Mendelova univerzita v Brně
Provozně ekonomická fakulta

Návrh metody pro synchronizaci dat mezi Android zařízeními

Diplomová práce

Vedoucí práce:
Ing. Jan Kolomazník, Ph. D.

Bc. Roman Valeš

Brno 2016

Na tomto místě bych rád poděkoval vedoucímu mé diplomové práce Ing. Janu Kolomazníkovi, Ph.D. za pomoc, ochotu a cenné rady, kterými přispěl k vypracování této práce.

Velké poděkování patří i mé rodině, která mi poskytla zázemí a podporu pro studium na Mendelově univerzitě.

Čestné prohlášení

Prohlašuji, že jsem práci: **Návrh metody pro synchronizaci dat mezi Android zařízeními,** vypracoval samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací*.

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 4. ledna 2016 _____

Abstract

Valeš, R. *Design of a method for data synchronization between Android devices*. Master thesis. Brno: Mendel University in Brno, 2016.
In this paper are examined methods, architectures and development, dealing with issues of data synchronization.

Moreover this paper contains scenarios of the most frequent utilizations, and by analyzing one of the scenarios a general design solving logical problems of data synchronization is created. For the designed solution was created an Android application, which is used as an instrument for presenting created methods, providing processes for data synchronization.

Keywords

Android, data synchronization, cloud, java, programming, application programming interface

Abstrakt

Valeš, R. *Návrh metody pro synchronizaci dat mezi Android zařízeními*. Diplomová práce. Brno: Mendelova univerzita v Brně, 2016.

V této práci jsou zmapovány způsoby, architektury a vývoj zabývající se problematikou synchronizace dat.

Dále jsou zpracovány scénáře nejčastějšího použití a pomocí rozebrání jednoho ze scénářů, je vytvořen obecný návrh, řešící logické problémy vázané se synchronizací dat. Pro navrhnuté řešení je vypracována Android aplikace, Tato aplikace slouží jako prostředek pro prezenci vytvořené metody zajišťující procesy pro synchronizaci dat.

Klíčová slova

Android, synchronizace dat, cloud, java, programování, aplikační programové rozhraní

Obsah

Seznam obrázků

Seznam tabulek

Seznam kódů

1 Úvod a cíl práce

1.1 Úvod

Počítače, výpočetní systémy a komunikační sítě se mění takovým způsobem, jakým se mění potřeby lidí. Od počátečních pokusů, poslat zprávu mezi dvě stanice, se vývoj posunul až k vytvoření globální komunikační sítě internet, která umožňuje s kýmkoliv odkudkoliv komunikovat přes celý svět. To umožňuje uživatelům sdílet různá data a to ať už se jedná o dokumenty, datové soubory nebo multimediální soubory.

Díky vysokým rychlostem je možné vytvářet webové aplikace, které zpracovávají data, kde je jejich výstupem informace, kterou uživatel vyžaduje a to nejčastěji v reálném čase (myšleno v rámci pár vteřin). To přispělo i k rozvinutí mobilních zařízení. Mobilní telefony se postupně rozvíjely a ze zásadně telekomunikačních zařízení se proměnily v multifunkční plnohodnotná zařízení podobně využitelná jako počítač.

Nejen tyto body vývoje vytvořily spoustu různorodých problematik v informačních technologiích. Jedna z těchto problematik je správa a ukládání dat v celosvětovém měřítku. Jelikož nyní jsou data distribuována celosvětově prostřednictvím internetu, vznikají online služby, kde jsou data ukládána. Většinou se jedná o data velice specifická (data pro jednu konkrétní firmu, pro zpravodajské služby, pro osobní účely jednotlivce, atd.).

Na tuto problematiku se váže už velice konkrétní úloha a to synchronizace dat. Není neobvyklé, že jsou lidé dennodenně v interakci s více chytrými zařízeními a to konkrétně s počítačem/notebookem, chytrým telefonem a tabletem. Jelikož každé zařízení má vlastní lokální úložiště dat, nastává zde často problém nedostupnosti aktuálního souboru na jiném zařízení, než byl původně zpracováván. Právě tuto problematiku řeší synchronizace dat. Cílem je zajistit, aby data byla aktualizována a dostupná pro všechna zařízení nebo pro všechny uživatele, které k těmto datům mají mít přístup.

1.2 Cíl práce

Cílem diplomové práce je navrhnutí metody pro řešení synchronizace dat mezi zařízeními se systémem Android. Pomocí této metody vytvořit konkrétní návrh pro řešení Android aplikace, která zajistí synchronizaci dat. Pomocí tohoto návrhu naimplementovat Android aplikaci.

K dosažení cíle je nutné popsat problematiku synchronizace dat, popsat jak se vyvíjela architektura a které technologie lze použít. Na základě analýzy vytvořit scénáře, které určí případy, kdy je vhodné synchronizaci dat aplikovat. Tyto scénáře určit i pro konkrétní možnosti na systém Android.

Pro vlastní práci vybrat konkrétní scénář pro synchronizaci dat. V návrhu metody pro synchronizaci dat určit technologie a detekovat konflikty, které mo-

hou vzniknout při synchronizaci dat. Vytvořit Android aplikaci pro prezenci navrhnutého řešení. Vyhodnotit řešení z hlediska spolehlivosti, latence a zátěže na zařízení.

2 Literární přehled

Literární rešerše vede k vytvoření přehledu o stavu oboru. Hlavním bodem je dohledání a uvedení dostupných informací k tématu diplomové práce. V této kapitole budou shrnuty prvky, které souvisí s pojmem synchronizace dat.

2.1 Sdílení dat

Pojem sdílení dat je v tomto případě chápán jako činnost, při které je pomocí hardwarových a softwarových prostředků zajištěno jejich kopírování nebo přesouvání do dalších zařízení. Tato akce vychází z faktu, že data, jako každá jiná komodita, uspokojující některou z potřeb lidských jedinců, jsou subjektem lidského zájmu. Tento lidský zájem a potřeba se projevuje snahou člověka tuto komoditu shromažďovat. Sdílení dat je nástrojem pro uspokojování těchto potřeb.[18]

Sdílení dat je v dnešní době zprostředkováno především prostřednictvím internetu. Zde ve formě architektury klient-server klient přistupuje k webovým službám (www, e-mail, DNS, apod.) a komunikuje se serverem, od něhož požaduje data pro konkrétní aplikaci. Klient zde může být považován jako konzument dat nebo případně i jako tvůrce dat, která se následně ukládají na serveru, kde jsou dále dostupná pro jiné klienty (pokud není vyžadováno speciálního přístupu - zabezpečení). Potřebné faktory pro sdílení dat:

- **Technologie** – udává rámec, pomocí něhož je možné k požadovaným datům přistupovat a určuje jak s těmito daty zacházet. Tyto technologie jsou především koncipovány na sdílení dat prostřednictvím internetu. Především kvůli velkému rozmachu tohoto trendu a velké oblibě uživatelů. Typickým příkladem je změna chápání webových stránek v posledních letech. Nyní v roce 2015 jsou www stránky nejčastěji implementovány ve formě webových aplikací namísto statických webových stránek. Jedná se o prostý vývoj požadovaný dobou, kdy jsou technologie vyvíjeny, tak aby člověk dosáhnul co největší efektivity a pohodlnosti při práci.[1]
- **Rychlost připojení** – s rozvojem IT napříč všemi sektory se postupně zvyšuje požadovaný výkon počítačů. Na tento fakt, se dále nabalují všechny prvky, které s tím souvisí. Je potřebný větší prostor pro ukládání zpracovávaných dat, protože požadavky na ukládání informací mají větší náročnost na uložení informace. Pro sdílení dat je důležitým prvkem jaké rychlosti je možné dosáhnout mezi komunikujícími prvky. Jak bylo zmíněno, doba vyžaduje čím dál vyšší rychlosti pro připojení, aby vůbec mělo smysl vytvářet webové aplikace. Kolem roku 2000 bylo běžné domácí připojení pomocí modemu s rychlostí 56kb/s, k roku 2015 je tato rychlost v domácnostech zvýšena minimálně v řádu stovek.[1]

2.2 Synchronizace

Obecně je slovo synchronizace podle českého jazyka[2] interpretováno jako *„uvádění v časový soulad, do společného rytmu (cyklu)“*.

Z pohledu informatiky definuje Hansmann[4] synchronizaci jako: *„Proces, který zajistí, že dvě sady dat budou vypadat identicky. K dosažení tohoto cíle musí docházet ke změně dat na jedné straně na základě změny dat na straně druhé.“*.

V dnešní době je běžné, že lidé používají různé typy chytré elektroniky (pc, chytrý telefon, tablet). Nastává tedy často případ, kdy zaměstnanec v práci používá počítač co má v práci, doma používá svůj osobní počítač a na cestách využívá notebook nebo chytrý telefon popřípadě tablet. Z tohoto důvodu, je potřebné zajistit, aby data, které přístroj ukládá, nebyly pokaždé jen na lokálním úložišti pro daný přístroj, ale byly buď centralizované anebo synchronizované. Je vyžadováno, aby data byla ukládána tak, aby byla vždy k dispozici v aktuální verzi, a uživatel k nim měl přístup ze všech zařízení. Přístup je v nejčastější podobě zprostředkován pomocí internetu (přístup možný odkudkoliv na světě). Tento popsaný koncept zaštituje pojem synchronizace dat.

Stejně jako sdílení tak i synchronizace může probíhat na celé řadě přenosných médií a kanálů. Je možné udržovat soubory v rámci jednoho počítače, počítače a přenosného hardwarového úložiště, počítače a webového úložiště, dvěma webovými úložišti atd. V kontextu této práce se pod pojmem synchronizace rozumí udržení stejného obsahu dat mezi dvěma a více zařízeními.

2.3 Synchronizační postupy

Existují různé postupy pro synchronizaci dat. Jako příklad lze uvést synchronizaci dat nahráním na přenosné médium (cd, dvd a nejčastěji flash disk). Tento postup neobsahuje automatické prvky, je potřeba aby se uživatel o správu dat staral manuálně a při každé změně dat, které by chtěl mít neustále aktuální, musel data přenést na médium a toto médium by musel mít neustále u sebe. Je to postup, který se určitě v malé míře stále vyskytuje. Nejčastěji jen pro jednotlivce a jeho vlastní potřeby. Ve firmách, kde jsou zavedené automatismy a podnikové postupy je tento postup nemyslitelný z mnoha důvodů (globalizace, logistiky, efektivnosti, konkurenceschopnosti podniku, atd.).

Všechny tyto postupy se liší v architektuře a logické výstavbě. Naopak cíl je pro všechny postupy stejný a to udržením chtěných dat v aktuálním stavu na všech zařízení, kde tyto data chceme využít.

2.3.1 Klient-server architektura

Tento vynález je definován serverem v podobě počítače. Na serveru běží aplikace, jejímž účelem je synchronizovat data pro ostatní klienty. Klient je zde uživatel, který má možnost ze svého chytrého zařízení přistoupit k serveru a požadovat potřebná data. Aplikace, která běží na serveru, se liší implementací, jejím

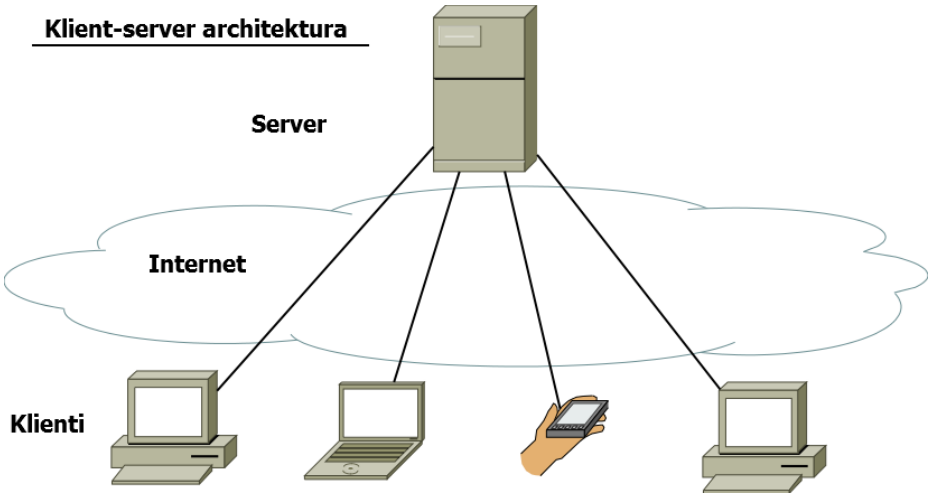
účelem je obsluhovat požadavky od klientů. Komunikační kanál mezi klientem a serverem je nejčastěji celosvětová síť internet, díky které je možné přistoupit k datům odkudkoliv z celého světa.[5]

Tato architektura se v posledních letech stala velice oblíbenou. Velký vliv na to měl rychle se rozšiřující internet. Většina komunikace je založena právě na komunikaci klienta se serverem. Nejčastějším a nejznámějším způsobem, je vyhledávání informací uživatelem na internetu. Uživatel pomocí webového prohlížeče zadá, co právě chce vyhledat. Tento požadavek se odešle na server, který tento požadavek zpracuje nad jeho dostupnou databází a odpoví klientovi výsledkem hledaného dotazu. Na podobný princip fungují i další internetové služby, které jsou zprostředkované serverem (DNS, email, atd.).[5]

S rostoucím vytížením těchto služeb rostou i požadavky na server, který se při velkém vytížení může stát nedostupným. Výhodou tohoto konceptu je, že lze buď rozšířit hardware serveru, aby zvládal vyšší nároky, anebo jen připojit více serverů a provoz tak mezi ně přerozdělit.

Synchronizování dat u služeb není povinností, ale funkcí, kterou je potřebné pro každou služby naimplementovat zvlášť. Tento problém nejčastěji řeší programátor, který vytváří až konkrétní klientskou aplikaci a všechny okolnosti a potřeby si musí vyřešit sám. Klient poté musí jen vyřešit vlastní nastavení dané aplikace pro synchronizaci dat.

V dnešní době jsou už dostupné i služby, které se přímo specializují pro správu a ukládání dat a zde už je možnost synchronizace přímo zabudována. Klient ale musí opět vyřešit vlastní nastavení (nejčastěji zadat přihlašovací údaje a nastavení složek pro automatickou synchronizaci dat).



Obr. 1 Klient-server architektura

Výhody

- Jednoduchý koncept
- Data dostupná i na serveru (třetí straně)

- Distribuce dat pro víceuživatelské systémy
- Možnost sdílení dat jiným uživatelům
- Škálovatelnost provozu (podle využití serveru případě celé služby)

Nevýhody

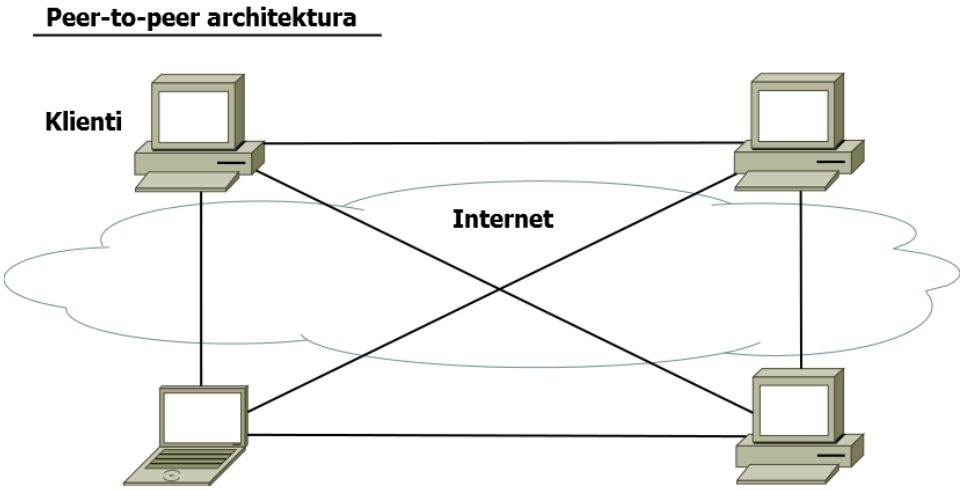
- Vyžaduje připojení k internetu (v nejlepším případě nepřetržitě)
- Závislost na poskytovateli připojení
- Závislost na dostupnosti serveru
- Nadměrná komunikace se serverem (omezeno rychlostí připojení serveru)
- Vyšší objem komunikace, vyžaduje vyšší nároky na hardware serveru (úložiště, procesor)
- Vyšší riziko při zabezpečení dat (více výskytů těchto dat)
- Nadměrná komunikace (komunikace s třetí stranou – serverem)

2.3.2 Peer-to-peer (P2P) architektura

Jedná se o vynález, který ke vzájemné komunikaci klientů nevyužívá centralizovanou službu, tedy nepoužívá server a komunikace tak probíhá přímo mezi klientskými zařízeními. Každé zařízení musí sledovat každou změnu sdílených dat, kterou provede a také musí sledovat vlastní data a jejich změnu od ostatních uživatelů. Pokud jsou dvě nebo více zařízení připraveni komunikovat, sdílí každý s každým jejich informace o změnách, které provedly a které byly provedeny na datech. Poté je možné v síti dosáhnout synchronizace. Změny na jednom zařízení jsou propagovány dále v síti, aniž by byla potřeba přímé komunikace mezi zařízeními.[6]

Tento koncept se uchytl při distribuci dat pomocí nástroje BitTorrent. Datové přenosy jsou rozkládány mezi všechny klienty, kteří data stahují. Principově je zde soubor rozdělen na menší bloky, které jsou poté na síti dostupné. Jak síť roste, tak se i zvyšuje přenosová rychlost klientů (více dostupných souborů). Zde je výhodou možného dosáhnutí maximální přenosové rychlosti klienta, kterého by klient při architektuře klient-server neměl možnost dosáhnout, protože by byl limitován ze strany serveru (jeho rychlostí připojení). Často se zde distribuují data o velkých velikostí.

Naopak pokud je síť bez dostatečného počtu klientů, může nastávat situace, že soubor není dostupný, protože zařízení kde je soubor uložen, není v danou chvíli dostupné.



Obr. 2 Peer-to-peer architektura

Výhody

- Nevyžaduje centralizované řešení (bez třetí strany – serveru)
- Přímá komunikace (omezení nadměrné komunikace)
- Při agregaci uzlů (seedů) se zvyšuje rychlost přenosu

Nevýhody

- Oproti klient-server architektuře složitější koncept
- Pokud není vlastník souboru dostupný a nikdo jiný kompletní soubor nesdílí, nelze přenos souboru dokončit do doby, než bude na síti celý soubor dostupný
- Bez aplikace není možnost jiného přístupu k souboru

2.4 Cloud computing

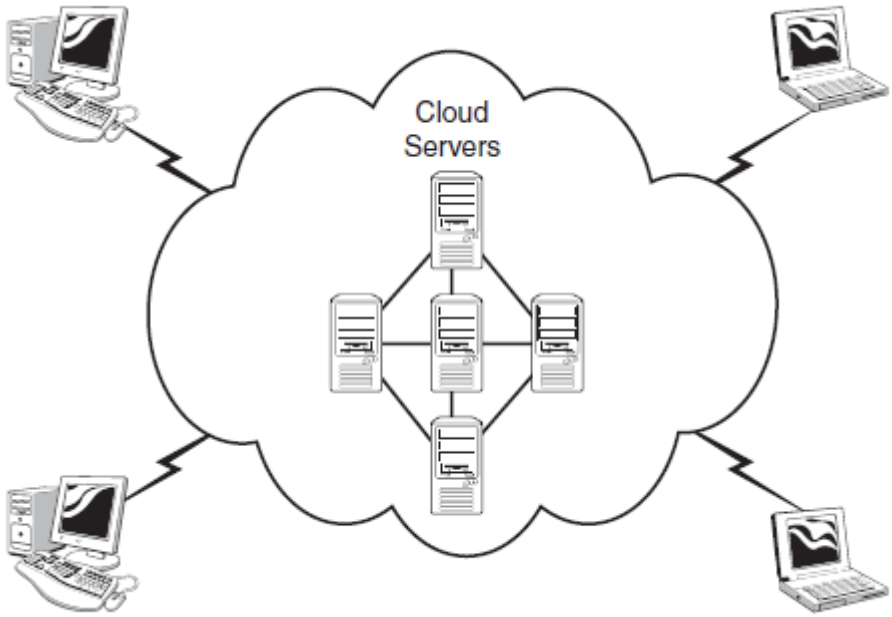
Pokud se chceme bavit o synchronizaci dat, je už nyní nemožné vynechat pojem „cloud computing“. Jedná se o relativně „mladou“ technologii, která už je ale v dnešní době velice rozšířená. Když si zadáme ve vyhledávání na google.com heslo „cloud computing“, nalezneme 170 milionu odkazů (2. polovina roku 2015).

Za předchůdce tohoto konceptu lze považovat modely klient-server a peer-to-peer komunikace. Hlavní podstatou je myšlenka centralizovaného úložiště a snadného přístupu klientů ke společným datům, která vede k lepší organizaci a efektivnosti. Další podstatnou myšlenkou je téma virtualizace, tedy způsob kdy více počítačů pracuje dohromady a zvyšuje tak výpočetní výkon.[14]

Přesnou definici nelze přesně stanovit, neboť tento pojem protíná mnoho oborů a v každém může mít odlišný význam. Definici, která se hodí pro tuto práci lze použít z knihy od Millera[14], kde je cloud computing definován jako: *„Cloud je kolekce počítačů a serverů, které jsou veřejně dostupné přes Internet. Hardware je typicky vlastněn a spravován třetí stranou (firmou) na konsoli-*

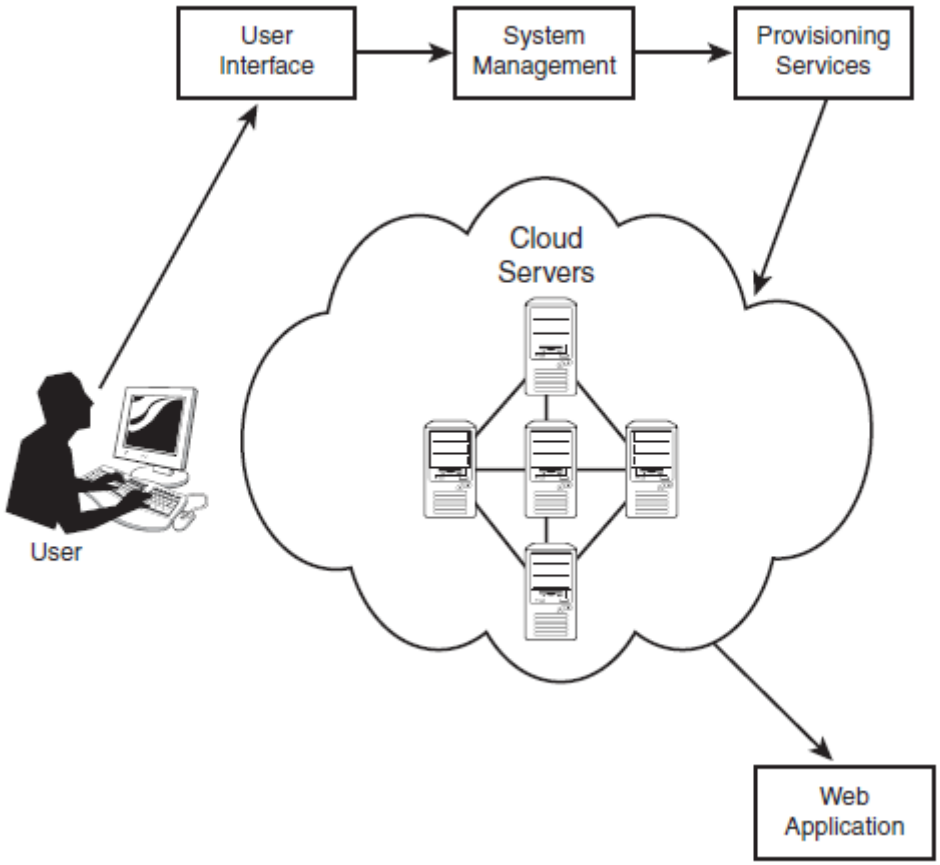
dovaném základě v jednom nebo více centralizovaných datových úložišť. Na těchto strojích může běžet jakýkoliv operační systém, na čem zde záleží, je výpočetní výkon každého stroje.“

Díky této definici si lze připojení k cloudu představit jako aplikaci (například webová aplikace), ke které se uživatel připojí přes Internet (například přes webový prohlížeč). Uživatel zde vidí jen soubory, které má dostupné, nemusí zde řešit, kde jsou tyto soubory fyzicky uložené. Hardware i operační systém na straně cloudu je pro uživatele neviditelný (nemá o něm žádné informace).[14] Tento popis lze graficky vyjádřit na Obr. 3.



Obr. 3 Připojení uživatele do cloudu[14]

Z pohledu cloudové služby se jedná o sérii akcí, které se musí vykonat, aby uživatel byl řádně obsloužen podle jeho požadavků. Je zde vyžadován určitý inteligentní management, který propojí všechny počítače do cloudu a přiřadí procesy speciálně vyžádané od konkrétního uživatele. Uživatel se připojí k aplikaci a vyžádá si akci. Na cloudové službě tento požadavek přejme System managment, který rozhodne, které zdroje budou využity k dané akci a poté zavolá Provisioning Services, který spustí příslušnou webovou službu a až poté je obsah požadavku předán uživatelovi.[14] Viz Obr. 4.



Obr. 4 Architektura z pohledu cloudové služby[14]

Podle Millera[14] jsou definovány i následné výhody a nevýhody používání cloud computingu.

Výhody cloudu

- **Nižší nároky na výkon pc** – aplikace, které běží na cloudu využívá svoje vlastní výpočetní zdroje. Na výkonu počítače uživatele zbývá pouze zobrazení informací, potřebuje tedy mnohem nižší nároky, než kdyby potřeboval vypočítávat celý proces. Klient nepotřebuje nové moderní stroje a ušetří tak výrazně na hardwaru (především jedná-li se o firmu).
- **Vylepšený výkon** – počítač uživatele nevykonává žádné složité procesy, které souvisí s používanou aplikací, všechny tyto akce se provádí na straně cloudu. Navíc na cloudu jsou tyto aplikace puštěné mnohem rychleji, protože spravují jen konkrétních pár programů a procesů načtených už v pamětích, čekající jen na požadavky ze strany uživatele.
- **Snížené náklady na infrastrukturu IT** – ve velkých firmách, lze snížit zavedením cloudu náklady IT oddělení. Není potřebné investovat obrovské částky do výkonných serverů (ve špičce dosahuje velkých nároků, ale mimo

špičku je jejich potenciál nevyužit). Výpočetní výkon zprostředkovaný pomocí cloudové služby řeší tento problém virtualizací, kde jsou zdroje přerozdělovány tak, jak je zrovna potřeba.

- **Méně IT údržby** – s menším počtem serverů a tedy i hardwaru se snižuje potřeba jejich údržby. Stejně také při využívání softwaru, který běží na cloudu, je snížena potřeba jeho údržby (troubleshooting[1], aktualizace) a je tedy potřeba spravovat jen základní náležitosti potřebné k funkčnímu klientovi (počítači).
- **Snížené náklady na software** – místo zajišťování licencí pro každý počítač ve firmě, se zajišťuje jen uživatelský přístup do cloudu. I kdyby software pro cloud byl stejně drahý jako desktopové aplikace (což většinou nebude), už zde hraje roli snížení nákladů v podobě menší potřeby údržby (nové instalace) a správy (přeinstalování, aktualizování) těchto desktopových aplikací.
- **Okamžité aktualizace** – software na cloudu je spravován jako jeden celek, a tedy jakákoliv změna se projeví okamžitě u všech uživatelů, kteří ji využívají. Například není potřebné každému uživateli na každém počítači provádět aktualizaci zvlášť (myšleno i jako upgrade[2] aplikace).
- **Zvýšený výpočetní výkon** – uživatel není limitován výpočetními zdroji. Má za sebou výpočetní sílu mnoho počítačů z cloudové služby, lze tedy dělat úkony, které by na běžném počítači nebyli ani možné.
- **Neomezená kapacita úložiště dat** – podobné jako u výkonu, nelze porovnávat možnosti běžného počítače oproti cloudovým službám, která obsahuje mnohonásobně vyšší počet harddisků.
- **Zvýšená ochrana dat** – data na cloudu jsou „někde“ uložena. Tyto data můžou fyzicky měnit svoji polohu dynamicky podle potřeby cloudu. Nezáleží, kde jsou data přesně uložena, ale záleží, že jsou neustále dostupná. Tyto data jsou i zálohována, takže jestliže se na cloudu některý z harddisků rozbije, je jeho záloha neustále dostupná. Tento fakt nelze u běžných počítačů zaručit.
- **Kompatibilita mezi operačními systémy** – uživatelé mohou přistupovat ke cloudové službě z jakéhokoliv operačního systému. V cloudu záleží na datech, ne na operačním systému.
- **Kompatibilita formátů dokumentu** – jestliže všichni uživatelé pracují ve stejné aplikaci, není možné, aby vznikala nekompatibilita dat.
- **Skupinová spolupráce** – jedna z nejdůležitějších výhod pří práci v cloudu. Sdílené dokumenty umožňují, že na jednom dokumentu může zároveň pracovat více uživatelů, změny v dokumentu jsou prováděny okamžitě a viditelně pro všechny. Jedná se o mnohem efektivnější způsob, než přeposílání dokumentu mezi spolupracovníky pomocí emailu.

---

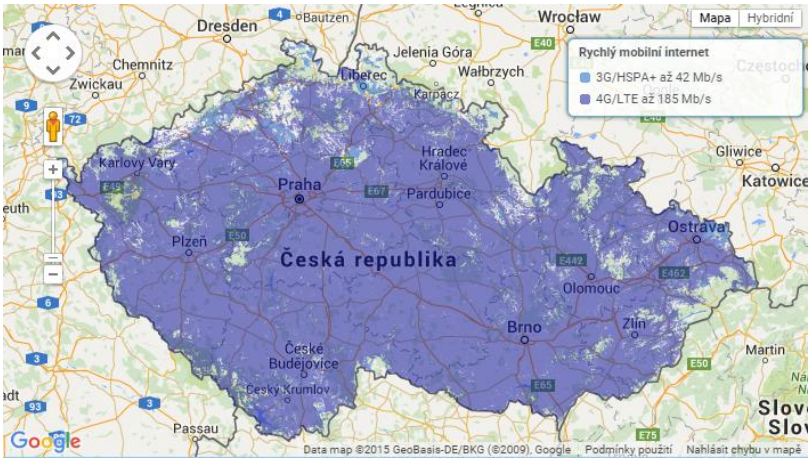
[1] Řešení IT problémů

[2] Inovace, zdokonalená verze

- **Univerzální přístup k dokumentům** – nyní je jedno odkud potřebujete data zpracovávat, díky cloudu jsou vždy přístupné v aktuální podobě prostřednictvím Internetu.
- **Aktuálnost dokumentu** – vždy je jistota, že data, s kterými pracujete, jsou právě ta, která jsou nejaktuálnější, i když je změnil někdo jiný.
- **Univerzálnost přístupu** – nyní už není důležité s jakým zařízením se ke cloudu přihlásíte, jestli jde o počítač v práci nebo o tablet na pracovní cestě, stále máte možný přístup ke cloudové službě a datům.

Nevýhody cloudu

- **Nutnost neustálého internetového připojení** – ke cloudu je nemožné se připojit, jestliže není dostupné internetové připojení. Tak jak je nemožné poslat email. Tato nevýhoda je všeobecného typu. U cloudu je tento fakt zásadního charakteru, protože nelze službu využívat ani částečně.
- **Pomalé internetové připojení** – v dnešní době jsou webové aplikace vyvíjeny s určitou mírou kvality a pohodlnosti pro uživatele. Tato kvalita vyžaduje vyšší nároky na objem dat, které jsou zpracovávány, v tomto případě záleží na rychlosti připojení k internetu. Jestliže šířka pásma nedokáže obsloužit požadované nároky, tak proces neběží optimálně (seká se nebo se zdlouhavě nahrává). Ilustrativně lze zobrazit na Obr. 5 pokrytí ČR rychlým 4G/LTE internetem od největšího poskytovatele O2 (přes 90% území ČR), které s šířkou pásma 185Mb/s bohatě dostačuje k využívání cloudových služeb.
- **Delší odezva** – i s rychlým internetem, je možné, že komunikace mezi počítačem klienta a fyzicky přiřazeným počítačem od cloudu může váznout (vytížené zdroje konkrétního počítače, záloha dat, problémy s internetem na straně cloudu, atd.).
- **Zabezpečení citlivých dat** – u cloudových služeb jsou všechna data ukládána právě zde. Jelikož nevíme, kde data jsou přesně ukládána, a také nevíme, kdo k nim má přístup, hrozí zde riziko úniku. I když jsou cloudové služby při ochraně dat na vysoké úrovni, stále zde není jistota. Proto je nutné mít na paměti toto nebezpečí.
- **Ztráta dat** – teoreticky se může stát, že všechny data, které máme v cloudu uložena, budou nějakým způsobem ztracena (vyhoření velké části serverovny, přírodní katastrofy, atd.). Jejich obnovení je takřka nemožné a trvalá ztráta dat, může znamenat obrovské finanční nebo jiné problémy pro klienty (především jedná-li se o firmu).



Obr. 5 Pokrytí ČR rychlým internetem od O2 v roce 2015[3]

Některé z výhod a nevýhod si můžou protiřečit. To je způsobeno především z pohledu využití cloudu. Každý zákazník může mít jinou představu, jakým způsobem chce cloudové služby používat a jaký typ si vybere.

2.4.1 Typy cloudových služeb

Typy cloudových služeb se liší v architektuře, která logicky vede k různým cílu jeho následného využití. Konkrétně se jedná o Communication as a Service (CaaS), Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Monitoring as a Service (MaaS), Database as a Service (DBaaS).

Software jako služba (SaaS)

Jedná se o službu, která běží pod cloudem ve formě aplikace. Ta je poté distribuována dále pro ostatní uživatele. Zákazníci neplatí za vlastnění softwaru, ale platí zde za jeho použití. Uživatelský přístup je zde řešen přes uživatelské rozhraní (UI[4]), které je přístupné přes web (webový prohlížeč).[15]

SaaS software je častěji aktualizován a to přímo na cloudu, takže je často aktuálnější než software nainstalovaný na lokálním počítači. Poskytovatel tedy není nucen složitě distribuovat změny mezi klienty. Zákazník tak dostává pravidelně vylepšovaný software bez dodatečných nákladů.

Platforma jako služba (PaaS)

Tento typ cloudové služby odstraňuje potřebu organizovat a řídit vrstvu infrastruktury (klient se nestará o hw a operační systém). Zákazník se tedy soustředí čistě jen na vývoj a správu vlastní aplikace, nemusí se starat o přerozdělování fyzických zdrojů (výpočetní výkon, úložiště, správa softwaru).[16]

---

[3] Zdroj www.O2.cz

[4] Z anglického „user interface“

Infractruture as a Service (IaaS)

„Definice IaaS je velmi jednoduchá. Zákazník si pronajme infrastrukturu – servery, úložiště a sítě – na vyžádání a to v modelu, kdy platí, o co požádá.“[17]

Vhodné pro strartupy nebo pro začínající podniky.

Databáze jako služba (DBaaS)

Zajišťuje podobnou funkcionalitu jako relační databáze typu SQL Server, MySQL a Oracle běžící na serverech.

Tato cloudová služba se vyznačuje flexibilností, škálovatelností a platformou na vyžádání orientovánu na samoobslužnou správu případně na jednoduché řízení. Produkty typu DBaaS poskytují dostatečné monitorující služby ke sledování výkonu a spotřeby s tím, že upozorňuje uživatele na potenciální problémy. Také mají možnost generovat určitý stupeň analytických údajů.[21]

3 Metodika

K dosažení cílů, které jsou určeny v zadání diplomové práce, je nezbytným prvotním krokem seznámení se s danou problematikou synchronizace dat a synchronizačních postupů na teoretické úrovni. K tomuto bodu je vhodné popsat architektury a jejich vývoj v čase. Uvést technologie, které řeší synchronizaci dat.

Z analýzy scénářů nečastějšího použití pro synchronizaci dat mezi Android zařízeními, se jeden ze scénářů zvolí a provede se jeho rozbor. Z informací zjištěných z rozboru scénáře, se určí body, které jsou nezbytné pro jeho splnění.

Vytvořením obecného návrhu řešení, se vytvoří metoda, pomocí které se zajistí všechny potřebné procesy vedoucí k synchronizaci dat a to i včetně ostatních procesů, které s touto synchronizací dat souvisí.

Na základě této metody, je poté možné vytvořit konkrétní návrh řešení, kde jsou určeny konkrétní způsoby, pomocí kterých lze vytvořit Android aplikaci, která primárně zajišťuje synchronizaci dat.

Pro prezentaci této metody bude naimplementována Android aplikace a to podle návrhu konkrétního řešení. Kroky implementace budou v práci popsány. Dále, bude daná metoda prostřednictvím Android aplikace otestována, optimalizována, vyhodnocena na základě spolehlivosti, latence a zátěže na zařízení.

V závěru práce budou shrnuty nejdůležitější body. Z nasbíraných informací, se vytvořená metoda vyhodnotí.

4 Přehled trhu

Průzkum, v jaké podobě se synchronizace dat nyní vyskytuje na trhu (rok 2015). Zvláště poté přímo pro systém Android.

4.1 Synchronizace přes uživatelský účet

Nejlepším způsobem nastínění vize pro synchronizování pomocí uživatelského účtu je vybrat konkrétní společnost a popsat daný proces. Jako ideální kandidát je společnost Google, která v minulosti zareagovala rychle na nově nastupovaný trend cloud computingu a zvolila, že nově online vytvářené služby budou založeny na cloudu. Tedy všechny data uživatele jsou přímo ukládaný na centralizovaném úložišti ve formě cloudového úložiště.

Jako unikátní identifikátor pro každého uživatele logicky zvolila emailovou adresu, díky čemu je možné pro konkrétního uživatele definovat jeho vlastní data z kterékoliv služby z rodiny Google Apps (Gmail, Calendar, Drive, Docs, atd.).

Výběr synchronizace dat právě od Googlu má svůj význam. Lze zde krásně určit, jaká různorodost dat lze synchronizovat.

- Data aplikací – jedná se o konkrétní data nejčastěji typu konfiguračního souboru, tyto data jsou malá a specifická pro používanou aplikaci, při smazání/přeinstalování aplikace data nezmizí a při obnově aplikace jsou data synchronizována do aplikace. Většinou drží informace o nastavení, které uživatel vykonal. Také dobrým příkladem je zde Google Calendar, kde jsou záznamy v kalendáři synchronizovány automaticky.
- Datové a multimediální soubory – nejčastěji soubory posílané přes email a soubory uložené uživatelem na Google Drive či fotky a videa z Google Photos. Jedná se o data střední/vyšší velikostí. Synchronizace je zde prováděna až na pokyn uživatel a to v případě, že používá klientskou aplikaci a vyžaduje mít data uložena i na lokálním úložišti daného zařízení (synchronizace dat může být nastavena automaticky při instalaci aplikace, tato možnost je nastavitelná).

4.2 Synchronizace dat přes klientskou aplikaci

Uživatel si může zvolit, jakým způsobem bude chtít obsluhovat svoje činnosti (spravovat svoje data). Má určité možnosti. Pokud mu nevyhovuje klientský přístup přes webového klienta, nabízí se možnost přístupu pomocí aplikace v případě, že je od výrobce dostupná.

Nejlepším znázorněním bude uvedení příkladu dostupného na trhu. Tedy při používání aplikace pro správu fotek, si uživatel například vybere možnost Google Photos. Zde je desktopové aplikace dostupná. Po nainstalování této aplikace, je zde prvotní nastavení pro synchronizaci, je totiž nezbytné vybrat složku

(případně i složky), kterou si uživatel přeje mít synchronizovanou. Jedná se o jedno z důležitých nastavení pro synchronizaci dat a je zde nutná interakce uživatele. Usnadněním ze strany výrobce, je určitá míra intuitivního přednastavení, zde v konkrétním případě přidané složky „Plochy“, „Obrázky“ a „Fotoaparáty a paměťově karty“. Tato aplikace poté běží na pozadí a pokaždé, když detekuje soubor v synchronizované složce, který má příponu definující fotku, provede upload fotky na úložiště (provede synchronizaci).

Jelikož toto úložiště je už nyní nejčastěji realizováno pomocí cloudových služeb a tedy úložišť v podobě cloudu je v další podkapitole představen i přehled trhu právě pro tento typ služby.

4.3 Synchronizace dat přes Android aplikaci

Android aplikace je opět prostředek, který umožňuje uživatelovi přístup k informacím, službě, zábavě nebo práci. Také umožňuje interakci uživatele k vytváření akcí. Může se jednat také o jediný možný přístup uživatele k obsahu.

Díky velké popularitě Android systému na chytrých zařízeních, je v dnešní době je na trhu přes milion Android aplikací, které jsou distribuované na Google Play. Tento jev podtrhují zákazníci. Používání aplikací je pro ně většinou velice intuitivní, jednoduché a hlavně rychlé bez zbytečného zdržování.

Smysl synchronizace dat je brán už jako konkrétní proces, který je vhodné do co největší míry automatizovat. V tomhle případě zde záleží na šikovnosti nebo zkušenostech vývojáře. V mnoha případech už tento proces není brán jako hlavní smysl aplikace, ale už jen jako služba, která se v drtivé většině odehrává na pozadí a maximálně podává uživatelovi oznámení o dokončené akci. Pro synchronizaci dat je většinou potřebné nastavit mnoho prvků a je zvykem ponechat na uživatelovi jen ty nejnutnější, aby byl co nejméně obtěžován.

Jako dobrým příkladem, se zde jeví Android aplikace Google Drive. Prostředí aplikace je uživatelsky přátelské a jednoduchého zobrazení. Smyslem aplikace je správa dat na cloudovém úložišti. Synchronizace dat je zde vhodná, především díky používání lokálního souborového systému daného zařízení. Je tedy vhodné nastavení složky, která má být neustále synchronizována. Jako vhodný příklad se zde hodí složka, kde jsou ukládány fotky z fotoaparátu uživatele.

Jako dalším případem synchronizace dat jsou zde Android hry. Pro ty byl vytvořen celý systém statistik, které fungují právě na bázi synchronizace. Je nutné, aby každá změna byla zaznamenána (změna nasbíraných mincí ve hře, změna v nejvyšším skóre dosažené uživatelem). Hlavním předpokladem je, že uživatel může vlastnit více chytrých zařízení a mít Android aplikaci/hru nainstalovanou na více zařízení a tedy, aby měl tyto data neustále synchronizována na všech zařízeních.

Jestliže je možné přistupovat k synchronizovaným datům z více míst, vytváří se zde podsekce problémů, která je nazvána jako „vznik konfliktů“. Jako příklad lze uvést hraní totožné hry ve stejný čas ze dvou zařízení. Pokud přijde na

server ve stejný čas více požadavků, je potřebné správně tyto požadavky vyhodnotit, aby nebyly nesprávné nebo nebyly ztraceny data (jedny přepíšou druhé).

4.4 Cloudové služby pro ukládání dat

Cloud je nesporně jedním z trendů současnosti. Doby, kdy se obsah 1,44MB diskety přenášel přes Internet třeba i deset minut, jsou naštěstí minulostí. Postupně si zvykáme na skutečnost, že máme data uložená „kdesi v oblacích“ – zpravidla na počítačích komerčních firem. Přináší to totiž celou řadu výhod – v první řadě dostupnost kdykoli a kdekoli, kde je Internet, v řadě druhé bezpečné uložení, kde se o správu, zabezpečení a zálohování starají profesionálové s technologiemi, které si běžný uživatel pro domácí nasazení nemůže dovolit. Argumentem pro cloudový disk bývá také synchronizace – například soubor, který rozpracujete doma, na vás bude čekat v identické kopii v práci apod. Cloud také významně usnadnil sdílení souborů s ostatními uživateli – už nejsou potřeba nejrůznější „letecké pošty“, protože stačí data nahrát do synchronizované složky a poslat příjemcům odkaz.

Jelikož při synchronizaci dat je cloud nesporně přínosnou technologií, bude proveden přehled dostupných cloudových služeb na trhu právě ve vztahu k synchronizaci dat.

4.4.1 Dropbox

Cloudová služba Dropbox se stala fenoménem a symbolem pro cloud.

Jakožto služba je zde rozdíl mezi placenou a neplacenou verzí. Hlavní rozdíl je v nabízené velikosti úložného prostoru pro data. Pro neplacenou verzi je defaultně nastavena hodnota 2 GB. Za větší úložný prostor je potřebné zaplatit paušální poplatek.

Verzi pro Android si podle údajů Obchodu Play stáhlo více než 100 milionů uživatelů. Průměrné hodnocení je 4,4 z 5 možných, aplikaci můžete instalovat na zařízení s Androidem 2.1, což navazuje na fakt, že od této verze je možná implementace pomocí Android frameworku Sync Adapter.[11]

Výhody Dropboxu[12]:

- Své soubory budete mít vždy u sebe, bez ohledu na to, kde se aktuálně nacházíte.
- Automatické ukládání fotografií a videí do vašeho Dropboxu.
- Snadné sdílení obrázků a dokumentů s rodinou a přáteli.
- Jednoduché ukládání e-mailové přílohy přímo do cloudu.
- Snadná editace dokumentů ve vašem Dropboxu.

4.4.2 Google Drive

Cloudová služba pod záštitou společnosti Google. Je propojena s Gmailem a umožňuje přímo z příloh ukládat data do cloudu. V neplacené verzi je standardně k dispozici 15 GB zdarma. Za placenou verzi se platí paušální poplatek.

Aplikaci Google Drive si podle údajů Obchodu Play stáhlo jednu miliardu uživatelů. Průměrně je hodnocen známkou 4,3, instalovat lze od verze Androidu 2. 1.[11]

Výhody Google Drive[12]:

- Ukládání všech souborů na jedno místo, takže k nim můžete přistupovat odkudkoliv a sdílet je s ostatními.
- Snadný přístup k fotografiím, dokumentům, videím a dalším souborům uloženým na Google Drive.
- Nahrávání souborů na Google Drive přímo ze zařízení se systémem Android.
- Sdílení libovolného souboru s ostatními uživateli a přístup k souborům, které ostatní sdílí s vámi.
- Jakýkoliv soubor můžete stáhnout, takže bude k dispozici, i když nemáte připojení k Internetu.
- Možnost vytvářet a upravovat dokumenty s podporou tabulek, komentářů a formátování textu.
- Vytváření a editace tabulek s podporou pro formátování textu, více listů a třídění.
- Zobrazení prezentací s animacemi a poznámkami pro přednášejícího.
- Podpora pro dokumenty PDF a formát MS Office.
- Tisk souborů uložených v cloudu pomocí Google Cloud Print
- Optimalizované rozhraní pro zařízení s větší obrazovkou – tablety s Androidem verze 3.0 +.

4.4.3 OneDrive

OneDrive dříve známá pod názvem SkyDrive je cloudový disk pro ukládání souborů, ke kterým můžete přistupovat z prakticky libovolného zařízení. Je od společnosti Microsoft. Automaticky se synchronizuje s telefony s operačním systémem Windows. K této synchronizaci stačí jen propojit emailový účet.

Na trhu je také dostupná aplikace pro android a to OneDrive, kde budete mít své soubory kdekoli a kdykoli po ruce a můžete je sdílet s ostatními. Aplikace také umí odesílat fotografie nebo videa z telefonu do služby OneDrive. Podle údajů Obchodu Play si aplikaci stáhlo přes deset miliónů uživatelů. Průměrně je hodnocen známkou 4,4.[11]

Výhody OneDrive[12]:

- Přístup ke všem datům uloženým na SkyDrive, včetně souborů, které s vámi někdo sdílí.
- Zobrazení naposledy použitých dokumentů.
- Možnost odeslat více fotografií nebo videí z telefonu či tabletu.

- Jednoduché sílení souborů a fotek odesláním odkazu v e-mailu nebo skrze jinou aplikaci.
- Možnost otevírat soubory z cloudu v jiných aplikacích.
- Správa souborů – odstraňování a vytváření nových složek.

4.5 Shrnutí

Z literární rešerše a přehledu trhu, jsme zjistili velice důležité informace pro řešení daného tématu diplomové práce. Jako klíčový bod je zde evidentní boom technologie založené na cloudu. Během pár let přešly velké mezinárodní firmy z architektury klient-server/p2p na perspektivnější technologii cloud computingu. Z předchozích textů, kde jsou dané koncepty detailně popsány, jsou výhody cloudu nesporné.

5 Analýza

Analýzou, zjištěných informací o problematice synchronizace dat, je snaha složitější informace rozdělit na jednodušší a pracovat s těmito informacemi na základní úrovni.

Ideálním způsobem jak zanalyzovat možné řešení pro synchronizaci dat, je vytvořit scénáře pro jeho nejčastější použití a vytvořit tak náhled na jeho využití.

Scénář popíše konkrétní situaci, pro které se synchronizace dat nabízí a uživatel ji vyžaduje nebo minimálně uvítá. Jak bylo uvedeno, synchronizace dat od uživatele není ve většině případů brána jako nejdůležitější bod jeho práce. Jedná se spíše o službu navíc, o které v neideálnějším případě uživatel vůbec nemusí vědět, ale přesto jsou jeho data synchronizována (všechny procesy běží automaticky v pozadí aplikace, bez nutného zásahu uživatele).

5.1 Analýza pro synchronizaci pracovních dat

Jedná se o praktický příklad z praxe. Právě využití technologických možností pro pohodlnější a efektivnější práci a jeho neustálá inovace, je způsob, kterým se společnosti snaží získat konkurenční výhodu.

Při dnešních možnostech pro připojení k internetu z jakéhokoliv chytrého zařízení, se logicky i práce stala flexibilnější. Takové možnosti sebou nesou i určité návazné problémy. Konkrétně v tomhle případě je problém v datech. Tyto data bez určité kontroly nebo procesů, které se starají o kontrolu dat, nebudou vždy na všech zařízeních aktuální.

Právě proces synchronizace dat zajistí, aby data byla na všech chtěných místech v aktuální podobě. Zaměstnanec má v práci počítač, kde pracuje s daty (dokumenty, speciální data). Tyto data jsou ukládány na lokální úložiště. Nyní je zde ještě nutné, zajistit proces pro synchronizaci dat. Jedná se o souhrn akcí, které vedou k tomu, aby tato data byla řádně synchronizována. K dosáhnutí je možné použít tří možných architektur:

- **P2p** – distribuce dat je prováděna klientskou aplikací. Jakákoliv změna v určeném místě každého lokálního úložiště zaměstnance je okamžitě přenesena do všech uzlů v síti.
- **Klient-server** – všechna chtěná data, která jsou potřebná synchronizovat, jsou zasílána na server, odkud jsou tyto data dále distribuována do všech nastavených synchronizovaných složek.
- **Cloud** – buď jsou data zpracovávána přímo na cloudovém úložišti a synchronizace je tak obejita jedním centralizovaným úložištěm, kde jsou data pro všechny zaměstnance spravována a aktualizována v reálném čase. Nebo je synchronizace dat nastavena zvlášť pro každého zaměstnance, který si definuje vlastní složku na lokálním úložišti, kde chce mít konkrétní data synchronizována.

Jak lze vidět u uvedených příkladů, není zde problém v typu zařízení, ze kterého se s daty manipuluje (pracuje). Samozřejmě je nutné zajistit, aby tento přístup byl možný, tedy jestliže zaměstnanec například využívá k práci i Android zařízení, je nutné, aby měl možnost s daty manipulovat, ať už pomocí Android aplikace nebo pomocí webového klienta a také, aby zde byl naimplementován proces, který se o synchronizaci dat stará interně. Pokud zde tento proces naimplementován není, je možné ještě využít druhé Android aplikace, která se o tento proces postará externě. Všechny uvedené příklady, nemusí být právě ideálním řešením, hlavním požadavkem u synchronizace dat, je tyto data držet v aktuálním stavu a až poté řešit případnou optimalizaci.

5.1.1 Scénáře použití pro synchronizaci pracovních dat

Ve scénářích není důležité, jakým způsobem je synchronizace dat řešena. Cílem je uvedení konkrétního příkladu. Ve většině firem bude jejich podnikatelský plán a firemní politika natolik rozdílná, že to co je pro jednu firmu ideální strategie, může být pro jinou firmu nevyhovující nebo nákladově nemožné.

1. Zaměstnanec Jiří, chce **synchronizovat jeho vytvořené soubory k projektu**. Jedná se o pár dokumentů každý do velikosti 10MB.
2. Fotografka Jana, chce **synchronizovat své fotky z pracovního focení, které jsou uložené na fotoaparátu**. Jedná se o 100 fotek, každá do velikosti 20MB.
3. Grafik Petr, chce mít **neustále synchronizovanou složku s dostupnými vzory od jeho kolegů k rozpracovanému projektu.**
4. Správce projektů Jan, chce mít **neustále přehled o vývoji každého projektu, a proto chce mít synchronizované složky každého projektu se soubory od zaměstnanců**. Jako správce těchto projektů má ke každé přístup.
5. Manažer Alois, vyžaduje mít **dostupné všechny pracovní smlouvy na všech chytrých zařízeních, aby je mohl využívat flexibilně a kdykoliv na cestách**. Po tomto požadavku na IT oddělení byla jeho chytrá zařízení nastavena, tak aby se všechny jeho pracovní soubory synchronizovaly.

5.2 Analýza pro synchronizaci osobních dat

Každý člověk, který vlastní a používá chytré zařízení, vytváří svojí činností data. Může se jednat o velice různorodá data. Například vytvořením fotky z digitálního fotoaparátu, vytvoří uživatel svoje osobní data, která jsou v oblasti informační technologie chápana jako datový soubor. Podobně je to i při vytvoření a uložení dokumentu nebo při uložení pozice v rozehrané hře a tak dále.

Při synchronizaci osobních dat je většinou nutné, aby si uživatel tento proces řešil sám. Nelze totiž předvídat, které složky nebo které soubory je každý

uživatel ochoten synchronizovat a také nelze určit souborové cesty k daným složkám.

5.2.1 Scénáře použití pro synchronizaci osobních dat

1. Student Pepa, chce mít **všechny svoje soubory nashromážděné během studia automaticky synchronizována i s jeho mobilním telefonem**, aby měl tyto data vždycky u sebe, jelikož jeho úložiště na telefonu je často plné, nepřeje si synchronizovat soubory větší než 15MB.
2. Fanoušek do filmů Jarek, hledá **řešení, jak svojí kupovanou filmovou sbírku, mít pro jistotu zálohovanou i na jiném, než jen na lokálním úložišti**. Protože počítač často nevypíná, ocenil by, kdyby **byla jeho sbírka automaticky aktualizována**, když zrovna počítač nevyužívá.
3. Hudebník Martin, by měl rád celou jeho **sbírku hudby na počítači dostupnou i na svém mobilním telefonu**, aby mohl poslouchat nové skladby i když není zrovna připojen k internetu během jeho cest do zahraničí.
4. Mladí manželé s různorodou pracovní dobou, chtějí **mít dostupné a synchronizované svoje kalendáře online**, aby věděli, kdy budou mít volný čas.
5. Studentka Jarmila, by ocenila, kdyby **všechny její fotky, které si upravuje a maže na svém počítači, byly poté stejně synchronizovány i na jejím chytrém telefonu**, například aby nemusela fotky opakovaně mazat právě i na telefonu.

5.3 Analýza pro synchronizaci dat na Androidu

Zařízení se systémem Android je multifunkčního charakteru. Neustálé vylepšování systému vede k novým možnostem využití.

Při prvním aktivování zařízení s Androidem, je uživatel vyzván k zadání email účtu. Právě správě účtu je věnován dostatečný prostor. V nastavení zařízení je pro tento účel věnována celá kategorie. Aplikace, které vyžadují držení účtu, jsou pak právě schopná pomocí těchto účtů synchronizovat data aplikace (Google, Dropbox, Skype, Messenger, atd.).

U aplikací určených pro synchronizaci souborových dat záleží na konkrétní implementaci. Například Android aplikace pro Dropbox, neumožňuje nastavení další složky, která by se automaticky synchronizovala, jako jediná je přednastavena pouze složka fotoaparátu.

U her pro Android, je opět využíváno synchronizace dat pomocí email účtu. Tento účet je nejčastěji právě Google účet (může být ale zvolen uživatelem). Pro správu her a uživatelových aplikačních dat je zde vytvořená služba „Hry Play“ (Play Games Services), ta se stará o aktualizaci (a tedy i synchronizaci) herních výsledků dosažených hráčem (žebříčky, dosažené zkušenostní body, sesbíraných mincí, uložené pozice, odemčené lokace, atd.). Pro vývojáře je dostupné API vy-

tvořené přímo společností Google, pro zajištění integrity a také pro pohodlnější implementaci. Je dobrým zvykem vyvíjet hry tak, aby je bylo možné spustit i v offline režimu. **Aplikační data jsou zatím uchovávaná na lokálním úložišti do doby, než bude zařízení opět v online režimu.** Poté je možné aktualizovat data. Na tuto situaci se váže **možnost vzniku konfliktu** při synchronizaci dat a je nutné tuto problematiku popsat, viz kapitola 6.

5.3.1 Scénáře použití pro synchronizaci dat na Androidu

1. Zaměstnanec Jiří, **chce synchronizovat jeho upravené soubory ze zařízení Android k projektu**, aby je měl aktuální i v práci na počítači. Jedná se o pár dokumentů každý do velikosti 10MB.
2. Hráč Filip, který **hraje hry doma na Android tabletu, by měl rád odemčené lokace, kterých už dosáhl i na svém telefonu**, který používá při cestách.
3. Manažer Alois nikdy neví, který den bude využívat které zařízení. Proto by ocenil **synchronizovanou složku s jeho záložkami v Chrome prohlížeči.**
4. Firma hledá na trhu vývojáře, který pro její už zavedenou a funkční webovou službu s mapami vytvoří **Android aplikaci pro prezenci geodat**. Tyto geodata obsahují **informace jak účelové pro potřeby na zpracování ze strany serveru, tak i prezenční**, které jsou poté zobrazovány a případně i upravovány uživateli a je zde tedy nutné **zajistit proces pro synchronizaci dat.**
5. Zapomětlivý čtenář Honza, **využívá ke čtení knížek Android aplikaci na svém chytrém telefonu i tabletu**. Jelikož si nikdy nepamatuje, na které stránce skončil, **ocenil by pokaždé záložku právě na poslední rozečtené stránce nezávisle na zařízení.**

6 Vznik konfliktů

Při procesu synchronizace dat z více zařízení existuje nebezpečí vzniku konfliktů. Konflikt vzniká při snaze aktualizace souboru z více zařízení ve stejný čas (cyklu synchronizace)[5], v tom případě že byl daný soubor změněn alespoň na dvou umístěních. Tyto konflikty se můžou vyskytnout v mnoha různých variacích a není tedy divu, že se i jejich následné řešení může lišit. Existují určité postupy, podle kterých lze postupovat, ty budou popsány v následujících podkapitolách a to i včetně možných scénářů vzniku konfliktu.

V případě, že je možné takto vzniklý konflikt vyřešit algoritmicky správně (korektní výsledky bez ztráty dat), je dobrým zvykem uživatele o tomto problému neinformovat a celý tento problém řešit automaticky v pozadí programu. Jestli není možné konflikt řešit bez interakce uživatele, nezbývá možnost než tuto situaci oznámit a nechat uživatele rozhodnout. Je dobrým zvykem vyhodnotit situaci pomocí některého algoritmu a dát uživatelovi na výběr z nejpravděpodobnějších možností řešení vzniklého konfliktu.

Ukázka řešení konfliktu

Robustní strategie na rozhodování konfliktů pro aplikace je popsána v dokumentaci pro Android vývojáře [17]. Ukázka je založena na ukládání herních statistik uživatele jako ukládání aplikačních dat do cloudu nazvanou „Cloud Save service“. Tato ukázka je v textu dále rozebrána a je na ní vysvětlen postup řešení konfliktu.

Cloud Save service popisuje obecně jak ukládat data z aplikací. Jedná se o multiplatformní řešení (Android, iOS, C++). Až při vybrání konkrétního řešení pro Android je zpřístupněno aplikační uživatelské rozhraní „Saved Games API“, s kterým pracuje vývojář aplikace.

Tato služba umožňuje vývojáři ukládat aplikační data každého uživatele do Google cloudu. Aplikace poté může data přijímat a aktualizovat při používání Android zařízení.

Ukládání a nahrávání postupu přes Cloud Save je přímočarého charakteru. Jediné na čem záleží je transformace hráčských informací (uložené pozice, získané zlaťáky, získané zkušenosti, atd.) do cloudu ve formě dat. Problém vzniká, pokud uživatel vlastní více zařízení. Jestliže chce uložit data ze dvou čí více zařízení, může vzniknout konflikt. Tento problém musí řešit vývojář. Struktura ukládaných dat do cloudu definuje, jak robustní konflikt může vzniknout. Je vhodné pečlivě navrhnout strukturu těchto dat, aby bylo možné logicky vyřešit všechny možné případy konfliktu.

---

[5] Proces synchronizace dat probíhá po určitých časových intervalech tak, aby byla docílena efektivita při spotřebě baterie zařízení. Tedy neběží neustále. Během tohoto intervalu je tedy možné, že se soubor změní na více místech současně. Ke konfliktu může také dojít, když zařízení pracuje v offline režimu.

Nejvhodnějším postupem pro pochopení robustnosti daného řešení, je jeho popis od prvotních nedostačujících řešení až k jeho finálnímu řešení. Nejlepším způsobem, je vybrat si kategorii, u které se bude řešit vznik konfliktu. Zásady lze poté použít pro jakoukoliv jinou kategorii a aplikaci, která ukládá data do cloudu.

Hry jsou velice oblíbené a rozšířené na Google Play a tak není důvod proč nepopsat tuto kategorii. Právě konflikt se konkrétně pro systém Android řeší v programové části v metodě zobrazené na Kód 1. V této metodě je poté řešena implementace včetně logiky pro řešení konfliktu.

```
@Override
public void onStateConflict(int  stateKey, String resolvedVersion,
  byte[] localData, byte[] serverData){
  //vyřeš konflikt, poté volej mAppStateClient.resolveConflict()
}
```

Kód 1. Ukázka programového kódu pro řešení konfliktu[17]

Pro vývojáře je důležité si uvědomit, že synchronizace dat probíhá na systému Android v pozadí. Proto je nutné zajistit, aby aplikace byla připravena přijmout zpětné volání mimo kontext, kde jsou původně generována data. Nic to nemění na tom, že je pokaždé nutné rozhodnout mezi dvěma konfliktními stavy v rámci dostupných sad dat, bez zásahu jakéhokoliv vnějšího kontextu.

Základní postup pro řešení konfliktů

V mnoha případech se jedná o konflikty, kdy lze jednoduše rozhodnout, která sada dat je ta správná. Následná implementace není náročná. Důležitým aspektem je mít aplikaci (hru) správně zanalyzovanou a mít přehled, které konflikty můžou nastat.

- **Nové je lepší než staré** – standardně je bráno, že nová data vždy nahrazují data stará. Například pokud si hráč změní u své postavy barvu vlasů, pak poslední změna přepíše předcházející stav. V tomhle případě, je vhodné s ukládanými daty uchovávat i *timestamp*[6] pro určení poslední změny (nutné zohlednit rozdílné časové zóny).
- **Jedna sada dat je očividně lepší než jiná** – záleží na situaci a reprezentaci sady dat. Například v závodní hře bude upřednostňována sada dat s nejlepšími výsledky (dosáhnutí nejnižšího času).
- **Sloučení dat jejich sjednocením** – řešení konfliktu na základě sjednocení dvou konfliktních množin. Například, pokud data reprezentují množinu úrovní, které už hráč odemknul. Výsledkem bude jednoduše sjednocení obou konfliktních množin a hráč nepřijde o žádnou z úrovní.

[6] Timestamp – časové razítko

Řešení konfliktů pro komplexnější případy

Právě u her vyvíjených pro Android často dochází ke vzniku složitějších konfliktů. Hráč vyžaduje od hry určitou kvalitu. Vývojáři proto musí vytvářet komplexní hry, proto aby hráče dokázal udržet co nejdéle právě u své hry. S tím souvisí samozřejmě i náročnější logika na výstavbu hry. Častými případy jsou hry, které obsahují hráčovi sbírat zaměnitelné předměty nebo jednotky jako jsou zlaté mince nebo zkušenostní body.

Vytvořme si příklad hry pro lepší přiblížení způsobu řešení komplexnějšího konfliktu. **Hra pojmenována „Zlatý běh“,** kde je cílem hráče v nekonečném běhu nasbírat co nejvíce zlaťáků a stát se velice bohatým. Každá mince, kterou hráč sebere, poputuje do jeho pokladnice. Následující sekce popisují strategie pro řešení synchronizačních konfliktů vzniklých při používání více zařízení.

1. **Ukládání pouze celkové hodnoty** – na první pohled se zdá, že data uložená na cloudu budou vyjádřena pouze velikostí hráčovi pokladnice. Což v některých případech není pravda. Pokud tedy v sadě dat nedržíme jinou informaci, možné řešení konfliktu bude velice omezené. Nejlepší možná volba je vybrání nejvyšší hodnoty z konfliktních množin (což nejspíš povede ke ztrátě dat).

Tab. 1 Ukládání pouze stavu pokladnice (neúspěšná strategie) [17]

Událost	Data na zařízení A	Data na zařízení B	Data na Cloud	Skutečná Data
Začáteční podmínky	20	20	20	20
Hráč nasbírá 10 zlatých na zařízení A	30	20	20	30
Hráč nasbírá 15 zlatých na zařízení B	30	35	20	45
Zařízení B ukládá stav do cloudu	30	35	35	45
Zařízení A se pokusí uložit stav do cloudu. Detekován konflikt.	30	35	35	45
Zařízení A vyřeší konflikt vybráním větší hodnoty z dvou možných.	35	35	35	45

Podle scénáře ilustrované na Tab. 1, má hráč na kontě 20 zlaťáku. Následně na zařízení A nasbírá 10 zlaťáků a 15 zlaťáků na zařízení B. Poté zařízení B uloží svůj stav do cloudu. Když se poté zařízení A pokusí o uložení, bude **detekován konflikt**. Podle tohoto algoritmu bude konflikt vyřešen zapsáním vyšší hodnoty z dostupných dat a zapíše 35 zlaťáků, čímž dojde ke ztrátě zlaťáků (ztrátě dat).

2. **Ukládání celkové hodnoty a pomocné hodnoty delty** – reprezentace datové sady je zde vyjádřena celkovou hodnotou (T) a pomocnou hodnotou delta (d), která udává poslední změnu celkové hodnoty. Vyjádřeny jsou dvo-

jicí (T, d). Takhle zvolená struktura umožňuje robustnější algoritmus pro řešení vzniklého konfliktu. Zápis a data definujeme:

- Lokální data: (T, d)
- Cloud data: (T´, d´)
- Výsledná data: (T´ + d, d)

Strategie při konfliktu mezi lokálními daty (T, d) a cloudovými daty (T´, d´) je řešena výslednými daty jako (T´ + d, d) neboli bereme deltu z lokálních dat a zahrnujeme do cloudových dat. Toto řešení zní slibně, ale hatí je dynamické mobilní prostředí:

- Hráč totiž může ukládat data, i když je zařízení offline. Tyto změny budou zařazeny k předložení, do doby než bude zařízení opět v režimu online.
- Proces synchronizace pracuje tak, že poslední změna přepisuje předcházející změnu. Tedy právě druhé zapsání je jediné, které bude posláno do cloudu (jakmile bude zařízení online), tak je delta z prvního zápisu ignorována.

Tab. 2 Ukládání stavu pokladnice a změny delty (neúspěšná strategie) [17]

Událost	Data na zařízení A	Data na zařízení B	Data na Cloud	Skutečná Data
Začáteční podmínky	(20, x)	(20, x)	(20, x)	20
Hráč nasbírá 100 zlatých na zařízení A	(120, +100)	(20, x)	(20, x)	120
Hráč nasbírá dalších 10 zlatých na zařízení A	(130, +10)	(20, x)	(20, x)	130
Hráč nasbírá 115 zlatých na zařízení B	(130, +10)	(135, +115)	(20, x)	245
Hráč nasbírá dalších 5 zlatých na zařízení B	(130, +10)	(140, +5)	(20, x)	250
Zařízení B ukládá stav do cloudu	(130, +10)	(140, +5)	(140, +5)	250
Zařízení A se pokusí uložit stav do cloudu. Detekován konflikt.	(130, +10)	(140, +5)	(140, +5)	250
Zařízení A vyřeší konflikt použitím lokální delty do cloudu	(150, +10)	(140, +5)	(150, +10)	250

Jak lze ze scénáře v Tab. 2 vyčíst po dvou zápisech na zařízení A i B, pošle zařízení B data do cloudu, v dalším kroku se o stejné uložení pokusí zařízení A. Zde **vzniká konflikt** – nesouhlasí skutečný stav pokladnice s cloudovými daty. Tato chyba je v podobě zapsání pouze druhé delty ze zařízení A do cloudu a je tedy

ztracená hodnota první delty zařízení A (rozdílem je první hodnota delty 100 – celková hodnota na cloudu 150, skutečná celková hodnota 250).

Tato strategie se může zdát opravitelná nemazáním delty po každém uložení, pak delta na každém zařízení bude znamenat počet nasbíraných zlaťáků na daném zařízení. Jak lze vidět z Tab. 3 tentokrát je důsledek **vzniku konfliktu** v přidání zlaťáků hráčovi.

Tab. 3 Modifikovaná předchozí strategie (neúspěšná strategie)[17]

Událost	Data na zařízení A	Data na zařízení B	Data na Cloud	Skutečná Data
Začáteční podmínky	(20, x)	(20, x)	(20, x)	20
Hráč nasbírá 100 zlatých na zařízení A	(120, +100)	(20, x)	(20, x)	120
Zařízení A uloží stav do cloudu	(120, +100)	(20, x)	(120, +100)	120
Hráč nasbírá dalších 10 zlatých na zařízení A	(130, +110)	(20, x)	(120, +100)	130
Hráč nasbírá 1 zlatý na zařízení B	(130, +110)	(21, +1)	(120, +100)	131
Zařízení B se pokusí uložit stav do cloudu. Detekován konflikt.	(130, +110)	(21, +1)	(120, +100)	131
Zařízení B vyřeší konflikt použitím lokální delty do cloudu	(130, +110)	(121, +1)	(121, +1)	131
Zařízení A se pokusí uložit stav do cloudu. Detekován konflikt.	(130, +110)	(121, +1)	(121, +1)	131
Zařízení A vyřeší konflikt použitím lokální delty do cloudu	(231, +110)	(121, +1)	(231, +1)	131

3. Řešením je **ukládání mezisoučtu na zařízení** – analýzou předchozích strategií, lze konstatovat, že chybí schopnost poznat, které zlaťáky už byly započítány a které ne, především pokud přichází po sobě následující commity (zápisy) z různých zařízení na cloud. Klíčem k řešení je změna struktury sady dat na slovník, který mapuje řetězce a čísla. Každý „key-value“ pár ve slovníku reprezentuje zásobník, který obsahuje zlaťáky ze zařízení. Suma těchto zlaťáků určuje stav pokladny. Strukturu slovníku lze zapsat jako (A:a, B:b, C:c, ...), kde „a, b, c“ je celkový počet zlaťáků ze zásobníku „A, B, C“ (zařízení). Zápis a data definujeme:

- Lokální data: (A:a, B:b, C:c, ...)

- Cloud data: (A:a´, B:b', C:c', ...)
- Výsledná data: (A:max (a,a´), B:max (b,b´), C:max (c,c´), ...)

V Tab. 4 testujeme strategii z předchozího scénáře. Jako počáteční stav pokladnice je uvedena hodnota 20 zlaťáků, psána jako X:20 a to pro všechna zařízení. Když je z kteréhokoliv zařízení nasbírán zlaťák, uloží se právě do svých lokálních dat. Při nahrávání do cloudu pokud se liší hodnoty ze zařízení, jsou zapsány. **Konflikty jsou řešeny držením celkového stavu každého zařízení.** Výsledná hodnota je poté suma všech zásobníků, což pro daný scénář je správně hodnota 131 zlaťáků.

Tab. 4 Strategie slovníku (úspěšná strategie)

Událost	Data na zařízení A	Data na zařízení B	Data na Cloud	Skutečná Data
Začáteční podmínky	(X:20, x)	(X:20, x)	(X:20, x)	20
Hráč nasbírá 100 zlatých na zařízení A	(X:20, A:100)	(X:20)	(X:20)	120
Zařízení A uloží stav do cloudu	(X:20, A:100)	(X:20)	(X:20, A:100)	120
Hráč nasbírá dalších 10 zlatých na zařízení A	(X:20, A:110)	(X:20)	(X:20, A:100)	130
Hráč nasbírá 1 zlatý na zařízení B	(X:20, A:110)	(X21, B:1)	(X:20, A:100)	131
Zařízení B se pokusí uložit stav do cloudu. Detekován konflikt.	(X:20, A:110)	(X:20, B:1)	(X:20, A:100)	131
Zařízení B vyřeší konflikt	(X:20, A:110)	(X:20, A:100, B:1)	(X:20, A:100, B:1)	131
Zařízení A se pokusí uložit stav do cloudu. Detekován konflikt.	(X:20, A:110)	(X:20, A:100, B:1)	(X:20, A:100, B:1)	131
Zařízení A vyřeší konflikt	(X:20, A:110, B:1)	(X:20, A:100, B:1)	(X:20, A:100, B:1) celkem 131	131

7 Technologie pro synchronizaci dat

7.1 Android Wi-fi Peer-to-Peer

Je metoda propojení dvou a více zařízení napřímo pomocí bezdrátové sítě (wi-fi), bez potřeby třetí strany. U systému Android je toto API[7] dostupné od verze 4.0. Android Wi-Fi P2P framework[8] je v souladu s Wi-Fi Alliances Wi-Fi Direct™ certifikačním programem.

„Certifikované přístroje Wi-fi Direct mohou být použity pro všechny druhy aplikací – pro sdílení dat, synchronizaci dat, hraní her a podobně. Jedná se o všechny akce, které lze dělat běžně na Wi-fi, akorát bez potřeby vyhledání internetového připojení.“[7]

Jedná se tedy o jednu z možností jak synchronizovat data mezi android zařízeními, ale s omezením pouze lokální sítě a to v dosahu dané bezdrátové sítě. Dané aplikační rozhraní obsahuje tři hlavní části:

- Metody, které jsou definovány v *„WifiP2pManager“* třídě.
- Listeners[9], které detekují změny během komunikace.
- Intents[10], jsou to systémové zprávy upozorňující aplikace na výskyt událostí, změnami hardwarové konfigurace (například vložení SD karty) přes události související s příchozími daty (například ukončení stahování) a událostmi aplikace (například změna stavu připojení).[8]

7.2 Android synchronizace přes SyncAdapter

Synchronizace dat mezi Android zařízením a webovým serverem nebo cloudem vytvoří podstatně zajímavější a především užitečnější aplikaci pro uživatele. Například přenesením dat do cloudu umožní uživatelům přistupovat k datům online. Tyto data tam jsou poté dostupná, i když už je zařízení, na kterém byly data vytvořena vypnuté. Někdy je pro uživatele nejvhodnějším způsobem data měnit z webového klienta a někdy s těmito daty pracovat z chytrého telefonu či tabletu. Tento přístup, lze ještě vylepšit na úroveň automatické synchronizace dat. Poté už uživatel vůbec nemusí řešit, kde jsou data uložena, a přesto bude mít vždy na každém zařízení přístup k aktuálním datům.

Android od verze 2.1 nabízí framework SyncAdapter, který synchronizaci dat umožňuje.[10]

---

[7] Application Programming Interface (API) – rozhraní pro programování aplikací (procedury, funkce, třídy a protokoly, které může programátor využívat)

[8] Framework – aplikační rámec

[9] Volně přeloženo jako „posluchače“

[10] Volně přeloženo jako „záměry“

7.3 Parse

Nástroj pro tvorbu aplikací na různé platformy od společnosti Facebook. Specializuje se na zjednodušení implementační části, snaží se vývojáři aplikace ulehčit implementační prvky, které lze nechat běžet na pozadí a není potřebné se jim věnovat. Všechno co vyvíjená aplikace potřebuje od ukládání dat až po to mít možnost propojení se sociálními účty. Data lze ukládat a uchovávat online bez potřeby správy jediného serveru.

Jeho výhodou je možnost komunikace napříč zařízeními a jejich různorodými platformami. Aplikace pro Android lze vyvíjet pomocí nástroje Parse Android SDK. Pro flexibilitu práce lze data uchovávat online a to ukládáním do cloudu.[9]

7.4 Google Cloud Platform

Je takřka nemožné, nejmenovat možnosti, které jsou nabízeny od společnosti Google. Jejich brzké zachycení technologie cloudu a velké investice do vývoje v minulosti, jsou nyní viditelné v jejich široké nabídce produktů nabízené jako cloudové služby z rodiny Google Cloud Platform.

Tyto služby jsou rozděleny do kategorií vyznačující se jedním okruhem zájmu pro daný problém. Jak lze dohledat na jejich webových stránkách[23] jedná se o kategorie, které zahrnují služby:

1. **Compute**
   - 1.1. Compute engine - pro velké zátěže, které jsou spouštěny na virtuálních strojích hostovaných na infrastruktuře Google.
   - 1.2. Preemtible VMS – virtuální stroje s nízkou cenou odolné proti chybám pracovní zátěže.
   - 1.3. App engine – platforma podporující webové aplikace a mobilní pozadí aplikací.
   - 1.4. Container engine – silný clusterový manažer a systém pro kontejnery[11] (úschovny).
2. **Management**
   - 2.1. Cloud Monitoring – přehled o výkonu a dostupnosti cloudových aplikací uživatele.
   - 2.2. Cloud Deployment Manager – umožňuje vývojářům navrhnout, sdílet a spravovat komplexní Google Cloud Platformy řešení používající jednoduché šablony.

[11] Z anglického „containers“

2.3. Cloud logging – logy vytvořené cloudovými služby nebo aplikacemi, umožňující revizi, případně odhalování bugů nebo tvoření reportů.

3. **Networking**
   3.1. Cloud networking – připojení sítí pomocí VPN[12] přímo ke Googlu.

4. Big data
   4.1. Bigquery – pro analýzu velkého množství dat na cloudu, rychlá práce pomocí SQL-like dotazů s petabajty v řádu vteřin.
   4.2. Dataflow – real-time datové procesy sloužící jako stream.
   4.3. Datalab – interaktivní nástroj pro hloubkové datové analýzy a vizualizace.
   4.4. Cloud PUB/SUB – globální služba pro real-time spolehlivý přenos zpráv a datového streamu.

5. **Services**
   5.1. Translate API – na vytvoření jazykových překladů pro své aplikace.
   5.2. Prediciton API – využití algoritmů pro strojové učení k vytváření predikcí.
   5.3. Cloude endpoints – služba podporující koncová zařízení přístupných z iOS, Androidu nebo javascriptových klientů.

6. **Storage**
   6.1. Cloud storage – pro ukládání datových objektů/souborů.
   6.2. Nearline – služba sloužící pro zálohování a obnovu dat.
   6.3. Cloud SQL – ukládání a spravování dat pomocí SQL dotazů.
   6.4. Bigtable – obrovské škálovatelné NoSQL databáze.
   6.5. Datastore – pro ukládání dat bez vzájemných relací, vyznačuje se jako NoSQL služba.

Pro tyto služby jsou k dispozici dokumentace. V těchto dokumentacích je často vysvětlen postup, jak danou službu využít případně propojit programově s klientskou aplikací. Pro tyto propojení jsou často vytvořená API, která slouží jako rozhraní mezi klientskou aplikací a cloudovou službou. Tyto API jsou velice robustní a jsou neustále vyvíjeny, aby podporovaly nové možnosti systému, pro které jsou vyvíjeny a to ať už se jedná o webové aplikace nebo mobilní aplikace se systémy Android, iOS, Windows mobile.

[12] VPN – virtual private network

7.5 NoSQL Storage Service API App42

Společnost ShepHertz, nabízí mnoho cloudových služeb, které jsou zařazeny do rodiny App42 Cloud API. Ty se dělí na kategorie, které se specializují na svůj obor. Jedná se o jednu z firem, která nabízí své cloudové služby na internetu. Není nutné zde opět vyjmenovávat, které služby společnost nabízí, tento účel už splnila předchozí podkapitola, která shrnula možnosti na trhu technologií cloudových služeb.

Konkrétní **API NoSQL Storage Service**, propojuje cloudovou službu s klientskou aplikací. Ta slouží k vybudování komunikace mezi klientskou aplikací a cloudovou službou. Na stránkách společnosti je vytvořená dokumentace[24] pro toto API, která popisuje jakým způsobem inicializovat a vybudovat komunikaci pro odesílání a přijímání dat. Podporuje mnoho systémů a to i včetně systému Android.

Toto API se vyznačuje podporou přenosu dat typu JSON. Specifikováním datové struktury, už se více konkretizuje účel pro jeho využití.

7.6 Ostatní

Technologických možnosti pro způsob implementace synchronizace dat je nespočetné množství. Velký podíl zaujímají cloudové služby, které jsou v dnešní době velmi populární. Společnosti, které nabízí své produkty prostřednictvím internetu je nesporně mnoho a záleží na vývojáři, pro které se rozhodne a které služby bude využívat.

8 Návrh řešení pro synchronizaci dat

Podklady rozebrané v analýze, uvádí způsoby vedoucí k zajištění procesu synchronizace dat. Tyto informace budou využity pro návrh řešení.

Jak je specifikováno v cílu práce, synchronizace dat musí být zajištěna mezi Android zařízeními, proto bude vybrán případ z popsaných scénářů pro Android. Pro ten bude vytvořen obecný návrh řešení, který bude podkladem pro konkrétní návrh řešení. Zde budou vybrány technologie, které povede k jeho úspěšné implementaci.

Pro návrh řešení byl zvolen následný scénář: *„Firma hledá na trhu vývojáře, který pro její už zavedenou a funkční webovou službu s mapami vytvoří Android aplikaci pro prezenci geodat. Tyto geodata obsahují informace jak účelové pro potřeby na zpracování ze strany serveru, tak i prezenční, které jsou poté zobrazovány a případně i upravovány uživateli a je zde tedy nutné zajistit proces pro synchronizaci dat.“*

8.1 Rozbor scénáře

Jako prvotní zpracování scénáře, je vhodné určit prvky, které by návrh řešení měl zabezpečit. Jak lze z textu vyčíst měly by být splněny body:

1. Vytvoření Android aplikace.
2. Prezentační vrstva pro uživatele, kde budou zobrazena data
3. Možnost správy uživatelských dat v aplikaci.
   3.1. Přidání bodu.
   3.2. Odebrání bodu.
   3.3. Změna bodu
4. Zajištění synchronizace dat mezi Android zařízeními.
5. Detekce konfliktů.

Android aplikace

Aplikace pro Android jsou vyvíjeny v programovacím jazyce Java. Mají vlastní specifickou strukturu sestávající z aplikačních komponent.

- **Aktivity** – Aktivity můžeme chápat jako analogii oken či dialogů aplikace pro stolní počítač. Aktivity propojené s layouty vytvářejí uživatelské rozhraní. Ty, které nejsou propojeny, většinou tvoří aktivity ve formě dodavatelů obsahu nebo služeb.[20]
- **Dodavatelé obsahu** – z anglického content providers. Zajišťují úroveň abstrakce jakýchkoliv dat uložených v zařízení, které jsou přístupná více aplikacím. Vývojový model Androidu podporuje vývojáře, aby zpřístupnili svá data i jiným aplikacím než pouze své vlastní. Tohoto lze dosáhnout prá-

vě pomocí vytvoření dodavatele obsahu, který současně poskytuje kontrolu nad způsobem přístupu k datům.[20]

- **Služby** – Aktivity a dodavatele obsahu jsou entity s krátkou životností a lze je kdykoliv vypnout. Služby jsou oproti tomu navržené tak, aby, pokud je potřeba, pokračovaly ve své práci nezávisle na jakékoliv aktivitě. Službu můžeme použít například k detekci aktualizací RSS zdroje nebo k přehrávání hudby, která pokračuje, dokonce i když už byla příslušná ovládací aktivita uzavřena.[20]
- **Záměry** – z anglického intent, jsou to systémové zprávy upozorňující aplikace na výskyt událostí, změnami hardwarové konfigurace (například vložení SD karty) přes události související s příchozími daty (například přijetí SMS zprávy) a událostmi aplikace (spuštění z hlavního menu zařízení). Záměry je možné také vytvářet.[20]

Prezentační vrstva

Prezentační vrstva musí zabezpečit uživatelské rozhraní (přístup k datům, která aplikace obsahuje) pro uživatele. Možností je celá řada a záleží na mnoha faktorech, jak tuto vrstvu prezentovat. Důležité je, aby naplňovala plného potenciálu vytvořené aplikace a dala tak možnost uživatelům pracovat s aplikací bez nutnosti znát logiku nebo výstavbu aplikace na vrstvách nižších. Také je vhodné si uvědomit, že je to jediné co uživatel v aplikaci vidí a je dobrým zvykem dělat tato uživatelská rozhraní uživatelsky přátelská[13] ať už z důvodu obchodního (firmy), tak i z důvodu pro pozitivní zpětnou vazbu (vývojáře).

V případě, kdy se nejedná o přímé konkrétní komerční zadání, ale pouze o návrh daného řešení, je zde kladen důraz především v logice a ve výstavbě aplikace. Prezentační vrstva zde není stěžejním cílem, splňuje jen základní požadavky pro předání informací k otestování vytvořené aplikace.

Správa geodat uživatelem

Geodata jsou v Databázi Národní knihovny[19] definována jako: *„Prostorová informace o objektu nebo jevu vztažená k jeho poloze na zemském povrchu. Objekt může být definován (zobrazen) bodem, linií, plochou nebo prostorovým tělesem a je určen prostřednictvím souřadnic a topologie, tj. vztahu k poloze jiných objektů. Informace obvykle zahrnuje geometrické údaje (popis polohy, tvaru a topologických vztahů) a popisné údaje (kvalitativní a kvantitativní charakteristiky). Typickým prezentačním médiem je mapa, jež může být k dispozici ve formě analogové (např. na papíře) nebo digitální.“*

[13] User friendly

Podle scénáře je specifikováno, že se jedná o data typu geodat (geografických dat). V našem případě se bude jednat především o **bodovou** vrstvu, určenou pomocí latitude[14] a longitude[15].

V aplikaci je potřeba mít vytvořené uživatelské prostředí, kde je možné uživatelem vlastnoručně **zadat nebo změnit informace typu geodat** (minimálně latitude a longitude). Tyto data ukládat na úložiště.

Pro **smazání bodové vrstvy** z úložiště je potřeba mít možnost zobrazení všech dostupných dat a umožnit uživateli vybrání bodu, který si přeje smazat. Tuto změnu zaznamenat a smazat z úložiště.

Synchronizace dat mezi Android zařízeními

Jak už bylo mnohokrát uvedeno, pro synchronizaci dat je hlavním bodem mít stejné data uložená jak na lokálním úložišti, tak i na externím úložišti a to nejlépe ihned.

Až podle zvolených technologií, bude rozhodnuto, jak bude synchronizace zajištěna a to i s ohledem na otestování daných možností. S touto problematikou je spojeno mnoho dalších problému, které nelze ignorovat a to, ať už se jedná o velkou spotřebu baterie nebo až po velké objemy přenesených dat.

Detekce konfliktů

Konflikty mohou nastat v případě, kdy je požadován přístup k datům od více uživatelů. Jakmile budou data změněny více jak dvěma uživateli, a započne proces synchronizace, bude zde detekován konflikt. Ve zpracovávaném scénáři, jsou všechna data dostupná pro všechny uživatele.

Pro lepší popis dané situace, zvolíme případ, kdy jeden uživatel soubor změnil a druhý uživatel tento soubor smazal. Možnosti řešení:

- Novější záznam má přednost.
- Jedna akce je přednější, než jiná (změna > smazání).
- Uživatel s vyšším oprávněním má vždy přednost.
- Oznámení situace uživatelovi a rozhodnutí nechat na něm.

U konfliktů je důležité znát informace o stavbě aplikace, použitých technologií a komponent, aby bylo možné správně konflikty detekovat a následně řešit.

8.2 Návrh obecného řešení

Z informací, které jsou nyní dostupné, se vytvoří obecný návrh řešení pro daný scénář. V tomto návrhu budou popsány možné praktiky, které vedou k splnění požadavků a budou sloužit jako podklad pro návrh konkrétního řešení.

Také je nutné dodržet body, které už jsou jasně definovány. Musí být vytvořena Android aplikace, která bude skrz zvolenou technologii komunikovat přes

[14] Zeměpisná šířka

[15] Zeměpisná délka

internet s druhou stranou, umožňující proces synchronizace dat. Synchronizace je možná jen v případě připojení k internetu. Všichni uživatelé mají přístup ke všem datům a tyto data nejsou chráněna žádnými právy, můžou být přidána, změněna nebo smazána. Dalším specifickým bodem je definice přenosových dat, které jsou typu geodat. Tímhle lze tedy říci, že dané **obecné řešení se vztahuje na komunikaci zajišťující synchronizaci dat mezi více Android zařízeními zpracovávající geodata**.

8.2.1 Vývojové prostředí pro Android aplikaci

Pro vývoj lze použít jeden z nástrojů. Záleží na preferenci vývojáře. Jednou z možností je **Android Studio**. Tento nástroj pro tvorbu Android aplikací, je nabízen přímo od společnosti Google. Jedná se o vývojové prostředí[16], které obsahuje nástroje pro vývoj softwaru[17], tyto nástroje jsou přímo ve Studiu volně ke stažení. Více informací o vývojovém prostředí, dokumentaci a možnostech vývoje Android aplikací (především pro dostupná API od Googlu) jsou volně dostupná pro vývojáře na webových stránkách.[18]

8.2.2 Data

Pro data typu geodata, byl vytvořen datový formát **JSON** *(JavaScript Object Notation)*, který se vyznačuje svým zápisem tak, že je čitelný i pro člověka a je vyjádřen ve formě páru atributu a jeho hodnoty. Jedná se o javascriptový objektový zápis nezávislý na počítačové platformě, určený pro přenos dat, která mohou být organizována v polích nebo uložena v objektech. Vstupem je datová struktura typu čísla, řetězce, boolean, objektu nebo z nich složených polí. Výstupem je vždy řetězec.[22]

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
        "streetAddress": "21 2nd Street",
         "city": "New York",
        "state": "NY",
         "postalCode": "10021-3100"
  },
  "phoneNumbers": [
        {
```

[16] IDE – Integrated Development Enviroment

[17] SDK – Software Development Kit

[18] http://developer.android.com/index.html

```
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
   "spouse": null
}
```

Kód 2. Ukázka JSON struktury popisující osobu[22]

Předchůdcem pro zasílání informací mezi klientem a serverem byl značkovací jazyk **XML** *(Extensible Markup Language)*. Jeho struktura je také pro člověka čitelná, rozdílem od JSONu je, že XML používá vyšší počet značek, čímž zvyšuje režii při přenosu dat.

Určením dat a datové struktury, které budou mezi úložišti posílány, se zpřesňuje konkrétní implementace. Až nyní po určení datové struktury, je jasně dané, že aplikace musí podporovat přenos dat typu JSON.

Parsování dat[19]

Jak lze vidět z ukázky pro Kód 2, tyto data lze v jisté míře člověkem přečíst a odvodit jejich význam. Nicméně pro pohodlné vytváření a sestavování JSON dokumentů, je vhodné vytvořit systém, který slouží k parsování těchto dat. Poté je možné programově zpracovávat data například jen voláním vhodné metody pro parsování (zpracování dat ze vstupu od uživatele/sestavení dat pro prezentaci dat uživatelovi).

JSON datová velikost

Velikost dat pro JSON se může měnit podle jeho struktury, poté záleží na vývojáři/zadavateli, které informace si přeje přenášet. Jak lze vidět v příloze B, kde je ukázka reálných dat pro JSON z praxe, může se jednat o velice komplexní strukturované informace. Tyto informace jsou přenášeny přes internet a je zde kladen důraz na co nejmenší velikost, která vede k rychlejšímu přenosu.

Pokud JSON uložíme do textového souboru (txt), bude velikost rovna 3 500B včetně mezer. Při práci s JSON, se využívá zápis sériový, neboli všechen text je bez mezer, tímto se velikost sníží na výsledných 1 800B. Při komunikaci jsou často na JSON nabaleny ještě režijní informace navíc, lze tedy počítat s 2kB. Pro dnešní dobu se jedná o velice nízkou hodnotu. To je také důvod, proč se JSON dostává velké oblibě a je využíván pro komunikaci mezi koncovými za-

[19] Z anglického slova parse, v IT volně přeloženo jako „dolování dat“

řízeními a databázemi. Při této komunikaci je jeden JSON pouze jedním řádkem databáze, která může obsahovat velký počet záznamů, které poté násobí daný přenos dat.

8.2.3 Lokální úložiště

Je určeno zpracovávaným zařízením. Řešení je jasně definováno pro systém Android. Ten umožňuje ukládat data pomocí:

- **Sdílených preferencí**[20] – používá se pro ukládání privátních dat, nejčastěji se jedná o nastavení aplikace preferované uživatelem.
- **Interní a externí úložiště** – jedná se o *filesystem*, kde se ukládají datové soubory.
- **SQLite Databáze** – ukládání strukturovaných dat do privátní vestavěné databáze pro systém Android na základě *SQLite* databáze, která zpracovává SQL dotazy dle standardně zavedených zvyklostí. Výhodou oproti ukládání dat do souborové struktury je fakt, že programově je nutné při každém čtení dat otvírat datový soubor a provádět v cyklu čtení po jednom znaku.

8.2.4 Externí úložiště

Pro externí úložiště je možno využít **server** nebo **cloud**. Jsou vyznačovány jako prostředník nebo centralizovaná část, přes kterou probíhá komunikace od koncových zákazníků.

Jak lze z předešlých textů odhadnout, některé technologické prvky jsou jasnou volbou pro jejich použití. Jak lze z vývoje technologie i trhu vidět, cloudové služby jsou jasným hitem mezi technologiemi pro ukládání dat. Jelikož se jedná obecný návrh, je zde stále prostor pro obě možnosti, v tomhle případě se jedná pouze jako doporučení použití technologie cloudu. Také je důležité brát v potaz vývojáře/zadavatele a jejich preference nebo možnosti.

Externí úložiště je možné používat jako ukládací prostor ve formě dat ukládaných na diskové úložiště nebo pro ukládání dat do databází. Pro strukturovanou formu typu dat JSON, je vhodné volit ukládání do databází.

8.2.5 Synchronizace dat

Proces synchronizace mezi danými úložišti lze řešit pomoci technologií jmenovaných v kapitole 7. Bude řešena na pozadí aplikace (background) a to buď voláním *callback* metod nebo implementací asynchronními třídami rozšiřující Android aktivity, které dovolují procesy zpracovávat mimo hlavní vlákno aplikace, kde se právě realizuje uživatel a pak tedy není tímto řešením nijak omezován.

Jelikož je synchronizace vyžadována v co nejkratším čase, je potřebné držet komunikaci mezi úložišti a zajišťovat tak proces synchronizace. Nejideálnějším

[20] Z anglického „Shared Preferences“

způsobem se zde jeví vytvořit Službu[21], která se bude právě o tento proces neustále starat.

Komunikace potřebná pro zajištění synchronizace dat z lokálního úložiště do cloudu a naopak je nejčastěji řešena pomocí **API** od společnosti, která nabízí cloudovou službu. Pokud nenabízí přímo API, tak by společnost měla minimálně určit způsob, podle kterého je možné vytvořit komunikaci s její databázovou službou (například sepsanou dokumentací).

Pro úspěšné navázání spojení, je toto spojení nutné inicializovat. Inicializace zajišťuje identifikaci API pro uživatele, který cloudovou službu využívá (aby společnost věděla, které API použít a kterému uživateli zaúčtovat). V manažerském módu API musí být dostupný určitý identifikátor, který je poté použit v klientské aplikaci. Tento identifikátor je většinou tvořen klíčem, příhodné je pojmenování tohoto identifikátoru je „apiKey“. Při vytvoření tohoto klíče se automaticky vytvoří i klíč tajný „secretKey“. Oba tyto klíče jsou předávány při inicializaci komunikace v klientské Android aplikaci.

Změny od uživatele

Uživatelé klientské aplikace mají možnost **vytvářet, měnit, a mazat** jednotlivé body (řádky) z databáze. Jakákoliv taková změna se musí projevit na obou stranách úložiště. Daná synchronizace je pak dána právě podle možností dostupných v API. Uživatelé podle scénáře mají přístup ke všem bodům uložených v cloudu, nejsou zde bezpečnostní omezení v podobě práv. V tomto případě není nutné vytvářet organizaci v podobě přihlášení a autorizace uživatele.[22] Jednoduše řečeno, každý uživatel aplikace má právo vytvářet body a právo mazat kterýkoliv bod. Úkolem procesu synchronizace dat, je držet tyto data na všech zařízeních a cloudu stejné.

Změny od centralizovaného prvku

Kdykoliv změní-li, vytvoří nebo smaže uživatel ve své aplikaci záznam, měl by být zavolán proces synchronizace, který danou změnu zavede na centrálu.

Tímhle vzniká pro všechny ostatní uživatele informace, kterou jejich lokální databáze nezahrnuje a jejich zařízení by mělo zajistit proces synchronizace dat od centrály.

Zde je dobré podotknout, možné porušení integrity databáze. Tento jev se řeší až při použití konkrétního API a jeho možností. Někdy je vhodnější omezit některé chtěné vlastnosti typu velikosti posílaných dat, rychlosti nebo zatížení zařízení pro zachování integrity a tedy docílit totožné reprezentace databáze mezi klientem a centrály.

---

[21] Service

[22] Pro potřeby této práce možné. V komerčním řešení je silně doporučeno autorizaci provádět, určit práva uživatele a sledovat nedobrou vůli uživatele (záškodné účty).

Proces synchronizace

Celý proces synchronizace je natolik komplexní a natolik závislý na různých proměnných, že nelze jednoznačně určit, jak často má probíhat, jakých datových velikostí může maximálně dosahovat a to vše ještě v závislosti na zátěž pro zpracování tohoto procesu na koncových Android zařízení s pohledu využití baterie, vytížení CPU a paměti.

Jsou ale dány dobré praktiky, které je vhodné dodržovat a následně laděním, testováním a optimalizací vylepšovat. Jako dobrá rada, je určení velikostí, které může aplikace vygenerovat komunikací na základě velikosti datového prvku, velikosti přenášených záznamů (počtu záznamů v databázi) a to vše v závislosti na čase. Poté lze definovat:

- Datový prvek JSON, $d = 2kB$.
- Počet záznamů v databázi, $q_x = 1, 100, 1000, 10000$.
- Proces synchronizace za hodinu, $t_y = 60\ (1min.), 1\ (1hod.), 0{,}25\ (4hod.)$.
- Velikost dat v závislosti na čase, $D_{x,y} = d * q_x * t_y [kB/hod.]$.
- Kde platí, že $x, y \in N$[23] .

Díky tomu lze psát:

- $D_{1,1} = 2 * 1 * 60 = 120[kB/hod.]$
- $D_{1,2} = 2 * 1 * 1 = 2[kB/hod.]$
- $D_{1,3} = 2 * 1 * 0{,}25 = 0{,}5[kB/hod.]$
- $D_{2,1} = 2 * 100 * 60 = 12\ 000[kB/hod.]$
- $D_{2,2} = 2 * 100 * 1 = 200[kB/hod.]$
- $D_{2,3} = 2 * 100 * 0{,}25 = 50[kB/hod.]$
- $D_{3,1} = 2 * 1000 * 60 = 120\ 000[kB/hod.]$
- $D_{3,2} = 2 * 1000 * 1 = 2\ 000[kB/hod.]$
- $D_{3,3} = 2 * 1000 * 0{,}25 = 500[kB/hod.]$
- $D_{4,1} = 2 * 10\ 000 * 60 = 1\ 200\ 000[kB/hod.]$
- $D_{4,2} = 2 * 10\ 000 * 1 = 20\ 000[kB/hod.]$
- $D_{4,3} = 2 * 10\ 000 * 0{,}25 = 5\ 000[kB/hod.]$

Na výsledky těchto údajů je vhodné se odkazovat a řídit se jimi. Díky tomu má poté vývojář určitou představu o velikosti probíhané komunikace mezi zařízením a databází.

Existuje zde možnost přenášené textové informace v aplikaci komprimovat a poté poslat, čímž by se dosáhlo snížení přenosové komunikace, ale za cenu zvýšeného využití procesoru na zařízeních a tedy i vyšší spotřeby baterie. A to

[23] N, vyjadřuje posloupnost přirozených čísel $(1, 2, 3, \ldots)$.

vše ještě za podmínky, že je možnost na cíleném centralizovaném prvku tyto informace dekomprimovat.

8.2.6 Detekce konfliktů

V případě přidání bodu uživatelem. Uživatel vytvoří novou bodovou vrstvu. Tento bod se nyní musí uložit do lokálního i centrálního úložiště. Z logiky akce, je předpokládáno uložení na centralizovaný prvek a poté až na lokální zařízení. Pokud se jedná o záznam, který obsahuje časové razítko, musí být toto časové razítko stejné na obou úložištích. Jestliže tento proces uložení proběhne bez neočekávaného stavu, je celý proces považován za korektní a není tedy v tomhle případě nikdy detekován konflikt.

V případě smazání bodu (řádku databáze) uživatelem. Uživatel na zařízení vybere bod, který chce smazat. Tento bod je na centrálním i lokálním úložišti jasně definován identifikátorem. Jako logické rozhodnutí pro algoritmus je smazání bodu prvotně na centrálním úložišti a poté až při odpovědi úspěšného smazání, proběhne smazání na lokálním úložišti. V tom případě má uživatel, který inicializoval smazání bodu, data synchronizována. Ostatní uživatelé, mají v tuto chvíli rozdílná data na lokálním a centrálním úložišti právě o tento jeden smazaný bod a to do doby, než bude spuštěn proces, který kontroluje, jestli jsou data na obou úložištích totožné (v synchronním stavu) a výsledkem zde tedy bude, že se smaže daný bod u ostatních lokálních úložišť. Jestliže, ale bude jeden z ostatních uživatelů měnit zrovna bod, který byl právě smazán z centrály a tuto změnu odešle, je detekován konflikt. Tento konflikt bude detekován, jako nemožnost porovnání unikátního identifikátoru záznamu z lokálního úložiště s centrálním úložištěm. Doporučené možnosti pro řešení konfliktu:

- Bude nastavena priorita pro smazání větší než pro změnu. Změna nebude zaznamenána a daný bod bude smazán z lokálního úložiště, odkud byla změna inicializována, tak aby byla zabezpečena synchronizace dat.[24]
- Novější informace přebíjí informaci starší. V tom případě, bude změněný záznam přidán do procesu pro vložení bodové vrstvy a je nutné zabezpečit, aby byly známy všechny jeho vlastnosti. Uložení musí proběhnout pod stejným identifikátorem. Na všech ostatních zařízeních bude nahrán, jakmile proběhne proces synchronizace.
- Rozhodnutí konfliktu bude ponecháno na uživateli, který způsobil konflikt. Data, která chtěl změnit, musí být drženy na lokálním úložišti do doby, než uživatel rozhodne, jestli chce daný záznam uložit nebo zahodit.

V případě změny uživatelem. Uživatel změní záznam pro bodovou vrstvu. Je tedy okamžitě zahájen proces pro uložení změny. Z logiky akce, je předpokládáno uložení změny na centralizovaný prvek a poté až na lokální zařízení. Změna je zaznamenána změnou poslední změny záznamu. Jestliže bude proces úspěš-

[24] Jedná se o nečekaný stav a uživatel by měl být informován.

ný, ostatní uživatelé budou v příštím cyklu synchronizace detekovat změnu právě pro tento bod. V případě, že nastane situace, kdy jeden z ostatních uživatelů změní právě tento bod, ještě před zahájením procesu synchronizace, měl by být detekován konflikt. Předpokládáme, že změna je vyjádřena změnou atributu poslední změny (na aktuálnější datum). Je tedy nutné ještě před změnou porovnat hodnoty atributů poslední změny, pokud budou jiné, pak je konflikt detekován. V tom případě se nabízí možnosti:

- Novější informace (změna) má přednost před starší změnou.[25] Změna bude přepsána.
- Novější změna, bude přidána ke změně starší. Možné jen v případě, že je daná možnost logicky uskutečnitelná.
- U uživatele, který prováděl novější změnu, bude konflikt ohlášen a rozhodnutí bude ponecháno na něm. Tato změna musí být držena na lokálním úložišti, dokud není uživatelem rozhodnuto o jejím osudu nebo vypršením časového limitu pro rozhodnutí (v případě, že je nastaveno).

Detekované konflikty není vhodné řešit zahozením novější informace. Minimálně ne bez uvědomění uživatele, který je za daný konflikt zodpovědný.

8.3 Návrh řešení synchronizace dat s implementací API App42

Bude vycházet z obecného návrhu řešení. Je zde nutné identifikovat technologie, které budou pro implementaci použity. Určení způsobu procesu synchronizace dat.

Finální návrh řešení, které bude řešit daný rozebraný scénář a to za použití cloudové služby **NoSQL Storage**, především kvůli specializaci na data typu JSON a dostupného API pro systém Android.

Android

Pro vývoj aplikace, bude použit programovací jazyk Java. Jako vývojový nástroj Android Studio. Technická část ladění[26] a testování aplikace bude prováděna na tabletu PiPO U9T[27] místo emulátoru dostupném v Android Studiu.

Data

Posílaná data do cloudu jsou typu JSON. Hlavní priorita aplikace není v rekonstrukci JSONu, tedy není nutné jej parsovat a upravovat rozložený. V aplikaci

[25] Může být matoucí pro uživatele, který udělal změnu jako první, při kontrole už může být změněná informace přepsána (pokud proběhne proces pro kontrolu změn na centralizovaném prvku).

[26] Z anglického „debug“

[27] Specifikace tabletu dostupné na http://www.devicespecifications.com/en/model/8bae2d41

bude vytvořena ukázka z reálných dat, která bude doplněna pouze o jednu z hodnot, ke které budou ještě přidány body lokace. Důležitým prvkem je zde pracovat s reálnými daty, pro zjištění velikosti přenášených dat v ostrém provozu při synchronizaci dat.

Prezentační vrstva

Android aplikace bude obsahovat uživatelské rozhraní pro přidání jednoho záznamu, kde bude uložena lokace (zeměpisná délka a šířka) společně s JSONem. Po vytvoření bodu uživatelem, jej bude možné odeslat pomocí tlačítka.

Uživatel musí mít přístup ke všem datům, k tomu je potřeba aktivita, která vypíše obsah všech dat. V Android aplikaci lze pro tuto možnost použít listový náhled na položky (řádky) databáze, pro přehlednější zobrazení bude vytvořen vlastní řádek.

Uživatel má také možnost vybrat který řádek chce smazat. Po zobrazení všech položek, může kliknutím na daný bod rozbalit detailnější zobrazení, kde má možnost tento záznam smazat. Bude-li tento bod smazán, nebude už pak v databázi existovat a není tedy možné, aby byl zobrazen v listové prezentaci dat.

Lokální úložiště – Android

Řešeno SQLite databází. API na cloudu ovlivní, jak bude vypadat fyzická lokální databáze. Databáze se skládá z:

- názvu databáze
- názvu tabulky

Tabulka obsahuje atributy s názvy sloupců a je číslována od nuly, kde je defaultně nastaven sloupec „_id“. Viz:

0. sloupec určuje id záznamu „_id“
1. sloupec určuje poslední změnu záznamu „updated_at“
2. sloupec určuje vytvoření záznamu „created_at“
3. sloupec určuje hodnotu id dokumentu na cloudu „doc_id“
4. sloupec určuje zabalený JSON pro jeden záznam „json“
5. sloupec určuje zeměpisnou šířku „latitude“
6. sloupec určuje zeměpisnou délku „longitude“

Externí úložiště – Cloud

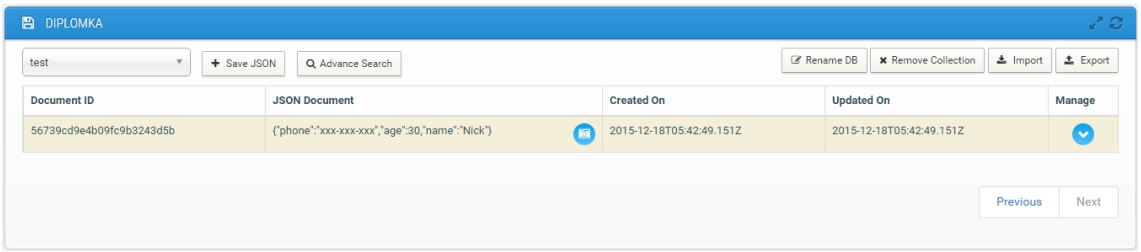
Jako ideální **řešení pro aplikaci** se jeví **Databáze jako služba (DBaaS)**. Pro konkrétní řešení nabízené od nějaké komerční společnosti, by mělo být nabídnuto minimálně rozhraní pro správu této databáze, případně i další navazující funkce.

Z informací zjištěných v teoretické části, víme, že tyto služby jsou na trhu dostupné. Liší se provedením, uživatelským rozhraním (zprostředkovaným webovou aplikací) a také cenou za nabízené služby.

Důležitým bodem pro uživatele cloudové služby, je způsob, jakým je řešen přístup z pohledu vývojáře, neboli jakým způsobem je možné zajistit komunikaci s klientskou Android aplikací. Pro dobrou pověst společnosti s ambicemi prosadit se na trhu je dodávat pro zajištění komunikace API a to i s dokumentací, kde jsou popsány způsoby jak naimplementovat potřebné prvky. Tyto API se samozřejmě liší podle platforem a systémů, pro které jsou určeny a v kterém programovacím jazyce jsou vyvíjeny.

A právě velice přehledná dokumentace, srozumitelné implementační kódy a podpora Androidu byly hlavním důvodem pro zvolení **NoSQL API od App42**. Pro správu databáze, je nutné se zaregistrovat na jejich webových stránkách[28]. Po přihlášení je poté k dispozici uživatelské rozhraní přes web, které umožní spravovat cloudovou službu. Zde jsou k dispozici různé funkce. Nezbytnou součástí pro implementační řešení jsou oba klíče potřebné k propojení klientské aplikace s touto cloudovou databází, které jsou zde v nastavení uvedeny.

Jedna z funkcí je i prohlížení dat v databázi. Jak lze vidět z Obr. 6, je zde databáze pojmenovaná „diplomka“ a kolekce pojmenovaná „test“. Jsou zde pevně definované sloupce a to Document ID, JSON Document, Created On, Updated On. V detailu se poté ještě nachází atribut pro ukládání lokace a vlastníka.



Obr. 6 Ukázka cloudové databáze z webového rozhraní

Synchronizace dat

Zpracování procesu synchronizace bývá časově náročnější, proto musí běžet na pozadí, aby nijak neomezoval uživatele při používání aplikace. Je možné uživatele pouze informovat o dokončení procesu notifikací (Toast[29]).

Pro zajištění komunikace je nutné inicializovat API v kódu aplikace, následně vybudovat spojení voláním metody buildStorageSerivce() datového typu StorageService (zajištěno v API). Od tohoto bodu je navázána komunikace mezi aplikací a cloudem.

Přidání bodu – v aplikaci vytvoříme bodovou vrstvu, tím dosáhneme pouze vytvoření hodnot pro atributy JSON, latitude a longtitude. Samozřejmě

[28] https://apphq.shephertz.com/register

[29] Toast – oznámení vyskakující uživatelovi na obrazovku, implementačně možné pro Android

by bylo možné s určitým algoritmem vytvořit stejný formát definován cloudem pro atributy času vytvoření a poslední změny. Jenže čas vytvoření v lokální databázi by se pak neshodoval s vytvořeným časem na cloudu a byla by porušena integrita databáze. Při poslání jakéhokoliv dotazu (akce) na cloud, je zpětně vracena odpověď pro dané API. Je zde tedy možnost, uložit prvně data do cloudu a ze zpětné odpovědi sestavit data pro lokální databázi vytvořeného bodu a uložit stejná data jak na cloudu, čímž docílíme integrity a zároveň i synchronizace (pro tento bod).

Smazání bodu – v aplikaci uživatel smaže jeden z bodů v databázi. Tento bod je poté odstraněn z databáze a následně je volána služba pro smazání tohoto bodu z cloudu. Jelikož musíme mít v databázi uložen i doc_id, stačí jen poslat s mazacím příkazem jeho hodnotu a bod (řádek) je z cloudu smazán. Jako odpověď je potom poslána zpráva zabalená jako JSON, v kterém jsou informace o provedené akci (úspěšné/chyba). Tímhle je synchronizace dat zajištěna.

Úprava bodu – není přímo nutné implementovat. V případě, že bude chtít uživatel daný bod změnit, tak ho může smazat a vytvořit jiný podle svých preferencí, čímž daný bod změní. Vytvoření bodu a smazání bodu už je logicky vyřešené, takže bude zachována synchronizace včetně integrity databáze.

Synchronizace dat od cloudu – doposud jsou vyřešeny všechny body, které byly měněny uživatelem. Uživatel, ale také může aplikaci využívat jen pro informační činnost bez žádné interakce a pouze sledovat záznamy v databázi. V tom případě, by se právě teď v aplikaci nezobrazily nové body, které byly přidány na cloud od jiných uživatelů. Je tedy nezbytné ještě zajistit proces kontroly nových bodů (případně změn) na cloudu. Můžou nastat různé situace, které je před implementací vhodné popsat právě teď při návrhu řešení, viz:

- **Přidání bodu jiným uživatelem** – stav je takový, že databáze na lokálním úložišti je rozdílná o bod, který je uložen navíc na cloudu. Daných řešení je pro tento problém více. Jedním z nich je porovnání posledního (nejnovějšího) updatu ze sloupce updated_at v lokální databázi se všemi body ze sloupce updated_at na cloudu. Každá vyšší hodnota na cloudu, poté znamená nový nebo změněný bod (v tomto případě by se mělo jednat o jednu hodnotu nového bodu). Pro jistotu, že se nejedná o změněný bod, se ještě porovná identifikátor doc_id z cloudu se všemi identifikátory doc_id z lokální databáze. Pokud budou pro všechny rozdílné, jedná se o nový bod. Pokud bude doc_id stejné, znamená to, že se jedná o změněný bod. Tenhle algoritmus tedy řeší nejen přidání bodu, ale také jeho změny. Při správném postupu, může algoritmus řešit i více změn než jeden bod.
- **Smazání bodu jiným uživatelem** – znamená, že na cloudu je méně záznamů, než na lokálním úložišti. To zjistíme porovnáním počtu záznamu obou úložišť. Informace o smazaném záznamu se nikde nezaznamenává. Tedy nevíme, který jeden záznam na lokální databázi přebývá. Jedinou možností je porovnání všech lokálních doc_id se všemi cloudovými doc_id. Algoritmus by měl postupně porovnávat od prvního lokálního doc_id se všemi cloudovými doc_id, pokud najde shodu, přeskakuje na další lokální

doc_id a opět porovnává se všemi řádky z cloudu. Až nastane situace, kdy nebude nalezena shoda, tak je tento lokální záznam smazán a tím zajištěna synchronizace dat pro smazaný bod. Pokud je těchto bodů více, je nutné tento proces neukončovat a dokončit celý, protože zde můžou být i další nesmazané body a jelikož nikdy nemůže s jistotou určit, že nebyl smazán bod, je nutné tento cyklus projet pokaždé a to až do konce.

Data na úložištích chceme mít v nejlepším případě neustále ve stejném (synchronním) stavu. Takový stav by znamenal neustálou kontrolu mezi komunikujícími úložišti. Tento proces by muselo zajišťovat zařízení, na kterém je aplikace spuštěna. To by vedlo k neustálému vytížení procesoru, a to by vedlo ke značně zátěži na baterii, nemluvě o neustálém přenosu dat. Proto je pro synchronizaci často volen určitý kompromis. Pro některé implementace se může opakující proces synchronizace pouštět v řádu hodin a u některého v řádu minut. Stěžejním je jak moc důležité je mít tyto data synchronní a také je důležité vědět, kolik uživatelů danou aplikaci používá.

Pro daný návrh řešení, kde chceme zpracovávat geodata a tyto data prezentovat uživatelům do aplikace, zde může být požadavek na aktuálnost důležitý. Proto by měl být nastaven na kratší časový úsek (30s-60s). Značný vliv zde bude mít pozdější testování, kde se z pokusných reálných dat udělá určitý přehled, který poodkryje, jak jsou diskutované problémy na sobě závislé. Je tedy nutné v Android aplikaci zajistit službu, která bude neustále provádět synchronizaci dat i v případě, kdy uživatel pracuje mimo aplikaci.

Detekce a řešení konfliktů

Vycházíme z poznatků, které byly navrženy v obecném řešení. Popisujeme konkrétní situace, které mohou nastat.

Přidání bodu uživatelem, jak bylo uvedeno v obecném návrhu, pro danou akci konflikt nevzniká.

Smazání bodu uživatelem, bod je zaslán na cloud s hodnotou doc_id pro určení daného záznamu.

- V případě, že tam tento doc_id neexistuje, je detekován konflikt, který znamená, že daný bod už smazal někdo jiný z uživatelů. V tom případě je odeslána odpověď od cloudu, že akce dopadla neúspěšně. Řešením je smazání pouze z lokálního úložiště zařízení, čím je konflikt pro tuto situaci vyřešen a data (z pohledu tohoto bodu) jsou synchronizována.
- V případě, že byl chtěný mazaný bod změněn (nutno programově zajistit kontrolu pro porovnání posledních změn na cloudu a na lokálním úložišti). Jestliže se bude časové razítko lišit, je detekován konflikt a to změnou chtěného mazaného bodu. Řešením je danou situace oznámit uživateli, jako detekci konfliktu a informovat ho o změně daného záznamu. Spustit proces pro kontrolu změněných dat na cloudu. A poté se uživatele zeptat, jestli má zájem daný bod smazat. Tímto je konflikt vyřešen.

Změna bodu uživatelem je pro tento návrh řešení, řešena smazáním bodu a následným vytvořením. Pro tyto případy už jsou konflikty vyřešeny, tím pádem zde neznámý konflikt není detekován.

9 Implementace pro API App42

V implementační části je cílem navrhnuté řešení převést do reálné podoby a to ve formě aplikace zajišťující popsané a definované funkce. Jako vhodné prvotní kroky při začátku implementace, je vytvořit v aplikaci prvky, které aplikace musí obsahovat a jejich implementace ulehčí práci vývojáři, protože díky nim bude mít zpětnou vazbu při implementaci dalších prvků. Jedná se o prezentační vrstvu ve formě grafického uživatelského rozhraní. Z navrhnutého řešení je jasně dáno, že aplikace bude obsahovat obrazovky s přidáním bodu a zobrazením všech bodů. Je tedy vhodné vytvořit hlavní obrazovku, která bude sloužit k přístupu k daným prezentačním aktivitám.

V průběhu implementace, může dojít k situacím, že navrhnutá řešení, jsou teoreticky správná, ale implementačně náročná. V některých případech omezená technologiemi nebo nemožná. To může být způsobeno nečekaným chováním komponent, nedostatečnými možnostmi API, vysokou latencí ze strany cloudu a podobnými neočekávanými událostmi. Pro finální řešení je tedy důležité volit kompromisy a držet se důležitých bodů pro splnění cíle.

Každá aplikace by měla mít určitou míru spolehlivosti. Vývojář by se měl snažit dělat aplikaci robustní, vůči neočekávaným stavům a tyto stavy odchytávat a řešit interně programově v aplikaci tak, aby aplikace nehavarovala. Maximálně může uživatele informovat, že daný proces nedopadl dle očekávání. Také by v aplikaci nikdy neměla nastat možnost, kdy se uživatelovi zobrazí špatná data, která nejsou korektní s akcí uživatele.

Grafické uživatelské prostředí

Zastřešuje prezentační vrstvu, jejím cílem je snadná práce s aplikací. V aplikaci je hlavní obrazovka, z které je možné se dostat pomocí tlačítek do tří částí:

- Přidání bodu
- Zobrazení všech bodů a detail pro vybraný bod, kde je možné daný bod smazat.
- Reporty, kde jsou zobrazeny informace ohledně synchronizace dat.

Přidání bodu je řešeno jednoduchými grafickými prvky Androidu a to *textView* a *editText*. Pokud uživatel zadá korektně body latitude, longitude, json a odešle pomocí tlačítka, jsou body zpracovány.

Zobrazení všech bodů je řešeno na základě SQLite databáze, která využívá pomocnou třídu *DatabaseHelper*, kde je programově vytvořený předpis pro vytvoření databáze. V případě, že uživatel klikne na tuto možnost, je přepnut do aktivity *ListActivity*, kde mu je zobrazen listový záznam pro všechny body na lokální databázi. Každý záznam zobrazuje id, čas vytvoření, čas poslední změny, identifikátor na cloudu, stringový zápis JSON, latitude a longitude. Každý záznam je speciálně zobrazen, podle předepsaného *row*, pro atraktivnější pohled daného zápisu. Jak lze vidět z Obr. 7, jsou zde zobrazeny všechny atributy z lo-

kální databáze. Vlevo je zobrazeno id společně s poslední změnou záznamu a čas vytvoření. Uprostřed je nahoře částečně zobrazen JSON řetězec a pod ním je identifikátor dokumentu určený cloudem. Napravo jsou zobrazeny lokační body latitude a longitude.

Obr. 7 Zobrazení pro data v ListView, prezentována vlastním řádkem přímo v zařízení

Smazání bodu, v případě, že uživatel vybere jeden bod z vybraných záznamů, zobrazí se mu *DetailActivity*, kde je zobrazen celý řetězec pro daný JSON a tlačítko pro smazání. V případě, že uživatel smaže bod nebo se jen vrátí zpět do listové aktivity, bude *ListView* aktualizováno, tak aby nebyl smazaný bod už viditelný, protože neexistuje.

Synchronizace od uživatele

Uživatel má možnosti jak v aplikaci svojí akcí ovlivnit data na cloudu. Tyto změny jsou naimplementovány, s okamžitou změnou jak na daném zařízení, tak i na cloudu.

Přidání bodu, jakmile jsou data uživatelem odeslána tlačítkem, je jejich obsah přebrán v aplikaci. Struktura lokální databáze z návrhu řešení vyžaduje uložení hodnot 6 atributů. V tom případě nyní chybí uživatelovi tři hodnoty, které jsou dostupné pouze na cloudu a to created_at, updated_at, doc_id.

Řešení je naimplementováno uložením bodu prvotně na cloudu. V tom případě je docíleno kompletního zápisu. Hodnoty, které chyběly, jsou cloudem vygenerovány.

API v tomhle případě pracuje s vybudovanou komunikací a při uložení bodu do cloudu, je vrácena odpověď s výsledkem akce včetně kompletního záznamu. Proto není potřeba, aby byl posílán další požadavek na cloud. Z odpovědi jsou programově vytaženy všechny potřebné informace a ty jsou uloženy lokální databáze zařízení a proces přidání bodu je ukončen. Dané řešení, kdy se pracuje pouze s a jedním požadavkem na cloud o velikosti jednoho záznamu, je v tomhle případě velice efektivně vyřešen a to včetně docílení synchronizace přidání bodu uživatelem. Viz ukázka Kód 3.

```
gp.setLat(new BigDecimal(mLat));
gp.setLng(new BigDecimal(mLng));
storageService.setGeoTag(gp);
StorageService storageService = App42API.buildStorageService();
Storage storage = storageService.insertJSONDocument
(Constants.CLOUD_DB_NAME, Constants.COLLECTION_NAME, mJson);
Storage.JSONDocument oneJsonDoc = storage.getJsonDocList().get(i);
```

Kód 3. Vybudování komunikace a vytvoření jednoho JSON dokumentu.

Smazání bodu, uživatel vybere v aplikaci bod pro smazání. Jakmile se rozhodne a klikne na tlačítko smazat, vyskočí ještě upozornění s ověřením, že je uživatel jistý s odstraněním tohoto bodu. Tento prvek je vhodné v aplikaci zobrazovat, neboť se jedná o nezměnitelnou akci a jedná se o poslední upozornění uživatele. Jakmile souhlasí, spustí se proces pro smazání. K bodu máme potřebné informace. Jako unikátní identifikátor zde slouží doc_id, který je tvořen automaticky na cloudu. Zahájí se proces pro smazání záznamu. Použitím hodnoty tohoto identifikátoru se prvotně smaže záznam z lokální databáze a v procesu je i voláno asynchronní třídy pro smazání bodu na cloudu a to *AsyncDelRow*, která zajistí smazání určeného bodu požadavkem *deleteDocumentById*, ten navrací i odpověď ve formě JSON, kde informuje o výsledku akce, viz Kód 4. Smazáním z obou úložišť je zajištěna synchronizace dat.

```
I/AsyncDelRow:{"app42":{"response":{"success":true,"storage":{"dbN
ame":"diplomka","collectionName":"test","jsonDoc":{"_id":{"$oid":"5682
e978e4b083980985216e"},JSON_SAMPLE"_$owner":{"owner":"Roman"},"_$creat
edAt":"2015-12-29T20:13:44.123Z","_$updatedAt":"2015-12-
29T20:13:44.123Z","loc":[15,-22]}
```

Kód 4. Ukázka JSON odpovědi cloudu na akci delete

Synchronizace od cloudu

Při implementaci algoritmu pro kontrolu nového bodu na cloudu App42. Byly zjištěny zásadní informace. Jediná možnost, jak z cloudu dostat všechny záznamy pro *updated_at*, je možnost použití metody *findAllDocuments*. Problémem, je že tato metoda stahuje celý obsah pro každý záznam (pole JSON dokumentů). A tedy daný algoritmus navržený pro kontrolu jen určitých bodů je zbytečný, protože přes komunikační spoj, už prošel celý obsah databáze na cloudu. V API

nebyla nalezena vhodnější metoda. Existují zde metody, které zpracovávají jeden konkrétní záznam, například tak, jak byla řešena komunikace od uživatele. Tento jeden záznam obsahuje všechny hodnoty, a pokud budeme vyžadovat všechny řádky, jsme opět na maximu celé databáze a tento problém tímto způsobem obejít nelze.

Kontrola přidaného bodu nebo smazání bodu, je tedy poté zabezpečena stažením celé databáze a přepsáním databáze na lokálním úložišti. Pro řešené API nebylo nalezeno jiného možného řešení. Nicméně to má výhodu zabezpečení integrity databáze. Data budou prezentovány totožně jak na lokálním zařízeni v databázi, tak i na cloudu. Tím je zajištěna synchronizace dat.

```
try {
    Storage storage = storageService.findAllDocuments(
            Constants.CLOUD_DB_NAME, Constants.COLLECTION_NAME);
    jsonDocList = storage.getJsonDocList();

    Log.i(TAG, "Velikost databaze> " + jsonDocList.size());
    return jsonDocList;
} catch (Exception ex){
    System.out.println("Exception Message> "+ex.getMessage());
}
```

Kód 5. Ukázka stažení záznamů z cloudu včetně odchycení a ošetření výjimky

Řešení konfliktů

Řešíme konflikty definované v návrhu řešení. V obou případech se jedná o řešení konfliktu pro mazání bodové vrstvy uživatele.

- Mazaný bod uživatele, už je na cloudu smazán a je tedy vracena neúspěšná odpověď od cloudu. V lokální databázi na zařízení tento záznam smaž. Uživatel bude informován o detekci konfliktu zapsáním reportu do *ReportsActivity*.
- Mazaný bod uživatele, byl na cloudu změněn. V aplikaci je prováděna kontrola této změny, v případě pozitivního nálezu, je uživatel informován o této situaci zapsáním reportu do *ReportsActivity*, včetně unikátního identifikátoru *doc_id*, díky čemu má možnost tento záznam dohledat. Tento záznam není smazán, a proto je informován i notifikací typu *Toast* v dlouhé periodě zobrazení.

Služba

Android umožňuje vytvořit aplikační komponentu, která se nazývá *Service*. Tato služba je využívána pro dlouhodobé operace, které běží na pozadí a nejsou přístupné uživatelovi prostřednictvím uživatelského prostředí aplikace.

Služba v aplikaci zajišťuje proces pro kontrolu změn na cloudu. Tento proces je popsán jako **synchronizace od cloudu** a je implementován asynchronní aktivitou *AsyncLoadJSONDoc*, která je volána v aplikaci ze služby *SyncServiceApp*. Jakmile proběhne celý proces, je vytvořen report, který je následně uložen.

Služba je vytvořena při startu aplikace a běží do té doby, než je aplikace ukončena (například použitím zpětného tlačítka z *MainActivity*). Tento ukončovací proces má za následek volání metody životního cyklu aktivity *onDestroy()*, kde je přidána metoda pro zastavení služby *stopService()*.

V případě, že je z aplikace jen „vyskočeno", je životní cyklus ve stavu *onPause()* a Služba je nadále aktivní. Pro proces synchronizace dat je nastaven časový úsek 60s.

Reporty

Pro aplikaci byla vytvořena aktivita *ReportsActivity*, která shromažďuje informace o procesech týkající se synchronizace dat. Podobně jako ListActivity, je tvořena databází, pro jeden záznam shromažďuje informace o id, názvu aktivity z které přichází signál (TAG[30]), času vytvoření a informace o typu procesu synchronizace dat. V aplikaci se objevují tyto popisy:

1. **Sync from cloud OK** – oznamuje, že proběhl proces pro kontrolu změn na cloudu.
2. **Sync to cloud OK. Item saved** – přidání bodu uživatelem.
3. **Sync to cloud. Item deleted** – smazání bodu uživatelem, zde jsou možné i další instance pro smazání. Jak bylo určeno, je možné vzniku dvou konfliktů.
   - 3.1. **Conflict Detected! Item it was deleted on cloud. Deleting from database. Sync OK** – v případě smazání záznamu dříve než akcí uživatele.
   - 3.2. **Conflict Detected! Item DOC_ID not deleted** – v případě, že byl záznam změněn v průběhu cyklu synchronizace od cloudu. Takovýto záznam nebude smazán. Uživatel bude informován i notifikací.

```
public static void Log(Context context, String tag, String log){
  saveToDatabase(context,tag,log);
}

private static void saveToDatabase(Context context, String tag,
                                   String log) {
  SQLiteDatabase db = (new DbHelp(context)).getWritableDatabase();
  ContentValues cv=new ContentValues();
  DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'kk:mm:ss.SSS
                                       'Z'");
  Date date = new Date();
  cv.put(Constants.DB_TAG, tag);
  cv.put(Constants.LOG, log);
```

[30] Obecně používaný vzor pro označení třídy. Z anglického *„tagging"*, tedy štítek.

```
        cv.put(Constants.TIME, df.format(date));
        db.insert(Constants.TABLE_NAME_REPORTS, Constants.TIME, cv);
        db.close();
    }
```

Kód 6. Implementace metody pro uložení logu do databáze

9.1 Optimalizace aplikace

Optimalizace je nedílnou součástí při vývoji softwaru. Jako případ optimalizace, může být bráno ladění aplikace, její vzájemná kompatibilita a také spolehlivost. V dnešní době, kdy existuje velké množství různorodých programovacích jazyků, systémů a různorodých platforem na kterých tyto prvky běží, pro ty jsou specifické, kterými vlastnostmi se vyznačují a kterými se liší od ostatních. Je tedy nasnadě, že se vyskytují problémy.

Optimalizace ve svém pravém slova smyslu, je brána jako proces pro výběr nejlepší varianty. Nejlepší varianta může být určena jako nejefektivnější pro danou akci. Zohledňují se i ostatní vlastnosti. Někdy je pro danou akci nejlepší řešení, které nezatěžuje procesor a je rychlé, naopak někdy je vhodné mít pro danou akci řešení robustní, kde jsou kontrolovány možné chyby. Často se tedy jedná o určitý kompromis, který závisí na typu akce.

Ladění

Jedná se o proces vývojáře, v kterém je potřebné pro dané implementační řešení otestovat jeho spolehlivost. Je možné, že zkušený vývojář, dokáže programovat a uvědomovat si stavy, které mohou nastat, s určitou predikcí. Běžnější je, že dané implementace je pokaždé nutné odladit[31] a tedy otestovat a případně opravit, aby daná implementace splňovala svoji podstatu.

Kontrola vstupů, jedná se o běžný způsob, kdy jsou tázány informace po uživatelovi. Jelikož existuje mnoho různých datových struktur, je na vývojáři, aby uživatelovi definoval, které vstupy jsou pro daný bod akceptovatelné. Vývojář, tento stav musí zabezpečit, neboť poté s těmito daty musí operovat uvnitř programu. Dobrou ukázkou, je ošetření vstupních hodnot pro vytváření bodové vrstvy v aplikaci. Pro zeměpisnou lokaci, je použit souřadný systém WGS84 a je tedy určen obor hodnot pro latitude v rozmezí -90 až 90 a longitude -180 až 180 a to pro datový typ float[32]. Pro JSON, není potřebné uživatele omezit na určitý datový typ. Pro všechny tři proměnné je ještě nutné zabezpečit kontroly *null* výskytů.[33] Viz ukázka pro Kód 7:

[31] Vychází z anglického *debugging*.

[32] V Javě používaný datový typ pro reálná čísla.

[33] V případě, že uživatel danou hodnotu vůbec nevyplní.

```
if (editText.getText().toString().isEmpty()){
      lat = -200;
} else {
      lat = Float.parseFloat(editText.getText().toString());
}

if (lat > 90 || lat < -90){
      Toast.makeText(this, "Latitude wrong scale <-90,90>, item
                  not saved.", Toast.LENGTH_SHORT).show();
      lat = -200;
} else {
      lat = Float.parseFloat(editText.getText().toString());
}

if ((lat == -200) || (lng == -200)){
      //nic
} else {
      Log.i(TAG, json + lat + lng);
      new SyncIniciate(this, json, lat, lng).execute();
}
```

Kód 7. Ukázka ošetření vstupu pro latitude

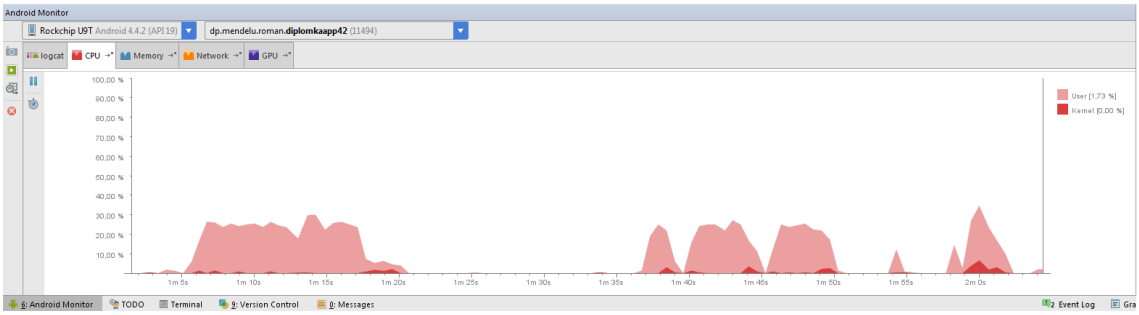
Podobně jsou řešeny všechny možné proměnné, které v aplikaci nabývají hodnot, a byl u nich zjištěn chybový stav.

Optimalizace

Pro optimalizační část aplikace, byl zvolen způsob šetření a analýzy zatěžovaných prvků. Pro získané vizuálních materiálů, je použit vývojový nástroj Android studio, v kterém byla aplikace vyvíjena. Obsahuje nástroj pro analýzu připojeného Android zařízení a nazývá se *Android Monitor*.

CPU, signály zobrazují vytížení procesoru, při zpracování instrukcí (například provedení určitého procesu[34]). V následujícím Obr. 8 jsou zobrazeny 3 akce. Prvotně se provede proces pro synchronizaci od cloudu. Z obrázku lze vyčíst vytížení procesoru na hranici 30% a to po dobu 10-15s, to lze přisoudit ukládání 100 hodnot do databáze. Jako další proces byl spuštěn proces pro zobrazení dat aktivitou *ListActivity*, v průběhu tohoto zobrazeni jsou viditelné „zuby“, které jsou způsobeny listováním. Jako poslední proces je zobrazení detailního záznamu a jeho smazání.

[34] Úkolu, programové metody

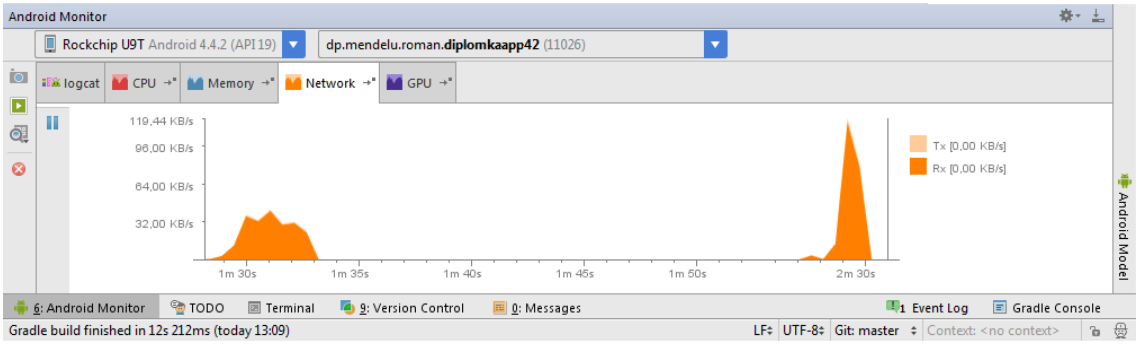


Obr. 8 Vytížení CPU pro 3 různé procesy

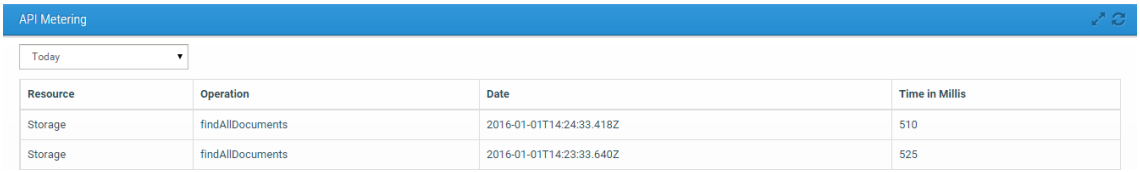
Data, popisují datovou šířku přenosového pásma. Z Obr. 9 lze vidět, že jsou zaznamenány 2 signály. Jedná se o zachycení procesu pro kontrolu nových dat z cloudu zajišťující synchronizaci dat. V tomto případě je posíláno 100 záznamů a to pro interval synchronizace nastavený v aplikaci na 60s, což znamená 60hitů/hodina. Jsou použita reálná komplexní data JSON (cca 2kB/záznam). Jak lze vidět z Obr. 10, tyto dva záznamy byly zachyceny i v cloudové službě. Přes webové rozhraní je možné zjistit, že zpracování těchto záznamů probíhá s cca odezvou 500ms (0,5s) a jednalo se o metodu *findAllDocuments*.

Jelikož aplikace vykazuje konstantní velikosti, bylo provedeno měření pouze o 10 hodnotách, data viz příloha C. Stanovení výsledné velikosti dat posílaných a přijímaných aplikací každou minutu je:

- Download – 209kB/min.
- Upload – 8,3kB/min.



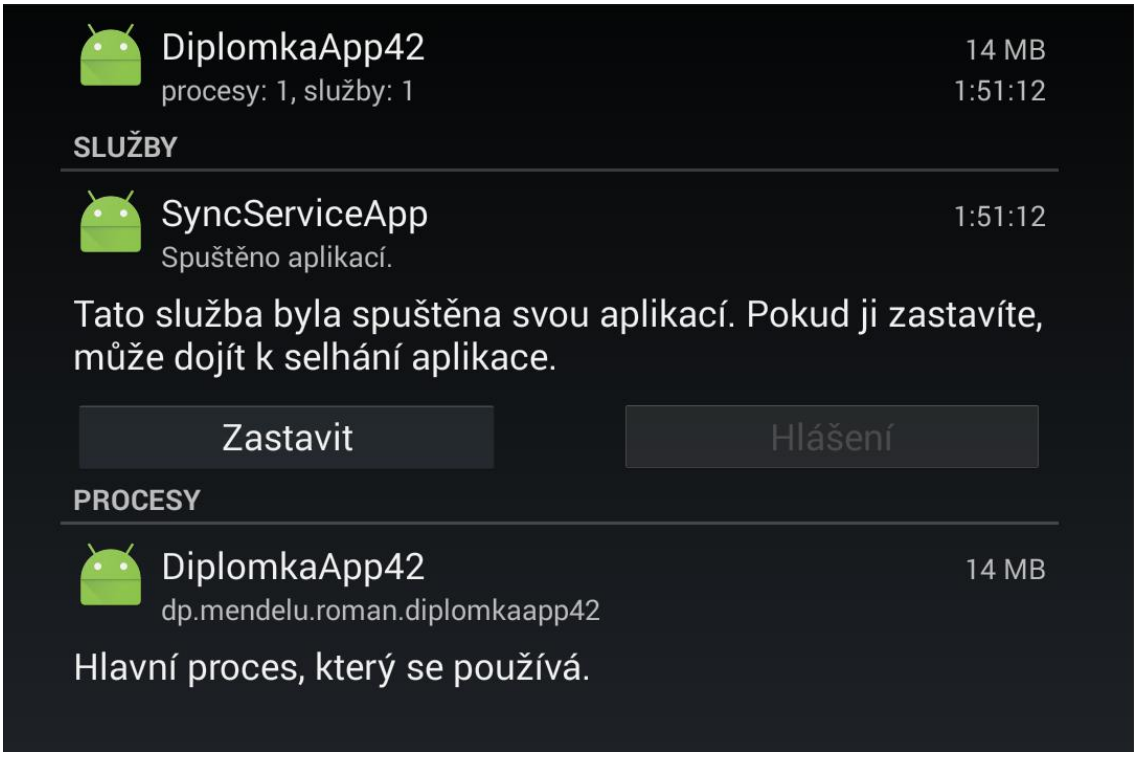
Obr. 9 Ukázka signálu datového přenosu pro proces synchronizace „Sync from cloud“



Obr. 10 Ukázka záznamu 2 signálů pro API z webové služby

Datové rychlosti pro požadavek pouze jednoho JSON dokumentu, jsou o řád níže a pohybují do velikosti 10kB.

Paměť, aplikace při běhu, zabírá cca 15MB z paměti RAM daného zařízení.



Obr. 11 Ukázka spuštěné aplikace v nastavení zařízení

Řešení konfliktů

Než byly detekované konflikty vyřešeny, prošly implementačními pokusy, dokud nesplňovaly daný návrh pro řešení. Řešení pro konflikty se nachází ve třídě pro mazaný bod směrem ke cloudu. Daný algoritmus je vyjádřen následně:

```
StorageService storageService = App42API.buildStorageService();
try {
  String checkConflicts = storageService.findDocumentById
  (Constants.CLOUD_DB_NAME, Constants.COLLECTION_NAME,
        mCloud_delete_doc).getJsonDocList().get(i).getUpdatedAt();
  if (checkConflicts.equals(mCloud_updated_doc)) {
        //not changed -> delete
        storageService.deleteDocumentById(Constants.CLOUD_DB_NAME,
              Constants.COLLECTION_NAME, mCloud_delete_doc);
              deleteByDoc(mCloud_delete_doc);
  }     else {
        //changed -> conflict
        ReportsActivity.Log(mContext, TAG, "Conflict Detected! Item
                          "+ mCloud_delete_doc + " not deleted");
  }catch (Exception ex) {
```

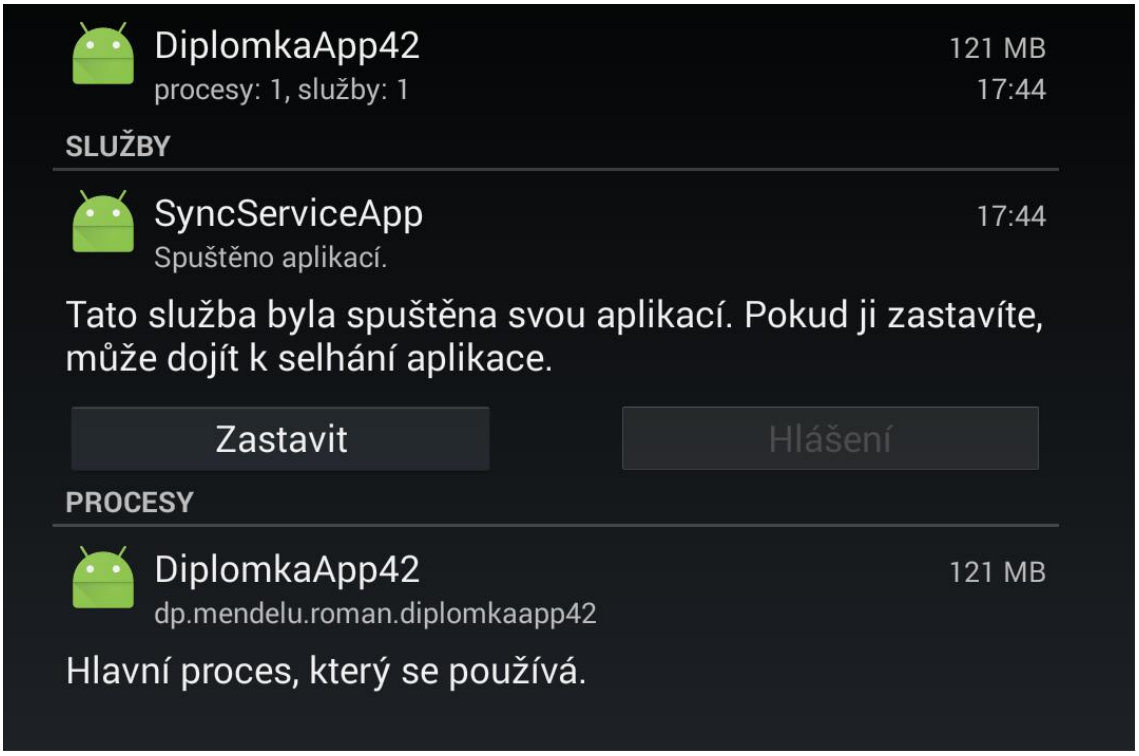
```
        ReportsActivity.Log(mContext, TAG, "Conflict Detected! Item it
              was deleted in Cloud, deleting from Database. Sync OK.");
    }
```

Kód 8. Řešení konfliktu pro synchronizaci dat při procesu smazání

Programové chyby

V aplikaci bylo zjištěno mnoho chyb. Za zmínku stojí dvě.

- Jako příhodný je zde právě Kód 8, kde se porovnávají časové razítka uložená v datovém typu *string*. Prvotní pokusy pomocí přirovnání s operátorem „==“ dávaly při stejných hodnotách hodnotu *false* a daný algoritmus tedy fungoval nekorektně. Byly provedeny kontroly, výpisy a konverze, dokud nebyla použita metoda pro porovnání dvou řetězců pomocí *eaquals()*.
- Aplikace při připojení k počítači ve fázi ladění, vykazovala neustále rostoucí znak při alokaci paměti. Tuto alokaci způsoboval proces kontroly změn na cloudu. Při odpojení a opětovném připojení k ladícímu nástroji, neměla alokace nadále rostoucí tendenci. Také v případě používání aplikace na zařízení, tento problém nevykazovala.



Obr. 12 Bug s alokací pamětí

9.2 Hodnocení

Z obecného návrhu řešení (metody) pro synchronizaci dat založené na synchronizaci strukturovaných JSON dat, byl vytvořen konkrétní návrh řešení a to po-

mocí technologie cloudu. Cloud zde vystupuje jako služba a je řešena pomocí API NoSQL Storage App42 od společnosti ShepHertz, kde je pro správu dat, dostupná databáze ve formě webové aplikace.

V Android aplikaci jsou naprogramovány procesy, které zajišťují uživatelovi aplikace synchronizaci dat. Tento proces je nastaven v aplikaci tak, aby uživatel měl každou minutu data synchronizována s cloudovou službou. Pro aplikaci je vytvořeno uživatelské rozhraní, aby byla daná implementace použitelná a bylo v ní možné otestovat zmiňované prvky.

Dané API během implementace a testování, vykazovalo určité nedostatky. Jelikož se jedná o NoSQL službu, byla zde opomenuta možnost metody, která by umožnila vývojáři pracovat s daty ve formě pouze celých sloupců. Tento nedostatek, poté vedl k práci pouze s kompletní cloudovou databází, což vedlo k větším přenosovým náročnostem aplikace, než bylo předpokládáno v návrhu.

Druhým zjištěným nedostatkem je, že API umožňuje při komunikaci klientské aplikace a cloudové služby přenášet pouze 100 záznamů ve formě pole JSON dokumentů. Při hlubším zkoumání programových kódů API, nebylo nalezeno omezení, které by mělo tento jev za vinu. Z daného zjištění, lze usoudit, že dané omezení je na straně cloudu. Jelikož daná cloudová služba je ve verzi TRIAL, může být dané omezení způsobeno právě tímto jevem. Bohužel i po prostudování stránek a fóra pro dané API, nebyla nalezena jasná odpověď. Jako poslední možnost, je ještě, že cloudová služba neumožňuje posílat větší objem záznamů.

Aplikace je nyní ve stavu *„funkční a plně synchronní“* a to včetně řešených konfliktů a reportů ukládající informace o procesech synchronizace dat uživatele daného zařízení.

9.3 Využití

Právě nedostatky, kterými dané API disponuje, je ovlivněno jeho možné použití/využití. Jako velký nedostatek, je zde omezení implementační, které umožňuje přenos pro maximálně 100 JSON dokumentů. Na to je vázán problém, že lze zabezpečit pouze synchronizaci pro 100 záznamů (v případě, že nebudeme uvažovat o komplexnějších algoritmech pro vícenásobnou kontrolu).

Nicméně navrhnutá aplikace lze využít pro různé instance komerčních využití. Jako možné nápady se jeví:

- Na vytvořenou aplikaci navázat grafické uživatelské rozhraní ve formě map. Tyto mapy můžou být řešeny opět pomocí API (například Google Maps, Mapy.cz). Pro proces synchronizace použita JSON data o bodových vrstvách, které by pro každou kolekci (město) umožňovalo vytvoření 100 záznamů o památkách/restaurací/kavárnách.
- Zakázka od firmy na aplikaci, která by umožňovala zaměstnancům v terénu, zaznamenávat geodata. Pro každý výjezd, by byla vytvořena vlastní kolekce na cloudu.

Samozřejmě, pokud by byla aplikace rozšířena o procesy navíc, je potřebné pro tyto procesy doplnit návaznosti s danou aplikací, ale v zásadě, je možné aplikaci využít.

Na závěr, bych podotknul, že daná aplikační logika se spíše odehrává za oponou uživatelského prostředí. A je také možné, že v některých případech, nastane stav, který nebyl ošetřen, protože nebyl odhalen, nebo nebyl pro danou práci jako prioritní. Uživatel by měl aplikaci využívat v dobré vůli a ne se snažit o hledání chyb. V případě, že by některé chyby byly nalezeny, je dobrým zvykem, podat vývojáři zpětnou vazbu a to ve formě, kterou sám uzná za vhodnou.

10 Diskuze

Navrhnutá metoda je vyjádřena jako obecný návrh řešení, které se vztahuje na komunikaci zajišťující synchronizaci dat mezi více Android zařízeními zpracovávající geodata typu JSON.

V metodě jsou určeny postupy pro kategorie, které jsou důležité při implementaci konkrétního použití. Tyto kategorie popisují vhodné možnosti, kterými se řídit při návrhu řešení pro Android aplikaci. Jsou zde popsány způsoby zajištění synchronizace dat a určují, které náležitosti by měly být splněny. Je popsána problematika konfliktů včetně detekce konfliktů, při použití této metody, by měly být minimálně tyto konflikty řešené.

V metodě je vytvořen matematický podklad pro určení velikosti šířky pásma při komunikaci dvou stran.

Technologické dostupné možnosti jsou určeny v samostatné kapitole. V metodě je ponechán prostor, pro zvolení technologie, která bude zajišťovat synchronizaci dat pro mezi Android zařízeními. Prvek spojující všechny uživatele, pro vytvoření komunikace mezi všemi zařízeními, v metodě není pevně stanoven. Tato volba je na vývojáři. Z této práce lze odvodit, že by jako centralizovaný prvek, mělo být použito cloudové řešení.

Navržená metoda, může být poté použita pro vypracování konkrétní Android aplikace. Pro přesný popis je potřeba znát podmínky, které musí aplikace zajistit. Je nutné definovat odhadovaný počet uživatelů, odhadovanou velikost databáze a záznamů. Náročnost na velikost šířky pásma pro centralizovanou složku, která zajišťuje komunikaci a následně i proces pro synchronizaci dat.

Právě, až po definování těchto podmínek, lze vytvořit konkrétní návrh řešení, protože je velice důležité vědět, jestli je Android aplikace vytvářena pro 100 uživatelů nebo pro 10 000 uživatelů. Jestli databáze bude obsahovat 100 záznamů nebo 10 000 záznamů. A právě pro tyto důvody, je obecné řešení (metoda) důležitým bodem, před vytvořením konkrétního návrhu.

11 Závěr

Cílem této diplomové práce bylo navržení metody, která zajistí synchronizaci dat mezi Android zařízeními. Z literárního přehledu je patrné, že technologie, která zprostředkovává komunikaci mezi zařízeními, prošla vývojem a nyní je směr jasně definován cloudovými možnostmi.

Z určených scénářů použití byl jeden ze scénářů zvolen. Jednalo se o praktickou volbu, tak aby z něj navrhnutá metoda pro dané řešení měla význam. Tento scénář, který byl specifický především ve zpracování JSON záznamů. Pro scénář byl zpracován rozbor, díky čemu byly určeny body, které jsou nezbytné k navrhnutí metody.

Tato metoda je určena jako obecný návrh pro synchronizaci dat mezi Android zařízeními s daty typu JSON. Tato metoda poté slouží jako podklad pro vytvoření instance pro konkrétní návrh řešení, pomocí kterého je možné vytvořit Android aplikaci, která splňuje požadavky navrhnuté metody.

Stěžejním bodem pro vytvoření konkrétního návrhu řešení, je zvolení technologie, pomocí které bude zajištěna synchronizace dat pro uživatele Android aplikace. Z dostupných možností byla zvolena technologie ve formě API NoSQL Storage, pomocí které je možné vybudovat komunikaci mezi klientem Android aplikace a cloudovou službou ve formě databáze jako služba.

Jakmile byly navrhnuty veškeré potřebné body, byl zahájen implementační proces, který měl za cíl podle navrhnutého řešení na základě vytvořené metody vytvořit Android aplikaci. Tato aplikace byla v implementační části této práce vytvořena a popsána. Daná aplikace je funkční a plně synchronní. Kterýkoliv uživatel, který by si ji právě teď nainstaloval, bude mít do jedné minuty data synchronizována s cloudovou databází a tedy i s ostatními uživateli. Daná aplikace splňuje požadované body obecného návrhu.

Android aplikace, prošla sérií testů a analýz, aby dosáhla spolehlivosti. V optimalizační části této práce jsou popsány postupy při vylepšování aplikace a postupy řešení detekovaných konfliktů. Také jsou popsány programové chyby, které byly v průběhu optimalizace zjištěny.

Navrhnutá metoda je důležitá pro svůj obecný podklad. Až určením konkrétních potřeb a technologií, se vytváří ucelenější přehled pro implementační část. A právě pro tyto důvody, považuji cíl diplomové práce za splněný a vytvořenou metodu za přínosnou.

12 Literatura

[1] KOŠÍČEK, M. *Podpora automatizovaného multiplatformního sdílení a synchronizace souborů*. Diplomová práce. Mendelova univerzita v Brně, 2014. 71 s.

[2] ABZ *slovník českých synonym* [online]. 2015 [cit. 2015-12-03]. Dostupné z: http://www.slovnik-synonym.cz/

[3] GOOGLE PATENTS. *Cloud-based device synchronization,* EVANS, E.Z. AND CABRERA,L.F. AND SIEGEL,H.B. AND DESANTIS, P.N., https://www.google.com/patents/US20120079095, 2012,Google Patents, US Patent App. 12/890,314

[4] HANSMANN, UWE; METTÄLÄ, RIKU; PURAKAYASTHA, APRATIM; THOMSON, PETER. *SyncML: Synchronizing and Managing Your Mobile Data*. 1st edition. Pearson Education, 2002. 320 s. ISBN978-0-13-009369-1.

[5] GOOGLE PATENTS. *Client-server model for synchronization of files,* BROWN, D.K. AND ROLANDER, T.A. AND SILBERSTEIN, R.D. AND WEIN, J. https://www.google.com/patents/WO2002075539A2?cl=en 2002, Google Patents, WO Patent App. PCT/US2002/007,781

[6] GOOGLE PATENTS. *Peer-to-peer data synchronization architecture,* PRASAD, S. AND SAGAR, R. AND JUDGE, F. AND KALRA, G.S., https://www.google.com/patents/US7680067 2010, Google Patents, US Patent 7,680,067

[7] WI-FI. *Wi-Fi Direct | Wi-Fi Alliance: Portable Wi-Fi® that goes with you anywhere* [online]. [cit. 2015-10-26]. Dostupné z: http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[8] DEVELOPER ANDROID. *Wi-Fi Peer-to-Peer. Android developers* [online]. 2015 [cit. 2015-10-26]. Dostupné z: http://developer.android.com /guide/topics/connectivity/wifip2p.html

[9] PARSE [online]. 2015 [cit. 2015-10-26]. Dostupné z: https://www.parse.com/

[10] DEVELOPER ANDROID. *Transferring Data Using Sync Adapters. Android Developers*[online]. 2015 [cit. 2015-10-26]. Dostupné z: http://developer.android.com/training/sync-adapters/index.html

[11] GOOGLE PLAY [online]. 2015 [cit. 2015-10-27]. Dostupné z: https://play.google.com/store

[12] SVĚTANDROIDA. *Svět Androida doporučuje: vaše data kdykoliv a kdekoliv - 7x cloudové disky - Svět Androida* [online]. 2013 [cit. 2015-10-27]. Dostupné z: http://www.svetandroida.cz/svetandroida-doporucuje-vase-data-kdykoliv-a-kdekoliv-7x-cloudove-disky-201307

[13] SVĚTANDROIDA. *Synchronizace souborů mezi Androidem a počítačem bez cloudu: BitTorrent Sync - Svět Androida* [online]. 2013 [cit. 2015-10-27]. Dostupné z: http://www.svetandroida.cz/synchronizace-souboru-mezi-androidem-a-pocitacem-bez-cloudu-bittorrent-sync-201308

[14] MILLER, MICHAEL. *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate* Online, 1.vydání,Que Publishing, 2009, 284 s, ISBN-13: 978-0-7897-3803-5

[15] AMAZON. *Types of Cloud Computing. Amazon Web Services (AWS) - Cloud Computing Services* [online]. Seattle: Amazon Web Services, ©2015 [cit. 2015-11-19]. Dostupné z: https://aws.amazon.com/types-of-cloud-computing/

[16] IBM. *Infrastructure as a service (IaaS). IBM Cloud Computing* [online]. India: IBM Corporation, ©2015 [cit. 2015-11-19]. Dostupné z: http://www.ibm.com/cloud-computing/in/en/what-is-iaas.html

[17] DEVELOPER ANDROID. *Resolving Cloud Save Conflicts. Android Developers* [online]. [cit. 2015-11-26]. Dostupné z: http://developer.android.com/training/ cloudsave/conflict-res.html

[18] HŘEBABECKÝ, JAN. *Projekty pro sdílení dat na internetu typu Rapidshare se zaměřením na konkrétní vybraný projekt a jeho rozbor*. Praha, 2011. Diplomová práce. Vysoká škola ekonomická v Praze.

[19] KUČEROVÁ, HELENA. *Geografická informace. In: KTD: Česká terminologická databáze knihovnictví a informační vědy (TD-KIV)* [online]. Praha: Národní knihovna ČR, 2003- [cit. 2015-11-26]. Dostupné z:http://aleph.nkp.cz/F/?func=direct&doc_number=000000397&local_base=KTD.

[20] MURPHY, MARK L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Computer Press, 2011, s. 23 ISBN 978-80-251-3194-7.

[21] ROUSE, MARGARET A ROBERT SHELDON. *Database as a Service (DBaaS)*. WhatIS [online]. 2014 [cit. 2015-11-27]. Dostupné z: http://whatis.techtarget.com/definition/Database-as-a-Service-DBaaS

[22] WIKIPEDIA. *JSON* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-11-27]. Dostupné z: https://en.wikipedia.org/wiki/JSON

[23] GOOGLE CLOUD. *Google Cloud Computing: Hosting Services & APIs* [online]. Google, 2015 [cit. 2015-11-30]. Dostupné z: https://cloud.google.com/

[24] SHEPHERTZ. *Manage JSON documents in NoSQL database on the Cloud: App42 Docs*. Backend as a Service [online]. India, 2015 [cit. 2015-11-30]. Dostupné z: http://api.shephertz.com/app42-docs/nosql-storage-service/#insert-json-document

Přílohy

Brno 2016

A Definice základních pojmů

- **Data** – celek reprezentovaný jako jeden nebo více souborů
- **Sdílení dat** – činnost, při které je pomocí hardwarových a softwarových prostředků zajištěno jejich kopírování nebo přesouvání do dalších zařízení
- **Peer-to-peer** – (doslova rovný s rovným), P2P nebo klient-klient je označení typu počítačových sítí, ve které spolu komunikují přímo jednotliví klienti (uživatelé). Opakem je klient-server.
- **Klient-server** – síťová architektura, kde komunikace mezi klienty probíhá pomocí prostředníka v podobě serveru.
- **Server** – v informatice se obecně jedná o počítač, který poskytuje služby. Nejčastěji interpretován v podobě počítačového programu, který zaručí danou funkčnost.
- **Počítačová síť** – je souhrnné označení pro technické prostředky, které realizují spojení a výměnu informací mezi počítači ať už na intranetu (domácí privátní sítě) nebo internetu (celosvětové veřejné sítě).
- **Cloudová služba** – jedná se o centralizovanou službu, která musí obsahovat výpočetní jednotku, úložiště dat a další zdroje, které jsou potřebné k sdílení mezi dvěma (či více) zařízeními, které mají přístup z alespoň jedné komunikační sítě k této službě. Tato služba musí obsluhovat žádosti pro přijmutí nebo odeslání dat od různých zařízení a případně tyto data zpracovat (transformace, mapováni, konverze), tak aby různá zařízení mohla komunikovat prostřednictvím této služby.
- **Synchronizace dat –** proces, který zajišťuje data ve stejné podobě.
- **Virtualizace** – možnost spustit v jednom fyzickém serveru několik operačních systémů ve formě virtuálních strojů, z nichž každý má přístup k výpočetním zdrojům základního serveru
- **Chytré zařízení** – menší zařízení, které umožňují uživatelům pokročilé akce podobající se akcím na počítačích.
- **API** – aplikační programové rozhraní, slouží k propojování složitějších programových částí mezi sebou.

B Ukázka reálných dat JSON

```
{
  "sync_id": "1234",
  "layers": // poskladano ze dotazu testu #8:
/layer/clienttestuser/stromy-v-brne
  [
    {
      "type": "Layer",
      "name": "stromy-v-brne",
      "scheme": "clienttestuser",
      "hash": "2014-09-22 11:40:07.355724",
      "properties":
      {
        "title": "Stromy v Brně",
        "description": "Stromy v campusu MENDELU",
        "kind": "point",
        "public": true,
        "created": "2013-12-20 13:00:00",
        "last_edit": "2013-12-20 13:00:00"
      },
      "attributes":
      [
        {
          "title": "Vyska",
          "name": "height",
          "kind": "double precision",
          "hash": "hash1"
        },
        {
          "title": "Sirka",
          "name": "width",
          "kind": "double precision",
          "hash": "hash2"
        }
      ],
      "features": // doplneno z #18: /layer/clienttestuser/stromy-v-
brne-123456/feature
      [
        {
          "type": "Feature",
          "id": 1,
          "hash": "hashFeature",
          "properties":
          {
            "created": "2013-12-20 13:00:00",
            "last_edit": "2013-12-20 13:00:00",
            "precision": 23.8
          },
          "attributes":
          [
            {
              "name": "height",
```

```
              "value": "15.5",
              "hash": "hash1"
            },
            {
              "name": "width",
              "value": "None",
              "hash": "hash2"
            }
          ],
          "geometry":
          {
            "type": "Point",
            "coordinates": [10.15, 4.2]
          }
        },
        {
          "type": "Feature",
          "id": 2,
          "hash": "hashFeature",
          "properties":
          {
            "created": "2013-12-20 13:00:00",
            "last_edit": "2013-12-20 13:00:00",
            "precision": 23.8
          },
          "attributes":
          [
            {
              "name": "height",
              "value": "15.5",
              "hash": "hash1"
            },
            {
              "name": "width",
              "value": "2.1",
              "hash": "hash2"
            }
          ],
          "geometry":
          {
            "type": "Point",
            "coordinates": [100.1, 200.2]
          }
        }
      ]
    }
    // podobne pro kazdou dalsi vrstvu
  ],
  "projects": // seznam projektu poskladan z #3
/project/clienttestuser/mereni-v-brne
  [
    {
      "type": "Project",
      "name": "mereni-v-brne",
      "scheme": "clienttestuser",
```

```
      "hash": "2014-09-22 11:40:07.355724",
      "properties":
      {
        "title": "Měření v Brně",
        "public": true,
        "description": "This project contains ...",
        "last_latitude": 1.1,
        "last_longitude": 1.2,
        "last_zoom_level": 1.3
      },
      "layers":
      [
        {
          "type": "LayerInProject",
          "hash": "2014-09-22 11:40:07.355724",
          "name": "stromy-v-brne-1",
          "scheme": "clienttestuser",
          "properties":
          {
            "color": "FFFFFF",
            "transparency": 50,
            "zoom_threshold": 20,
            "visibility": true,
            "order": 1,
            "symbology": "nazev"
          }
        },
        {
          "type": "LayerInProject",
          "hash": "2014-09-22 11:40:07.355724",
          "name": "stromy-v-brne-2",
          "scheme": "clienttestuser",
          "properties":
          {
            "color": "FFFFFF",
            "transparency": 50,
            "zoom_threshold": 20,
            "visibility": true,
            "order": 1,
            "symbology": "nazev"
          }
        }
      ]
    }
  ]
}
```

C Data měření

- Upload;Download
- 8,10;209,6
- 9,10;209,5
- 8,00;209,6
- 8,80;209,6
- 8,10;209,5
- 8,20;206,6
- 7,80;206,5
- 8,00;209,6
- 8,00;209,4
- 8,70;209,5

D DVD

K diplomové práci je přiloženo DVD se soubory. Jedná se především o Android projekt a programové kódy v něm obsaženy.