VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

IMPLEMENTACE NÁSTROJE PRO PROJEKTOVÝ MANAGEMENT NA PLATFORMĚ ANDROID

IMPLEMENTATION OF PROJECT MANAGEMENT TOOL ON ANDROID PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vojtěch Šimša**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Lenka Smolíková, Ph.D.**
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Šimša Vojtěch, Bc.

Informační management (6209T015)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Implementace nástroje pro projektový management na platformě Android

v anglickém jazyce:

Implementation of Project Management Tool on Android Platform

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Návrh řešení a přínos návrhů řešení
Závěr
Seznam použité literatury

Seznam odborné literatury:

DOLEŽAL, J. a kol. Projektový management podle IPMA. 1. vyd. Praha: Grada Publishing, 2009. 512 s. ISBN 978-80-247-2848-3.
FIALA, P. Řízení projektů. 2. vyd. přepr. VŠE v Praze: Nakladatelství Oeconomica, 2008. 186 s. ISBN 978-80-245-1413-0.
FOTR, J. a I. SOUČEK. Investiční rozhodování a řízení projektů. 1. vyd. Praha: Grada Publishing, 2010. 416 s. ISBN 978-80-247-3293-0.
ROSENAU, M. Řízení projektů. 3. vyd. Brno: Computer Press, 2007. 344 s.ISBN 978-80-251-1506-0.
SVOZILOVÁ, A. Projektový management. 1. vyd. Praha: Grada Publishing, 2006. 356 s.ISBN 80-247-1501-5.

Vedoucí diplomové práce: Ing. Lenka Smolíková, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc.	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 29.2.2016

Abstrakt

Diplomová práce se zabývá návrhem a vývojem aplikace pro operační systém Android, poskytující podporu pro projektový management. K realizaci je využito nástrojů a metod pro vývoj mobilních aplikací. Implementace proběhla v programovacím jazyce Java s využitím příslušných knihoven.

Abstract

This diploma thesis is focused on mobile application design and development under the Android platform. It covers the design and implementation of project management support tool to be used on Android devices. This goal is achieved using tools and methodics of mobile application development. The program is implemented in Java, using required libraries.

Klíčová slova

Android, projektový management, mobilní aplikace, Sugar ORM, MPAndroidChart

Key words

Android, project management, mobile application, Sugar ORM, MPAndroidChart

Bibliografická citace práce

ŠIMŠA, V. *Implementace nástroje pro projektový management na platformě Android.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 88 s. Vedoucí diplomové práce Ing. Lenka Smolíková, Ph.D..

Čestné prohlášení:

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 23. května 2016 …….

Bc. Vojtěch Šimša

Poděkování

Vedení Základní školy a Mateřské školy Brno, Merhautova 37 za podporu a tamnímu pedagogickému sboru za vstřícný přístup.

Obsah

Úvod

Moderní technologie v dnešní době pronikají do všech odvětví lidské činnosti. Výjimkou není ani projektový management, kde podpora softwarových nástrojů už dnes poskytuje značnou výhodu a zjednodušení práce. I přes rapidní rozvoj mobilních zařízení však je velmi málo nástrojů, které by podporovaly projektový management ve všech fázích projektu a které by zároveň bylo možné využít na přenosných zařízeních, bez nutnosti přístupu k PC.

Cílem této práce je navrhnout a realizovat aplikaci pro podporu projektového managementu na platformě Android, podle (17) v současné době nejrozšířenější mobilní platformě vůbec. Má představovat nástroj zejména pro podporu plánování a realizace projektu a přinést tak další možnosti mobility pro projektové manažery.

Práce vznikala ve spolupráci se školou Základní škola a Mateřská škola Brno, Merhautova 37, příspěvková organizace, přesněji s místním pedagogickým sborem a projektovým vedením. Při návrhu konkrétní funkcionality tak bylo vycházeno především z požadavků této organizace, nicméně aplikace byla navržena pro použití projektovými manažery v libovolném oboru.

V první části jsou vypsány metody a postupy, využité v této práci. Dalšími kapitolami jsou pak teoretická východiska, zahrnující základní informace nutné pro vývoj aplikací pro platformu Android a teoretický základ podporovaných metod projektového managementu, analýza současného stavu, zahrnující analýzu ZŠ Merhautova metodami kritické analýzy a vlastnosti existujících systémů, a popis návrhu a implementace aplikace. V této poslední kapitole je kromě samotného návrhu a popisu implementace obsaženo i ekonomické zhodnocení celého projektu.

1 Cíle práce, metody a postupy zpracování

Cílem této práce je navrhnout a realizovat aplikaci pro podporu projektového managementu, efektivně využitelnou na přenosných zařízeních. Tato aplikace bude obsahovat podporu pro základní metody projektového managementu, dále pak umožní sledovat stav projektu. Součástí práce bude i analýza současného stavu a vybraných dalších aplikací na trhu. Aplikace bude vyvíjena pro platformu Android v jazyce Java s využitím potřebných knihoven.

Použité metody analýzy současného stavu organizace, tedy ZŠ Merhautova, jsou standardními metodami kritické analýzy, tedy SLEPT, Porterova analýza, Analýza 3C, 7S, SWOT a analýza trendů ovlivňujících firmu. Většina organizací a institucí používá – respektive vyžaduje – pouze některé z těchto metod, konkrétní požadavky se však liší, z tohoto důvodu byly zpracovány všechny.

Z hlediska návrhu aplikace a její implementace byla použita metoda extrémního programování, podpořená zkušeností autora s vývojem aplikací pro mobilní zařízení. Tato metoda je sice méně formalizovaná než jiné přístupy, formální popis vhodný optimální pro zapsání do diplomové práce tedy chybí, přináší však dlouhodobě dobré výsledky ve vývoji software a je efektivně využitelná v samostatné práci.

2 Teoretická východiska práce

V této kapitole budou rozebrány teoretické základy použité v práci. Základní část tvoří informace o platformě Android a teoretický základ použitých metod projektového managementu, dále budou zmíněny základní metody používané při vývoji programů použitelné v kontextu mobilních aplikací.

2.1 Android

Android je open source[1] platforma známá především z oblasti mobilních zařízení a nositelné elektroniky, ačkoli bývá používán i v jiných oblastech, například chytré domácí spotřebiče (10). Základními vlastnostmi jsou nízké nároky na hardware a přenositelnost mezi velkým množstvím typů zařízení. Kromě samotného operačního systému je v platformě zahrnut také middleware[2] a základní aplikace.

2.1.1 Historie

Původní, nepříliš známá, společnost Android Inc. byla založena v roce 2003 a následně v roce 2005 odkoupena společností Google Inc. Pod novým vlastníkem jeden ze zakladatelů Android Inc., Andy Rubin, vyvinul platformu založenou na linuxovém jádře a v roce 2007 získal několik patentů na poli mobilních technologií. O další vývoj se zasloužilo především konsorcium *OHA – Open Handset Alliance*, založené v listopadu téhož roku pod záštitou Google. Z původních 34 firem se dnes rozrostlo na více než 80. Prvním telefonem s operačním systémem Android byl v roce 2008 HTC Dream, využívající nepojmenovanou verzi Android 1.0. (8)

[1] Open source software je počítačový software s otevřeným zdrojovým kódem. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost – licenci software, která umožňuje, při dodržení jistých podmínek (kupříkladu GNU licence), uživatelům zdrojový kód využívat, prohlížet a upravovat

[2] Softwarová vrstva, která zprostředkovává funkce operačního systému aplikacím

Verze	Jméno	API	Aktivní podíl
2.2	Froyo	8	0.1 %
2.3.3 – 2.3.7	Gingerbread	10	2.2 %
4.0.3 – 4.0.4	Ice Cream Sandwich	15	2.0 %
4.1.x	Jelly Bean	16	7.2 %
4.2.x		17	10.0 %
4.3		18	2.9 %
4.4	KitKat	19	32.5 %
5.0	Lollipop	21	16.2 %
5.1		22	19.4 %
6.0	Marshmallow	23	7.5 %

Tabulka 1: Zastoupení verzí systému Android ke 2. květnu 2016. Zdroj: (8)

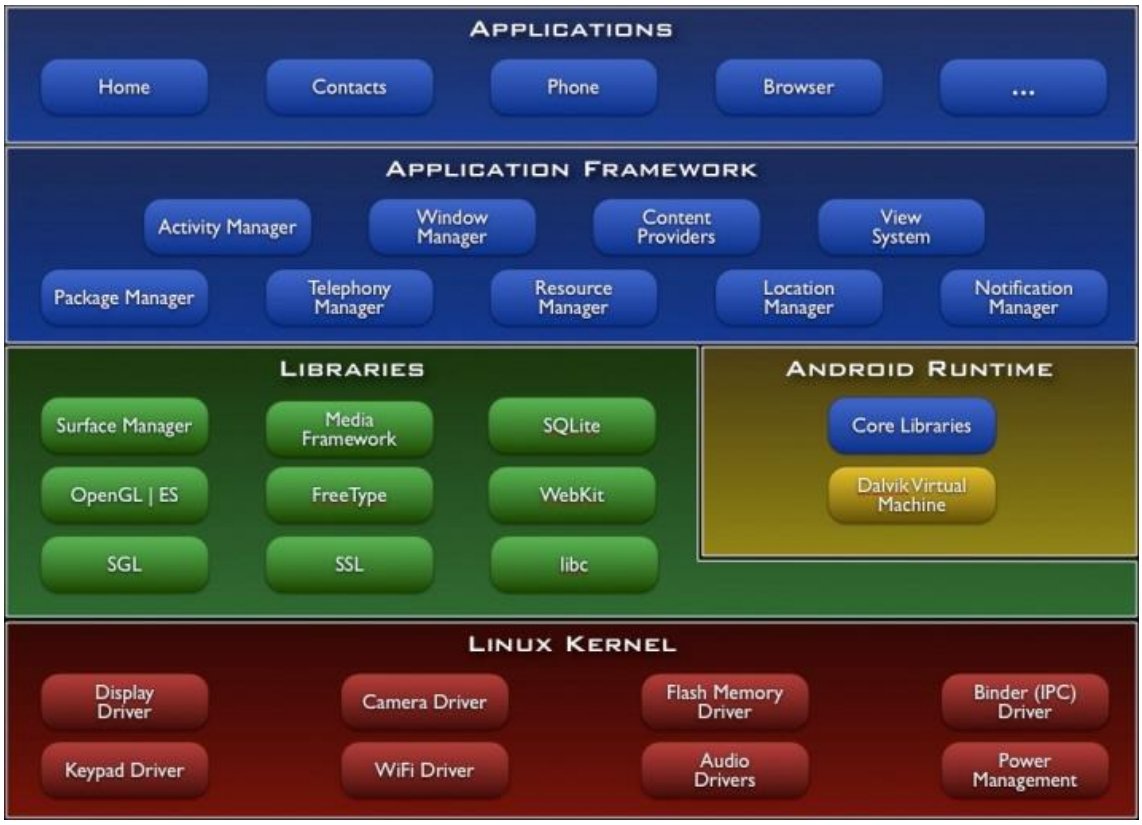
Počínaje verzí 1.5 jsou verze pojmenovány v abecedním pořadí podle v Americe oblíbených zákusků. Každá aktualizace přináší nové API[3] rozšiřující funkcionalitu předešlého. Vzhledem k rychlému vývoji nových verzí toto přináší nutnost hledání kompromisu mezi lepší funkcionalitou a moderními prvky na straně jedné a procentuálním pokrytím používaných přístrojů na straně druhé. V současné době je podle (8) možné podporovat až 99,8 % používaných zařízení pokud by byla podporována verze 2.3.3 Gingerbread a vyšší. Toto pokrytí je zajímavé, nicméně je vykoupeno nutností znovu implementovat značnou část funkcionality, jejíž použití dnes patří ke standardu i k Best Practice[4]. Proto se jako výhodnější jeví použít jako nejstarší podporovanou verzi 4.0.3 Ice Cream Sandwich, která lépe reflektuje generační posun ve vývoji. Konkrétně se jedná o fakt, že 2.3.3 Gingerbread používá API verze 10, zatímco 4.0.3 Ice Cream Sandwich verzi 14. V těchto čtyřech verzích došlo k výraznému posunu v unifikaci nástrojů a přístupů k tvorbě aplikací pro různé úhlopříčky a rozlišení

[3] API - Application Programming Interface - protokol komunikace mezi softwarovými komponentami
[4] Best Practice - překládá se jako nejlepší praxe, osvědčená praxe, je pojmem pro osvědčené postupy, procesy či osvědčené metody řízení, pomocí kterých se ve více organizacích dosáhlo dobrých výsledků a používají se proto jako doporučení pro ostatní

displejů, lze tedy dosáhnout lepší konzistence výsledků. Tímto posunem je pokrytí sníženo na 96,8 % (8), což je stále zcela dostačující úroveň. Dalším faktem hovořícím pro tuto volbu je, že tato tři procenta potenciálního trhu zastupují především velmi stará zařízení, jejichž výpočetní výkon a kapacity pamětí umožňují běh moderních aplikací pouze s velkými obtížemi, zejména v případě aplikací využívajících databáze.

2.1.2 Architektura

Architektura systému Android je rozdělena do pěti vrstev zobrazených na následujícím obrázku.



Obrázek 1: Architektura systému Android. Zdroj: (8)

- **Aplikace:** Nejvyšší vrstva architektury. Je tvořena aplikacemi, které používá uživatel. Lze ji rozšiřovat pomocí obchodu Google Play a jiných zdrojů.

Implicitně pak obsahuje aplikaci fotoaparátu, telefonní seznam, prohlížeč fotek a další.

- **Aplikační Framework:** Vrstva architektury, která zprostředkovává aplikacím přístup k hardwarovým zdrojům zařízení. Cílem je jednoduché znovupoužití existujících komponent, aniž by byla ohrožena bezpečnost jednotlivých aplikací. Základními prvky jsou:
    - poskytovatelé obsahu: prostředek sdílení dat mezi aplikacemi
    - správce zdrojů: přiděluje zdroje jednotlivým aplikacím na základě jejich požadavků a reálných možností systému
    - správce aktivit: řídí životní cyklus aktivity a přepínání mezi nimi
- **Knihovny:** Platforma Android obsahuje velké množství knihoven, přístupných skrze Aplikační Framework. Tímto je zajištěn unifikovaný přístup ke zdrojům na různých zařízeních. Základními knihovnami, podporovanými společností Google Inc. a dostupnými ve všech zařízeních a distribucích jsou například:
    - SQLite: odlehčená databázová knihovna
    - OpenGL: platformně nezávislá knihovna pro 3D grafiku
    - SGL: knihovna základního 2D zobrazení
    - Media Framework: – knihovna pro práci s multimédii

  Další kategorií knihoven jsou knihovny neoficiální nebo oficiálně nepodporované. Tyto bývají dílem nezávislých vývojářů nebo vývojářských studií a, stejně jako oficiální knihovny, významným způsobem rozšiřují dostupné funkce a zobrazení v systému. Příkladem takové knihovny může být například knihovna *Sugar ORM* podrobněji rozebraná v sekci 4.2.3. Tyto knihovny jsou dnes nějakým způsobem částí prakticky všech aplikací, jejich využití však může přinést potíže s kompatibilitou některých zařízení. Tento fakt je způsoben především tím, že nezávislí tvůrci zřídka vlastní časové a finanční prostředky aby knihovnu otestovali na všech existujících variacích systému ještě před vydáním (respektive zpřístupněním, tyto knihovny bývají k dispozici zdarma), knihovny tak podléhají stálému vývoji.
- **Běhové prostředí:** Vrstva obsluhující běh aplikací. Každá aplikace běží ve svém vlastním procesu a s vlastní instancí virtuálního stroje *Dalvik*. Tento

nahrazuje klasický *JVM*[5] známý z PC a je jednou z nejvýznamnějších částí systému Android. Oproti JVM, které jsou založeny na zásobníkové architektuře, používá Dalvik registrovou architekturu a je optimalizován pro minimální paměťovou náročnost. Spouští soubory ve formátu *.dex*, což jsou zdrojové kódy jazyka Java zkompilované příslušným nástrojem do šestnáctibytového instrukčního kódu. Samotný virtuální stroj má přístup k systémovému jádru za účelem správy běžících vláken a nízkoúrovňového přístupu do paměti.

- **Linuxové jádro:** Nejnižší softwarová vrstva založená na monolitickém linuxovém jádru verze 2.6. Obstarává základní systémové služby jako je správa paměti, správa procesů, síťové prostředky a práce s ovladači hardware. Tato vrstva vytváří abstrakci hardware pro zbytek systému a uživatelská aplikace by k ní neměla získat přímý přístup. (4)

[5] Java virtual machine, virtuální stroj Javy

2.2 Vývoj aplikací

Tato kapitola pokrývá teoretický základ pro vývoj aplikací určených pro platformu Android. Pro detailnější vysvětlení všech variant včetně ukázek vzorových kódů lze doporučit zejména (8) nebo (4). Dále zmiňuje základní principy vývoje obecných programů, aplikovatelné na vývoj aplikací pro mobilní zařízení.

2.2.1 Varianty aplikací z pohledu vývoje

V současné době existují tři základní přístupy k tvorbě aplikací pro přenosná zařízení. Každý z nich přináší své výhody a nevýhody, především v oblasti ceny vývoje – a tedy i výsledného produktu, úrovni přístupu ke zdrojům zařízení a nutné specializaci tvůrce.

- **Nativní vývoj** Přístup dobře známý z oblasti osobních počítačů a serverů. Aplikace je spouštěna přímo v operačním systému zařízení. Tento přístup vykazuje nejlepší výsledky co do využití zdrojů zařízení, lze tedy vytvářet relativně složité aplikace spustitelné i na slabších přístrojích. Dále je zaručen přístup programátora ke standardním ovládacím prvkům, lze tedy dosáhnout uživatelského rozhraní, na které jsou zákazníci "zvyklí" v kontextu daného systému. Tímto způsobem vytvořené aplikace jsou v zásadě nepřenositelné, tedy není možné bez zásadních úprav spustit aplikaci určenou například pro Android na zařízení používajícím iOS. Zároveň se obvykle jedná o nejdražší variantu vývoje, zejména v případě kdy je cílem podporovat více platforem – v takovém případě se prakticky jedná o vývoj samostatných aplikací. V tomto případě se také dříve zmíněná výhoda "známých ovládacích prvků" mění v potenciální nevýhodu - ovládání působí rozdílným dojmem na různých platformách, nebo je nutné toto ovládání na některé z platforem od základu přepracovat. (8)

  Výhody:

  - Možnost umístit aplikaci do standardního obchodního kanálu, jako je Google Play
  - Možnost použití standardizovaných ovládacích prvků
  - Aplikace použitelné bez přístupu k internetu
  - Lepší možnosti zabezpečení

Nevýhody:

- Vyšší náklady na vývoj
- Pro efektivní distribuci musí aplikace projít schvalovacím procesem příslušného obchodního kanálu
- Omezení segmentu trhu z důvodu specializace na zvolený operační systém

- **Webový vývoj** Aplikace spouštěné ve webovém prohlížeči. Jejich kód obvykle bývá psán v HTML5, JavaScriptu a CSS. Tyto aplikace je typicky možné spustit na libovolném zařízení, v některých případech i na klasickém PC. Tímto je umožněno vytvářet a spravovat pouze jednu verzi aplikace a pokrýt široký segment trhu. Tyto aplikace navíc bývá možné spustit i bez předchozího schválení oficiálním distribučním kanálem, je tedy jednodušší jejich dodání jedinému zákazníkovi. Zcela zásadní výhodou pak je odstranění omezení výpočetního výkonu přenosných zařízení, protože výpočetně náročné operace mohou probíhat na straně serveru a aplikace samotná může sloužit pouze jako terminál. Nevýhodou je nutnost rozhodnout jaké ovládací prvky bude aplikace podporovat – typickým příkladem je kolize Android proti iOS, kde Android umisťuje ovládací panel na horní okraj pracovní plochy, zatímco iOS na spodní. Vývojář se tak musí rozhodnout, kterou ze skupin bude preferovat a počítat s možností odmítnutí od skupiny druhé. Dalším problémem je ztížený, v některých případech až nemožný, přístup k funkcím zařízení a nutnost připojení k internetu. (8)

  Výhody:

  - Použití nezávisle na platformě
  - Nižší náklady na vývoj
  - Rychlejší uvedení na trh
  - Potenciál pro využití výkonu serveru, na kterém aplikace běží

Nevýhody:

- Nemožnost použití jednotných obchodních kanálů
- Nutnost optimalizace pro různé webové prohlížeče
- Horší přístup ke zdrojům zařízení
- Obtížnější zabezpečení
- Nutnost webového připojení

- **Hybridní vývoj** Kombinace předchozích přístupů. Aplikace je spouštěna přímo v přístroji, čímž je dosaženo lepšího přístupu ke zdrojům, je však stále koncipována podobně jako webová stránka. Takto je možné vytvořit jednotný obsah pro všechny systémy a zapouzdřit jej do tzv. *wrapperu*[6]. Tento nástroj umožňuje spustit aplikaci vytvořenou pomocí HTML5, CSS a JavaScript jako nativní, vytváří tak mezivrstvu mezi aplikací a zařízením obdobně jako webový prohlížeč v případě webových aplikací. Tento přístup se může zdát jako optimální, naráží však na vlastní problémy. Hybridně vytvořené aplikace je sice možné umístit na oficiální distribuční kanály, nicméně Apple málokdy schválí jejich uveřejnění. Google má v tomto přívětivější přístup, uživatelé Androidu však statisticky hybridní aplikace příliš dobře nepřijímají (8).

  Výhody:

  - Kratší čas a nižší cena vývoje než plně nativní aplikace
  - Možnost použít standardizované ovládací prvky
  - Jednodušší přenos mezi platformami

  Nevýhody:

  - Nutnost udržovat samostatný zdrojový kód pro různé platformy (byť v menším rozsahu než u nativních aplikací)
  - Aplikace musí projít schvalovacím procesem obchodních kanálů

Pokud by bylo možné předpokládat stoprocentní pokrytí přístupnou sítí internet, byl by webový vývoj pravděpodobně nejužívanější variantou, zejména z důvodu nižších

[6] Česky asi nejlépe přeložitelné jako zapouzdřovač, mezi nejznámější patří například IBM Worklight (9)

nákladů. V současné době však toto stále nelze považovat za samozřejmost. Výsledná volba tak záleží především na určení aplikace a požadavcích konkrétního zákazníka nebo volbě obchodního kanálu.

Na území ČR dosud není plné pokrytí veřejnou sítí WiFi, a ačkoli lze očekávat, že v prostorách, kde bude aplikace vzniklá v rámci této práce používána, bude dostupná lokální síť, nelze toto považovat za automatické. Mobilní připojení je pak velmi závislé na operátorovi, a to jak co do rychlosti tak co do ceny. V kombinaci s problémy zabezpečení přenášených informací byl proto webový vývoj zavržen. Varianta hybridního vývoje by sice byla patrně rychlejší, nicméně vzhledem k nutnosti přístupu k databázi zařízení a celkovému snížení potenciálního výkonu zařízení byla také opuštěna. Pro účely této práce proto bylo rozhodnuto o použití nativního vývoje a další varianty nebudou hlouběji rozebírány.

2.2.2 Android SDK

Android je otevřená platforma, jejíž úspěch závisí především na ochotě vývojářů a firem vytvářet pro ni aplikace. Za účelem zjednodušení tohoto procesu tak Google vytvořil *Android SDK*, což je zkratka pro Android Software Development Kit. Jedná se o soubor nástrojů potřebných pro vývoj a testování aplikací použitelný na běžných PC. Součástí je možnost vytváření virtuálních zařízení s různými verzemi API, je tedy možné testovat všechny podporované verze systému bez nutnosti fyzického přístupu k daným zařízením. Nejširší podpora je poskytována pro specializované vývojové prostředí Android Studio (11), ve formě modulu Android Development Tools je pak dostupná pro většinu vývojových prostředí. Základním jazykem pro nativní vývoj aplikací na platformu Android je Java, ačkoli s využitím *AndroidNDK* [7] je možné použít i C/C++ (4).

[7] Sada nástrojů, umožňující implementaci částí aplikace v jazycích nižší úrovně. Typické využití je u činností náročných na výpočetní výkon, jako jsou herní enginy, zpracování signálů v reálném čase nebo fyzikální simulace

2.2.3 Struktura aplikace

Z pohledu zdrojových souborů je aplikace pro Android rozdělena do základní struktury adresářů:

- **assets/** - statické soubory nutné pro běh aplikace, jako je video, zvuk a podobně. Jedná se o prvky, které nejsou přímo obsaženy v grafickém rozhraní aplikace
- **gen/** - soubory vygenerované při překladu aplikace
- **libs/** - předem nezávisle zkompilované knihovny, používané v rámci aplikace
- **res/** - zdroje pro grafické rozhraní, jako jsou ikony, rozvržení prvků pracovní plochy a podobně
- **src/** - zdrojové kódy aplikace

Na nejvyšší úrovni jsou také dva soubory, obsahující informace o samotné aplikaci:

- **AndroidManifest.xml** – popis aplikace a jejích komponent, obsahuje název, slovní popis pro zveřejnění, podporované verze Androidu a další
- **build.xml** – pokyny pro překlad a instalaci aplikace, možné je také jeho rozložení do souborů *build.gradle* pro jednotlivé moduly - v takovém případě je automaticky syntetizován při překladu aplikace

Tuto strukturu lze podle potřeby rozšiřovat, typické je zejména dělení adresáře *src* podle určení obsažených tříd na managery, třídy aktivit a další. Tato dodatečná struktura už není povinná a slouží především k lepší orientaci ve zdrojových kódech (4).

Z hlediska softwarových komponent je pak rozdělení následující:

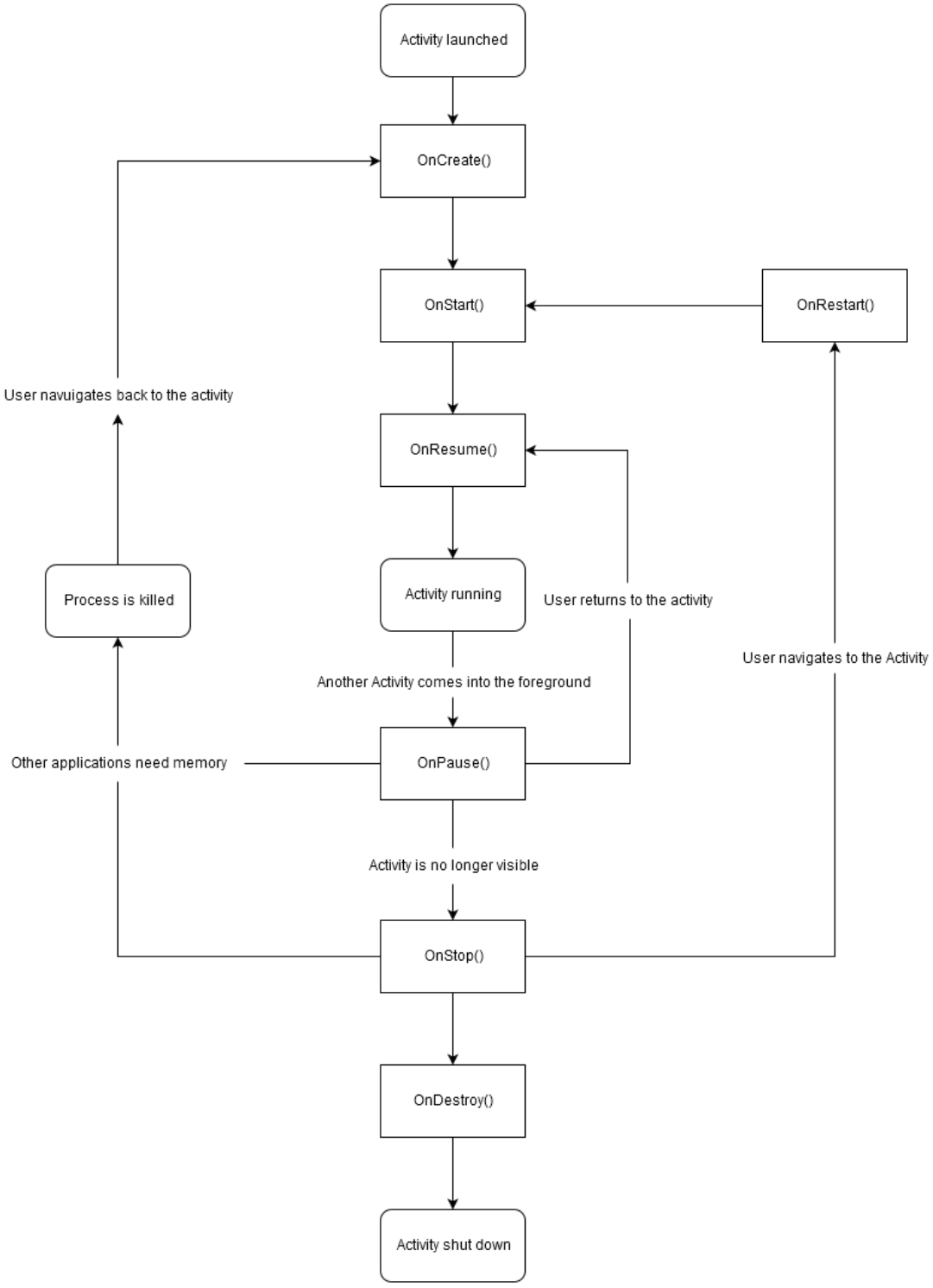
- **Aktivita:** Základní stavební kámen aplikace, řešící logiku celého programu (8). Obdobou by mohlo být okno u aplikace pro PC. Životní cyklus aktivity viz. Obrázek 2. Aktivita se vždy nachází v jednom z následujících stavů:
    - aktivní – aktivita je na popředí a uživatel s ní může interagovat za použití grafického rozhraní
    - pozastavená – aktivita je spuštěna a aktivní, na popředí se však nachází nějaké upozornění. S touto aktivitou není možná interakce, dokud není uvedena zpět na popředí.

    - zastavená – aktivita byla spuštěna, ale na popředí je jiná aktivita. Interakce se zastavenou aktivitou je možná pouze prostřednictvím upozornění
    - mrtvá – aktivita nebyla dosud spuštěna nebo byla násilně ukončena

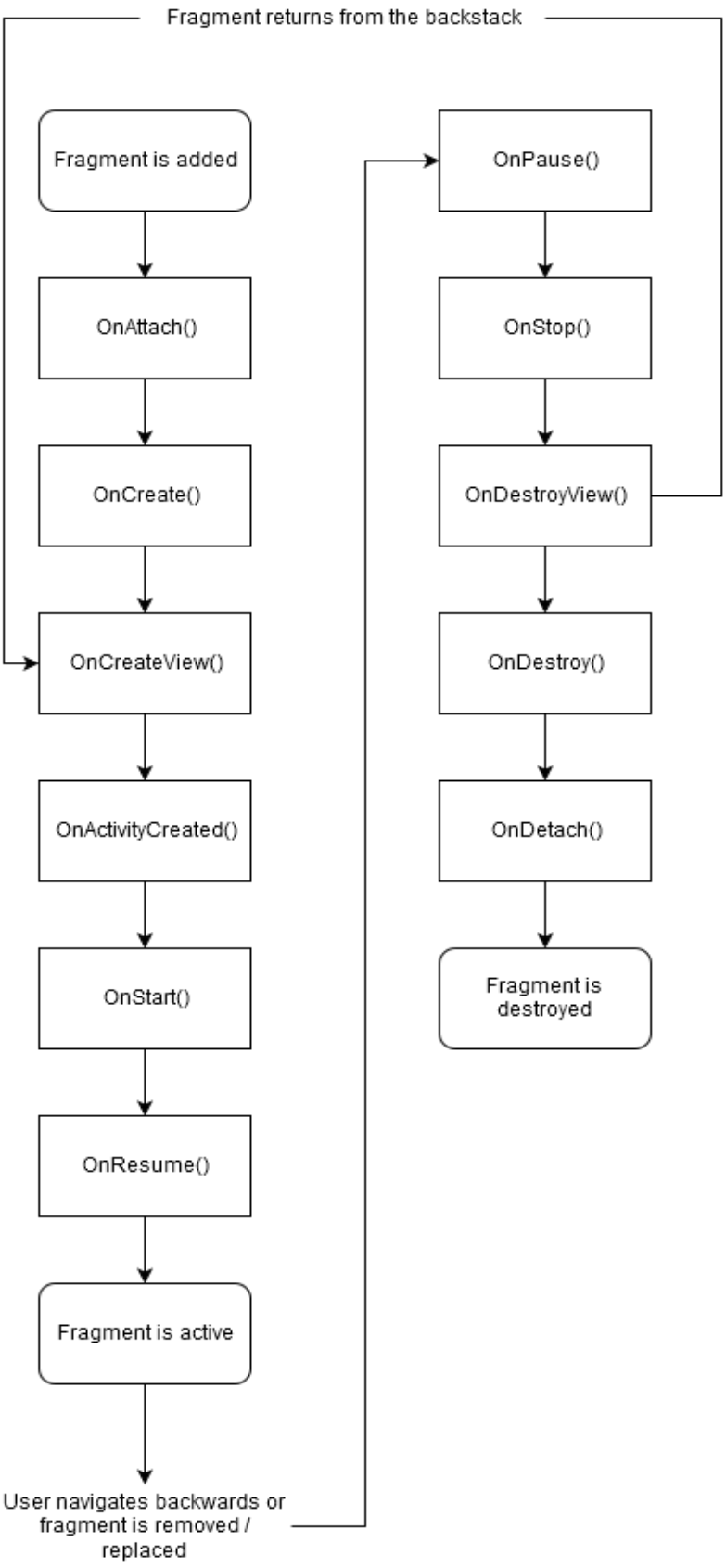
- **Fragmenty:** Komponenty s podobnou funkcionalitou jako Aktivita. Jedná se o zapouzdřený blok s vlastním životním cyklem (Obrázek 3), který je možné vložit do aktivity a následně za běhu aplikace zaměňovat. Jejich použití výrazně zplošťuje strukturu aplikace a umožňuje efektivní přenos základních ovládacích prvků implementovaných v mateřské aktivitě, které jsou společné všem fragmentům v ní obsaženým.
- **Služby:** Komponenty běžící na pozadí nezávisle na aktivitách, zajišťující běh dlouhodobých procesů. Služba má opět svůj vlastní životní cyklus (Obrázek 4). Jako příklad lze uvést přehrávání hudby, které se nezastaví, pokud uživatel spustí jinou aplikaci.
- **Poskytovatelé obsahu:** Komponenty zajišťující přístup k datům. Odstiňují aktivity od přímého přístupu k datům a povolují pouze přístup k informacím, na které má aplikace právo. Tato práva byla až do verze *6.0 Marshmallow* přidělena na základě požadavku při instalaci, od verze *6.0 Marshmallow* výše jsou kontrolována - a přidělována na základě uživatelského zásahu - za běhu aplikace. Tato změna přináší lepší uživatelskou kontrolu nad činností aplikací, zároveň však zapříčinila nekompatibilitu značného množství starších aplikací s novými zařízeními.
- **Broadcast receiver:** Komponenta sloužící k reakci na systémová volání. Mezi tato volání může patřit například informace, že byl vypnut displej, dochází baterie a podobně, stejně jako volání vytvořená používanými aplikacemi.

Komponenty mezi sebou typicky komunikují prostřednictvím asynchronní zprávy nazývané *Intent*. Tato zpráva slouží k aktivaci příslušné komponenty a zároveň předává informace o kontextu, který je vyžadován. Výjimku zde tvoří poskytovatelé obsahu, tyto komponenty mají samostatný systém předávání informací z důvodu komplexnosti požadovaných dat (4).

Za zmínku stojí zejména prvek nazývaný *Fragment*. V tomto případě nelze hovořit o samostatné komponentě v doslovném smyslu slova, protože pro svou funkci vyžaduje zapouzdření v aktivitě (8), jedná se však o velmi významný stavební blok v moderních aplikacích. Jedná se principiálně o aktivitu v aktivitě, s vlastním životním cyklem závislým na životním cyklu rodičovské aktivity. Fragmenty je možné přidávat a odebírat i v rámci běhu aktivity, jedná se tedy o velmi účinný nástroj pro tvorbu uživatelského rozhraní s více panely. Vytváření fragmentů je možné až od API verze 11 (viz sekce 2.1.1).



Obrázek 2: Životní cyklus Aktivity. Zdroj: (8)



Obrázek 3: Životní cyklus Fragmentu. Zdroj: (8)

Obrázek 4: Životní cyklus Služby (Service). Zdroj: (8)

2.2.4 Uložení dat

Platforma Android typicky umožňuje tři formy uložení dat: interní v paměti zařízení, interní na vyměnitelné kartě a vzdálené (4).

Vzdálená úložiště se vyznačují obvykle relativně velkou kapacitou na poměry mobilních zařízení. Může se jednat o specifické servery firem, vzdálená úložiště konkrétních aplikací a tzv. *Cloud*[8]. Ve všech případech je nutné uvažovat zabezpečení zejména přenosu, který se odehrává prostřednictvím implicitně nezabezpečené sítě internet. Z podstaty je pak jasné že pro využití takového úložiště je nutné připojení k internetu, což jej vylučuje z variant pro aplikace určené k použití offline.

Interní vyměnitelná karta se vyznačuje nižším objemem potenciálně uložených dat než vzdálená úložiště, typicky však nabízí rychlejší čtení i zápis a stoprocentní dostupnost. Data na vyměnitelné kartě je možné libovolně číst a upravovat, bez použití šifrování se tedy jedná o umístění nevhodné pro jakékoli citlivé informace. Zásadní nevýhodou je, že od použití vyměnitelných karet se postupně upouští, zejména od nástupu cloudových technologií a služeb. Nelze proto automaticky předpokládat, že karta bude dostupná na všech zařízeních, což limituje potenciální segment trhu pro vyvíjenou aplikaci.

Interní úložiště představuje vnitřní paměť zařízení, která je mimo jiné nutná pro samotný běh systému. Lze proto předpokládat že bude dostupná. Ze všech základních variant úložišť má nejmenší kapacitu a nejvyšší cenu za megabyte, v současné době je však dostatečná pro většinu aplikací kromě velkých databází. Systém Android navíc v této paměti izoluje data aplikací od ostatních tak, že aby byla přístupná je nutné přístup explicitně povolit, do určité míry tedy lze považovat toto úložiště za bezpečné[9]. Toto zabezpečení lze dále podpořit šifrováním, ať už samotných dat v rámci aplikace nebo celého úložiště v rámci systému.

[8] Metoda přístupu k využití výpočetní techniky, nejen uložení dat. Založena na sdílení výpočetních prostředků a jejich využívání jako služby.

[9] Výjimkou jsou zařízení, na kterých byl proveden takzvaný root. Tímto oficiálně nepodporovaným zákrokem jsou vybraným aplikacím zpřístupněny veškeré zdroje zařízení, včetně zdrojů běžně chráněných. Tento zákrok však používají téměř výhradně zkušení uživatelé, kteří vědí, jak svá zařízení chránit jiným způsobem.

V současné době většina zařízení podporuje funkci *USB OTG*, která je nutná k připojení USB zařízení jako jsou flash disky, externí disky a další (8). Tyto však nelze uvažovat, protože pro jejich připojení je nutný adaptér vstupu - žádné Android zařízení, možná s výjimkou některých Android TV, nemá USB port v plné velikosti. Tento problém možná bude odstraněn nástupem USB typu C, stále však zůstává problém s vyčnívajícím, nepříliš pevným a pevně uchyceným tělem samotného paměťového zařízení. Dalším problémem pak typicky bývá potřeba přídavného napájení, ať už prostřednictvím síťového adaptéru nebo powerbanky[10]. V roce 2012 pak Samsung představil zařízení s kódovým označením *SE-218BB*, které umožňuje přehrávat CD a DVD na platformě Android, tento koncept se však neuchytil a v současné době nejsou obdobná zařízení v nabídce. Z těchto důvodů nelze o podobných technologiích uvažovat jako o variantě úložiště potřebného pro běh aplikace.

[10] Externí baterie připojitelná přímo k telefonu / tabletu nebo k napájenému zařízení

2.2.5 Specifika vývoje aplikací pro použití v korporacích

S obecným rozšířením platformy Android došlo k rozšíření i v rámci pracovního prostředí. Toto rozšíření a nerespektování bezpečnostních zásad mělo za následek úniky citlivých informací a obchodních tajemství. Jako reakce vznikl program *Android for Work* (8), který má vývojáře navést k tvorbě pracovně využitelných, bezpečných aplikací. Plná podpora programu je od verze 5.0, API úrovně 21, což je bohužel v současné době pouze zhruba jedna třetina používaných zařízení (viz sekce 2.1.1). I tak je však vhodné co nejvíce z navrhované funkcionality implementovat, jednak z důvodu budoucího použití, ale i z důvodu uvažovaného zpětného rozšíření podpory až do verze 4.x Ice Cream Sandwich. Většina požadavků je již delší dobou součástí Best Practice, některé však získaly na důležitosti nad rámec tohoto pojmu.

Základní změnou, kterou API úrovně 21 přineslo, je možnost vytváření takzvaných *Managed Profiles*, tedy uživatelských profilů, které podléhají vyšší správě. Tento přístup je z PC známý už delší dobu a programů pro správu těchto profilů existuje velké množství, příkladem budiž *Active Directory* (12). Správce může aplikacím přidělovat – nebo odebírat – oprávnění v závislosti na očekávané funkci aplikací a pozici uživatele v korporátní struktuře. Tato změna přináší omezení pro samotné aplikace:

- Zprávy typu Intent implicitně nemohou být posílány z jednoho profilu (např. pracovní) do druhého (např. osobní) (8). Toto omezení může mít za následek omezení komunikace mezi aplikacemi, je tedy nutné jej vzít v úvahu především při odesílání dat. V kombinaci s následným omezením může toto při nevhodném přístupu vést k nestabilitě aplikací.
- Administrátor může omezit aplikace dostupné na spravovaném profilu. Může se tak stát, že aplikace jako třeba emailový klient, standardní součásti systému dosud považované za samozřejmé, budou nedostupné. Je tedy nutné zvážit, jaké prostředky bude aplikace skutečně potřebovat a zda by nebylo možné při zachování funkcionality využít jiné.
- Spravované a nespravované profily mají oddělená úložiště v interní paměti, může se tak stát že některé soubory nebudou dostupné. Požadavek na otevření

nenalezeného souboru vede k pádu aplikace, je proto vhodné nepoužívat v zprávách typu Intent pevné adresy.

Dalším bodem je reakce aplikace na samotná omezení, která administrátor aplikoval. Zde je především nutné v popisu aplikace jasně deklarovat, které funkce aplikace vyžaduje pro správný chod, aby se administrátor měl podle čeho orientovat. Koncept omezení je pro aplikace přístupný již delší dobu, dosud je však bylo možné měnit pouze v nastavení v rámci samotné aplikace (8). Příkladem může být výběr, zda aplikace může stahovat aktualizace s použitím mobilní sítě nebo pouze přes WiFi. Nyní je však možné, že se omezení změní i činností jiné aplikace. V takovém případě není ovládaná aplikace automaticky informována, je tedy nutné pravidelně kontrolovat případné změny. Kontrola by měla proběhnout v následujících momentech života aplikace (8):

- onResume() - ve chvíli kdy je aplikace spuštěna nebo obnovena. Reakce v tomto bodě životního cyklu lze, za použití dobře použité dědičnosti tříd, znovupoužít v různých aplikacích s podobným kontextem.
- V případě příchozí notifikace o změně, tuto funkcionalitu je nutné implementovat specificky pro každou aplikaci.

2.3 Podporované metody projektového managementu

V této kapitole jsou popsány teoretické základy metod, pro které aplikace poskytuje podporu. Cílem není kompletní popis jednotlivých metod ani návod pro jejich praktickou aplikaci, pouze sumarizace principu s důrazem na prvky, které je možné podporovat vhodným softwarovým nástrojem. Tyto konkrétní metody byly zvoleny po konzultaci se zadavatelem práce, další inspirací byla desktopová verze programu 2-plan (viz sekce 3.3). Následující odstavec, včetně jeho struktury, je vložen z důvodu licenčního ujednání pro použití metody RIPRAN.

Informace o metodě RIPRAN

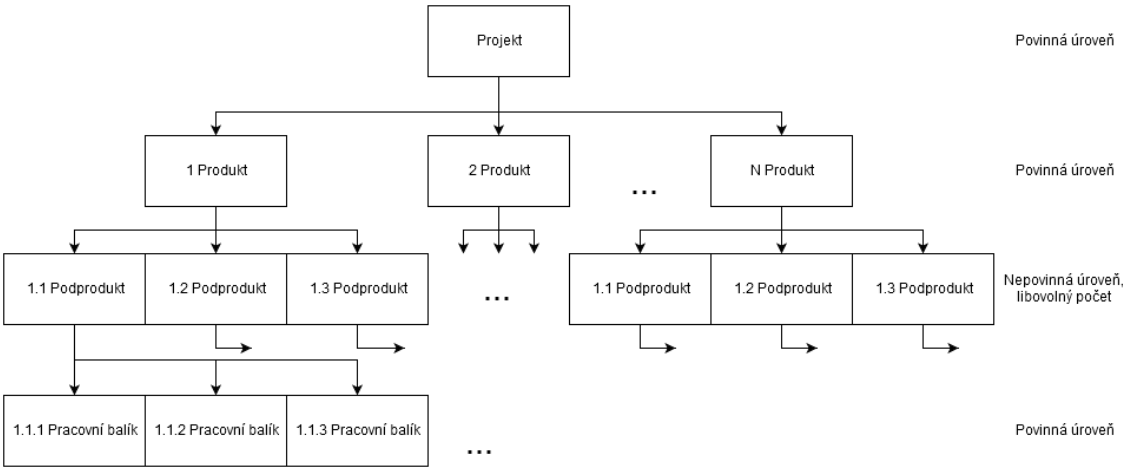
Metoda RIPRAN (RIsk PRoject ANalysis) je určena zejména pro analýzu projektových rizik. Autorem metody je B. Lacko. Metoda vznikla původně pro analýzu rizik automatizačních projektů v rámci výzkumného záměru na VUT v Brně. Praxe ukázala, že po určitých úpravách je metodu možno aplikovat pro analýzu rizik širokého spektra různých projektů a v určitých případech i pro analýzu jiných druhů rizik než jsou projektová rizika. RIPRAN™ je ochranná známka, registrovaná autorem v Úřadu průmyslového vlastnictví Praha pod reg. 283536. (18)

2.3.1 Work Breakdown Structure

Work Breakdown Structure, často zkracováno (a v textu dále označováno) jako WBS (1), nemá jednoznačný český ekvivalent. Jedná se o dokument obsahující hierarchický rozklad cíle projektu, respektive předmětu plnění projektu, na jednotlivé výsledky, respektive produkty, až na jednotlivé pracovní balíky. Zároveň tímto názvem bývá označena metoda tvorby tohoto dokumentu (3). Tato metoda svým využitím spadá do fáze plánování projektu, v dalších fázích pak její výstup může sloužit jako reference. Není zde řešena závislost mezi elementy, pouze jejich suma, která musí být dostatečná pro naplnění všech cílů projektu. Rozklad úkolů je specifický pro každý projekt a konkrétní projektový tým. Základní pravidla, která jsou předpokládána pro správnou funkci, jsou:

- 100 % - stanovuje, že WBS obsahuje 100 % definovaného cíle a zachycuje všechny dodávky (vnější, vnitřní, prozatímní) z hlediska práce, která musí být vykonána
- Nepřekrývání obsahu elementů – doplňuje předchozí pravidlo. Podstatou WBS je, že nesmí docházet k překryvu mezi jednotlivými elementy.
- Plánování obsahu, ne činnosti – WBS neřeší, jak nebo kdy bude výsledku dosaženo, pouze výsledek samotný
- Úroveň detailu – metoda sama neurčuje, do jakých podrobností mají jednotlivé segmenty být rozloženy. Obecně platná doporučení však říkají, že žádný element by neměl být delší než 80 hodin a delší než jedno dohodnuté vykazovací období.

WBS se obvykle sestavuje na základě brainstormingu, provedeného samotnými členy projektového týmu (3).



Obrázek 5: Grafické znázornění obecné struktury WBS. Zdroj: (3)

Konkrétními hodnotami by mohly být například následující:

Projekt	1 Produkt	2 Produkt	…
Pořádání konference	Odborný program	Propagace	…

Tabulka 2: Ukázka přiřazení objektů WBS, část 1. Zdroj: (3)

Následně je možné jednotlivé produkty detailněji rozpracovat:

2.1 Podprodukt	2.2 Podprodukt	2.3 Podprodukt	2.4 Podprodukt	…
Tvorba a rozeslání pozvánek	Zajištění WWW prezentace	Zajištění registračního systému	Tvorba a distribuce propagační publikace	…

Tabulka 3: Ukázka přiřazení objektů WBS, část 2. Zdroj: (3)

Tato úroveň již odpovídá pracovním balíkům, je tedy nejnižší úrovní WBS podle IPMA (3). Další rozpad už by znamenal seznam jednotlivých činností, který je předmětem metody kritické cesty, viz 2.3.2.

2.3.2 Kritická cesta

Metoda kritické cesty, anglicky Critical Path Method, zkráceně CPM, je matematický algoritmus plánování průběhu činností projektu. Základem algoritmu je sestrojení modelu ve formě orientovaného grafu, obsahujícího (3):

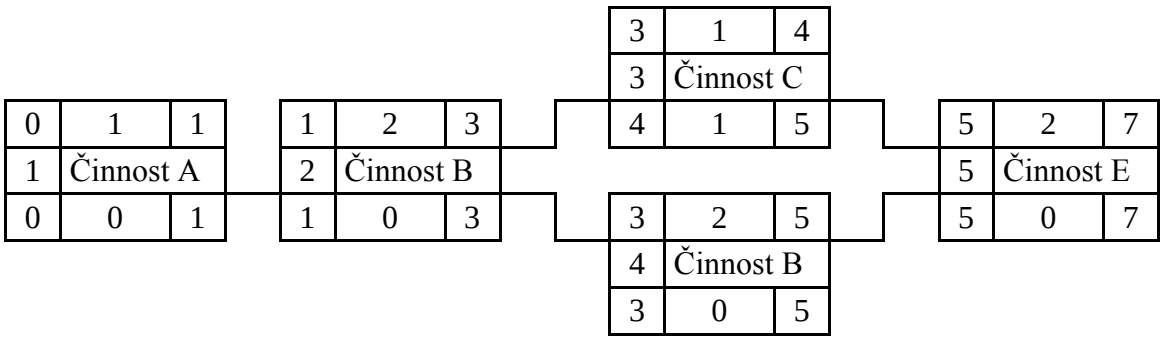
- všechny aktivity nutné pro úspěšné dokončení projektu
- čas nutný pro jednotlivé fáze
- návaznosti mezi jednotlivými aktivitami

V tomto modelu je následně vypočítána nejdelší cesta od počátku ke konci projektu, nazývaná kritickou cestou. Zpoždění libovolné aktivity na kritické cestě by vedlo ke zpoždění celého projektu. Aktivity neležící na kritické cestě mají toleranci zpoždění, danou nejdřívějším a nejpozdějším možným začátkem. Pokud aktivita začne v příslušném časovém rozmezí, nemusí dojít ke zpoždění celého projektu. Konkrétní hodnoty těchto mezí jsou také předmětem metody kritické cesty.

Vzhledem k faktu, že pro výpočet trvání jsou používány obvykle odhady, je vhodné model průběžně upravovat aby odpovídal realitě (3). Kritická cesta se tak může vlivem zpoždění měnit. Algoritmus nalézá uplatnění zejména ve fázích plánování, řízení projektu a vyhodnocení. Ve fázi plánování slouží k odhadu doby nutné k realizaci a v případě pevně stanovených termínů i k odhadu realizovatelnosti projektu, ve fázi

řízení projektu slouží k efektivnějšímu řízení času a celkově k lepšímu přehledu o stavu projektu (3). Ve fázi vyhodnocení pak může být použit k určení příčiny selhání.

Původní verze algoritmu byla aplikována bez pomoci počítačů a zahrnovala vazby mezi aktivitami pouze z pohledu souslednosti, tedy pro začátek aktivity B je nutné, aby byla dokončena aktivita A. Tento algoritmus bývá v současné době doplňován o zohlednění zdrojů nutných k průběhu aktivit (3).



Obrázek 6: Ukázka jednoduchého síťového grafu kritické cesty.

Zdroj: vlastní zpracování

Výpočet kritické cesty běžně sestává z pěti fází – přípravy, stanovení logických vazeb, samotného výpočtu, kontroly a vyznačení kritické cesty.

Ve fázi přípravy je nutné připravit seznam všech činností projektu, respektive jeho části, která je aktuálně řešena, např. mezi dvěma milníky. Tyto jsou označeny jednoznačnými identifikátory, při ručním zpracování bývají volena písmena, v praxi obvykle identifikátor ze softwarové podpory a stanovit délky jejich trvání.

V druhém kroku je nutné stanovit logické vazby mezi jednotlivými činnostmi. V praxi se osvědčil manuální způsob s využitím nalepovacích lístečků, osobní zkušenosti a znalosti jsou v této fázi nezastupitelné. Z tohoto důvodu není dost dobře možné tuto fázi automatizovat.

Třetím krokem je samotný výpočet. Vstupem je v prvních dvou krocích vytvořený předpřipravený graf. Ruční výpočet probíhá ve třech průchodech.

Možný začátek	Délka trvání	Možný konec
Identifikátor	Textový popis	
Nutný začátek	Rezerva celková / Rezerva volná	Nutný konec

Obrázek 7: Uzel kritické cesty. Zdroj: (1)

První průchod, zleva doprava, zaplňuje pole "možný začátek" a "možný konec," kde možný začátek se rovná maximu z množiny možných konců předchozích uzlů a možný konec se rovná možnému začátku plus doba trvání činnosti reprezentované daným uzlem.

Před začátkem druhého průchodu je hodnota možného konce koncového uzlu grafu přepsána do pole "nutný konec." V tomto průchodu jsou stanoveny hodnoty "nutný začátek" a "nutný konec," kde nutný začátek je roven nutnému konci příslušného uzlu minus doba trvání. Nutný konec je pak určen jako minimum z množiny nutných začátků následujících uzlů.

Ve třetím průchodu jsou vypočteny volné a celkové rezervy. Celková rezerva se počítá na každém uzlu samostatně jako rozdíl mezi nutným koncem a možným koncem daného uzlu. Volná rezerva je pak stanovena jako minimum z možných začátků následujících uzlů minus možný konec počítaného uzlu. Logicky pro koncový uzel jsou obě rezervy rovny nule.

Čtvrtou fází zpracování je kontrola. Jako nutné minimum je stanoveno, že (3):

- v žádné části grafu se nesmí vyskytovat záporné hodnoty
- pro rezervy každého uzlu platí, že rezerva celková je větší nebo rovna rezervě volné, z čehož vyplývá, že kde je rezerva celková rovna nule, je i rezerva volná nulová
- možný a nutný začátek počátečního uzlu se rovnají

V pátém kroku je pak již jen vyznačena kritická cesta. Toto vyznačení bývá provedeno barevným odlišením daných prvků, typicky červenou barvou. Takto vyznačeny jsou všechny uzly s nulovými rezervami a vazby mezi nimi. Následně proběhne ještě kontrola, jestli takto získaná kritická cesta vede bez přerušení od počátečního uzlu ke koncovému. Kritická cesta se může větvit, v takovém případě jsou označeny všechny větve.

2.3.3 RIPRAN

Metoda RIPRAN je empirická metoda pro analýzu rizik projektů. Jejím autorem je B. Lacko a vznikla v rámci výzkumného záměru na VUT v Brně. Akceptuje filosofii jakosti, a proto obsahuje činnosti zajišťující jakost procesu analýzy rizika, vyžadované normou ISO 10 006. Je také navržena tak, že respektuje zásady pro Risk Project Management popsané v materiálech PMI a IPMA (18).

Prakticky dělí proces analýzy rizik na 5 fází, koncipovaných jako navazující procesy:

- příprava analýzy rizika
- identifikace rizika
- kvantifikace rizika
- odezva na riziko
- celkové zhodnocení rizika

Ve fázi přípravy je cílem formalizace podkladů pro analýzu rizik. Vstupem je popis metody, formuláře metody a pokyny a informace vázané k analýze rizik. Výstupem je časový plán analýzy rizik, sestavení týmu pro analýzu rizik a rozhodnutí o použitých stupnicích, kontrolních seznamech a podobně.

Fáze identifikace rizika má za cíl nalezení hrozeb a scénářů. Vstupem je především popis projektu, dále pak historická data o minulých projektech, prognózy vlivů a zkušenosti. Výstupem je seznam dvojic hrozba – scénář s případnými komentáři. Tato fáze je kritická pro správné vyhodnocení rizik, závisí však především na lidském faktoru týmu, který není možné zastoupit.

Fáze kvantifikace rizika slouží k ohodnocení pravděpodobnosti dříve formulovaných scénářů, velikosti škod a vyhodnocení míry rizika. Vstupy jsou tedy dvojice hrozba – scénář, statistická data z minulých projektů a zkušenosti. Výstupem pak jsou N-tice [hrozba, scénář, pravděpodobnost, škoda] a seznamy pro doplnění návrhu projektu, pro informace k možným operativním zásahům a pro následující snižování rizika. Dalším výstupem je předběžná hodnota akceptovatelného rizika a pokyny pro hodnocení souhrnného rizika projektu.

Ve fázi odezva na riziko, také nazývané fáze snižování rizika, je cílem určit, která rizika lze efektivně snížit a navrhnout opatření, která k tomuto snížení povedou. Vstupem jsou N-tice z předchozího kroku a seznam pro snižování rizika. Výstupem jsou, kromě opatření pro snížení rizik, nové hodnoty rizika.

Poslední fází je celkové vyhodnocení rizika. Výstupem je celkové zhodnocení úrovně rizika projektu a zpráva o průběhu analýzy.

2.3.4 Skórovací metoda

Tato metoda je další z metod analýzy rizik, popsaná například v (1) a dalších publikacích. Její přínos spočívá především v dobře přehledné vizualizaci rizik projektu, společně s jejich rozdělením podle dopadu a pravděpodobnosti. Tato metoda je zpracována ve třech fázích:

- identifikace rizika
- ohodnocení rizika
- návrhy opatření ke snížení rizika

Východiskem metody je znalost nebezpečí ze čtyř hlavních oblastí rizik (1):

- technická oblast projektu
- finanční oblast projektu
- personální oblast projektu
- obchodní oblast projektu

V těchto oblastech jsou vybrány rizikové faktory relevantní pro daný projekt, příkladem může být selhání technologie nutné k realizaci nebo pozdní dodávka subdodavatele. Jejich pravděpodobnost i dopad jsou ohodnoceny na desetibodové stupnici (1). Toto hodnocení, stejně jako výběr relevantních rizikových faktorů je

úkolem projektového týmu a zkušenost v něm hraje nezastupitelnou roli, proto není možné jej efektivně automatizovat[11].

Konkrétní hodnocení rizika je pak určeno součinem hodnot pravděpodobnosti a dopadu rizika, rozsahem jsou hodnoty 1 (nejmenší) až 100 (nejzávažnější). Kromě tohoto hodnocení je výstupem dvojrozměrný bodový graf, který znázorňuje rizika rozdělená do čtyř kvadrantů podle závažnosti (1).

Obrázek 8: Mapa rizik. Zdroj: (1)

[11] V dnešní době by bylo teoreticky možné využít analýzy Big Data k vytvoření návrhu, který by následně byl pouze revidován. K tomuto by však byl nutný datový sklad a výpočetní výkon řádově přesahující možnosti současných přenosných zařízení i tohoto projektu.

2.4 Metodiky vývoje software

Software lze vyvíjet pomocí několika metod, běžně dělených na klasické a agilní.

Klasické metodiky jsou zaměřeny především na procesy a jsou řízeny přesným plánem, stanoveným na počátku vývoje. Integrace jednotlivých modulů a testování se obvykle provádí až na konci vývoje, je tedy zvýšená pravděpodobnost nekompatibility modulů z důvodu špatné interpretace funkcionality a dalších problémů. Často bývají používány na tvorbu dokumentů bez větší přidané hodnoty. (5)

Agilní metodiky jsou specifické svým zaměřením na lidi, tedy vývojáře, programátory ale i uživatele. Velmi významným aspektem je komunikace. Podrobné plány jsou tvořeny pouze pro krátké časové úseky, nazývané iterace[12]. Tyto jsou typicky dlouhé maximálně dva měsíce, běžné jsou však i týdenní bloky. Moduly software jsou integrovány a testovány průběžně, což má za výsledek menší chybovost a umožňuje flexibilně měnit návrh podle aktuální potřeby.

Novým přístupem je modelem řízený vývoj, v literatuře označován jako *MDA* (Model Driven Architecture) nebo *MDD* (Model Driven Development). Hlavním principem je oddělení aplikační a podnikové logiky od technologické platformy. Jsou vytvářeny modely nezávislé na technologické platformě, které jsou následně implementovány v prostředí libovolné platformy. Technicky se však nedá mluvit o metodice, protože popsán je pouze přístup k návrhu, nikoli postup řešení. V kontextu mobilních aplikací nachází uplatnění zejména v případě, kdy zákazník vyžaduje aplikaci pro více platforem – například Android a iOS. V praxi se však využívá spíše zřídka, protože zákazník obvykle zadá práci pouze na jedné platformě a následně na další, nezřídka různým společnostem. Výhoda tohoto přístupu je tak převážena vyšší náročností na čas. (5)

[12] Tento pojem bývá použit v literatuře, používaná terminologie se však může mezi firmami lišit. Společnost DactylGroup, ve které pracuje autor této práce, například používá termín Sprint.

Klasické metody (5)

- **Moderní strukturovaná analýza:** Metodika využívající klasický vodopádový model. Při analýze je vytvářen model sestávající z datového modelu a funkční hierarchie – sada diagramů datových toků. Ve fázi návrhu je pak tento model podroben transakční a transformační analýze.
- **Unifikovaný proces vývoje:** Průmyslový standard kostry vývojového procesu. Používá UML[13] notaci, řízen je požadavky, případy užití a riziky. Tato metodika vyžaduje přizpůsobení pro konkrétní firmu.

Agilní metodiky (5)

- **Párové programování:** Metodika založená na úzké spolupráci dvou programátorů. První je zodpovědný za tvorbu aplikační logiky, zatímco druhý tuto transformuje do konkrétních konstrukcí příslušného programovacího jazyka.

- **SCRUM:** Tato metodika je založena na velmi ploché týmové hierarchii. Tým je složen z maximálně sedmi osob, z nichž jedna má roli týmového vedoucího, tzv. SCRUM master. Využívá velmi krátkých iterací, před každou z nich je realizována plánovací schůzka. Následně je po každé iteraci provedena retrospekce, kdy je vyhodnocen dosažený stav.

- **Extrémní programování:** Metodika založená na upřednostňování fungujícího a intuitivního software před rozsáhlou dokumentací. Je vytvářen co nejjednodušší návrh a testování probíhá okamžitě. Nutná je co nejrychlejší zpětná vazba od uživatele, v optimálním případě je proto uživatel přímo členem vývojového týmu.

Při implementaci programu v rámci této práce byla použita metoda extrémního programování. Klasické metodiky při vývoji mobilních aplikací obecně nenacházejí příliš velké využití, vzhledem k specifické struktuře těchto aplikací je totiž transformace modelu do konkrétního kódu poměrně náročná. Z agilních metod pak byla vybrána ta, která umožňuje efektivně pracovat v týmu o jednom člověku.

[13] Unified Modeling Language

3 Analýza současného stavu

Společností, pro kterou vzniká aplikace implementovaná v rámci této práce, je základní škola působící na území města Brna. Škola má dvě samostatné budovy, jejichž součástí jsou celkem čtyři třídy mateřské školy - dvě na budově Merhautova a dvě na detašovaném pracovišti v budově Vranovská. V současné době vzdělává celkem 510 žáků ZŠ a 109 dětí v MŠ.

Pro všechny žáky a rodiče bylo v roce 2010/2011 zřízeno díky partnerství v projektu "Inkluzívní vzdělávání žáků ve školách spádových sociálně vyloučeným lokalitám", reg.č. CZ.1.07/1.2.00/14.0014 Školní poradenské pracoviště s odborníky - psycholožkou, speciální pedagožkou, sociálním pedagogem, výchovnou poradkyní a preventistou sociálně patologických jevů. Díky navazujícímu projektu CZ.1.07/1.2.17/02.0035 "Vzdělávání pedagogů v oblasti speciální pedagogiky a multikulturní výchovy s cílem prevence rasismu a xenofobie a tvorba metodik pro práci s dětmi a žáky se speciálními vzdělávacími potřebami v těchto oblastech na základních a mateřských školách" je možné v započaté práci ŠPP plynule pokračovat.

Organizační struktura je hierarchická, stejně jako v ostatních veřejných školách tohoto stupně.

Primárním uživatelem bude základní škola, nikoli škola mateřská. Z tohoto důvodu jsou veškeré anlýzy zaměřeny primárně na prvostupňové a druhostupňové vsdělávání, v obecné rovině však je možné je rozšířit i na oddělení mateřské školy.

3.1 Kritická analýza

3.1.1 SLEPT analýza

Sociální faktory

Škola se nachází ve městě Brně, městská část Černá Pole. Ze sociálního hlediska je její spádová oblast oblastí se zvýšenou koncentrací sociálně nepřizpůsobivých obyvatel se všemi specifiky, které toto přináší.

Legislativa

Legislativně je fungování školy ovlivněno především Školským zákonem, dále pak samozřejmě Občanským zákoníkem.

Ekonomika

Rozpočet školy je závislý na rozpočtu zřizovatele, tedy obce. Vzhledem k této závislosti je škola přímo ovlivněna celkovou ekonomickou situací státu a kraje. Celkově lze tuto situaci charakterizovat jako stabilní v porovnání se soukromým sektorem, nárůst i pokles se pohybuje v řádu jednotek procent.

Politické faktory

Škola musí zohlednit změny v politické situaci (program vlády, personální obsazení ministra školství), které mohou ovlivnit situaci školství jako takového. V současné době se jedná především o takzvanou inkluzi školství.

Technologické faktory

Změny v technologiích jsou v oboru školství, zejména základního, méně zásadní než v jiných oborech. Projevují se především v oblasti správy informací o žácích a zaměstnancích, ze zákona je však stále nutné uchovávat fyzické kopie všech záznamů, čímž je účinnost omezena. Přínosnější mohou být v oblasti PR školy, kde použití moderních technologií při výuce působí pozitivně na rodiče potenciálních žáků.

3.1.2 Porterova analýza

Konkurenti a charakter jejich soupeření

Konkurenty jsou ostatní základní školy v oblasti, ať už soukromé nebo veřejné.

Segment soukromých škol není, zejména v této geografické oblasti, příliš významný, nepředstavuje tak větší hrozbu. Hlavní výhodou ZŠ Merhautova oproti soukromým školám je fakt, že studium je hrazeno z veřejných financí, je tedy pro rodiče bezplatné.

V případě veřejných škol je majorita žáků rozdělena předem podle takzvaných spádových oblastí. Rodiče s eminentním zájmem o zařazení svého dítěte do jiné školy, než je jejich spádová, riskují odmítnutí z kapacitních důvodů a statisticky jich není mnoho. Tyto odiče se škola snaží zaujmout rozsahem dalších nabízených služeb, jako je organizování volnočasových aktivit a kroužků, dále kvalitou pedagogů a použitím moderní techniky při výuce.

Noví účastníci trhu, potencionální účastníci a hrozba vstupu nových účastníků

V současné době se počet dětí nastupujících do prvních ročníků zvyšuje, zvyšuje se proto i poptávka po místech na školách. Roydílem oproti běžnému trhu zboží a služeb je však plánování školských zařízení na úrovni obce. Toto snižuje pravděpodobnost vstupu nového hráče ze segmentu veřejných škol na trh na minimum, lze předpokládat spíše navyšování kapacit stávajících škol. V segmentu soukromých škol toto riziko existuje, lze však předpokládat že vzhledem k sociálně ekonomické situaci v oblasti nebude přímo ohrožena pozice konkrétně této školy.

Odběratelé a jejich vliv

Odběratele lze rozdělit na dvě skupiny, žáky a jejich zákonné zástupce – rodiče.

Žáci školy nejsou právně způsobilí a jako takoví mají velmi malý oficiální vliv. Nelze také hovořit o vyjednávací síle, nemohou totiž samostatně rozhodnout o změně dodavatele (školy) ani přímo ovlivnit nástup dalších ročníků. Mají však velký vliv na druhou skupinu odběratelů. Přímý vliv mají na hodnocení školy co do uplatnění absolventů, které se následně promítá do financování ze strany zřizovatele. Žákům se škola snaží nabídnout především kvalitní výuku a podmínky pro osobní rozvoj.

Druhou skupinou odběratelů jsou rodiče žáků. Tito mají vyjednávací pozici silnou, v případě masového odlivu by mohla škola i ukončit činnost. Předávají si také informace, podle kterých se nerozhodní rodiče rozhodují, na kterou školu přihlásit svého potomka. I je se škola snaží zaujmout nabídkou volnočasových aktivit pro děti a kvalitou výuky, zároveň se jim snaží vyjít co nejvíce vstříc ve specifických požadavcích a potřebách jednotlivých žáků. Další devizou v nabídce je přidružená mateřská škola, která umožňuje plynulejší přechod prvňáčka do vzdělávacího procesu - odpadne šok ze změny prostředí.

Dodavatelé a jejich vliv

ZŠ Merhautova veškerou výuku zajišťuje vlastním pedagogickým sborem, v tomto směru je tedy nezávislá. Jiná je situace v případě volnočasových aktivit, které jsou v převážné většině řešeny externě. Vzhledem ke konkurenčnímu prostředí jednotlivě nemají příliš silnou vyjednávací pozici, je však nutné počítat s jejich vlivem jako celku.

Dalšími dodavateli jsou firmy dodávající služby a produkty technického charakteru, jako jsou kancelářské potřeby, výmalba objektů a další. Tyto firmy bývají najímány jednorázově či krátkodobě, podle povahy jimi dodávaného zboží nebo služby. Jedinými dlouhodobými dodavateli jsou dodavatelé energií, kteří proto mají také nejsilnější vyjednávací pozici. V případě ostatních dodavatelů je nutné počítat především s možností blokování projektů pokud by byla zvolena konkurenční firma (neúspěšná firma napadne výběrové řízení).

Za určitou formu dodavatelského vztahu by bylo možné označit vztah s rodiči žáků, kdy rodiče dodávají škole právě tyto žáky. Vliv rodičů byl však již popsán v předchozím bloku.

Náhradní výrobky nebo služby

Určitou alternativou ke službám školy mohou být soukromí učitelé. Tato praxe však v České republice není běžná a je svázána tak velkým množstvím regulací, že nelze předpokládat její nárůst.

3.1.3 Analýza 3C

Konkurence

Konkurence je reprezentována ostatními školami. Významnost konkurenčního vztahu se zmenšuje se vzrůstající geografickou vzdáleností. Soukromé školy by potenciálně mohly nabízet lépe cílené služby ve stejné geografické oblasti, jejich finanční nároky jim však v dané oblasti citelně snižují konkurenční potenciál. Veřejné školy mohou nabízet obdobné služby za stejných podmínek, vzhledem k centrálnímu plánování jejich zřizování však vstupuje v platnost faktor vzdálenosti.

Dodavatelé

Dodavatelé škole dodávají především služby ve formě volnočasových aktivit pro žáky. Další skupinou pak jsou dodavatelé materiálu, energií a dalších položek nutných pro chod veřejné vdělávací instituce.

Zákazníci

Zákazníky školy jsou žáci, kteří školu navštěvují. Druhou skupinou jsou pak jejich rodiče, kteří nemají přímý osobní užitek, rozhodují však kterou školu jejich dítě navštěvuje. Za specifického zákazníka může být považována obec, která školu financuje.

3.1.4 7 S

Strategie

Strategie ZŠ Merhautova vychází z úmyslu poskytovat nejlepší možné vzdělání a podporu osobního rozvoje co nejširšímu okruhu žáků. Tato snaha je podpořena vycházením vstříc specifickým potřebám konkrétních žáků a poskytováním poradenských služeb v oblasti osobního rozvoje.

Struktura

Škola má vícevrstvou stromovou organizační strukturu. Vedena je ředitelkou, které přímo podléhají zástupci pro první a druhý stupeň, účetní oddělení a zaměstnanci se

specifickými funkcemi, jako je IT, projektový management a podobně. Konkrétním zástupcům pak podléhají učitelé podle struktury konkrétního stupně vzdělávání, vychovatelky a další personál.

Skupina / Spolupracovníci

Výuka – většina personálu se nějakým zůsobem podílí na výuce. Jedná se o učitele prvního i druhého stupně.

Organizace – veškeré organizační věci spadají pod ředitelku, která může některá rozhodnutí delegovat na své zástupce.

Údržba – K fungování školy je nutná i technická údržba, ať se jedná o úklid, výpočetní techniku nebo budovu samotnou. Tyto osoby jsou jediné, u kterých není vyžadována žádná úroveň učitelské kvalifikace.

Motivace – Pracovníci jsou kromě tabulkového platu hodnoceni i takzvanou osobní složkou platu, která reflektuje jejich přínos škole. Toto je jediná měřitelná forma motivace, neměřitelnou, ale v tomto oboru důležitější, složkou je pocit dobře vykonané práce.

Systémy

Škola využívá informační systém pro správu interních dat, ze zákona však zároveň musí uchovávat veškeré záznamy ve fyzické (papírové) podobě. Velké množství záznamů tak v praxi v informačním systému uloženo není.

Styl řízení

Styl řízení je, jako ve školství tradičně, autoritativní, zpětná vazba od učitelů bývá probrána na pravidelných poradách.

Sdílené hodnoty

Kultura ve společnosti je tradičně spíše formální, v tomto oboru však všichni sdílejí především snahu předat co nejkvalitnější vzdělání dalším generacím. Vzhledem ke školskému zákonu v je pedagogickém sboru vyžadováno vysokoškolské vzdělání.

Schopnosti

Většina pozic je obsazena lidmi z učitelského prostředí, ať už přímo pedagogy, pedagogy volného času nebo psychology se zaměřením na děti a jejich vývoj. K této sadě shopností pak jsou pro každou pozici vyžadovány schopnosti specifické – manažerské pro vedení školy, účetnictví pro účetní a podobně. Primární úlohu v organizaci pak hrají schopnosti komunikační.

3.1.5 Trendy ovliuvňující firmu

Demografie

Po několika letech postupného úbytku prvňáčků lze pozorovat opětovný nárůst, způsobený nástupem dětí silnějších ročníků. Tento je v současné době patrný zejména v mateřských školách, ale bude se postupně projevovat i v pozdějších vzdělávacích ústavech. Škola po oficiálním sloučení na rozdíl od jiných institucí nezrušila své nové pracoviště, proto by měla mít dostatečné prostorové kapacity pro otevření dalších tříd. Zároveň spolupracuje s Pedagogickou fakultou Masarykovy Univerzity v Brně, má tedy dobrou pozici pro získávání dalších učitelů.

Volný čas

V současné době přetrvávající je trend naplnění volného času dětí organizovanou činností ve formě kroužků a kurzů. Tento je částečně poháněn touhou rodičů zlepšit postavení svého potomka ve světě, částečně potřebou umístit jej do péče jiné osoby zatímco rodiče se věnují jiné činnosti – například práci. Lze tedy předpokládat že bude přetrvávat i nadále. Škola svojí nabídkou kroužků a kurzů organizovaných v rámci školní družiny i mimo ni umožňuje toto naplnit v dítěti známém prostředí, zároveň bez nutnosti jej někam odvézt nebo doprovodit, čímž šetří rodičům čas.

Informační technologie

Informační technologie v dnešní době prostupují do všech oblastí lidské činnosti a školství není vyjímkou. ZŠ Merhautova proto postupně vybavuje své třídy interaktivními tabulemi, které, v kombinaci s připojeným počítačem, žákům poskytují

intenzivnější prožitek než klasické. Zlepšuje se tak jejich pozornost a tím i objem naučené látky. V současné době také probíhají projekty, které mají i žákům z nízkopříjmových skupin umožnit naučit se ovládat nejnovější technologie a zlepšit tak jejich potenciální uplatnění.

Inkluze vzdělávání

Inkluze vzdělávání je v současné době často skloňovaný pojem, prakticky však už nějakou dobu probíhá. Jedná se o začleňování znevýhodněných žáků do běžných tříd. S dosud probíhající formou inkluze má ZŠ Merhautova bohaté zkušenosti, dokonce se účastnila i několika pilotních projektů v této oblasti, otázkou však zůstává, jakou konkrétní podobu nabere tento záměr nyní.

3.1.6 SWOT analýza

a) Strengths – silné stránky
b) Weaknesses – slabé stránky
c) Opportunities – příležitosti
d) Threats – hrozby

ad a) Silné stránky

stabilní organizace

zkušení pedagogové

zkušenost s prací se znevýhodněnými žáky

moderní vybavení

ad b) Slabé stránky

slabší vstupní "materiál"

finančně pracuje v kalendářním roce, reálná činnost organizována podle školního roku

problematická lokalita

nedostatečné venkovní tělovývhovné zařízení

ad c) Příležitosti

oprava venkovního hřiště

zvýšení objemu praktické výuky

zvýšení množství kulturních akcí navštívenýcch v rámci výuky

ad d) Hrozby

zvýšení koncentrace sociálně nepřizpůsobivých obyvatel ve spádové oblasti

opravy okolních ulic způsobující rušivý hluk

nutnost uvolnění budovy na žádost městské části

pokles uplatnění absolventů

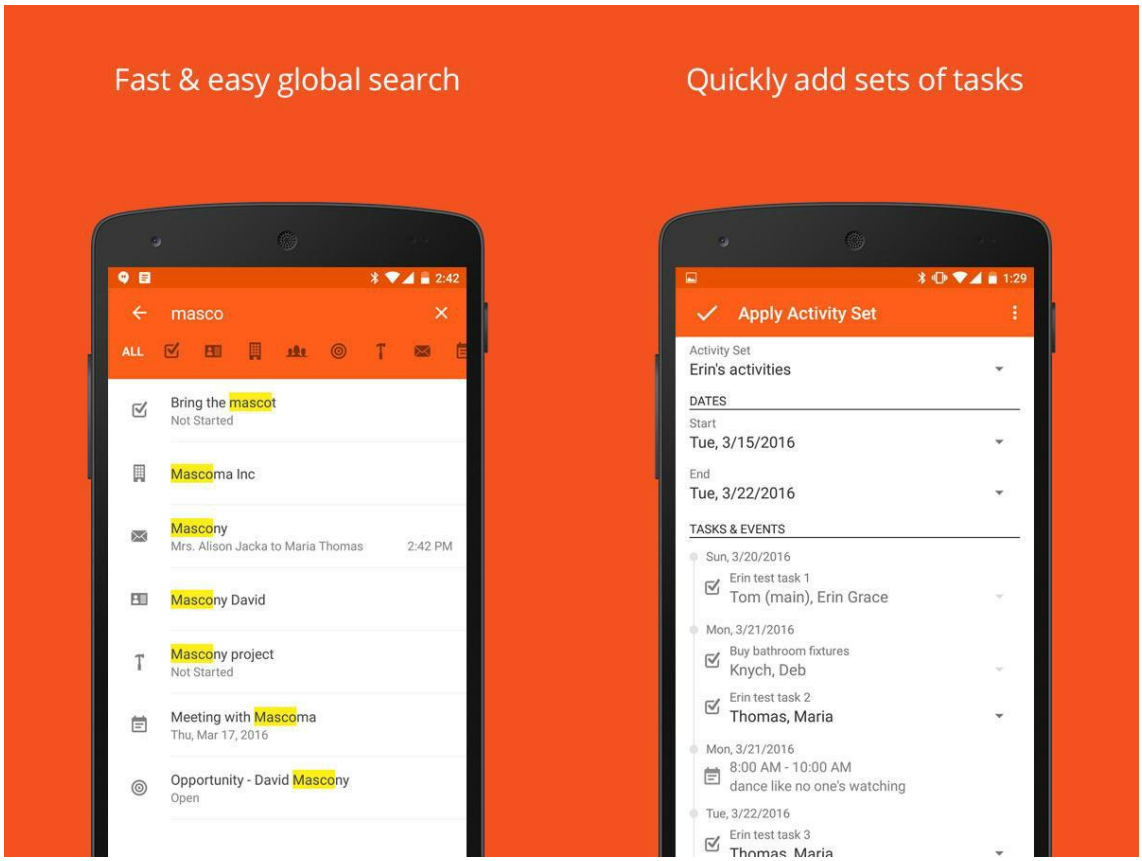
3.2 Současný informační systém a nástroje

Škola v současné době vlastní školní server. Tento slouží primárně k uložení informací o žácích a jejich prospěchu, dále pak interních dokumentů školy a podobně. Vzhledem k omezené přístupnosti se však nedá mluvit přímo o informačním systému. Dalším faktem je, že vzhledem k již dříve zmíněné povinnosti archivovat veškeré dokumenty v papírové formě, je po většinu doby značná část informací neúplná či zastaralá. Podporu projektového managementu pak tento systém neposkytuje žádnou. Tento stav je známý a jeho řešení je v dlouhodobém plánu, v současnosti je však podpora z této strany mizivá. Výhodou tohoto stavu je, že vyvíjená aplikace nemá silného, zavedeného konkurenta v rámci organizace, s existující vyhlídkou na integraci s novým serverem (vlastnost u běžně dostupného softwaru neexistující) tak může být snadno přijata.

3.3 Existující systémy

Na trhu v současné době existuje velké množství různých systémů a aplikací, které vytvářejí softwarovou podporu pro projektový management, a to jak pro PC, tak pro přenosná zařízení. Nabízená funkcionalita se však mezi těmito formami významně liší. Zatímco PC řešení typicky poskytují podporu pro všechny fáze projektu, mobilní aplikace obvykle podporují pouze operativní řízení.

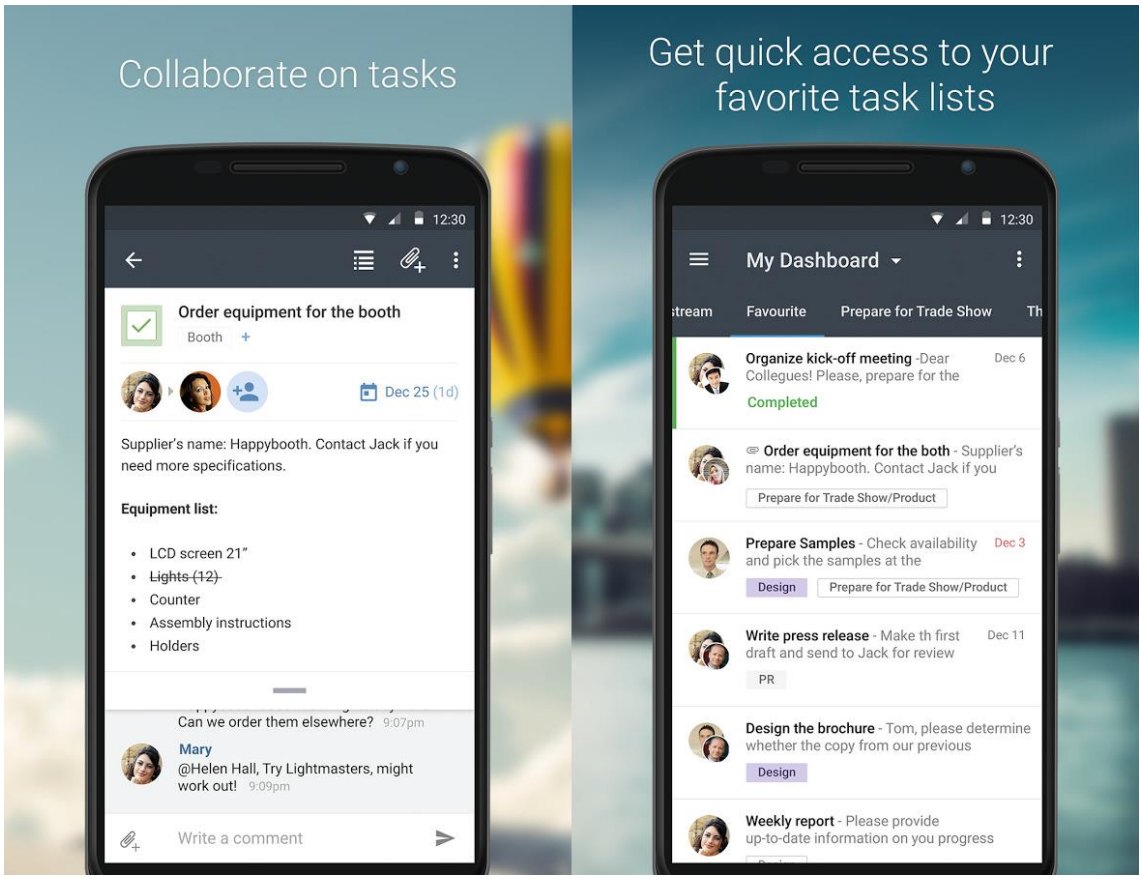
Tuto situaci dobře ilustruje například oblíbená aplikace Insightly (6). V této aplikaci je velmi dobře zpracovaný systém seznamů úkolů včetně upomínek a hlášení termínů, nepodporuje však žádnou formu odhadu a snižování rizik.



Obrázek 9: Aplikace Insightly. Zdroj: (6)

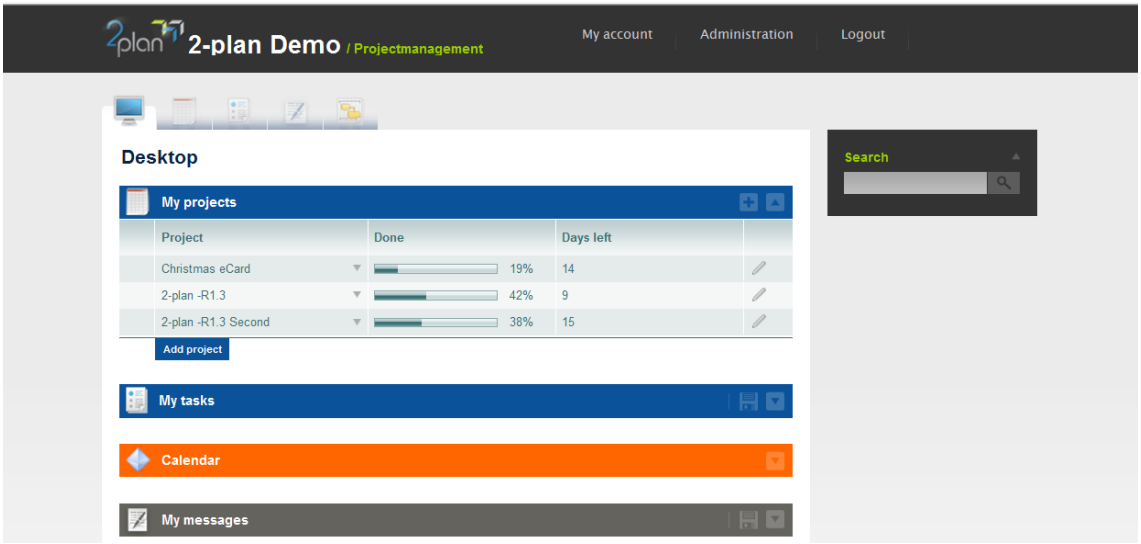
Dalším příkladem může být mobilní verze desktopové aplikace Wrike (7). Tato aplikace umožňuje zobrazovat seznamy úkolů a Ganttovy diagramy, stejně jako jejich tvorbu. Dalším pozitivem je optimalizace pro telefony i tablety. Poskytovaná podpora ostatních

fází projektového řízení je však opět nízká až žádná. Zajímavou funkcionalitou je pak možnost přidávat k jednotlivým položkám fotografie, video a audio záznamy.



Obrázek 10: Aplikace Wrike. Zdroj: (7)

Druhou skupinou aplikací, které mohou být použity na mobilních zařízeních, jsou programy s webovým rozhraním. Jedná se o programy určené primárně k použití z PC, ovládání s použitím dotykového displeje je tak často velmi obtížné, je však možné. Poměrně dobře využitelným příkladem může být 2-plan (13). Tento systém má na desktopovou aplikaci nezvykle velké ovládací prvky, je tedy možné je, s trochou šikovnosti, využít i na mobilním zařízení. Provozovatel také nabízí doprovodnou mobilní aplikaci, která poskytuje podporu operativního řízení. Zásadní nevýhodou tohoto řešení je fakt, že veškerá data jsou uložena na serverech poskytovatele, využití tohoto systému tak závisí na internetovém připojení a může být vnímáno jako bezpečnostní riziko.



Obrázek 11: Webové rozhraní 2-plan. Zdroj: (13)

4 Návrh řešení a přínos návrhů řešení

V této kapitole bude v první části aplikace rozebrána z hlediska návrhu, v druhé části pak bude popsána konkrétní implementace. V obou částech jsou zahrnuty přínosy, které konkrétní bod přináší, v sekci implementace pak i hodnocení alternativ.

4.1 Návrh

Tato kapitola se zabývá návrhem aplikace a blíže rozebírá podporovanou funkcionalitu.

4.1.1 Celkový návrh

Obecně osvědčený přístup při návrhu mobilní aplikace je maximalizovat samotnou pracovní plochu s tím, že nástroje jsou převážně skryty. Toto je dáno především velikostí obrazovky přenosných zařízení v kombinaci s velikostí ovládacích prvků, nutnou pro efektivní práci za pomoci prstů, nikoli kurzoru myši. Většina akcí se tak zobrazuje v kontextu s označeným prvkem. Příkladem může být otevření softwarové klávesnice při výběru textového pole nebo nabídka aplikovatelných možností při označení pole variantního vstupu. Zbývající prvky ovládání, nezávislé na kontextu aplikace, byly do verze 3.0 Honeycomb (viz sekce 2.1.1) sdruženy pod tlačítkem *Menu,* často reprezentovaným samostatným hardwarovým prvkem. V novějších verzích je tento přístup opouštěn ve prospěch prvku *ActionBar*, zobrazeného na žádost uživatele, respektive *Toolbar*, odkaz zobrazujícího pouze minimum relevantních ovládacích prvků (8). Na pracovní ploše jsou zobrazeny pouze ovládací prvky, u kterých by skrytí přineslo nepřípustné zpomalení práce, tedy zástupci nejvíce používaných akcí. O které konkrétně se jedná je dáno kompromisem mezi přehledností plochy a dobou nutnou k provedení operací. Toto je záležitostí konkrétního kontextu a nelze proto rozhodnout obecně. Vzhledem k rozsahu zobrazovaných informací je aplikace optimalizovaná pro tablety, tedy zařízení s úhlopříčkou sedm palců a větší.

V časovém rámci diplomové práce nemá autor přístup k vhodnému aplikačnímu serveru (viz 3.2), jehož rozhraní je určující pro vývoj aplikace určené k práci online a je pro každou implementaci serveru unikátní. Aplikace tak bude vyvíjena pro práci offline

s tím, že veškeré operace s daty budou realizovány prostřednictvím pomocné třídy. V případě rozšíření funkcionality o komunikaci se serverem tak bude nutné pouze doplnit tuto třídu o synchronizaci, závislou na implementaci serveru, a aplikace pak bude schopná plnohodnotně fungovat bez dalších úprav.

4.1.2 Android for Work v aplikaci

Jak bylo řečeno v sekci 2.2.5, existuje program *Android for Work*, který určuje některá specifika aplikací určených pro korporátní využití. Tyto mají dopad především na aplikace softwarově integrované do dalších systémů, o čemž se v případě vyvíjené aplikace neuvažuje. Vyvíjená aplikace je plně samostatná a nevyužívá žádné z kritických oprávnění, reakce na změnu uživatelského profilu ze vzdáleného zdroje tak není nutná. Jedinou výjimkou v tomto případě je oprávnění pro zápis do databáze, v tomto případě však je potřebná funkcionalita obsažena v použité knihovně.

4.1.3 Zabezpečení

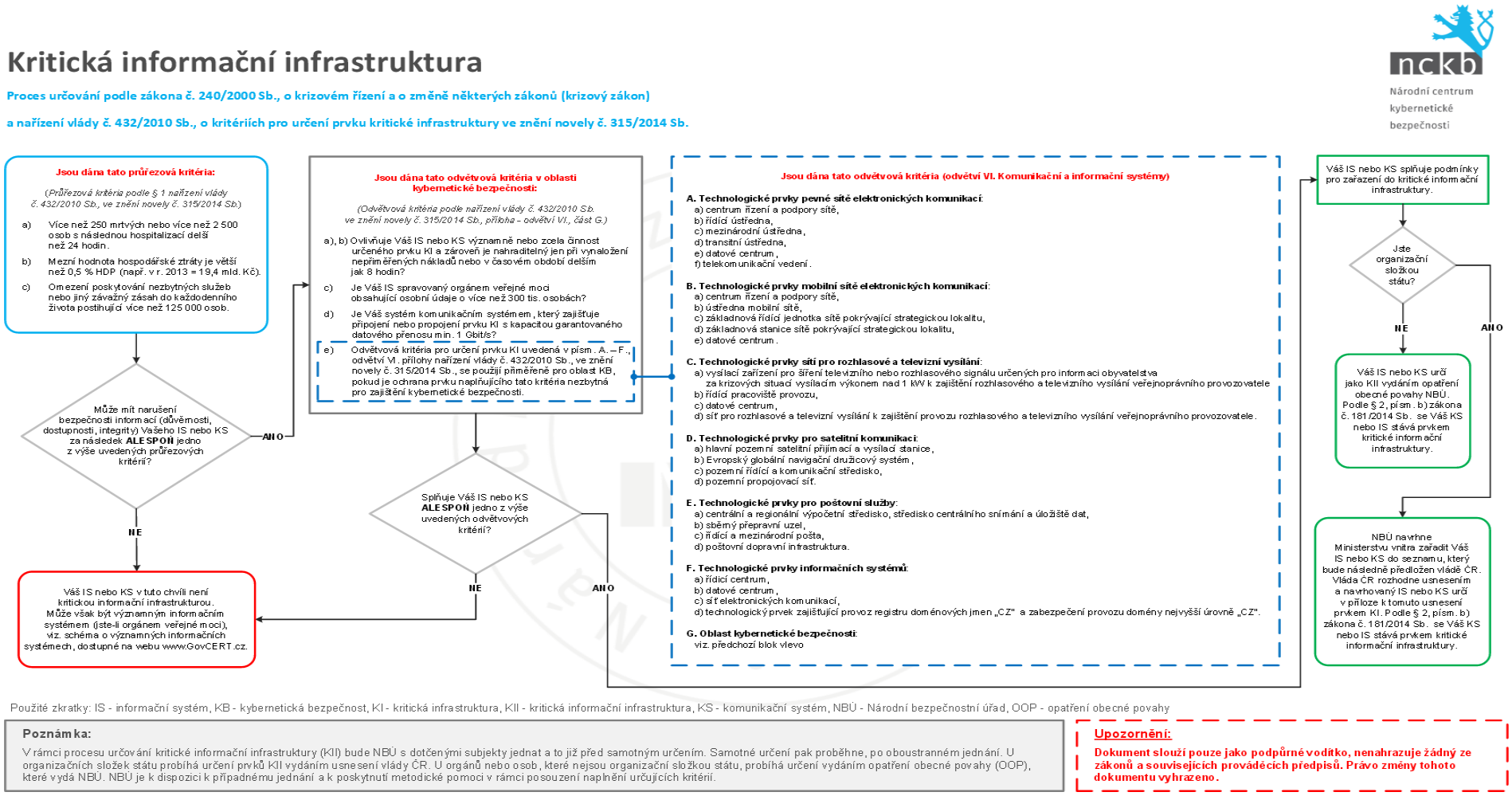
Zabezpečení aplikace lze rozdělit na dvě oblasti - zabezpečení dostupnosti a zabezpečení dat (2).

Oblast dostupnosti je důležitá zejména v průmyslových řešeních a systémech, v případě uživatelských aplikací není tak podstatná - zdržením z důvodu nedostupnosti nevznikají velké škody. Zároveň je jednoduché aplikaci v případě potřeby restartovat, není nutné projít cyklem technologických procesů odstávky[14]. Z toho důvodu není nutné ji příliš řešit, samozřejmě za dodržení běžných opatření co se týče provozu zařízení a serverů.

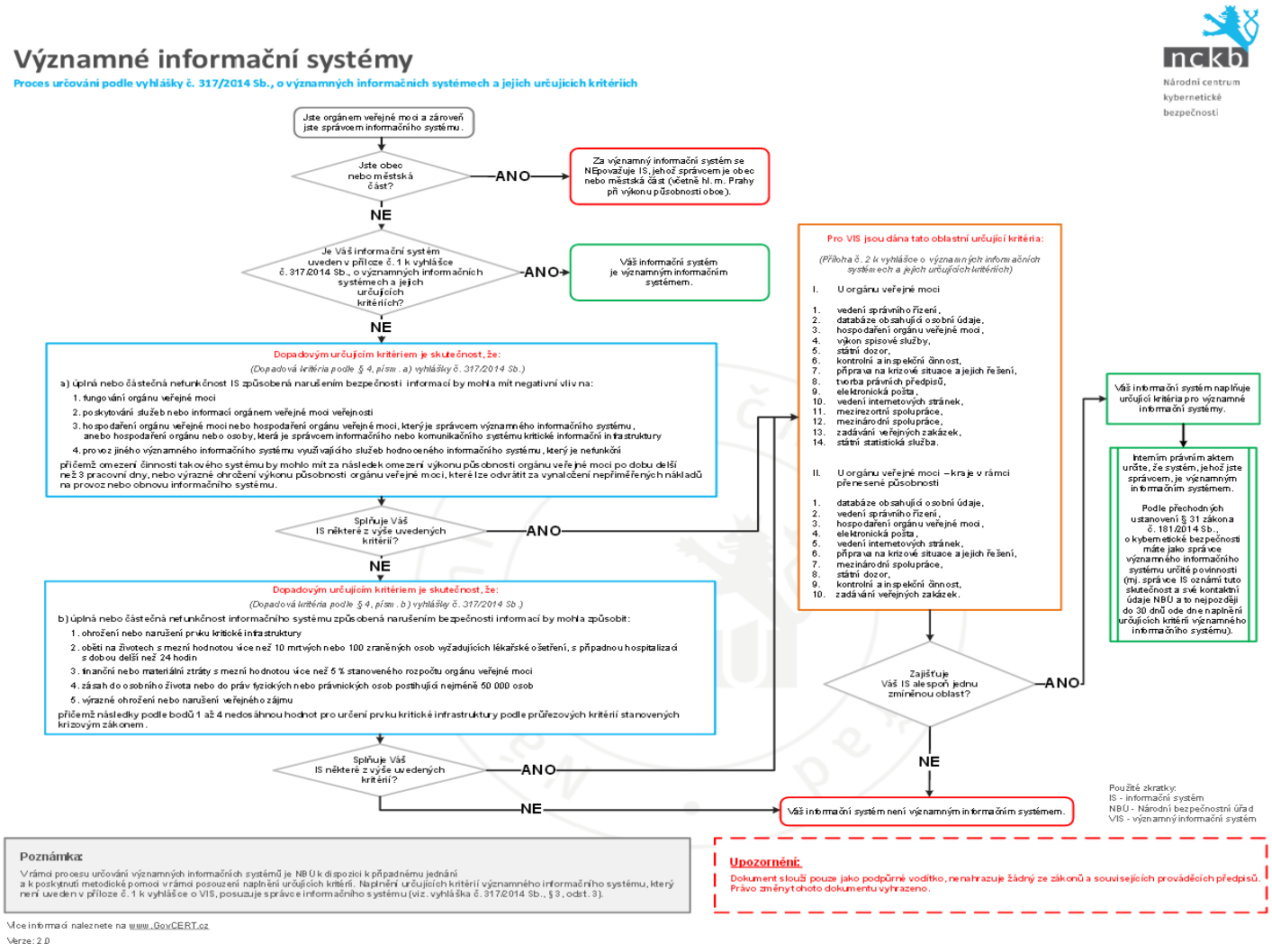
Oblast zabezpečení dat je naopak kritická, dokonce kritická natolik, že je od 1. ledna 2015 ošetřena zákonem o kybernetické bezpečnosti (14). Tento zákon přesně definuje nutné zabezpečení pro prvky *kritické infrastruktury* a *významných informačních systémů*, stejně jako definuje tyto pojmy. Národní centrum kybernetické

[14] Opačným případem by bylo například řízení jaderné elektrárny, kde restart zařízení může znamenat několikaměsíční proces

bezpečnosti pro lepší orientaci vytvořilo grafiky Obrázek 1 a Obrázek 13. Podle těchto dokumentů navrhovaná aplikace nespadá do kritické infrastruktury, v případě využití orgánem veřejné moci by se však podle striktního výkladu mohlo jednat o významný informační systém. Vzhledem k faktu, že zabezpečení vyžadované zákonem není možné implementovat na přenosných zařízeních bez serverové podpory, není použití v tomto kontextu doporučeno.



Obrázek 12: Určení kritické infrastruktury. Zdroj: (19)



Obrázek 13: Určení významného informačního systému. Zdroj: (19)

Základní zabezpečení mobilních zařízení je buď biometrické (otisk prstu, případně rozpoznání obličeje), nebo dvoufaktorová autentizace. Biometrické zabezpečení je podmíněno hardwarovým vybavením zařízení a nebude proto použito, nicméně je možné ho použít k odemčení samotného zařízení, bez kterého není možné aplikaci spustit. Dvoufaktorová autentizace je běžnější, typicky bývá využíváno uživatelské jméno a heslo. Tento model vyžaduje autorizaci ze serveru, protože uložení hesel v zařízení je samo o sobě považováno za porušení bezpečnosti. Vzhledem k offline návrhu (viz 4.1.1) tak není dost dobře možné tento model implementovat, ačkoli je pro něj připraveno rozhraní. Kontrola uživatele tedy proběhne, je však vždy vrácena jako pozitivní. Jedná se o běžnou, ne však žádoucí funkcionalitu a doplnění příslušné oblasti by mělo být prvním úkolem při použití aplikace s webovým serverem. V důsledku těchto voleb je aplikace prakticky ponechána otevřená. Ve většině případů je za dostatečné zabezpečení aplikací pro Android považována uzamčená vstupní obrazovka, v tomto případě však je silně doporučeno využití dodatečných prvků ochrany, jako je *AppLock* nebo *Smart AppLock*

4.1.4 Podpora pro WBS

Softwarová podpora pro WBS, s ohledem na omezený výkon přenosných zařízení, je možná ve formě formuláře, do kterého jsou vypisovány jednotlivé položky. Tento formulář pak může upozorňovat na nesrovnalosti v návrhu co do hodnot, tedy například že uvažovaný návrh nepokrývá 100 % projektu.

Identifikátor	Popis	Náklady	Podíl práce

Obrázek 14: Formulář pro zápis WBS. Zdroj: vlastní zpracování

Výhodou tvorby tohoto formuláře v rámci specializované aplikace oproti papírové formě je, že může přesně reflektovat potřeby konkrétního projektu co do rozsahu. V tomto případě bude v základním stavu zobrazovat pouze jeden řádek, který bude v rámci tvorby WBS rozvíjen.

Identifikátor	Popis	Náklady	Podíl práce
	Jméno projektu	0	0

Obrázek 15: Základní stav vstupního formuláře. Zdroj: vlastní zpracování

Klikem na kód nebo popis úkolu bude vytvořen další podúkol, spadající pod něj. V následném dialogu bude možné upravit hodnoty rozpočtu a práce spojené s tímto podúkolem. Tyto bude možné upravit i dále. Samozřejmostí je možnost mazání omylem vytvořených nebo neplatných položek.

Hlavním výstupem tohoto bloku bude záznam v databázi reprezentující položky projektu. Tento výstup by také mělo být možné uložit ve formátu vhodném pro případný tisk a elektronickou archivaci.

4.1.5 Podpora pro metodu kritické cesty

Metoda kritické cesty využívá síťového grafu, do kterého jsou zakresleny jednotlivé činnosti a vazby mezi nimi. Potřebným vstupem je tedy seznam těchto činností, dob jejich trvání a vazeb mezi nimi. Pro zadání těchto seznamů je opět možné využít tabulkové struktury, konkrétně ve formátu ukázaném na Obrázek 16. V tomto formuláři jsou zahrnuty pouze položky zadávané uživatelem, další – jako je například jednoznačný identifikátor každé činnosti – je možné doplňovat automaticky. Položka „Předchozí činnosti“ obsahuje seznam identifikátorů činností, které jsou přímými předchůdci dané činnosti. Vazbu na následující (řízené) činnosti je možné doplnit automatiky.

Popis činnosti	Doba trvání	Předchozí činnosti

Obrázek 16: Formulář popisu činností pro metodu kritické cesty. Zdroj: vlastní zpracování

Výstupem metody je vizualizace postupu činností se zvýrazněnou kritickou cestou. Dalšími výstupy jsou doba trvání celého projektu a data dokončení jednotlivých činností, potřebné pro zobrazení stavu projektu.

4.1.6 Podpora pro RIPRAN

Metoda RIPRAN používá formuláře pro formální zápis výstupů jednotlivých fází. Tyto formuláře musí být v podpoře obsaženy. Opět je zde možné použít přizpůsobení rozsahu potřebám konkrétního projektu, podobně jako u podpory WBS v sekci 4.1.4

Pořadové číslo	Hrozba	Scénář	Poznámky

Obrázek 17: Formulář identifikace hrozeb pro metodu RIPRAN. Zdroj: (18)

Pořadové číslo	Hrozba	Scénář	Pravděpodobnost	Dopad	Hodnota rizika	Poznámky

Obrázek 18: Formulář kvantifikace hrozeb pro metodu RIPRAN. Zdroj: (18)

Pořadové číslo	Návrh na opatření	Nová hodnota rizika	Náklady na opatření	Zodpovědnost	Poznámka

Obrázek 19: Formulář snižování rizika pro metodu RIPRAN. Zdroj: (18)

Primárním výstupem je v tomto případě sada dokumentů ve formátu vhodném pro tisk a elektronickou archivaci, která dokládá provedení a výsledek jednotlivých fází. Tyto dokumenty musí, kromě samotných dat o projektu, obsahovat také jednoznačnou identifikaci projektu, osoby zodpovědné za vytvoření a schválení příslušného dokumentu a datum (respektive čas) vyhotovení. V rámci programu by zejména ve fázi snižování rizika měly být graficky zvýrazněny položky s vyššími hodnotami rizika, tedy ty vyžadující zvýšenou pozornost.

4.1.7 Podpora pro skórovací metodu

Tato metoda umožňuje efektivní podporu ve formě automatického zpracování vložených informací do grafického výstupu. Vstupním bodem informací je, jako u předchozích, formulář pro popis a hodnocení rizik. U této metody je formalizována i tabulka pro vstup hodnocení rizik od vícečlenného týmu, kterou by bylo vhodné použít i

v ostatních případech hodnocení rizik – konkrétně u RIPRAN fáze kvantifikace rizika v sekci 4.1.6.

Pořadové číslo	Rizikový faktor	Poznámka

Obrázek 20: Formulář identifikace rizikových faktorů pro skórovací metodu. Zdroj: (1)

Kvantifikace rizik členy analytického týmu	1.	2.	3.	4.	5.	6.	7.	8.	Průměrná hodnota	x
Možnost výskytu (1-10)										
Dopad (1-10)										
Ocenění rizika = průměr pravděpodobnosti x průměr dopadu										

Obrázek 21: Formulář k ocenění rizik pro stanovený rizikový faktor. Zdroj(1)

Vzhledem k relativně malé úrovni detailu hodnocení, stupnici 10 bodů pro obě kritéria, může docházet ke značnému překryvu bodů zobrazených v mapě rizik, v krajním případě je možným výsledkem i graf se zobrazeným jediným bodem, přestože rizikových faktorů je v projektu mnoho. Graf je tedy nutné implementovat tak, aby relativně malý potenciální obor hodnot nesnižoval čitelnost. Toho může být docíleno buď zobrazováním počtu rizik v překryvu, nebo například tvarovým odlišením jednotlivých bodů.

Stejně jako u předchozích metod i zde by neměla chybět možnost vytvoření dokumentu pro archivaci, zahrnující kromě výpisu rizikových faktorů a mapy rizik i identifikaci projektu, zodpovědné osoby a čas vyhotovení.

4.2 Implementace

Struktura aplikace pro Android je poměrně specifická. V obecné rovině je jedna obrazovka realizována alespoň jednou, častěji však více třídami. Z tohoto důvodu pro popis implementace není vhodné použít diagram tříd, typický nástroj řešení tohoto problému. Následující kapitola proto bude zahrnovat popis implementace funkcionality řazený podle jednotlivých implementačních oblastí. Z důvodu větší čitelnosti byly ukázané zdrojové kódy v jazyce Java nahrazeny pseudokódem[15]. Vložené obrázky z použití aplikace byly pořízeny jako screenshoty na zařízení Samsung GT-N8000 se systémem Android 4.1.2, Api 16. Jedná se tedy o desetipalcový tablet s téměř nejstarším podporovaným systémem. Z tohoto důvodu nejsou některé grafické prvky tak dokonalé, jak by mohly být při zobrazení na novějších zařízeních, funkčnost však je zachována.

4.2.1 Zpětná kompatibilita

Podle návrhu má být možné aplikaci spustit na systémech 4.0.3 Ice Cream Sandwich a novějších. Není proto nutné implementovat starší rozhraní typu menu, místo něj je použit novější *Toolbar*. Tato a další funkcionalita však není v nejstarších verzích podporována přímo, z tohoto důvodu jsou použity knihovny kompatibility, *v-4-appcompat* a *v-7-appcompat* (8).

Tyto knihovny poskytují implementaci velkého množství chybějících funkcí moderních prvků, k jejich implementaci však využívají zdroje dostupné na původním systému. Jejich vzhled se proto může lišit od jejich moderních vzorů. Některé funkce pak nejsou implementovány vůbec, příkladem může být selektor *Wheel* sloužící k výběru hodnot z přednastavených možností s podporou plynulého přechodu mezi koncem a začátkem pole dat. Tento je implementován již od verze 1, nicméně pouze jako součást prvku *DatePicker* který, jak název napovídá, slouží k zadání data ze strany uživatele a obsahuje tak poměrně specifické rozložení komponent. Samostatný *Wheel* je

[15] Kompaktní a neformální způsob zápisu počítačového algoritmu, který používá strukturní konvence programovacích jazyků, avšak nezahrnuje detailní syntaxi, jako jsou deklarace proměnných, subprocedury nebo jiné konstrukce specifické pro konkrétní programovací jazyk. Zápis může být pro srozumitelnost částečně doplněn popisy podrobností v přirozeném jazyce nebo kompaktně vyjádřeným matematickým zápisem.

pak podporován až ve verzi 16 a vyšší, tedy o dvě generace novější než je verze cílová, a v knihovnách obsažen není. V tomto a dalších podobných případech je použit prvek odpovídající cíli, ačkoli často může působit zastaralým dojmem – konkrétně v tomto případě se jedná o prvek *Spinner*, který umožňuje stejný přístup k výběru hodnot, je však orientován pouze ve směru začátek – konec, nelze plynule přejít z poslední položky na první.

4.2.2 Fragment nebo aktivita

Základním návrhovým rozhodnutím je, jestli jednotlivé obrazovky aplikace vytvořit pomocí aktivit nebo jako aktivity s fragmenty (viz 2.2.3). V případě vyvíjené aplikace byl zvolen přístup zahrnující samostatné aktivity, protože ovládací prvky na jednotlivých obrazovkách přes podobný vzhled vyžadují samostatnou implementaci, použití fragmentů by tak přineslo pouze komplikovanější zdrojový kód. Toto řešení také umožňuje využít implicitní reakce systému na stisk tlačítka "Zpět" na zařízení, kdy je otevřená aktivita ukončena a do popředí navrácena aktivita předchozí – z pohledu uživatele se tedy vrátí o obrazovku zpět. Tuto funkci by v případě použití fragmentů bylo nutno implementovat znovu.

4.2.3 Uložení dat

Aplikace je určena pro práci v režimu offline, je tedy nutné řešit perzistentní[16] uložení dat v zařízení. K tomuto účelu je využito databáze *SQLite*, podporované všemi systémy Android. Oproti tradiční SQL databázi známé ze serverových prostředí má několik specifik.

Veškerá data jsou ukládána ve formě textových řetězců a velikost jednotlivých datových typů tak není stanovena – odpadá tedy dělení na *smallint, bigint* a podobně. Toto řešení přináší značné výhody v implementaci, kdy odpadá nutnost pokročilé typové kontroly, specializace datových polí a udržování prázdných míst pro pole s nulovou hodnotou, zároveň však přináší riziko špatné interpretace dat. K omezení

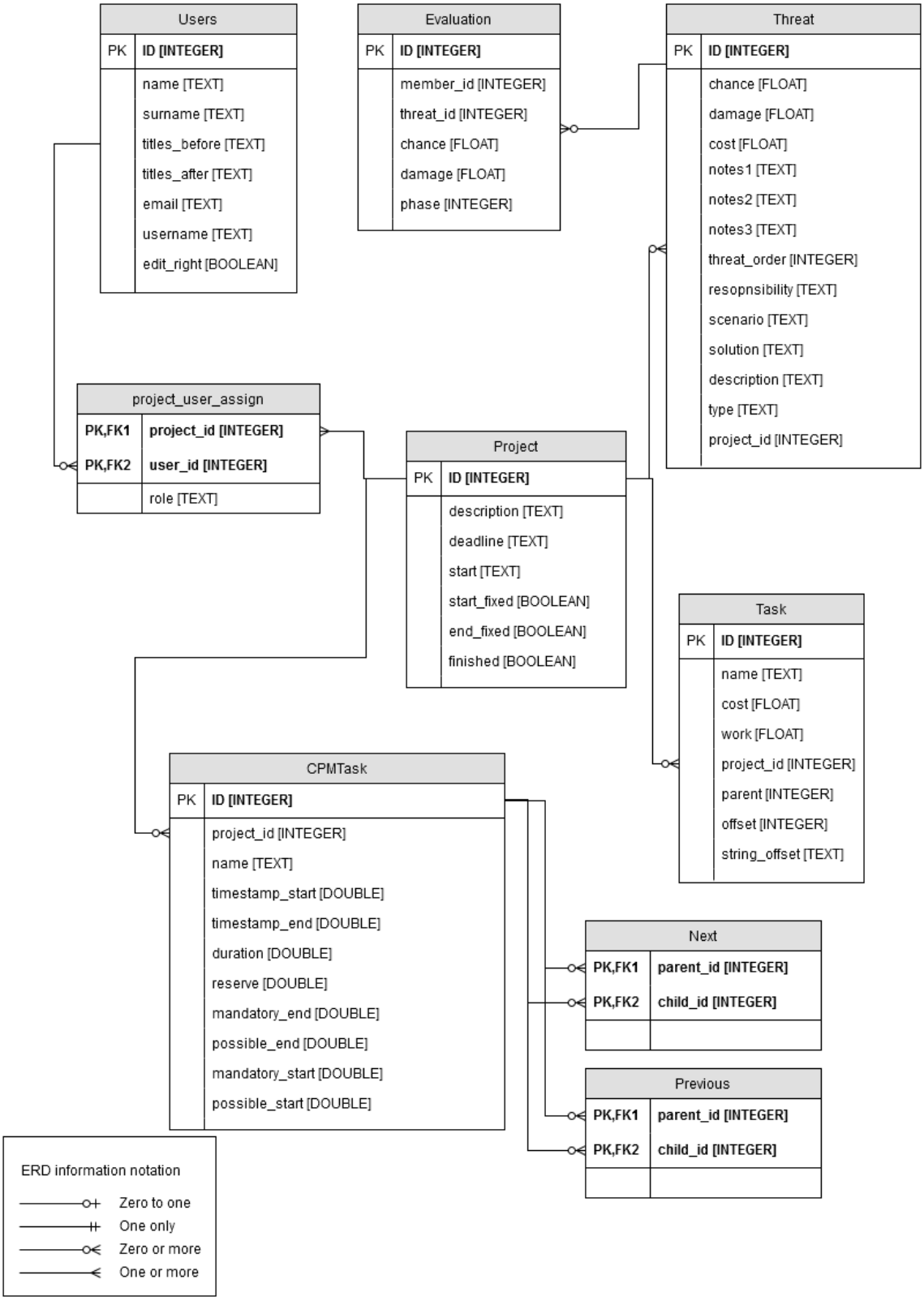
[16] Trvalé, tak aby uložená data byla dostupná i při budoucích běžích aplikace

tohoto rizika byly opětovně zavedeny typy, které v tomto případě slouží pouze k omezení zadávaných hodnot a jejich kontrole ještě před samotným zápisem do databáze. Prováděná kontrola je tak rychlejší a méně náročná na zdroje. Zároveň tak je možné pracovat s předpokladem, že v poli označeném jako *integer* bude řetězec interpretovatelný jako číslo.

V tomto projektu je pro práci s databází použita podpůrná knihovna *Sugar ORM* (15). Tato knihovna zjednodušuje a zpřehledňuje interakci s databází v zařízení, dříve obsluhovanou výhradně pomocí nativní třídy *DatabaseHandler*. Umožňuje tak práci s většími datovými celky na úrovni tříd, což v praxi velmi zjednodušuje implementaci databázových modelů – cílovým stavem je možnost práce s databázemi čistě na úrovni tříd, bez nutnosti znalosti jazyka SQL. Programátor se však tímto vzdává určité úrovně kontroly nad přesným děním, včetně pořadí některých operací a implementací některých bloků. V současné době, vzhledem k úrovni vývoje této knihovny, je proto pořád vhodné vytvořit si kvalitní návrh databáze. Typickým příkladem, kdy by tento měl být využit, jsou vazby mezi jednotlivými tabulkami. Knihovna *Sugar ORM* sice přímo podporuje vazby typu 0/1:1, vazby 0/1:N však v současné verzi přímo podporovány nejsou[17]. Vzhledem k povaze tohoto projektu je však jejich využití nutné, proto bylo vytvořeno schéma zobrazené jako Obrázek 22.

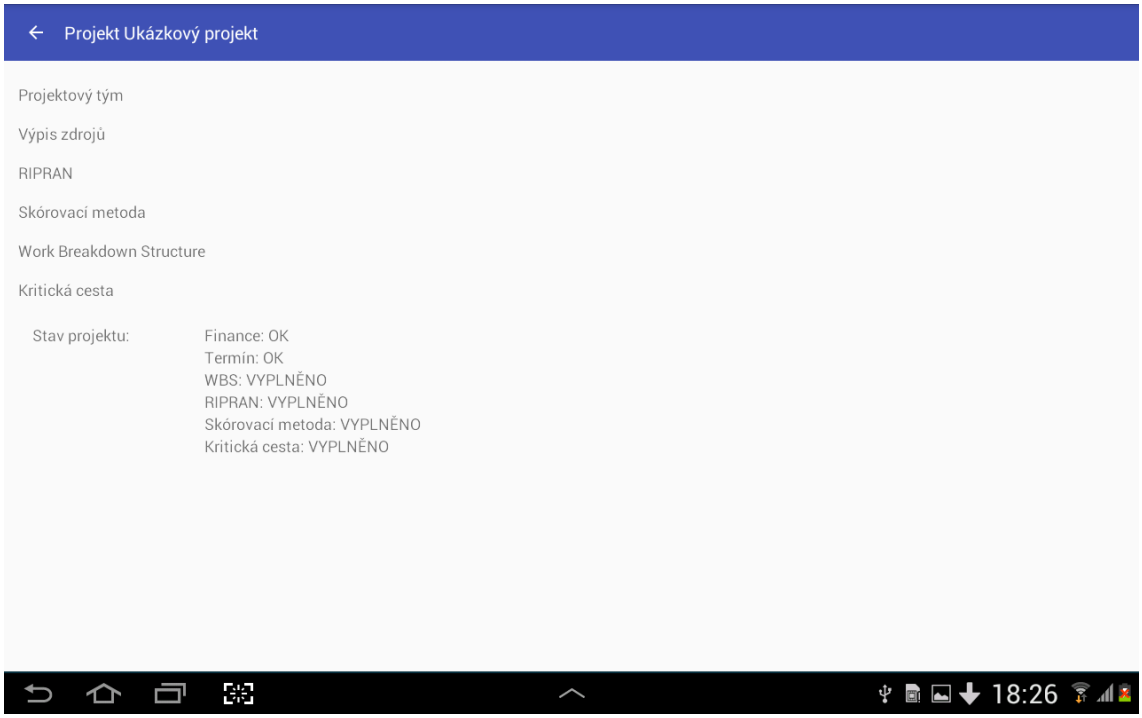
Toto databázové schéma je typu sněhová vločka, typickém pro databáze podobného rozsahu a určení. Formálně splňuje nároky Druhé normální formy, nároky Třetí normální formy nejsou splněny z důvodu tabulky *CPMTask*, kde hodnoty počátků, trvání a konců jsou závislé. Všechny zmíněné hodnoty jsou však potřebné k výpočtu, navíc tento návrh slouží primárně k lepší orientaci v problematice vazeb, tento fakt tak není na závadu.

[17] Samotná vazba 1:N podporována je, problém však nastává při vícenásobné kumulaci těchto vazeb. Problém spočívá ve faktu, že v databázi SQLite není dost dobře možné vytvořit fungující index, veškeré vazby jsou proto vytvářeny metodou serializace a následné deserializace, tedy převedení třídy na textový řetězec a následně zpět. Tato funkcionalita je knihovnou podporována, byť je nutné určit pravidla pro serializaci konkrétních tříd. V případě, že by však odkazovaná třída obsahovala odkaz na další třídy, došlo by ke značné kumulaci dat v jedné "buňce" databáze a tato by nebyla uložena, čímž by byla vazba efektivně ztracena.



Obrázek 22: Databázové schéma. Zdroj: vlastní zpracování

4.2.4 Logika aplikace



Obrázek 23: Základní obrazovka projektu. Zdroj: vlastní zpracování

Obrázek 23 reprezentuje základní obrazovku projektu. Tato obrazovka obsahuje základní informace a hlavně rozcestník pro další akce spojené s projektem.

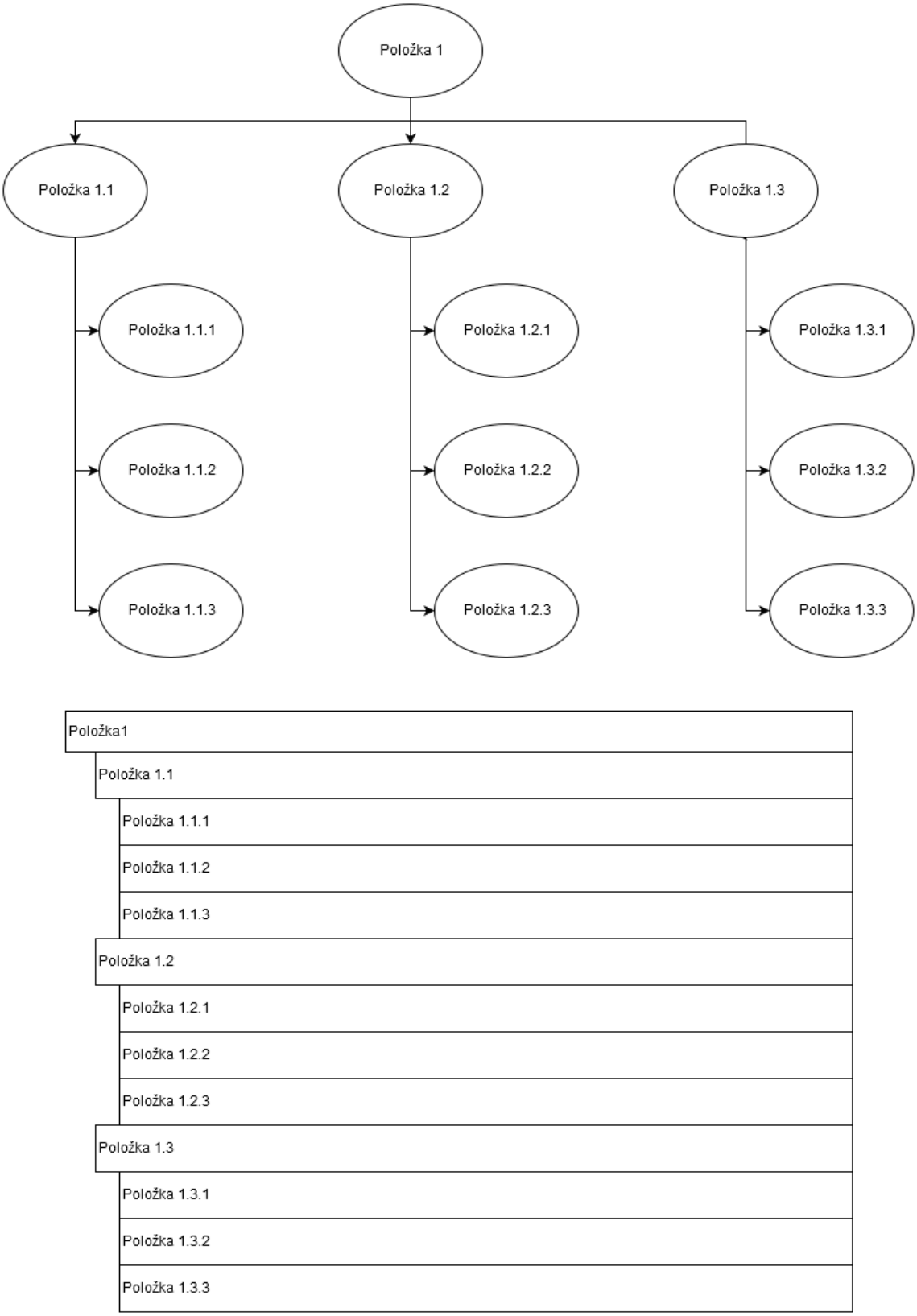
Položky *Projektový tým* a *Výpis zdrojů* jsou odkazy vedoucí na příslušné seznamy, v těchto obrazovkách je také možné přiřazovat a vytvářet položky těchto seznamů.

Blok *Stav projektu* obsahuje systémové informace o zvoleném projektu. Položky *WBS*, *RIPRAN*, *Skórovací metoda* a *Kritická cesta* ukazují, jestli má zvolený projekt v databázi záznamy odpovídající těmto kategoriím. Položka *Finance* vychází z informací WBS a indikuje, jestli je dodržen rozpočet. V případě že není je zobrazena aktuální odchylka od očekávané částky. Položka *Termín* vychází z Metody kritické cesty a indikuje, zda je dodržován časový plán. V případě že není, je vypsáno aktuální zpoždění. Pokud má projekt zadaný časový rámec, probíhá i kontrola, jestli je možné projekt v tomto časovém rámci dokončit.

4.2.5 WBS

Z technického hlediska je WBS stromovou strukturou. Tuto je možné zobrazit dvěma způsoby – jako větvený strom nebo jako seznam se zanořováním úrovní, viz Obrázek 24. Varianta větveného stromu je využitelná zejména při ručním zpracování nebo při zobrazení na velkých obrazovkách, v kontextu přenosných zařízení je velmi nepraktická. Z tohoto důvodu byla zvolena varianta seznamu se zanořováním. Ta je v ručním zpracování prakticky nepoužitelná, protože její použití by znamenalo nutnou znalost rozsahu jednotlivých uzlů už v době jejich tvorby, není tedy dost dobře možné takto zapsaný strom rozšiřovat. V případě zpracování s podporou výpočetní techniky však tento problém odpadá, protože je bez problému možné překreslit aktuálně zobrazený strom a přidat příslušný počet uzlů.

Tento seznam lze na platformě android implementovat dvěma způsoby. Prvním je prvek *ListView*, který je přímo určen k zobrazování podobných seznamů. Jeho adaptér tedy obsahuje mnoho předdefinovaných funkcí, jako je posouvání, postupné načítání dat v případě velmi dlouhých seznamů a další. Je však primárně určen k zobrazení jednorozměrných seznamů, jako je například seznam kontaktů nebo položek v nákupním košíku. Pro potřeby zobrazení WBS je však nutné zohlednit i zanoření jednotlivých uzlů. Z tohoto důvodu bylo rozhodnuto o vlastní implementaci za pomoci prvku *LinearLayout*. Tento představuje abstraktní kontejner, do kterého jsou přidávány další prvky podle potřeby. Logika přidávání prvků byla vytvořena na základě datového modelu uzlu jako funkce *vykresli stromovou strukturu* v bloku Obrázek 25. Tato funkce je volána vždy při vstupu do příslušné aktivity a následně při úpravě datového souboru, tedy při tvorbě, mazání a úpravách jednotlivých položek.



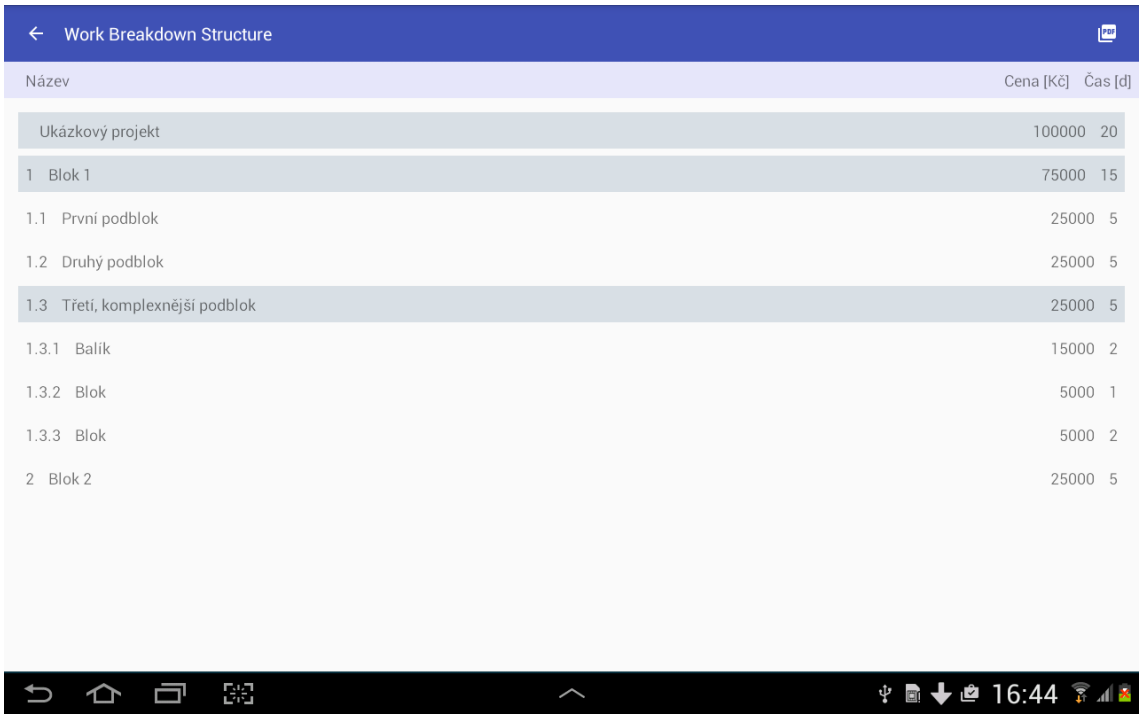
Obrázek 24: Ekvivalentní stromová struktura zobrazená jako větvený strom (horní) a seznam se zanořováním úrovní (spodní). Zdroj: *vlastní zpracování*

Algoritmus využívá rekurze, má tedy velmi úsporný zdrojový kód a rychlé zpracování. Problémy, vyplývající z omezení reálných zařízení co do rozsahu paměti, by na funkčnost neměla mít vliv, protože platforma Android má dobře zvládnutý management zdrojů a paměť tak je uvolňována podle potřeby. Ke ztrátě dat nedochází, protože tato jsou v případě potřeby odkládána do úložiště zařízení. Zároveň, i přes omezený výkon přenosných zařízení, při současných možnostech výpočetní techniky je rozsah operace, kde se problémy zapříčiněné rekurzivitou projeví, natolik velký, že je toto omezení prakticky bezpředmětné.

```
vykresli stromovou strukturu {
        najdi kořen stromu pro příslušný projekt
        vykresli uzel (kořen)
}

vykresli uzel ( uzel x ) {
        vykresli layout uzlu
        pro každý následující uzel y: vykresli uzel (y)
        nastav hodnoty podílu práce, ceny
        vrať výsledné hodnoty práce, ceny volající funkci
}
```

Obrázek 25: Pseudokód algoritmu pro vykreslení stromové struktury WBS. Zdroj: vlastní zpracování



Obrázek 26: WBS v aplikaci. Zdroj: vlastní zpracování

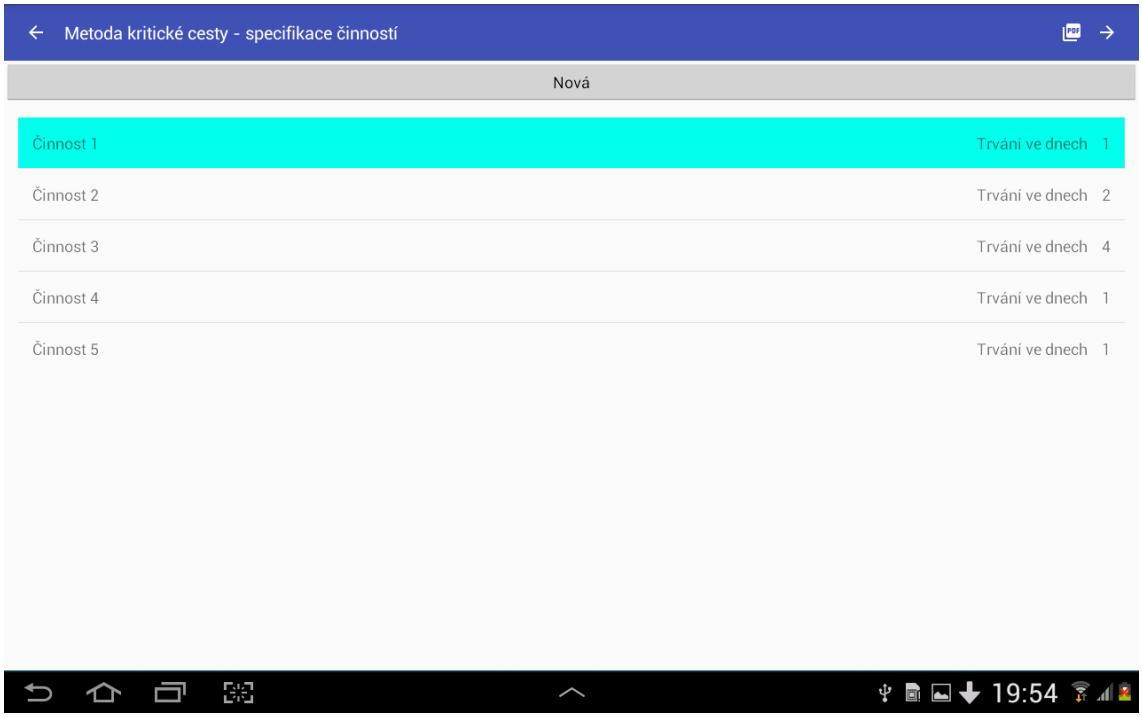
Obrázek 26 zobrazuje screenshot z WBS ukázkového projektu. Bylo zde využito pozadí k vizuálnímu odlišení položek, jejichž hodnoty cena a čas jsou počítány z hodnot jejich potomků.

Vkládání nových položek je iniciováno dotykem nadřazené položky. Nová položka je vložena implicitně jako koncová, má tedy možnost zadání hodnot ceny a trvání. Tyto hodnoty je možné nechat volné, v tom případě budou interpretovány jako 0. Pokud položka má vytvořenu alespoň jednu podpoložku, jsou tyto hodnoty vypočítány jako suma podpoložek a uloženy v rámci vykreslování stromu.

4.2.6 Kritická cesta

První dva kroky výpočtu kritické cesty, tedy přípravu a stanovení logických vazeb, není možné dost dobře automatizovat, protože vychází ze zkušenosti autora, respektive týmu. Tyto dvě fáze jsou tak v aplikaci zastoupeny jako vstup, kam jsou uživatelem vloženy jednotlivé položky a vazby mezi nimi. Tyto dvě fáze mohou probíhat do značné míry souběžně, proto je při tvorbě nové položky možné z již existujících položek vybrat přímé předchůdce. Samozřejmě tyto vazby se mohou v průběhu tvorby měnit a

doplňovat, je tedy možné již existující položky upravovat. Vazby na následující uzly aplikace doplňuje sama. Pro vytvoření nové položky je nutné zadat její popis a dobu trvání. Tato doba trvání je implicitně ve dnech, nicméně implementována je jako násobek milisekund, je tedy možné ji bez problému nastavit na jakékoli časové úseky. V průběhu projektu je pak možné po doteku položky, reprezentující činnost, tuto činnost označit za dokončenou pro potřeby sledování průběhu projektu.



Obrázek 27: Seznam činností pro Metodu kritické cesty s jednou dokončenou činností. Zdroj: vlastní zpracování

Při výpočtu kritické cesty se softwarovou podporou nelze vycházet ze stejných předpokladů jako u výpočtu manuálního. Jedná se především o fakt, že počítač postrádá přehled vlastní každému člověku. Je tak nutné přesně definovat kritéria každé operace, což je v případě návodu pro manuální výpočet zbytečné. Pokud však tato kritéria jsou jednou vytvořena, je možné dosáhnout výrazně vyšší efektivity a rozšířit funkcionalitu výrazně za možnosti lidského mozku. Tohoto faktu bývá využíváno i u výpočtu kritické cesty. V rámci implementace pro společnost *Medtronic* (16) byl vytvořen algoritmus, umožňující snadnou rozšiřitelnost a vysokou efektivitu zpracování. Pseudokód algoritmu je zde uložen jako Obrázek 28.

Tento algoritmus má však jednu zásadní vadu – využívá vlastností programovacího jazyka, který na platformě Android není podporován. Bylo by sice možné chybějící funkce dopsat a algoritmus implementovat i přes odlišné prostředí, zpracování by však bylo velmi náročné na paměť a celkově relativně málo efektivní. Proto pro potřeby tohoto projektu byl vytvořen algoritmus jiný, vycházející z manuálního zpracování. Tento algoritmus prakticky využívá čtyř průchodů oproti standardním třem. V novém průchodu, přidaném na první místo, je jednosměrně vázaný seznam uzlů doplněn na seznam vázaný dvousměrně. Tím, že jsou vazby doplňovány až v okamžiku zpracování, je zajištěno, že tyto vazby budou stoprocentně aktuální. Odpadá tak nutnost kontrolovat, zda nebyla návaznost porušena například odstraněním nějakého uzlu, čímž se zároveň výrazně zjednoduší funkce pro přidávání a odebírání jednotlivých uzlů.

Pro vizualizaci kritické cesty byl zvolen Ganttův diagram místo tradičního síťového grafu. Ganttův diagram pro svou názornost nutně nevyžaduje vizualizaci vazeb mezi jednotlivými činnostmi, což je hlavní důvod této volby – zobrazit síťový graf na platformě Android je velmi komplikovaný proces, jeho zobrazení tak, aby se jednotlivé linky znázorňující vazby pokud možno neprotínaly, je pak téměř nemožné. Ganttův diagram také lépe reflektuje aspekt času, který je podstatnou součástí metody kritické cesty.

Výsledný graf je implementován jako dvousměrně posunovatelný *ListView,* kde každý řádek představuje jednu činnost. Činnosti jsou pro účely tohoto seznamu řazeny podle nutného začátku od první po nejpozdější. Každý řádek obsahuje identifikaci činnosti, blok odsazení, blok úvodní rezervy, blok nutné činnosti a blok zadního odsazení. Součet šířek těchto prvků je pro všechny činnosti stejný, zobrazení tak probíhá v měřítku. Blok identifikace je vždy stejně široký a slouží k orientaci, o kterou položku se jedná. Další bloky jsou prázdné, oddělené barvou pozadí. Oba bloky odsazení mají pozadí bílé a reflektují období, kdy není možné činnost provádět. Blok rezervy reflektuje rezervu činnosti, pozadí je světlejší variantou barvy následujícího bloku. Blok nutné činnosti reflektuje období mezi nutným začátkem a nutným koncem činnosti. Jeho barva je nastavena na zelenou (běžná nezpožděná činnost), modrou (nezpožděná činnost ležící na kritické cestě) nebo červenou (zpožděná činnost).

Let *n* be the number of tasks,

i and *j* be any one of the tasks,

source be the beginning node of the network,

S be a set of the unchecked tasks,

d[i] be the distance at any *i* (where 'distance' is the longest amount of time to arrive at *i*), *pred[i]* be the predecessor of *i*,

A[i] be a set of all arcs emanating from *i*,

(i, j) be an arc from *i* to *j*,

time[(i, j)] be the time on the arc (i, j), and

Update(i) be a function that takes as a parameter a task.

S := {1, 2, … , n}

d[i] := -1 for all i ≠ source

d[source] := 0

pred[i] := -1

while |S| < n, do

let *i* be an element of *S* which satisfies *d[i] = max*(*d[j]* such that *j* is an element of *S*)

S := S – {i}

Update(i)

Update(i)

for each (i, j) in A[i] do

if d[j] < d[i] + time[(i, j)] then

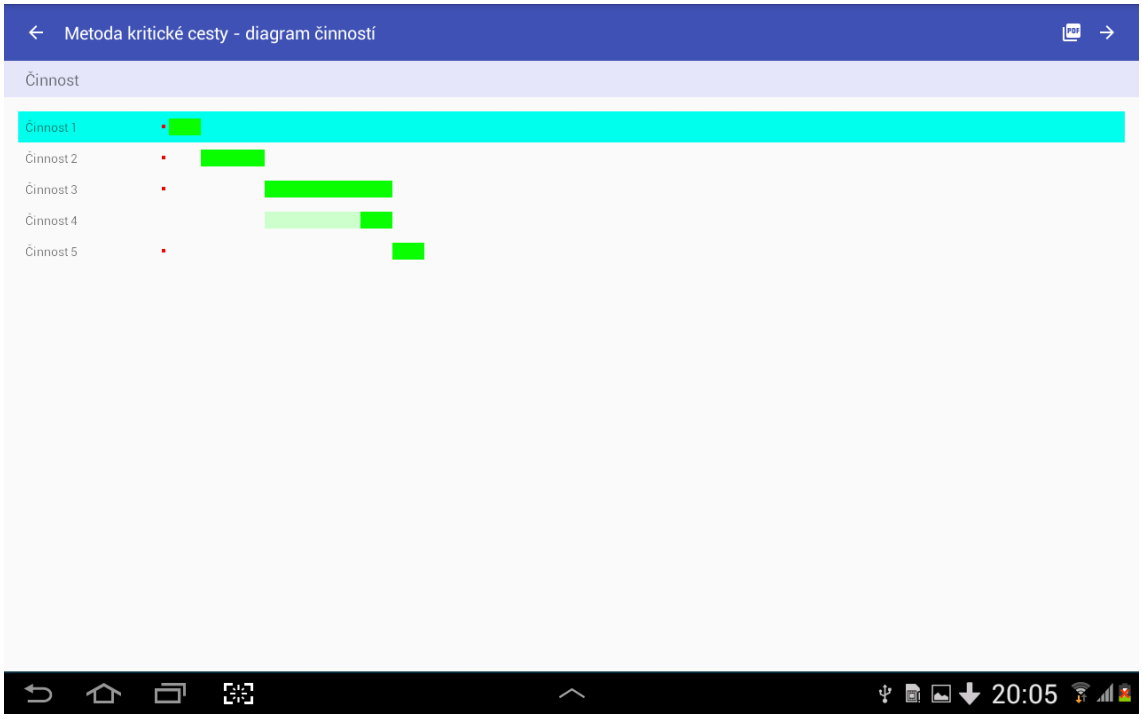
d[j] = d[i] + time[(i, j)] and

pred[j] = i

Update(j)

Obrázek 28: Pseudokód algoritmu výpočtu kritické cesty vytvořený pro společnost Medtronic. Zdroj: (16)

Přepočítání kritické cesty se provádí při každém otevření příslušné aktivity, vizualizovaná cesta je tak aktuální i v případě, kdy zpožděním dojde k její změně. Funkce zpoždění je dostupná pouze v případě, že projekt má nastavený pevný začátek nebo konec – od tohoto data se pak odvíjí veškeré výpočty. Při výpočtu kritické cesty proběhne výpočet standardním způsobem a výsledek je následně posunut tak, aby zvolený pevný bod odpovídal začátku nebo konci vypočítané kritické cesty. V případě že je zvolen pevný konec pak proběhne kontrola, zda nutný začátek projektu neleží v minulosti – v takovém případě je zobrazeno hlášení o nemožnosti dokončení projektu. Tato kontrola se pak provádí vždy při vykreslení kritické cesty nebo při dotazu ze strany uživatele. V případě, že je projektu poskytnut takovýto časový rámec, je pak přes Ganttův diagram promítnuta linka reprezentující aktuální datum.



Obrázek 29: Vizualizace diagramu činností bez určeného časového rámce. Zdroj: vlastní zpracování

Obrázek 29 zobrazuje diagram jednoduchého projektu, odpovídající dříve zobrazenému seznamu (Obrázek 27). Podbarvené pozadí indikuje dokončenou činnost, červené body indikují, že příslušná činnost leží na kritické cestě. V případě, že by projekt měl přiřazený časový rámec, by v řádku s popisem byla zobrazena data a přes obrazovku by byla promítnuta svislá čára reprezentující aktuální datum.

4.2.7 RIPRAN

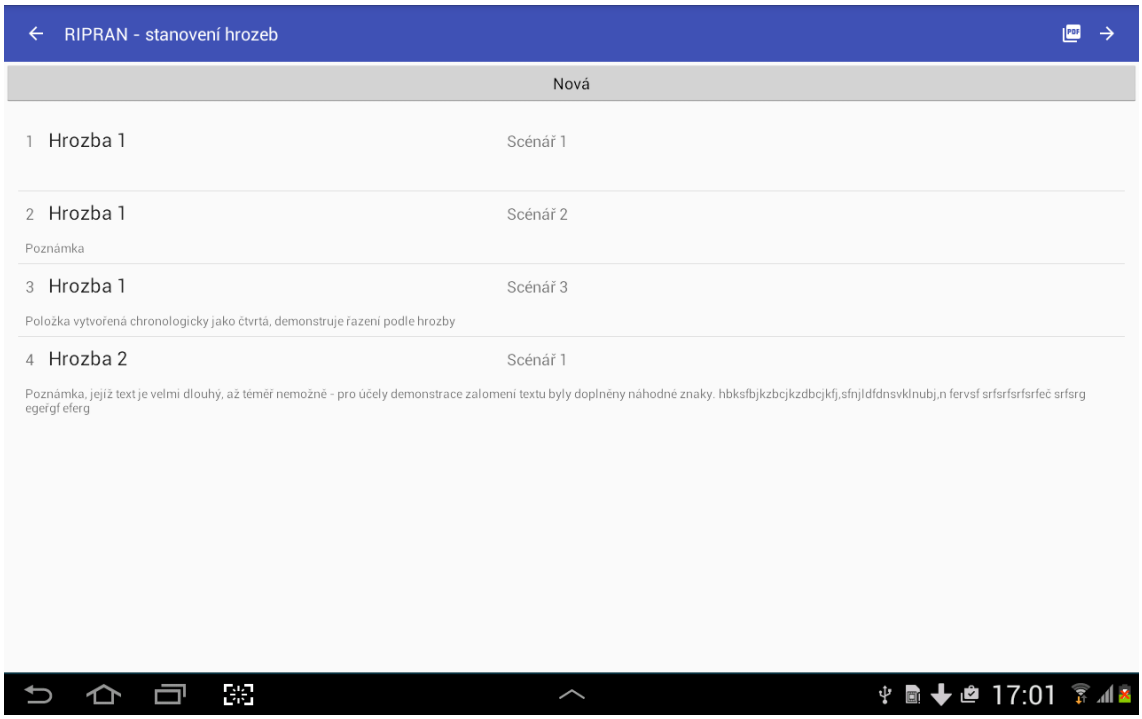
Podpora pro metodu RIPRAN spočívá především v automatizaci zpracování formulářů, je proto relativně jednoduchá. V samostatných aktivitách jsou zobrazeny jednotlivé formuláře z Obrázek 17, Obrázek 18 a Obrázek 19.

V první aktivitě probíhá přidávání jednotlivých hrozeb a scénářů. Tyto lze vytvořit buď jako nové, nebo jako varianty již existujících. Vytvoření nové hrozby znamená zadání popisu hrozby, scénáře a volitelně poznámky. Variantní vytvoření umožňuje použít již existující pár hrozba – scénář a vytvořit nový na podobném základě. Prakticky lze tuto možnost využít zejména při popisu hrozby s mnoha scénáři, kdy díky tomuto přístupu není nutné opakovaně zadávat popis hrozby. Formulář je implementován jako *ListView*, kde dotekem jednotlivých položek je vyvolána možnost úpravy, mazání a uložení nové hrozby. Dialog pro popis hrozby bez existujícího základu je volán samostatným tlačítkem. Pro vytvoření nové hrozby je minimem zadání jejího popisu, hrozba bez popisu není uložena.

Při vytvoření nové hrozby dochází k seřazení seznamu podle názvu hrozby. Tato funkce umožňuje zobrazovat scénáře stejných hrozeb pod sebou i v případě, že mezi jejich vytvořením proběhly další operace. Zároveň je nastaveno pořadové číslo hrozeb tak, aby toto pořadí bylo zachováno i v dalších formulářích.

Druhá aktivita obsluhuje formulář hodnocení hrozeb. Zde j dotykem vyvolán dialog pro ohodnocení konkrétní hrozby, založený na formuláři Formulář k ocenění rizik pro stanovený rizikový faktor. Je možné zadat hodnocení od všech, nebo pouze vybraných, členů projektového týmu.

Ve třetí aktivitě lze stejným způsobem vyvolat dialog pro popis návrhu řešení hrozby. Do pole „zodpovědnost“ je zde možné zadat libovolný textový řetězec, čímž je umožněno určit za zodpovědnou osobu i osobu bez existujícího záznamu v databázi programu.



Obrázek 30: Stanovení hrozeb v metodě RIPRAN. Zdroj: vlastní zpracování

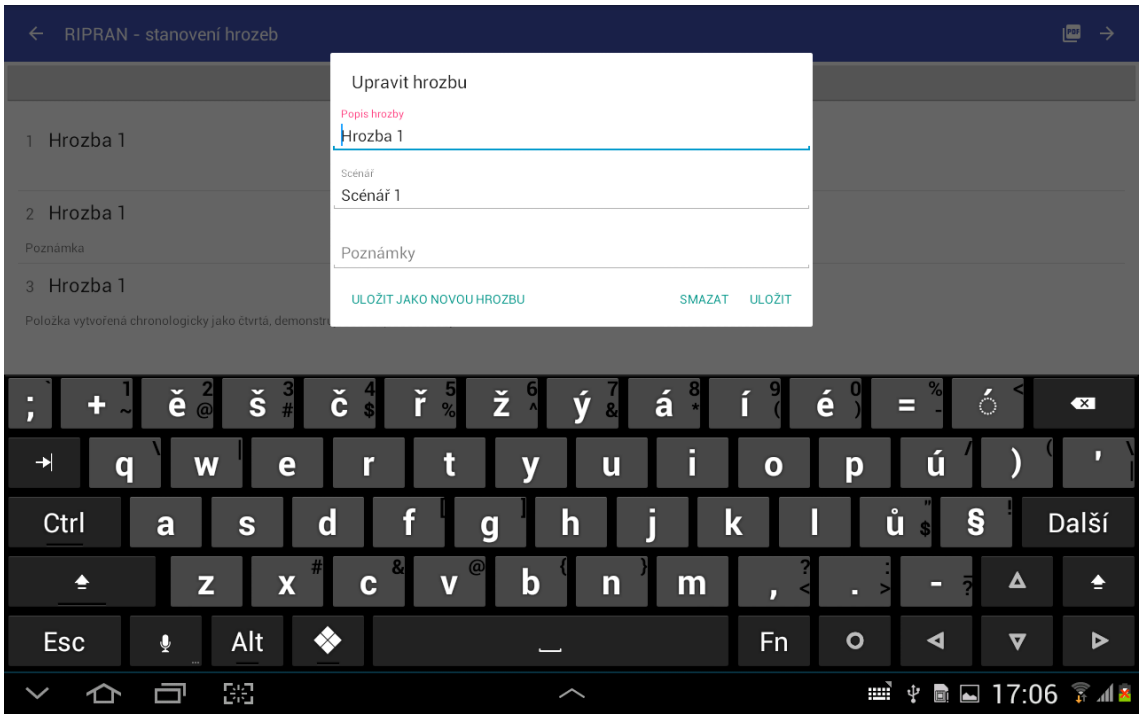
Obrázek 30 zobrazuje formulář stanovení hrozeb v metodě RIPRAN. Třetí scénář hrozby *Hrozba 1* byl chronologicky vytvořen jako poslední položka, systém jej automaticky přesunul k ostatním scénářům *Hrozba 1*[18].

Tvorba hrozeb probíhá buď klikem na tlačítko Nová, nebo dotekem existující hrozby. Obrázek 31 Zobrazuje formulář tvorby hrozby podle existující předlohy, tedy druhého případu. V první případě je zobrazeno pouze tlačítko *Uložit*. Tento dialog také umožňuje upravit nebo smazat příslušnou hrozbu.

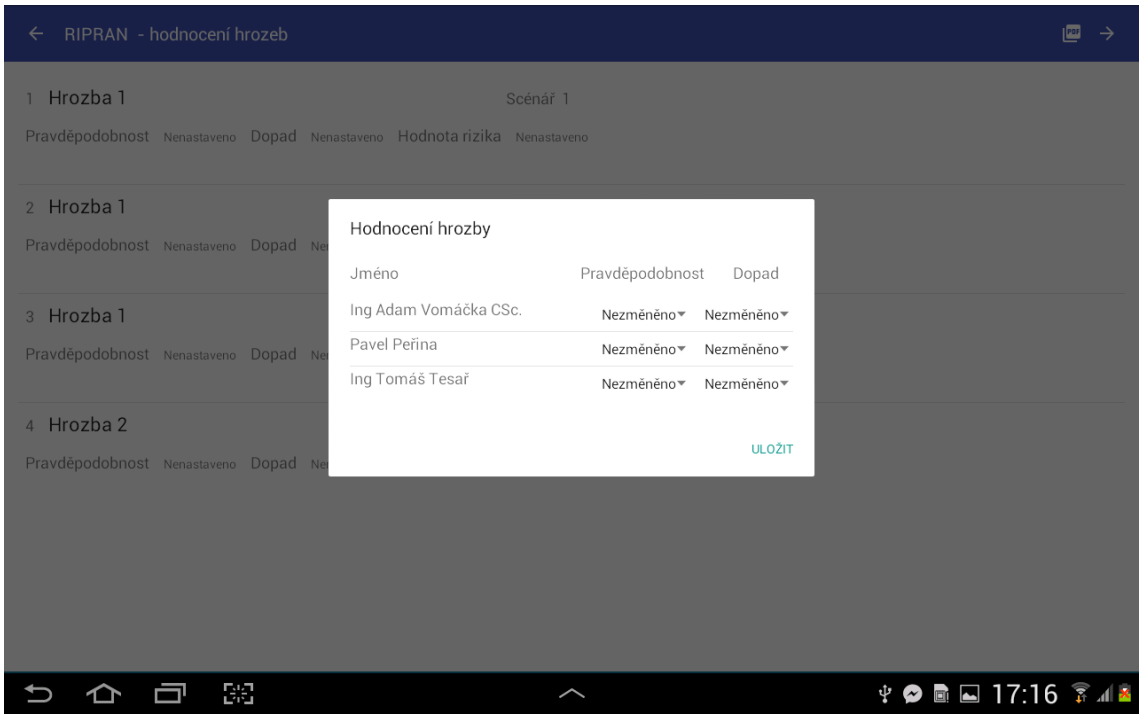
Obrázek 32 zobrazuje formulář hodnocení hrozby, vyvolaný dotekem konkrétní hrozby. Hodnocení může nabývat hodnot podle pětistupňové tabulky hodnocení rizik, přičemž význam konkrétních hodnot je ponechán na řešitelském týmu. Seznam členů

[18] Vzhledem k velikosti tištěného dokumentu jsou poznámky u jednotlivých hrozeb poměrně špatně čitelné, na reálném zařízení je velikost písma zcela dostačující.

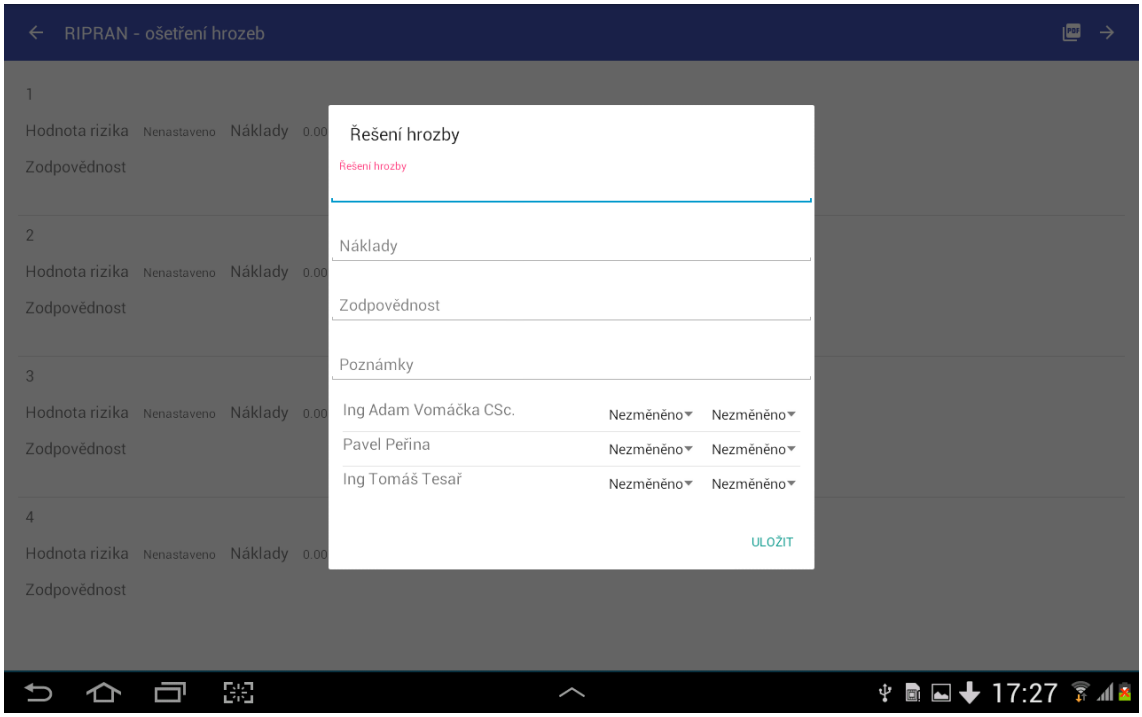
týmu je generován automaticky podle členů přiřazených ke konkrétnímu projektu. Hodnota *Nezměněno* reprezentuje, že je použita původní hodnota, v případě, že se jedná o první hodnocení příslušné hrozby touto osobou, tato hodnota značí „bez hodnocení“ a dále není zahrnuta do výpočtů. Výsledné hodnoty, zanesené do tabulky viditelné na pozadí, jsou průměrem zadaných hodnot.



Obrázek 31: Tvorba nové hrozby podle existujícího vzoru. Zdroj: vlastní zpracování



Obrázek 32: Hodnocení hrozby. Zdroj: vlastní zpracování



Obrázek 33: Ošetření hrozby. Zdroj: vlastní zpracování

Obrázek 33 zobrazuje dialog pro návrh opatření k dané hrozbě. V pozadí je vidět seznam hrozeb, kam jsou tato opatření zaznamenána. Popis opatření je zobrazen vedle čísla hrozby, implicitně je zobrazeno prázdné pole – žádný návrh.

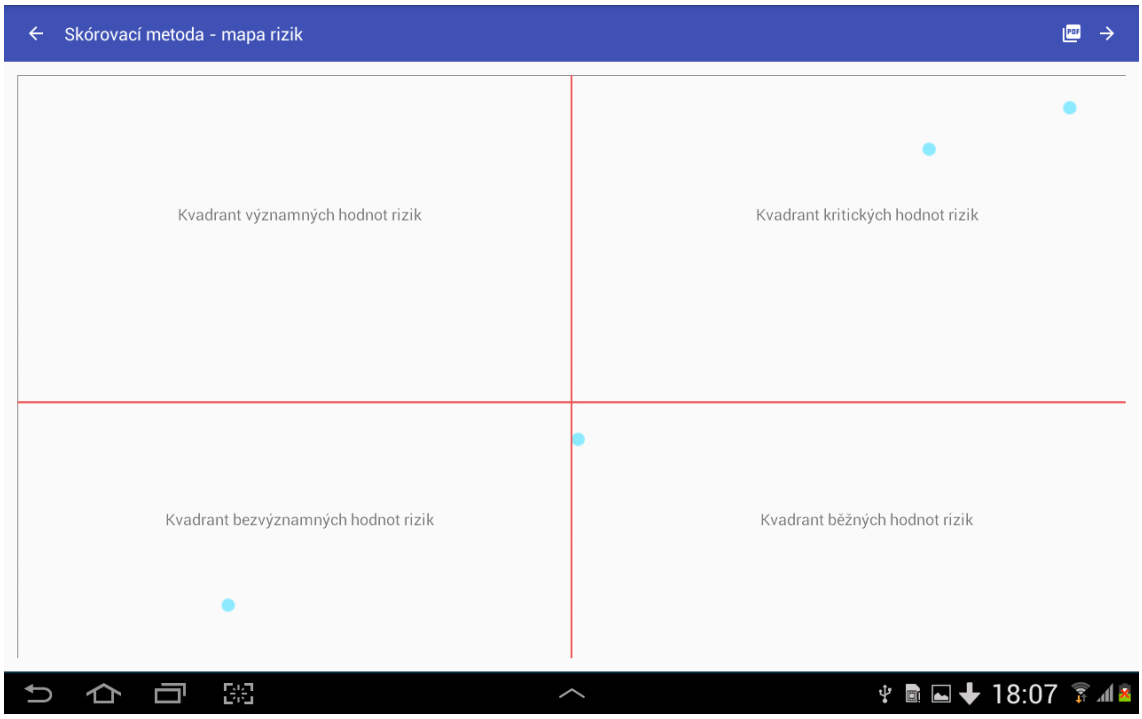
4.2.8 Skórovací metoda

Skórovací metoda má podobnou implementaci jako metoda RIPRAN popsaná v sekci 4.2.7. Oproti ní jsou změněny vstupní formuláře tak, aby odpovídaly návrhu z kapitoly 4.1.7, principiálně jsou však obdobné a nebudou zde proto zobrazeny

Zásadní odlišností je vizualizace výsledků metody. Ve třetí a páté aktivitě jsou zobrazeny mapy hodnocení rizik (Obrázek 34). Z implementačního hlediska se jedná o bodové grafy. První graf obsahuje rozložení rizik tak, jak byla stanovena, druhý graf pak po aplikaci navržených opatření.

Implementace grafů obecně je na platformě Android poměrně složitá, i přes značné množství existujících knihoven totiž dosud neexistuje plnohodnotná varianta. V této implementaci byla použita knihovna MPAndroidChart, která je v současné době pravděpodobně nejkompletnější, stále však probíhá její aktivní vývoj[19]. Zásadní nedostatek této knihovny je, že nelze zadat bod ve formátu [pozice x, pozice y], pouze [offset x, pozice y]. Offset, na rozdíl od pozice, může být pouze celé číslo a prakticky udává, v kolikátém dílku osy X má být daný bod zobrazen. Za účelem zvýšení přesnosti byla proto osa x uměle rozdělena na deset tisíc dílků a desetinné číslo, reprezentující pozici na ose X, vynásobeno tisícem. Celočíselná část tohoto výsledku pak je použita jako offset pro účely grafu.

[19] Aktivní vývoj v tomto případě znamená, že jsou průběžně doplňovány nové funkce a opravovány nalezené chyby. Jsou tak vytvářeny nové verze knihovny, což může mít za následek potíže při překladu aplikace ze zdrojových kódů – je nutno použít aktuální verzi, a, pokud došlo ke změně ovlivňující použité funkce, příslušně upravit zdrojový kód. Tato vlastnost se projevuje pouze při překladu ze zdrojového kódu, již přeložené aplikace zůstávají nedotčeny a využívají knihovnu v té formě, ve které byla v době překladu.



Obrázek 34: Mapa rizik projektu. Zdroj: vlastní zpracování

4.2.9 Možnosti dalšího vývoje

Primárním úkolem dalšího vývoje je integrace aplikace s datovým serverem. Tato operace je bohužel specifická pro každou firmu, bude tedy nutné ji provést pro každého zájemce znovu. Z hlediska uživatelské přívětivosti lze doporučit zejména modifikaci rozložení aplikace pro použití na zařízeních menších úhlopříček, dalším krokem by měla být průběžná aktualizace grafického návrhu. V případě zájmu by pak bylo možné rozšířit podporu na další platformy, tedy iOS a případně Windows Phone.

4.3 Přínosy a ekonomické zhodnocení

Hlavním přínosem práce je vytvoření nástroje, poskytujícího podporu pro projektový management ve všech fázích projektu. V kontextu mobilních zařízení je takto široká podpora vzácná, běžnější je podpora pouze pro operativní řízení. Z pohledu ZŠ Merhautova, s jejíž podporou tato práce vznikala, pak je zásadní výhodou oproti konkurenčním aplikacím výhled na snadnou integraci s budoucím serverem školy. V současné době je aplikace přizpůsobená pro činnost offline a tato funkcionalita bude zachována i po případné integraci, je tedy možné využití i v oblastech s omezeným přístupem k internetu.

Z ekonomického hlediska lze přínosy aplikace jen obtížně kvantifikovat. Jejich těžiště spočívá především v úspoře času a zpřehlednění procesů vůči ručnímu zpracování. Proti v současnosti běžnému strojovému zpracování pak aplikace přináší zejména výhody spojené s přenosností výpočetního zařízení, její využití tak lze očekávat hlavně v méně formálním prostředí, případně v kontextu projektů probíhajících mimo kancelář.

Jediným počitatelným nákladem na vývoj aplikace byl čas vývojem strávený. Tento čas lze, se započítáním úvodní analýzy, návrhu a samotného programování, odhadnout na cca 240 hodin. Tato hodnota je relativně vysoká, což je způsobeno zejména faktem, že se celý vývoj časově překrýval s akademickým rokem a běžným pracovním vytížením, bylo tedy nutné myšlenkově přepínat mezi značným množstvím aktivit. Veškeré využité nástroje jsou (v době vzniku aplikace) dostupné zdarma nebo pod licencí open source. Při uvedení na otevřený trh by pak bylo nutné počítat s náklady na prezentaci a propagaci aplikace.

Aplikace je vyvíjena s úmyslem šíření zdarma. Tato politika umožňuje maximalizaci uživatelské základny a tím šíření dobrého jména autora, což je samo o sobě přínosem. Druhotným jevem je pak zlepšení vyjednávací pozice při jednání o dalších projektech. Monetizace formou reklamních panelů je sice v oblasti software šířeného zdarma běžná, nicméně autor ji shledává značně obtěžující a proto nebyla uplatněna. Model šíření zdarma však nevylučuje monetizaci aplikace formou úprav na přání, tedy úprava aplikace – například pro spolupráci s konkrétním firemním serverem – může být pro zadavatele zpoplatněna. Konkrétně tato úprava by vzhledem k návrhu

aplikace neměla zabrat více než 20 hodin včetně testování, v případě silně odlišného databázového návrhu serveru by však tato hodnota mohla být vyšší. Další změny v aplikaci, například podpora jiné metody projektového managementu, pak může být časově – a tedy i finančně – náročnější, zejména na základě nutných úprav datového modelu, obecně by však mělo být možné ji zvládnout v horizontu do 80 hodin včetně analýzy.

5 Závěr

Hlavním cílem této diplomové práce bylo navrhnout a realizovat nástroj pro podporu projektového managementu pro tablety s operačním systémem Android.

V první části byly shrnuty teoretické základy nutné pro vývoj aplikace na platformě Android a teoretické základy metod projektového managementu, pro které aplikace poskytuje podporu. Dále byla analyzována současná situace zadávající organizace a zmíněny existující programy a navržena samotná aplikace. Poslední kapitola se pak zabývá návrhem a implementací konkrétních prvků aplikace.

Aplikace byla navržena tak, aby poskytovala podporu projektového managementu ve všech fázích projektu, funkcionalita typická spíše pro desktopové a serverové aplikace. Z důvodu absence vhodného aplikačního serveru v době tvorby byla navržena pro práci offline s tím, že spolupráci se serverem je možné doplnit úpravou jedné třídy.

Návrh aplikace, vznikající v rámci práce, byla průběžně konzultována s projektovým vedením ZŠ Merhautova a návrh upravován podle zpětné vazby. Funkčnost aplikace pak bude v rámci oponentury předvedena oponentovi práce, osobě nezúčastněné ve vývoji a proto nezaujaté.

Ukázalo se, že vzhledem k rychlosti vývoje mobilních technologií existuje poměrně málo aktuálních tištěných publikací, v příslušných částech proto byly použity zejména digitální zdroje. V tomto oboru však právě digitální zdroje obsahují nejrelevantnější informace a aktuálně doporučované postupy. V oblasti projektového managementu, tedy obsahu vyvíjené aplikace, byla využita potřebná literatura, zejména pak (1), a (18), v případě metody kritické cesty pak byla velmi inspirativní zpráva (16).

Jako autor doufám, že má práce usnadní činnost projektovým manažerům, kteří nechtějí vést své projekty přikovaní k PC. Zároveň věřím, že vzhledem k nabízené funkcionalitě a zvolenému obchodnímu modelu je tento cíl splnitelný a má práce najde své využití v běžné praxi.

6 Seznam použité literatury

Knižní zdroje

1. DOLEŽAL, J. a kol. *Projektový management podle IPMA.* Praha: Grada Publishing, 2009. 512 s. ISBN 978-80-247-2848-3.
2. ONDRÁK, Viktor, Petr Sedlák, Vladimír Mazálek. *Problematika ISMS v manažerské informatice.* Brno: Akademické nakladatelství CERM, 2013. ISBN 978-80-7204-872-4.
3. JEŽKOVÁ, Z. *Projektové řízení: jak zvládnout projekty.* Kuřim: Akademické centrum studentských aktivit, 2013. 381 s. ISBN 978-80-905297-1-7.
4. ALLEN, G. *Android 4: Průvodce programováním mobilních aplikací.* Computer Press, 2013. ISBN 9788025137826.

Internetové zdroje

5. Richta, K. *Metodiky vývoje software, MDA*. Přednáška. Praha:ČVUT, 2011.
6. #1 Free CRM for Small Business. *Insightly.* [online]. [2016] [cit. 2016-05-15]. Dostupné z: *http://www.insightly.com*
7. Try Wrike – Project Management Software. *Wrike.* [online]. [2016] [cit. 2016-05-15]. Dostupné z: *http://try.wrike.com/project-management-getapp/?utm_source=getapp&utm_medium=cpc&utm_campaign=listing_project_management*
8. Android Developers. *Android Developers.* [online]. [2016] [cit. 2016-05-15]. Dostupné z: *http://developer.android.com/index.html*
9. IBM MobileFirst Platform Foundation. *IBM.* [online]. [2016] [cit. 2016-05-15]. Dostupné z: *http://www-03.ibm.com/software/products/cs/mobilefirstfoundation*
10. Samsung Newsroom. *Samsung Electronics Co,.Ltd.*. [online]. [2016] [cit. 2016-05-15]. Dostupné z: *https://news.samsung.com/global/*
11. Android Studio Overview. *Android Developers.* [online]. [2016] [cit. 2016-05-15]. Dostupné z: *http://developer.android.com/tools/studio/index.html*
12. TN Active Directory. *Microsoft Technet.* [online]. [2016] [cit. 2016-05-15]. Dostupné z: *https://technet.microsoft.com/en-us/library/cc977985.aspx*

13. 2-plan Project Management Systems. *2-plan Management Software.* [online]. [2015] [cit. 2016-05-15]. Dostupné z: *http://2-plan.com/*
14. ČESKO. Zákon č. 181/2014 Sb. ze dne 23. července 2014 o kybernetické bezpečnosti a o změně souvisejících zákonů (zákon o kybernetické bezpečnosti). [on-line]. [2014] [cit. 2016-05-15]. Dostupné z: *https://www.nbu.cz/download/nodeid-1384/*
15. Sugar ORM – Insanely easy way to work with Android database. *Sugar ORM.* [online]. 2016 [cit. 2016-05-15]. Dostupné z: *http://satyan.github.io/sugar/*
16. FLAHERTY, C. a kol. *Critical Path Networks – Final Report.* [online]. [2005] [cit. 2016-05-15]. Dostupné z: http://eecs.mines.edu/Courses/csci370/FS2005/Medtronic.doc
17. Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015. *Gartner.* [online]. 18.2.2016 [cit. 2016-05-15]. Dostupné z: *http://www.gartner.com/newsroom/id/3215217*
18. RIPRAN – Metoda pro analýzu projektových rizik. *RIPRAN.* [online]. [2014] [cit. 2016-05-15]. Dostupné z: *http://ripran.cz/*

Seznam obrázků a tabulek