie hráča
    else if (mRecorder.isPlayerOut()){
        // Obnov piktogram vyoutovaného hráča a odpočítaj počet strikov
String s = mRecorder._Record.getLast().toString();
        int player = Character.getNumericValue(s.charAt(3));
        gameScreen1Fragment.renewBatter(player-1);
        mRecorder.mActualOutCount -= 1;}
    // Zmaž záznam z Modelu a zobraz aktuálny záznam
mRecorder.removeLastEntryFromRecord();
    displayScore();}
```

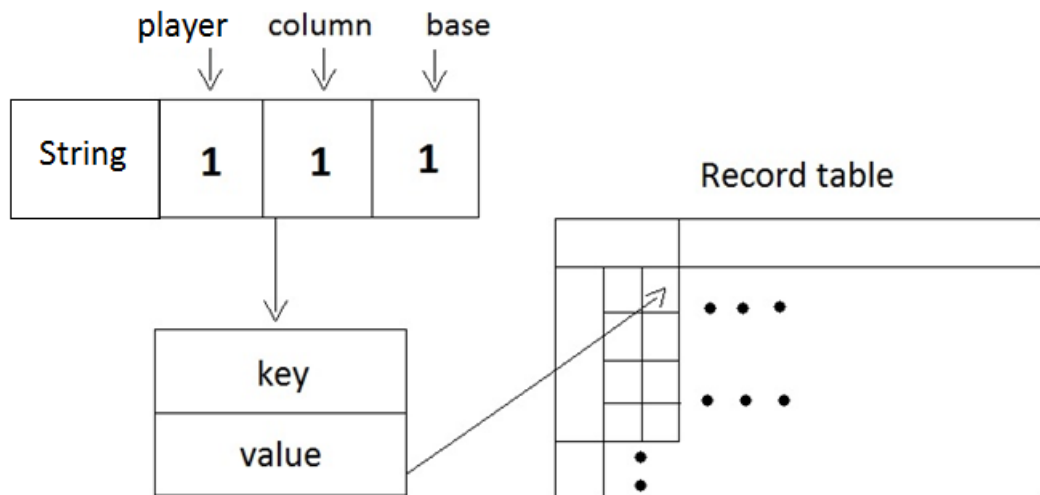
Hlavným fragmentom je fragment na plný zápis zápasu. V konštruktoze tohoto fragmentu sa inicializuje tabuľka skóre a všetky piktogramy bežcov a nastavujú sa listeners, ktoré sa starajú o udalosti týkajúce sa daného piktogramu. Každý piktogram hráča má nastavené

dve listenery. Jeden sa stará o reakciu na kliknutie a druhý o reakciu na pridrżanie prsta na hráčov. Listener na obyčajné kliknutie označí zakliknutého hráča. Druhý listener, na pridrżanie hráča začne drag event, ktorý umožní hráča posunúť na inú métu. Tento fragment ďalej obsahuje metódy, ktoré manipulujú z piktogramami bežcov. Medzi tieto metódy patria metódy :

1. `strikeOutBatter(int player)` - Skryje piktogram daného hráča. Metóda sa používa pri outovaní hráča.
2. `renewBatter(int player)` - Opätovne zobrazí daného hráča. Metóda sa používa pri obnovení vyoutovaného hráča.
3. `getReachedBase(int id)` - Vrátí métu na ktorú bol presunutý bežec. Metóda sa používa pri zápise presunutia hráča na inú métu.
4. `setAttackingTeam(int team)` - Nastaví grafické údaje na obrazovke pri zmene tímu. Metóda sa používa pri zmene útočiaceho tímu.
5. `movePlayerToBase(int player, int base)` - Posunie zadaného hráča na zadanú métu. Táto metóda sa používa pri nútenom posune hráčov po métách, prípadne pri spätnom vrátení.
6. `setActualOutCount()` – Získa z modelu dáta o aktuálnom počte outov útočiaceho tímu a zobrazí ich na ScoreBoarde.
7. `clearFieldOnInningChange()` - Metóda vymaže všetkých hráčov z ihriska pri zmene inningu.

Ďalším fragmentom je fragment, ktorý zobrazuje tabuľku priebehu zápasu. Tento fragment obsahuje event handler, ktorý zavolá metódu na naplnenie tabuľky iba vtedy, ak sa tabuľka dostane na obrazovku a je viditeľná. V tomto prípade sa volá metóda `fillTable()`, ktorá tabuľku naplní podľa aktuálne útočiaceho tímu. Metóda z modelu vytiahne dátovú štruktúru, ktorá bola podrobne popísaná v kapitole 5.2 . V prvom kroku metóda vyplní záhlavie tabuľky hráčmi aktuálne útočiaceho tímu. Následne si z dátovej štruktúry `TableRecord` zloží trojciferné číslo zložené z čísla hráča, stĺpca a méty. Po získaní tohoto získa metóda z Hash mapy presnú pozíciu bunky na zápis. Táto Hash mapa obsahuje práve trojciferné zložené číslo ako kľúč, ktorý je priradený k odpovedajúcej bunke v tabuľke. K samotnému textu, ktorý je taktiež súčasťou `TableRecordu` sa následne podľa príznakov

priradia prislúchajúce piktogramy a nastaví sa farebné zobrazenie záznamu podľa toho či bol hráč vyoutovaný, prípadne, či získal pre svoj tím bod. Na *obrázku 45* je vidieť získanie pozície k zápisu.



Obrázok 45. Schéma zápisu dát do tabuľky.

Tretím fragmentom je fragment, ktorý zobrazuje štatistiky aktuálneho pálkara. Tento fragment obsahuje v podstate jednu metódu, ktorá zobrazí aktuálne štatistiky na obrazovke. Tieto údaje získa aktivita z modelu. Z modelu sa do aktivity zavolá aktuálny hráč na pálke. Trieda hráča potom obsahuje metódy na výpočet štatistík.

Ukážka metódy triedy Player na výpočet štatistík :

```
public double getStrokeTypeRate(int type) {
    switch (type) {
        case 1:
            return (double)mStraightCount / (double)mTotalCount;
        case 2:
            return (double)mBendCount / (double)mTotalCount;
        case 3:
            return (double)mBounceCount / (double)mTotalCount;
    }
    return 0;
}
```

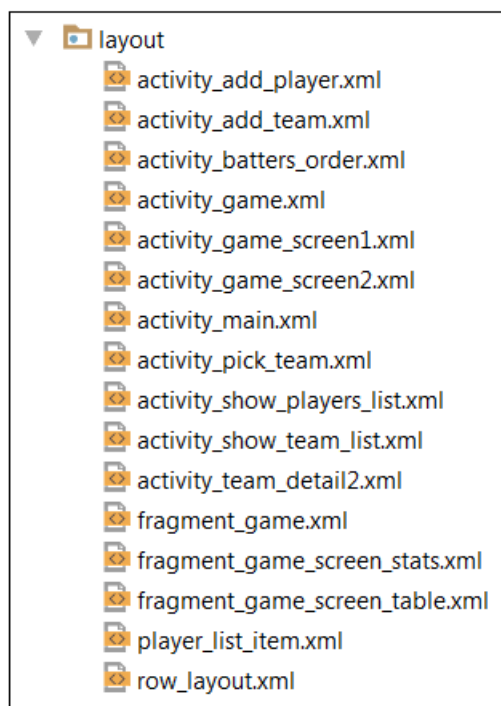
Vyššie je vidieť ukážka spočítania štatistík, kedy daná metóda z členských premenných triedy Player spočíta pravdepodobnosť druhu odpalu a vráti túto pravdepodobnosť späť do aktivity. Výstup sa následne upraví a zobrazí na obrazovke.

Posledným fragmentom v aktivite zápisu hry je fragment, ktorý umožňuje spätný prepis priebehu zápasu a zobrazuje získané body hráčov v danej zmene. Na vrchu obrazovky je možné vybrať zmenu a tím. Pri každej zmene v týchto dvoch objektoch typu Spinner sa podľa vybraných údajov z modelu získá dátová štruktúra VirtualTable. Tá následne podľa vybranej zmeny a tímu nastaví stavy odpovedajúcich Checkboxov. Pri spätnej zmene

zápisu sa zmení hodnota odpovedajúceho Checkboxu a táto zmenená hodnota sa následne zapíše do tabuľky priebehu zápasu.

5.4 XML Layout

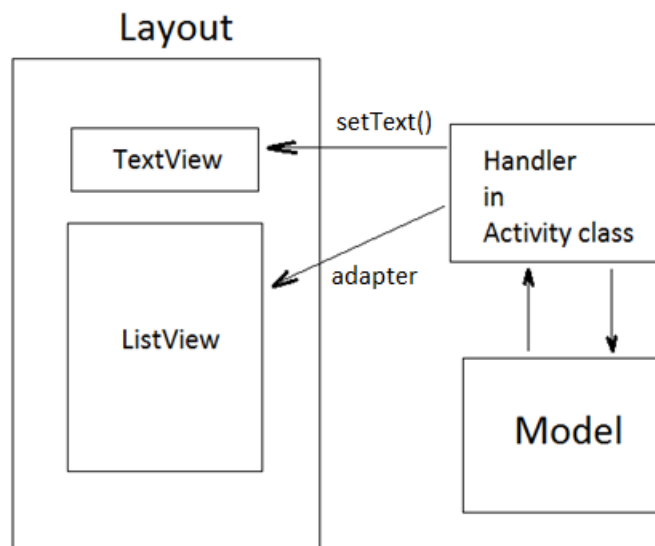
Rozloženie všetkých obrazoviek aplikácie je popísané v balíčku Layout. Tento balíček obsahuje všetky XML súbory, ktoré popisujú aktivity a fragmenty aplikácie. Vo väčšine aktivít je použité lineárne rozloženie prvkov. Výnimkou sú obrazovky na zaznamenávanie hry, v ktorých je použité relatívne rozloženie. Obrazovka s tabuľkou priebehu zápasu zase používa tabuľkové rozloženie.



Obrázok 46. Obsah balíčka layout.

V aktivitách Team manažera sa layout skladá z textových polí (TextView), vstupných textových polí (EditText), bežných tlačidiel (Button) a zoznamu položiek (ListView). ListView je kontajner, ktorý je možné plniť dynamicky určitou dátovou štruktúrou pomocou objektu adaptéra. Tento objekt sa po naplnení priradí danému ListView. Popis rozloženia položky v ListView je špecifikovaný v samostatnej triede (row_layout). Taktiež všetky TextView v týchto aktivitách sú plnené dynamicky. V triede aktivity je podľa špecifického id vytvorená inštancia potrebného prvku TextView a metódou je do nej zapísaný odpovedajúci text podľa práve načítaného hráča alebo tímu. Pri prvkoch typu EditText sa taktiež v triede aktivity získa inštancia tohoto prvku a naopak pri metóde

kliknutia na tlačidlo sa z EditTextu získajú vložené dáta, ktoré sa následne ďalej spracujú. Na obrázku 47 je zobrazený princíp vloženia dát na obrazovku.



Obrázok 47. Schematické zobrazenie zápisu dát na View.

V aktivitách samotného zápisu hry sú použité taktiež prvky typu TextView, ktoré sú dynamicky plnené dátami z modelu aplikácie. Pozadie a jednotlivé piktogramy, ktoré sú v aplikáciách použité sú tvorené prvkom ImageView. ImageView je kontajner, ktorý obsahuje obrázkový zdroj (png prípadne bitmapu). Všetky tlačidlá v aktivitách záznamu zápasu majú vo svojom XML popise v parametri android:onClick určenú metódu, ktorá je vždy definovaná v triede príslušnej aktivity. Táto metóda je potom volaná vždy v momente stlačenia tlačidla. Špecifikú úpravu má fragment zobrazujúci tabuľkový záznam zápasu. V tomto fragmente je použité tabuľkové rozloženie. Ide však o rozloženie, nie samotnú tabuľku. TableLayout teda iba definuje rozloženie prvkov v tabuľkovom rastry. V aplikáciách sú týmito prvkami textové polia TextView. Tabuľkové rozloženie implicitne nepovoľuje zobrazenie ohraničenia tabuľky. V aplikáciách je teda vytvorené ohraničenie bunky pomocou drawable XML zdrojového súboru a toto ohraničenie je následne použité ako pozadie jednotlivých TextView v tabuľkovom rozložení čím je vytvorená výsledná tabuľka. Každé bunka v tabuľke má potom pridelené špecifické ID cez ktorú je možné plniť jej obsah. Vo fragmentoch plného zápisu a fragmente na zobrazenie štatistík aktuálneho pálkara je použité podobné rozloženie prvkov, rozdiel je iba v zdrojom dát ktorým sú dané fragmenty plnené. Vo fragmente spätného zápisu sú použité taktiež elementy typu CheckBox ktorých stav je menení vzhľadom na dáta z modelu. Pri zmene stavu CheckBoxu užívateľom je zase pomocou príslušného listeneru táto informácia odchytená a poslaná späť na Model.

ZÁVĚR

Cieľom tejto práce bolo vyvinutie aplikácie na zápis priebehu zápasu a následné spracovanie hráčskych štatistík. S dôrazom na zjednodušenie celého procesu zápisu, nakoľko ten je pre bežného hráča komplikovaný. Táto aplikácia má slúžiť na zápis priebehu zápisu a následné spracovanie týchto údajov.

V teoretickej časti sú popísané tri oblasti, ktoré sú potrebné k pochopeniu funkcionality aplikácie. Prvou je popis samotnej platformy Android a aplikácií pracujúcich na tomto operačnom systéme. Nasleduje sekcia popisu základou samotnej hry a rešerš o dostupných aplikáciách, ktoré slúžia na zaznamenávanie prebehu zápasu. V poslednom bode sú rozobrané základné dynamické dátové štruktúry, ktoré sú použité v samotnej aplikácii na zaznamenávanie dát.

Následne bola vyvinutá samotná aplikácia. Aplikácie je vytvorená natívne pre platformu Android. Má dve hlavné časti. Jedna sa stará o správu databáze. Ide o jednoduché užívateľské rozhranie, ktoré vkladá, maže a zobrazuje dáta z databáze užívateľovi, ktorý tak môže spravovať vlastné tímy a hráčov. Druhá časť sa stará o samotný zápis priebehu zápasu. Pozostáva zo štyroch obrazoviek, medzi ktorými je možné ľubovoľne listovať. Dve z týchto obrazoviek slúžia na samotný zápis zápasu. Tretia je tabuľka, ktorá zobrazuje doterajší priebeh zápasu. Tá štvrtá potom obrazovka, ktorá zobrazuje štatistiky aktuálneho pákara. Slúži teda ako určitý napovedač, ktorý užívateľovi radí, kde a ako bude pravdepodobne aktuálny pákar odpalovať. Aplikácia v neposlednom rade umožňuje export týchto výsledkov na ďalšie spracovanie.

Praktická časť práce sa skladá z dvoch hlavných kapitol. Tá prvá je popis jednotlivých obrazoviek a ich funkcionality z pohľadu užívateľa. Súčasťou tejto časti je aj príloha užívateľská príručka, ktorá popisuje základné herné situácie a postup ich zaznamenania aplikáciou. V druhej kapitole je najskôr popísaná architektúra a vnútorné fungovanie aplikácie. Následne sú popísané jednotlivé časti aplikácie a úlohy o ktoré sa starajú.

Hlavným rozdielom vytvorenej aplikácie oproti aplikáciám z obdobnou funkcionality je jednoduchosť zápisovej logiky, kedy je možné každú hernú situáciu zaznamenať na relatívne malý počet úkonov. Aplikácia je teda prínosom pre zapisovateľov, ktorí nemajú mnoho skúseností so samotnou hrou a nezvládli by tak zložitý zápis, alebo samotných hráčov, ktorí k zdĺhavému zápisu nemajú dostatok času.

SEZNAM POUŽITÉ LITERATURY

- [1] Historie Androidu v kostce aneb Od verze 1.0 až po Android M. KILIÁN, Karel. *Svět Androida* [online]. Praha, 2014 [cit. 2016-05-06]. Dostupné z: <http://www.svetandroida.cz/historie-androidu-201506>
- [2] Krátké ohlédnutí za historií Androidu. MYSLIVEČEK, David. *Svět Androida* [online]. Praha, 2014 [cit. 2016-05-06]. Dostupné z: <http://www.svetandroida.cz/kratke-ohlednuti-za-historii-androidu-201305>
- [3] A Very Brief History of Android. GRAY, Chuck. *Librarypoint* [online]. Fredericksburg [cit. 2016-05-06]. Dostupné z: http://www.librarypoint.org/android_history
- [4] Android - Architecture. *Tutorialspoint* [online]. Hyderabad, 2016 [cit. 2016-05-06]. Dostupné z: http://www.tutorialspoint.com/android/android_architecture.htm
- [5] Android Architecure. *Javatpoint* [online]. Noida, c2011-2016 [cit. 2016-05-06]. Dostupné z: <http://www.javatpoint.com/android-software-stack>
- [6] An Overview of the Android Architecture. *Techotopia* [online]. 2016 [cit. 2016-05-06]. Dostupné z: http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture
- [7] Android Interfaces and Architecture. *Source.android* [online]. [cit. 2016-05-06]. Dostupné z: <https://source.android.com/devices/>
- [8] Application Fundamentals. *Developer.android* [online]. [cit. 2016-05-06]. Dostupné z: <http://developer.android.com/guide/components/fundamentals.html#Components>
- [9] Starting an Activity. *Developer.android* [online]. [cit. 2016-05-06]. Dostupné z: <http://developer.android.com/training/basics/activity-lifecycle/starting.html>
- [10] Saving Data in SQL Databases. *Developer.android* [online]. [cit. 2016-05-06]. Dostupné z: <http://developer.android.com/training/basics/data-storage/databases.html>
- [11] Managing Projects Overview. *Developer.android* [online]. [cit. 2016-05-06]. Dostupné z: <http://developer.android.com/tools/projects/index.html>
- [12] Android Studio Overview. *Developer.android* [online]. [cit. 2016-05-06]. Dostupné z: <http://developer.android.com/tools/studio/index.html>

- [13] Datové struktury: Fronta. *Algoritmy.net* [online]. Brno, 2015 [cit. 2016-05-06]. Dostupné z:<https://www.algoritmy.net/article/28/Fronta>
- [14] LACKO, Ľuboslav. Vývoj aplikací pro Android. 1. vyd. Brno: Computer Press, 2015, 472 s. ISBN 978-80-251-4347-6.
- [15] GRIFFITHS, Dawn a David GRIFFITHS. Head First Android Development. O'Reilly Media, 2015. ISBN 978-1-4493-6213-3.
- [16] SCHILDT, Herbert. Mistrovství - Java. 1. vyd. Brno: Computer Press, 2014, 1224 s. Mistrovství. ISBN 978-80-251-4145-8.
- [17] Android SQLite Essentials. Packt Publishing, 2014. ISBN 1783282959.
- [18] CASTRO, Elizabeth a Bruce HYSLOP. HTML5 a CSS3: názorný průvodce tvorbou WWW stránek. 1. vyd. Brno: Computer Press, 2012, 439 s. ISBN 978-80-251-3733-8.
- [19] Příručka pro zapisovatele v softbalu [online]. Praha, 2005 [cit. 2016-01-28]. Dostupné z: <http://www.pegas.sedlcany.cz/files/documents/zapisovatel.pdf>
- [20] SÜSS, Vladimír. Praha: BACH s.r.o., 2000.
- [21] IScore. *IScore Baseball* [online]. [cit. 2016-05-17]. Dostupné z: <http://iscoresports.com/baseball/>
- [22] Score Finger. *Score Finger Official* [online]. [cit. 2016-05-17]. Dostupné z: <http://scorefinger.com/>
- [23] Linked List. *Oracle Documentation* [online]. 2016 [cit. 2016-05-17]. Dostupné z:<https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html>
- [24] Algoritmy. *Algoritmy* [online]. Brno, 2015 [cit. 2016-05-17]. Dostupné z: <https://algoritmy.net/>
- [25] CORMEN, Thomas H. *Introduction to algorithms*. 3rd ed. Cambridge, Mass.: MIT Press, c2009. ISBN 02-625-3305-7.
- [26] Binary Tree. *Encrypted* [online]. 2010 [cit. 2016-05-17]. Dostupné z:<https://encrypt3d.wordpress.com/category/data-structures/binary-tree/>
- [27] Baseball Field. *Conceptdraw* [online]. Odessa, 2016 [cit. 2016-05-17]. Dostupné z:<http://www.conceptdraw.com/diagram/template-baseball-field>
- [28] Android Developers. *Android* [online]. 2016 [cit. 2016-05-17]. Dostupné z:<https://developer.android.com/index.html>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface.
APK	Android Application Package.
ART	Android Runtime.
FIFO	First In First Out
LIFO	Last In First Out
GPS	Global Positioning System
NFC	Near Field Communication
MVC	Model View Controller
SDK	Software Development Kit
URI	Uniform Resource Identifier
USB	Universal Serial Bus
XML	Extensible Markup Language
Listener	Naslúchač, prvok ktorý zachytáva systémové udalosti
Handler	Obsluha, prvok ktorý sa stará o reakciu na udalosti

SEZNAM OBRÁZKŮ

<i>Obrázok 1. Prehľad verzií Androidu.[28]</i>	11
<i>Obrázok 2. Prehľad rozšírenia jednotlivých verzií.[28]</i>	13
<i>Obrázok 3. Ukážka Android systémov od rôznych výrobcov.[3]</i>	14
<i>Obrázok 4. Architektúra systému Android.[4]</i>	15
<i>Obrázok 5. Životný cyklus aktivity.[8]</i>	20
<i>Obrázok 6. Štruktúra projektu v AS.[11]</i>	23
<i>Obrázok 7. Monitorovanie výkonu v Android studiu.[12]</i>	25
<i>Obrázok 8. Allocation tracker v Android Studiu.[12]</i>	25
<i>Obrázok 9. Schema softbalového ihriska.[27]</i>	26
<i>Obrázok 10. Obrazovka pre záznam.[21]</i>	28
<i>Obrázok 11. Dodatočná tabuľka na zápis vyoutovania hráča. [21]</i>	29
<i>Obrázok 12. Obrazovka pri odpale lopty. [21]</i>	30
<i>Obrázok 13. Obrazovka k záznamu pohybu lopty. [21]</i>	30
<i>Obrázok 14. Obrazovka pre záznam pohybu bežcov.[21]</i>	31
<i>Obrázok 15. Obrazovka pre záznam zápasu Scorefinger. [22]</i>	32
<i>Obrázok 16. Jednosmerný lineárny zoznam.[24]</i>	34
<i>Obrázok 17. Obojsmerný lineárny zoznam[24]</i>	34
<i>Obrázok 18. Cyklický lineárny zoznam.[24]</i>	34
<i>Obrázok 19. Štruktúra typu FIFO.[24]</i>	35
<i>Obrázok 20. Štruktúra typu LIFO.[24]</i>	36
<i>Obrázok 21. Binárny strom.[26]</i>	36
<i>Obrázok 22. Princíp Hashovacej tabuľky.[24]</i>	37
<i>Obrázok 23. Náhľad na obrazovku správy tímov.</i>	40
<i>Obrázok 24. Náhľad na obrazovku správy hráčov.</i>	41
<i>Obrázok 25. Náhľad na obrazovku výberu tímov.</i>	42
<i>Obrázok 26. Náhľad na obrazovku nastavenia poradia páľkarov.</i>	43
<i>Obrázok 27. Náhľad na obrazovku zápisu zápasu.</i>	44
<i>Obrázok 28. Náhľad na obrazovku pre zobrazenie štatistík.</i>	45
<i>Obrázok 29. Náhľad na obrazovku zobrazujúcu priebeh zápasu.</i>	46
<i>Obrázok 30. Náhľad na obrazovku</i>	47
<i>Obrázok 31. Štruktúra Java tried aplikácie.</i>	48
<i>Obrázok 32. Schéma jednotlivých častí aplikácie.</i>	49

<i>Obrázok 33. Štruktúra zdrojov aplikácie.</i>	<i>49</i>
<i>Obrázok 34. Prepojenie tabuliek v databázi.</i>	<i>50</i>
<i>Obrázok 35. Triedy balíčka na správu databáze.</i>	<i>51</i>
<i>Obrázok 36. Ukážka práce s Cursorom.</i>	<i>52</i>
<i>Obrázok 37. Triedy balíčka starajúce.....</i>	<i>53</i>
<i>Obrázok 38. Dátová štruktúra na záznam priebehu</i>	<i>53</i>
<i>Obrázok 39. Príklad zápisu do lineárneho zoznamu.</i>	<i>54</i>
<i>Obrázok 40. Dátová štruktúra pre skrátená zápis.</i>	<i>55</i>
<i>Obrázok 41. Schéma prepojenia aktivít.</i>	<i>58</i>
<i>Obrázok 42. Prechod medzi aktivitami Team manažera.</i>	<i>58</i>
<i>Obrázok 43. Schéma komunikácie medzi aktivitami.</i>	<i>61</i>
<i>Obrázok 44. zloženie akvity hry.</i>	<i>61</i>
<i>Obrázok 45. Schéma zápisu dát do tabuľky.</i>	<i>64</i>
<i>Obrázok 46. Obsah balíčka layout.</i>	<i>65</i>
<i>Obrázok 47. Schematické zobrazenie zápisu dát na View.</i>	<i>66</i>
Príloha 1	
<i>Obrázok 1.. Obrazovky tímového manažéra.</i>	<i>74</i>
<i>Obrázok 2. Obrazovky výberu tímov.</i>	<i>75</i>
<i>Obrázok 3. Náhľad na hlavnú obrazovku.</i>	<i>76</i>
<i>Obrázok 4. Náhľad na herné obrazovky.</i>	<i>77</i>
<i>Obrázok 5. Priebeh zápisu hernej situácie.</i>	<i>78</i>
<i>Obrázok 6. Priebeh zápisu hernej situácie.</i>	<i>79</i>
<i>Obrázok 7. Priebeh zápisu hernej situácie.</i>	<i>80</i>
<i>Obrázok 8. Priebeh zápisu hernej situácie.</i>	<i>81</i>
<i>Obrázok 9. Priebeh zápisu hernej situácie.</i>	<i>82</i>
<i>Obrázok 10. Priebeh zápisu hernej situácie.</i>	<i>83</i>
<i>Obrázok 11. Priebeh zápisu hernej situácie.</i>	<i>84</i>
<i>Obrázok 12. Priebeh zápisu hernej situácie.</i>	<i>85</i>
<i>Obrázok 13. Priebeh zápisu hernej situácie.</i>	<i>86</i>
<i>Obrázok 14. Priebeh zápisu hernej situácie.</i>	<i>87</i>

SEZNAM PŘÍLOH

Príloha P I: Uživatelská příručka.

PŘÍLOHA P I: UŽIVATELSKÁ PŘÍRUČKA

5.5 Správca týmov

Prvou súčasťou aplikácie Softball Tracker je jednoduchý manažér na správu tímov a hráčov. Tento manažér sa skladá z dvoch hlavných obrazoviek. Prvou je obrazovka na správu tímov (ľavá obrazovka, *obrázok 1*). Na tejto obrazovke je možné vyplnením formulára po stlačení tlačidla vložiť nový tím. Označením tímu a stlačení tlačidla Delete Team je označený tím odstránený z databáze. Odstránením tímu sú automaticky odstránení aj všetci jeho hráči.

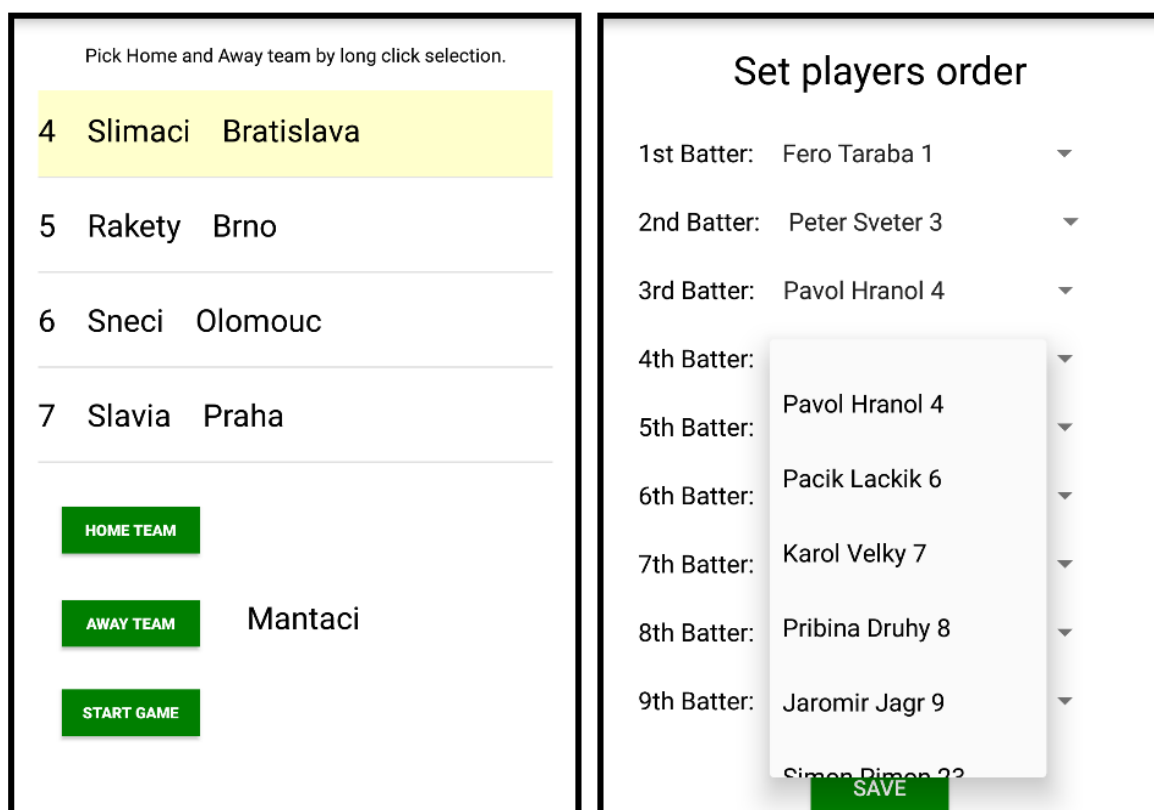
The image shows two side-by-side screenshots of a web application interface for team management. The left screenshot, titled 'List of teams', displays a numbered list of six teams: 1 Zraloci Zlin, 2 Mantaci Brno, 3 Buldoci Slusovice, 4 Slimaci Bratislava, 5 Rakety Brno, and 6 Sneci Olomouc. At the bottom of this list are two green buttons: 'ADD TEAM' and 'DELETE TEAM'. The right screenshot shows the detailed view for the 'Zraloci' team. It includes fields for 'Team name: Zraloci', 'Team City: Zlin', and 'Team Id: 1'. Below these are lists of players with their numbers and names: 45 Fero Taraba, 23 Jozo Maly, 45 Peter Sveter, 89 Pavol Hranol, 77 Pacik Lackik, 46 Karol Velky, and 58 Pribina Druhy. At the bottom of this view are two green buttons: 'ADD PLAYER' and 'DELETE PLAYER'.

Obrázok 1. Obrazovky tímového manažéra.

Po kliknutí na daný tím je otvorená druhá obrazovka (pravá obrazovka, *obrázok 1*). Na tejto obrazovke sú zobrazené základné info o danom tíme. Pod týmito informáciami sa nachádza zoznam hráčov daného tímu. Po stlačení tlačidla Add Player sa otvorí formulár pomocou, ktorého je možné danému tímu pridať nového hráča. Označením hráča kliknutím a následným stlačením tlačidla Delete Player je označený hráč vymazaný z daného tímu.

5.6 Spustenie hry

Po spustení novej hry je hráč presnutý na obrazovku výberu tímov (ľavá obrazovka, *obrázok 2*). Na vrchnej časti tejto obrazovky je zobrazený zoznam všetkých tímov v databázi. Užívateľ následne vyberie hrajúce tímy tak, že najskôr označí tím zo zoznamu a tlačidlom Home Team / Away Team mu ho pridá do zápasu. Ak je tím označený ako domáci, prípadne hosť je možné klikom na tento tím prejsť na ďalšiu obrazovku.

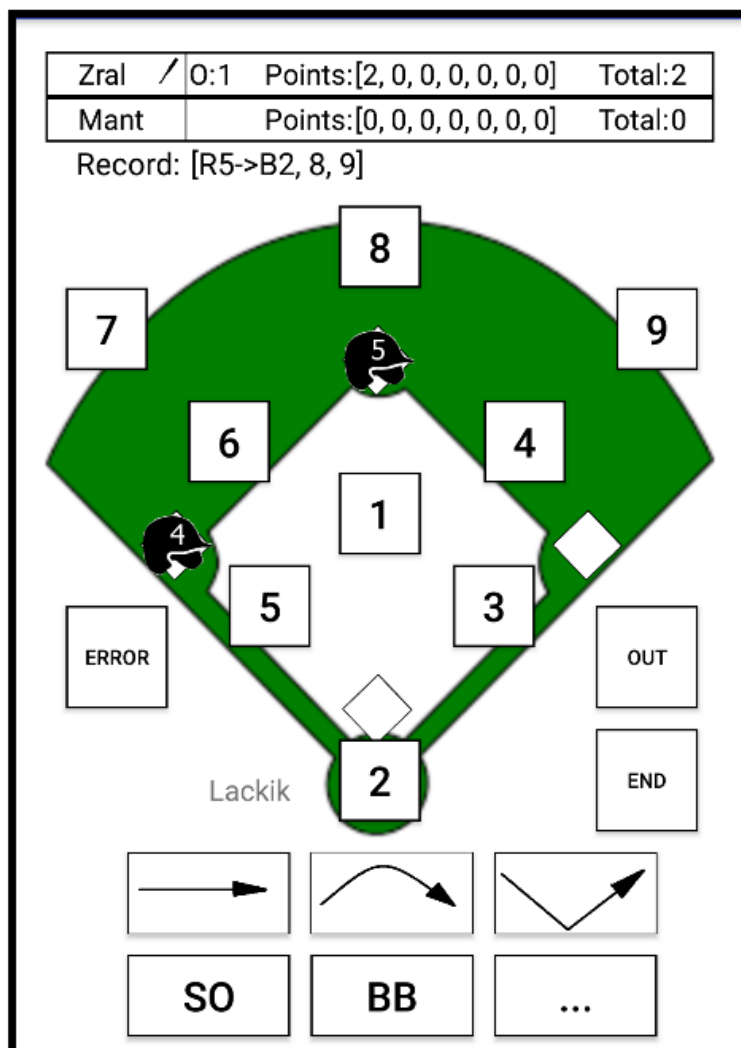


Obrázok 48. Obrazovky výberu tímov.

Na tejto obrazovke je možné nastaviť poradie páľkarov v zápase (pravá obrazovka, *obrázok 2*). Na tejto obrazovke je zobrazené poradie páľkarov. Každá poradová pozícia má pri sebe Spinner, ktorý obsahuje všetkých hráčov daného tímu. Pomocou týchto Spinnerov je možné priradiť na danú pozíciu hráča tímu. Na tejto obrazovke je potrebné zadať poradie všetkých hráčov a žiadny hráč nemôže byť zadaný na dvoch pozíciách. V prípade, že tieto dve podmienky niesú splnené je použité poradie hráčov z databáze. Toto poradie je použité taktiež v prípade, že užívateľ nezadal špecifické poradie páľkarov. Poradie je možné zadať iba pre jeden tím.

5.7 Zápisy hry

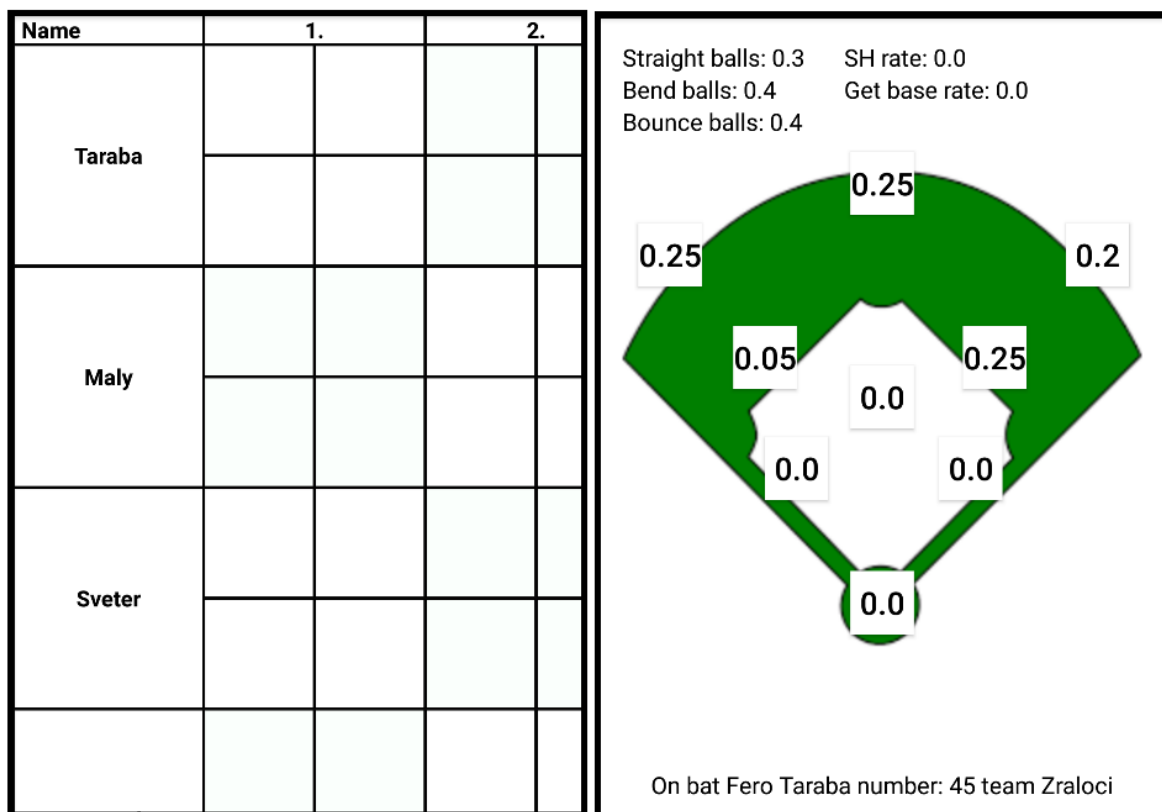
Hlavná obrazovka na plný zápis priebehu zápasu je vidieť na *obrázku 3*. Vo vrchnej časti obrazovky sa nachádza tabuľka, ktorá ukazuje aktuálny stav zápasu. Zobrazuje tím, ktorý je aktuálne na pólke, aktuálny počet outov pre útočiaci tím, body dosiahnuté v jednotlivých zmenách a celkové skóre zápasu. Pod touto tabuľkou sa nachádza záznam, ktorý zobrazuje situácie, ktoré užívateľ zadal v aktuálnom zápise.



Obrázok 3. Náhľad na hlavnú obrazovku.

Následuje samotné pole, v ktorom sú rozložený jednotlivý poliarci brániaceho tímu, označený číslami 1-9. Na vrcholoch bieleho kosoštvorca sú potom rozmiestnené jednotlivé mety, po ktorých sa pohybujú bežci. V spodnej časti obrazovky sú rozmiestnené tlačítka, ktoré slúžia na záznam priebehu hry. Záznam hry prebieha tak, že užívateľ zaznačí pohyb lopty v poli po odpale stlačením tlačidiel hráčov, ktorý sa do rozohry zapojili. V prípade, že sa daný hráč pri svojom kontakte s loptou dopustil chyby, musí užívateľ stlačiť pred

zadaním tohoto hráča tlačidlo Error. Trojica tlačidiel pod hracou plochou označuje druh odpalu. Tento údaj je dobrovoľný a je ho možné zadať v hociktorom bode zápisu rozohry. Na situáciu vyoutovania pálkara po troch chybných odpaloch a situáciu štyroch chybných nadhohoch slúžia tlačidlá SO respektíve BB. Pohyb hráča na metách sa zaznamenáva posunutím príslušného piktogramu hráča na danú métu. V prípade, že bol bežec na méte vyoutovaný bežca je potrebné označiť kliknutím a následne vyoutovať stlačením tlačidla Out. Systémové tlačidlo „back“ slúži v rámci tejto aktivity na odmazanie posledného vloženého záznamu, pričom taktiež vracia pohyb hráčov po metách a ich vyoutovanie. Tlačidlo v pravom dolnom rohu otvára ďalšie menu, ktoré obsahuje nie tak časté herné situácie. Tlačidlo End slúži na ukončenie rozohry.

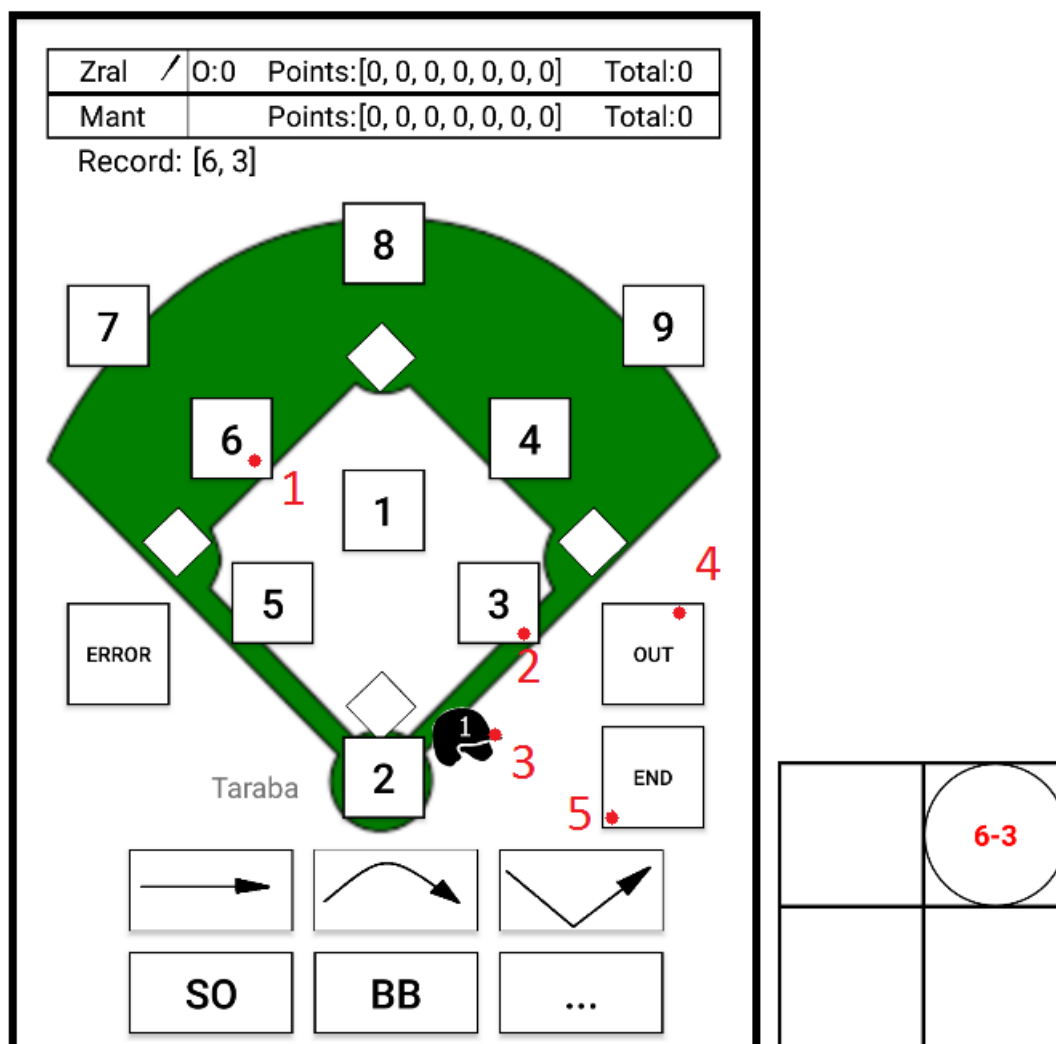


Obrázok 4. Náhl'ad na herné obrazovky.

Po stlačení tohoto tlačidla je do obrazovky záznamu priebehu zápasu (ľavá obrazovka na obrázku 4) vložená zadaná rozohra. Obrazovka na pravej strane obrázku 4 zobrazuje štatistiky aktuálneho pálkara vo všetkých jeho zaznamenaných zápasoch.

5.7.1 Odpal do ľavej časti vnútorného poľa, páľkar out

Popis situácie: Páľkar odpáľil loptu do ľavej časti vnútorného poľa. Loptu chytila spojka (6) a prihodila na prvú métu poliarovy (3), ktorý na prvej méte páľkara vyoutoval.



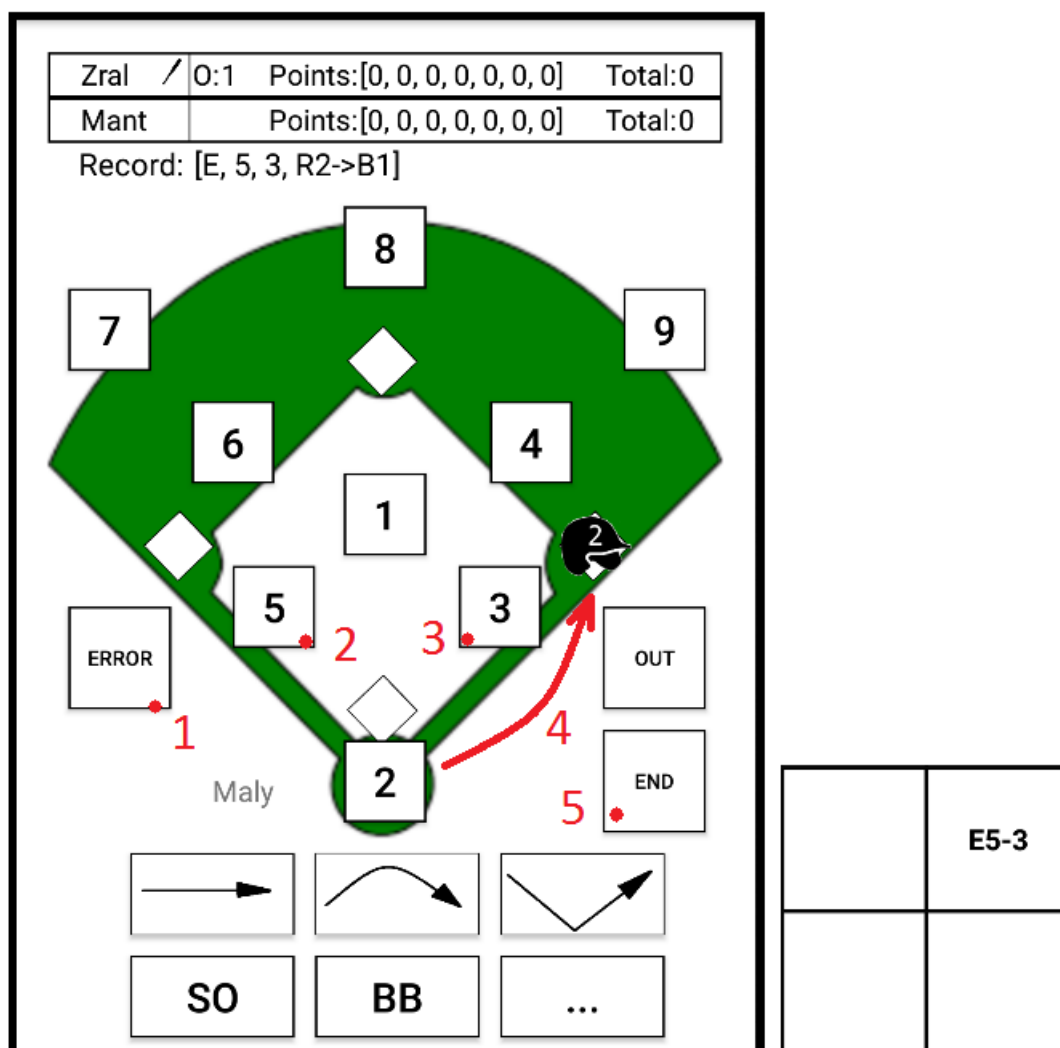
Obrázok 49. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačítko [6].
2. Následne stlačte tlačidlo [3], týmto ste zaznačili pohyb lopty.
3. Kliknutím označte páľkara, páľkar sa po kliku zvirazní, čím je signalizované, že je práve označený a ďalšia akcia sa týka jeho.
4. Stlačte tlačítko [Out] čím zapíšete že páľkar bol vyoutovaný.
5. Stlačte tlačidlo [END] čím ukončíte zápis situácie. Na páľku sa posunie ďalší hráč.

5.7.2 Odpaľ do ľavej časti vnútorného poľa, páľkar safe

Popis situácie: Páľkar odpáľil loptu do ľavej časti vnútorného poľa. Źiadna méta nie je obsadená. Poliar (5) zle spracuje loptu a prihadzuje na prvú métu, poliar (3). Páľkar na métu dobehol pred príhodom a získa prvú métu.



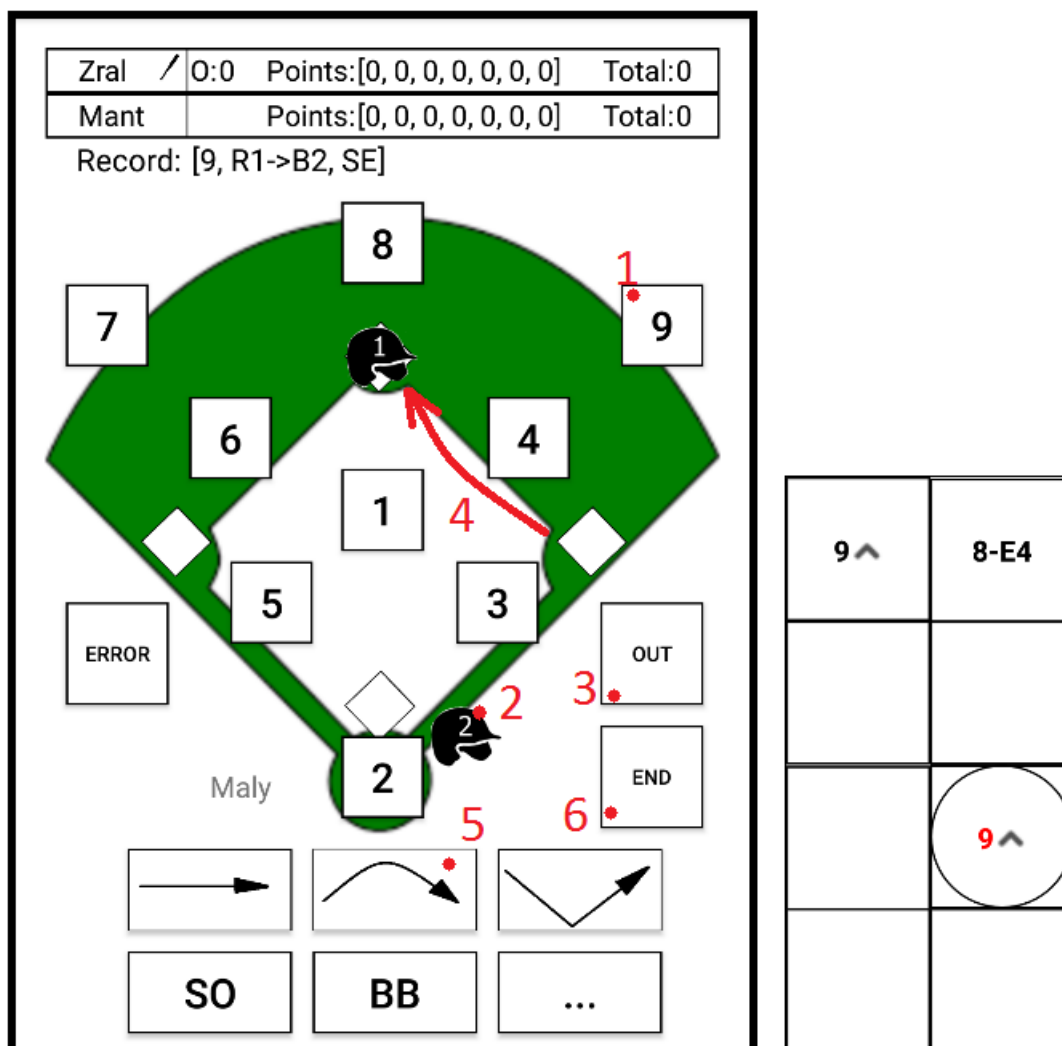
Obrázok 6. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [ERROR], ktoré signalizuje chybu následne označeného poliara.
2. Stlačte tlačidlo [5].
3. Stlačte tlačidlo [3], týmto ste označili pohyb lopty.
4. Následne po presunte páľkara na prvú métu.
5. Ukončíte zápis stlačením tlačidla [END], čím sa skončí zápis a páľkar sa stáva bežcom na prvej méte. Na páľku sa posunie nasledujúci hráč.

5.7.3 Vysoký odpal do pravého zadného pola

Popis situácie: Páľkar odpálil vysokým odpalom loptu do pravej časti vonkajšieho poľa. Prvá méta je obsadená bežcom. Poliar (9) chytá loptu zo vzduchu, priamo z odpalu. Páľkar je vyoutovaný a bežec na druhej méte stihol dobehnúť na druhú métu.



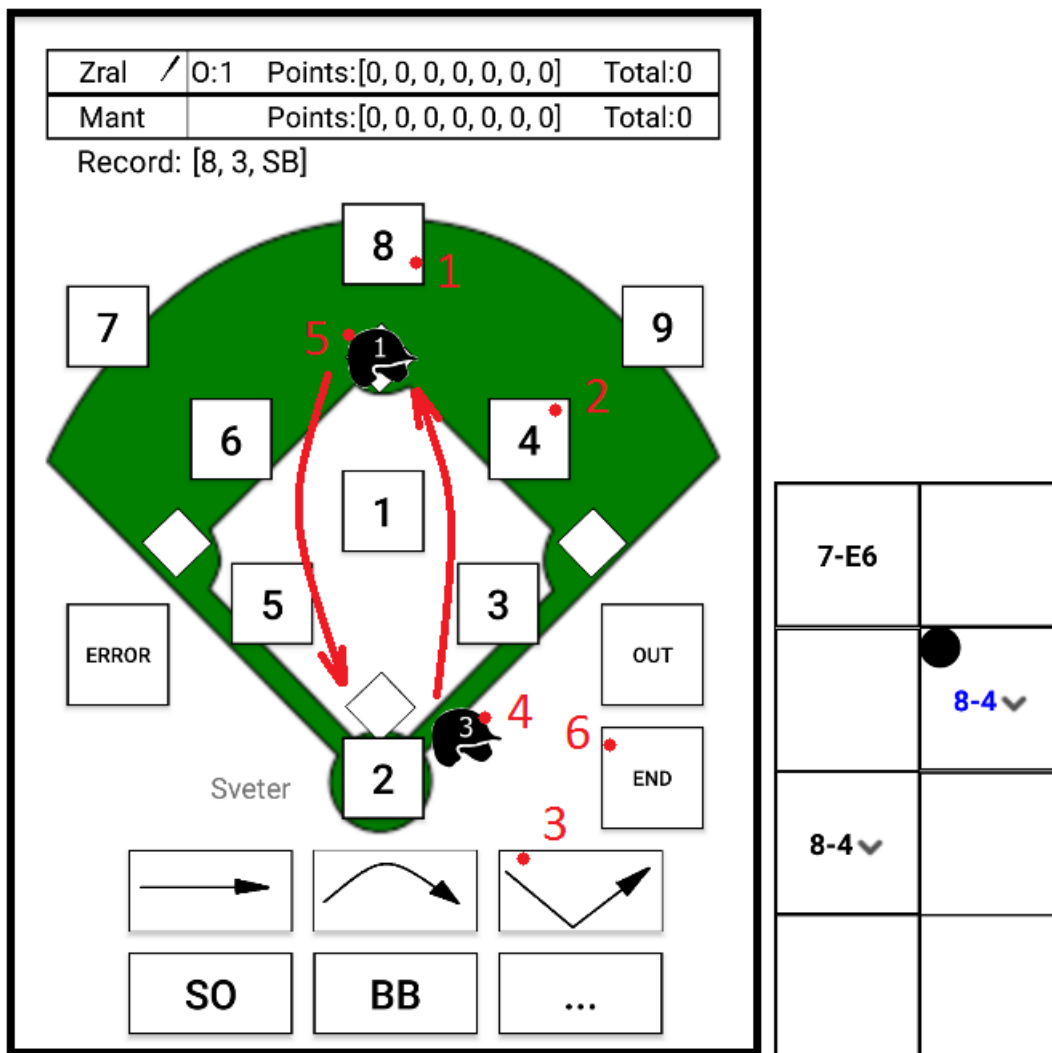
Obrázok 7. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [9].
2. Kliknutím označte páľkara.
3. Stlačte tlačidlo [OUT], čím zaznamenáte vyoutovanie páľkara.
4. Presunte bežca z prvej méty na druhú métu.
5. Stlačte tlačidlo vysokého odpalu (to môže byť stlačené v ľubovoľnom poradí).
6. Ukončite zápis stlačením tlačidla [END].

5.7.4 Odpal po zemi do stredného vonkajšieho poľa

Popis situácie: Páľkar odpálil odrazom od zeme do strednej časti vonkajšieho poľa. Druhá méta je obsadená bežcom. Poliar (8) loptu chytí a prihadzuje na druhú métu poliarovi (4). Páľkar získava druhú métu. Bežec z druhej méty dobieha na domácu métu a skóruje.



Obrázok 8. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [8].
2. Stlačte tlačidlo [3], týmto ste označili pohyb lopty.
3. Stlačte tlačidlo odrazeného odpalu.
4. Presunte páľkara na druhú métu.
5. Presunte bežca z druhej méty na domácu métu.
6. Ukončíte zápis stlačením tlačidla [END], bežec na domácej méte získava bod.

5.7.5 Prvá meta obsadená, páľkar BB

Popis situácie: Nadhadzovač nadhodil páľkarovy tri chybné nadhody, čím páľkar získava prvú metu. Prvá meta je obsadená bežcom, čím sa bežec z prvej méty nútene posúva na druhú metu.

Zral /	0:0	Points:[0, 0, 0, 0, 0, 0, 0]	Total:0
Mant		Points:[0, 0, 0, 0, 0, 0, 0]	Total:0
Record: [BB, R1->B2, BB, R2->B1]			

BB	8-E4
	BB

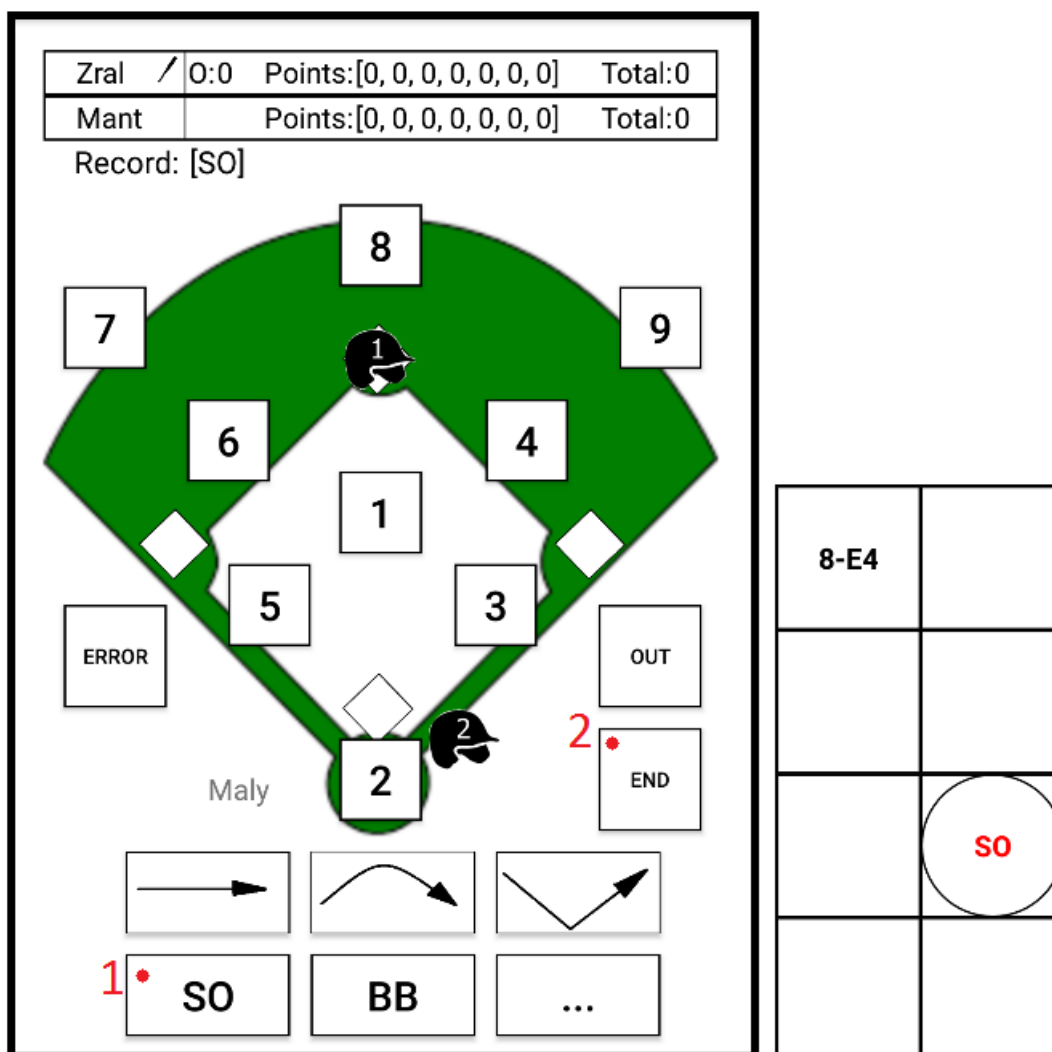
Obrázok 9. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [BB] .
2. Ukončíte zápis stlačením tlačidla [END] .
3. Páľkar je automaticky posunutý na prvú metu.
4. Bežec z prvej méty je tlačný a automaticky posunutý na druhú metu.

5.7.6 Druhá meta obsadená páľkar SO

Popis situácie: Páľkar je po troch chybných odpaloch Strike out. Druhá meta je obsadená bežcom.



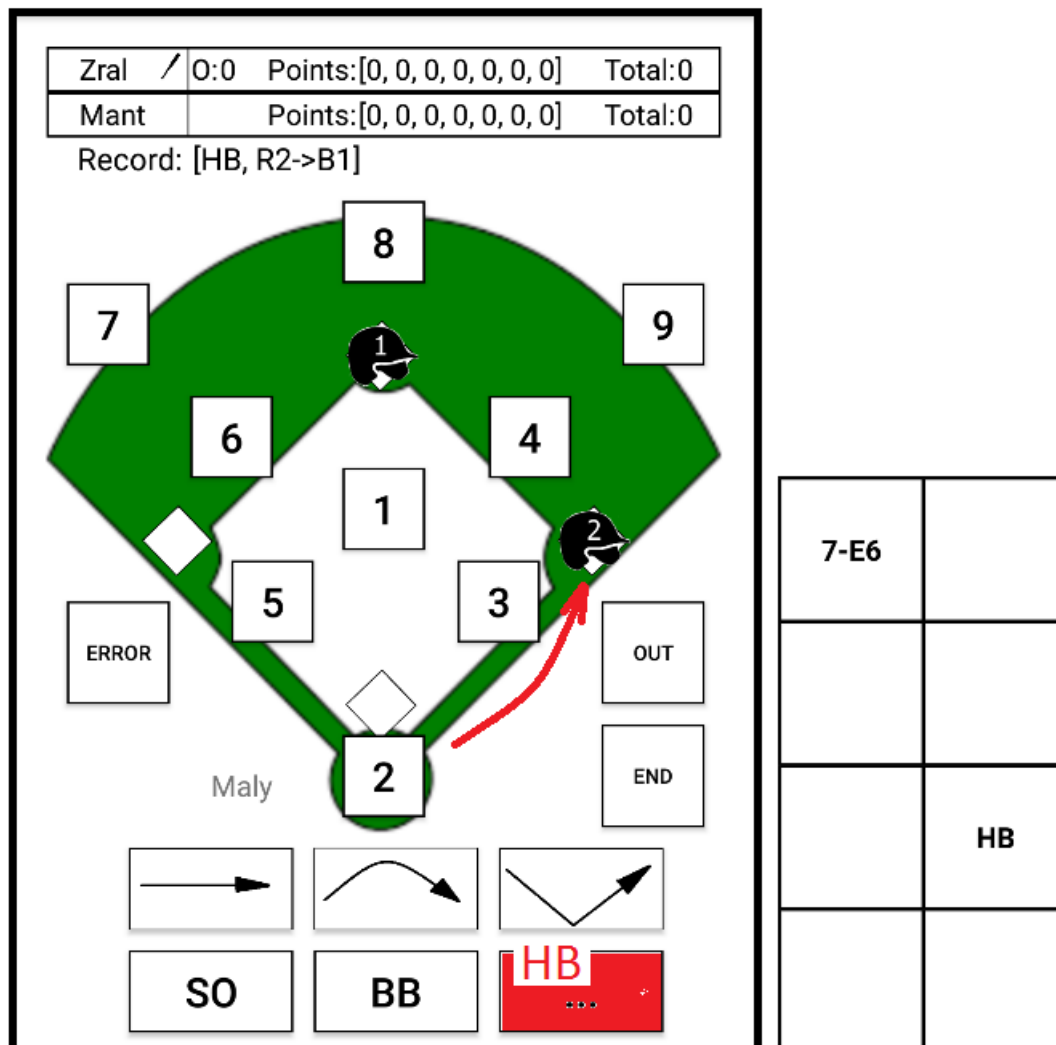
Obrázok 10. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [SO].
2. Ukončíte zápis stlačením tlačidla [END], čím ukončíte zápis. Páľkar je vyoutovaný, útočiacemu tímu je pripísaný jeden out a na páľku sa automaticky posúva ďalší hráč.

5.7.7 Druhá meta obsadená, páľkar HB

Popis situácie: Nadhadzovač svojim odpalom trafil páľkara. Na druhej méte je bežec, ktorý na nej po tejto situácii ostáva.



Obrázok 11. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [...], čím aplikácie otvorí menu z dodatočnými možnosťami špecifických herných situácií.
2. Z tohoto menu vyberte možnosť HB.
3. Ukončíte zápis stlačením tlačidla [END]
4. Páľkar je automaticky posunutý na prvú metu.
5. Bežec na druhej mete ostáva na svojom mieste.

5.7.8 Prvé dve méty obsadené, bežci outovaný

Popis situácie: Pálkar odpálil na poliara v zadnom poli (8) ktorý prihodil ploiarovy (4). Ten vyoutoval oboch bežcov v poli. Pálkar ostáva na pálke.

Zral /	0:0	Points:[0, 0, 0, 0, 0, 0, 0]	Total:0
Mant		Points:[0, 0, 0, 0, 0, 0, 0]	Total:0
Record: []			

ERROR	OUT
Sveter	END
7	4,6

SO	BB
→	↘
↗	↘

7-E6	
8-4	
8-4	9-E4

Obrázok 12. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stlačte tlačidlo [8]
2. Stlačte tlačidlo [5], týmto ste zaznačili pohyb lopty.
- 3-4. Označte bežca 1 a a stlačte tlačidlo [OUT].
- 5-6. Označte bežca 2 a stlačte tlačidlo [OUT].
7. Ukončíte zápis stlačením tlačidla [END].

5.7.9 Prvá meta obsadená, páľkar SH

Popis situácie: Páľkar trafil ulievku. Loptu získava vnútorný poliar (5), ktorý prihadzuje na prvú méty poliarovy (3), ktorý outuje páľkara. Bežec z prvej méty získava druhú méty.

Zral /	0:0	Points:[0, 0, 0, 0, 0, 0, 0]	Total:0
Mant		Points:[0, 0, 0, 0, 0, 0, 0]	Total:0

Record: [SH, 5, 3, R1->B2]

SH5-3	8-E4
	SH5-3

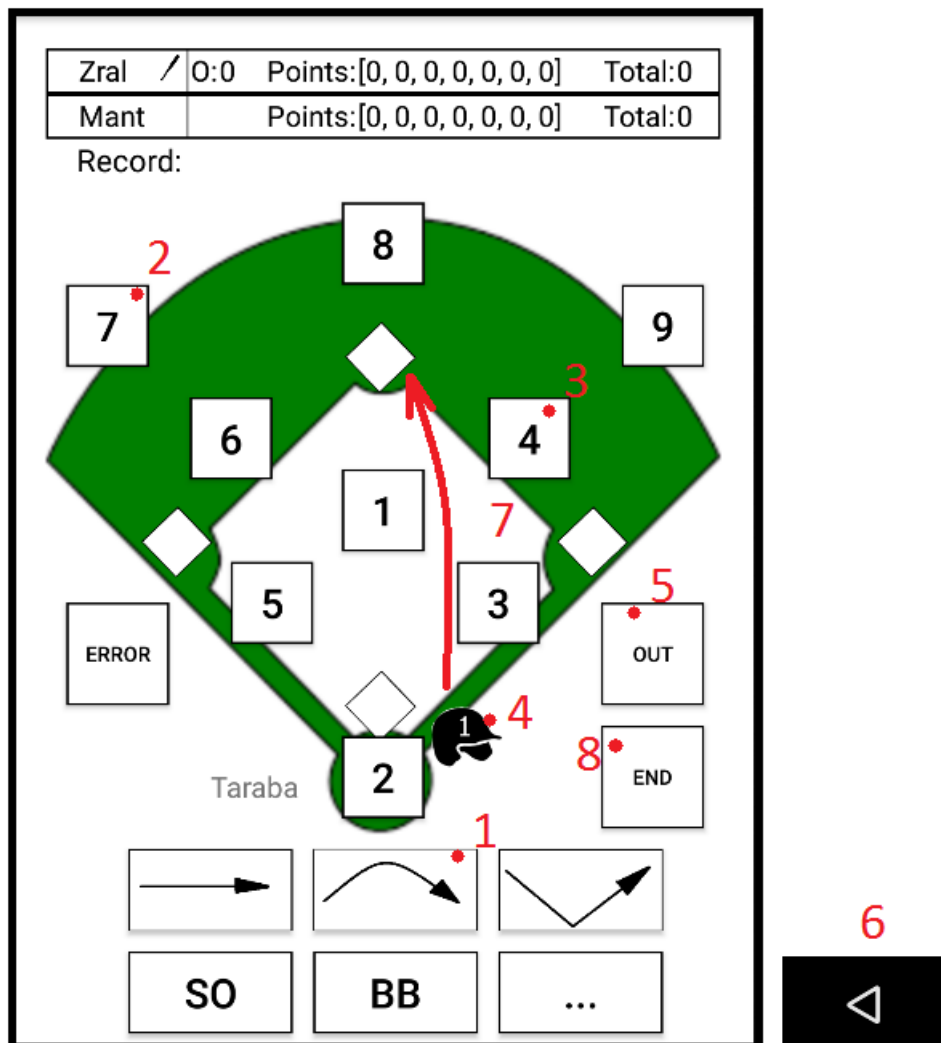
Obrázok 13. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Z menu vyberte možnosť SH.
2. Stlačte tlačidlo [5].
3. Stlačte tlačidlo [3], týmto ste označili pohyb lopty.
4. Presunže bežca z prvej méty na druhú.
5. Kliknutím označte páľkara.
6. Stlačte tlačidlo [OUT].
7. Ukončíte zápis stlačením tlačidla [END].

5.7.10 Ukážka opravy zápisu

Popis situácie: Vysoký odpal do zadného ľavého poľa. Poliar (7) získava loptu a prihadzuje poliarovy (4). Užívateľ chybne stlačí tlačidlo Out, avšak bežec dobehol na druhú métu.



Obrázok 14. Priebeh zápisu hernej situácie.

Postup zápisu:

1. Stačte tlačidlo vysokého odpalu.
- 2 - 3. Stlačte tlačidlo [7]. Stlačte tlačidlo [5], týmto ste zaznačili pohyb lopty.
- 4 - 5. Označte pálkara a stlačte tlačidlo [OUT].
6. Stlačte systémové tlačidlo „Back“, týmto vrátite posledný zapísaný krok.
7. Presunťe bežca na druhú métu.
8. Ukončíte zápis stlačením tlačidla [END].